ELEC 2600 Lab Exercise #5

Submission Deadline: May 6, 2025 (Tuesday), 6:00 PM

Objectives:

- To study the joint behavior of two random variables by ploting a 2D scatter plot and a contour plot of the 2D Gaussian distribution.
- To build a threshold classifier on one feature (one random variable), perform pattern recognition, and evaluate the performance of the classifier.
- To build a threshold classifier on two features (two random variables), perform pattern recognition, and evaluate the performance of the classifier.

In this lab, we will look at how python can be used to analyze data using the pandas data analysis package, how we can model data using probability distributions studied in class, and perform pattern recognition.

We will be using a data file that has been obtained from https://www.kaggle.com/mustafaali96/weight-height. This dataset has 10,000 rows and 3 columns named “Gender”, “Height” and “Weight”. Each row provides the gender, height and weight of a person. **Height is given in inches and weight in pounds**.

In this problem, we extract two smaller datasets from this dataset: "train_data" and "test_data".

We will use the "train_data" dataset to visualize the joint behavior of the two random variables, height and weight, by plotting a 2D scatter plot and a contour plot of the 2D Gaussian distribution fit to this dataset. We will also visualize the conditional distributions of height and weight given gender. By analyzing the empirical distributions of the Height and Weight, we will build threshold classifiers that compares either the height or the weight to a fixed threshold to classify an entry as being either "Male" or "Female."

We will evaluate the performance of the classifiers on the "test_data" dataset. We compare the classification results with the "ground truth" labels contained in the Gender column to compute the empirical classification accuracy. The reason we use a separate dataset to evaluate the performance is that this better reflects the actual performance of the classifier in the field, where it will encounter new people, not previously seen during training.

Finally, we will build a threshold classifier that acts on a linear combination of the two features (e.g., the height and the weight), and compare its performance with the classifiers based on height or weight alone.

Download the file "Lab5_Weight_height.csv" from the course website to complete the lab.

We recommend you to use the pandas package, numpy package and the scipy and matplotlib.pyplot libraries in python, as you have done in previous labs.

For more information about pandas, please see https://pandas.pydata.org/

After you have completed the notebook, export it as pdf for submission. Go to File, click Download as, click HTML (.html), then convert the html file to pdf file.

We first load all the packages we need for this lab.

```
# load pandas, a data analysis package
# load numpy, a scientific computing package
# load scipy.stats, a module contains probability functions
# load matplotlib.pyplot, a framework provides a Matlab-like plotting
import pandas as pd
import numpy as np
import scipy.stats
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from scipy.stats import multivariate_normal

# threshold classifier
def threshold_classifier(feature, threshold):
    # if feature value is greater than the threshlod, classify to
Male. Otherwise, Female
    output = np.where(feature>=threshold, 'Male', 'Female')
    return output

# compare the two sets of labels
def compare_labels(testlabel, ground_truth):
    correct = np.where(testlabel==ground_truth,1, 0)
    accuracy = sum(correct)/np.size(correct)
    return accuracy
```

Below, we load the datafile using **pandas.read_csv** function into a **pandas.DataFrame**. We then create two subsets, "train_data" and "test_data" of size "n_train" and "n_test". The datasets should have an equal number of males and females, so the number of data in each dataset should be even.

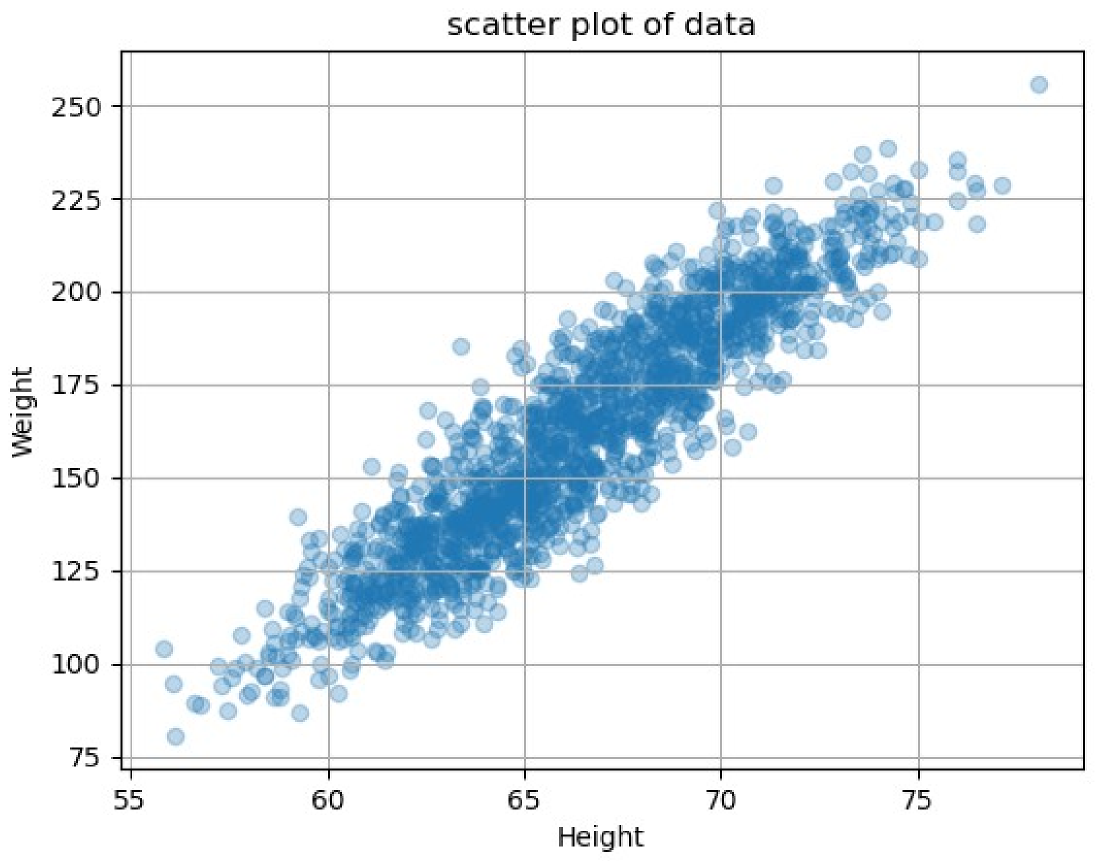
```
# read data
raw_data = pd.read_csv('Lab5_Weight_height.csv')

# sizes of smaller datasets
n_test = 800
n_train = 1600

# constructing the train and test datasets
# note the first 5000 rows of raw_data are males and the last 5000 are
females.
test_data = pd.concat([ raw_data[0:(n_test//2)], raw_data[5000:
(5000+n_test//2)] ])
train_data = pd.concat([ raw_data[n_test//2:(n_test//2+n_train//2)], \
                       raw_data[(5000+n_test//2) :
(5000+n_test//2+n_train//2)] ])

test_data
```

```
      Gender     Height      Weight
0       Male  73.847017  241.893563
1       Male  68.781904  162.310473
2       Male  74.110105  212.740856
3       Male  71.730978  220.042470
4       Male  69.881796  206.349801
...      ...        ...         ...
5395  Female  62.285114  132.091499
5396  Female  64.328491  135.819094
5397  Female  62.518234  129.279568
5398  Female  62.892831  131.130447
5399  Female  62.396154  127.032748

[800 rows x 3 columns]
```

Below, we visualize the joint behavior of two random variables by plotting a 2D scatter plot of height and weight.

We plot the 2D scatter plots using ax.scatter.

```python
# create figure
fig, ax = plt.subplots()

# create scatter plot
ax.scatter(train_data['Height'], train_data['Weight'], alpha=0.3)

plt.xlabel('Height')
plt.ylabel('Weight')
plt.grid(True)
ax.set_title('scatter plot of data')

plt.show()
```



We start by modelling the joint distribution of height and weight using a 2D Gaussian distribution $f_{X,Y}(x,y)$ with mean vector m and covariance matrix C, where X and Y are the height and weight of a person, respectively.

The code below estimates the mean and covariance that best fit the data using the **pandas.DataFrame.mean** and **pandas.DataFrame.cov** functions

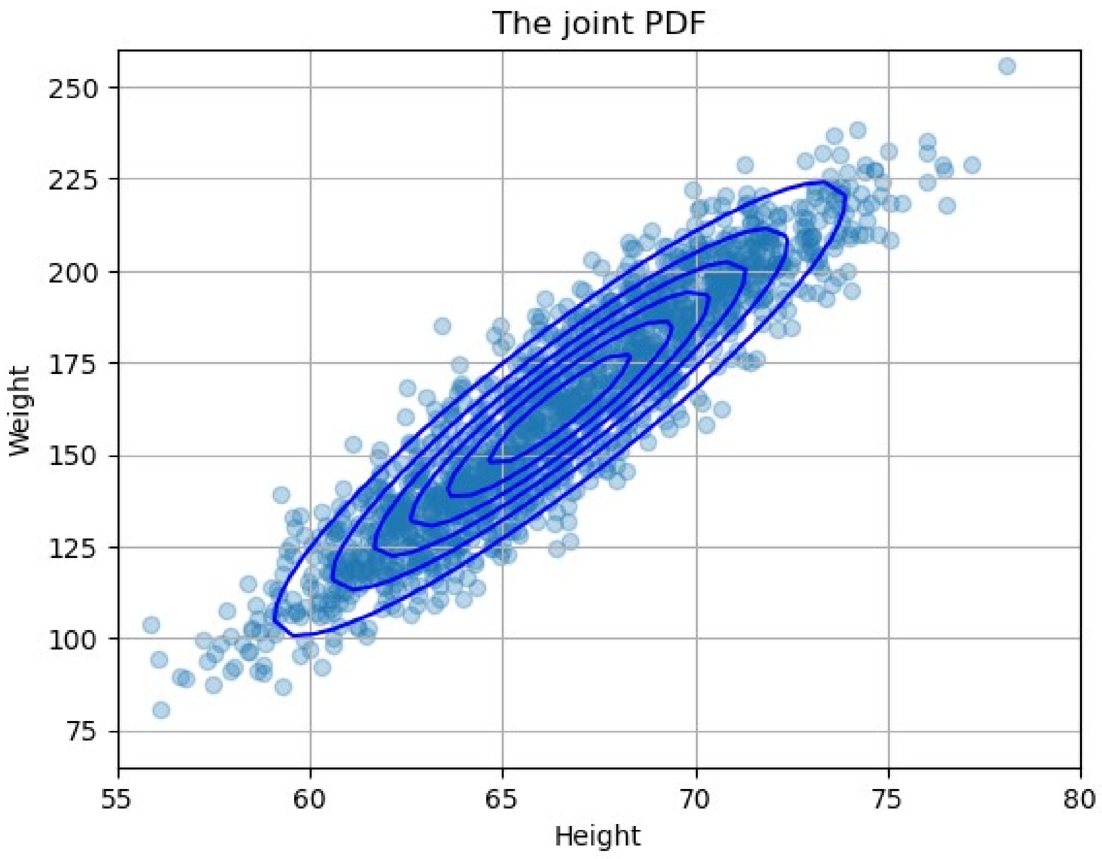
```
# Compute the mean vector, covariance matrix, and the correlation
coefficient matrix
m = train_data[['Height','Weight']].mean()
C = train_data[['Height','Weight']].cov()
rho = train_data[['Height','Weight']].corr(method = 'pearson')

print('The mean vector of height and weight is\n')
print(m)

print('\nThe covariance matrix of height and weight is\n')
print(C)

print('\nThe correlation coefficient matrix of height and weight is\
n')
print(rho)
```

```
The mean vector of height and weight is

Height     66.493410
Weight    162.367104
dtype: float64

The covariance matrix of height and weight is

            Height      Weight
Height   14.367088  109.919064
Weight  109.919064  994.581994

The correlation coefficient matrix of height and weight is

          Height    Weight
Height  1.000000  0.919535
Weight  0.919535  1.000000
```

The code below overlays a contour plot of the 2D Gaussian that best fits the data onto the scatter plot.

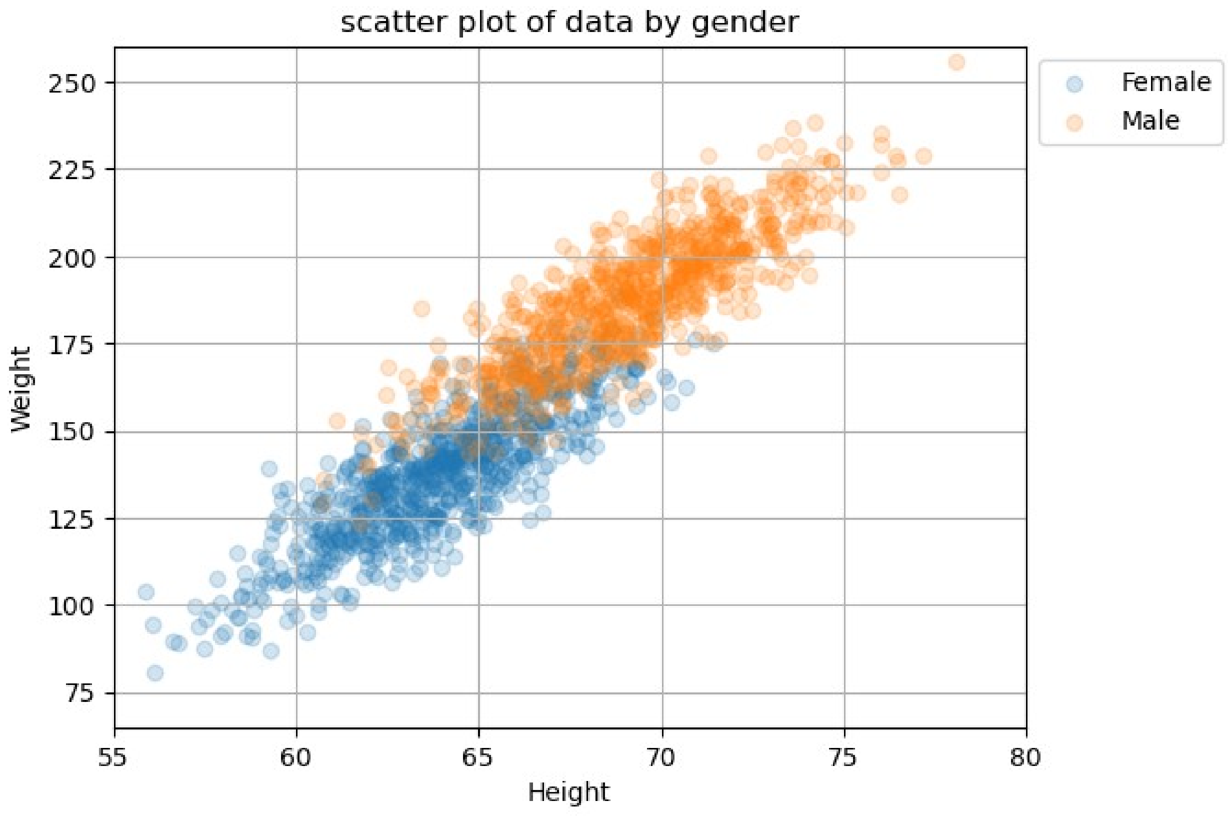
```
# create a multivariate Gaussian random variable with mean m and
covariance matrix C
dist = multivariate_normal(m, C)

# create limits of plot
axlim = [55, 80, 65, 260]

# create set of (x,y) positions to compute the pdf at
x, y = np.meshgrid( np.linspace(axlim[0], axlim[1]),
np.linspace(axlim[2], axlim[3]) )
pos = np.dstack((x, y))

# compute the joint Gaussian pdf
f_XY = dist.pdf(pos)

# create the plot
fig, ax = plt.subplots()

# scatter plot
ax.scatter(train_data['Height'], train_data['Weight'], alpha=0.3)

# pdf contour plot
ax.contour(x, y, f_XY, colors='b', alpha = 1)

# set axes and labels
ax.axis(axlim)
plt.xlabel('Height')
plt.ylabel('Weight')
plt.grid(True)
```

```
ax.set_title('The joint PDF')

plt.show()
```



Now, we generate the scatter plot of height and weight for each gender separately. We separate data into different groups using the **pandas.DataFrame.groupby** command.

```
# create figure
fig, ax = plt.subplots()

# create one scatter plot for each gender
for name, groups in train_data.groupby('Gender'):
    ax.scatter(x='Height', y='Weight', data = groups, label = name,
alpha = 0.2)

# set axes and labels
ax.axis(axlim)
ax.legend(loc = 'upper left', bbox_to_anchor = (1, 1))
plt.xlabel('Height')
plt.ylabel('Weight')
plt.grid(True)
plt.title('scatter plot of data by gender')
plt.show()
```



We can see from the plot above that males are generally taller and heavier than females.

Thus, to obtain a simple automatic classifier, we might consider applying a threshold to either the height or the weight. In other words, a person is classified as male if his/her height or weight is larger than a threshold, T, otherwise, the person is classified as a female.

The code below uses the function **threshold_classifier** defined above to implement and test a classifier on the height. This corresponds to drawing a vertical line on the scatter plot at the threshold and classifying everything on the right as male and everything on the left as female. When we choose the threshold to be 70 inches, which is quite tall, most of the females are correctly classified, but quite a few males are classified as female. The code also shows some of the classification results for males and females.

We use the function **compare_labels** to evaluate the performance of this classifier on the test data by comparing the output of the classifier with the ground truth labels.

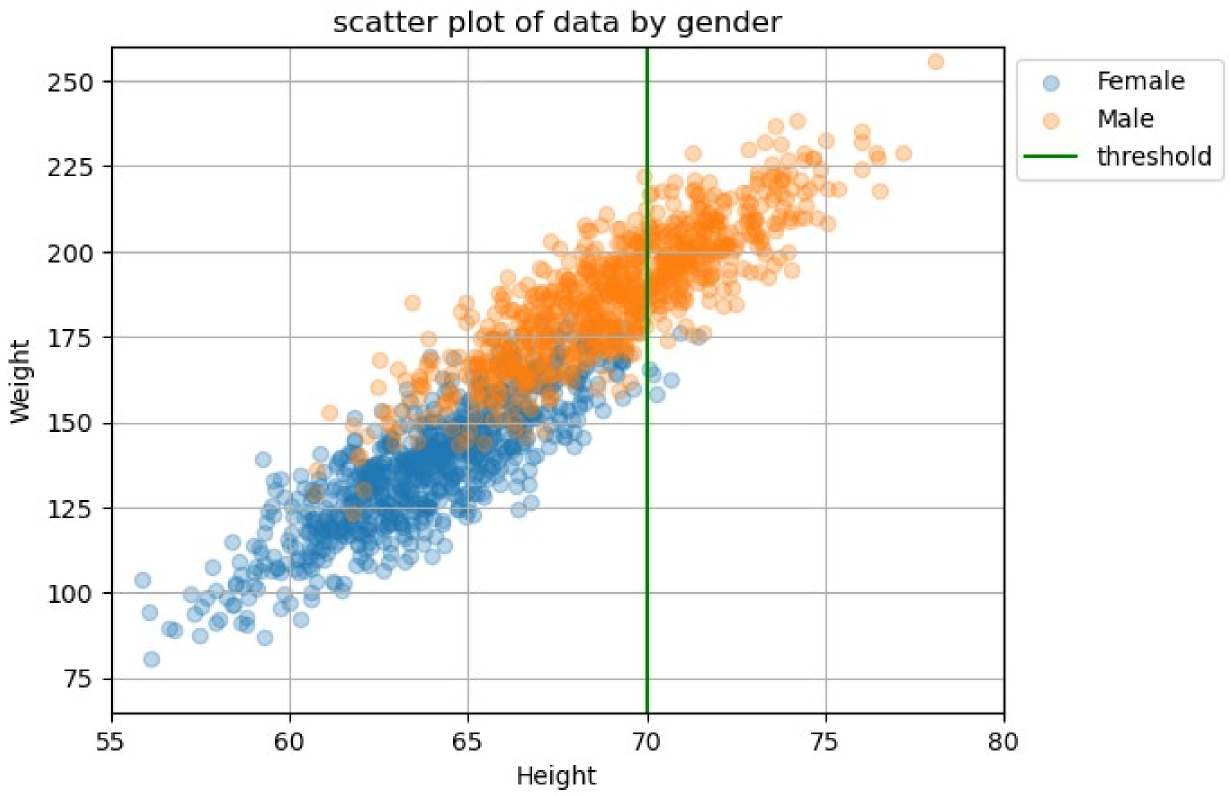
The accuracy is the ratio of the correct classifications to the size of the test dataset.

You can try changing the value of the threshold, e.g. to 63, to see what happens to the errors.

Note that thresholding by weight corresponds to drawing a horizontal, rather than vertical line at the threshold. Given the shape of the scatter plot, you might be able to guess which classifier will perform better: one based on thresholding by height or one based on thresholding by weight.

```
# get the ground truth labels
ground_truth = test_data['Gender'].values.tolist()

# run the threshold classifier
threshold = 70 # choose a threshold for comparison
output = threshold_classifier(test_data['Height'], threshold)

accuracy = compare_labels(output, ground_truth)
print(f'The accuracy is {accuracy}')

# create figure
fig, ax = plt.subplots()

# create one scatter plot for each gender
for name, groups in train_data.groupby('Gender'):
    ax.scatter(x='Height', y='Weight', data = groups, label = name,
alpha = 0.3)
ax.plot([threshold, threshold],axlim[2:4],'g',label = 'threshold')

# set axes and labels
ax.axis(axlim)
ax.legend(loc = 'upper left', bbox_to_anchor = (1, 1))
plt.xlabel('Height')
plt.ylabel('Weight')
plt.grid(True)
plt.title('scatter plot of data by gender')

plt.show()

d = {'ground_truth': ground_truth, 'height': test_data['Height'],
'classifier output': output}
classifier_compare = pd.DataFrame(data = d)
n_print = 8
print(classifier_compare.head(n_print))
print(classifier_compare.tail(n_print))
```

```
The accuracy is 0.65
```



```
  ground_truth     height classifier output
0         Male  73.847017              Male
1         Male  68.781904            Female
2         Male  74.110105              Male
3         Male  71.730978              Male
4         Male  69.881796            Female
5         Male  67.253016            Female
6         Male  68.785081            Female
7         Male  68.348516            Female
     ground_truth     height classifier output
5392       Female  63.642045            Female
5393       Female  69.825826            Female
5394       Female  60.162510            Female
5395       Female  62.285114            Female
5396       Female  64.328491            Female
5397       Female  62.518234            Female
5398       Female  62.892831            Female
5399       Female  62.396154            Female
```

Part a (15 points) : Choosing the threshold

In the code above, the threshold was chosen arbitrarily. Here, you will use probability theory to choose the threshold in a more principled manner based on data in the training set.

To do this, we note that the probability that this classifier makes an error is given by

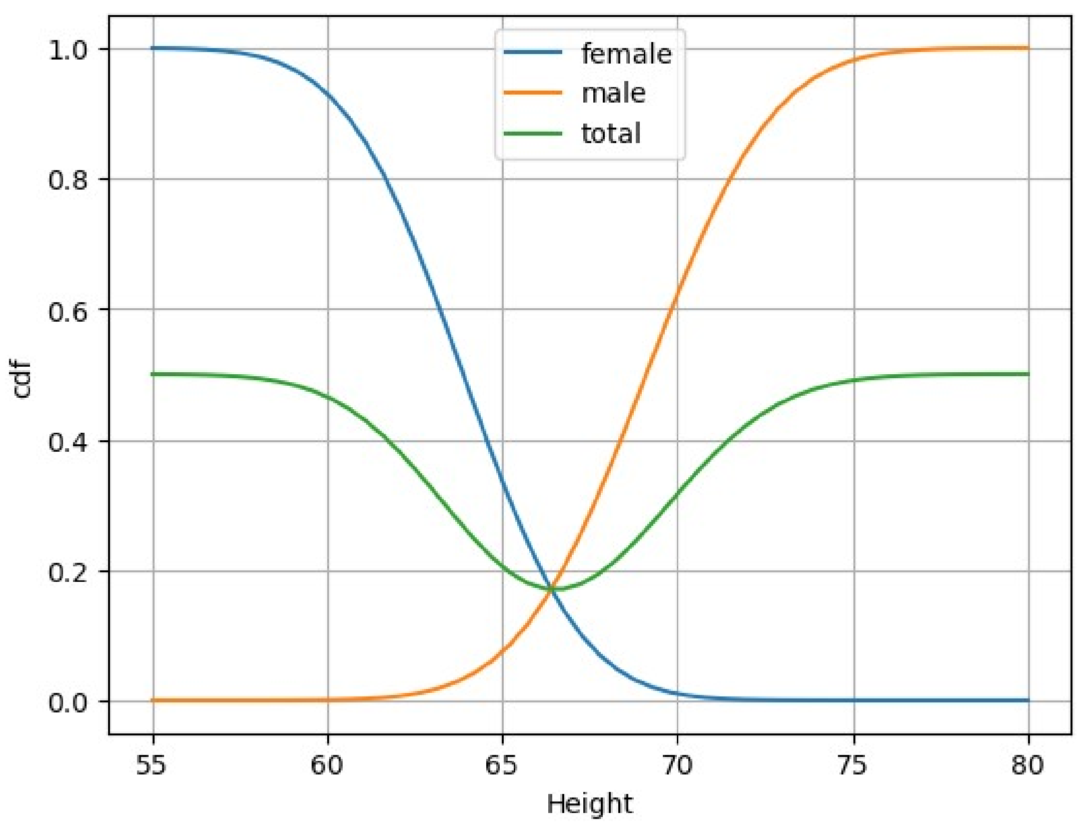
$\mathrm{P}[\mathrm{Female and } X>T] + \mathrm{P}[\mathrm{Male and } X \leq T] $ $ = \mathrm{P} [X > T | \mathrm{Female}] \mathrm{P} [\mathrm{Female}] + \mathrm{P} [X \leq T | \mathrm{Male}] \mathrm{P} [\mathrm{Male}] $

Compute the empirical conditional means and standard deviations of the height given the genders Female and Male on the training set.

Plot the probability that this classifier makes an error as a function of T for 50 points ranging from 55 to 80, under the assumption that the conditional probability density functions of the height X given the gender G, $f_{X \vee G}(x \vee g)$ and $f_{Y \vee G}(y \vee g)$ for $g \in \{Female, Male\}$, are given by Gaussian distributions with the conditional means and standard deviations computed empirically from the training dataset.

Find the optimal value of T (i.e., the value of the threshold at which classifier's error is minimum) and the corresponding probability that this classifier makes an error.

```python
x= np.linspace(55, 80, 50)
female= train_data[train_data['Gender']=="Female"]
male= train_data[train_data['Gender']=="Male"]
female_p= len(female)/len(train_data)
male_p= len(male)/len(train_data)
n_cdf1 = scipy.stats.norm.cdf(x, loc = male['Height'].mean(), scale =
male['Height'].std())
n_cdf2 = scipy.stats.norm.sf(x, loc =  female['Height'].mean(), scale
= female['Height'].std())
total_cdf = male_p * n_cdf1 + female_p * n_cdf2
plt.plot(x, n_cdf2, label = 'female')
plt.plot(x, n_cdf1, label = 'male')
plt.plot(x, total_cdf, label = 'total')
plt.legend()

plt.grid()
plt.ylabel('cdf')
plt.xlabel('Height')

plt.show()

indx= np.argmin(total_cdf)
p_error= total_cdf[indx]
print(f'The minimum probability of error is {p_error} at height
{x[indx]}')
```



```
The minimum probability of error is 0.17068029550137023 at height
66.73469387755102
```

Part b (7 points)

Use the function **threshold_classifier** defined above to implement and test this new classifier on the height test data. Compute its empirical classficiation accuracy using **compare_labels** (defined above). In other words, compare the classification results with the ground truth, count the number of correct classfication and divided by the number of people.

```
# put your code below and print out the result after running your code
#
=====================================================================
height_thres= x[indx]
ground_truth = test_data['Gender'].values.tolist()


output = threshold_classifier(test_data['Height'], height_thres)

accuracy = compare_labels(output, ground_truth)
print(f'The accuracy is {accuracy}')

The accuracy is 0.8375
```

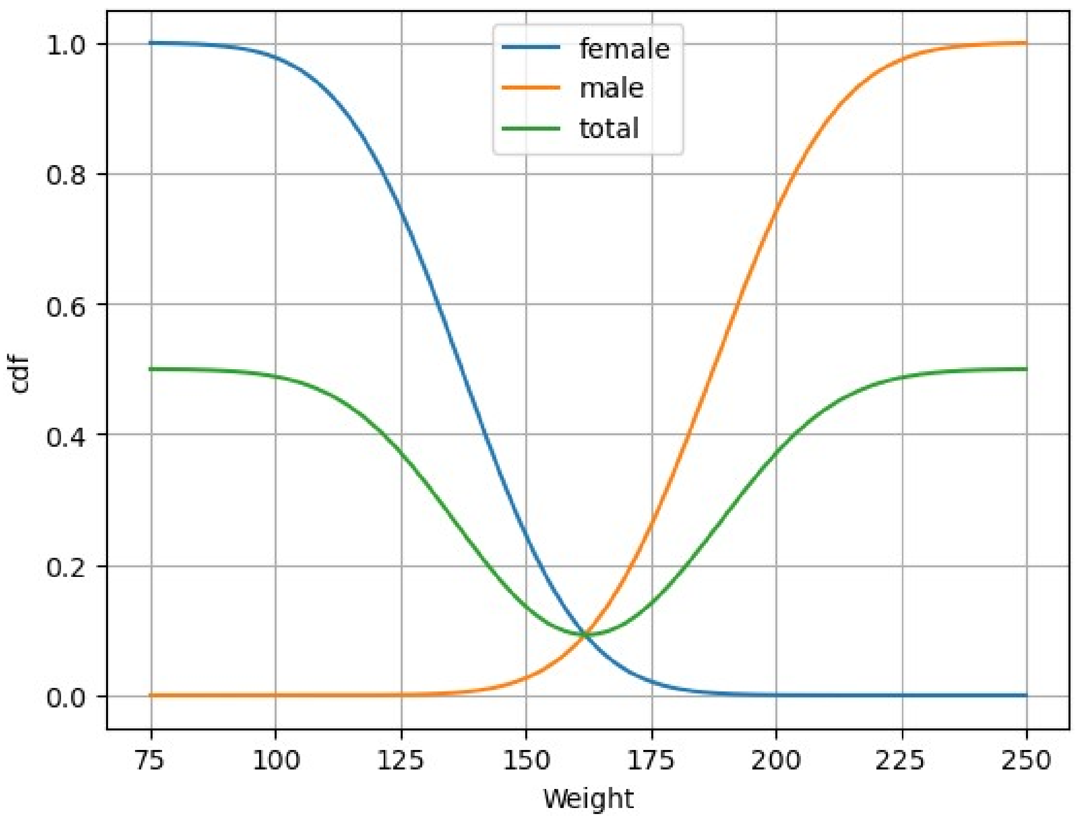
Question : Is the accuracy of this classifier better than that obtained using an arbitrary threshold = 70?

Answer: Yes

Part c (20 points) :

Repeat the steps above (**Parts a and b**) for designing a classifier based on weight. Consider thresholds ranging from 75 to 250 pounds.

```
# put your code below and print out the result after running your code
#
======================================================================

x= np.arange(75, 250, 5e-2)

n_cdf1 = scipy.stats.norm.cdf(x, loc = male['Weight'].mean(), scale =
male['Weight'].std())
n_cdf2 = scipy.stats.norm.sf(x, loc =  female['Weight'].mean(), scale
= female['Weight'].std())
total_cdf = male_p * n_cdf1 + female_p * n_cdf2
plt.plot(x, n_cdf2, label = 'female')
plt.plot(x, n_cdf1, label = 'male')
plt.plot(x, total_cdf, label = 'total')
plt.legend()

plt.grid()
plt.ylabel('cdf')
plt.xlabel('Weight')

plt.show()

indx= np.argmin(total_cdf)
p_error= total_cdf[indx]
print(f'The minimum probability of error is {p_error} at weight
{x[indx]}')
print(f"the accuracy is {1- p_error}")
```



```
The minimum probability of error is 0.0928753940150826 at weight
162.14999999999503
the accuracy is 0.9071246059849174
```

Question : Which feature (height or weight) results in the better performing classifier? And why?

Answer: Weight. The minimum error achieveable by the weight classifier (0.0929) is smaller than that of height (0.171).