Assignment 1

Problem 1

There will be 4 components, Since probabilities needs to summed to 1,which is necessity of probability so we have add other component that will be 1 - 0.3 - 0.4 - 0.1 = 0.2

Problem 2

Let f_1, f_2, f_3 and f_4 are the density function of bivariate gaussian distribution and π_1, π_2, π_3 and π_4 are the mixing probabilities such that they sum up to be 1 and $\theta_i = (\mu_{i1}, \mu_{i2}, \sigma_{i2}, \sigma_{i3}, \rho_i)$, then the underlying density can be written as

$$f(x|\theta) = \sum_{i=1}^{i=4} \pi_i \cdot f_i(x|z_i, \theta_i)$$

Problem 3

```python
import random
import math
import matplotlib.pyplot as plt
import numpy as np

# Box-Muller Method

def std_norm_generator(size):
  variate = np.array([])
  for i in range(size):
    u1 = random.uniform(0, 1)
    u2 = random.uniform(0, 1)
    r = -2*math.log(u1)
    v = 2*math.pi*u2
    z1 = math.sqrt(r)*math.cos(v)
    z2 = math.sqrt(r)*math.sin(v)
    variate = np.append(variate , z1)
    variate = np.append(variate , z2)
  return variate

def mixture(norm_list):
  norm_variate = np.array([])

  # Taking First Gaussian with Variance = 2 , Mean = 5
  # Taking Second Gaussian as Variance = 3 , Mean = 13
  # Mixing Probability is 0.4 , 0.6

  for variate in norm_list:
    r = random.uniform(0, 1)
    if r < 0.4:
      norm_variate = np.append(norm_variate , math.sqrt(2)*variate+5)
    else:
      norm_variate = np.append(norm_variate , math.sqrt(3)*variate+13)
  return norm_variate
```

Estimation

```
# Parameter Estimation  using EM Algorithm

dnorm = lambda  x , mu ,sigma : (1/(sigma*np.sqrt(2*math.pi)))*np.exp(-0.5*((x-mu)/sigma)**2)

# Estimation

def Estimate(variate,initial = np.array([7,8,6,4,0.8])):


  # Expectation Step


  e = (initial[4]*dnorm(variate , initial[2], initial[3]))/(((1-initial[4])*dnorm(variate , initial[0], initial[1]))+        (initial[4]*dnorm(variate , initial[2], initial[3])))
  mu1 = np.sum((1-e)*variate)/np.sum(1-e)
  sigma1 = np.sqrt(np.sum((1-e)*((variate - mu1)**2))/np.sum(1-e))
  mu2 = np.sum(e*variate)/np.sum(e)
  sigma2 = np.sqrt(np.sum((e)*((variate - mu2)**2))/np.sum(e))
  pi = np.sum(e)/len(variate)

  return np.array([mu1 , sigma1 , mu2 , sigma2 , pi])


intiat = Estimate(mixture_distribution)
for i in range(1000):
  intiat = Estimate(mixture_distribution , intiat)

print(intiat)
```

Problem 4

```
# Importing Libraries

import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
from sklearn.datasets import load_iris, load_breast_cancer, load_wine


def get_class_levels(data):
    return len(np.unique(data['target']))
# IRIS Dataset

iris = load_iris(as_frame=True)
class_levels = get_class_levels(iris)
kmeans = KMeans(n_clusters=class_levels, random_state=0).fit(iris['data'].to_numpy())
gmm = GaussianMixture(n_components=class_levels).fit(iris['data'].to_numpy())
print(gmm.means_, gmm.weights_)

# Wine Dataset
wine = load_wine(as_frame=True)
class_levels = get_class_levels(wine)
```

```python
kmeans = KMeans(n_clusters=class_levels, random_state=0).fit(wine['data'].to_numpy())
gmm = GaussianMixture(n_components=class_levels).fit(wine['data'].to_numpy())
print(gmm.means_, gmm.weights_)

# Breast Cancer Dataset

breast_cancer = load_breast_cancer(as_frame=True)
class_levels = get_class_levels(breast_cancer)
kmeans = KMeans(n_clusters=class_levels, random_state=0).fit(breast_cancer['data'].to_numpy())
gmm = GaussianMixture(n_components=class_levels).fit(breast_cancer['data'].to_numpy())
print(gmm.means_, gmm.weights_)

```