<u>ReadMe</u>

1.) Read the whole file.

2.) Total 3 types of distributions were used to model data distribution within the dataset. 1. Histograms, 2. Uni-axial Gaussian Model, and 3. Uni-axial Gaussian Mixture models.

3.) File 'Results' has all parameters values with distribution graphs.

4.) Datasets used. 1. Iris Data set, 2. Indian Liver Patient Dataset

5.) Note: My major is civil engineering and this is the first time I used python. So files are hard coded specifically, categorical distribution files. Gaussian model and Gaussian mixture model files can be used for other datasets by changing the input variables as explained below.

<u>Categorical Files:</u> File name starts with 'Cat_'.

<u>Step 1</u>: Import Libraries and read excel file. Save column data in variables.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_excel('Indian Liver Patient Dataset.xlsx')
data = data.to_numpy()
classdata = data[:,10]
```

<u>Step 2</u>: Create variables corresponding to data type. Loop through all data and count the no. of occurrence of that data type.

```
# Create variables corresponding to class type in data
a = 0 # for Male
b = 0 # for Female

for x in classdata:
    if x == 'Male':
        a = a+1
    else:
        b = b+1
```

<u>Step 3</u>:Sum the total no. of data/events occurrences. Find probability by dividing data type to total no. of data/event occurrence.

```
# Total weight to find probability/Normalization
sum = a + b

# Probaility of getting Male,Female
pa = a/sum #Male
pb = b/sum #Female
```

Step 4: Print the probability of data type occurrence and plot the data to visualize distributions.

```
# Print the probility of data type
print('Probaility of Male =', pa)
print('Probaility of Female =', pb)

# Plotting categorical Data to visualize
xaxis = ['Male','Female']
yaxis = [pa,pb]
plt.title('Categorical Distribution of Data')
plt.xlabel('Gender Type')
plt.ylabel('Probability')
plt.bar(xaxis,yaxis)
plt.show()
```

Gaussian Files: File name 'Gaussian.py'

Step 1: Import Libraries and read excel file. Save column data in variable. To run this file enter excel file name containing data in red rectangle box. Type the no. of column in green box to find Gaussian distribution of that data.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_excel('Indian Liver Patient Dataset.xlsx')
data = data.to_numpy()
sepal_length = data[:,9]
```

Step 2: Create bins of the data to display the frequency, find out mean and standard deviation of the data and print these parameters.

```
# bins and histogram
count, bins, ignored = plt.hist(sepal_length,15, density=True)
print(count)
print(bins)

#Calculate mean and std
mean = np.mean(sepal_length)
sd = np.std(sepal_length)
print('Mean of this data =',mean)
print('Standard Deviation of this data =',sd)
```

Step 3: Create function to find out the Gaussian probability density of the data point and apply function to data.

```
# Creating a Function
def normal_dist(bins, mean, sd):
    #prob_density = (np.pi*sd) * np.exp(-0.5*((data_age-mean)/sd)**2)
    prob_density = 1/(sd*np.sqrt(2*np.pi)) * np.exp(-(bins.astype(float) - mean)**2/(2*sd**2))
    return prob_density

#Apply function to data.
pdf = normal_dist(bins,mean,sd)
```

Step 4: Plot the results.

```
#Plotting the results
plt.plot(bins,pdf, color = 'red')
plt.xlabel(xlabel)
plt.ylabel('Probability Density')
plt.title(Title)
plt.show()
```

Gaussian Mixture Files: File name 'GaussianMixture.py' (Code gives different parameters each time)

Step 1: Import Libraries and read excel file. Save column data in variable. To run this file enter excel file name containing data in red rectangle box. Type the no. of column in green box to find Gaussian distribution of that data.

```
import numpy as np
import pandas as pd
import math
import matplotlib.pyplot as plt

# Read the data
data = pd.read_excel('iris.xlsx')
data = data.to_numpy()
sepal_length = data[:,2]
```

Step 2: Input no. of classes, and iteration no.

```
# Input: no. of classes, no. of iterations
k = 2
iter = 150
```

Step 3: Divide the data into bins

```
# Calculate Mean and Standard Deviation
count, bins, ignored = plt.hist(sepal_length, 20, density=True)
```

Step 4: Initialize some variables in E-step and M-step

```
# Create Pi,mu, and sigma
mu = np.random.randint(5, size=(k,1))
sigma = np.random.randint(5, size=(k,1))
pi = np.random.dirichlet(np.ones(k),size=1).transpose()

Gamma = np.zeros((len(bins),k))
pdf = np.zeros((len(bins),k))
xGamma = np.zeros((len(bins),k))
xsigma = np.zeros((len(bins),k))
```

mu (mean) = $[\quad]_{k\times 1}$

k = no. of classes

1 → one variable

sigma (variance) = $[\quad]_{k\times 1}$

k = no. of classes

1 → one variable

Pi (weightage) = $[\quad]_{k\times 1}$

k = no. of classes

1 → one variable

2. **E step**. Evaluate the responsibilities using the current parameter values

$$\boxed{\gamma(z_{nk})} = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}. \qquad (9.23)$$

Gamma = $[\quad]_{n\times k}$

n → no of data points

k → no of classes

PDF = $[\quad]_{n\times k}$

↓

Probability Density Function

n → no of data points

k → no of classes

Step 4: Create function to find out the Gaussian probability density of the data point.

```python
# Creating a Function for calculating gaussian distribution probability
def normal_dist(data, mean, sd):
    prob_density = 1/(sd*np.sqrt(2*np.pi)) * np.exp(-(data.astype(float) - mean)**2/(2*sd**2))
    return prob_density
```

Step 5: E-Step

```python
# E STEP CALCULATION
def estep(mu,sigma,pi):
    # Calculate PDF
    for x in range(k):
        pdf[:,x] = normal_dist(bins,mu[x],sigma[x])

    # Calculate pipdf2
    pipdf2 = pdf.copy()
    for x in range(k):
        pipdf2[:,x] = pi[x] * pipdf2[:,x]
    pipdf2 = pipdf2.sum(axis=1)

    # Calculate pipdf
    pipdf = pdf.copy()
    for x in range(k):
        pipdf[:,x] = pi[x] * pdf[:,x]

    # Calculate Gamma
    for x in range(k):
        Gamma[:,x] = pipdf[:,x] / pipdf2[:]

    return Gamma
```

$$\boxed{\gamma(z_{nk})} = \frac{\boxed{\pi_k \boxed{\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}}}{\boxed{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}}.$$

Step 5: M-Step

```python
# M STEP CALCULATION
def mstep(resposibility):
    xGamma = np.zeros((len(bins),k))
    xsigma = np.zeros((len(bins),k))
    # Calculate NK
    NK = Gamma.sum(axis=0)

    # Calculate new mu
    for x in range(k):
        xGamma[:,x] = bins * Gamma[:,x]
    xGamma = xGamma.sum(axis=0)
    munew = xGamma / NK

    # Calculate Sigma
    for x in range(k):
        xsigma[:,x] = Gamma[:,x] * (bins-munew[x])* (bins-munew[x])
    xsigma = xsigma.sum(axis=0)
    sigmanew = np.sqrt(xsigma / NK)

    # Calculate pinew
    pinew = NK / len(bins)

    return munew,sigmanew,pinew
```

3. **M step**. Re-estimate the parameters using the current responsibilities

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \tag{9.24}$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right)^{\mathrm{T}} \tag{9.25}$$

$$\pi_k^{\text{new}} = \frac{N_k}{N} \tag{9.26}$$

where

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}). \tag{9.27}$$

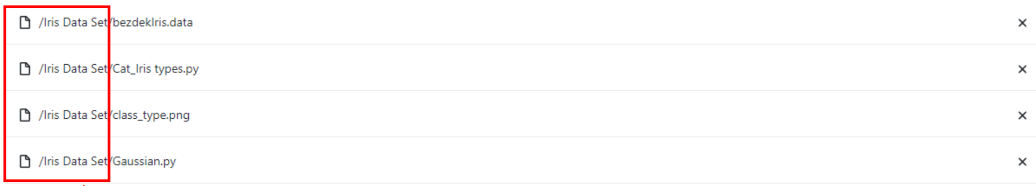
Step 6: Iterations. (input value)

```
for x in range(iter):
    responsibility = estep(mu,sigma,pi)
    mu,sigma,pi = mstep(responsibility)

```

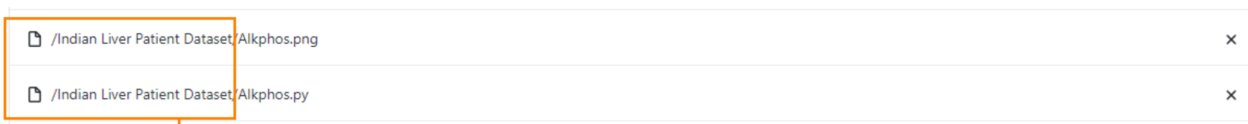
Step 7: Calculate final PDFs, print parameters, and plot the distribution. PDF need to be multiplied by weightages.

```
pdf1 = normal_dist(bins,mu[0],math.sqrt(sigma[0]))
pdf2 = normal_dist(bins,mu[1],math.sqrt(sigma[1]))

print('mu =', mu)
print('sigma =', sigma)
print('pi =', pi)


#Plotting the results
plt.plot(bins,pi[0]*pdf1, color = 'red')
plt.plot(bins,pi[1]*pdf2, color = 'red')
plt.xlabel('Age')
plt.ylabel('Probability Density')
plt.show()
```

Files on Github



These files are for
Iris data set



These files are for
Indian Liver Patient
Dataset