Name :	**Dipangshu Roy**
Roll No :	**001811001014**
Department :	**Information Technology**
Class :	**4th Year 1st Sem**
Subject :	**Machine Learning Lab**

Assignment 3

Construct a machine learning based model for classification using Python for the following UCI datasets:

UCI datasets (can be loaded from the package itself):

1. Wine Dataset: https://archive.ics.uci.edu/ml/datasets/wine
2. Ionosphere Dataset: https://archive.ics.uci.edu/ml/datasets/Ionosphere
3. Wisconsin Breast Cancer Dataset : https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

> **Github Link of this Assignment's Python Scripts :** https://github.com/Droyder7/ML-Assignment-3

HMM Classifier

1. Wine Dataset

Accuracy Comparison Table :

Dataset	Classifier	With or Without Tuning	Train-Test Ratio	Accuracy
Wine Dataset	Gaussian HMM	Without Tuning	70-30	96
			60-40	**98**
			50-50	91
			40-60	88
			30-70	61
		With Tuning	70-30	64
			60-40	66
			50-50	58
			40-60	40
			30-70	36
	GMM HMM	Without Tuning	70-30	92
			60-40	31
			50-50	37
			40-60	31
			30-70	60
		With Tuning	70-30	64
			60-40	56
			50-50	38
			40-60	24
			30-70	31
	Multinomial HMM	Without Tuning	70-30	40
			60-40	31
			50-50	38
			40-60	36
			30-70	28
		With Tuning	70-30	42
			60-40	44
			50-50	38
			40-60	36
			30-70	36

From. this table we can see that Guassian HMM classifier without Tuning with 60-40 split has highest accuracy of **98** .

Code of this model :

```
import pandas as pd
import numpy as np

# Dataset Preparation
df = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-
databases/wine/wine.data",header=None)

df.columns = ['Class','Alcohol','Malic acid','Ash','Alcalinity of
ash','Magnesium','Total phenols','Flavanoids',
            'Nonflavanoid phenols','Proanthocyanins','Color
intensity','Hue','OD280/OD315 of diluted wines','Proline']


X = df.drop(['Class'], axis=1)
y = df['Class']

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.6,test_size=0.4)

# Feature Scaling
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Classification
from hmmlearn import hmm

classifier = hmm.GaussianHMM(n_components=3, covariance_type="full")
classifier.fit(X_train)

y_pred = classifier.predict(X_test)

size = len(y_pred)
strings = np.empty(size, np.unicode_)

for i in range (size):
    if y_pred[i] == 0:
      strings[i] = 1
    elif y_pred[i] == 1:
      strings[i] = 2
    else:
      strings[i] = 3

strings = strings.astype(np.int)

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print("Confusion Matrix:")
print(confusion_matrix(y_test, strings))

print("-------------------------------------------------")
print("-------------------------------------------------")
```

```
print("Performance Evaluation")
print(classification_report(y_test, strings))


print("-------------------------------------------------")
print("-------------------------------------------------")

print("Accuracy:")
print(accuracy_score(y_test, strings))

import matplotlib.pyplot as plt
import seaborn as sns
cm = confusion_matrix(y_test, strings)
sns.heatmap(cm, annot=True, fmt="d")
plt.show()
```

```
Confusion Matrix:
[[22  0  0]
 [ 0 32  1]
 [ 0  0 17]]
```

```
Performance Evaluation
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        22
           2       1.00      0.97      0.98        33
           3       0.94      1.00      0.97        17

    accuracy                           0.99        72
   macro avg       0.98      0.99      0.99        72
weighted avg       0.99      0.99      0.99        72
```

```
Accuracy:
0.9861111111111112
```

2. Ionosphere Dataset

Accuracy Comparison Table :

Ionosphere Dataset	**Gaussian HMM**	**Without Tuning**	**70-30**	**79**
			60-40	78
			50-50	76
			40-60	65
			30-70	63
		With Tuning	**70-30**	75
			60-40	68
			50-50	73
			40-60	71
			30-70	63
	GMM HMM	**Without Tuning**	**70-30**	29
			60-40	21
			50-50	34
			40-60	72
			30-70	39
		With Tuning	**70-30**	30
			60-40	26
			50-50	27
			40-60	23
			30-70	26
	Multinomial HMM	**Without Tuning**	**70-30**	45
			60-40	40
			50-50	60
			40-60	65
			30-70	60
		With Tuning	**70-30**	35
			60-40	32
			50-50	30
			40-60	26
			30-70	22

From. this table we can see that Guassian HMM classifier without Tuning with 70-30 split has highest accuracy of **79** .

Code of this model :

```
import pandas as pd
import numpy as np

# Dataset Preparation
df = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-
databases/ionosphere/ionosphere.data",header=None)


df.columns =
['1','2','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18','19'

,'20','21','22','23','24','25','26','27','28','29','30','31','32','33','34','Class']


X = df.drop(['Class'], axis=1)
y = df['Class']

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3)


# Feature Scaling
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Classification
from hmmlearn import hmm

classifier = hmm.GaussianHMM(n_components=3, covariance_type="full")
classifier.fit(X_train)

y_pred = classifier.predict(X_test)
```

```
size = len(y_pred)
strings = np.empty(size, np.unicode_)

for i in range (size):
    if y_pred[i] == 1:
      strings[i] = ("g")
    else:
      strings[i] = ("b")

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print("Confusion Matrix:")
cm = confusion_matrix(y_test, strings)
print(cm)

print("-------------------------------------------------------")
print("-------------------------------------------------------")


print("Performance Evaluation")
print(classification_report(y_test, strings))


print("-------------------------------------------------------")
print("-------------------------------------------------------")

print("Accuracy:")
print(accuracy_score(y_test, strings))

import matplotlib.pyplot as plt
import seaborn as sns
sns.heatmap(cm, annot=True, fmt="d")
plt.show()
```

```
Confusion Matrix:
[[39  0]
 [22 45]]
```

```
Performance Evaluation
              precision    recall  f1-score   support

           b       0.64      1.00      0.78        39
           g       1.00      0.67      0.80        67

    accuracy                           0.79       106
   macro avg       0.82      0.84      0.79       106
weighted avg       0.87      0.79      0.79       106
```

```
Accuracy:
0.7924528301886793
```

2. Breast Cancer Dataset

Accuracy Comparison Table :

Breast Cancer Dataset	**Gaussian HMM**	**Without Tuning**	**70-30**	92
			60-40	89
			50-50	92
			40-60	89
			30-70	87
		With Tuning	**70-30**	92
			60-40	93
			50-50	93
			40-60	**94**
			30-70	81
	GMM HMM	**Without Tuning**	**70-30**	93
			60-40	93
			50-50	92
			40-60	94
			30-70	93
		With Tuning	**70-30**	12
			60-40	92
			50-50	91
			40-60	91
			30-70	88
	Multinomial HMM	**Without Tuning**	**70-30**	59
			60-40	47
			50-50	51
			40-60	53
			30-70	57
		With Tuning	**70-30**	60
			60-40	62
			50-50	63
			40-60	62
			30-70	63

From. this table we can see that Guassian HMM classifier with Tuning with 40-60 split has highest accuracy of **94** .

Code of this model :

```
import pandas as pd
import numpy as np

# Dataset Preparation
df = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data",header=None)

df.columns =
['1','Class','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18',
'19'
           ,'20','21','22','23','24','25','26','27','28','29','30','31','32']


X = df.drop(['1','Class'], axis=1)
y = df['Class']

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.6)

# Feature Scaling
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# values for random grid
n_components = [int(x) for x in np.linspace(1,5, num = 5)]

covariance_type = ['spherical', 'tied', 'full', 'diag']

algorithm = ['viterbi', 'map']

n_iter = [int(x) for x in np.linspace(10, 1000, num = 5)]

# Create the random grid
```

```
random_grid = {
               'covariance_type': covariance_type,
               'algorithm': algorithm,
               'n_iter': n_iter}

from pprint import pprint

pprint(random_grid)
print()

# Classification
from hmmlearn import hmm

classifier = hmm.GaussianHMM(n_components=2)

from sklearn.model_selection import GridSearchCV
rf_random = GridSearchCV(classifier, random_grid, refit=True)

rf_random.fit(X_train)

y_pred = rf_random.predict(X_test)

size = len(y_pred)
strings = np.empty(size, np.unicode_)

for i in range (size):
    if y_pred[i] == 1:
      strings[i] = ("M")
    else:
      strings[i] = ("B")

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print("Confusion Matrix:")
cm = confusion_matrix(y_test, strings)
print(cm)

print("--------------------------------------------------")
print("--------------------------------------------------")


print("Performance Evaluation")
print(classification_report(y_test, strings))


print("--------------------------------------------------")
print("--------------------------------------------------")

print("Accuracy:")
print(accuracy_score(y_test, strings))

import matplotlib.pyplot as plt
import seaborn as sns
sns.heatmap(cm, annot=True, fmt="d")
plt.show()
```

```
Confusion Matrix:
[[215   1]
 [ 19 107]]
```

```
Performance Evaluation
              precision    recall  f1-score   support

           B       0.92      1.00      0.96       216
           M       0.99      0.85      0.91       126

    accuracy                           0.94       342
   macro avg       0.95      0.92      0.94       342
weighted avg       0.95      0.94      0.94       342
```

```
Accuracy:
0.9415204678362573
```

CNN

Accuracy Comparison Table :

Convolutional Neural Networks(CNN)	
Dataset	**Accuracy**
CIFAR-10	68
MNIST	99
SAVEE	29
EmoDB	50

Other Deep Learning Models

Accuracy Comparison Table :

Models	Dataset	Accuracy
VGG-16	CIFAR-10	9.8
	MNIST	10.95
	SAVEE	12.92
	EmoDB	25
ResNet-50	CIFAR-10	27
	MNIST	99
	SAVEE	99
	EmoDB	92
Recurrent Neural Networks (RNN)	CIFAR-10	29
	MNIST	97
	SAVEE	43
	EmoDB	55
AlexNet	CIFAR-10	7.5
	MNIST	11.69
	SAVEE	23.74
	EmoDB	23.36
GoogLeNet	CIFAR-10	26.6
	MNIST	99
	SAVEE	38
	EmoDB	36