Problem 4

In [1]:

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

In [2]:

```
data_spam = pd.read_csv('spambase.data', header = None)
data_spam.rename(columns = {57 : 'spam'}, inplace = True)
```

In [3]:

```
data_spam
```

Out[3]:

	0	1	2	3	4	5	6	7	8	9	...	48	49	50	51	52	53	54	55	56	spam
0	0.00	0.64	0.64	0.0	0.32	0.00	0.00	0.00	0.00	0.00	...	0.000	0.000	0.0	0.778	0.000	0.000	3.756	61	278	1
1	0.21	0.28	0.50	0.0	0.14	0.28	0.21	0.07	0.00	0.94	...	0.000	0.132	0.0	0.372	0.180	0.048	5.114	101	1028	1
2	0.06	0.00	0.71	0.0	1.23	0.19	0.19	0.12	0.64	0.25	...	0.010	0.143	0.0	0.276	0.184	0.010	9.821	485	2259	1
3	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31	0.63	...	0.000	0.137	0.0	0.137	0.000	0.000	3.537	40	191	1
4	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31	0.63	...	0.000	0.135	0.0	0.135	0.000	0.000	3.537	40	191	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4596	0.31	0.00	0.62	0.0	0.00	0.31	0.00	0.00	0.00	0.00	...	0.000	0.232	0.0	0.000	0.000	0.000	1.142	3	88	0
4597	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00	0.00	...	0.000	0.000	0.0	0.353	0.000	0.000	1.555	4	14	0
4598	0.30	0.00	0.30	0.0	0.00	0.00	0.00	0.00	0.00	0.00	...	0.102	0.718	0.0	0.000	0.000	0.000	1.404	6	118	0
4599	0.96	0.00	0.00	0.0	0.32	0.00	0.00	0.00	0.00	0.00	...	0.000	0.057	0.0	0.000	0.000	0.000	1.147	5	78	0
4600	0.00	0.00	0.65	0.0	0.00	0.00	0.00	0.00	0.00	0.00	...	0.000	0.000	0.0	0.125	0.000	0.000	1.250	5	40	0

4601 rows × 58 columns

In [4]:

```
X = data_spam.drop(['spam'], axis = 1)
y = data_spam['spam']
```

In [5]:

```
X
```

Out[5]:

	0	1	2	3	4	5	6	7	8	9	...	47	48	49	50	51	52	53	54	55	56
0	0.00	0.64	0.64	0.0	0.32	0.00	0.00	0.00	0.00	0.00	...	0.0	0.000	0.000	0.0	0.778	0.000	0.000	3.756	61	278
1	0.21	0.28	0.50	0.0	0.14	0.28	0.21	0.07	0.00	0.94	...	0.0	0.000	0.132	0.0	0.372	0.180	0.048	5.114	101	1028
2	0.06	0.00	0.71	0.0	1.23	0.19	0.19	0.12	0.64	0.25	...	0.0	0.010	0.143	0.0	0.276	0.184	0.010	9.821	485	2259
3	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31	0.63	...	0.0	0.000	0.137	0.0	0.137	0.000	0.000	3.537	40	191
4	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31	0.63	...	0.0	0.000	0.135	0.0	0.135	0.000	0.000	3.537	40	191
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4596	0.31	0.00	0.62	0.0	0.00	0.31	0.00	0.00	0.00	0.00	...	0.0	0.000	0.232	0.0	0.000	0.000	0.000	1.142	3	88
4597	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00	0.00	...	0.0	0.000	0.000	0.0	0.353	0.000	0.000	1.555	4	14
4598	0.30	0.00	0.30	0.0	0.00	0.00	0.00	0.00	0.00	0.00	...	0.0	0.102	0.718	0.0	0.000	0.000	0.000	1.404	6	118
4599	0.96	0.00	0.00	0.0	0.32	0.00	0.00	0.00	0.00	0.00	...	0.0	0.000	0.057	0.0	0.000	0.000	0.000	1.147	5	78
4600	0.00	0.00	0.65	0.0	0.00	0.00	0.00	0.00	0.00	0.00	...	0.0	0.000	0.000	0.0	0.125	0.000	0.000	1.250	5	40

4601 rows × 57 columns

In [6]:

```
y
```

Out[6]:

```
0       1
1       1
2       1
3       1
4       1
       ..
4596    0
4597    0
4598    0
4599    0
4600    0
Name: spam, Length: 4601, dtype: int64
```

In [7]:

```
# Scale the features, as the original values have wide ranges
X = StandardScaler().fit_transform(X)
```

In [8]:

```
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size = 0.2, stratify = y)
```

In [9]:

```
import numpy as np
from sklearn.mixture import GaussianMixture
from sklearn.metrics import accuracy_score

def train_gmm(X, y , n_components = 7):
    gmms = []
    classes = np.unique(y)
    for class_id in classes:
        class_data = X[y == class_id]
        gmm = GaussianMixture(n_components = 7)
        gmm.fit(class_data)
        gmms.append(gmm)
    return gmms

def predict_gmm(X, gmms):
    n_samples, _ = X.shape
    n_classes = len(gmms)
    posteriors = np.zeros((n_samples, n_classes))
    for class_id, gmm in enumerate(gmms):
        class_posteriors = gmm.score_samples(X)
        posteriors[:, class_id] = class_posteriors
    return np.argmax(posteriors, axis = 1)

def supervised_gmm(X_train, y_train, X_test, K = 7):
    gmms = train_gmm(X_train, y_train, n_components = 7)
    y_pred = predict_gmm(X_test, gmms)
    return y_pred

y_pred = supervised_gmm(X_train, y_train, X_test)
acc = accuracy_score(y_test, y_pred)
print("Accuracy:", acc)
```

```
Accuracy: 0.8870792616720955
```

In [10]:

```
import numpy as np
from sklearn.mixture import GaussianMixture
from sklearn.metrics import accuracy_score
from sklearn.datasets import fetch_openml
fashion = fetch_openml('Fashion-MNIST', version = 1)
X = fashion.data / 255.0
y = fashion.target.astype(int)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, stratify = y)


def train_gmm(X, y, n_components = 5):
    gmms = []
    classes = np.unique(y)
    for class_id in classes:
        class_data = X[y == class_id]
        gmm = GaussianMixture(n_components = 5)
        gmm.fit(class_data)
        gmms.append(gmm)
    return gmms

def predict_gmm(X, gmms):
    n_samples, _ = X.shape
    n_classes = len(gmms)
    posteriors = np.zeros((n_samples, n_classes))
    for class_id, gmm in enumerate(gmms):
        class_posteriors = gmm.score_samples(X)
        posteriors[:, class_id] = class_posteriors
    return np.argmax(posteriors, axis = 1)

def supervised_gmm(X_train, y_train, X_test, K = 5):
    gmms = train_gmm(X_train, y_train, n_components = 5)
    y_pred = predict_gmm(X_test, gmms)
    return y_pred

y_pred = supervised_gmm(X_train, y_train, X_test)
acc = accuracy_score(y_test, y_pred)
print("Accuracy:", acc)
```

```
Accuracy: 0.7590714285714286
```