**Data Science HW**

**201635833 Lee SungMin**

**Data types, Statistical Data , and Feature names**

import numpy as np  
import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
from sklearn import preprocessing  
from sklearn import linear\_model  
import xlrd  
  
  
df = pd.read\_excel("bmi\_data\_phw3.xlsx",'dataset',index\_col=None)  
  
print(df)

#print total data  
print(df.columns.tolist())

#print data columns  
print(df.describe())

#print statistical data  
print(df.dtypes)

#print data types

**텍스트, 앉아있는, 검은색, 테이블이(가) 표시된 사진

자동 생성된 설명**

**Plot histogram(Height & Weight)**

HeightSet = df['Height (Inches)']  
WeightSet = df['Weight (Pounds)']

#Extract Height and Weight from dataset  
  
  
plt.subplot(211) #locate Height   
plt.hist(HeightSet,bins=10) #set bins as 10  
plt.title('Height Histogram')  
plt.xlabel('Height')  
plt.ylabel('Num of people')  
  
plt.subplot(212) #locate Weight  
plt.hist(WeightSet,bins=10,color='red') #set bins as 10  
plt.title('Weight Histgram')  
plt.xlabel('Weight')  
plt.ylabel('Num of people')  
  
plt.show()

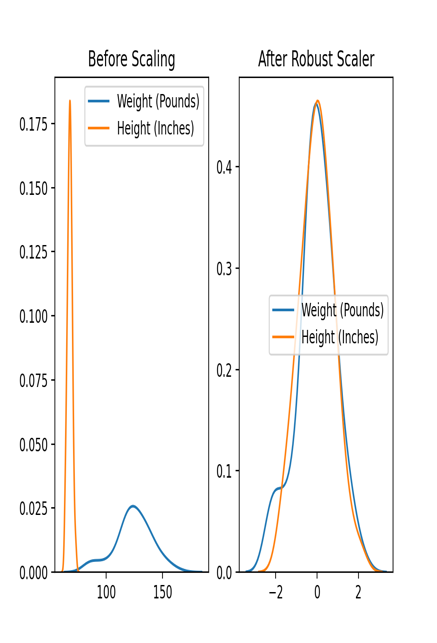
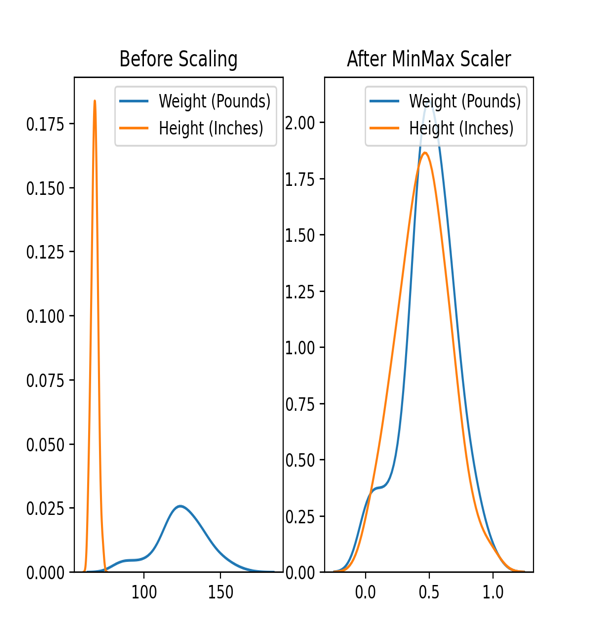
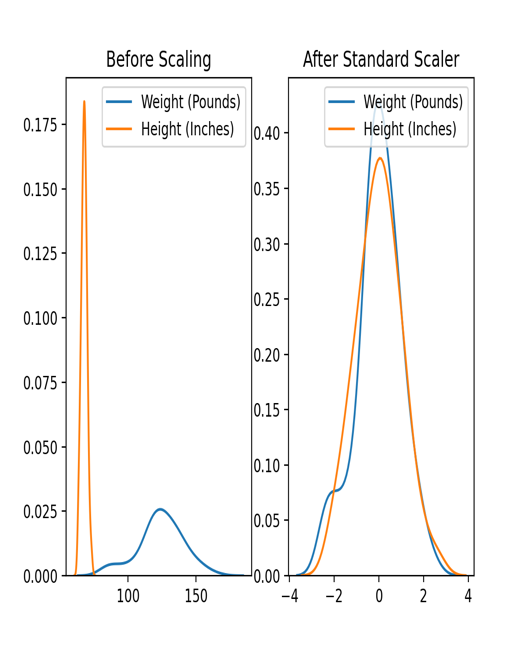
#show histogram

**스크린샷이(가) 표시된 사진

자동 생성된 설명**

**Plot Scaling result of height and Weight**

#StandardScaler using Height and Weight np.array made from above  
dataset = np.array(df[['Height (Inches)','Weight (Pounds)']]) # make data set with Height and Weight  
scaler = preprocessing.StandardScaler() #Use Standard Scaler  
scaled\_df = scaler.fit\_transform(dataset)  
scaled\_df = pd.DataFrame(scaled\_df,columns=['Height (Inches)','Weight (Pounds)'])  
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(6, 5)) #Because of show before, after use subplot  
ax1.set\_title('Before Scaling') # include this line to under 2 line is setting of before graph  
sns.kdeplot(df['Weight (Pounds)'], ax=ax1)  
sns.kdeplot(df['Height (Inches)'], ax=ax1)  
ax2.set\_title('After Standard Scaler') ) #include this line and 2 lines in below is setting for after scaler  
sns.kdeplot(scaled\_df['Weight (Pounds)'], ax=ax2)  
sns.kdeplot(scaled\_df['Height (Inches)'], ax=ax2)  
plt.show() #show result as graph  
  
#MinMaxScaler using Height and Weight np.array made from above  
dataset = np.array(df[['Height (Inches)','Weight (Pounds)']]) # make data set with Height and Weight  
scaler = preprocessing.MinMaxScaler() #Use MinMaxScaler  
scaled\_df = scaler.fit\_transform(dataset)  
scaled\_df = pd.DataFrame(scaled\_df,columns=['Height (Inches)','Weight (Pounds)'])  
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(6, 5)) #Because of show before, after use subplot  
ax1.set\_title('Before Scaling') # include this line to under 2 line is setting of before graph  
sns.kdeplot(df['Weight (Pounds)'], ax=ax1)  
sns.kdeplot(df['Height (Inches)'], ax=ax1)  
ax2.set\_title('After Standard Scaler') ) #include this line and 2 lines in below is setting for after scaler  
sns.kdeplot(scaled\_df['Weight (Pounds)'], ax=ax2)  
sns.kdeplot(scaled\_df['Height (Inches)'], ax=ax2)  
plt.show() #show result as graph  
  
#RobustScaler using Height and Weight np.array made from above  
dataset = np.array(df[['Height (Inches)','Weight (Pounds)']]) # make data set with Height and Weight  
scaler = preprocessing.RobustScaler() #User Robust Scaler  
scaled\_df = scaler.fit\_transform(dataset)  
scaled\_df = pd.DataFrame(scaled\_df,columns=['Height (Inches)','Weight (Pounds)'])  
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(6, 5)) #Because of show before, after use subplot  
ax1.set\_title('Before Scaling' # include this line to under 2 line is setting of before graph)  
sns.kdeplot(df['Weight (Pounds)'], ax=ax1)  
sns.kdeplot(df['Height (Inches)'], ax=ax1)  
ax2.set\_title('After Standard Scaler') #include this line and 2 lines in below is setting for after scaler  
sns.kdeplot(scaled\_df['Weight (Pounds)'], ax=ax2)  
sns.kdeplot(scaled\_df['Height (Inches)'], ax=ax2)  
plt.show() #show result as graph

****

**Compute linear regression equation E for dataset D**

#predict ze and find similarity with original dataset  
reg = linear\_model.LinearRegression()  
# make linear regression model  
reg.fit(HeightSet.values.reshape(-1,1),WeightSet)  
prdWeight = reg.predict(HeightSet[:,np.newaxis])  
#make equation E to predict and prdWeight is predicted result according to Equation E  
e = np.zeros(len(WeightSet))  
#make size 100 np array are made up with 0  
for i in range(0,len(WeightSet)):  
 e[i] = WeightSet[i] - prdWeight[i]  
#calculate E follow by pdf’s option  
ze = ((e-e.mean()))/e.std()  
#compute Ze follow the Equation in PDF  
alpha = 1  
# set Alpha values as 1  
df['Changed BMI'] = df['BMI']  
# create new column named Changed BMI  
tempdf = (df.values)  
  
for i in range(0 , len(ze)):  
 if (ze[i]<-alpha):  
 tempdf[i][5]=0  
 elif(ze[i]>alpha):  
 tempdf[i][5]=4  
# compare Ze with alpha and replace if Ze < -alpha or Ze> alpha to 0 and 4  
  
df2 = pd.DataFrame(tempdf,columns=[['Sex','Age','Height (Inches)','Weight (Pounds)','BMI','Changed BMI']])  
  
temp1 = df2.values  
count = 0  
for i in range(0, len(temp1)):  
 if(temp1[i][4] == temp1[i][5]):  
 count +=1  
#Check the same value after changing  
print("Original dataset Similarity : ",count)

#print compared by original ‘BMI’ and ‘Changed BMI’ so count mean similarity compare with original data set

**For Male Case**

#predict Male's Ze and find similarity with original dataset extract Sex = Male  
dfM = df.loc[df['Sex']=='Male']  
HeightM = dfM['Height (Inches)']  
WeightM = dfM['Weight (Pounds)']  
#Extract Male Height and Weight also total DF where Sex=’Male’  
regM = linear\_model.LinearRegression()  
# make linear regression model for Male  
regM.fit(HeightM.values.reshape(-1,1),WeightM)  
prdMWeight = regM.predict(HeightM[:,np.newaxis])  
#make equation E for Male to predict and prdMWeight is predicted result according to Equation E for Male  
eM = np.zeros(len(HeightM))  
#make np array with 0 size of HeightM  
eM = WeightM - prdMWeight  
#follow given Equation in PDf  
zeM = (eM - eM.mean())/eM.std()  
#calculate Ze for Male followed by Equation in PDF  
tempM = dfM.values  
zeMarray = zeM.values  
print('Male part')  
  
for i in range(0,len(dfM)):  
 if (zeMarray[i] < -alpha):  
 tempM[i][5] = 0  
 elif (zeMarray[i] > alpha):  
 tempM[i][5] = 4  
# compare ZeM with alpha and replace if ZeM < -alpha or ZeM> alpha to 0 and 4  
  
countM = 0  
for i in range(0,len(tempM)):  
 if tempM[i][4] == tempM[i][5]:  
 countM+=1  
#count Same data before change and changed  
print('Male Data Size : ',len(dfM))  
print('Male similarity : ',countM)

#show Similarity

**Female case**

#predict Female's Ze and find similarity with original dataset extract Sex = Female  
dfF = df[df['Sex']=='Female']  
  
HeightF = dfF['Height (Inches)']  
WeightF = dfF['Weight (Pounds)']  
#Extract Female Height and Weight also total DF where Sex=’Female’  
regF = linear\_model.LinearRegression()  
regF.fit(HeightF.values.reshape(-1,1),WeightF)  
prdFWeight = regF.predict(HeightF[:,np.newaxis])  
#make equation E for Female to predict and prdFWeight is predicted result according to Equation E for female  
eF = np.zeros(len(HeightF))  
#make np array with 0 size of HeightF  
eF = WeightF - prdFWeight  
#follow given Equation in PDf  
zeF= ((eF - eF.mean()))/eF.std()  
#calculate Ze for Male followed by Equation in PDF  
tempF = dfF.values  
zeFarray = zeF.values  
print('Female part')  
  
for i in range(0,len(dfF)):  
 if (zeFarray[i] < -alpha):  
 tempF[i][5] = 0  
 elif (zeFarray[i] > alpha):  
 tempF[i][5] = 4  
# compare ZeF with alpha and replace if ZeF < -alpha or ZeF> alpha to 0 and 4  
  
countF = 0  
for i in range(0,len(tempM)):  
 if tempM[i][4] == tempM[i][5]:  
 countF+=1  
#count Same data before change and changed  
print('Female Data Size : ',len(dfF))  
print('Female similarity : ',countF)

#show Similarity

**Regression print histogram :**

#print 'Ze' histogram  
plt.subplot(131)  
plt.xlabel('ze')  
plt.ylabel("Frequancy")  
plt.title("Distribution of Ze")  
plt.hist(ze,bins=10,rwidth= .8 , color='green')

#Male part

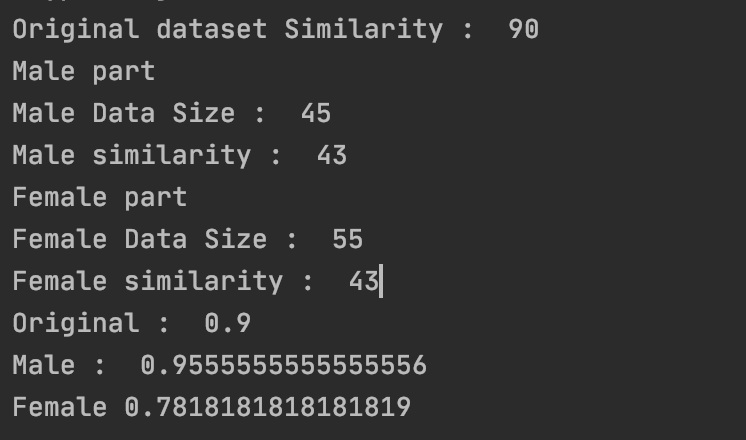
plt.subplot(132)  
plt.hist(zeM,bins=10, rwidth= .8)  
plt.xlabel('Ze for Male')  
plt.ylabel('Frequency')  
plt.title('Distibution of Ze(Male)')

#Female Part

plt.subplot(133)  
plt.hist(zeF,bins = 10 , rwidth= 0.8 ,color='red')  
plt.title('Distribution of Ze(Female')  
plt.xlabel('Ze for Female')  
plt.ylabel('Frequency')  
  
plt.show()

**Regression result screenshot시계이(가) 표시된 사진

자동 생성된 설명**

****