Lab4

1. Linear regression source code

import pandas as pd

import numpy as np

from sklearn.preprocessing import LabelEncoder,StandardScaler

from sklearn.linear\_model import LinearRegression

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

from sklearn.metrics import mean\_squared\_error,r2\_score

from math import sqrt

# read csv

df = pd.read\_csv("data/housing.csv")

print('california housing data')

print(df)

# data scaling & encoding

print(df.isnull().sum())

print()

df.total\_bedrooms = df.total\_bedrooms.fillna(df.total\_bedrooms.mean())

print(df.isnull().sum())

print()

label\_encode = LabelEncoder()

df['ocean\_proximity'] = label\_encode.fit\_transform(df['ocean\_proximity'])

columns = df.columns

scaler = StandardScaler()

scaled\_df = pd.DataFrame(scaler.fit\_transform(df), columns=columns)

print('scaled california housing data')

print(scaled\_df)

# create scaled dataframe & target dataframe

X = scaled\_df[['longitude', 'latitude', 'housing\_median\_age', 'total\_rooms', 'total\_bedrooms', 'population', 'households', 'median\_income', 'ocean\_proximity']]

y = scaled\_df['median\_house\_value']

for i in range(0, 2):

    if(i==0):

        # split random train & test data

        X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, train\_size=0.8, test\_size=0.2, random\_state=1, stratify=None)

        print('train & test data that train size is 0.8')

        print(X\_train, y\_train, X\_test, y\_test)

output screen capture

# linear regression

        linear\_regression = LinearRegression()

        linear\_regression.fit(X\_train, y\_train)

        LinearRegression(copy\_X = True, fit\_intercept=True, n\_jobs=None, normalize=False)

        y\_predict = linear\_regression.predict(X\_test)

        print(sqrt(mean\_squared\_error(y\_test, y\_predict)))

        print(r2\_score(y\_test, y\_predict))

        print("--------------------------------------------------------")

    elif(i==1):

        # split random train & test data

        X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, train\_size=0.6, test\_size=0.4, random\_state=0, stratify=None)

        print('train & test data that train size is 0.6')

        print(X\_train, y\_train, X\_test, y\_test)

        # linear regression

        linear\_regression = LinearRegression()

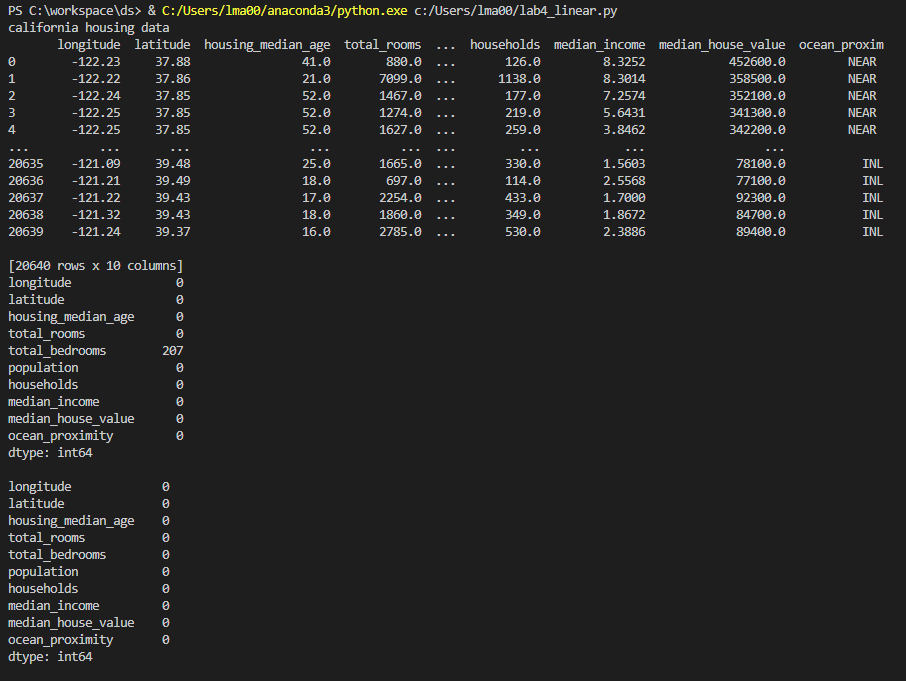
        linear\_regression.fit(X\_train, y\_train)

        LinearRegression(copy\_X = True, fit\_intercept=True, n\_jobs=None, normalize=False)

        y\_predict = linear\_regression.predict(X\_test)

        print(sqrt(mean\_squared\_error(y\_test, y\_predict)))

        print(r2\_score(y\_test, y\_predict))



텍스트, 명판이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

1. Decision tree source code

import numpy as np

import pandas as pd

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

from sklearn.preprocessing import StandardScaler

from sklearn import tree

from sklearn.metrics import accuracy\_score

# read csv

# separate data by ';'

df = pd.read\_csv('data/winequality-red.csv',index\_col = None, sep=';')

print('red wine quality data')

print(df)

# create target dataset

X = df.drop(['quality'], axis=1)

y = df[['quality']]

for i in range(0, 3):

    if i==0:

        # split random train & test data

        X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, train\_size=0.9, test\_size=0.1, random\_state=0)

        print('train & test data that train size is 0.9')

        print(X\_train, X\_test, y\_train, y\_test)

        # standardscaling

        sc = StandardScaler()

        sc.fit(X\_train)

        train\_std = sc.transform(X\_train)

        test\_std = sc.transform(X\_test)

        # decision tree

        decision\_tree = tree.DecisionTreeClassifier(criterion='entropy', random\_state=0)

        decision\_tree.fit(X\_train, y\_train)

        # predict & evaluate

        y\_pred\_tr = decision\_tree.predict(X\_test)

        print(accuracy\_score(y\_test, y\_pred\_tr))

    elif i==1:

        # split random train & test data

        X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, train\_size=0.8, test\_size=0.2, random\_state=0)

        print('train & test data that train size is 0.8')

        print(X\_train, X\_test, y\_train, y\_test)

output screen capture

# standardscaling

        sc = StandardScaler()

        sc.fit(X\_train)

        train\_std = sc.transform(X\_train)

        test\_std = sc.transform(X\_test)

        # decision tree

        decision\_tree = tree.DecisionTreeClassifier(criterion='entropy', random\_state=0)

        decision\_tree.fit(X\_train, y\_train)

        # predict & evaluate

        y\_pred\_tr = decision\_tree.predict(X\_test)

        print(accuracy\_score(y\_test, y\_pred\_tr))

    elif i==2:

        # split random train & test data

        X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, train\_size=0.7, test\_size=0.3, random\_state=0)

        print('train & test data that train size is 0.7')

        print(X\_train, X\_test, y\_train, y\_test)

        # standardscaling

        sc = StandardScaler()

        sc.fit(X\_train)

        train\_std = sc.transform(X\_train)

        test\_std = sc.transform(X\_test)

        # decision tree

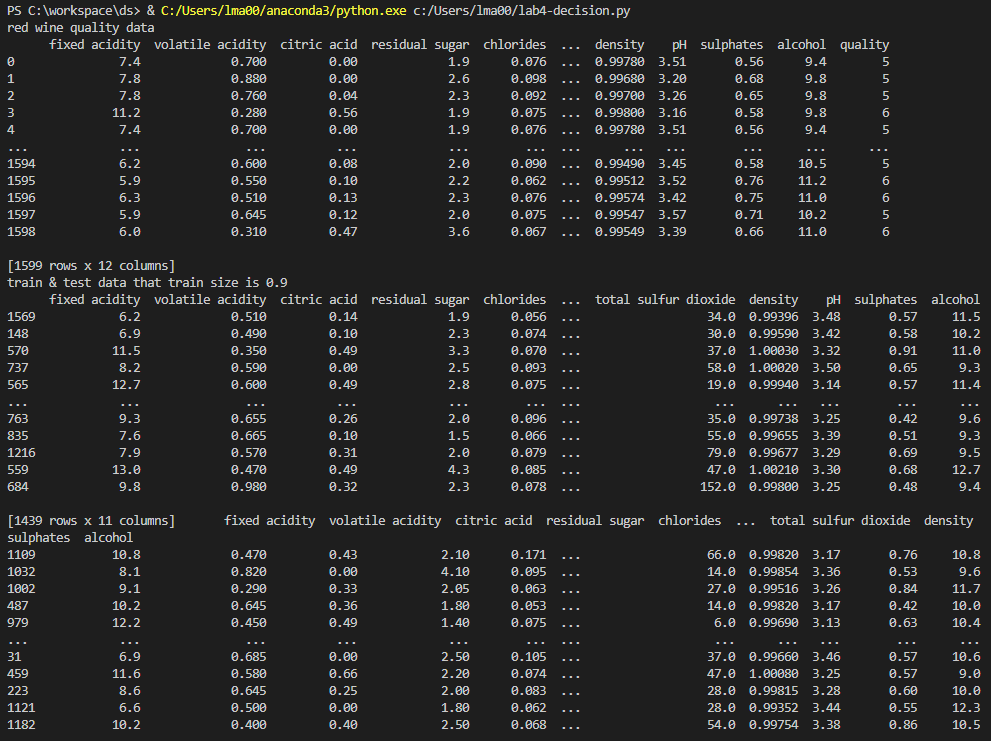
        decision\_tree = tree.DecisionTreeClassifier(criterion='entropy', random\_state=0)

        decision\_tree.fit(X\_train, y\_train)

        # predict & evaluate

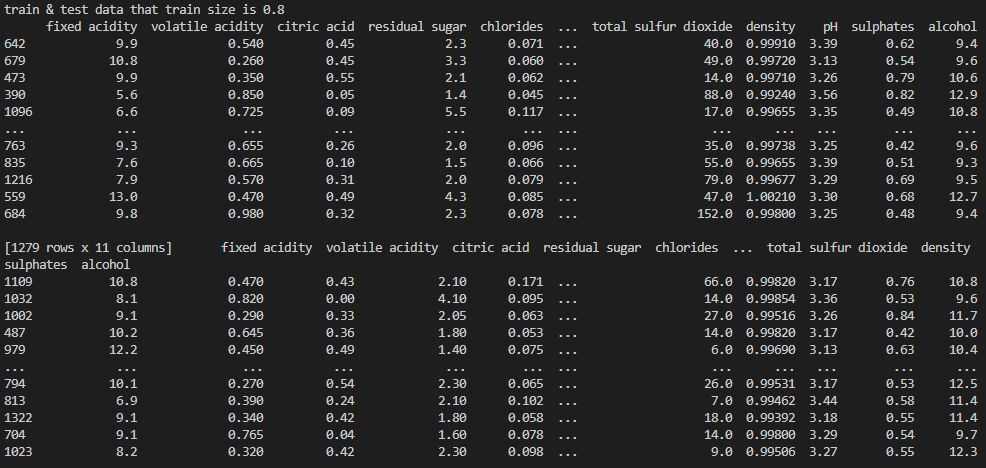
        y\_pred\_tr = decision\_tree.predict(X\_test)

        print(accuracy\_score(y\_test, y\_pred\_tr))



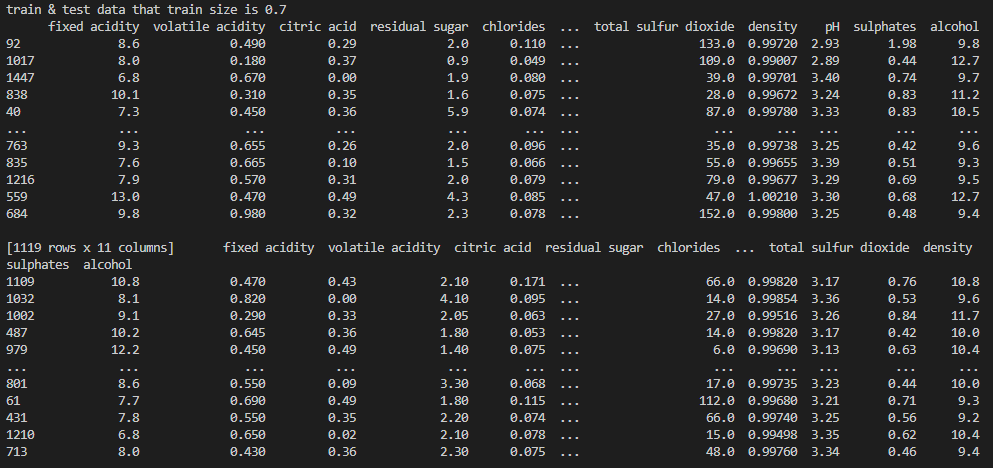
텍스트이(가) 표시된 사진

자동 생성된 설명



텍스트이(가) 표시된 사진

자동 생성된 설명



텍스트이(가) 표시된 사진

자동 생성된 설명

1. Knn

import numpy as np

import pandas as pd

from statistics import mean

from sklearn.model\_selection import cross\_val\_score

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import GridSearchCV

from sklearn.model\_selection import RandomizedSearchCV

# read csv

data = pd.read\_csv('data/mnist\_test.csv')

# create target dataset

X = data.drop(columns=['label'])

y = data['label'].values

k\_arr = [3, 5]

for i in range(0, 2):

        knn = KNeighborsClassifier(n\_neighbors=k\_arr[i])

        score = cross\_val\_score(knn, X, y, cv=5, scoring='accuracy')

        print('mean score')

        print(mean(score))

        knn\_2 = KNeighborsClassifier()

        grid = {'n\_neighbors': np.arange(1, 24)}

        knn\_gridsearch = GridSearchCV(knn\_2, grid, cv=5)

        knn\_gridsearch.fit(X, y)

        print(knn\_gridsearch.best\_params\_)

        print(knn\_gridsearch.best\_score\_)

        knn\_2 = KNeighborsClassifier()

        knn\_random = RandomizedSearchCV(knn\_2, grid, cv=5)

        knn\_random.fit(X, y)

        print(knn\_random.best\_params\_)

        print(knn\_random.best\_score\_)

Output screen capture

텍스트이(가) 표시된 사진

자동 생성된 설명