

# Project Report

## Human Activity Recognition From Smart Phone Data

Name: Keshav Anand

Course: AI and ML

(Batch-4)

Duration: 12 months

Problem Statement: Human activity recognition from smart phone data using HMM.

### Prerequisites

---

What things you need to install the software and how to install them:

Python 3.6 This setup requires that your machine has latest version of python. 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 and easier 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/> You will also need to download and install below 3 packages after you install either python or anaconda from the steps above Sklearn (scikit-learn) numpy scipy if you have chosen to install python 3.6 then run below commands in command prompt/terminal to install these packages `pip install -U scikit-learn` `pip install numpy` `pip install scipy` if you have chosen to install anaconda then run below commands in anaconda prompt to install these packages `conda install -c scikit-learn` `conda install -c anaconda numpy` `conda install -c anaconda scipy`

### Dataset used:

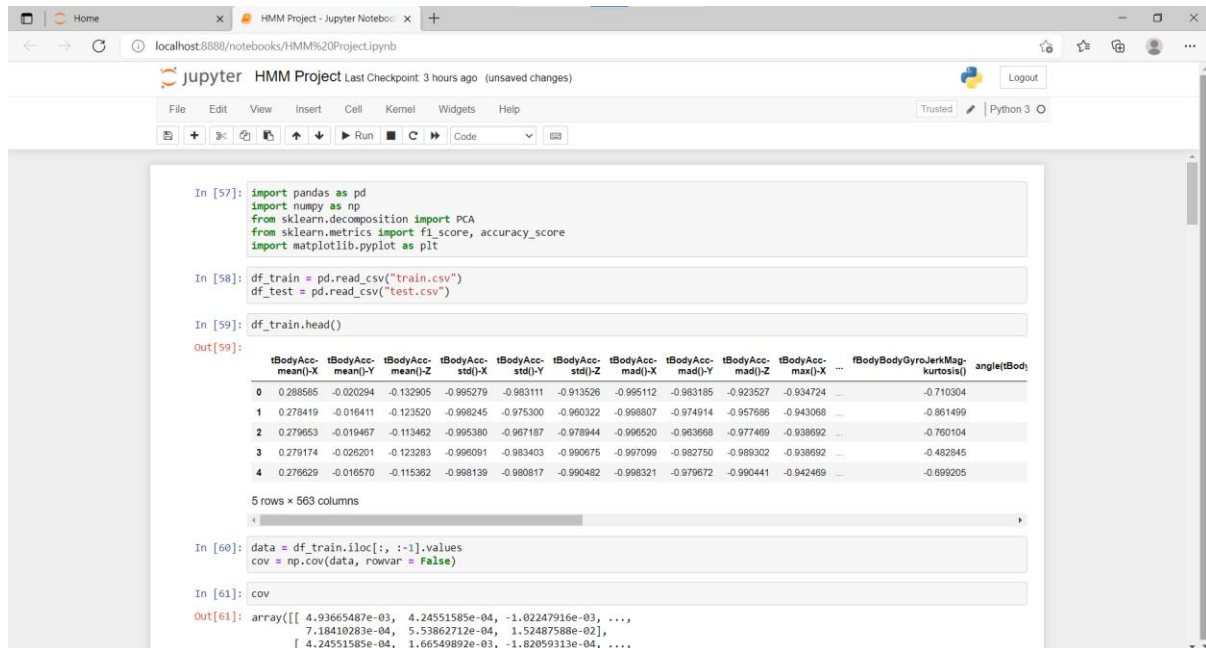
---

The dataset used is Human Activity Recognition with Smartphones dataset which is available on a website called Kaggle.com .

## Method used for Recognition:

### HMM (Hidden Markov Model)

## Screenshots of Source Code and Output:



```
In [57]: 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

In [58]: df_train = pd.read_csv("train.csv")
df_test = pd.read_csv("test.csv")

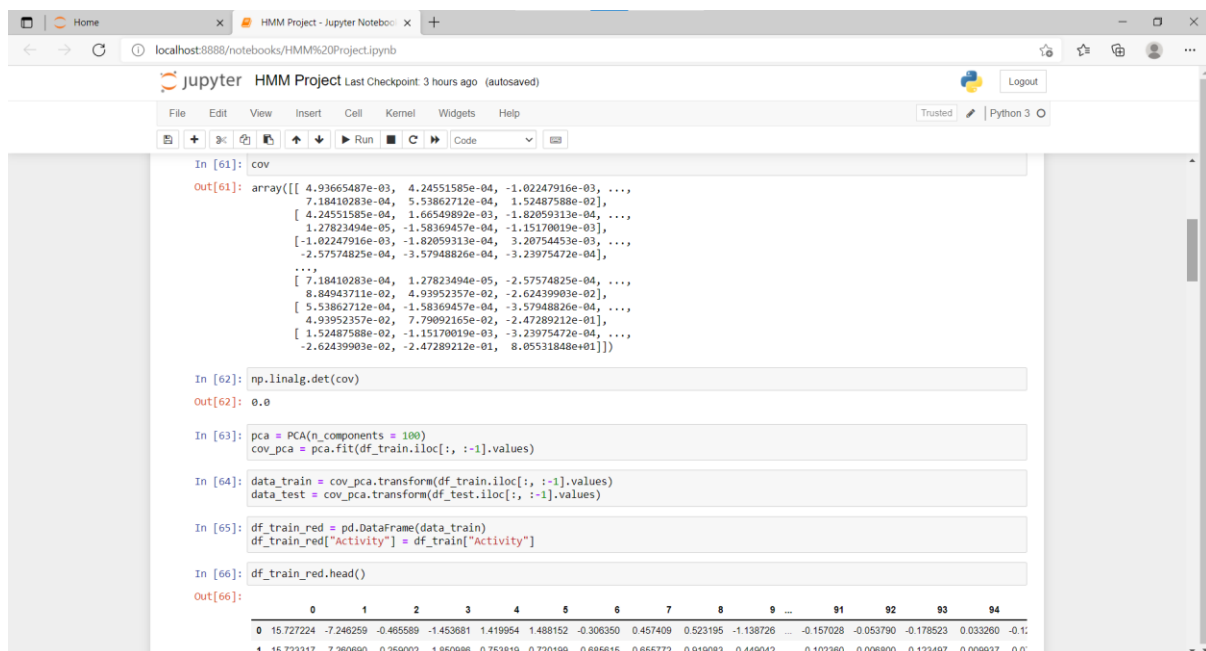
In [59]: df_train.head()
Out[59]:
```

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	tBodyAcc-mad()-Y	tBodyAcc-mad()-Z	tBodyAcc-max()-X	...	tBodyGyroJerkMag-kurtosis()	angle(tBody)
0	0.288585	-0.020294	-0.132905	-0.995279	-0.983111	-0.913526	-0.995112	-0.983185	-0.923527	-0.934724	...	-0.710304	
1	0.278419	-0.016411	-0.123520	-0.998245	-0.975300	-0.960322	-0.998807	-0.974914	-0.957686	-0.943068	...	-0.861499	
2	0.279653	-0.019467	-0.113462	-0.995380	-0.967187	-0.978944	-0.996520	-0.963668	-0.977469	-0.938692	...	-0.760104	
3	0.279174	-0.026201	-0.123283	-0.996091	-0.983403	-0.990675	-0.997099	-0.982750	-0.989302	-0.938692	...	-0.452845	
4	0.278629	-0.016570	-0.115362	-0.998139	-0.980817	-0.990482	-0.998321	-0.979672	-0.990441	-0.942469	...	-0.699205	

```
5 rows x 563 columns

In [60]: data = df_train.iloc[:, :-1].values
cov = np.cov(data, rowvar = False)

In [61]: cov
Out[61]: array([[ 4.93665487e-03,  4.24551585e-04, -1.02247916e-03, ...,
                  7.18410283e-04,  5.53862712e-04,  1.52487588e-02],
                 [ 4.24551585e-04,  1.66549892e-03, -1.82059313e-04, ...,
                  5.53862712e-04,  1.66549892e-03, -1.82059313e-04],
                 [ -1.02247916e-03, -1.82059313e-04,  3.20754453e-03, ...,
                  1.52487588e-02, -1.82059313e-04,  3.20754453e-03],
                 ...,
                 [ 7.18410283e-04,  1.27823494e-05, -2.57574825e-04, ...,
                  8.84943711e-02,  4.93952357e-02, -2.62439903e-02],
                 [ 5.53862712e-04, -1.58369457e-04, -3.57948826e-04, ...,
                  4.93952357e-02,  7.79892165e-02, -2.47289212e-01],
                 [ -2.57574825e-04, -3.57948826e-04, -3.23975472e-04, ...,
                  1.52487588e-02, -1.5170019e-03,  3.23975472e-04],
                 [ -2.62439903e-02, -2.47289212e-01,  8.05531848e+01]])
```



```
In [61]: cov
Out[61]: array([[ 4.93665487e-03,  4.24551585e-04, -1.02247916e-03, ...,
                  7.18410283e-04,  5.53862712e-04,  1.52487588e-02],
                 [ 4.24551585e-04,  1.66549892e-03, -1.82059313e-04, ...,
                  5.53862712e-04,  1.66549892e-03, -1.82059313e-04],
                 [ -1.02247916e-03, -1.82059313e-04,  3.20754453e-03, ...,
                  1.52487588e-02, -1.82059313e-04,  3.20754453e-03],
                 ...,
                 [ 7.18410283e-04,  1.27823494e-05, -2.57574825e-04, ...,
                  8.84943711e-02,  4.93952357e-02, -2.62439903e-02],
                 [ 5.53862712e-04, -1.58369457e-04, -3.57948826e-04, ...,
                  4.93952357e-02,  7.79892165e-02, -2.47289212e-01],
                 [ -2.57574825e-04, -3.57948826e-04, -3.23975472e-04, ...,
                  1.52487588e-02, -1.5170019e-03,  3.23975472e-04],
                 [ -2.62439903e-02, -2.47289212e-01,  8.05531848e+01]])

In [62]: np.linalg.det(cov)
Out[62]: 0.0

In [63]: pca = PCA(n_components = 100)
cov_pca = pca.fit(df_train.iloc[:, :-1].values)

In [64]: data_train = cov_pca.transform(df_train.iloc[:, :-1].values)
data_test = cov_pca.transform(df_test.iloc[:, :-1].values)

In [65]: df_train_red = pd.DataFrame(data_train)
df_train_red["Activity"] = df_train["Activity"]

In [66]: df_train_red.head()
Out[66]:
```

	0	1	2	3	4	5	6	7	8	9	...	91	92	93	94
0	15.727224	-7.246259	-0.465589	-1.453681	1.419954	1.488152	-0.306350	0.457409	0.523195	-1.138726	...	-0.157028	-0.053790	-0.178523	0.033260
1	15.723317	-7.260690	-0.259002	-1.850886	0.753819	0.720199	0.685615	-0.655772	-0.919083	-0.449042	...	-0.102360	-0.006800	-0.123497	0.009937

```

In [67]: df_train_red_STAND = df_train_red[df_train_red["Activity"] == "STANDING"]
df_train_red_SIT = df_train_red[df_train_red["Activity"] == "SITTING"]
df_train_red_WALK = df_train_red[df_train_red["Activity"] == "WALKING"]
df_train_red_LAY = df_train_red[df_train_red["Activity"] == "LAYING"]
df_train_red_WALK_D = df_train_red[df_train_red["Activity"] == "WALKING_DOWNSTAIRS"]
df_train_red_WALK_U = df_train_red[df_train_red["Activity"] == "WALKING_UPSTAIRS"]

In [68]: print(df_train_red_SIT.shape)
(1286, 101)

In [69]: df_test_red.dropna(inplace = True)

In [70]: df_test_red = pd.DataFrame(data_test)
df_test_red["Activity"] = df_test["Activity"]

In [71]: df_test_red.shape
Out[71]: (2947, 101)

In [72]: # Calculating true labels
labels_true = []
for i in range(len(df_test_red)):
    if (df_test_red['Activity'].iloc[i] == 'STANDING'):
        labels_true.append(0)
    if (df_test_red['Activity'].iloc[i] == 'SITTING'):
        labels_true.append(1)
    if (df_test_red['Activity'].iloc[i] == 'LAYING'):
        labels_true.append(2)
    if (df_test_red['Activity'].iloc[i] == 'WALKING'):
        labels_true.append(3)
    if (df_test_red['Activity'].iloc[i] == 'WALKING_UPSTAIRS'):
        labels_true.append(4)
    if (df_test_red['Activity'].iloc[i] == 'WALKING_DOWNSTAIRS'):
        labels_true.append(5)

```

```

In [72]: # Calculating true labels
labels_true = []
for i in range(len(df_test_red)):
    if (df_test_red['Activity'].iloc[i] == 'STANDING'):
        labels_true.append(0)
    if (df_test_red['Activity'].iloc[i] == 'SITTING'):
        labels_true.append(1)
    if (df_test_red['Activity'].iloc[i] == 'LAYING'):
        labels_true.append(2)
    if (df_test_red['Activity'].iloc[i] == 'WALKING'):
        labels_true.append(3)
    if (df_test_red['Activity'].iloc[i] == 'WALKING_UPSTAIRS'):
        labels_true.append(4)
    if (df_test_red['Activity'].iloc[i] == 'WALKING_DOWNSTAIRS'):
        labels_true.append(5)
labels_true = np.array(labels_true)
labels_true.shape
Out[72]: (2947,)

In [73]: ! pip install hmmlearn
Requirement already satisfied: hmmlearn in c:\users\keshav anand\appdata\local\programs\python\python37\lib\site-packages (0.2.6)
Requirement already satisfied: scipy>=0.19 in c:\users\keshav anand\appdata\local\programs\python\python37\lib\site-packages (from hmmlearn) (1.7.1)
Requirement already satisfied: scikit-learn>=0.16 in c:\users\keshav anand\appdata\local\programs\python\python37\lib\site-packages (from hmmlearn) (0.24.2)
Requirement already satisfied: numpy>=1.10 in c:\users\keshav anand\appdata\roaming\python\python37\site-packages (from hmmlearn) (1.20.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\keshav anand\appdata\local\programs\python\python37\lib\site-packages (from scikit-learn>=0.16->hmmlearn) (2.2.0)
Requirement already satisfied: joblib>=0.11 in c:\users\keshav anand\appdata\local\programs\python\python37\lib\site-packages (from scikit-learn>=0.16->hmmlearn) (1.0.1)

WARNING: You are using pip version 21.1.3; however, version 21.2.4 is available.
You should consider upgrading via the 'c:\users\keshav anand\appdata\local\programs\python\python37\python.exe -m pip install -

```

```

In [74]: from hmmlearn import hmm

In [79]: Implementing hmm
fitting hmm for each activity
HMM_F1score(N, M, labels_true):
    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 SIT.iloc[:, 0:100].values)
    hmm_lay.fit(df_train_red LAY.iloc[:, 0:100].values)
    hmm_walk.fit(df_train_red WALK.iloc[:, 0:100].values)
    hmm_walk_d.fit(df_train_red WALK_D.iloc[:, 0:100].values)
    hmm_walk_u.fit(df_train_red WALK_U.iloc[:, 0:100].values)

    # calculating f1 score
    labels_predict = []
    for i in range(len(df_test_red)):
        log_likelihood_value = 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))),
        labels_predict.append(np.argmax(log_likelihood_value))
    labels_predict = np.array(labels_predict)

    F1 = f1_score(labels_true, labels_predict, average = 'micro')
    acc = accuracy_score(labels_true, labels_predict)
    return F1, acc

In [80]: states = np.arange(1, 36, 1)
states

Out[80]: array([ 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,
        35])

```

```

In [82]: F1_value_states = []
acc_value_states = []
for i in states:
    print("HMM has been trained for num_states = {}".format(i))
    f1, acc = HMM_F1score(i, 1, labels_true)
    F1_value_states.append(f1)
    acc_value_states.append(acc)
fig, ax = plt.subplots(2,1)

ax[0].plot(F1_value_states)
ax[1].plot(acc_value_states)

plt.show()

HMM has been trained for num_states = 1
HMM has been trained for num_states = 2
HMM has been trained for num_states = 3
HMM has been trained for num_states = 4
HMM has been trained for num_states = 5
HMM has been trained for num_states = 6
HMM has been trained for num_states = 7
HMM has been trained for num_states = 8
HMM has been trained for num_states = 9
HMM has been trained for num_states = 10
HMM has been trained for num_states = 11
HMM has been trained for num_states = 12
HMM has been trained for num_states = 13
HMM has been trained for num_states = 14
HMM has been trained for num_states = 15
HMM has been trained for num_states = 16
HMM has been trained for num_states = 17
HMM has been trained for num_states = 18
HMM has been trained for num_states = 19
HMM has been trained for num_states = 20
HMM has been trained for num_states = 21
HMM has been trained for num_states = 22
HMM has been trained for num_states = 23

```

