**PHW 3**

**Source Code**

import pandas as pd

import numpy as np

from pyparsing import col

import seaborn as sns

import matplotlib.pyplot as plt

import scipy.stats as ss

from sklearn import preprocessing

from sklearn import linear\_model

# Read data from file to dataf(dataframe)

dataf = pd.read\_excel("bmi\_data\_phw3.xlsx")

# Make dataframe to use original, male, female outlier

dataf\_original = dataf.copy()

dataf\_male = dataf.copy()

dataf\_female = dataf.copy()

# print statistical data

print("\nDataset Statistical Data")

print(dataf.describe())

# print feature name

print("\nFeature Name")

print(dataf.columns)

# print data type

print("\nData Type")

print(dataf.dtypes)

# make height histogram by BMI

histo=sns.FacetGrid(dataf,col="BMI")

histo.map(plt.hist,"Height (Inches)",bins=10,color="m")

plt.show()

# make weight histogram by BMI

histo=sns.FacetGrid(dataf,col="BMI")

histo.map(plt.hist,"Weight (Pounds)",bins=10,color="m")

plt.show()

# Scaling in sequence -> 0 = Standard, 1 = MinMax, 2 = Robust

for i in range(3):

    if i==0:

        scaler = preprocessing.StandardScaler()

        title="Standard Scaling"

    elif i==1:

        scaler = preprocessing.MinMaxScaler()

        title="MinMax Scaling"

    elif i==2:

        scaler = preprocessing.RobustScaler()

        title="Robust Scaling"

    scaled\_df=scaler.fit\_transform(dataf[["Height (Inches)","Weight (Pounds)"]])

    scaled\_df=pd.DataFrame(scaled\_df,columns=["Height (Inches)","Weight (Pounds)"])

    fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(6,5))

    ax1.set\_title("Before Scaling")

    sns.kdeplot(dataf["Height (Inches)"], ax=ax1)

    sns.kdeplot(dataf["Weight (Pounds)"], ax=ax1)

    ax2.set\_title(title)

    sns.kdeplot(scaled\_df["Height (Inches)"], ax=ax2)

    sns.kdeplot(scaled\_df["Weight (Pounds)"], ax=ax2)

    plt.show()

# Compute linear regression equation E

height=np.array(dataf\_original["Height (Inches)"])

weight=np.array(dataf\_original["Weight (Pounds)"])

E=linear\_model.LinearRegression()

E.fit(weight[:,np.newaxis],height)

# Compute e with E

e = weight-E.predict(weight[:,np.newaxis])

# Compute Ze with e

Ze = ss.zscore(e)

# Plot a histogram showing the distribution of Ze

plt.hist(Ze, bins=10, rwidth=0.8)

plt.title('Distribution of Ze')

plt.xlabel('Ze')

plt.ylabel('Nubmer')

plt.show()

# Add Ze data to dataframe to show alpha

dataf\_original["Ze"] = Ze

# Decide a value alpha and display

print("Estimate BMI value is 0")

print(dataf\_original.loc[dataf\_original["Ze"]<-1.5])

print()

print("Estimate BMI value is 4")

print(dataf\_original.loc[dataf\_original["Ze"]>1.5])

# Devide group Male

dataf\_male = dataf\_male.loc[dataf\_male["Sex"]=="Male"].copy()

# Compute linear regression equation E

height=np.array(dataf\_male["Height (Inches)"])

weight=np.array(dataf\_male["Weight (Pounds)"])

E=linear\_model.LinearRegression()

E.fit(weight[:,np.newaxis],height)

# Compute e with E

e = weight-E.predict(weight[:,np.newaxis])

# Compute Ze with e

Ze = ss.zscore(e)

# Plot a histogram showing the distribution of Ze

plt.hist(Ze, bins=10, rwidth=0.8)

plt.title('Distribution of Ze (Male)')

plt.xlabel('Ze')

plt.ylabel('Nubmer')

plt.show()

# Add Ze data to dataframe to show alpha

dataf\_male["Ze"] = Ze

# Decide a value alpha and display

print("Estimate BMI value is 0")

print(dataf\_male.loc[dataf\_male["Ze"]<-1.5])

print()

print("Estimate BMI value is 4")

print(dataf\_male.loc[dataf\_male["Ze"]>1.5])

# Devide group Female

dataf\_female = dataf\_female.loc[dataf\_female["Sex"]=="Female"].copy()

# Compute linear regression equation E

height=np.array(dataf\_female["Height (Inches)"])

weight=np.array(dataf\_female["Weight (Pounds)"])

E=linear\_model.LinearRegression()

E.fit(weight[:,np.newaxis],height)

# Compute e with E

e = weight-E.predict(weight[:,np.newaxis])

# Compute Ze with e

Ze = ss.zscore(e)

# Plot a histogram showing the distribution of Ze

plt.hist(Ze, bins=10, rwidth=0.8)

plt.title('Distribution of Ze (Female)')

plt.xlabel('Ze')

plt.ylabel('Nubmer')

plt.show()

# Add Ze data to dataframe to show alpha

dataf\_female["Ze"] = Ze

# Decide a value alpha and display

print("Estimate BMI value is 0")

print(dataf\_female.loc[dataf\_female["Ze"]<-1.5])

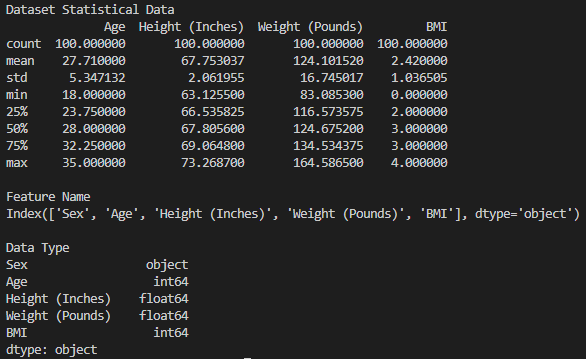
print()

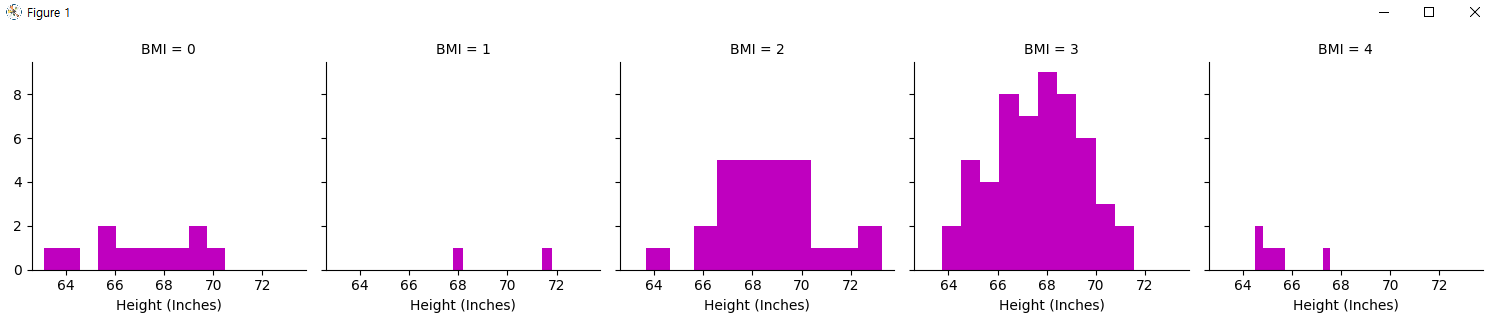
print("Estimate BMI value is 4")

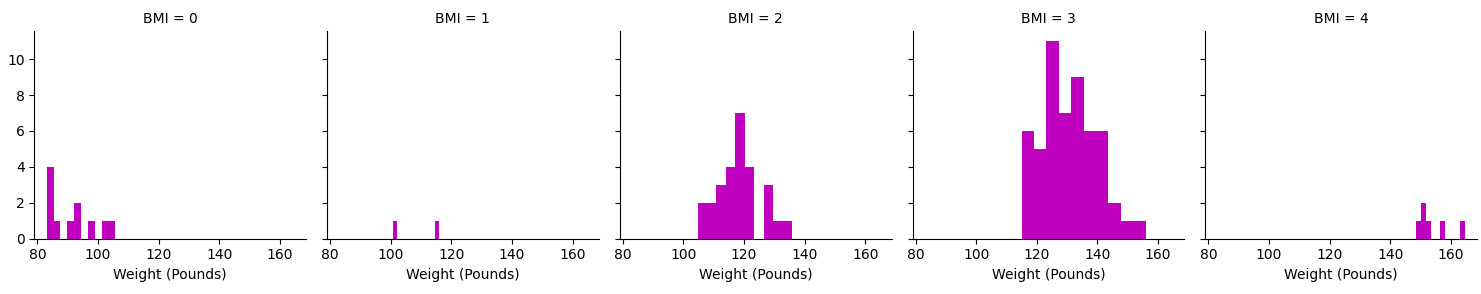
print(dataf\_female.loc[dataf\_female["Ze"]>1.5])

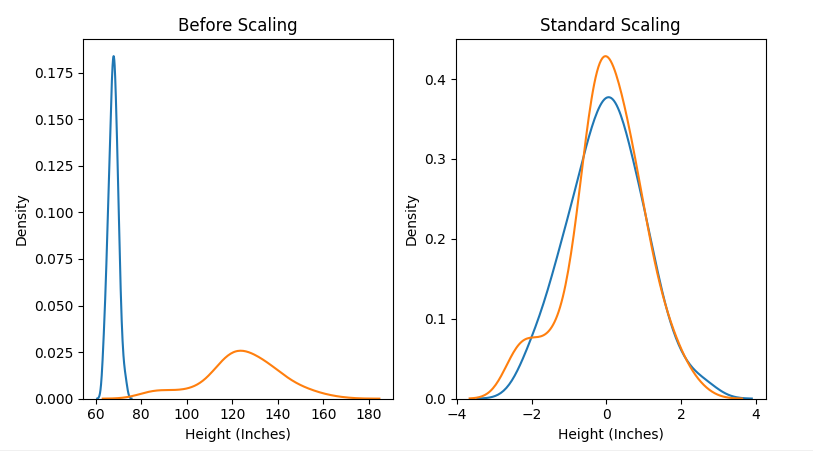
**Output Screen Capture**

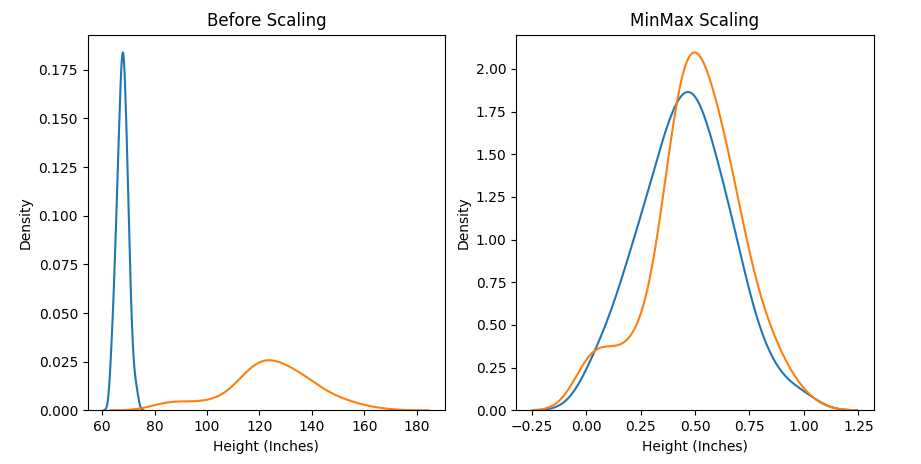
**1. Peek into the dataset (data exploration)**

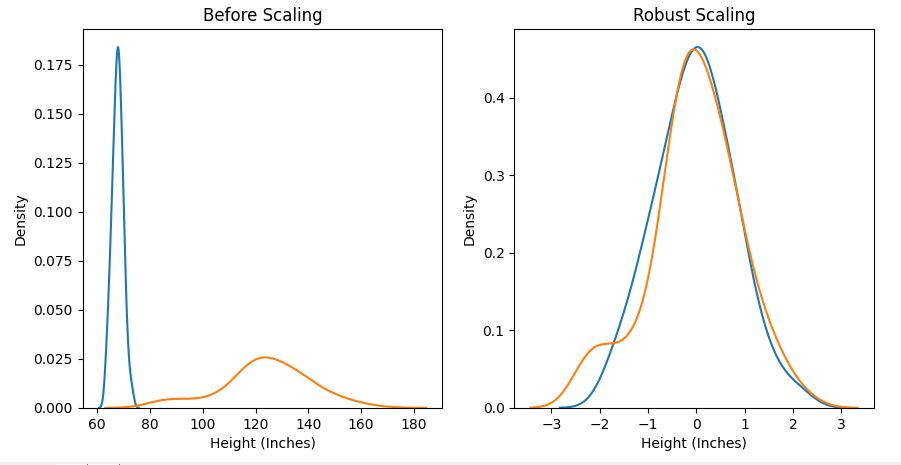




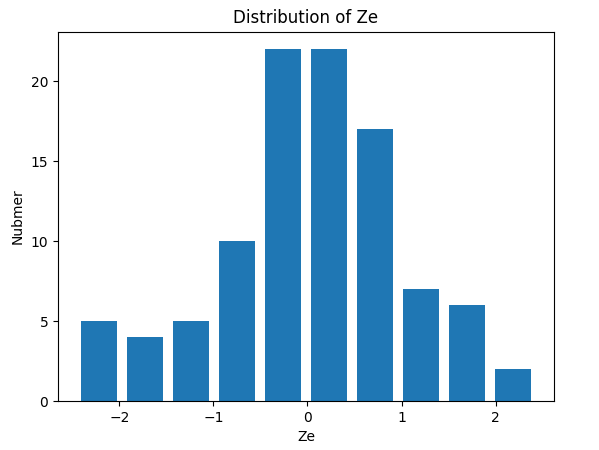


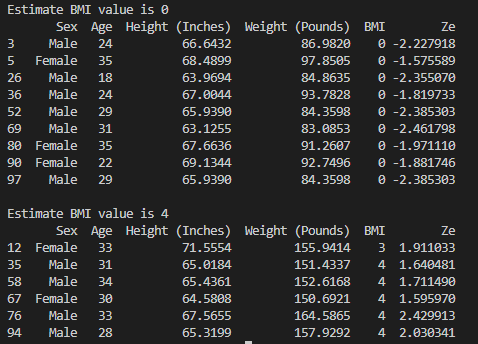




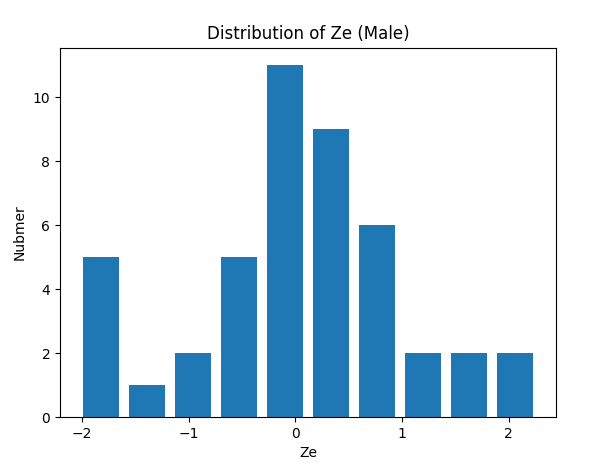


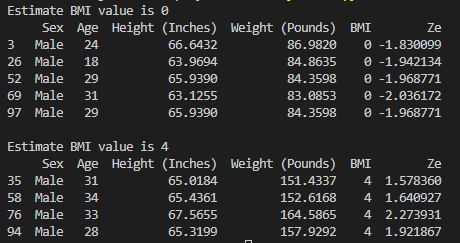
**2. Find Outlier People**



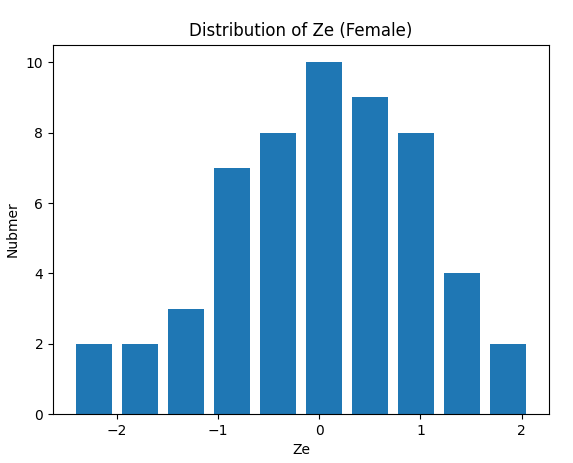


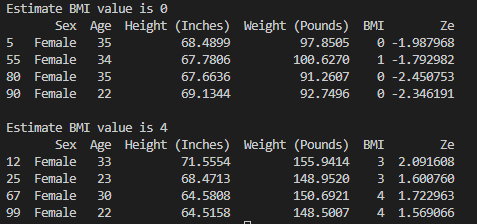
**3. Divide dataset (Male)**





**3. Divide dataset (Female)**





**Discuss (Compare BMI estimates with the actual BMI values in the given dataset)**

In some case and some BMI values in estimates are same with real dataset but some is different. So our team learn below.

Estimate result is pretty similar to real data but not exactly same, So we learn that validation is need in data part.

**Discuss (Active Learning : Jupyter Notebook or Colaboratory)**

We can run code one by one. (Not a all code in project)

We don’t have to type print function when print dataset

In Colaboratory we can doing work together simultaneously because we can share code with others in real time. But we have to mount dataset into google drive to use. It already installed lots of programming package so it has more comfortable environment than own laptop. It can solve memory error that occur in local environment.

**Members role**

Our team have done PHW3 individually first.

After individual work we have meeting at Webex and discuss about PHW3

**이준희 – Data exploration, Find outlier in original dataset (main) / 25%**

**조현식 – Data exploration, Find outlier in original dataset (main) / 25%**

**이민아 – Data exploration), Find outlier in divide dataset (main) / 25%**

**차원우 – Data exploration, Find outlier in divide dataset (main / 25%**