**Week 12 class: Lab4**

**201835533 조현식**

**# Problem1 - Linear Regression**

**<Source code>**

import numpy as np

import pandas as pd

from sklearn import linear\_model

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

from sklearn import preprocessing

import matplotlib.pyplot as plt

# Get data

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

# Data Exploration

print(df.info(), end="\n\n")

# Drop 'ocean\_proximity'

df = df.drop(columns=['ocean\_proximity'])

# Fill NaN with 0

df = df.fillna(0)

# Data Exploration

print(df.info(), end="\n\n")

# Data Scaling

scaler = preprocessing.StandardScaler()

scaled\_df = scaler.fit\_transform(df)

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

# Split the dataset (X,y)

X = scaled\_df.drop(columns=['median\_house\_value']).values

y = df['median\_house\_value'].values

# Split the dataset (4/5(3/5) for training and 1/5(2/5) for testing)

for a,b in zip([0.8,0.6],[0.2,0.4]):

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,train\_size=a,test\_size=b,shuffle=True)

# Linear regression -> fit & predict

reg = linear\_model.LinearRegression()

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

y\_pred=reg.predict(X\_test)

# Evaluation

print("<Result of tested model(training: ",a,", testing: ",b,")>", sep="")

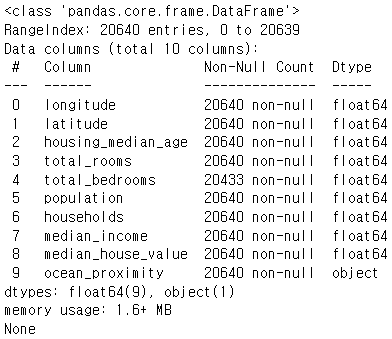
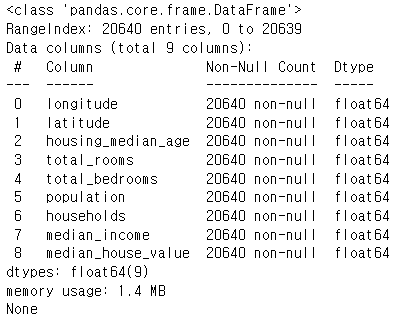
print(pd.DataFrame(y\_pred))

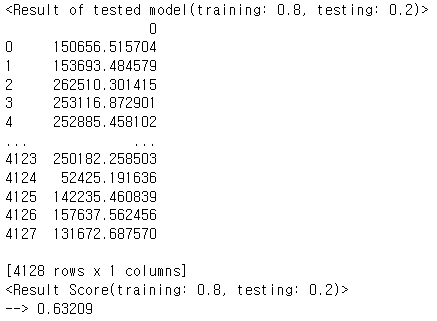
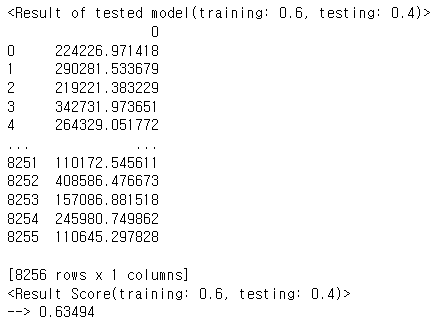
print("<Result Score(training: ",a,", testing: ",b,")>", sep="")

print("-->",np.round(reg.score(X\_test, y\_test),5))

print()

**<Output Screen>**

**# Problem2 - Decision Tree**

**<Source code>**

import numpy as np

import pandas as pd

from sklearn import preprocessing

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

from sklearn.metrics import accuracy\_score

from sklearn import tree

import matplotlib.pyplot as plt

# Get data

df = pd.read\_csv("winequality-red.csv",sep=";")

# Data exploration

print(df.info(),end="\n\n")

# Data Scaling

scaler = preprocessing.StandardScaler()

scaled\_df = scaler.fit\_transform(df)

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

# Split the dataset (X,y)

X = scaled\_df.drop(columns=['quality']).values

y = df['quality'].values

# Split the dataset 9/10(8/10,7/10) for training and 1/10(2/10,3/10) for testing)

for a,b in zip([0.9,0.8,0.7],[0.1,0.2,0.3]):

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,train\_size=a,test\_size=b,shuffle=True)

# Decision tree -> fit & predict

tr = tree.DecisionTreeClassifier(criterion='entropy')

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

y\_pred\_tr = tr.predict(X\_test)

# Plotting

plt.figure()

tree.plot\_tree(tr,filled=True)

plt.title("Decision Tree (training: "+str(a)+", testing: "+str(b)+")")

plt.show()

# Evaluation

print("<Result of test model(training: ",a,", testing: ",b,")>", sep="")

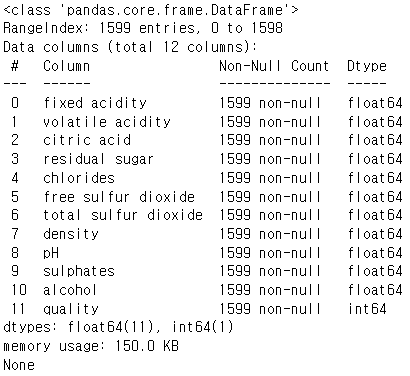
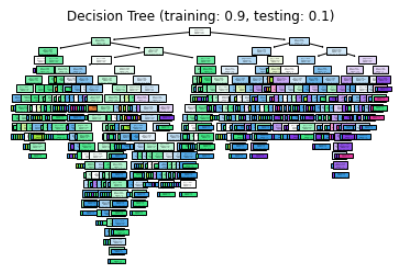
print(pd.DataFrame(y\_pred\_tr))

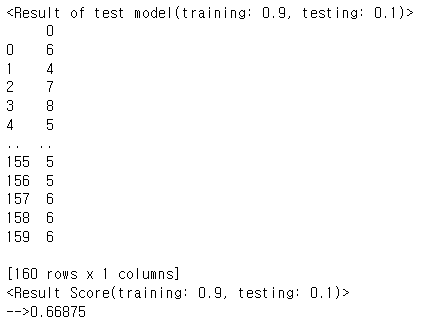
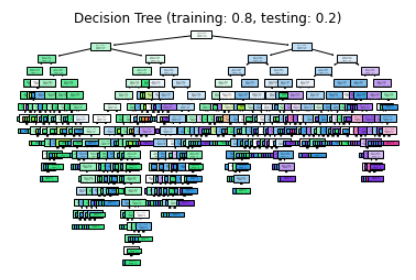
print("<Result Score(training: ",a,", testing: ",b,")>", sep="")

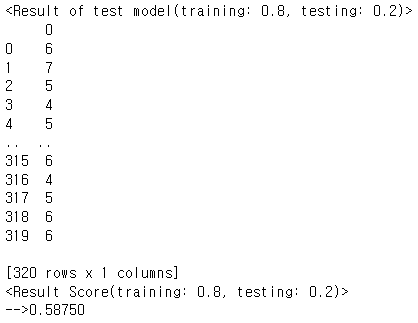
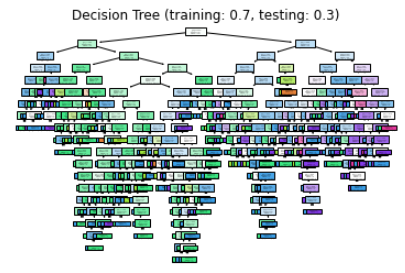
print('-->%.5f' % accuracy\_score(y\_test, y\_pred\_tr))

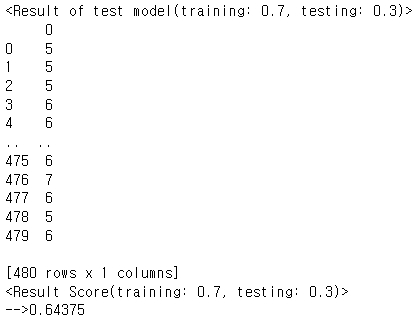
print()

**<Output Screen>**



**# Problem3 - K-NN**

**<Source code>**

import numpy as np

import pandas as pd

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

from sklearn import preprocessing

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import KFold

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

from sklearn.model\_selection import GridSearchCV

from sklearn.model\_selection import RandomizedSearchCV

# Get data

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

# Data Exploration

print(df.info(),end="\n\n")

print(df.head(),end="\n\n")

# Data Scaling

scaler = preprocessing.StandardScaler()

scaled\_df = scaler.fit\_transform(df)

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

# Split the dataset (X,y)

X = scaled\_df.drop(columns=['label']).values

y = df['label'].values

# Split the dataset (4/5 for training and 1/5 for testing)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,train\_size=0.8,test\_size=0.2,shuffle=True,stratify=y)

# Prepare cross validation

kfold = KFold(5,shuffle=True,random\_state=1)

# For Check Loop

idx=0

# Split the dataset into 5 subsets of equal size

for train, test in kfold.split(X):

# For Check Loop

idx+=1

print("Split[",idx,"]",sep="")

for k in [3,5]:

# Set train data & test data

X\_train, X\_test, y\_train, y\_test = X[train],X[test],y[train],y[test]

# KNN -> fit & predict

knn = KNeighborsClassifier(n\_neighbors=k)

cv\_scores = cross\_val\_score(knn,X\_train,y\_train,cv=5)

# Evaluation

print("<Result Score(k=",k,")>",sep="")

print("Scores:",np.round(cv\_scores,5),"/ mean:",np.round(np.mean(cv\_scores),5))

# GridSearchCV

param\_grid={'n\_neighbors':np.arange(1,25)}

knn\_gscv = GridSearchCV(knn,param\_grid,cv=5)

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

# Evaluation

print("Best params in GridSearchCV:",knn\_gscv.best\_params\_)

print("Best score in GridSearchCV:",knn\_gscv.best\_score\_)

# RandomizedSearchCV

param\_r\_grid={'n\_neighbors':np.arange(1,25)}

knn\_rgscv = RandomizedSearchCV(knn,param\_r\_grid,cv=5,scoring='accuracy')

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

# Evaluation

print("Best params in RandomizedSearchCV:",knn\_rgscv.best\_params\_)

print("Best score in RandomizedSearchCV:",knn\_rgscv.best\_score\_)

print()

**<Output Screen>**

