**LAB 3**

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

**Source Code**

import pandas as pd

import numpy as np

from pyparsing import col

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn import preprocessing

from sklearn import linear\_model

# Read data frome file to dataf(dataframe)

dataf = pd.read\_csv("bmi\_data\_lab3.csv")

# 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)

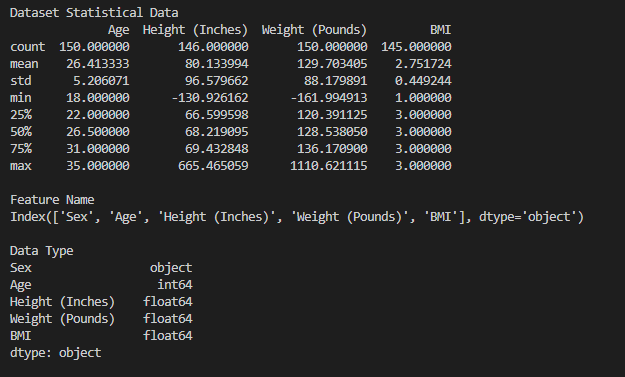
    ax2.set\_title(title)

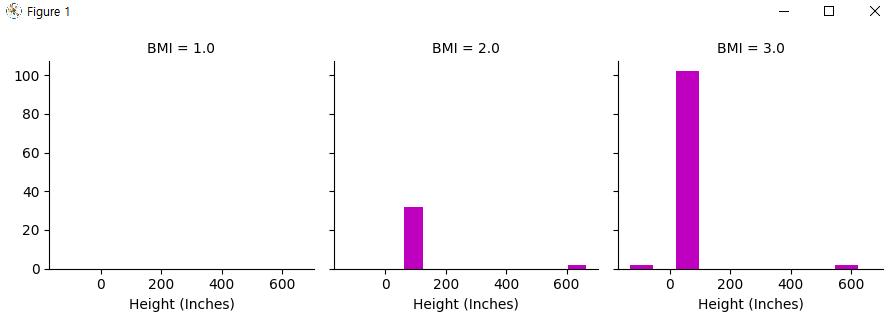
    sns.kdeplot(scaled\_df["Height (Inches)"], ax=ax2)

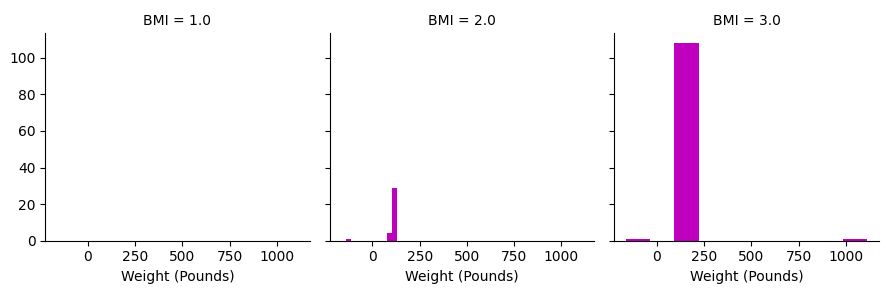
    sns.kdeplot(scaled\_df["Weight (Pounds)"], ax=ax2)

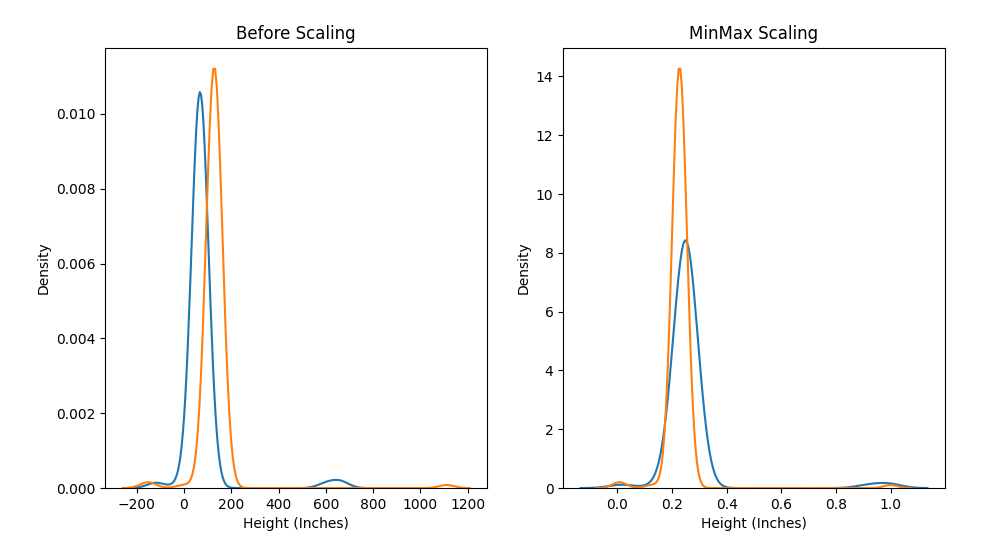
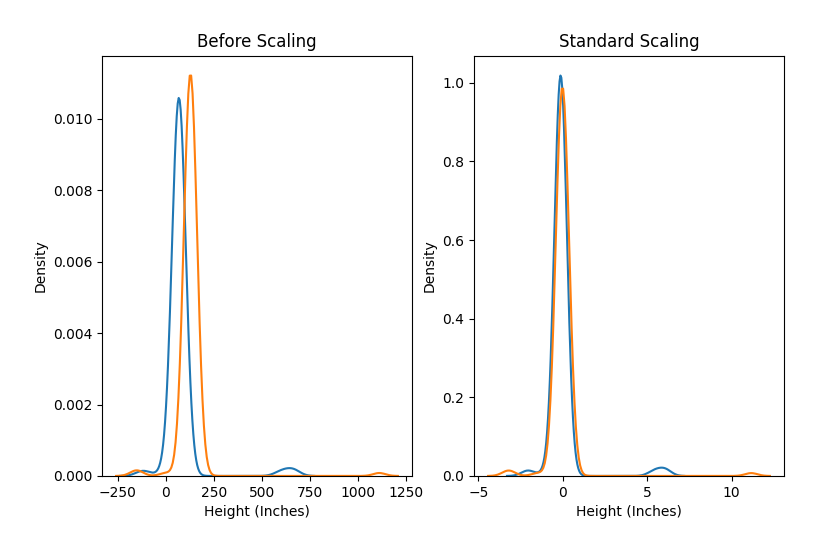
    plt.show()

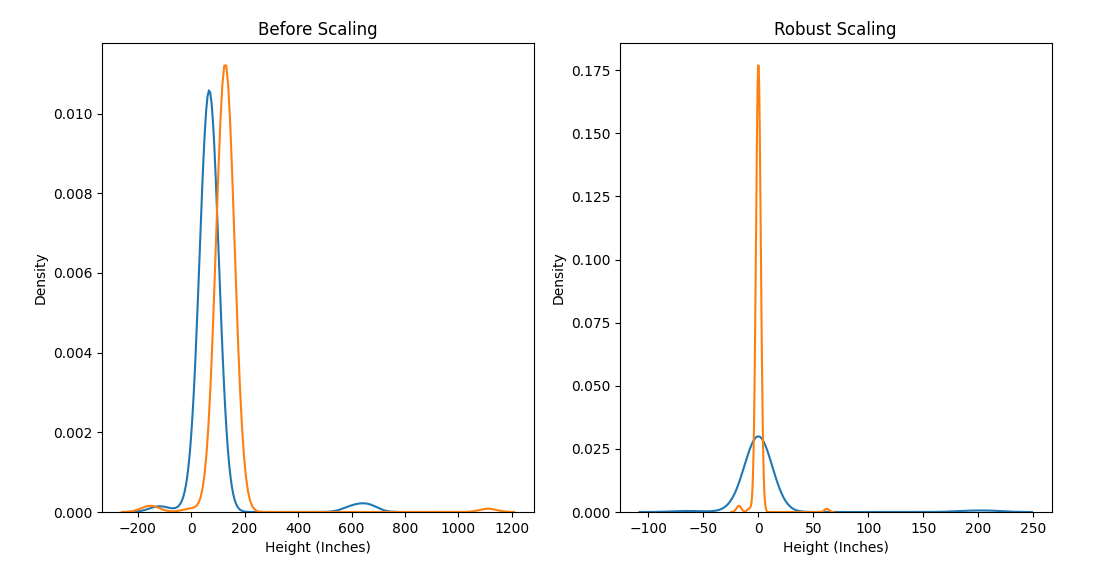
**Output Screen Capture**











**2. Missing value manipulation (simple)**

**Source Code**

import pandas as pd

import numpy as np

from pyparsing import col

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn import preprocessing

from sklearn import linear\_model

# Read data frome file to dataf(dataframe)

dataf = pd.read\_csv("bmi\_data\_lab3.csv")

# missing value manipulation

ms\_df = dataf

# height minus or over 80 inches are dirty data

# weight minus or over 300 pounds are dirty data

# change dirty data to nan

for i in range(len(ms\_df)):

    if ms\_df.loc[i,"Height (Inches)"]>80 or ms\_df.loc[i,"Height (Inches)"]<=0:

        ms\_df.loc[i,"Height (Inches)"] = np.nan

    if ms\_df.loc[i,"Weight (Pounds)"]>300 or ms\_df.loc[i,"Weight (Pounds)"]<=0:

        ms\_df.loc[i,"Weight (Pounds)"] = np.nan

# print data and count NAN

for i in ms\_df:

    print("\n"+i)

    print(ms\_df[ms\_df[i].isna()])

    print("Nan count : "+ str(len(ms\_df[ms\_df[i].isna()])))

# extract all rows without NAN

print("Extract all rows without NAN")

print(ms\_df.dropna(axis=0,how="any",inplace=False))

# make 4 dataframe to each fillna function

mean\_df=ms\_df.copy()

median\_df=ms\_df.copy()

ff\_df=ms\_df.copy()

bf\_df=ms\_df.copy()

# fill NAN with mean value

mean\_df["Height (Inches)"]=mean\_df["Height (Inches)"].fillna(mean\_df["Height (Inches)"].mean())

mean\_df["Weight (Pounds)"]=mean\_df["Weight (Pounds)"].fillna(mean\_df["Weight (Pounds)"].mean())

mean\_df["BMI"]=mean\_df["BMI"].fillna(mean\_df["BMI"].mean())

# fill NAN with median value

median\_df["Height (Inches)"]=median\_df["Height (Inches)"].fillna(median\_df["Height (Inches)"].median())

median\_df["Weight (Pounds)"]=median\_df["Weight (Pounds)"].fillna(median\_df["Weight (Pounds)"].median())

median\_df["BMI"]=median\_df["BMI"].fillna(median\_df["BMI"].median())

# print dataframe

print("\nFill NAN with Mean value")

print(mean\_df)

print("\nFill NAN with Median value")

print(median\_df)

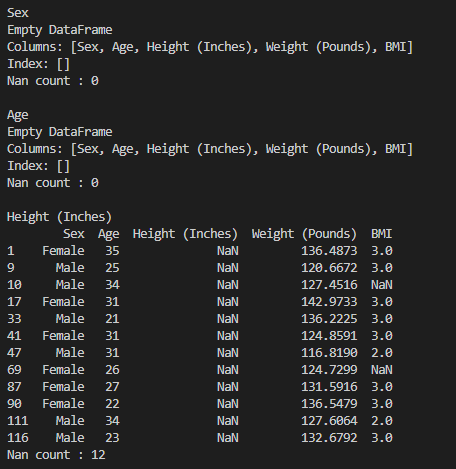
print("\nUsing ffill function")

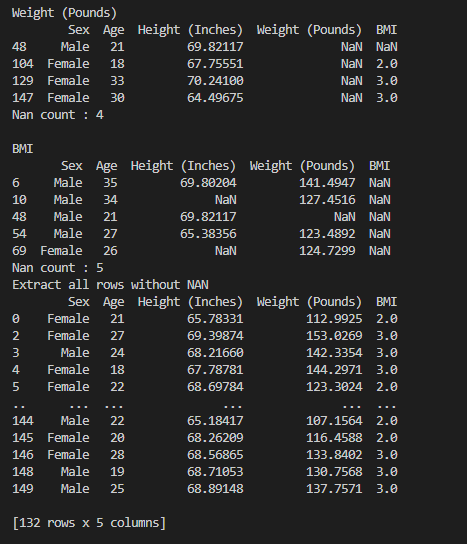
print(ff\_df.ffill())

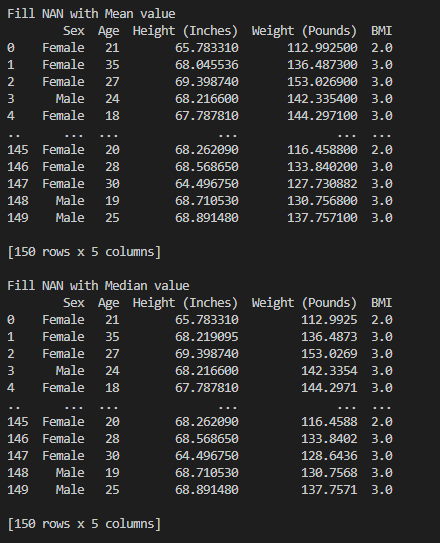
print("\nUsing bfill function")

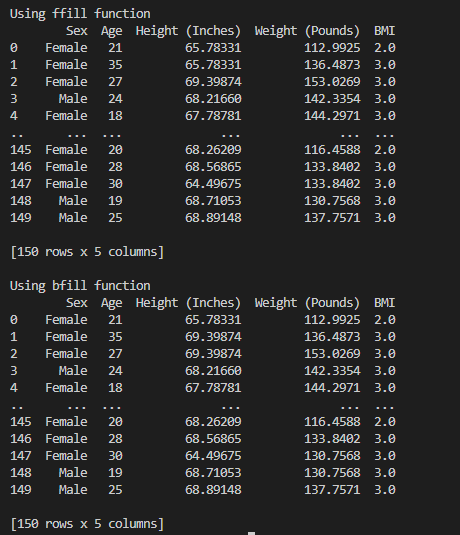
print(bf\_df.bfill())

**Output Screen Capture**









**3. Missing value manipulation (more elaborate)**

**Source Code**

import pandas as pd

import numpy as np

from pyparsing import col

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn import preprocessing

from sklearn import linear\_model

# Read data frome file to dataf(dataframe)

dataf = pd.read\_csv("bmi\_data\_lab3.csv")

# missing value manipulation

ms\_df = dataf

# height minus or over 80 inches are dirty data

# weight minus or over 300 pounds are dirty data

# change dirty data to nan

for i in range(len(ms\_df)):

    if ms\_df.loc[i,"Height (Inches)"]>80 or ms\_df.loc[i,"Height (Inches)"]<=0:

        ms\_df.loc[i,"Height (Inches)"] = np.nan

    if ms\_df.loc[i,"Weight (Pounds)"]>300 or ms\_df.loc[i,"Weight (Pounds)"]<=0:

        ms\_df.loc[i,"Weight (Pounds)"] = np.nan

reg\_df=ms\_df.copy()

reg\_df=reg\_df.dropna(axis=0,how="any",inplace=False)

# Compute linear regression equation E

height=np.array(reg\_df["Height (Inches)"])

weight=np.array(reg\_df["Weight (Pounds)"])

reg=linear\_model.LinearRegression()

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

px=np.array([height.min()-1,height.max()+1])

py=reg.predict(px[:,np.newaxis])

plt.scatter(height, weight)

plt.plot(px,py,color="r")

plt.title('Result (ALL)', fontsize=20)

all\_df=ms\_df.copy()

# fill NAN to predict by E

for i in range(len(all\_df)):

    if np.isnan(all\_df.loc[i,"Height (Inches)"]):

        all\_df.loc[i,"Height (Inches)"]=(all\_df.loc[i,"Weight (Pounds)"]-reg.intercept\_)/reg.coef\_

        plt.scatter([all\_df.loc[i,"Height (Inches)"]],[all\_df.loc[i,"Weight (Pounds)"]],c="black")

    if np.isnan(all\_df.loc[i,"Weight (Pounds)"]):

        all\_df.loc[i,"Weight (Pounds)"]=reg.predict([[all\_df.loc[i,"Height (Inches)"]]])

        plt.scatter([all\_df.loc[i,"Height (Inches)"]],[all\_df.loc[i,"Weight (Pounds)"]],c="black")

plt.show()

# make array

height\_f=[]

weight\_f=[]

height\_m=[]

weight\_m=[]

height\_b1=[]

weight\_b1=[]

height\_b2=[]

weight\_b2=[]

height\_b3=[]

weight\_b3=[]

# input value to array

for i in reg\_df.index:

    if reg\_df.loc[i,"Sex"]=="Female":

        height\_f += [reg\_df.loc[i,"Height (Inches)"]]

        weight\_f += [reg\_df.loc[i,"Weight (Pounds)"]]

    if reg\_df.loc[i,"Sex"]=="Male":

        height\_m += [reg\_df.loc[i,"Height (Inches)"]]

        weight\_m += [reg\_df.loc[i,"Weight (Pounds)"]]

    if reg\_df.loc[i,"BMI"]==1:

        height\_b1 += [reg\_df.loc[i,"Height (Inches)"]]

        weight\_b1 += [reg\_df.loc[i,"Weight (Pounds)"]]

    if reg\_df.loc[i,"BMI"]==2:

        height\_b2 += [reg\_df.loc[i,"Height (Inches)"]]

        weight\_b2 += [reg\_df.loc[i,"Weight (Pounds)"]]

    if reg\_df.loc[i,"BMI"]==3:

        height\_b3 += [reg\_df.loc[i,"Height (Inches)"]]

        weight\_b3 += [reg\_df.loc[i,"Weight (Pounds)"]]

# make np.array

height\_f=np.array(height\_f)

weight\_f=np.array(weight\_f)

height\_m=np.array(height\_m)

weight\_m=np.array(weight\_m)

height\_b1=np.array(height\_b1)

weight\_b1=np.array(weight\_b1)

height\_b2=np.array(height\_b2)

weight\_b2=np.array(weight\_b2)

height\_b3=np.array(height\_b3)

weight\_b3=np.array(weight\_b3)

# Female regression

reg\_f=linear\_model.LinearRegression()

reg\_f.fit(height\_f[:,np.newaxis],weight\_f)

px=np.array([height\_f.min()-1,height\_f.max()+1])

py=reg\_f.predict(px[:,np.newaxis])

plt.scatter(height\_f, weight\_f)

plt.plot(px,py,color="r")

plt.title('Result (Female)', fontsize=20)

fd\_df=ms\_df.copy()

for i in range(len(fd\_df)):

    if fd\_df.loc[i,"Sex"]=="Female":

        if np.isnan(fd\_df.loc[i,"Height (Inches)"]):

            fd\_df.loc[i,"Height (Inches)"]=(fd\_df.loc[i,"Weight (Pounds)"]-reg\_f.intercept\_)/reg\_f.coef\_

            plt.scatter([fd\_df.loc[i,"Height (Inches)"]],[fd\_df.loc[i,"Weight (Pounds)"]],c="black")

        if np.isnan(fd\_df.loc[i,"Weight (Pounds)"]):

            fd\_df.loc[i,"Weight (Pounds)"]=reg\_f.predict([[fd\_df.loc[i,"Height (Inches)"]]])

            plt.scatter([fd\_df.loc[i,"Height (Inches)"]],[fd\_df.loc[i,"Weight (Pounds)"]],c="black")

plt.show()

# Male regression

reg\_m=linear\_model.LinearRegression()

reg\_m.fit(height\_m[:,np.newaxis],weight\_m)

px=np.array([height\_m.min()-1,height\_m.max()+1])

py=reg\_m.predict(px[:,np.newaxis])

plt.scatter(height\_m, weight\_m)

plt.plot(px,py,color="r")

plt.title('Result (Male)', fontsize=20)

ma\_df=ms\_df.copy()

for i in range(len(ma\_df)):

    if ma\_df.loc[i,"Sex"]=="Male":

        if np.isnan(ma\_df.loc[i,"Height (Inches)"]):

            ma\_df.loc[i,"Height (Inches)"]=(ma\_df.loc[i,"Weight (Pounds)"]-reg\_m.intercept\_)/reg\_m.coef\_

            plt.scatter([ma\_df.loc[i,"Height (Inches)"]],[ma\_df.loc[i,"Weight (Pounds)"]],c="black")

        if np.isnan(ma\_df.loc[i,"Weight (Pounds)"]):

            ma\_df.loc[i,"Weight (Pounds)"]=reg\_m.predict([[ma\_df.loc[i,"Height (Inches)"]]])

            plt.scatter([ma\_df.loc[i,"Height (Inches)"]],[ma\_df.loc[i,"Weight (Pounds)"]],c="black")

plt.show()

# BMI 1 regression

reg\_b1=linear\_model.LinearRegression()

reg\_b1.fit(height\_b1[:,np.newaxis],weight\_b1)

px=np.array([height\_b1.min()-1,height\_b1.max()+1])

py=reg\_b1.predict(px[:,np.newaxis])

plt.scatter(height\_b1, weight\_b1)

plt.plot(px,py,color="r")

plt.title('Result (BMI(1))', fontsize=20)

b1\_df=ms\_df.copy()

for i in range(len(b1\_df)):

    if b1\_df.loc[i,"BMI"]==1:

        if np.isnan(b1\_df.loc[i,"Height (Inches)"]):

            b1\_df.loc[i,"Height (Inches)"]=(b1\_df.loc[i,"Weight (Pounds)"]-reg\_b1.intercept\_)/reg\_b1.coef\_

            plt.scatter([b1\_df.loc[i,"Height (Inches)"]],[b1\_df.loc[i,"Weight (Pounds)"]],c="black")

        if np.isnan(b1\_df.loc[i,"Weight (Pounds)"]):

            b1\_df.loc[i,"Weight (Pounds)"]=reg\_b1.predict([[b1\_df.loc[i,"Height (Inches)"]]])

            plt.scatter([b1\_df.loc[i,"Height (Inches)"]],[b1\_df.loc[i,"Weight (Pounds)"]],c="black")

plt.show()

# BMI 2 regression

reg\_b2=linear\_model.LinearRegression()

reg\_b2.fit(height\_b2[:,np.newaxis],weight\_b2)

px=np.array([height\_b2.min()-1,height\_b2.max()+1])

py=reg\_b2.predict(px[:,np.newaxis])

plt.scatter(height\_b2, weight\_b2)

plt.plot(px,py,color="r")

plt.title('Result (BMI(2))', fontsize=20)

b2\_df=ms\_df.copy()

for i in range(len(b2\_df)):

    if b2\_df.loc[i,"BMI"]==2:

        if np.isnan(b2\_df.loc[i,"Height (Inches)"]):

            b2\_df.loc[i,"Height (Inches)"]=(b2\_df.loc[i,"Weight (Pounds)"]-reg\_b2.intercept\_)/reg\_b2.coef\_

            plt.scatter([b2\_df.loc[i,"Height (Inches)"]],[b2\_df.loc[i,"Weight (Pounds)"]],c="black")

        if np.isnan(b2\_df.loc[i,"Weight (Pounds)"]):

            b2\_df.loc[i,"Weight (Pounds)"]=reg\_b2.predict([[b2\_df.loc[i,"Height (Inches)"]]])

            plt.scatter([b2\_df.loc[i,"Height (Inches)"]],[b2\_df.loc[i,"Weight (Pounds)"]],c="black")

plt.show()

# BMI 3 regression

reg\_b3=linear\_model.LinearRegression()

reg\_b3.fit(height\_b3[:,np.newaxis],weight\_b3)

px=np.array([height\_b3.min()-1,height\_b3.max()+1])

py=reg\_b3.predict(px[:,np.newaxis])

plt.scatter(height\_b3, weight\_b3)

plt.plot(px,py,color="r")

plt.title('Result (BMI(3))', fontsize=20)

b3\_df=ms\_df.copy()

for i in range(len(b3\_df)):

    if b3\_df.loc[i,"BMI"]==3:

        if np.isnan(b3\_df.loc[i,"Height (Inches)"]):

            b3\_df.loc[i,"Height (Inches)"]=(b3\_df.loc[i,"Weight (Pounds)"]-reg\_b3.intercept\_)/reg\_b3.coef\_

            plt.scatter([b3\_df.loc[i,"Height (Inches)"]],[b3\_df.loc[i,"Weight (Pounds)"]],c="black")

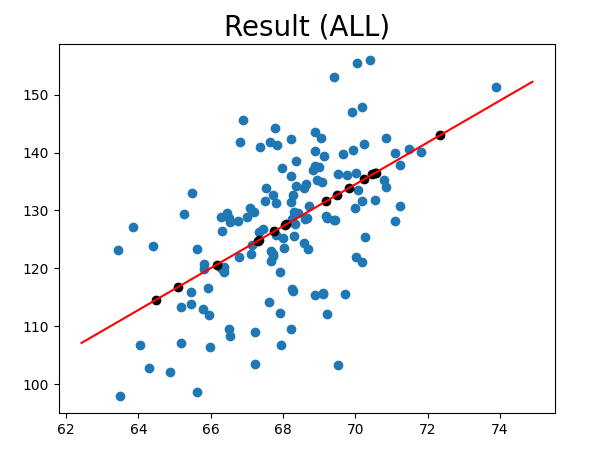
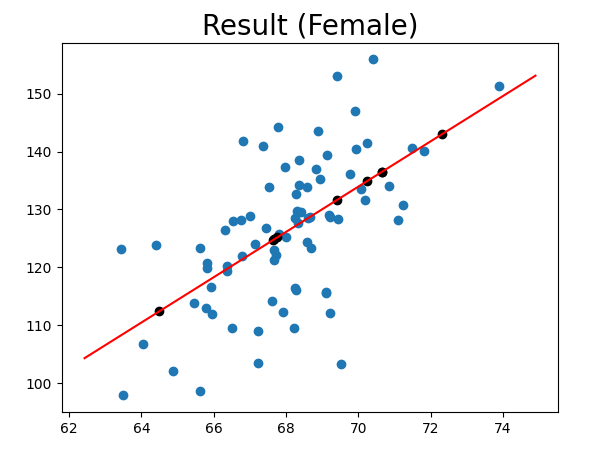
        if np.isnan(b3\_df.loc[i,"Weight (Pounds)"]):

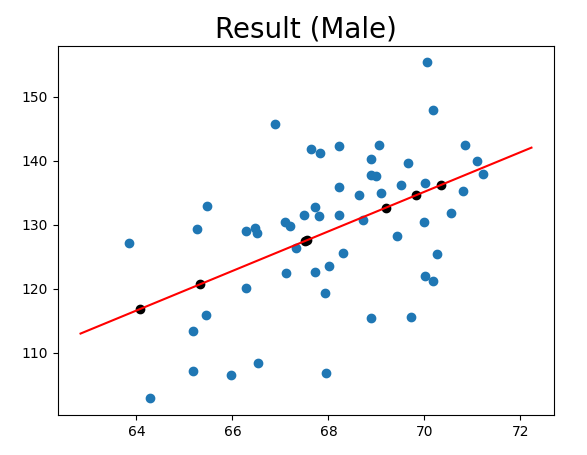
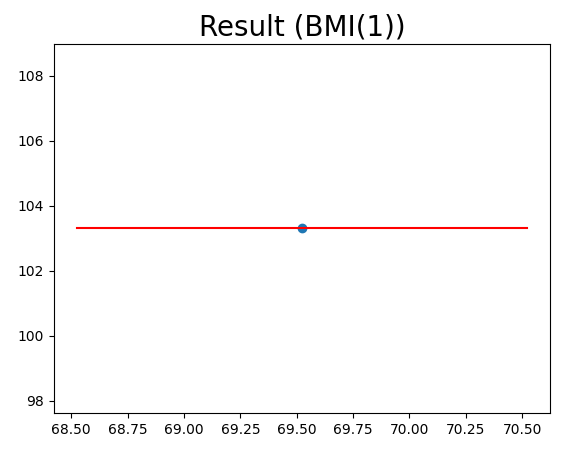
            b3\_df.loc[i,"Weight (Pounds)"]=reg\_b3.predict([[b3\_df.loc[i,"Height (Inches)"]]])

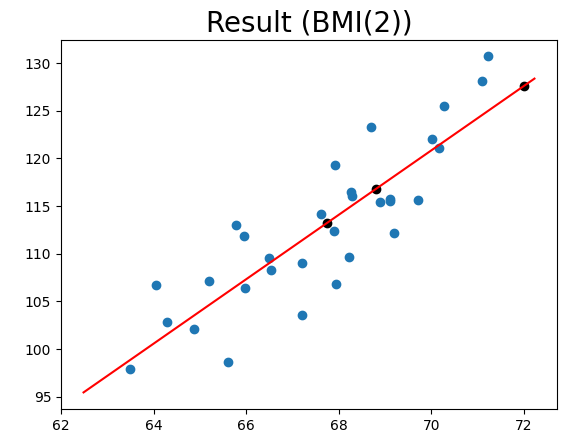
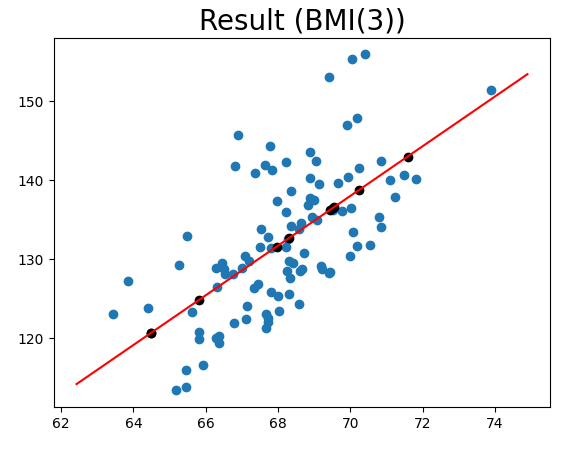
            plt.scatter([b3\_df.loc[i,"Height (Inches)"]],[b3\_df.loc[i,"Weight (Pounds)"]],c="black")

plt.show()

**Output Screen Capture**

**Members role**

**이준희 – 1, 2, 3 (all data regression) / 25%**

**조현식 – 1, 2, 3 (Female data regression) / 25%**

**이민아 – 1, 2, 3 (BMI data regression) / 25%**

**차원우 – 1, 2, 3 (BMI data regression) / 25%**