# INTELLIGENT SYSTEMS LAB-10 (01/11/2021)

*__NAME__: Harshvardhan Agarwal | __SEC__: A | __REG NO__.: 201800524*

## PROBLEM STATEMENT

Write your own GMM implementation, using the EM algorithm for parameter learning. Learn a GMM with 10 components on your data in PCA space.

1. Submit pdf file with code and output
2. Try for different test cases

## PROBLEM SOLUTION

## SOURCE CODE AND OUTPUT

**Dataset Used:** Iris Dataset

In [3]:
```
df = pd.DataFrame(iris.data,columns=iris.feature_names)
df.head()
```

Out[3]:

| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) |
|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 |

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pandas import DataFrame
from sklearn import datasets
from sklearn.mixture import GaussianMixture

# Load the Iris dataset

iris = datasets.load_iris()

# Select the first two columns

X = iris.data[:, :2]
```

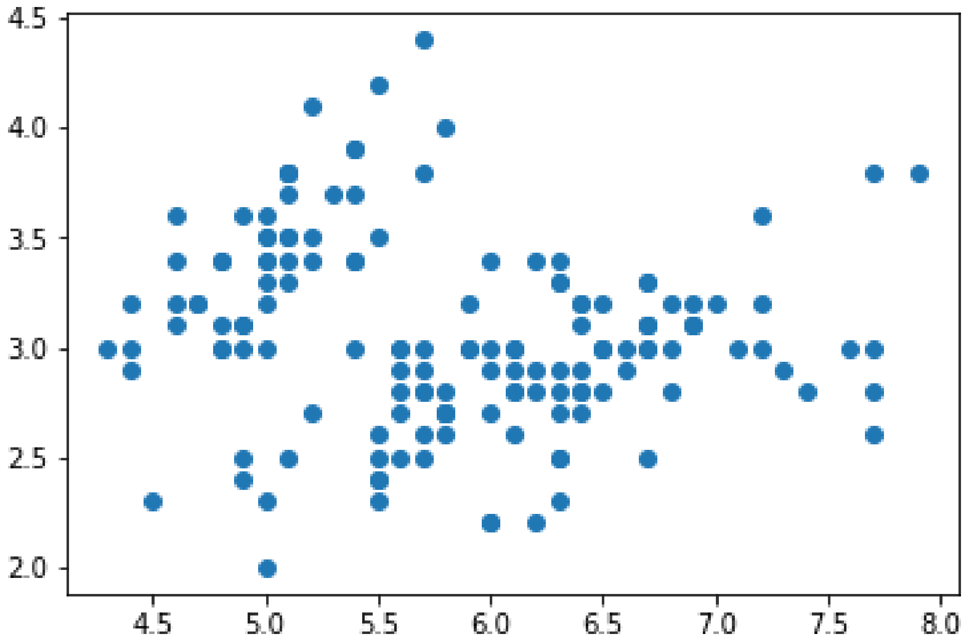
**# Turn it into a dataframe**

d = pd.DataFrame(X)

d11 = pd.DataFrame(X)

**# Plot the data**

plt.scatter(d[0], d[1])




gmm = GaussianMixture(n_components = 10)

gmm1 = GaussianMixture(n_components = 20)

**# Fit the GMM model for the dataset which expresses the dataset as a mixture of 3 Gaussian Distribution**

gmm.fit(d)

```
In [15]: gmm.fit(d)
Out[15]: GaussianMixture(n_components=10)
```

gmm1.fit(d11)

```
In [16]: gmm1.fit(d11)
Out[16]: GaussianMixture(n_components=20)
```

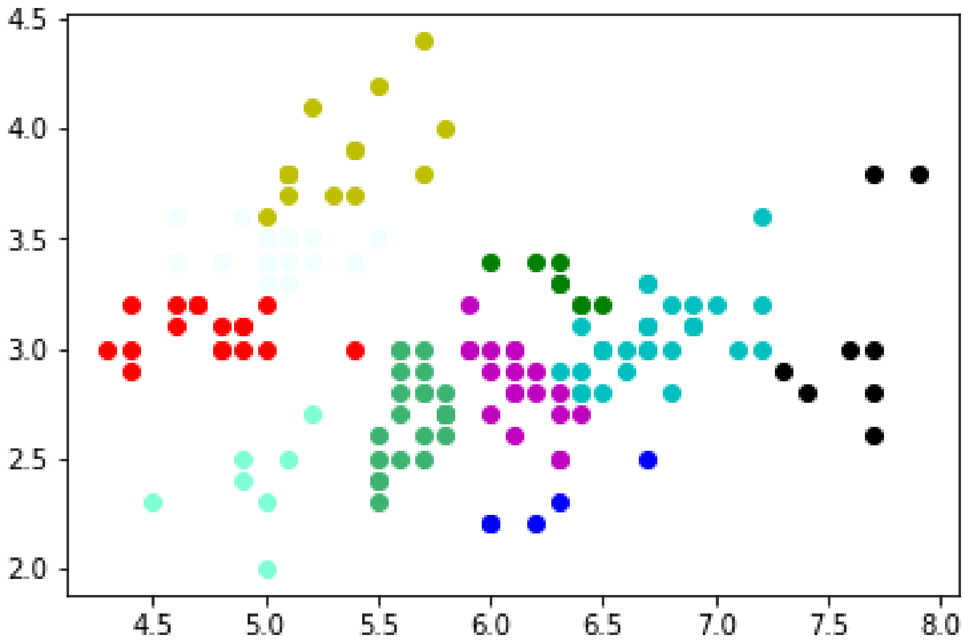
**# Assign a label to each sample**

```
labels = gmm.predict(d)

labels1 = gmm1.predict(d11)

d['Labels'] = labels
d0 = d[d['Labels'] == 0]
d1 = d[d['Labels'] == 1]
d2 = d[d['Labels'] == 2]
d3 = d[d['Labels'] == 3]
d4 = d[d['Labels'] == 4]
d5 = d[d['Labels'] == 5]
d6 = d[d['Labels'] == 6]
d7 = d[d['Labels'] == 7]
d8 = d[d['Labels'] == 8]
d9 = d[d['Labels'] == 9]

d11['Labels1'] = labels1
d01 = d11[d11['Labels1'] == 0]
d1111 = d11[d11['Labels1'] == 1]
d21 = d11[d11['Labels1'] == 2]
d31 = d11[d11['Labels1'] == 3]
d41 = d11[d11['Labels1'] == 4]
d51 = d11[d11['Labels1'] == 5]
d61 = d11[d11['Labels1'] == 6]
d71 = d11[d11['Labels1'] == 7]
d81 = d11[d11['Labels1'] == 8]
d91 = d11[d11['Labels1'] == 9]
d101 = d11[d11['Labels1'] == 10]
d111 = d11[d11['Labels1'] == 11]
d121 = d11[d11['Labels1'] == 12]
d131 = d11[d11['Labels1'] == 13]
d141 = d11[d11['Labels1'] == 14]
d151 = d11[d11['Labels1'] == 15]
d161 = d11[d11['Labels1'] == 16]
d171 = d11[d11['Labels1'] == 17]
d181 = d11[d11['Labels1'] == 18]
d191 = d11[d11['Labels1'] == 19]
```

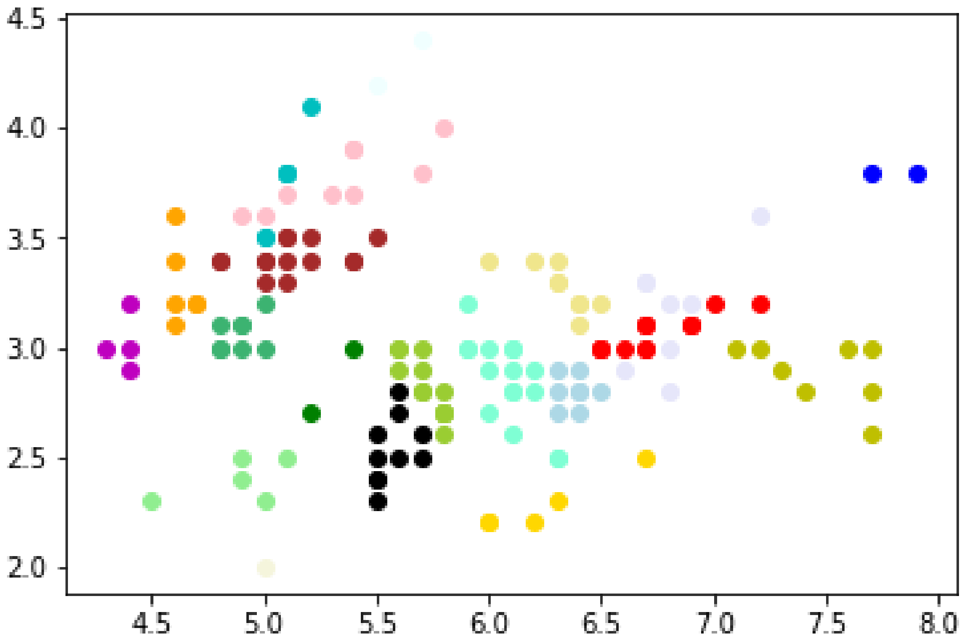
**# Plot the clusters in the same plot**

```
plt.scatter(d0[0], d0[1], c = 'b')
plt.scatter(d1[0], d1[1], c = 'g')
plt.scatter(d2[0], d2[1], c = 'r')
plt.scatter(d3[0], d3[1], c = 'c')
plt.scatter(d4[0], d4[1], c = 'm')
plt.scatter(d5[0], d5[1], c = 'y')
plt.scatter(d6[0], d6[1], c = 'k')
plt.scatter(d7[0], d7[1], c = 'aquamarine')
```

```
plt.scatter(d8[0], d8[1], c =  'mediumseagreen')
plt.scatter(d9[0], d9[1], c = 'azure')
```



```
plt.scatter(d01[0], d01[1], c = 'b')
plt.scatter(d1111[0], d1111[1], c = 'g')
plt.scatter(d21[0], d21[1], c = 'r')
plt.scatter(d31[0], d31[1], c = 'c')
plt.scatter(d41[0], d41[1], c = 'm')
plt.scatter(d51[0], d51[1], c = 'y')
plt.scatter(d61[0], d61[1], c = 'k')
plt.scatter(d71[0], d71[1], c = 'aquamarine')
plt.scatter(d81[0], d81[1], c = 'mediumseagreen')
plt.scatter(d91[0], d91[1], c = 'azure')
plt.scatter(d101[0], d101[1], c = 'beige')
plt.scatter(d111[0], d111[1], c = 'brown')
plt.scatter(d121[0], d121[1], c = 'gold')
plt.scatter(d131[0], d131[1], c = 'khaki')
plt.scatter(d141[0], d141[1], c = 'lavender')
plt.scatter(d151[0], d151[1], c = 'lightblue')
plt.scatter(d161[0], d161[1], c = 'lightgreen')
plt.scatter(d171[0], d171[1], c = 'orange')
plt.scatter(d181[0], d181[1], c = 'pink')
plt.scatter(d191[0], d191[1], c = 'yellowgreen')
```




**# Print the converged log-likelihood value**

print(gmm.lower_bound_)

```
In [26]: print(gmm.lower_bound_)
         -1.2514526602513558
```

print(gmm1.lower_bound_)

```
In [27]: print(gmm1.lower_bound_)
         -0.6094935972601598
```

**# Print the number of iterations needed for the log-likelihood value to converge**

print(gmm.n_iter_)

```
In [29]: print(gmm.n_iter_)
         13
```

print(gmm1.n_iter_)

```
In [30]: print(gmm1.n_iter_)
         21
```