Guided Assignment On Unsupervised Learning

Name: **Shreya Shrivastava**

Course: **Artificial Intelligence And Machine Learning**

Batch Four

Duration: 12 Months

Problem statement: Perform activity recognition on the dataset using a **Hidden Markov Model.** Then perform the same task using a different classification algorithm (logistic regression/decision tree) of your choice and compare the performance of the two algorithms

Prerequisites:

The libraries as well as things required in order for the program to work:

I. **Python 3.6** : The following url https://www.python.org/downloads/ can be referred to download python. Once you have python downloaded and installed, you will need to setup PATH variables (if you want to run python program directly, detail instructions are below in how to run software section). To do that check this: https://www.pythoncentral.io/add-python-to-path-python-is-not-recognized-as-an-internal-or-external-command/. Setting up PATH variable is optional as you can also run program without it and more instruction are given below on this topic. Second option is to download anaconda and use its anaconda prompt to run the commands. To install anaconda check this url : https://www.anaconda.com/download/

II. **HMM Learn :** If you have chosen to install python 3.6,then run the following command in command prompt/terminal to install this package :

```
pip install -U hmmlearn
```

If using Anaconda then run the following command in anaconda prompt to install this package:

conda install -c anaconda hmmlearn

III. **ADDITIONAL PACKAGES :** You will also need to download and install below 4 packages- numpy, scikit learn,matplotlib and pandasafter you install either python or anaconda from the steps above. If you have chosen to install python 3.6,then run the following commands in command prompt/terminal to install these packages :

NUMPY: pip install -U numpy

MATPLOTLIB: pip install -U matplotlib

SCIKIT LEARN: pip install -U scikit-learn

PANDAS: pip install -U pandas

If using Anaconda then run the following commands in anaconda prompt to install these packages:

NUMPY: conda install -c anaconda numpy

MATPLOTLIB: conda install -c anaconda matplotlib

SCIKIT LEARN: conda install -c scikit-learn

PANDAS: conda install -c pandas

IV. DATASET LINK:

https://www.kaggle.com/uciml/human-activity-recognition-with-smartphones

V. METHODS USED:

A. HIDDEN MARKOV MODELS

B. LINEAR REGRESSION

C. PRINCIPAL COMPONENT ANALYSIS

THE PROJECT :

1. Importing the libraries and loading the training as well as test dataset as a pandas dataframe.

```
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.metrics import f1_score,accuracy_score
import matplotlib.pyplot as plt
from hmmlearn import hmm
from sklearn.linear_model import LogisticRegression

df_train=pd.read_csv("/Users/shreyashrivastava/Desktop/AI COURSE/PROJECTS/Human activity recognition/archive/train.csv")
df_test=pd.read_csv("/Users/shreyashrivastava/Desktop/AI COURSE/PROJECTS/Human activity recognition/archive/test.csv")
```

2. Converting our data(without the labels) into a numpy array & calculating covariance along the columns.

```
data=df_train.iloc[:,:-1].values
cov=np.cov(data,rowvar=False)
```

3. Performing PCA on the data and transforming both the datasets so they now have 100 features each.

```
pca=PCA(100)
cov_pca=pca.fit(df_train.iloc[:,:-1].values)
data_train=cov_pca.transform(df_train.iloc[:,:-1].values)
data_test=cov_pca.transform(df_test.iloc[:,:-1].values)
df_train_red=pd.DataFrame(data_train)
df_train_red["Activity"]=df_train["Activity"]
```

4. Collecting instances of each acting and segregating them to fit the HMM model on each of the datasets.

```
df_train_red_STAND = df_train_red[df_train_red["Activity"]=="STANDING"]
df_train_red_SITTING = df_train_red[df_train_red["Activity"]=="SITTING"]
df_train_red_LAYING = df_train_red[df_train_red["Activity"]=="LAYING"]
df_train_red_WALKING = df_train_red[df_train_red["Activity"]=="WALKING"]
df_train_red_WALKING_DOWNSTAIRS = df_train_red[df_train_red["Activity"]=="WALKING_DOWNSTAIRS"]
df_train_red_WALKING_UPSTAIRS = df_train_red[df_train_red["Activity"]=="WALKING_UPSTAIRS"]
```

5. Dropping Null values and converting the categorical labels into numerical data.

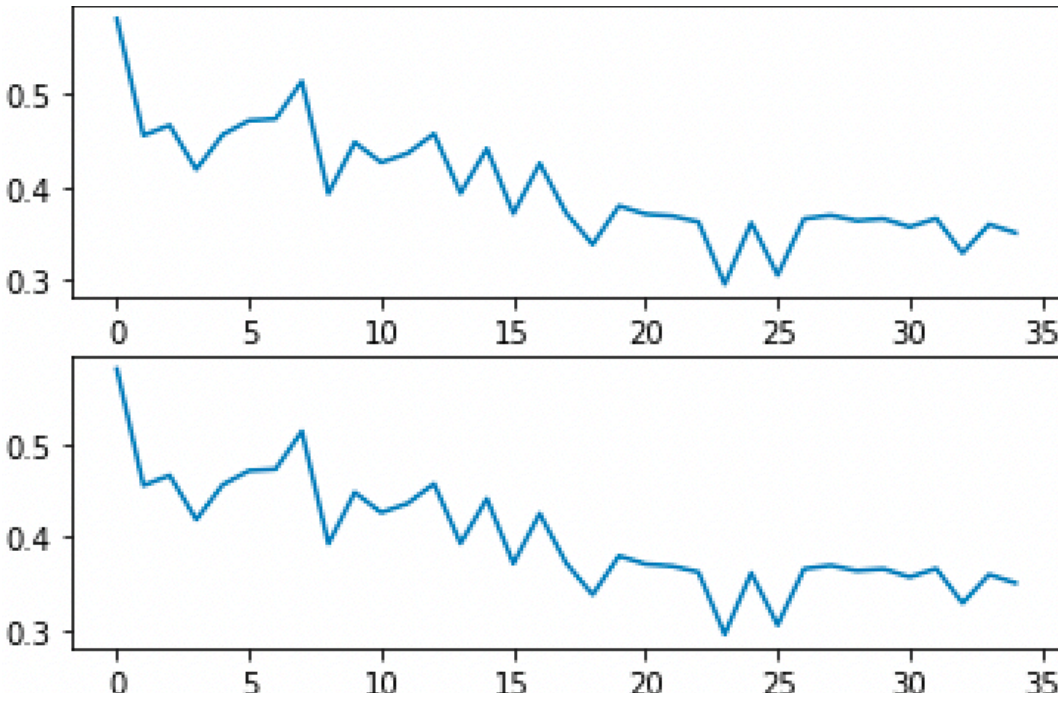
```
df_test.dropna(inplace=True)
df_test_red=pd.DataFrame(data_test)
df_test_red["Activity"]=df_test["Activity"]

labels_act=[]
for i in range(len(df_test_red)):
    if (df_test_red["Activity"].iloc[i]=="STANDING"):
        labels_act.append(0)
    if (df_test_red["Activity"].iloc[i]=="SITTING"):
        labels_act.append(1)
    if (df_test_red["Activity"].iloc[i]=="LAYING"):
        labels_act.append(2)
    if (df_test_red["Activity"].iloc[i]=="WALKING"):
        labels_act.append(3)
    if (df_test_red["Activity"].iloc[i]=="WALKING_UPSTAIRS"):
        labels_act.append(4)
    if (df_test_red["Activity"].iloc[i]=="WALKING_DOWNSTAIRS"):
        labels_act.append(5)
labels_act=np.array(labels_act)
```

6. Writing a simple function to implement hmm and calculate f1 score as well as accuracy. Also we're plotting the final results.

```
def hmm_f1_acc(N,M,labels_act):
    hmm_stand=hmm.GMMHMM(n_components=N,n_mix=M,covariance_type="diag")
    hmm_sit=hmm.GMMHMM(n_components=N,n_mix=M,covariance_type="diag")
    hmm_lay=hmm.GMMHMM(n_components=N,n_mix=M,covariance_type="diag")
    hmm_walk=hmm.GMMHMM(n_components=N,n_mix=M,covariance_type="diag")
    hmm_walk_d=hmm.GMMHMM(n_components=N,n_mix=M,covariance_type="diag")
    hmm_walk_u=hmm.GMMHMM(n_components=N,n_mix=M,covariance_type="diag")

    hmm_stand.fit(df_train_red_STAND.iloc[:,0:100].values)
    hmm_sit.fit(df_train_red_SITTING.iloc[:,0:100].values)
    hmm_lay.fit(df_train_red_LAYING.iloc[:,0:100].values)
    hmm_walk.fit(df_train_red_WALKING.iloc[:,0:100].values)
    hmm_walk_u.fit(df_train_red_WALKING_UPSTAIRS.iloc[:,0:100].values)
    hmm_walk_d.fit(df_train_red_WALKING_DOWNSTAIRS.iloc[:,0:100].values)


    y_pred = []
    for i in range(len(df_test_red)):
        log_like=np.array([hmm_stand.score(df_test_red.iloc[i,0:100].values.reshape((1,100))),
                           hmm_sit.score(df_test_red.iloc[i,0:100].values.reshape((1,100))),hmm_lay.score(df_test_red.iloc[i,0:100].values.reshape((1,100))),
                           hmm_walk_u.score(df_test_red.iloc[i,0:100].values.reshape((1,100))),hmm_walk_d.score(df_test_red.iloc[i,0:100].values.reshape((1,100))),
                           hmm_stand.score(df_test_red.iloc[i,0:100].values.reshape((1,100)))])
        y_pred.append(np.argmax(log_like))
    y_pred=np.array(y_pred)

    f_1= f1_score(labels_act,y_pred,average="micro")
    acc=accuracy_score(labels_act,y_pred)
    return f_1,acc

states= np.arange(1,36,1)
f1_val_state = []
acc_val_state= []
for i in states:
    print("HMM has been trained for {} states".format(i))
    f1,acc=hmm_f1_acc(i,1,labels_act)
    f1_val_state.append(f1)
    acc_val_state.append(acc)
fig,ax= plt.subplots(2,1)
ax[0].plot(f1_val_state)
ax[1].plot(acc_val_state)
plt.show()
```

FINAL RESULT

```
HMM has been trained for 1 states
HMM has been trained for 2 states
HMM has been trained for 3 states
HMM has been trained for 4 states
HMM has been trained for 5 states
HMM has been trained for 6 states
HMM has been trained for 7 states
HMM has been trained for 8 states
HMM has been trained for 9 states
HMM has been trained for 10 states
HMM has been trained for 11 states
HMM has been trained for 12 states
HMM has been trained for 13 states
HMM has been trained for 14 states
HMM has been trained for 15 states
HMM has been trained for 16 states
HMM has been trained for 17 states
HMM has been trained for 18 states
HMM has been trained for 19 states
HMM has been trained for 20 states
HMM has been trained for 21 states
HMM has been trained for 22 states
HMM has been trained for 23 states
HMM has been trained for 24 states
HMM has been trained for 25 states
HMM has been trained for 26 states
HMM has been trained for 27 states
HMM has been trained for 28 states
HMM has been trained for 29 states
HMM has been trained for 30 states
HMM has been trained for 31 states
HMM has been trained for 32 states
HMM has been trained for 33 states
HMM has been trained for 34 states
HMM has been trained for 35 states
```



THE ACCURACY AND F_1 VAL PLOTS

7.implementing logistic regression and calculating f1 scores and accuracy. We then proceed to Print the final f1 scores as well as accuracy in the end.

```
clf = LogisticRegression(solver='lbfgs', max_iter=1000)
labels_tr=[]
for i in range(len(df_train_red)):
    if (df_train_red["Activity"].iloc[i]=="STANDING"):
        labels_tr.append(0)
    if (df_train_red["Activity"].iloc[i]=="SITTING"):
        labels_tr.append(1)
    if (df_train_red["Activity"].iloc[i]=="LAYING"):
        labels_tr.append(2)
    if (df_train_red["Activity"].iloc[i]=="WALKING"):
        labels_tr.append(3)
    if (df_train_red["Activity"].iloc[i]=="WALKING_UPSTAIRS"):
        labels_tr.append(4)
    if (df_train_red["Activity"].iloc[i]=="WALKING_DOWNSTAIRS"):
        labels_tr.append(5)
labels_tr=np.array(labels_tr)
labels_tr.shape

clf.fit(df_train_red.iloc[:,0:100].values,labels_tr)
predictions = clf.predict(df_test_red.iloc[:,0:100].values)
f1_2= f1_score(labels_act,predictions,average="micro")
acc2=accuracy_score(labels_act,predictions)

print("F1 SCORE using HMM:",np.round(f1,decimals=2))
print("Accuracy using HMM:",np.round(acc,decimals=2))

print("F1 SCORE using Logistic Regression:",np.round(f1_2,2))
print("Accuracy using Logistic Regression:",np.round(acc2,2))
```

```
F1 SCORE using HMM: 0.35
Accuracy using HMM: 0.35
F1 SCORE using Logistic Regression: 0.94
Accuracy using Logistic Regression: 0.94
```