In [2]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6
qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%
b&response_type=code&scope=email%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2
www.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.readonly
ttps%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly

Enter your authorization code:
··········
Mounted at /content/drive
```

In [ ]:

```
!unzip '/content/drive/My Drive/Assign - 23 Human Activity Recognition/Copy of
HumanActivityRecognition.zip' -d '/content/drive/My Drive/Assign - 23 Human Activity Recognition/'
```

```
Archive:  /content/drive/My Drive/Assign - 23 Human Activity Recognition/Copy of
HumanActivityRecognition.zip
   creating: /content/drive/My Drive/Assign - 23 Human Activity Recognition/HAR/
  inflating: /content/drive/My Drive/Assign - 23 Human Activity Recognition/HAR/.DS_Store
   creating: /content/drive/My Drive/Assign - 23 Human Activity Recognition/__MACOSX/
   creating: /content/drive/My Drive/Assign - 23 Human Activity Recognition/__MACOSX/HAR/
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/._.DS_Store
   creating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/.ipynb_checkpoints/
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/.ipynb_checkpoints/HAR_EDA-checkpoint.ipynb
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/.ipynb_checkpoints/HAR_LSTM-checkpoint.ipynb
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/.ipynb_checkpoints/HAR_LSTM_1-checkpoint.ipynb
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/.ipynb_checkpoints/HAR_PREDICTION_MODELS-checkpoint.ipynb
  inflating: /content/drive/My Drive/Assign - 23 Human Activity Recognition/HAR/HAR_EDA.ipynb
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/._HAR_EDA.ipynb
  inflating: /content/drive/My Drive/Assign - 23 Human Activity Recognition/HAR/HAR_LSTM.ipynb
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/._HAR_LSTM.ipynb
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/HAR_PREDICTION_MODELS.ipynb
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/._HAR_PREDICTION_MODELS.ipynb
  inflating: /content/drive/My Drive/Assign - 23 Human Activity Recognition/HAR/t-
sne_perp_10_iter_1000.png
  inflating: /content/drive/My Drive/Assign - 23 Human Activity Recognition/__MACOSX/HAR/._t-
sne_perp_10_iter_1000.png
  inflating: /content/drive/My Drive/Assign - 23 Human Activity Recognition/HAR/t-
sne_perp_20_iter_1000.png
  inflating: /content/drive/My Drive/Assign - 23 Human Activity Recognition/__MACOSX/HAR/._t-
sne_perp_20_iter_1000.png
  inflating: /content/drive/My Drive/Assign - 23 Human Activity Recognition/HAR/t-
sne_perp_2_iter_1000.png
  inflating: /content/drive/My Drive/Assign - 23 Human Activity Recognition/__MACOSX/HAR/._t-
sne_perp_2_iter_1000.png
  inflating: /content/drive/My Drive/Assign - 23 Human Activity Recognition/HAR/t-
sne_perp_50_iter_1000.png
  inflating: /content/drive/My Drive/Assign - 23 Human Activity Recognition/__MACOSX/HAR/._t-
sne_perp_50_iter_1000.png
  inflating: /content/drive/My Drive/Assign - 23 Human Activity Recognition/HAR/t-
sne_perp_5_iter_1000.png
  inflating: /content/drive/My Drive/Assign - 23 Human Activity Recognition/__MACOSX/HAR/._t-
sne_perp_5_iter_1000.png
   creating: /content/drive/My Drive/Assign - 23 Human Activity Recognition/HAR/UCI_HAR_Dataset/
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/.DS_Store
   creating: /content/drive/My Drive/Assign - 23 Human Activity
```

```
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/._.DS_Store
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/_DS_Store
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/.__DS_Store
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/activity_labels.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/._activity_labels.txt
   creating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/csv_files/
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/csv_files/test.csv
   creating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/csv_files/
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/csv_files/._test.csv
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/csv_files/train.csv
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/csv_files/._train.csv
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/features.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/._features.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/features_info.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/._features_info.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/README.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/._README.txt
   creating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/test/
   creating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/test/Inertial Signals/
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/test/Inertial Signals/body_acc_x_test.txt
   creating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/test/
   creating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/test/Inertial Signals/
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/test/Inertial Signals/._body_acc_x_test.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/test/Inertial Signals/body_acc_y_test.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/test/Inertial Signals/._body_acc_y_test.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/test/Inertial Signals/body_acc_z_test.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/test/Inertial Signals/._body_acc_z_test.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/test/Inertial Signals/body_gyro_x_test.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/test/Inertial Signals/._body_gyro_x_test.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/test/Inertial Signals/body_gyro_y_test.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/test/Inertial Signals/._body_gyro_y_test.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/test/Inertial Signals/body_gyro_z_test.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/test/Inertial Signals/._body_gyro_z_test.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/test/Inertial Signals/total_acc_x_test.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/test/Inertial Signals/._total_acc_x_test.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/test/Inertial Signals/total_acc_y_test.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/test/Inertial Signals/._total_acc_y_test.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/test/Inertial Signals/total_acc_z_test.txt
```

```
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/test/Inertial Signals/._total_acc_z_test.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/test/subject_test.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/test/._subject_test.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/test/X_test.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/test/._X_test.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/test/y_test.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/test/._y_test.txt
   creating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/train/
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/train/.DS_Store
   creating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/train/
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/train/._.DS_Store
   creating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/train/Inertial Signals/
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/train/Inertial Signals/body_acc_x_train.txt
   creating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/train/Inertial Signals/
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/train/Inertial Signals/._body_acc_x_train.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/train/Inertial Signals/body_acc_y_train.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/train/Inertial Signals/._body_acc_y_train.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/train/Inertial Signals/body_acc_z_train.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/train/Inertial Signals/._body_acc_z_train.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/train/Inertial Signals/body_gyro_x_train.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/train/Inertial Signals/._body_gyro_x_train.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/train/Inertial Signals/body_gyro_y_train.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/train/Inertial Signals/._body_gyro_y_train.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/train/Inertial Signals/body_gyro_z_train.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/train/Inertial Signals/._body_gyro_z_train.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/train/Inertial Signals/total_acc_x_train.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/train/Inertial Signals/._total_acc_x_train.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/train/Inertial Signals/total_acc_y_train.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/train/Inertial Signals/._total_acc_y_train.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/train/Inertial Signals/total_acc_z_train.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/train/Inertial Signals/._total_acc_z_train.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/train/subject_train.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/train/._subject_train.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/train/X_train.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/train/._X_train.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/train/y_train.txt
  inflating: /content/drive/My Drive/Assign - 23 Human Activity
Recognition/__MACOSX/HAR/UCI_HAR_Dataset/train/._y_train.txt
```

1.Loading of Data

In [ ]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning:
pandas.util.testing is deprecated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

In [ ]:

```
#getting the features from features.txt which is engineered by experts

features = list()
with open('/content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/features.txt') as f:
    features = [line.split()[1] for line in f.readlines()]

print('No of features = ',len(features))
```

```
No of features =  561
```

In [ ]:

```
features
```

Out[ ]:

```
['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',
 'tBodyAcc-max()-Y',
 'tBodyAcc-max()-Z',
 'tBodyAcc-min()-X',
 'tBodyAcc-min()-Y',
 'tBodyAcc-min()-Z',
 'tBodyAcc-sma()',
 'tBodyAcc-energy()-X',
 'tBodyAcc-energy()-Y',
 'tBodyAcc-energy()-Z',
 'tBodyAcc-iqr()-X',
 'tBodyAcc-iqr()-Y',
 'tBodyAcc-iqr()-Z',
 'tBodyAcc-entropy()-X',
 'tBodyAcc-entropy()-Y',
 'tBodyAcc-entropy()-Z',
 'tBodyAcc-arCoeff()-X,1',
 'tBodyAcc-arCoeff()-X,2',
 'tBodyAcc-arCoeff()-X,3',
 'tBodyAcc-arCoeff()-X,4',
 'tBodyAcc-arCoeff()-Y,1',
 'tBodyAcc-arCoeff()-Y,2',
 'tBodyAcc-arCoeff()-Y,3',
 'tBodyAcc-arCoeff()-Y,4',
 'tBodyAcc-arCoeff()-Z,1',
 'tBodyAcc-arCoeff()-Z,2',
 'tBodyAcc-arCoeff()-Z,3',
 'tBodyAcc-arCoeff()-Z,4',
 'tBodyAcc-correlation()-X,Y',
 'tBodyAcc-correlation()-X,Z',
 'tBodyAcc-correlation()-Y,Z'
```

```
 'tBodyAcc-correlation()-1,2',
'tGravityAcc-mean()-X',
'tGravityAcc-mean()-Y',
'tGravityAcc-mean()-Z',
'tGravityAcc-std()-X',
'tGravityAcc-std()-Y',
'tGravityAcc-std()-Z',
'tGravityAcc-mad()-X',
'tGravityAcc-mad()-Y',
'tGravityAcc-mad()-Z',
'tGravityAcc-max()-X',
'tGravityAcc-max()-Y',
'tGravityAcc-max()-Z',
'tGravityAcc-min()-X',
'tGravityAcc-min()-Y',
'tGravityAcc-min()-Z',
'tGravityAcc-sma()',
'tGravityAcc-energy()-X',
'tGravityAcc-energy()-Y',
'tGravityAcc-energy()-Z',
'tGravityAcc-iqr()-X',
'tGravityAcc-iqr()-Y',
'tGravityAcc-iqr()-Z',
'tGravityAcc-entropy()-X',
'tGravityAcc-entropy()-Y',
'tGravityAcc-entropy()-Z',
'tGravityAcc-arCoeff()-X,1',
'tGravityAcc-arCoeff()-X,2',
'tGravityAcc-arCoeff()-X,3',
'tGravityAcc-arCoeff()-X,4',
'tGravityAcc-arCoeff()-Y,1',
'tGravityAcc-arCoeff()-Y,2',
'tGravityAcc-arCoeff()-Y,3',
'tGravityAcc-arCoeff()-Y,4',
'tGravityAcc-arCoeff()-Z,1',
'tGravityAcc-arCoeff()-Z,2',
'tGravityAcc-arCoeff()-Z,3',
'tGravityAcc-arCoeff()-Z,4',
'tGravityAcc-correlation()-X,Y',
'tGravityAcc-correlation()-X,Z',
'tGravityAcc-correlation()-Y,Z',
'tBodyAccJerk-mean()-X',
'tBodyAccJerk-mean()-Y',
'tBodyAccJerk-mean()-Z',
'tBodyAccJerk-std()-X',
'tBodyAccJerk-std()-Y',
'tBodyAccJerk-std()-Z',
'tBodyAccJerk-mad()-X',
'tBodyAccJerk-mad()-Y',
'tBodyAccJerk-mad()-Z',
'tBodyAccJerk-max()-X',
'tBodyAccJerk-max()-Y',
'tBodyAccJerk-max()-Z',
'tBodyAccJerk-min()-X',
'tBodyAccJerk-min()-Y',
'tBodyAccJerk-min()-Z',
'tBodyAccJerk-sma()',
'tBodyAccJerk-energy()-X',
'tBodyAccJerk-energy()-Y',
'tBodyAccJerk-energy()-Z',
'tBodyAccJerk-iqr()-X',
'tBodyAccJerk-iqr()-Y',
'tBodyAccJerk-iqr()-Z',
'tBodyAccJerk-entropy()-X',
'tBodyAccJerk-entropy()-Y',
'tBodyAccJerk-entropy()-Z',
'tBodyAccJerk-arCoeff()-X,1',
'tBodyAccJerk-arCoeff()-X,2',
'tBodyAccJerk-arCoeff()-X,3',
'tBodyAccJerk-arCoeff()-X,4',
'tBodyAccJerk-arCoeff()-Y,1',
'tBodyAccJerk-arCoeff()-Y,2',
'tBodyAccJerk-arCoeff()-Y,3',
'tBodyAccJerk-arCoeff()-Y,4',
'tBodyAccJerk-arCoeff()-Z,1',
'tBodyAccJerk-arCoeff()-Z,2',
'tBodyAccJerk-arCoeff()-Z,3',
'tBodyAccJerk-arCoeff()-Z,4',
```

```
'tBodyAccJerk-arCoeff()-Z,4',
'tBodyAccJerk-correlation()-X,Y',
'tBodyAccJerk-correlation()-X,Z',
'tBodyAccJerk-correlation()-Y,Z',
'tBodyGyro-mean()-X',
'tBodyGyro-mean()-Y',
'tBodyGyro-mean()-Z',
'tBodyGyro-std()-X',
'tBodyGyro-std()-Y',
'tBodyGyro-std()-Z',
'tBodyGyro-mad()-X',
'tBodyGyro-mad()-Y',
'tBodyGyro-mad()-Z',
'tBodyGyro-max()-X',
'tBodyGyro-max()-Y',
'tBodyGyro-max()-Z',
'tBodyGyro-min()-X',
'tBodyGyro-min()-Y',
'tBodyGyro-min()-Z',
'tBodyGyro-sma()',
'tBodyGyro-energy()-X',
'tBodyGyro-energy()-Y',
'tBodyGyro-energy()-Z',
'tBodyGyro-iqr()-X',
'tBodyGyro-iqr()-Y',
'tBodyGyro-iqr()-Z',
'tBodyGyro-entropy()-X',
'tBodyGyro-entropy()-Y',
'tBodyGyro-entropy()-Z',
'tBodyGyro-arCoeff()-X,1',
'tBodyGyro-arCoeff()-X,2',
'tBodyGyro-arCoeff()-X,3',
'tBodyGyro-arCoeff()-X,4',
'tBodyGyro-arCoeff()-Y,1',
'tBodyGyro-arCoeff()-Y,2',
'tBodyGyro-arCoeff()-Y,3',
'tBodyGyro-arCoeff()-Y,4',
'tBodyGyro-arCoeff()-Z,1',
'tBodyGyro-arCoeff()-Z,2',
'tBodyGyro-arCoeff()-Z,3',
'tBodyGyro-arCoeff()-Z,4',
'tBodyGyro-correlation()-X,Y',
'tBodyGyro-correlation()-X,Z',
'tBodyGyro-correlation()-Y,Z',
'tBodyGyroJerk-mean()-X',
'tBodyGyroJerk-mean()-Y',
'tBodyGyroJerk-mean()-Z',
'tBodyGyroJerk-std()-X',
'tBodyGyroJerk-std()-Y',
'tBodyGyroJerk-std()-Z',
'tBodyGyroJerk-mad()-X',
'tBodyGyroJerk-mad()-Y',
'tBodyGyroJerk-mad()-Z',
'tBodyGyroJerk-max()-X',
'tBodyGyroJerk-max()-Y',
'tBodyGyroJerk-max()-Z',
'tBodyGyroJerk-min()-X',
'tBodyGyroJerk-min()-Y',
'tBodyGyroJerk-min()-Z',
'tBodyGyroJerk-sma()',
'tBodyGyroJerk-energy()-X',
'tBodyGyroJerk-energy()-Y',
'tBodyGyroJerk-energy()-Z',
'tBodyGyroJerk-iqr()-X',
'tBodyGyroJerk-iqr()-Y',
'tBodyGyroJerk-iqr()-Z',
'tBodyGyroJerk-entropy()-X',
'tBodyGyroJerk-entropy()-Y',
'tBodyGyroJerk-entropy()-Z',
'tBodyGyroJerk-arCoeff()-X,1',
'tBodyGyroJerk-arCoeff()-X,2',
'tBodyGyroJerk-arCoeff()-X,3',
'tBodyGyroJerk-arCoeff()-X,4',
'tBodyGyroJerk-arCoeff()-Y,1',
'tBodyGyroJerk-arCoeff()-Y,2',
'tBodyGyroJerk-arCoeff()-Y,3',
'tBodyGyroJerk-arCoeff()-Y,4',
'tBodyGyroJerk-arCoeff()-Z,1',
```

```
'tBodyGyroJerk-arCoeff()-Z,1',
'tBodyGyroJerk-arCoeff()-Z,2',
'tBodyGyroJerk-arCoeff()-Z,3',
'tBodyGyroJerk-arCoeff()-Z,4',
'tBodyGyroJerk-correlation()-X,Y',
'tBodyGyroJerk-correlation()-X,Z',
'tBodyGyroJerk-correlation()-Y,Z',
'tBodyAccMag-mean()',
'tBodyAccMag-std()',
'tBodyAccMag-mad()',
'tBodyAccMag-max()',
'tBodyAccMag-min()',
'tBodyAccMag-sma()',
'tBodyAccMag-energy()',
'tBodyAccMag-iqr()',
'tBodyAccMag-entropy()',
'tBodyAccMag-arCoeff()1',
'tBodyAccMag-arCoeff()2',
'tBodyAccMag-arCoeff()3',
'tBodyAccMag-arCoeff()4',
'tGravityAccMag-mean()',
'tGravityAccMag-std()',
'tGravityAccMag-mad()',
'tGravityAccMag-max()',
'tGravityAccMag-min()',
'tGravityAccMag-sma()',
'tGravityAccMag-energy()',
'tGravityAccMag-iqr()',
'tGravityAccMag-entropy()',
'tGravityAccMag-arCoeff()1',
'tGravityAccMag-arCoeff()2',
'tGravityAccMag-arCoeff()3',
'tGravityAccMag-arCoeff()4',
'tBodyAccJerkMag-mean()',
'tBodyAccJerkMag-std()',
'tBodyAccJerkMag-mad()',
'tBodyAccJerkMag-max()',
'tBodyAccJerkMag-min()',
'tBodyAccJerkMag-sma()',
'tBodyAccJerkMag-energy()',
'tBodyAccJerkMag-iqr()',
'tBodyAccJerkMag-entropy()',
'tBodyAccJerkMag-arCoeff()1',
'tBodyAccJerkMag-arCoeff()2',
'tBodyAccJerkMag-arCoeff()3',
'tBodyAccJerkMag-arCoeff()4',
'tBodyGyroMag-mean()',
'tBodyGyroMag-std()',
'tBodyGyroMag-mad()',
'tBodyGyroMag-max()',
'tBodyGyroMag-min()',
'tBodyGyroMag-sma()',
'tBodyGyroMag-energy()',
'tBodyGyroMag-iqr()',
'tBodyGyroMag-entropy()',
'tBodyGyroMag-arCoeff()1',
'tBodyGyroMag-arCoeff()2',
'tBodyGyroMag-arCoeff()3',
'tBodyGyroMag-arCoeff()4',
'tBodyGyroJerkMag-mean()',
'tBodyGyroJerkMag-std()',
'tBodyGyroJerkMag-mad()',
'tBodyGyroJerkMag-max()',
'tBodyGyroJerkMag-min()',
'tBodyGyroJerkMag-sma()',
'tBodyGyroJerkMag-energy()',
'tBodyGyroJerkMag-iqr()',
'tBodyGyroJerkMag-entropy()',
'tBodyGyroJerkMag-arCoeff()1',
'tBodyGyroJerkMag-arCoeff()2',
'tBodyGyroJerkMag-arCoeff()3',
'tBodyGyroJerkMag-arCoeff()4',
'fBodyAcc-mean()-X',
'fBodyAcc-mean()-Y',
'fBodyAcc-mean()-Z',
'fBodyAcc-std()-X',
'fBodyAcc-std()-Y',
'fBodyAcc-std()-Z',
```

```
'fBodyAcc-std()-Z',
'fBodyAcc-mad()-X',
'fBodyAcc-mad()-Y',
'fBodyAcc-mad()-Z',
'fBodyAcc-max()-X',
'fBodyAcc-max()-Y',
'fBodyAcc-max()-Z',
'fBodyAcc-min()-X',
'fBodyAcc-min()-Y',
'fBodyAcc-min()-Z',
'fBodyAcc-sma()',
'fBodyAcc-energy()-X',
'fBodyAcc-energy()-Y',
'fBodyAcc-energy()-Z',
'fBodyAcc-iqr()-X',
'fBodyAcc-iqr()-Y',
'fBodyAcc-iqr()-Z',
'fBodyAcc-entropy()-X',
'fBodyAcc-entropy()-Y',
'fBodyAcc-entropy()-Z',
'fBodyAcc-maxInds-X',
'fBodyAcc-maxInds-Y',
'fBodyAcc-maxInds-Z',
'fBodyAcc-meanFreq()-X',
'fBodyAcc-meanFreq()-Y',
'fBodyAcc-meanFreq()-Z',
'fBodyAcc-skewness()-X',
'fBodyAcc-kurtosis()-X',
'fBodyAcc-skewness()-Y',
'fBodyAcc-kurtosis()-Y',
'fBodyAcc-skewness()-Z',
'fBodyAcc-kurtosis()-Z',
'fBodyAcc-bandsEnergy()-1,8',
'fBodyAcc-bandsEnergy()-9,16',
'fBodyAcc-bandsEnergy()-17,24',
'fBodyAcc-bandsEnergy()-25,32',
'fBodyAcc-bandsEnergy()-33,40',
'fBodyAcc-bandsEnergy()-41,48',
'fBodyAcc-bandsEnergy()-49,56',
'fBodyAcc-bandsEnergy()-57,64',
'fBodyAcc-bandsEnergy()-1,16',
'fBodyAcc-bandsEnergy()-17,32',
'fBodyAcc-bandsEnergy()-33,48',
'fBodyAcc-bandsEnergy()-49,64',
'fBodyAcc-bandsEnergy()-1,24',
'fBodyAcc-bandsEnergy()-25,48',
'fBodyAcc-bandsEnergy()-1,8',
'fBodyAcc-bandsEnergy()-9,16',
'fBodyAcc-bandsEnergy()-17,24',
'fBodyAcc-bandsEnergy()-25,32',
'fBodyAcc-bandsEnergy()-33,40',
'fBodyAcc-bandsEnergy()-41,48',
'fBodyAcc-bandsEnergy()-49,56',
'fBodyAcc-bandsEnergy()-57,64',
'fBodyAcc-bandsEnergy()-1,16',
'fBodyAcc-bandsEnergy()-17,32',
'fBodyAcc-bandsEnergy()-33,48',
'fBodyAcc-bandsEnergy()-49,64',
'fBodyAcc-bandsEnergy()-1,24',
'fBodyAcc-bandsEnergy()-25,48',
'fBodyAcc-bandsEnergy()-1,8',
'fBodyAcc-bandsEnergy()-9,16',
'fBodyAcc-bandsEnergy()-17,24',
'fBodyAcc-bandsEnergy()-25,32',
'fBodyAcc-bandsEnergy()-33,40',
'fBodyAcc-bandsEnergy()-41,48',
'fBodyAcc-bandsEnergy()-49,56',
'fBodyAcc-bandsEnergy()-57,64',
'fBodyAcc-bandsEnergy()-1,16',
'fBodyAcc-bandsEnergy()-17,32',
'fBodyAcc-bandsEnergy()-33,48',
'fBodyAcc-bandsEnergy()-49,64',
'fBodyAcc-bandsEnergy()-1,24',
'fBodyAcc-bandsEnergy()-25,48',
'fBodyAccJerk-mean()-X',
'fBodyAccJerk-mean()-Y',
'fBodyAccJerk-mean()-Z',
```

```
'fBodyAccJerk-std()-X',
'fBodyAccJerk-std()-Y',
'fBodyAccJerk-std()-Z',
'fBodyAccJerk-mad()-X',
'fBodyAccJerk-mad()-Y',
'fBodyAccJerk-mad()-Z',
'fBodyAccJerk-max()-X',
'fBodyAccJerk-max()-Y',
'fBodyAccJerk-max()-Z',
'fBodyAccJerk-min()-X',
'fBodyAccJerk-min()-Y',
'fBodyAccJerk-min()-Z',
'fBodyAccJerk-sma()',
'fBodyAccJerk-energy()-X',
'fBodyAccJerk-energy()-Y',
'fBodyAccJerk-energy()-Z',
'fBodyAccJerk-iqr()-X',
'fBodyAccJerk-iqr()-Y',
'fBodyAccJerk-iqr()-Z',
'fBodyAccJerk-entropy()-X',
'fBodyAccJerk-entropy()-Y',
'fBodyAccJerk-entropy()-Z',
'fBodyAccJerk-maxInds-X',
'fBodyAccJerk-maxInds-Y',
'fBodyAccJerk-maxInds-Z',
'fBodyAccJerk-meanFreq()-X',
'fBodyAccJerk-meanFreq()-Y',
'fBodyAccJerk-meanFreq()-Z',
'fBodyAccJerk-skewness()-X',
'fBodyAccJerk-kurtosis()-X',
'fBodyAccJerk-skewness()-Y',
'fBodyAccJerk-kurtosis()-Y',
'fBodyAccJerk-skewness()-Z',
'fBodyAccJerk-kurtosis()-Z',
'fBodyAccJerk-bandsEnergy()-1,8',
'fBodyAccJerk-bandsEnergy()-9,16',
'fBodyAccJerk-bandsEnergy()-17,24',
'fBodyAccJerk-bandsEnergy()-25,32',
'fBodyAccJerk-bandsEnergy()-33,40',
'fBodyAccJerk-bandsEnergy()-41,48',
'fBodyAccJerk-bandsEnergy()-49,56',
'fBodyAccJerk-bandsEnergy()-57,64',
'fBodyAccJerk-bandsEnergy()-1,16',
'fBodyAccJerk-bandsEnergy()-17,32',
'fBodyAccJerk-bandsEnergy()-33,48',
'fBodyAccJerk-bandsEnergy()-49,64',
'fBodyAccJerk-bandsEnergy()-1,24',
'fBodyAccJerk-bandsEnergy()-25,48',
'fBodyAccJerk-bandsEnergy()-1,8',
'fBodyAccJerk-bandsEnergy()-9,16',
'fBodyAccJerk-bandsEnergy()-17,24',
'fBodyAccJerk-bandsEnergy()-25,32',
'fBodyAccJerk-bandsEnergy()-33,40',
'fBodyAccJerk-bandsEnergy()-41,48',
'fBodyAccJerk-bandsEnergy()-49,56',
'fBodyAccJerk-bandsEnergy()-57,64',
'fBodyAccJerk-bandsEnergy()-1,16',
'fBodyAccJerk-bandsEnergy()-17,32',
'fBodyAccJerk-bandsEnergy()-33,48',
'fBodyAccJerk-bandsEnergy()-49,64',
'fBodyAccJerk-bandsEnergy()-1,24',
'fBodyAccJerk-bandsEnergy()-25,48',
'fBodyAccJerk-bandsEnergy()-1,8',
'fBodyAccJerk-bandsEnergy()-9,16',
'fBodyAccJerk-bandsEnergy()-17,24',
'fBodyAccJerk-bandsEnergy()-25,32',
'fBodyAccJerk-bandsEnergy()-33,40',
'fBodyAccJerk-bandsEnergy()-41,48',
'fBodyAccJerk-bandsEnergy()-49,56',
'fBodyAccJerk-bandsEnergy()-57,64',
'fBodyAccJerk-bandsEnergy()-1,16',
'fBodyAccJerk-bandsEnergy()-17,32',
'fBodyAccJerk-bandsEnergy()-33,48',
'fBodyAccJerk-bandsEnergy()-49,64',
'fBodyAccJerk-bandsEnergy()-1,24',
'fBodyAccJerk-bandsEnergy()-25,48',
'fBodyGyro-mean()-X',
```

```
'fBodyGyro-mean()-Y',
'fBodyGyro-mean()-Z',
'fBodyGyro-std()-X',
'fBodyGyro-std()-Y',
'fBodyGyro-std()-Z',
'fBodyGyro-mad()-X',
'fBodyGyro-mad()-Y',
'fBodyGyro-mad()-Z',
'fBodyGyro-max()-X',
'fBodyGyro-max()-Y',
'fBodyGyro-max()-Z',
'fBodyGyro-min()-X',
'fBodyGyro-min()-Y',
'fBodyGyro-min()-Z',
'fBodyGyro-sma()',
'fBodyGyro-energy()-X',
'fBodyGyro-energy()-Y',
'fBodyGyro-energy()-Z',
'fBodyGyro-iqr()-X',
'fBodyGyro-iqr()-Y',
'fBodyGyro-iqr()-Z',
'fBodyGyro-entropy()-X',
'fBodyGyro-entropy()-Y',
'fBodyGyro-entropy()-Z',
'fBodyGyro-maxInds-X',
'fBodyGyro-maxInds-Y',
'fBodyGyro-maxInds-Z',
'fBodyGyro-meanFreq()-X',
'fBodyGyro-meanFreq()-Y',
'fBodyGyro-meanFreq()-Z',
'fBodyGyro-skewness()-X',
'fBodyGyro-kurtosis()-X',
'fBodyGyro-skewness()-Y',
'fBodyGyro-kurtosis()-Y',
'fBodyGyro-skewness()-Z',
'fBodyGyro-kurtosis()-Z',
'fBodyGyro-bandsEnergy()-1,8',
'fBodyGyro-bandsEnergy()-9,16',
'fBodyGyro-bandsEnergy()-17,24',
'fBodyGyro-bandsEnergy()-25,32',
'fBodyGyro-bandsEnergy()-33,40',
'fBodyGyro-bandsEnergy()-41,48',
'fBodyGyro-bandsEnergy()-49,56',
'fBodyGyro-bandsEnergy()-57,64',
'fBodyGyro-bandsEnergy()-1,16',
'fBodyGyro-bandsEnergy()-17,32',
'fBodyGyro-bandsEnergy()-33,48',
'fBodyGyro-bandsEnergy()-49,64',
'fBodyGyro-bandsEnergy()-1,24',
'fBodyGyro-bandsEnergy()-25,48',
'fBodyGyro-bandsEnergy()-1,8',
'fBodyGyro-bandsEnergy()-9,16',
'fBodyGyro-bandsEnergy()-17,24',
'fBodyGyro-bandsEnergy()-25,32',
'fBodyGyro-bandsEnergy()-33,40',
'fBodyGyro-bandsEnergy()-41,48',
'fBodyGyro-bandsEnergy()-49,56',
'fBodyGyro-bandsEnergy()-57,64',
'fBodyGyro-bandsEnergy()-1,16',
'fBodyGyro-bandsEnergy()-17,32',
'fBodyGyro-bandsEnergy()-33,48',
'fBodyGyro-bandsEnergy()-49,64',
'fBodyGyro-bandsEnergy()-1,24',
'fBodyGyro-bandsEnergy()-25,48',
'fBodyGyro-bandsEnergy()-1,8',
'fBodyGyro-bandsEnergy()-9,16',
'fBodyGyro-bandsEnergy()-17,24',
'fBodyGyro-bandsEnergy()-25,32',
'fBodyGyro-bandsEnergy()-33,40',
'fBodyGyro-bandsEnergy()-41,48',
'fBodyGyro-bandsEnergy()-49,56',
'fBodyGyro-bandsEnergy()-57,64',
'fBodyGyro-bandsEnergy()-1,16',
'fBodyGyro-bandsEnergy()-17,32',
'fBodyGyro-bandsEnergy()-33,48',
'fBodyGyro-bandsEnergy()-49,64',
'fBodyGyro-bandsEnergy()-1,24',
```

```
'fBodyGyro-bandsEnergy()-25,48',
'fBodyAccMag-mean()',
'fBodyAccMag-std()',
'fBodyAccMag-mad()',
'fBodyAccMag-max()',
'fBodyAccMag-min()',
'fBodyAccMag-sma()',
'fBodyAccMag-energy()',
'fBodyAccMag-iqr()',
'fBodyAccMag-entropy()',
'fBodyAccMag-maxInds',
'fBodyAccMag-meanFreq()',
'fBodyAccMag-skewness()',
'fBodyAccMag-kurtosis()',
'fBodyBodyAccJerkMag-mean()',
'fBodyBodyAccJerkMag-std()',
'fBodyBodyAccJerkMag-mad()',
'fBodyBodyAccJerkMag-max()',
'fBodyBodyAccJerkMag-min()',
'fBodyBodyAccJerkMag-sma()',
'fBodyBodyAccJerkMag-energy()',
'fBodyBodyAccJerkMag-iqr()',
'fBodyBodyAccJerkMag-entropy()',
'fBodyBodyAccJerkMag-maxInds',
'fBodyBodyAccJerkMag-meanFreq()',
'fBodyBodyAccJerkMag-skewness()',
'fBodyBodyAccJerkMag-kurtosis()',
'fBodyBodyGyroMag-mean()',
'fBodyBodyGyroMag-std()',
'fBodyBodyGyroMag-mad()',
'fBodyBodyGyroMag-max()',
'fBodyBodyGyroMag-min()',
'fBodyBodyGyroMag-sma()',
'fBodyBodyGyroMag-energy()',
'fBodyBodyGyroMag-iqr()',
'fBodyBodyGyroMag-entropy()',
'fBodyBodyGyroMag-maxInds',
'fBodyBodyGyroMag-meanFreq()',
'fBodyBodyGyroMag-skewness()',
'fBodyBodyGyroMag-kurtosis()',
'fBodyBodyGyroJerkMag-mean()',
'fBodyBodyGyroJerkMag-std()',
'fBodyBodyGyroJerkMag-mad()',
'fBodyBodyGyroJerkMag-max()',
'fBodyBodyGyroJerkMag-min()',
'fBodyBodyGyroJerkMag-sma()',
'fBodyBodyGyroJerkMag-energy()',
'fBodyBodyGyroJerkMag-iqr()',
'fBodyBodyGyroJerkMag-entropy()',
'fBodyBodyGyroJerkMag-maxInds',
'fBodyBodyGyroJerkMag-meanFreq()',
'fBodyBodyGyroJerkMag-skewness()',
'fBodyBodyGyroJerkMag-kurtosis()',
'angle(tBodyAccMean,gravity)',
'angle(tBodyAccJerkMean),gravityMean)',
'angle(tBodyGyroMean,gravityMean)',
'angle(tBodyGyroJerkMean,gravityMean)',
'angle(X,gravityMean)',
'angle(Y,gravityMean)',
'angle(Z,gravityMean)']
```

1.1 Training and Test data

In [ ]:

```
# get the data from txt files to pandas dataffame
X_train = pd.read_csv('/content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/train/X_train.txt', delim_whitespace=True, header=None,
names=features)

# add subject column to the dataframe
X_train['subject'] = pd.read_csv('/content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/train/subject_train.txt', header=None, squeeze=True)

y_train = pd.read_csv('/content/drive/My Drive/Assign - 23 Human Activity
```

```
Recognition/HAR/UCI_HAR_Dataset/train/y_train.txt', names=['Activity'], squeeze=True)
y_train_labels = y_train.map({1: 'WALKING', 2:'WALKING_UPSTAIRS',3:'WALKING_DOWNSTAIRS',\
                        4:'SITTING', 5:'STANDING',6:'LAYING'})

# put all columns in a single dataframe
train = X_train
train['Activity'] = y_train
train['ActivityName'] = y_train_labels
```

In [ ]:

```
# get the data from txt files to pandas datafframe
X_test = pd.read_csv('UCI_HAR_dataset/test/X_test.txt', delim_whitespace=True, header=None, names=f
eatures)

# add subject column to the dataframe
X_test['subject'] = pd.read_csv('UCI_HAR_dataset/test/subject_test.txt', header=None, squeeze=True)

# get y labels from the txt file
y_test = pd.read_csv('UCI_HAR_dataset/test/y_test.txt', names=['Activity'], squeeze=True)
y_test_labels = y_test.map({1: 'WALKING', 2:'WALKING_UPSTAIRS',3:'WALKING_DOWNSTAIRS',\
                        4:'SITTING', 5:'STANDING',6:'LAYING'})


# put all columns in a single dataframe
test = X_test
test['Activity'] = y_test
test['ActivityName'] = y_test_labels
```

1.2 Changing features names

In [ ]:

```
columns = train.columns

# Removing '()' from column names
columns = columns.str.replace('[()]','')
columns = columns.str.replace('[-]', '')
columns = columns.str.replace('[,]','')

train.columns = columns
test.columns = columns
```

In [ ]:

```
train = pd.read_csv('/content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/csv_files/train.csv')
train.head()
```

Out[ ]:

	tBodyAccmeanX	tBodyAccmeanY	tBodyAccmeanZ	tBodyAccstdX	tBodyAccstdY	tBodyAccstdZ	tBodyAccmadX	tBodyAccmadY	t
0	0.288585	-0.020294	-0.132905	-0.995279	-0.983111	-0.913526	-0.995112	-0.983185	
1	0.278419	-0.016411	-0.123520	-0.998245	-0.975300	-0.960322	-0.998807	-0.974914	
2	0.279653	-0.019467	-0.113462	-0.995380	-0.967187	-0.978944	-0.996520	-0.963668	
3	0.279174	-0.026201	-0.123283	-0.996091	-0.983403	-0.990675	-0.997099	-0.982750	
4	0.276629	-0.016570	-0.115362	-0.998139	-0.980817	-0.990482	-0.998321	-0.979672	

5 rows × 564 columns

In [ ]:

```
test = pd.read_csv('/content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/csv_files/train.csv')
test.head()
```

Out[ ]:

	tBodyAccmeanX	tBodyAccmeanY	tBodyAccmeanZ	tBodyAccstdX	tBodyAccstdY	tBodyAccstdZ	tBodyAccmadX	tBodyAccmadY	t
0	0.288585	-0.020294	-0.132905	-0.995279	-0.983111	-0.913526	-0.995112	-0.983185	
1	0.278419	-0.016411	-0.123520	-0.998245	-0.975300	-0.960322	-0.998807	-0.974914	
2	0.279653	-0.019467	-0.113462	-0.995380	-0.967187	-0.978944	-0.996520	-0.963668	
3	0.279174	-0.026201	-0.123283	-0.996091	-0.983403	-0.990675	-0.997099	-0.982750	
4	0.276629	-0.016570	-0.115362	-0.998139	-0.980817	-0.990482	-0.998321	-0.979672	

5 rows × 564 columns

1.3 Checking for duplicates

In [ ]:

```
print('The number of duplicates in training data:', sum(train.duplicated()))
print('The number of duplicates in testing data:', sum(test.duplicated()))
```

```
The number of duplicates in training data: 0
The number of duplicates in testing data: 0
```

1.4 Checking for null values

In [ ]:

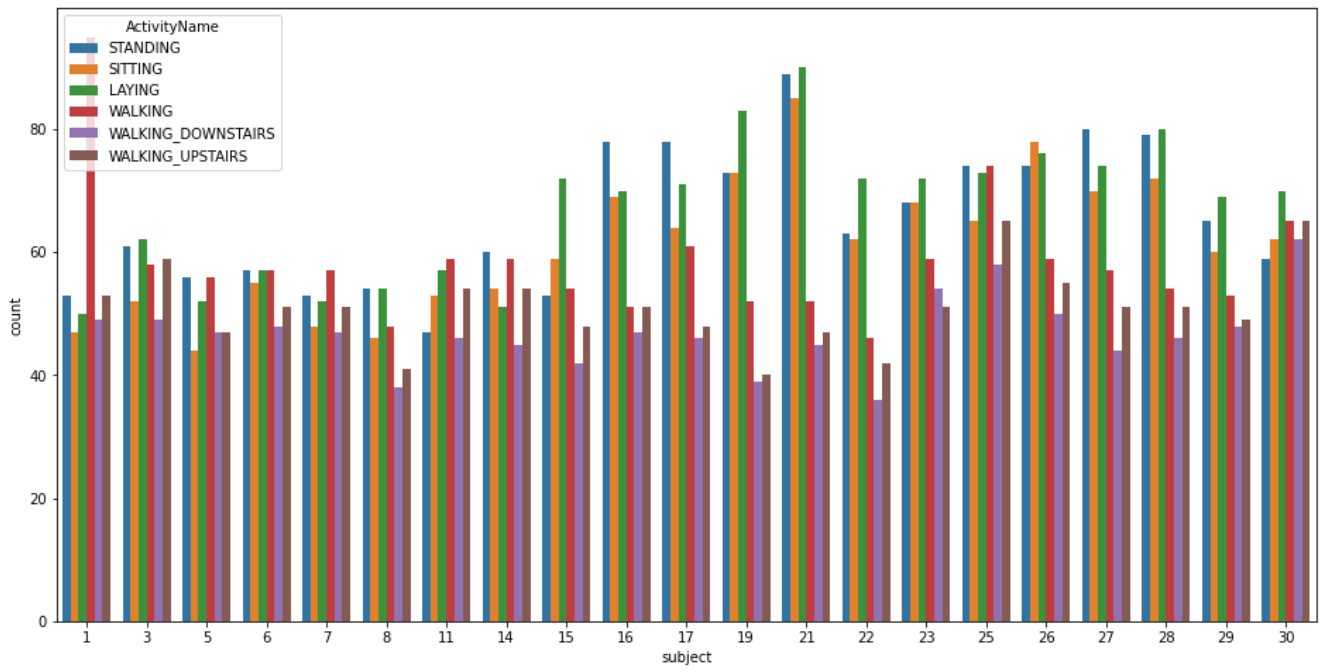
```
print(train.isna().values.sum())
print(test.isna().values.sum())
```

```
0
0
```

1.5 Checking for data imbalance

In [ ]:

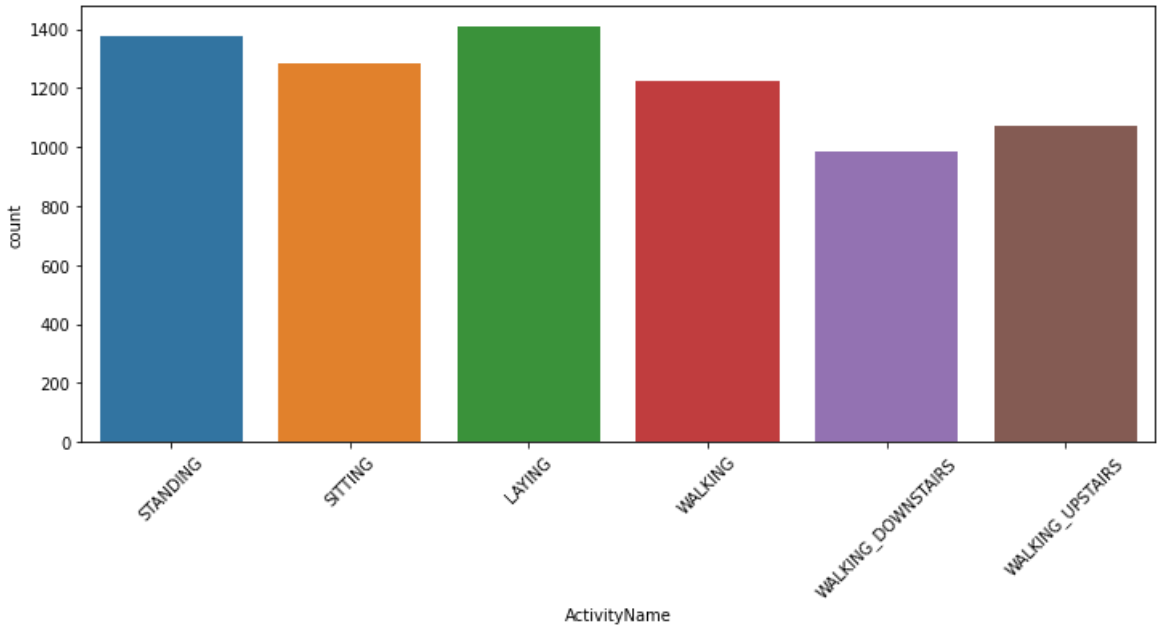
```
plt.figure(figsize=(16,8))
sns.countplot(x = 'subject', hue='ActivityName', data=train)
plt.show()
```



1.6 Checking the distribution of classes

In [ ]:

```
plt.figure(figsize=(12,5))
sns.countplot(x='ActivityName', data=train)
plt.xticks(rotation=45)
plt.show()
```

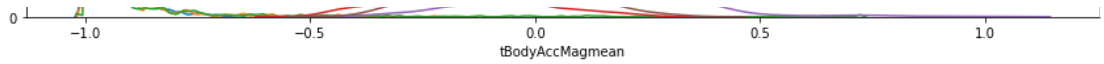


2.Univariate Analysis

In [ ]:

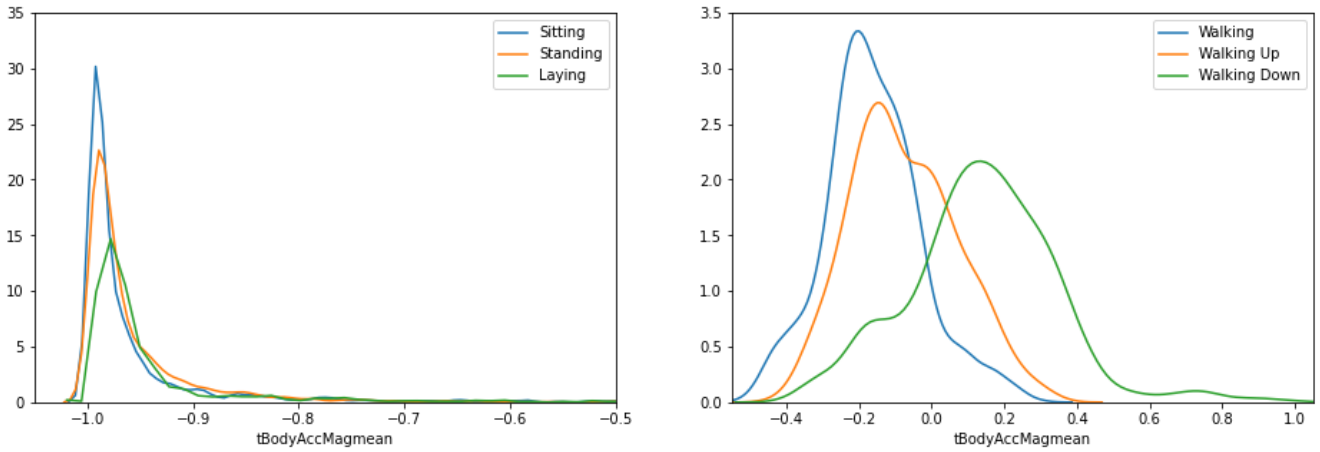
```
#plt.figure(figsize=(12,5))
facetGrid = sns.FacetGrid(train, hue='ActivityName', size=6, aspect=2)
facetGrid.map(sns.distplot, 'tBodyAccMagmean', hist=False).add_legend()

plt.annotate("Stationary Activities", xy=(-0.956,17), xytext=(-0.9, 23), size=20, arrowprops= dict(
arrowstyle="simple",connectionstyle="arc3,rad=0.1"))
plt.annotate("Moving Activities", xy=(0,3), xytext=(0.2, 9), size=20, arrowprops= dict(arrowstyle="
simple",connectionstyle="arc3,rad=0.1"))

plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/axisgrid.py:243: UserWarning: The `size` parameter
has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```



In [ ]:

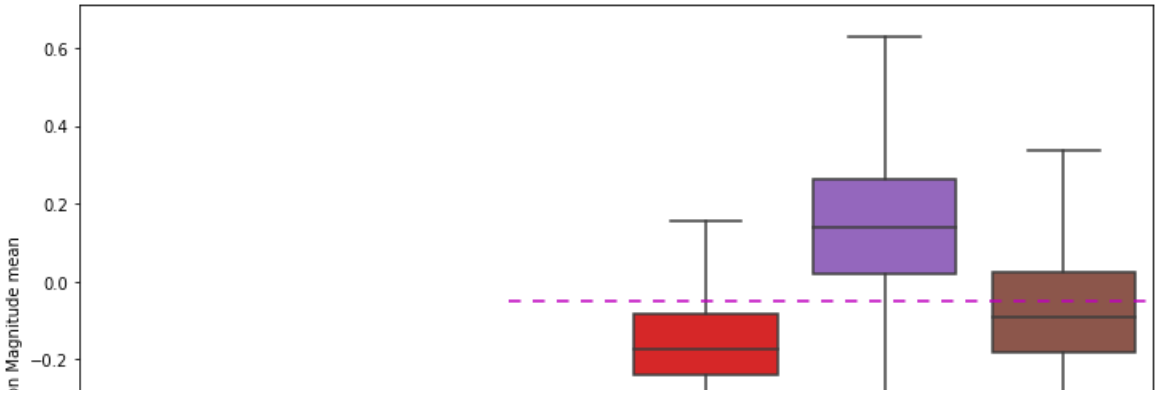
```
df1 = train[train['Activity']==1]
df2 = train[train['Activity']==2]
df3 = train[train['Activity']==3]
df4 = train[train['Activity']==4]
df5 = train[train['Activity']==5]
df6 = train[train['Activity']==6]

plt.figure(figsize=(16,5))
plt.subplot(1,2,1)
sns.distplot(df4['tBodyAccMagmean'], hist=False, label='Sitting')
sns.distplot(df5['tBodyAccMagmean'], hist=False, label='Standing')
sns.distplot(df6['tBodyAccMagmean'], hist=False, label='Laying')
plt.axis([-1.05, -0.5, 0, 35])
plt.legend()

plt.subplot(1,2,2)
sns.distplot(df1['tBodyAccMagmean'], hist=False, label='Walking')
sns.distplot(df2['tBodyAccMagmean'], hist=False, label='Walking Up')
sns.distplot(df3['tBodyAccMagmean'], hist=False, label='Walking Down')
plt.axis([-0.55, 1.05, 0, 3.5])
plt.legend()
plt.show()
```

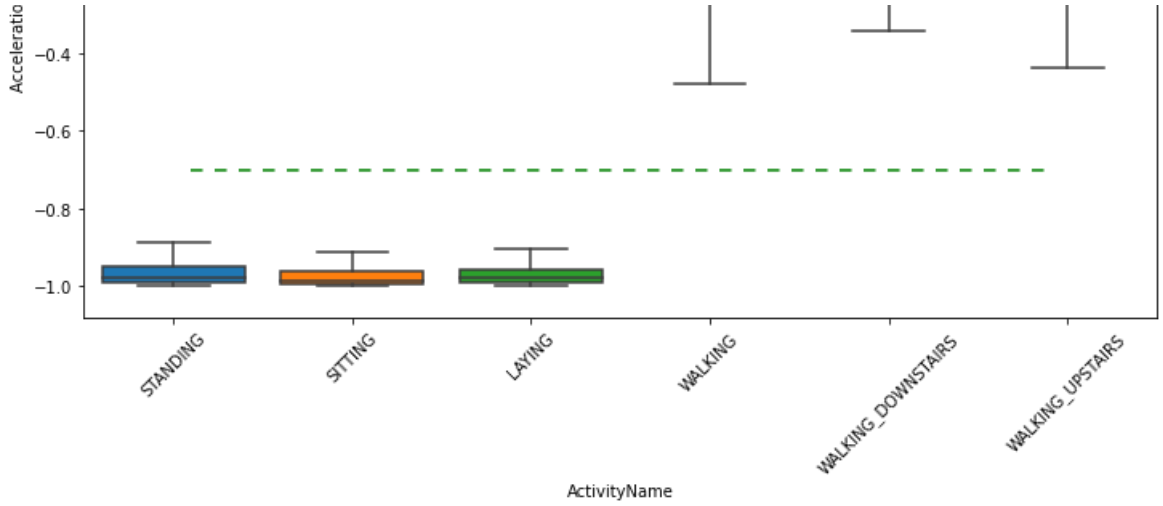


2.1 Mangnitude of an acceleration

In [ ]:

```
plt.figure(figsize=(12,8))
sns.boxplot(x='ActivityName', y='tBodyAccMagmean', data=train, showfliers=False, saturation=1)
plt.ylabel('Acceleration Magnitude mean')
plt.axhline(y=-0.7, xmin=0.1, xmax=0.9,dashes=(5,5), c='g')
plt.axhline(y=-0.05, xmin=0.4, dashes=(5,5), c='m')
plt.xticks(rotation=45)
plt.show()
```

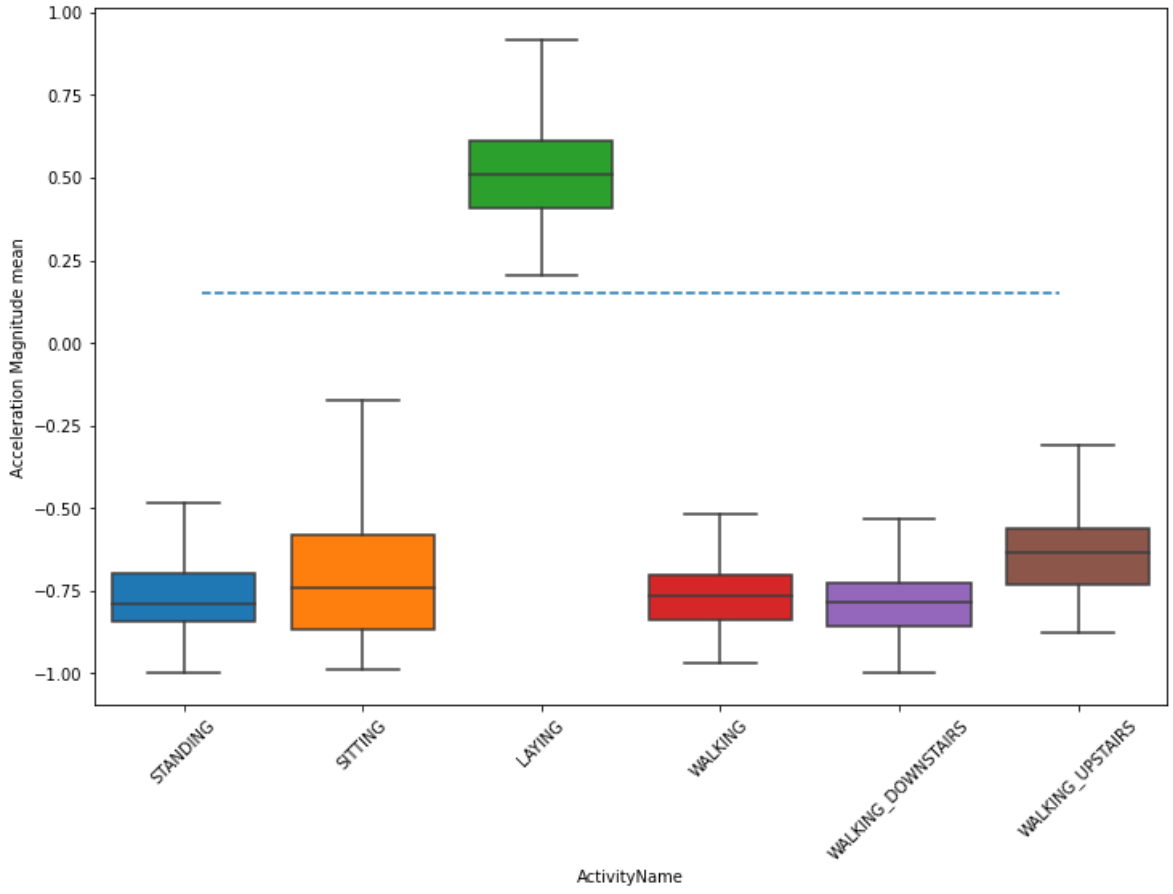




2.2 Position of gravity

In [ ]:

```
plt.figure(figsize=(12,8))
sns.boxplot(x='ActivityName', y='angleXgravityMean', data=train, showfliers=False, saturation=1)
plt.ylabel('Acceleration Magnitude mean')
plt.axhline(y=0.15, xmin=0.1, xmax=0.9, linestyle='--')
plt.xticks(rotation=45)
plt.show()
```



3.Applying t-SNE

In [ ]:

```
from sklearn.manifold import TSNE
```

In [ ]:

```
def perform_tsne(X_data, y_data, perplexities, n_iter=1000):

    for index,perplexity in enumerate(perplexities):
        tsne= TSNE(n_components=2, verbose=2, perplexity=perplexity)
        X_reduced = tsne.fit_transform(X_data)

        #creating a new df for plotting
        df = pd.DataFrame({'x1':X_reduced[:,0],
                           'x2':X_reduced[:,1],
                           'label':y_data})

        #plot lm plot mainily used after regression done (for plotting with optimum line)
        sns.lmplot(data=df, x='x1', y='x2', hue='label', fit_reg=False, size=8,
palette="Set1",markers=['^','v','s','o', '1','2'])
        plt.title("perplexity : {} and max_iter : {}".format(perplexity, n_iter))
        print('saving this plot as image in present working directory...')
        plt.show()
        print('Done')
```

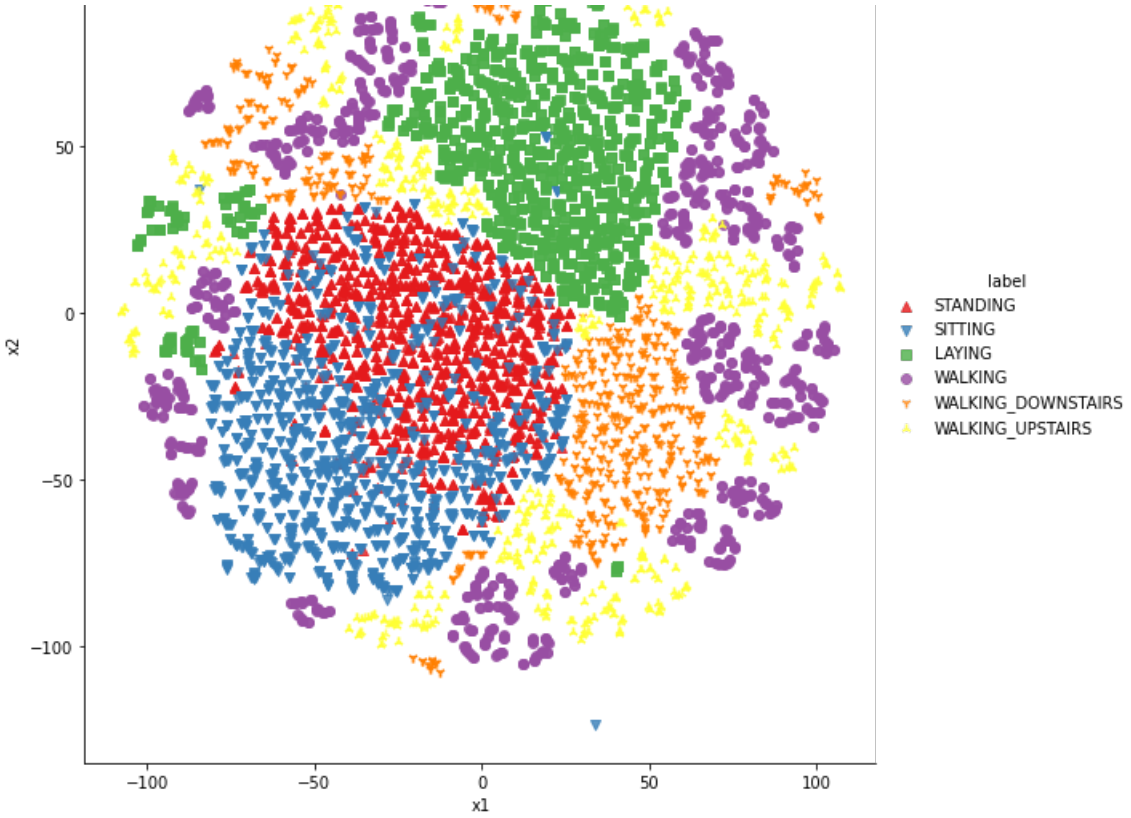
In [ ]:

```
X_pre_tsne = train.drop(['subject', 'Activity','ActivityName'], axis=1)
y_pre_tsne = train['ActivityName']
perform_tsne(X_data = X_pre_tsne,y_data=y_pre_tsne, perplexities =[2,5,10,20,50])
```

```
[t-SNE] Computing 7 nearest neighbors...
[t-SNE] Indexed 7352 samples in 0.470s...
[t-SNE] Computed neighbors for 7352 samples in 39.318s...
[t-SNE] Computed conditional probabilities for sample 1000 / 7352
[t-SNE] Computed conditional probabilities for sample 2000 / 7352
[t-SNE] Computed conditional probabilities for sample 3000 / 7352
[t-SNE] Computed conditional probabilities for sample 4000 / 7352
[t-SNE] Computed conditional probabilities for sample 5000 / 7352
[t-SNE] Computed conditional probabilities for sample 6000 / 7352
[t-SNE] Computed conditional probabilities for sample 7000 / 7352
[t-SNE] Computed conditional probabilities for sample 7352 / 7352
[t-SNE] Mean sigma: 0.635854
[t-SNE] Computed conditional probabilities in 0.039s
[t-SNE] Iteration 50: error = 124.7889252, gradient norm = 0.0259888 (50 iterations in 6.079s)
[t-SNE] Iteration 100: error = 107.2060394, gradient norm = 0.0268191 (50 iterations in 3.206s)
[t-SNE] Iteration 150: error = 100.9500732, gradient norm = 0.0196198 (50 iterations in 2.336s)
[t-SNE] Iteration 200: error = 97.5597992, gradient norm = 0.0134663 (50 iterations in 2.208s)
[t-SNE] Iteration 250: error = 95.2138290, gradient norm = 0.0137577 (50 iterations in 2.208s)
[t-SNE] KL divergence after 250 iterations with early exaggeration: 95.213829
[t-SNE] Iteration 300: error = 4.1161933, gradient norm = 0.0015603 (50 iterations in 1.952s)
[t-SNE] Iteration 350: error = 3.2078633, gradient norm = 0.0010180 (50 iterations in 1.815s)
[t-SNE] Iteration 400: error = 2.7788248, gradient norm = 0.0007109 (50 iterations in 1.870s)
[t-SNE] Iteration 450: error = 2.5152574, gradient norm = 0.0005720 (50 iterations in 1.874s)
[t-SNE] Iteration 500: error = 2.3322468, gradient norm = 0.0004793 (50 iterations in 1.865s)
[t-SNE] Iteration 550: error = 2.1944561, gradient norm = 0.0004137 (50 iterations in 1.935s)
[t-SNE] Iteration 600: error = 2.0852098, gradient norm = 0.0003691 (50 iterations in 1.930s)
[t-SNE] Iteration 650: error = 1.9956943, gradient norm = 0.0003266 (50 iterations in 1.952s)
[t-SNE] Iteration 700: error = 1.9199437, gradient norm = 0.0003021 (50 iterations in 1.940s)
[t-SNE] Iteration 750: error = 1.8549099, gradient norm = 0.0002766 (50 iterations in 1.949s)
[t-SNE] Iteration 800: error = 1.7983253, gradient norm = 0.0002560 (50 iterations in 1.983s)
[t-SNE] Iteration 850: error = 1.7482818, gradient norm = 0.0002413 (50 iterations in 1.928s)
[t-SNE] Iteration 900: error = 1.7036929, gradient norm = 0.0002254 (50 iterations in 1.937s)
[t-SNE] Iteration 950: error = 1.6635983, gradient norm = 0.0002115 (50 iterations in 1.943s)
[t-SNE] Iteration 1000: error = 1.6274554, gradient norm = 0.0002008 (50 iterations in 1.937s)
[t-SNE] KL divergence after 1000 iterations: 1.627455
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/regression.py:573: UserWarning: The `size`
parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```

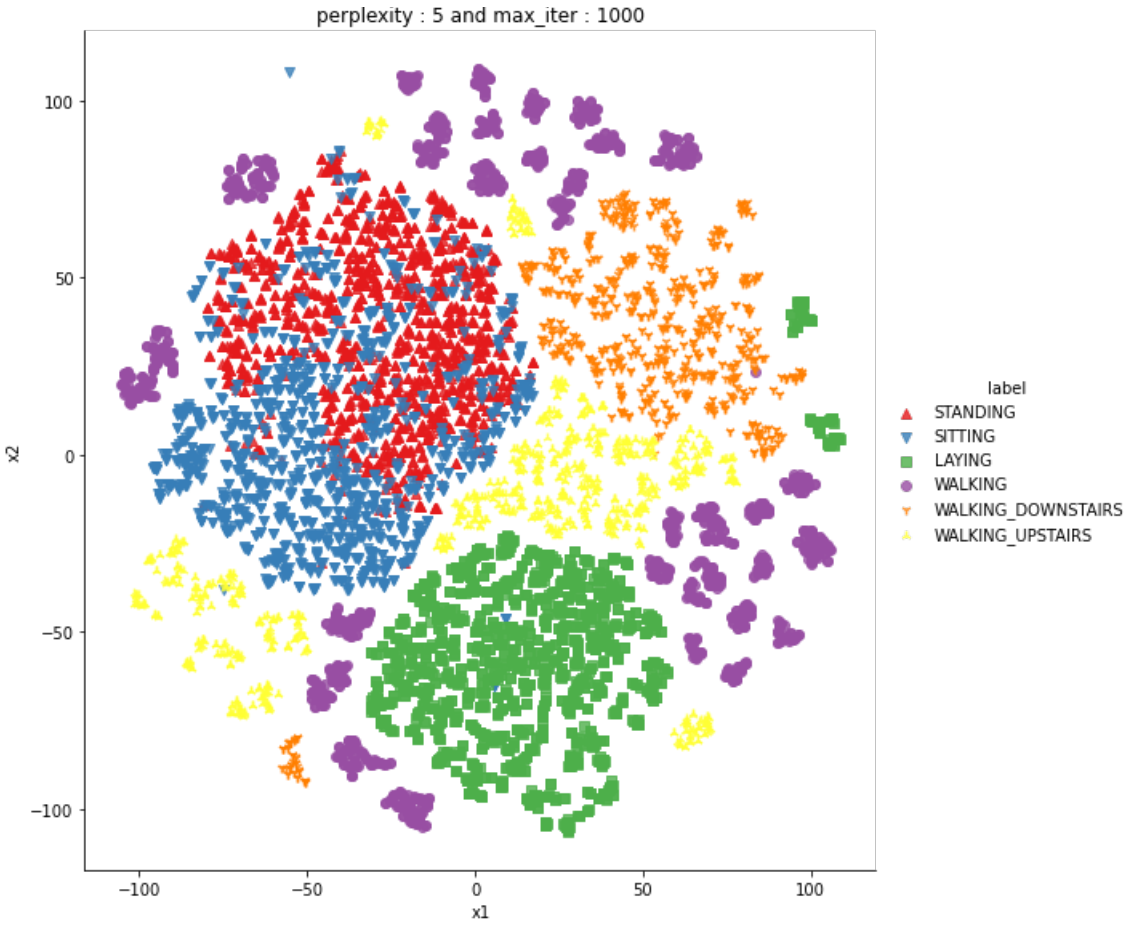
```
saving this plot as image in present working directory...
```




```
Done
[t-SNE] Computing 16 nearest neighbors...
[t-SNE] Indexed 7352 samples in 0.627s...
[t-SNE] Computed neighbors for 7352 samples in 39.817s...
[t-SNE] Computed conditional probabilities for sample 1000 / 7352
[t-SNE] Computed conditional probabilities for sample 2000 / 7352
[t-SNE] Computed conditional probabilities for sample 3000 / 7352
[t-SNE] Computed conditional probabilities for sample 4000 / 7352
[t-SNE] Computed conditional probabilities for sample 5000 / 7352
[t-SNE] Computed conditional probabilities for sample 6000 / 7352
[t-SNE] Computed conditional probabilities for sample 7000 / 7352
[t-SNE] Computed conditional probabilities for sample 7352 / 7352
[t-SNE] Mean sigma: 0.961265
[t-SNE] Computed conditional probabilities in 0.067s
[t-SNE] Iteration 50: error = 114.0185089, gradient norm = 0.0218635 (50 iterations in 5.959s)
[t-SNE] Iteration 100: error = 97.7227173, gradient norm = 0.0154143 (50 iterations in 2.253s)
[t-SNE] Iteration 150: error = 93.2074738, gradient norm = 0.0098429 (50 iterations in 1.956s)
[t-SNE] Iteration 200: error = 91.1990356, gradient norm = 0.0067135 (50 iterations in 1.922s)
[t-SNE] Iteration 250: error = 90.0177536, gradient norm = 0.0046048 (50 iterations in 1.926s)
[t-SNE] KL divergence after 250 iterations with early exaggeration: 90.017754
[t-SNE] Iteration 300: error = 3.5725126, gradient norm = 0.0014673 (50 iterations in 1.910s)
[t-SNE] Iteration 350: error = 2.8130083, gradient norm = 0.0007548 (50 iterations in 1.861s)
[t-SNE] Iteration 400: error = 2.4323199, gradient norm = 0.0005205 (50 iterations in 1.888s)
[t-SNE] Iteration 450: error = 2.2149634, gradient norm = 0.0004009 (50 iterations in 1.934s)
[t-SNE] Iteration 500: error = 2.0706520, gradient norm = 0.0003326 (50 iterations in 1.922s)
[t-SNE] Iteration 550: error = 1.9654067, gradient norm = 0.0002826 (50 iterations in 1.931s)
[t-SNE] Iteration 600: error = 1.8840668, gradient norm = 0.0002470 (50 iterations in 1.892s)
[t-SNE] Iteration 650: error = 1.8190072, gradient norm = 0.0002179 (50 iterations in 1.888s)
[t-SNE] Iteration 700: error = 1.7654541, gradient norm = 0.0001979 (50 iterations in 1.885s)
[t-SNE] Iteration 750: error = 1.7199605, gradient norm = 0.0001814 (50 iterations in 1.881s)
[t-SNE] Iteration 800: error = 1.6810946, gradient norm = 0.0001655 (50 iterations in 1.868s)
[t-SNE] Iteration 850: error = 1.6473318, gradient norm = 0.0001528 (50 iterations in 1.907s)
[t-SNE] Iteration 900: error = 1.6175196, gradient norm = 0.0001414 (50 iterations in 1.879s)
[t-SNE] Iteration 950: error = 1.5911634, gradient norm = 0.0001349 (50 iterations in 1.883s)
[t-SNE] Iteration 1000: error = 1.5674787, gradient norm = 0.0001273 (50 iterations in 1.901s)
[t-SNE] KL divergence after 1000 iterations: 1.567479
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/regression.py:573: UserWarning: The `size`
parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```

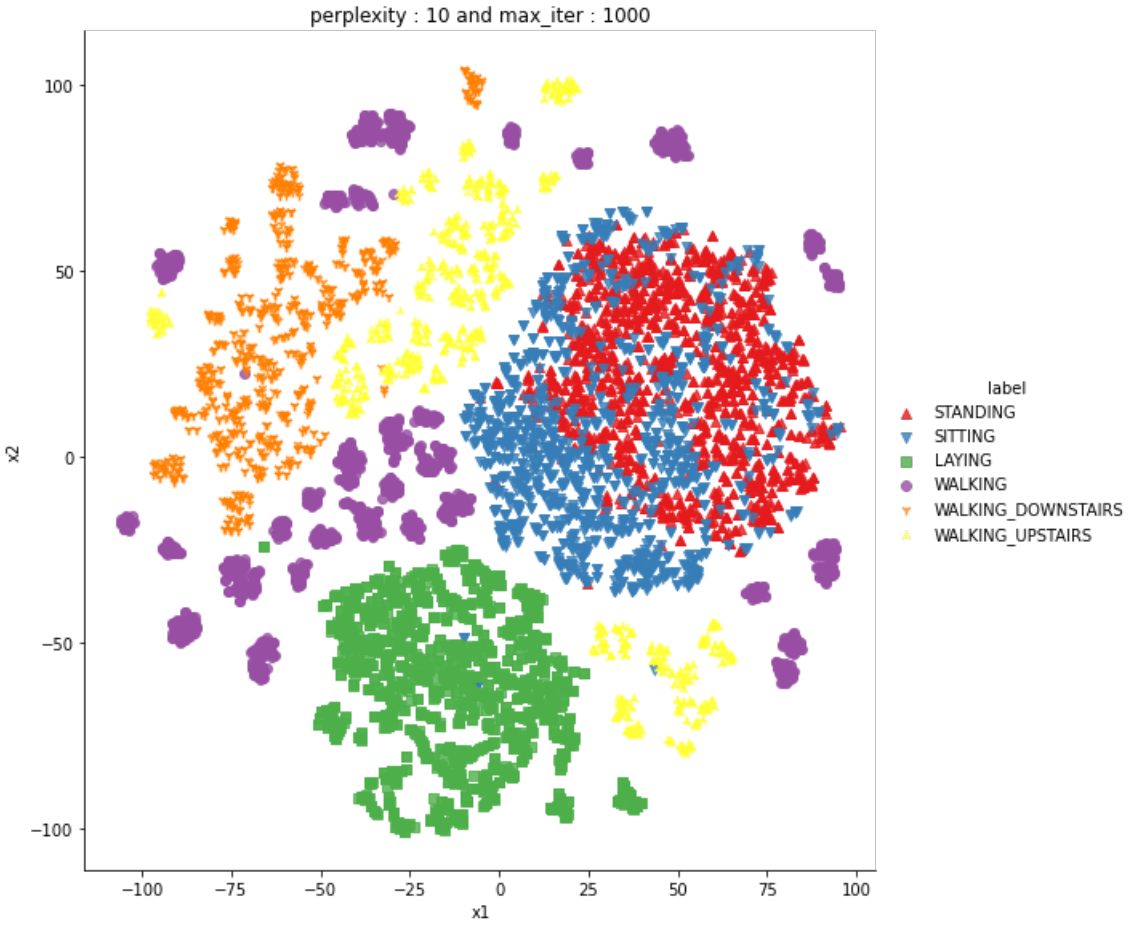
```
saving this plot as image in present working directory...
```




```
Done
[t-SNE] Computing 31 nearest neighbors...
[t-SNE] Indexed 7352 samples in 0.630s...
[t-SNE] Computed neighbors for 7352 samples in 40.574s...
[t-SNE] Computed conditional probabilities for sample 1000 / 7352
[t-SNE] Computed conditional probabilities for sample 2000 / 7352
[t-SNE] Computed conditional probabilities for sample 3000 / 7352
[t-SNE] Computed conditional probabilities for sample 4000 / 7352
[t-SNE] Computed conditional probabilities for sample 5000 / 7352
[t-SNE] Computed conditional probabilities for sample 6000 / 7352
[t-SNE] Computed conditional probabilities for sample 7000 / 7352
[t-SNE] Computed conditional probabilities for sample 7352 / 7352
[t-SNE] Mean sigma: 1.133828
[t-SNE] Computed conditional probabilities in 0.125s
[t-SNE] Iteration 50: error = 105.7421417, gradient norm = 0.0184827 (50 iterations in 4.469s)
[t-SNE] Iteration 100: error = 90.4412537, gradient norm = 0.0102276 (50 iterations in 2.565s)
[t-SNE] Iteration 150: error = 87.3087616, gradient norm = 0.0059609 (50 iterations in 2.104s)
[t-SNE] Iteration 200: error = 86.0333710, gradient norm = 0.0056068 (50 iterations in 2.051s)
[t-SNE] Iteration 250: error = 85.3234100, gradient norm = 0.0028100 (50 iterations in 2.036s)
[t-SNE] KL divergence after 250 iterations with early exaggeration: 85.323410
[t-SNE] Iteration 300: error = 3.1339159, gradient norm = 0.0013938 (50 iterations in 2.004s)
[t-SNE] Iteration 350: error = 2.4887671, gradient norm = 0.0006497 (50 iterations in 1.992s)
[t-SNE] Iteration 400: error = 2.1686134, gradient norm = 0.0004229 (50 iterations in 2.025s)
[t-SNE] Iteration 450: error = 1.9840684, gradient norm = 0.0003129 (50 iterations in 2.037s)
[t-SNE] Iteration 500: error = 1.8657923, gradient norm = 0.0002500 (50 iterations in 2.042s)
[t-SNE] Iteration 550: error = 1.7825463, gradient norm = 0.0002096 (50 iterations in 2.051s)
[t-SNE] Iteration 600: error = 1.7201005, gradient norm = 0.0001818 (50 iterations in 2.030s)
[t-SNE] Iteration 650: error = 1.6710354, gradient norm = 0.0001617 (50 iterations in 2.041s)
[t-SNE] Iteration 700: error = 1.6316726, gradient norm = 0.0001442 (50 iterations in 2.034s)
[t-SNE] Iteration 750: error = 1.5990770, gradient norm = 0.0001281 (50 iterations in 2.036s)
[t-SNE] Iteration 800: error = 1.5716115, gradient norm = 0.0001198 (50 iterations in 2.073s)
[t-SNE] Iteration 850: error = 1.5487018, gradient norm = 0.0001108 (50 iterations in 2.049s)
[t-SNE] Iteration 900: error = 1.5292275, gradient norm = 0.0001041 (50 iterations in 2.055s)
[t-SNE] Iteration 950: error = 1.5123469, gradient norm = 0.0000976 (50 iterations in 2.067s)
[t-SNE] Iteration 1000: error = 1.4976254, gradient norm = 0.0000919 (50 iterations in 2.061s)
[t-SNE] KL divergence after 1000 iterations: 1.497625
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/regression.py:573: UserWarning: The `size`
parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```

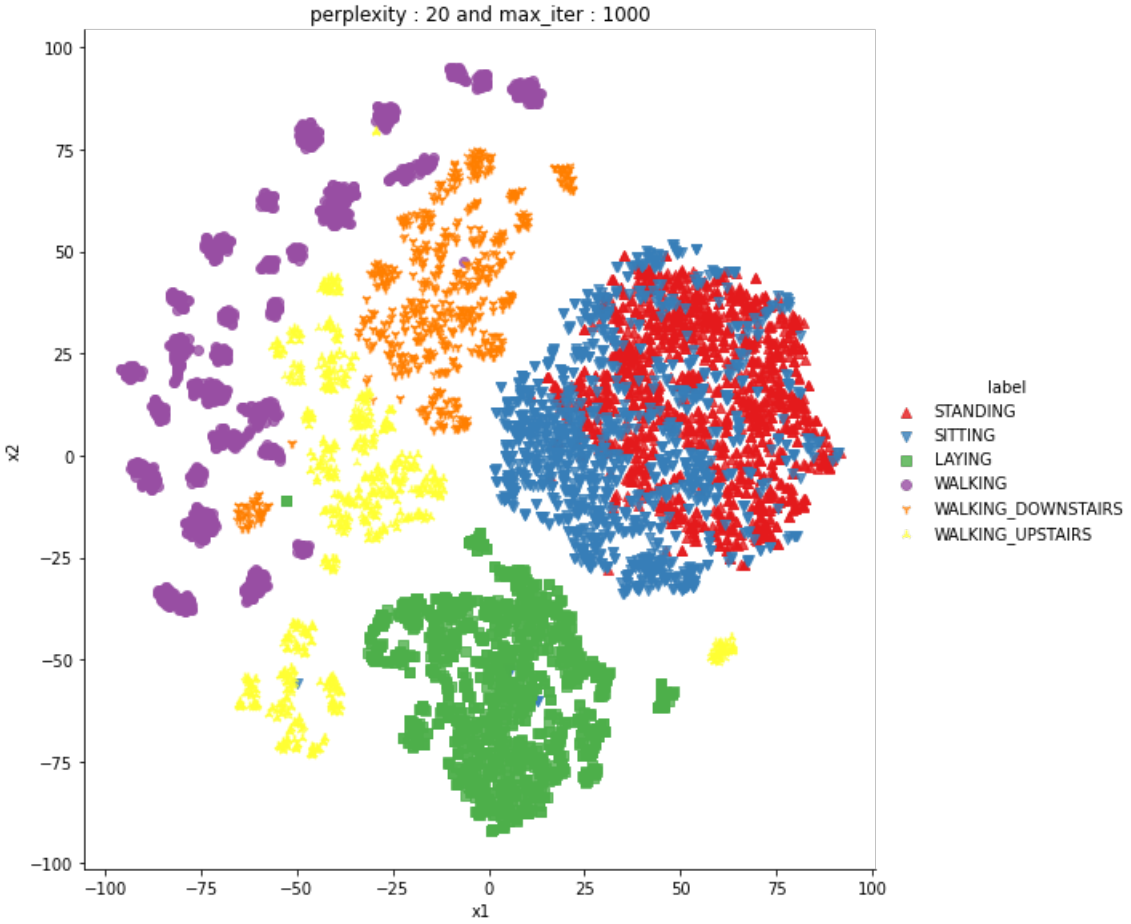
```
saving this plot as image in present working directory...
```




```
Done
[t-SNE] Computing 61 nearest neighbors...
[t-SNE] Indexed 7352 samples in 0.581s...
[t-SNE] Computed neighbors for 7352 samples in 41.287s...
[t-SNE] Computed conditional probabilities for sample 1000 / 7352
[t-SNE] Computed conditional probabilities for sample 2000 / 7352
[t-SNE] Computed conditional probabilities for sample 3000 / 7352
[t-SNE] Computed conditional probabilities for sample 4000 / 7352
[t-SNE] Computed conditional probabilities for sample 5000 / 7352
[t-SNE] Computed conditional probabilities for sample 6000 / 7352
[t-SNE] Computed conditional probabilities for sample 7000 / 7352
[t-SNE] Computed conditional probabilities for sample 7352 / 7352
[t-SNE] Mean sigma: 1.274335
[t-SNE] Computed conditional probabilities in 0.250s
[t-SNE] Iteration 50: error = 97.9168091, gradient norm = 0.0180808 (50 iterations in 3.376s)
[t-SNE] Iteration 100: error = 84.0771027, gradient norm = 0.0077034 (50 iterations in 2.517s)
[t-SNE] Iteration 150: error = 81.9161835, gradient norm = 0.0036698 (50 iterations in 2.256s)
[t-SNE] Iteration 200: error = 81.1607971, gradient norm = 0.0024739 (50 iterations in 2.240s)
[t-SNE] Iteration 250: error = 80.7808075, gradient norm = 0.0019368 (50 iterations in 2.241s)
[t-SNE] KL divergence after 250 iterations with early exaggeration: 80.780807
[t-SNE] Iteration 300: error = 2.7023807, gradient norm = 0.0013082 (50 iterations in 2.266s)
[t-SNE] Iteration 350: error = 2.1672068, gradient norm = 0.0005762 (50 iterations in 2.175s)
[t-SNE] Iteration 400: error = 1.9168081, gradient norm = 0.0003474 (50 iterations in 2.158s)
[t-SNE] Iteration 450: error = 1.7704682, gradient norm = 0.0002482 (50 iterations in 2.194s)
[t-SNE] Iteration 500: error = 1.6761820, gradient norm = 0.0001938 (50 iterations in 2.201s)
[t-SNE] Iteration 550: error = 1.6116860, gradient norm = 0.0001583 (50 iterations in 6.233s)
[t-SNE] Iteration 600: error = 1.5650036, gradient norm = 0.0001345 (50 iterations in 2.708s)
[t-SNE] Iteration 650: error = 1.5296665, gradient norm = 0.0001166 (50 iterations in 2.204s)
[t-SNE] Iteration 700: error = 1.5022783, gradient norm = 0.0001058 (50 iterations in 2.203s)
[t-SNE] Iteration 750: error = 1.4808836, gradient norm = 0.0000953 (50 iterations in 2.190s)
[t-SNE] Iteration 800: error = 1.4634231, gradient norm = 0.0000878 (50 iterations in 2.201s)
[t-SNE] Iteration 850: error = 1.4492576, gradient norm = 0.0000851 (50 iterations in 2.248s)
[t-SNE] Iteration 900: error = 1.4377539, gradient norm = 0.0000799 (50 iterations in 2.202s)
[t-SNE] Iteration 950: error = 1.4278898, gradient norm = 0.0000753 (50 iterations in 2.178s)
[t-SNE] Iteration 1000: error = 1.4196749, gradient norm = 0.0000706 (50 iterations in 2.205s)
[t-SNE] KL divergence after 1000 iterations: 1.419675
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/regression.py:573: UserWarning: The `size`
parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```

```
saving this plot as image in present working directory...
```

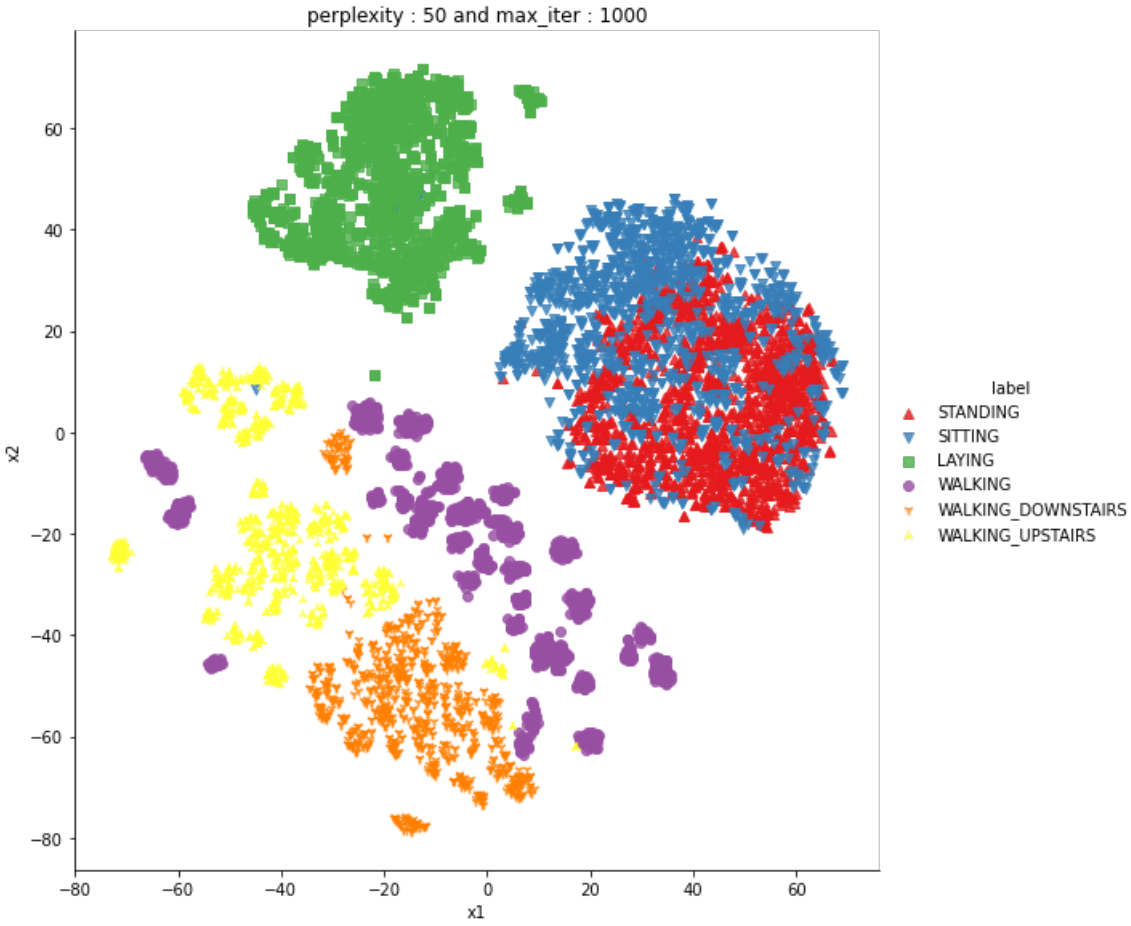



```
Done
[t-SNE] Computing 151 nearest neighbors...
[t-SNE] Indexed 7352 samples in 0.587s...
[t-SNE] Computed neighbors for 7352 samples in 42.468s...
[t-SNE] Computed conditional probabilities for sample 1000 / 7352
[t-SNE] Computed conditional probabilities for sample 2000 / 7352
[t-SNE] Computed conditional probabilities for sample 3000 / 7352
[t-SNE] Computed conditional probabilities for sample 4000 / 7352
[t-SNE] Computed conditional probabilities for sample 5000 / 7352
[t-SNE] Computed conditional probabilities for sample 6000 / 7352
[t-SNE] Computed conditional probabilities for sample 7000 / 7352
[t-SNE] Computed conditional probabilities for sample 7352 / 7352
[t-SNE] Mean sigma: 1.437672
[t-SNE] Computed conditional probabilities in 0.597s
[t-SNE] Iteration 50: error = 86.1462936, gradient norm = 0.0224882 (50 iterations in 5.255s)
[t-SNE] Iteration 100: error = 75.6794968, gradient norm = 0.0052310 (50 iterations in 4.064s)
[t-SNE] Iteration 150: error = 74.7102890, gradient norm = 0.0022697 (50 iterations in 3.326s)
[t-SNE] Iteration 200: error = 74.3220215, gradient norm = 0.0014518 (50 iterations in 3.310s)
[t-SNE] Iteration 250: error = 74.1314392, gradient norm = 0.0013451 (50 iterations in 3.282s)
[t-SNE] KL divergence after 250 iterations with early exaggeration: 74.131439
[t-SNE] Iteration 300: error = 2.1503248, gradient norm = 0.0011810 (50 iterations in 3.045s)
[t-SNE] Iteration 350: error = 1.7540185, gradient norm = 0.0004869 (50 iterations in 2.864s)
[t-SNE] Iteration 400: error = 1.5858097, gradient norm = 0.0002783 (50 iterations in 2.811s)
[t-SNE] Iteration 450: error = 1.4925530, gradient norm = 0.0001883 (50 iterations in 2.836s)
[t-SNE] Iteration 500: error = 1.4330434, gradient norm = 0.0001395 (50 iterations in 2.810s)
[t-SNE] Iteration 550: error = 1.3922095, gradient norm = 0.0001103 (50 iterations in 2.899s)
[t-SNE] Iteration 600: error = 1.3631427, gradient norm = 0.0000933 (50 iterations in 2.842s)
[t-SNE] Iteration 650: error = 1.3421729, gradient norm = 0.0000837 (50 iterations in 2.854s)
[t-SNE] Iteration 700: error = 1.3270123, gradient norm = 0.0000748 (50 iterations in 2.885s)
[t-SNE] Iteration 750: error = 1.3154559, gradient norm = 0.0000716 (50 iterations in 2.847s)
[t-SNE] Iteration 800: error = 1.3065617, gradient norm = 0.0000638 (50 iterations in 2.919s)
[t-SNE] Iteration 850: error = 1.2991855, gradient norm = 0.0000604 (50 iterations in 2.896s)
[t-SNE] Iteration 900: error = 1.2934437, gradient norm = 0.0000589 (50 iterations in 2.939s)
[t-SNE] Iteration 950: error = 1.2887629, gradient norm = 0.0000542 (50 iterations in 2.941s)
```

```
[t-SNE] Iteration 1000: error = 1.2845986, gradient norm = 0.0000527 (50 iterations in 2.927s)
[t-SNE] KL divergence after 1000 iterations: 1.284599
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/regression.py:573: UserWarning: The `size`
parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```

```
saving this plot as image in present working directory...
```




```
Done
```

4. Classical ML models

In [ ]:

```
train = pd.read_csv('/content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/csv_files/train.csv')
test = pd.read_csv('/content/drive/My Drive/Assign - 23 Human Activity
Recognition/HAR/UCI_HAR_Dataset/csv_files/test.csv')
print(train.shape, test.shape)
```

```
(7352, 564) (2947, 564)
```

In [ ]:

```
train.head()
```

Out[ ]:

	tBodyAccmeanX	tBodyAccmeanY	tBodyAccmeanZ	tBodyAccstdX	tBodyAccstdY	tBodyAccstdZ	tBodyAccmadX	tBodyAccmadY	t
0	0.288585	-0.020294	-0.132905	-0.995279	-0.983111	-0.913526	-0.995112	-0.983185	
1	0.278419	-0.016411	-0.123520	-0.998245	-0.975300	-0.960322	-0.998807	-0.974914	

2	0.279665	-0.019467	-0.113462	-0.995380	-0.967187	-0.978944	-0.996520	-0.963668	
3	0.279174	-0.026201	-0.123283	-0.996091	-0.983403	-0.990675	-0.997099	-0.982750	
4	0.276629	-0.016570	-0.115362	-0.998139	-0.980817	-0.990482	-0.998321	-0.979672	

5 rows × 564 columns

In [ ]:

```
X_train = train.drop(['subject', 'Activity', 'ActivityName'], axis=1)
X_test = test.drop(['subject', 'Activity', 'ActivityName'], axis=1)
```

In [ ]:

```
y_train = train.ActivityName
y_test = test.ActivityName
```

In [ ]:

```
print('X_train and y_train : ({},{})'.format(X_train.shape, y_train.shape))
print('X_test  and y_test  : ({},{})'.format(X_test.shape, y_test.shape))
```

```
X_train and y_train : ((7352, 561),(7352,))
X_test  and y_test  : ((2947, 561),(2947,))
```

4.1 General function to perform any model

Function to plot the confusion matrix

In [ ]:

```
import itertools
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
plt.rcParams["font.family"] = 'DejaVu Sans'

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

Generic function to run any model specified

In [ ]:

```
from datetime import datetime
```

```
def perform_model(model, X_train, y_train, X_test, y_test, class_labels, cm_normalize=True, \
                  print_cm=True, cm_cmap=plt.cm.Greens):


    # to store results at various phases
    results = dict()

    train_start_time = datetime.now()
    model.fit(X_train, y_train)
    train_end_time = datetime.now()
    results['training_time'] =  train_end_time - train_start_time
    print('training_time: {}'.format(results['training_time']))


    # predict test data
    print('Predicting test data')
    test_start_time = datetime.now()
    y_pred = model.predict(X_test)
    test_end_time = datetime.now()
    results['testing_time'] = test_end_time - test_start_time
    print('testing time{}'.format(results['testing_time']))
    results['predicted'] = y_pred


    # calculate overall accuracty of the model
    accuracy = metrics.accuracy_score(y_true=y_test, y_pred=y_pred)
    # store accuracy in results
    results['accuracy'] = accuracy
    print('----------------------')
    print('|      Accuracy      |')
    print('----------------------')
    print('\n    {}\n\n'.format(accuracy))


    # confusion matrix
    cm = metrics.confusion_matrix(y_test, y_pred)
    results['confusion_matrix'] = cm
    if print_cm:
        print('--------------------')
        print('| Confusion Matrix |')
        print('--------------------')
        print('\n {}'.format(cm))

    # plot confusin matrix
    plt.figure(figsize=(8,8))
    plt.grid(b=False)
    plot_confusion_matrix(cm, classes=class_labels, normalize=True, title='Normalized confusion
matrix', cmap = cm_cmap)
    plt.show()

    # get classification report
    print('-------------------------')
    print('| Classifiction Report |')
    print('-------------------------')
    classification_report = metrics.classification_report(y_test, y_pred)
    # store report in results
    results['classification_report'] = classification_report
    print(classification_report)

    # add the trained  model to the results
    results['model'] = model

    return results
```

Method to print the gridsearch Attributes

In [ ]:

```
def print_grid_search_attributes(model):
    # Estimator that gave highest score among all the estimators formed in GridSearch
    print('-------------------------')
    print('|      Best Estimator     |')
```

```
    print('---------------------------')
    print('\n\t{}\n'.format(model.best_estimator_))


    # parameters that gave best results while performing grid search
    print('---------------------------')
    print('|     Best parameters     |')
    print('---------------------------')
    print('\tParameters of best estimator : \n\n\t{}\n'.format(model.best_params_))


    #  number of cross validation splits
    print('---------------------------------')
    print('|   No of CrossValidation sets   |')
    print('---------------------------------')
    print('\n\tTotal numbre of cross validation sets: {}\n'.format(model.n_splits_))


    # Average cross validated score of the best estimator, from the Grid Search
    print('--------------------------')
    print('|        Best Score       |')
    print('--------------------------')
    print('\n\tAverage Cross Validate scores of best estimator :
\n\n\t{}\n'.format(model.best_score_))
```

4.2 Logistic Regression with Grid Search

In [ ]:

```
from sklearn import linear_model
from sklearn import metrics

from sklearn.model_selection import GridSearchCV
```

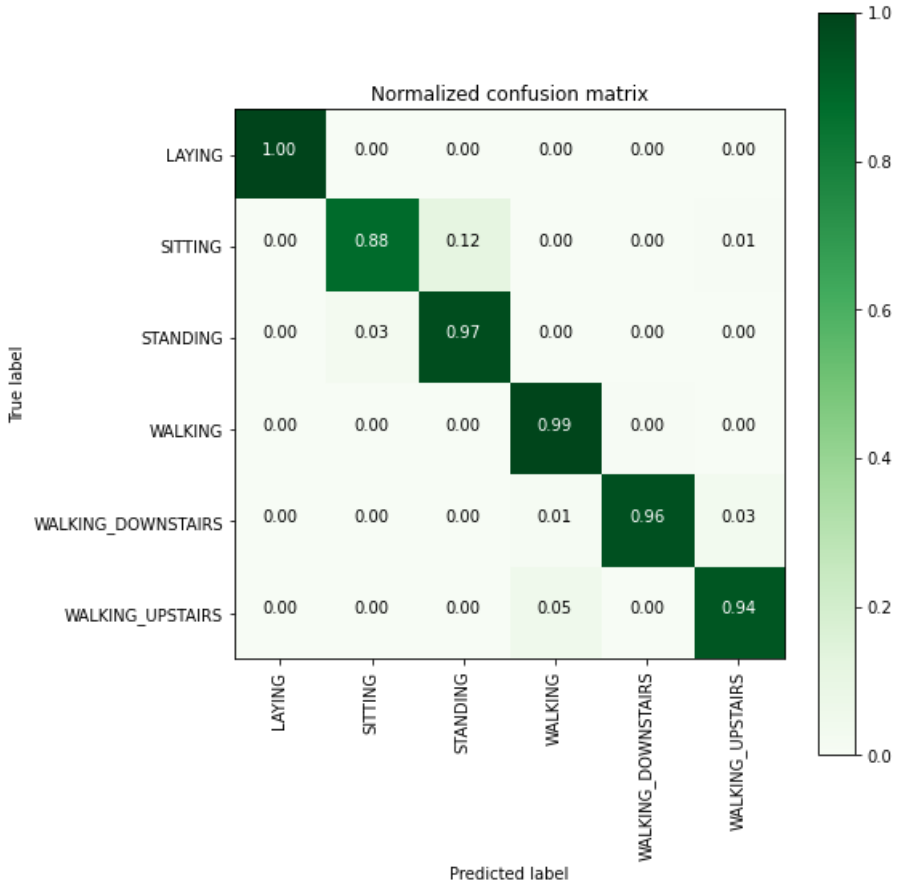
In [ ]:

```
# start Grid search
parameters = {'C':[0.01, 0.1, 1, 10, 20, 30], 'penalty':['l2','l1']}
log_reg = linear_model.LogisticRegression()
log_reg_grid = GridSearchCV(log_reg, param_grid=parameters, cv=3, verbose=1, n_jobs=-1)
log_reg_grid_results =  perform_model(log_reg_grid, X_train, y_train, X_test, y_test, class_labels=
labels)
```

```
Fitting 3 folds for each of 12 candidates, totalling 36 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  36 out of  36 | elapsed:   25.5s finished
```

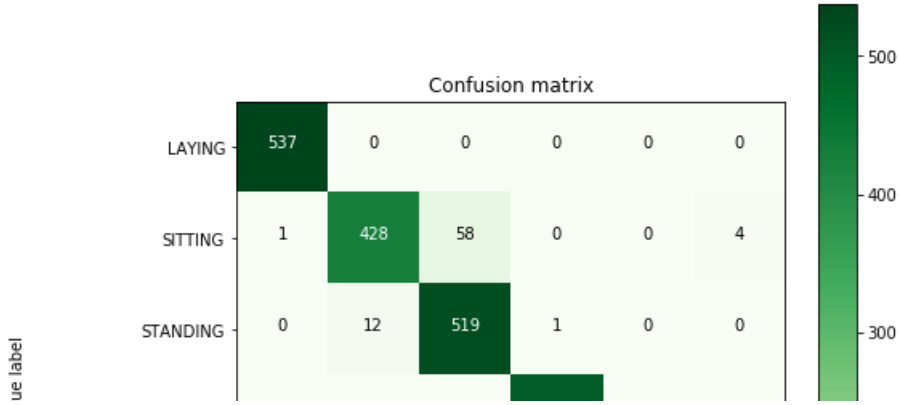
```
training_time: 0:00:28.591985
Predicting test data
testing time0:00:00.009525
----------------------
|      Accuracy      |
----------------------

    0.9586019681031558


--------------------
| Confusion Matrix |
--------------------

 [[537   0   0   0   0   0]
 [  0 431  57   0   0   3]
 [  0  15 517   0   0   0]
 [  0   0   0 493   2   1]
 [  0   0   0   4 402  14]
 [  0   0   0  25   1 445]]
```

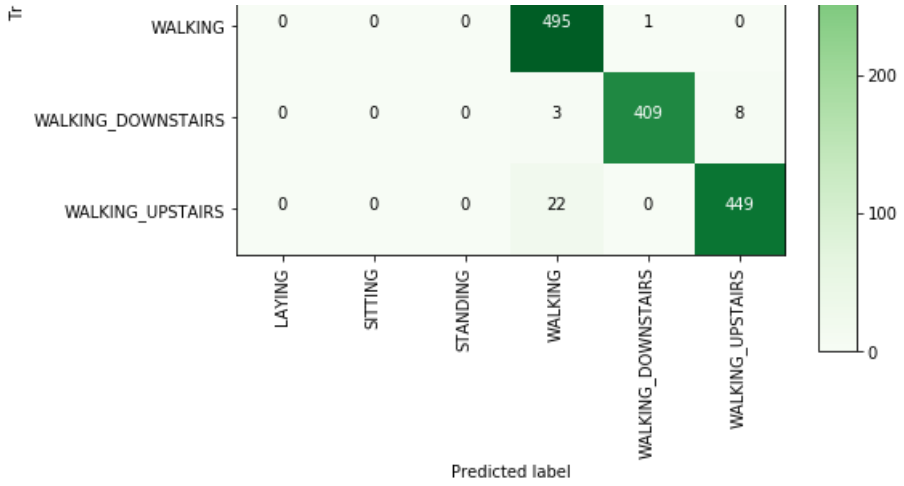



```
-------------------------
| Classifiction Report |
-------------------------
                    precision    recall  f1-score   support

           LAYING       1.00      1.00      1.00       537
          SITTING       0.97      0.88      0.92       491
         STANDING       0.90      0.97      0.93       532
          WALKING       0.94      0.99      0.97       496
WALKING_DOWNSTAIRS       0.99      0.96      0.97       420
  WALKING_UPSTAIRS       0.96      0.94      0.95       471

         accuracy                           0.96      2947
        macro avg       0.96      0.96      0.96      2947
     weighted avg       0.96      0.96      0.96      2947
```

In [ ]:

```
plt.figure(figsize=(8,8))
plt.grid(b=False)
plot_confusion_matrix(log_reg_grid_results['confusion_matrix'], classes=labels, cmap=plt.cm.Greens
, )
plt.show()
```





In [ ]:

```
# observe the attributes of the model
print_grid_search_attributes(log_reg_grid_results['model'])
```

```
--------------------------
|      Best Estimator     |
--------------------------

 LogisticRegression(C=30, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)

--------------------------
|     Best parameters     |
--------------------------
 Parameters of best estimator :

 {'C': 30, 'penalty': 'l2'}

---------------------------------
|   No of CrossValidation sets   |
---------------------------------

 Total numbre of cross validation sets: 3

--------------------------
|        Best Score       |
--------------------------

 Average Cross Validate scores of best estimator :

 0.9461371055495104
```

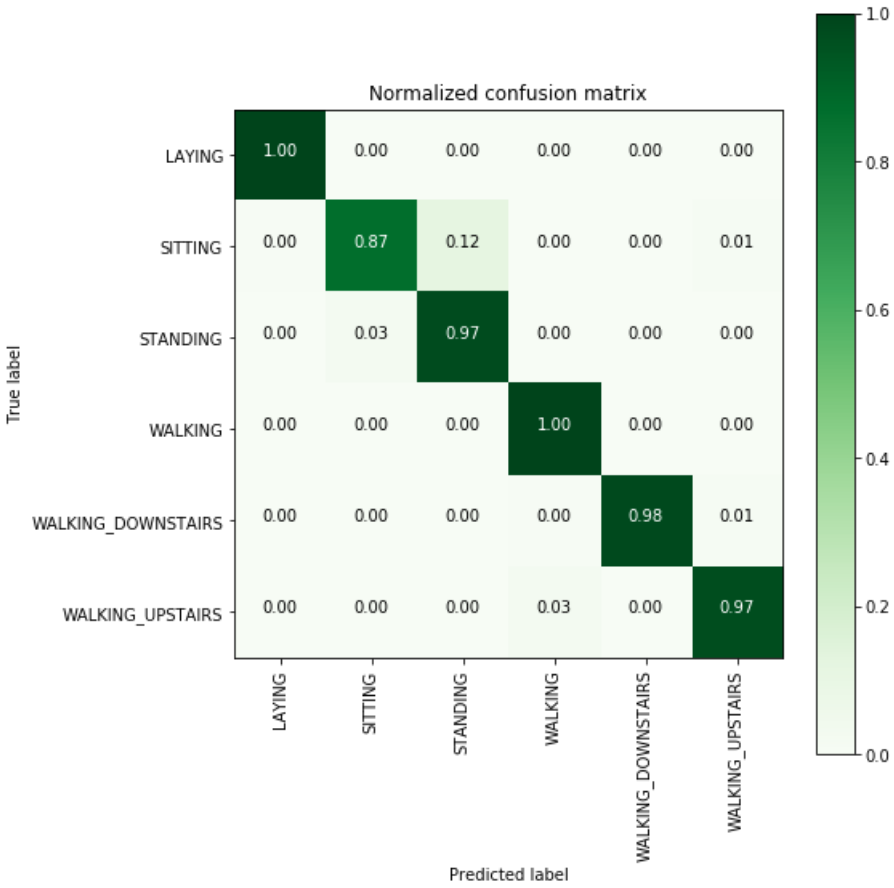
4.3Linear SVC with GridSearch

In [ ]:

```
parameters = {'C':[0.125, 0.5, 1, 2, 8, 16]}
lr_svc = LinearSVC(tol=0.00005)
lr_svc_grid = GridSearchCV(lr_svc, param_grid=parameters, n_jobs=-1, verbose=1)
lr_svc_grid_results = perform_model(lr_svc_grid, X_train, y_train, X_test, y_test, class_labels=lab
els)
```

```
training the model..
Fitting 3 folds for each of 6 candidates, totalling 18 fits

[Parallel(n_jobs=-1)]: Done  18 out of  18 | elapsed:   24.9s finished

Done
```

```
training_time(HH:MM:SS.ms) - 0:00:32.951942


Predicting test data
Done


testing time(HH:MM:SS:ms) - 0:00:00.012182


---------------------
|      Accuracy      |
---------------------

    0.9660671869697998


--------------------
| Confusion Matrix |
--------------------

 [[537   0   0   0   0   0]
 [  2 426  58   0   0   5]
 [  0  14 518   0   0   0]
 [  0   0   0 495   0   1]
 [  0   0   0   2 413   5]
 [  0   0   0  12   1 458]]
```




```
--------------------------
| Classifiction Report |
--------------------------
                    precision    recall  f1-score   support

            LAYING       1.00      1.00      1.00       537
           SITTING       0.97      0.87      0.92       491
          STANDING       0.90      0.97      0.94       532
           WALKING       0.97      1.00      0.99       496
WALKING_DOWNSTAIRS       1.00      0.98      0.99       420
  WALKING_UPSTAIRS       0.98      0.97      0.97       471
```

```
       avg / total       0.97      0.97      0.97      2947
```

In [ ]:

```
print_grid_search_attributes(lr_svc_grid_results['model'])
```

```
--------------------------
|      Best Estimator     |
--------------------------

 LinearSVC(C=8, class_weight=None, dual=True, fit_intercept=True,
     intercept_scaling=1, loss='squared_hinge', max_iter=1000,
     multi_class='ovr', penalty='l2', random_state=None, tol=5e-05,
     verbose=0)

--------------------------
|     Best parameters     |
--------------------------
 Parameters of best estimator :

 {'C': 8}

---------------------------------
|   No of CrossValidation sets   |
---------------------------------

 Total numbre of cross validation sets: 3

--------------------------
|        Best Score       |
--------------------------

 Average Cross Validate scores of best estimator :

 0.9465451577801959
```

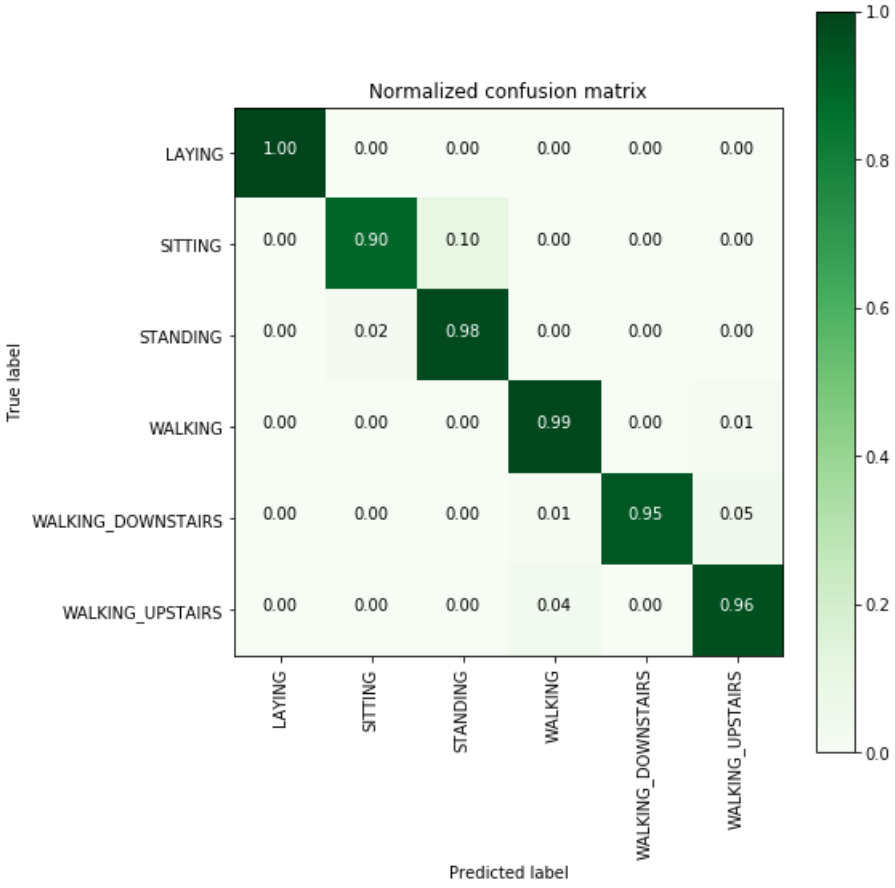
4.4Kernel SVM with GridSearch

In [ ]:

```
from sklearn.svm import SVC
parameters = {'C':[2,8,16],\
              'gamma': [ 0.0078125, 0.125, 2]}
rbf_svm = SVC(kernel='rbf')
rbf_svm_grid = GridSearchCV(rbf_svm,param_grid=parameters, n_jobs=-1)
rbf_svm_grid_results = perform_model(rbf_svm_grid, X_train, y_train, X_test, y_test, class_labels=l
abels)
```

```
training the model..
Done


training_time(HH:MM:SS.ms) - 0:05:46.182889


Predicting test data
Done


testing time(HH:MM:SS:ms) - 0:00:05.221285


---------------------
|      Accuracy      |
---------------------

    0.9626739056667798


--------------------
```

```
| Confusion Matrix |
--------------------

 [[537   0   0   0   0   0]
 [  0 441  48   0   0   2]
 [  0  12 520   0   0   0]
 [  0   0   0 489   2   5]
 [  0   0   0   4 397  19]
 [  0   0   0  17   1 453]]
```




```
-------------------------
| Classifiction Report |
-------------------------
                    precision    recall  f1-score   support

            LAYING       1.00      1.00      1.00       537
           SITTING       0.97      0.90      0.93       491
          STANDING       0.92      0.98      0.95       532
           WALKING       0.96      0.99      0.97       496
WALKING_DOWNSTAIRS       0.99      0.95      0.97       420
  WALKING_UPSTAIRS       0.95      0.96      0.95       471

       avg / total       0.96      0.96      0.96      2947
```

In [ ]:

```
print_grid_search_attributes(rbf_svm_grid_results['model'])
```

```
-------------------------
|      Best Estimator     |
-------------------------

 SVC(C=16, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.0078125, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)

-------------------------
|     Best parameters     |
```

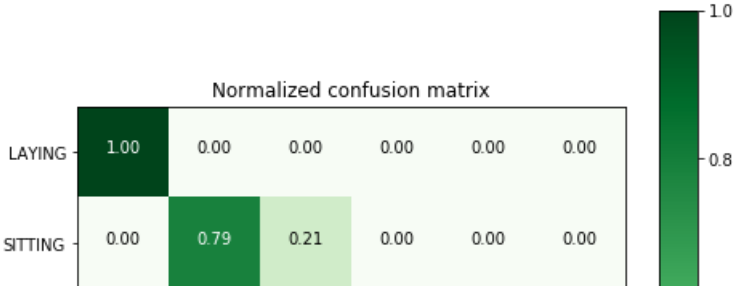
```
--------------------------
 Parameters of best estimator :

 {'C': 16, 'gamma': 0.0078125}

---------------------------------
|   No of CrossValidation sets   |
---------------------------------

 Total numbre of cross validation sets: 3

--------------------------
|        Best Score       |
--------------------------

 Average Cross Validate scores of best estimator :

 0.9440968443960827
```

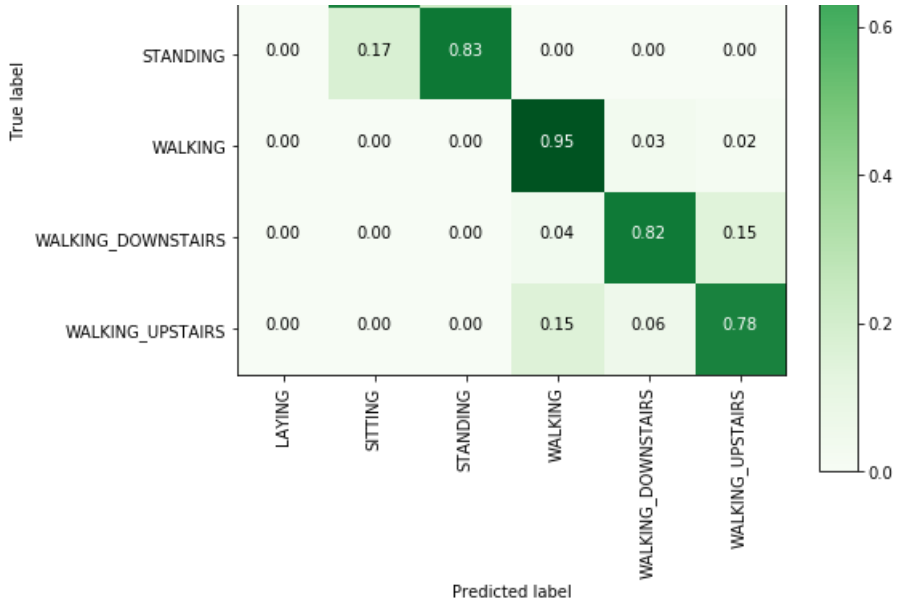
4.5Decision Trees with GridSearchCV

In [ ]:

```
from sklearn.tree import DecisionTreeClassifier
parameters = {'max_depth':np.arange(3,10,2)}
dt = DecisionTreeClassifier()
dt_grid = GridSearchCV(dt,param_grid=parameters, n_jobs=-1)
dt_grid_results = perform_model(dt_grid, X_train, y_train, X_test, y_test, class_labels=labels)
print_grid_search_attributes(dt_grid_results['model'])
```

```
training the model..
Done


training_time(HH:MM:SS.ms) - 0:00:19.476858


Predicting test data
Done


testing time(HH:MM:SS:ms) - 0:00:00.012858


---------------------
|      Accuracy      |
---------------------

    0.8642687478791992


--------------------
| Confusion Matrix |
--------------------

 [[537   0   0   0   0   0]
 [  0 386 105   0   0   0]
 [  0  93 439   0   0   0]
 [  0   0   0 472  16   8]
 [  0   0   0  15 344  61]
 [  0   0   0  73  29 369]]
```





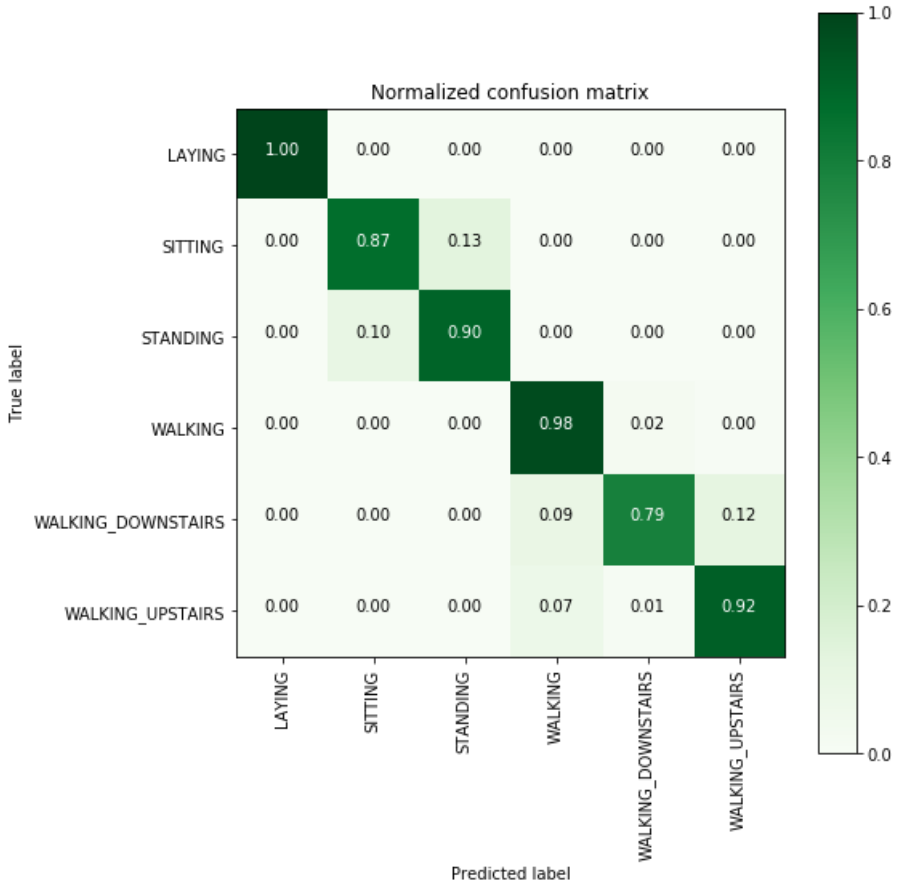

```
-------------------------
| Classifiction Report |
-------------------------
                    precision    recall  f1-score   support

            LAYING       1.00      1.00      1.00       537
           SITTING       0.81      0.79      0.80       491
          STANDING       0.81      0.83      0.82       532
           WALKING       0.84      0.95      0.89       496
WALKING_DOWNSTAIRS       0.88      0.82      0.85       420
  WALKING_UPSTAIRS       0.84      0.78      0.81       471

       avg / total       0.86      0.86      0.86      2947

-------------------------
|      Best Estimator     |
-------------------------

 DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=7,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')

-------------------------
|     Best parameters     |
-------------------------
 Parameters of best estimator :

 {'max_depth': 7}

---------------------------------
|   No of CrossValidation sets   |
---------------------------------

 Total numbre of cross validation sets: 3

-------------------------
|        Best Score       |
-------------------------

 Average Cross Validate scores of best estimator :

 0.8369151251360174
```

4.6Random Forest Classifier with GridSearch

In [ ]:

```
from sklearn.ensemble import RandomForestClassifier
params = {'n_estimators': np.arange(10,201,20), 'max_depth':np.arange(3,15,2)}
rfc = RandomForestClassifier()
rfc_grid = GridSearchCV(rfc, param_grid=params, n_jobs=-1)
rfc_grid_results = perform_model(rfc_grid, X_train, y_train, X_test, y_test, class_labels=labels)
print_grid_search_attributes(rfc_grid_results['model'])
```

```
training the model..
Done


training_time(HH:MM:SS.ms) - 0:06:22.775270


Predicting test data
Done


testing time(HH:MM:SS:ms) - 0:00:00.025937


---------------------
|      Accuracy      |
---------------------

    0.9131319986426875


--------------------
| Confusion Matrix |
--------------------

 [[537   0   0   0   0   0]
 [  0 427  64   0   0   0]
 [  0  52 480   0   0   0]
 [  0   0   0 484  10   2]
 [  0   0   0  38 332  50]
 [  0   0   0  34   6 431]]
```




```
-------------------------
| Classifiction Report |
```

```
-------------------------
                    precision    recall  f1-score   support

            LAYING       1.00      1.00      1.00       537
           SITTING       0.89      0.87      0.88       491
          STANDING       0.88      0.90      0.89       532
           WALKING       0.87      0.98      0.92       496
WALKING_DOWNSTAIRS       0.95      0.79      0.86       420
  WALKING_UPSTAIRS       0.89      0.92      0.90       471

       avg / total       0.92      0.91      0.91      2947

-------------------------
|      Best Estimator     |
-------------------------

 RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=7, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=70, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)

-------------------------
|     Best parameters     |
-------------------------
 Parameters of best estimator :

 {'max_depth': 7, 'n_estimators': 70}

---------------------------------
|   No of CrossValidation sets   |
---------------------------------

 Total numbre of cross validation sets: 3

-------------------------
|        Best Score       |
-------------------------

 Average Cross Validate scores of best estimator :

 0.9141730141458106
```

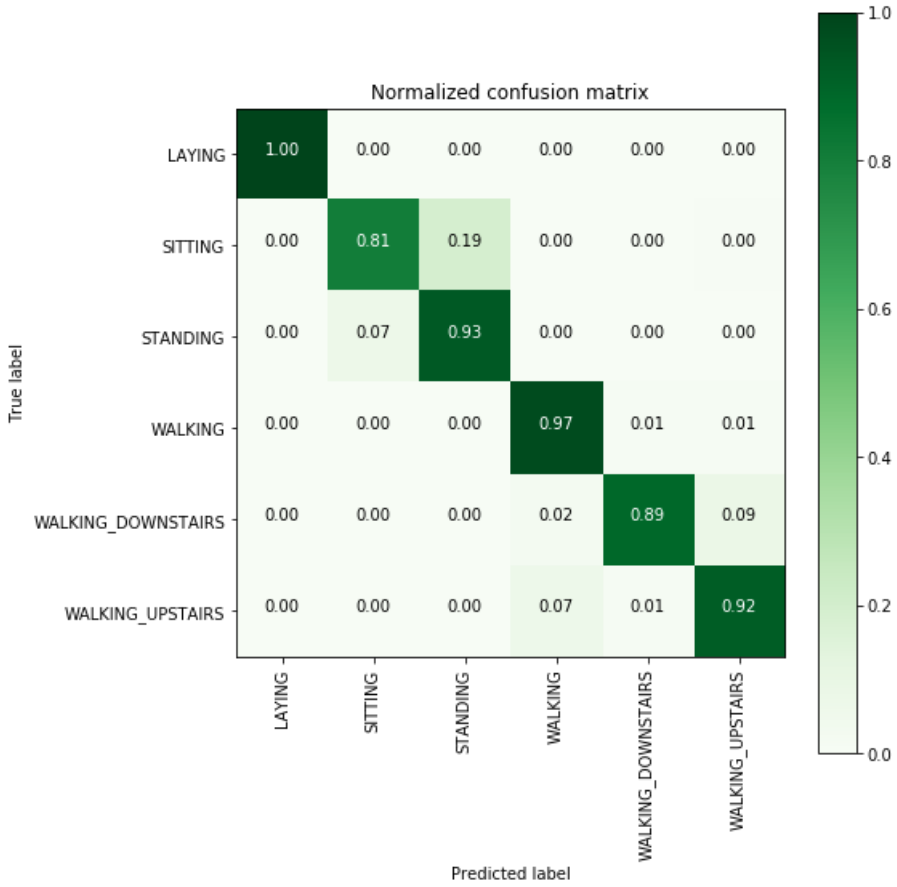
4.7Gradient Boosted Decision Trees With GridSearch

In [ ]:

```
from sklearn.ensemble import GradientBoostingClassifier
param_grid = {'max_depth': np.arange(5,8,1), \
              'n_estimators':np.arange(130,170,10)}
gbdt = GradientBoostingClassifier()
gbdt_grid = GridSearchCV(gbdt, param_grid=param_grid, n_jobs=-1)
gbdt_grid_results = perform_model(gbdt_grid, X_train, y_train, X_test, y_test, class_labels=labels)
print_grid_search_attributes(gbdt_grid_results['model'])
```

```
training the model..
Done


training_time(HH:MM:SS.ms) - 0:28:03.653432


Predicting test data
Done


testing time(HH:MM:SS:ms) - 0:00:00.058843


---------------------
|      Accuracy      |
```

```
---------------------

    0.9222938581608415


--------------------
| Confusion Matrix |
--------------------

 [[537   0   0   0   0   0]
 [  0 396  93   0   0   2]
 [  0  37 495   0   0   0]
 [  0   0   0 483   7   6]
 [  0   0   0  10 374  36]
 [  0   1   0  31   6 433]]
```




```
-------------------------
| Classifiction Report |
-------------------------
                    precision    recall  f1-score   support

            LAYING       1.00      1.00      1.00       537
           SITTING       0.91      0.81      0.86       491
          STANDING       0.84      0.93      0.88       532
           WALKING       0.92      0.97      0.95       496
WALKING_DOWNSTAIRS       0.97      0.89      0.93       420
  WALKING_UPSTAIRS       0.91      0.92      0.91       471

       avg / total       0.92      0.92      0.92      2947

-------------------------
|      Best Estimator     |
-------------------------

 GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=5,
              max_features=None, max_leaf_nodes=None,
              min_impurity_decrease=0.0, min_impurity_split=None,
              min_samples_leaf=1, min_samples_split=2,
              min_weight_fraction_leaf=0.0, n_estimators=140,
              presort='auto', random_state=None, subsample=1.0, verbose=0,
              warm start=False)
```

```
                    _

--------------------------
|     Best parameters     |
--------------------------
 Parameters of best estimator :

 {'max_depth': 5, 'n_estimators': 140}

---------------------------------
|   No of CrossValidation sets   |
---------------------------------

 Total numbre of cross validation sets: 3

--------------------------
|        Best Score       |
--------------------------

 Average Cross Validate scores of best estimator :

 0.904379760609358
```

4.8Comparing all models

In [ ]:

```
print('\n                      Accuracy     Error')
print('                     ----------   --------')
print('Logistic Regression : {:.04}%       {:.04}%'.format(log_reg_grid_results['accuracy'] * 100,\
                                                  100-(log_reg_grid_results['accuracy'] * 100)))

print('Linear SVC          : {:.04}%       {:.04}% '.format(lr_svc_grid_results['accuracy'] * 100,\
                                                       100-(lr_svc_grid_results['accuracy'] * 100)

print('rbf SVM classifier  : {:.04}%      {:.04}% '.format(rbf_svm_grid_results['accuracy'] * 100,\
                                                         100-(rbf_svm_grid_results['accuracy'] * 1
)))

print('DecisionTree        : {:.04}%      {:.04}% '.format(dt_grid_results['accuracy'] * 100,\
                                                       100-(dt_grid_results['accuracy'] * 100)))

print('Random Forest       : {:.04}%      {:.04}% '.format(rfc_grid_results['accuracy'] * 100,\
                                                          100-(rfc_grid_results['accuracy'] * 100)
print('GradientBoosting DT : {:.04}%      {:.04}% '.format(rfc_grid_results['accuracy'] * 100,\
                                                       100-(rfc_grid_results['accuracy'] * 100)))
```

```
                     Accuracy     Error
                     ----------   --------
Logistic Regression : 96.27%       3.733%
Linear SVC          : 96.61%       3.393%
rbf SVM classifier  : 96.27%      3.733%
DecisionTree        : 86.43%      13.57%
Random Forest       : 91.31%      8.687%
GradientBoosting DT : 91.31%      8.687%
```

5.Deep Learning model

In [1]:

```
# Activities are the class labels
# It is a 6 class classification
ACTIVITIES = {
    0: 'WALKING',
    1: 'WALKING_UPSTAIRS',
    2: 'WALKING_DOWNSTAIRS',
    3: 'SITTING',
    4: 'STANDING',
    5: 'LAYING',
}
```

In [2]:

```
def confusion_matrix(y_true, y_pred):
    y_true = pd.Series(ACTIVITIES[y] for y in np.argmax(y_true, axis=1))
    y_pred = pd.Series(ACTIVITIES[y] for y in np.argmax(y_pred, axis=1))

    return pd.crosstab(Y_true, Y_pred, rownames=['True'], colnames=['Pred'])
```

5.1 Loading the data:

- raw signals is in inertial Signals folder of train and test

In [3]:

```
import os
import pandas as pd
import numpy as np
```

In [4]:

```
# Raw data signals
SIGNALS = [
    "body_acc_x",
    "body_acc_y",
    "body_acc_z",
    "body_gyro_x",
    "body_gyro_y",
    "body_gyro_z",
    "total_acc_x",
    "total_acc_y",
    "total_acc_z"
]
```

In [9]:

```
# Utility function to read the data from csv file
def _read_csv(filename):
    return pd.read_csv(filename, delim_whitespace=True, header=None)

# Utility function to load the load
def load_signals(subset):
    signals_data = []
    for signal in SIGNALS:
        filename = f'{subset}/Inertial Signals/{signal}_{subset}.txt'
    signals_data.append(np.array(_read_csv(filename)))
    # Transpose is used to change the dimensionality of the output,
    # aggregating the signals by combination of sample/timestep.
    # Resultant shape is (7352 train/2947 test samples, 128 timesteps, 9 signals)
    return np.transpose(signals_data, (1, 2, 0))
```

In [10]:

```
def load_y(subset):
    filename = f'{subset}/y_{subset}.txt'
    print(filename)
    y = _read_csv(filename)[0]
    return np.array(pd.get_dummies(y))
```

In [11]:

```
def load_data():
    X_train, X_test = load_signals('train'), load_signals('test')
    y_train, y_test = load_y('train'), load_y('test')

    return X_train, X_test, y_train, y_test
```

In [12]:

```
X_train, X_test, y_train, y_test = load_data()
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
train/y_train.txt
test/y_test.txt
(7352, 128, 1)
(7352, 6)
(2947, 128, 1)
(2947, 6)
```

In [ ]:

```
len(set([tuple(category) for category in y_train]))
```

Out[ ]:

```
6
```

In [13]:

```
timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
num_class = len(set([tuple(category) for category in y_train]))
print(timesteps)
print(input_dim)
print(num_class)
```

```
128
1
6
```

In [ ]:

```
# Initializing parameters
epochs = 30
batch_size = 16
n_hidden = 32
```

In [ ]:

```
model = Sequential()
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim)))
model.add(Dropout(0.5))
model.add(Dense(num_class, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_2 (LSTM)                (None, 32)                5376
_________________________________________________________________
dropout_2 (Dropout)          (None, 32)                0
_________________________________________________________________
dense (Dense)                (None, 6)                 198
=================================================================
Total params: 5,574
Trainable params: 5,574
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:

```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [ ]:

```
# Training the model
model.fit(X_train,
          y_train,
          batch_size=batch_size,
          validation_data=(X_test, y_test),
          epochs=epochs)
```

```
Epoch 1/30
460/460 [==============================] - 21s 46ms/step - loss: 1.2867 - accuracy: 0.4498 - val_l
oss: 1.1176 - val_accuracy: 0.4676
Epoch 2/30
460/460 [==============================] - 21s 46ms/step - loss: 1.0009 - accuracy: 0.5413 - val_l
oss: 0.9394 - val_accuracy: 0.5779
Epoch 3/30
460/460 [==============================] - 21s 46ms/step - loss: 0.8613 - accuracy: 0.6175 - val_l
oss: 0.8312 - val_accuracy: 0.6257
Epoch 4/30
460/460 [==============================] - 21s 46ms/step - loss: 0.9245 - accuracy: 0.6055 - val_l
oss: 0.8149 - val_accuracy: 0.6220
Epoch 5/30
460/460 [==============================] - 21s 46ms/step - loss: 0.7102 - accuracy: 0.6674 - val_l
oss: 0.7966 - val_accuracy: 0.6200
Epoch 6/30
460/460 [==============================] - 21s 46ms/step - loss: 0.6464 - accuracy: 0.7100 - val_l
oss: 0.8417 - val_accuracy: 0.6586
Epoch 7/30
460/460 [==============================] - 21s 46ms/step - loss: 0.5687 - accuracy: 0.7722 - val_l
oss: 0.6999 - val_accuracy: 0.7333
Epoch 8/30
460/460 [==============================] - 21s 46ms/step - loss: 0.5186 - accuracy: 0.8164 - val_l
oss: 0.5982 - val_accuracy: 0.7957
Epoch 9/30
460/460 [==============================] - 21s 46ms/step - loss: 0.4665 - accuracy: 0.8562 - val_l
oss: 0.6864 - val_accuracy: 0.7374
Epoch 10/30
460/460 [==============================] - 21s 46ms/step - loss: 0.3793 - accuracy: 0.8841 - val_l
oss: 0.3570 - val_accuracy: 0.8836
Epoch 11/30
460/460 [==============================] - 21s 46ms/step - loss: 0.3210 - accuracy: 0.9045 - val_l
oss: 0.5727 - val_accuracy: 0.8266
Epoch 12/30
460/460 [==============================] - 21s 46ms/step - loss: 0.3013 - accuracy: 0.9135 - val_l
oss: 0.4094 - val_accuracy: 0.8785
Epoch 13/30
460/460 [==============================] - 22s 47ms/step - loss: 0.2626 - accuracy: 0.9162 - val_l
oss: 0.3376 - val_accuracy: 0.8948
Epoch 14/30
460/460 [==============================] - 23s 50ms/step - loss: 0.2646 - accuracy: 0.9193 - val_l
oss: 0.3875 - val_accuracy: 0.8738
Epoch 15/30
460/460 [==============================] - 23s 51ms/step - loss: 0.2408 - accuracy: 0.9263 - val_l
oss: 0.3625 - val_accuracy: 0.8806
Epoch 16/30
460/460 [==============================] - 21s 47ms/step - loss: 0.2867 - accuracy: 0.9146 - val_l
oss: 0.3929 - val_accuracy: 0.8867
Epoch 17/30
460/460 [==============================] - 23s 51ms/step - loss: 0.2303 - accuracy: 0.9290 - val_l
oss: 0.4756 - val_accuracy: 0.8707
Epoch 18/30
460/460 [==============================] - 24s 53ms/step - loss: 0.2204 - accuracy: 0.9325 - val_l
oss: 0.3333 - val_accuracy: 0.8992
Epoch 19/30
460/460 [==============================] - 25s 53ms/step - loss: 0.1917 - accuracy: 0.9380 - val_l
oss: 0.2877 - val_accuracy: 0.9131
Epoch 20/30
460/460 [==============================] - 24s 52ms/step - loss: 0.1788 - accuracy: 0.9408 - val_l
oss: 0.3716 - val_accuracy: 0.9077
Epoch 21/30
460/460 [==============================] - 24s 52ms/step - loss: 0.1872 - accuracy: 0.9403 - val_l
oss: 0.4027 - val_accuracy: 0.9006
Epoch 22/30
460/460 [==============================] - 24s 52ms/step - loss: 0.1932 - accuracy: 0.9406 - val_l
```

```
oss: 0.2458 - val_accuracy: 0.9158
Epoch 23/30
460/460 [==============================] - 25s 54ms/step - loss: 0.1739 - accuracy: 0.9406 - val_l
oss: 0.2768 - val_accuracy: 0.9148
Epoch 24/30
460/460 [==============================] - 24s 52ms/step - loss: 0.1825 - accuracy: 0.9421 - val_l
oss: 0.3055 - val_accuracy: 0.9074
Epoch 25/30
460/460 [==============================] - 24s 52ms/step - loss: 0.1721 - accuracy: 0.9429 - val_l
oss: 0.3111 - val_accuracy: 0.9111
Epoch 26/30
460/460 [==============================] - 25s 54ms/step - loss: 0.1808 - accuracy: 0.9429 - val_l
oss: 0.3808 - val_accuracy: 0.9023
Epoch 27/30
460/460 [==============================] - 24s 52ms/step - loss: 0.1578 - accuracy: 0.9438 - val_l
oss: 0.3045 - val_accuracy: 0.9199
Epoch 28/30
460/460 [==============================] - 24s 51ms/step - loss: 0.1713 - accuracy: 0.9460 - val_l
oss: 0.2625 - val_accuracy: 0.9189
Epoch 29/30
460/460 [==============================] - 24s 53ms/step - loss: 0.1753 - accuracy: 0.9416 - val_l
oss: 0.3111 - val_accuracy: 0.9111
Epoch 30/30
460/460 [==============================] - 24s 51ms/step - loss: 0.1462 - accuracy: 0.9471 - val_l
oss: 0.4735 - val_accuracy: 0.9063
```

Out[ ]:

```
<tensorflow.python.keras.callbacks.History at 0x7f22725cb898>
```

Task:

- Hyperparameter tuning - done using keras tuner --> https://www.youtube.com/watch?v=vvC15l4CY1Q&t=1428s
- Different dropout rate (higher dropout rate upto 0.7)
- Two layer lstm

In [15]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
```

In [16]:

```
from kerastuner.tuners import RandomSearch
from kerastuner.engine.hyperparameters import  HyperParameters
```

In [20]:

```
# Initializing parameters
epochs = 50
batch_size = 64
```

In [30]:

```
#https://neurospace.io/blog/2019/04/using-talos-for-feature-hyperparameter-optimization/
def build_model(hp):
    model = Sequential()

    #layer 1
    model.add(LSTM(hp.Int('lstm_layer_1', min_value=32, max_value=512, step=16), input_shape=(times
teps, input_dim), return_sequences=True))
    model.add(Dropout(hp.Choice('dropout_1', values=[0.3, 0.4, 0.5])))

    #layer 2
    model.add(LSTM(hp.Int('lstm_layer_2', min_value=32, max_value=512, step=16)))
    model.add(Dropout(hp.Choice('dropout_2', values=[0.3, 0.4, 0.5])))

    #output
    model.add(Dense(num_class, activation='sigmoid'))
```

```
    # Compiling the model
    model.compile(loss='categorical_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])
    model.summary()

    return model
```

In [20]:

```
tuner= RandomSearch(build_model, objective='val_accuracy', max_trials=3, executions_per_trial=1, di
rectory='output', project_name='Human Activity Detection')
tuner.search(X_train, y_train, epochs=epochs, batch_size=batch_size, validation_data=(X_test, y_tes
t))
```

```
Epoch 1/50
230/230 [==============================] - 13s 57ms/step - loss: 1.3081 - accuracy: 0.4950 - val_l
oss: 0.9772 - val_accuracy: 0.5779
Epoch 2/50
230/230 [==============================] - 12s 53ms/step - loss: 0.9099 - accuracy: 0.5773 - val_l
oss: 0.8541 - val_accuracy: 0.5972
Epoch 3/50
230/230 [==============================] - 12s 54ms/step - loss: 0.7999 - accuracy: 0.6153 - val_l
oss: 0.8225 - val_accuracy: 0.6176
Epoch 4/50
230/230 [==============================] - 12s 54ms/step - loss: 0.8222 - accuracy: 0.6190 - val_l
oss: 0.9007 - val_accuracy: 0.5422
Epoch 5/50
230/230 [==============================] - 12s 54ms/step - loss: 0.8383 - accuracy: 0.5703 - val_l
oss: 0.8283 - val_accuracy: 0.6013
Epoch 6/50
230/230 [==============================] - 12s 54ms/step - loss: 0.7633 - accuracy: 0.6183 - val_l
oss: 0.8030 - val_accuracy: 0.6037
Epoch 7/50
230/230 [==============================] - 12s 53ms/step - loss: 0.7382 - accuracy: 0.6269 - val_l
oss: 0.7668 - val_accuracy: 0.6383
Epoch 8/50
230/230 [==============================] - 12s 54ms/step - loss: 0.7310 - accuracy: 0.6325 - val_l
oss: 0.7705 - val_accuracy: 0.6200
Epoch 9/50
230/230 [==============================] - 12s 54ms/step - loss: 0.6965 - accuracy: 0.6462 - val_l
oss: 0.7579 - val_accuracy: 0.6230
Epoch 10/50
230/230 [==============================] - 12s 54ms/step - loss: 0.7088 - accuracy: 0.6415 - val_l
oss: 0.8226 - val_accuracy: 0.6233
Epoch 11/50
230/230 [==============================] - 12s 53ms/step - loss: 0.7661 - accuracy: 0.6143 - val_l
oss: 0.7549 - val_accuracy: 0.6074
Epoch 12/50
230/230 [==============================] - 12s 53ms/step - loss: 0.8967 - accuracy: 0.6088 - val_l
oss: 0.8533 - val_accuracy: 0.5925
Epoch 13/50
230/230 [==============================] - 12s 53ms/step - loss: 0.7232 - accuracy: 0.6468 - val_l
oss: 0.8306 - val_accuracy: 0.6030
Epoch 14/50
230/230 [==============================] - 12s 54ms/step - loss: 0.6766 - accuracy: 0.6574 - val_l
oss: 0.8190 - val_accuracy: 0.5989
Epoch 15/50
230/230 [==============================] - 12s 54ms/step - loss: 0.7740 - accuracy: 0.5894 - val_l
oss: 0.8858 - val_accuracy: 0.5670
Epoch 16/50
230/230 [==============================] - 12s 53ms/step - loss: 0.7139 - accuracy: 0.6181 - val_l
oss: 0.7471 - val_accuracy: 0.6101
Epoch 17/50
230/230 [==============================] - 12s 53ms/step - loss: 0.6634 - accuracy: 0.6572 - val_l
oss: 0.7189 - val_accuracy: 0.6159
Epoch 18/50
230/230 [==============================] - 12s 53ms/step - loss: 0.6191 - accuracy: 0.6657 - val_l
oss: 0.7080 - val_accuracy: 0.6132
Epoch 19/50
230/230 [==============================] - 12s 53ms/step - loss: 0.5959 - accuracy: 0.6746 - val_l
oss: 0.9856 - val_accuracy: 0.5860
Epoch 20/50
230/230 [==============================] - 12s 53ms/step - loss: 0.6627 - accuracy: 0.6638 - val_l
oss: 0.9546 - val_accuracy: 0.5670
Epoch 21/50
```

```
Epoch 21/50
230/230 [==============================] - 12s 53ms/step - loss: 0.6087 - accuracy: 0.6929 - val_l
oss: 1.0976 - val_accuracy: 0.5843
Epoch 22/50
230/230 [==============================] - 12s 53ms/step - loss: 0.6273 - accuracy: 0.6861 - val_l
oss: 0.8144 - val_accuracy: 0.6607
Epoch 23/50
230/230 [==============================] - 12s 53ms/step - loss: 0.5633 - accuracy: 0.7221 - val_l
oss: 0.7328 - val_accuracy: 0.7027
Epoch 24/50
230/230 [==============================] - 12s 54ms/step - loss: 0.5241 - accuracy: 0.7363 - val_l
oss: 0.6887 - val_accuracy: 0.7190
Epoch 25/50
230/230 [==============================] - 12s 53ms/step - loss: 0.5422 - accuracy: 0.7633 - val_l
oss: 0.6755 - val_accuracy: 0.7272
Epoch 26/50
230/230 [==============================] - 12s 53ms/step - loss: 0.4879 - accuracy: 0.7913 - val_l
oss: 0.6414 - val_accuracy: 0.7397
Epoch 27/50
230/230 [==============================] - 12s 53ms/step - loss: 0.4602 - accuracy: 0.7899 - val_l
oss: 0.6861 - val_accuracy: 0.7316
Epoch 28/50
230/230 [==============================] - 12s 53ms/step - loss: 0.4125 - accuracy: 0.7994 - val_l
oss: 0.6284 - val_accuracy: 0.7808
Epoch 29/50
230/230 [==============================] - 12s 53ms/step - loss: 0.4186 - accuracy: 0.8003 - val_l
oss: 0.6035 - val_accuracy: 0.7893
Epoch 30/50
230/230 [==============================] - 12s 53ms/step - loss: 0.5598 - accuracy: 0.7542 - val_l
oss: 0.6139 - val_accuracy: 0.7489
Epoch 31/50
230/230 [==============================] - 12s 53ms/step - loss: 0.4244 - accuracy: 0.7998 - val_l
oss: 0.5173 - val_accuracy: 0.7845
Epoch 32/50
230/230 [==============================] - 12s 53ms/step - loss: 0.3758 - accuracy: 0.8162 - val_l
oss: 0.5978 - val_accuracy: 0.7754
Epoch 33/50
230/230 [==============================] - 12s 53ms/step - loss: 0.3638 - accuracy: 0.8161 - val_l
oss: 0.7298 - val_accuracy: 0.7445
Epoch 34/50
230/230 [==============================] - 12s 53ms/step - loss: 0.3501 - accuracy: 0.8214 - val_l
oss: 0.5711 - val_accuracy: 0.8005
Epoch 35/50
230/230 [==============================] - 12s 53ms/step - loss: 0.3659 - accuracy: 0.8179 - val_l
oss: 0.5366 - val_accuracy: 0.8195
Epoch 36/50
230/230 [==============================] - 12s 53ms/step - loss: 0.3308 - accuracy: 0.8626 - val_l
oss: 0.5754 - val_accuracy: 0.7822
Epoch 37/50
230/230 [==============================] - 12s 53ms/step - loss: 0.3714 - accuracy: 0.8327 - val_l
oss: 0.4618 - val_accuracy: 0.8646
Epoch 38/50
230/230 [==============================] - 12s 53ms/step - loss: 0.2851 - accuracy: 0.8859 - val_l
oss: 0.4085 - val_accuracy: 0.8873
Epoch 39/50
230/230 [==============================] - 12s 53ms/step - loss: 0.3204 - accuracy: 0.8871 - val_l
oss: 0.4200 - val_accuracy: 0.8812
Epoch 40/50
230/230 [==============================] - 12s 53ms/step - loss: 0.2776 - accuracy: 0.9072 - val_l
oss: 0.4033 - val_accuracy: 0.8839
Epoch 41/50
230/230 [==============================] - 12s 53ms/step - loss: 0.2204 - accuracy: 0.9302 - val_l
oss: 0.4260 - val_accuracy: 0.8768
Epoch 42/50
230/230 [==============================] - 12s 53ms/step - loss: 0.2550 - accuracy: 0.9270 - val_l
oss: 0.4872 - val_accuracy: 0.8653
Epoch 43/50
230/230 [==============================] - 12s 53ms/step - loss: 0.1964 - accuracy: 0.9388 - val_l
oss: 0.3835 - val_accuracy: 0.8884
Epoch 44/50
230/230 [==============================] - 12s 53ms/step - loss: 0.1640 - accuracy: 0.9403 - val_l
oss: 0.3417 - val_accuracy: 0.9023
Epoch 45/50
230/230 [==============================] - 12s 53ms/step - loss: 0.1828 - accuracy: 0.9406 - val_l
oss: 0.7611 - val_accuracy: 0.8280
Epoch 46/50
230/230 [==============================] - 12s 53ms/step - loss: 0.1627 - accuracy: 0.9495 - val_l
oss: 0.4056 - val_accuracy: 0.8985
```

```
oss: 0.4090 - val_accuracy: 0.8965
Epoch 47/50
230/230 [==============================] - 12s 53ms/step - loss: 0.1634 - accuracy: 0.9474 - val_l
oss: 0.4114 - val_accuracy: 0.8955
Epoch 48/50
230/230 [==============================] - 12s 53ms/step - loss: 0.2071 - accuracy: 0.9373 - val_l
oss: 0.7394 - val_accuracy: 0.8317
Epoch 49/50
230/230 [==============================] - 12s 53ms/step - loss: 0.2369 - accuracy: 0.9226 - val_l
oss: 0.4994 - val_accuracy: 0.8500
Epoch 50/50
230/230 [==============================] - 12s 53ms/step - loss: 0.1963 - accuracy: 0.9285 - val_l
oss: 0.4090 - val_accuracy: 0.8962
```

Trial complete

Trial summary

|-Trial ID: f0a70ad8e7363c1a476406c77faff3c0

|-Score: 0.9022734761238098

|-Best step: 0

Hyperparameters:

|-dropout_1: 0.7

|-dropout_2: 0.5

|-lstm_layer_1: 32

|-lstm_layer_2: 32

```
Epoch 1/50
230/230 [==============================] - 15s 64ms/step - loss: 1.2430 - accuracy: 0.4961 - val_l
oss: 1.0084 - val_accuracy: 0.6179
Epoch 2/50
230/230 [==============================] - 14s 61ms/step - loss: 0.8583 - accuracy: 0.6227 - val_l
oss: 0.8091 - val_accuracy: 0.6407
Epoch 3/50
230/230 [==============================] - 14s 61ms/step - loss: 0.7285 - accuracy: 0.6594 - val_l
oss: 0.6891 - val_accuracy: 0.6325
Epoch 4/50
230/230 [==============================] - 14s 61ms/step - loss: 0.7240 - accuracy: 0.6733 - val_l
oss: 0.6906 - val_accuracy: 0.7173
Epoch 5/50
230/230 [==============================] - 14s 61ms/step - loss: 0.5296 - accuracy: 0.7692 - val_l
oss: 0.5057 - val_accuracy: 0.7570
Epoch 6/50
230/230 [==============================] - 14s 61ms/step - loss: 0.4543 - accuracy: 0.7715 - val_l
oss: 0.4493 - val_accuracy: 0.7560
Epoch 7/50
230/230 [==============================] - 14s 61ms/step - loss: 0.3433 - accuracy: 0.8013 - val_l
oss: 0.3766 - val_accuracy: 0.7665
Epoch 8/50
230/230 [==============================] - 14s 61ms/step - loss: 0.3124 - accuracy: 0.8143 - val_l
oss: 0.5161 - val_accuracy: 0.7628
Epoch 9/50
230/230 [==============================] - 14s 61ms/step - loss: 0.2890 - accuracy: 0.8268 - val_l
oss: 0.3958 - val_accuracy: 0.7689
Epoch 10/50
230/230 [==============================] - 14s 61ms/step - loss: 0.2756 - accuracy: 0.8418 - val_l
oss: 0.4547 - val_accuracy: 0.7530
Epoch 11/50
230/230 [==============================] - 14s 61ms/step - loss: 0.2477 - accuracy: 0.8803 - val_l
oss: 0.3819 - val_accuracy: 0.9043
Epoch 12/50
230/230 [==============================] - 14s 61ms/step - loss: 0.2516 - accuracy: 0.9200 - val_l
```

oss: 0.3579 - val_accuracy: 0.9002
Epoch 13/50
230/230 [==============================] - 14s 60ms/step - loss: 0.1883 - accuracy: 0.9449 - val_l
oss: 0.2688 - val_accuracy: 0.9074
Epoch 14/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1619 - accuracy: 0.9460 - val_l
oss: 0.2500 - val_accuracy: 0.9091
Epoch 15/50
230/230 [==============================] - 14s 60ms/step - loss: 0.1829 - accuracy: 0.9406 - val_l
oss: 0.2851 - val_accuracy: 0.9084
Epoch 16/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1764 - accuracy: 0.9418 - val_l
oss: 0.2305 - val_accuracy: 0.9145
Epoch 17/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1439 - accuracy: 0.9514 - val_l
oss: 0.2280 - val_accuracy: 0.9158
Epoch 18/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1440 - accuracy: 0.9472 - val_l
oss: 0.9861 - val_accuracy: 0.7665
Epoch 19/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1743 - accuracy: 0.9415 - val_l
oss: 0.1807 - val_accuracy: 0.9308
Epoch 20/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1286 - accuracy: 0.9523 - val_l
oss: 0.2010 - val_accuracy: 0.9253
Epoch 21/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1375 - accuracy: 0.9518 - val_l
oss: 0.2790 - val_accuracy: 0.9036
Epoch 22/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1316 - accuracy: 0.9483 - val_l
oss: 0.4169 - val_accuracy: 0.8856
Epoch 23/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1587 - accuracy: 0.9431 - val_l
oss: 0.2771 - val_accuracy: 0.9114
Epoch 24/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1427 - accuracy: 0.9486 - val_l
oss: 0.2313 - val_accuracy: 0.9179
Epoch 25/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1206 - accuracy: 0.9523 - val_l
oss: 0.2301 - val_accuracy: 0.9189
Epoch 26/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1751 - accuracy: 0.9404 - val_l
oss: 0.3006 - val_accuracy: 0.9080
Epoch 27/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1651 - accuracy: 0.9440 - val_l
oss: 0.2986 - val_accuracy: 0.9145
Epoch 28/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1774 - accuracy: 0.9418 - val_l
oss: 0.2787 - val_accuracy: 0.9128
Epoch 29/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1529 - accuracy: 0.9430 - val_l
oss: 0.2200 - val_accuracy: 0.9186
Epoch 30/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1615 - accuracy: 0.9471 - val_l
oss: 0.1911 - val_accuracy: 0.9287
Epoch 31/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1383 - accuracy: 0.9501 - val_l
oss: 0.2763 - val_accuracy: 0.9162
Epoch 32/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1500 - accuracy: 0.9476 - val_l
oss: 0.2310 - val_accuracy: 0.9250
Epoch 33/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1508 - accuracy: 0.9465 - val_l
oss: 0.2585 - val_accuracy: 0.9141
Epoch 34/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1563 - accuracy: 0.9446 - val_l
oss: 0.2451 - val_accuracy: 0.9135
Epoch 35/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1383 - accuracy: 0.9471 - val_l
oss: 0.2398 - val_accuracy: 0.9196
Epoch 36/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1349 - accuracy: 0.9495 - val_l
oss: 0.2289 - val_accuracy: 0.9114
Epoch 37/50
230/230 [==============================] - 14s 60ms/step - loss: 0.1298 - accuracy: 0.9489 - val_l
oss: 0.2340 - val_accuracy: 0.9226
Epoch 38/50

```
Epoch 38/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1270 - accuracy: 0.9521 - val_l
oss: 0.2868 - val_accuracy: 0.9155
Epoch 39/50
230/230 [==============================] - 14s 60ms/step - loss: 0.1428 - accuracy: 0.9517 - val_l
oss: 0.2366 - val_accuracy: 0.9247
Epoch 40/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1360 - accuracy: 0.9498 - val_l
oss: 0.2306 - val_accuracy: 0.9199
Epoch 41/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1132 - accuracy: 0.9558 - val_l
oss: 0.2246 - val_accuracy: 0.9233
Epoch 42/50
230/230 [==============================] - 14s 60ms/step - loss: 0.1286 - accuracy: 0.9517 - val_l
oss: 0.4619 - val_accuracy: 0.8911
Epoch 43/50
230/230 [==============================] - 14s 60ms/step - loss: 0.1270 - accuracy: 0.9478 - val_l
oss: 0.3329 - val_accuracy: 0.9196
Epoch 44/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1037 - accuracy: 0.9565 - val_l
oss: 0.3532 - val_accuracy: 0.9135
Epoch 45/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1131 - accuracy: 0.9584 - val_l
oss: 0.2360 - val_accuracy: 0.9213
Epoch 46/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1222 - accuracy: 0.9546 - val_l
oss: 0.3185 - val_accuracy: 0.9135
Epoch 47/50
230/230 [==============================] - 14s 60ms/step - loss: 0.1060 - accuracy: 0.9582 - val_l
oss: 0.2109 - val_accuracy: 0.9186
Epoch 48/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1081 - accuracy: 0.9553 - val_l
oss: 0.2840 - val_accuracy: 0.9128
Epoch 49/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1097 - accuracy: 0.9548 - val_l
oss: 0.3230 - val_accuracy: 0.9087
Epoch 50/50
230/230 [==============================] - 14s 61ms/step - loss: 0.1105 - accuracy: 0.9551 - val_l
oss: 0.3373 - val_accuracy: 0.9070
```

Trial complete

Trial summary

|-Trial ID: 9310da732da59560df13dd738cdd4932

|-Score: 0.9307770729064941

|-Best step: 0

Hyperparameters:

|-dropout_1: 0.3

|-dropout_2: 0.3

|-lstm_layer_1: 96

|-lstm_layer_2: 32

```
Epoch 1/50
230/230 [==============================] - 17s 72ms/step - loss: 1.2271 - accuracy: 0.4770 - val_l
oss: 1.6834 - val_accuracy: 0.3929
Epoch 2/50
230/230 [==============================] - 16s 69ms/step - loss: 0.9136 - accuracy: 0.5569 - val_l
oss: 0.8151 - val_accuracy: 0.6135
Epoch 3/50
131/230 [================>.............]Buffered data was truncated after reaching the output size
limit.
```

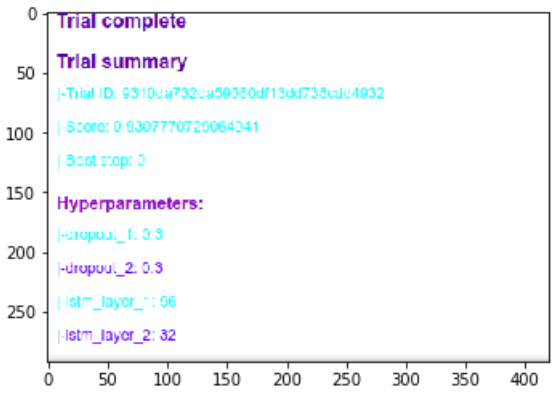
6. Best Model

In [2]:

```
import cv2
import matplotlib.pyplot as plt
%matplotlib inline

img = cv2.imread('/home/prem-kumar/Documents/Screenshot from 2020-06-17 12-51-58.png')
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

Out[2]:

```
<matplotlib.image.AxesImage at 0x7fe40ad0d190>
```



Note:

- I took a screenshot where i got over 93.07% accuracy.

In [36]:

```
best_model = tuner.get_best_models(num_models=1)[0]
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_4 (LSTM)                (None, 128, 96)           37632
_________________________________________________________________
dropout_4 (Dropout)          (None, 128, 96)           0
_________________________________________________________________
lstm_5 (LSTM)                (None, 32)                16512
_________________________________________________________________
dropout_5 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 198
=================================================================
Total params: 54,342
Trainable params: 54,342
Non-trainable params: 0
_________________________________________________________________
```

Summary:

In [3]:

```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ['Model', 'layer', 'No of LSTM units', 'Dropout Rate', 'Final Accuracy']
x.add_row(['LSTM', 'layer-1', '96', '0.3', '-'])
```

```
x.add_row(['LSTM', 'layer-2', '32', '0.3', '0.9307'])
print(x)
```

```
+-------+---------+------------------+--------------+----------------+
| Model |  layer  | No of LSTM units | Dropout Rate | Final Accuracy |
+-------+---------+------------------+--------------+----------------+
|  LSTM | layer-1 |        96        |     0.3      |       -        |
|  LSTM | layer-2 |        32        |     0.3      |     0.9307     |
+-------+---------+------------------+--------------+----------------+
```