# HumanActivityRecognition

This project is to build a model that predicts the human activities such as Walking, Walking_Upstairs, Walking_Downstairs, Sitting, Standing or Laying.

This dataset is collected from 30 persons(referred as subjects in this dataset), performing different activities with a smartphone to their waists. The data is recorded with the help of sensors (accelerometer and Gyroscope) in that smartphone. This experiment was video recorded to label the data manually.

## How data was recorded

By using the sensors(Gyroscope and accelerometer) in a smartphone, they have captured '3-axial linear acceleration'(*tAcc-XYZ*) from accelerometer and '3-axial angular velocity' (*tGyro-XYZ*) from Gyroscope with several variations.

> prefix 't' in those metrics denotes time.
>
> suffix 'XYZ' represents 3-axial signals in X , Y, and Z directions.

### Feature names

1. These sensor signals are preprocessed by applying noise filters and then sampled in fixed-width windows(sliding windows) of 2.56 seconds each with 50% overlap. ie., each window has 128 readings.
2. From Each window, a feature vector was obtianed by calculating variables from the time and frequency domain.

> In our dataset, each datapoint represents a window with different readings

3. The accelertion signal was saperated into Body and Gravity acceleration signals(***tBodyAcc-XYZ*** and ***tGravityAcc-XYZ***) using some low pass filter with corner frequecy of 0.3Hz.
4. After that, the body linear acceleration and angular velocity were derived in time to obtian *jerk signals* (***tBodyAccJerk-XYZ*** and ***tBodyGyroJerk-XYZ***).
5. The magnitude of these 3-dimensional signals were calculated using the Euclidian norm. This magnitudes are represented as features with names like *tBodyAccMag*, *tGravityAccMag*, *tBodyAccJerkMag*, *tBodyGyroMag* and *tBodyGyroJerkMag*.
6. Finally, We've got frequency domain signals from some of the available signals by applying a FFT (Fast Fourier Transform). These signals obtained were labeled with ***prefix 'f'*** just like original signals with ***prefix 't'***. These signals are labeled as ***fBodyAcc-XYZ***, ***fBodyGyroMag*** etc.,.
7. These are the signals that we got so far.

   - tBodyAcc-XYZ
   - tGravityAcc-XYZ
   - tBodyAccJerk-XYZ
   - tBodyGyro-XYZ
   - tBodyGyroJerk-XYZ
   - tBodyAccMag
   - tGravityAccMag
   - tBodyAccJerkMag
   - tBodyGyroMag
   - tBodyGyroJerkMag
   - fBodyAcc-XYZ
   - fBodyAccJerk-XYZ
   - fBodyGyro-XYZ
   - fBodyAccMag
   - fBodyAccJerkMag
   - fBodyGyroMag

- fBodyGyroJerkMag

8. We can esitmate some set of variables from the above signals. ie., We will estimate the following properties on each and every signal that we recoreded so far.

    - ***mean()***: Mean value
    - ***std()***: Standard deviation
    - ***mad()***: Median absolute deviation
    - ***max()***: Largest value in array
    - ***min()***: Smallest value in array
    - ***sma()***: Signal magnitude area
    - ***energy()***: Energy measure. Sum of the squares divided by the number of values.
    - ***iqr()***: Interquartile range
    - ***entropy()***: Signal entropy
    - ***arCoeff()***: Autorregresion coefficients with Burg order equal to 4
    - ***correlation()***: correlation coefficient between two signals
    - ***maxInds()***: index of the frequency component with largest magnitude
    - ***meanFreq()***: Weighted average of the frequency components to obtain a mean frequency
    - ***skewness()***: skewness of the frequency domain signal
    - ***kurtosis()***: kurtosis of the frequency domain signal
    - ***bandsEnergy()***: Energy of a frequency interval within the 64 bins of the FFT of each window.
    - ***angle()***: Angle between to vectors.

9. We can obtain some other vectors by taking the average of signals in a single window sample. These are used on the angle() variable' `

    - gravityMean
    - tBodyAccMean
    - tBodyAccJerkMean
    - tBodyGyroMean
    - tBodyGyroJerkMean

## Y_Labels(Encoded)

- In the dataset, Y_labels are represented as numbers from 1 to 6 as their identifiers.
    - WALKING as **1**
    - WALKING_UPSTAIRS as **2**
    - WALKING_DOWNSTAIRS as **3**
    - SITTING as **4**
    - STANDING as **5**
    - LAYING as **6**

## Train and test data were saperated

- The readings from ***70%*** of the volunteers were taken as ***trianing data*** and remaining ***30%*** subjects recordings were taken for ***test data***

## Data

- All the data is present in 'UCI_HAR_dataset/' folder in present working directory.
    - Feature names are present in 'UCI_HAR_dataset/features.txt'
    - ***Train Data***
        - 'UCI_HAR_dataset/train/X_train.txt'
        - 'UCI_HAR_dataset/train/subject_train.txt'
        - 'UCI_HAR_dataset/train/y_train.txt'
    - ***Test Data***
        - 'UCI_HAR_dataset/test/X_test.txt'
        - 'UCI_HAR_dataset/test/subject_test.txt'
        - 'UCI_HAR_dataset/test/y_test.txt'

## Data Size :

> 27 MB

# Quick overview of the dataset :

- Accelerometer and Gyroscope readings are taken from 30 volunteers(referred as subjects) while performing the following 6 Activities.
    1. Walking
    2. WalkingUpstairs
    3. WalkingDownstairs
    4. Standing
    5. Sitting
    6. Lying.
- Readings are divided into a window of 2.56 seconds with 50% overlapping.
- Accelerometer readings are divided into gravity acceleration and body acceleration readings, which has x,y and z components each.
- Gyroscope readings are the measure of angular velocities which has x,y and z components.
- Jerk signals are calculated for BodyAcceleration readings.
- Fourier Transforms are made on the above time readings to obtain frequency readings.
- Now, on all the base signal readings., mean, max, mad, sma, arcoefficient, engerybands,entropy etc., are calculated for each window.
- We get a feature vector of 561 features and these features are given in the dataset.
- Each window of readings is a datapoint of 561 features.

## Problem Framework

- 30 subjects(volunteers) data is randomly split to 70%(21) test and 30%(7) train data.
- Each datapoint corresponds one of the 6 Activities.

## Problem Statement

- Given a new datapoint we have to predict the Activity

In [0]:

```
import numpy as np
import pandas as pd

# get the features from the file features.txt
features = list()
with open('UCI_HAR_Dataset/features.txt') as f:
    features = [line.split()[1] for line in f.readlines()]
print('No of Features: {}'.format(len(features)))
```

## Obtain the train data

In [0]:

```
# get the data from txt files to pandas dataffame
X_train = pd.read_csv('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('UCI_HAR_dataset/train/subject_train.txt', header=None, squeeze=True)

y_train = pd.read_csv('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
train.sample()
```

In [0]:

```
train.shape
```

Out[0]:

```
(7352, 564)
```

## Obtain the test data

In [0]:
```
# get the data from txt files to pandas dataffame
X_test = pd.read_csv('UCI_HAR_dataset/test/X_test.txt', delim_whitespac
e=True, header=None, names=features)

# 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=['Activit
y'], squeeze=True)
y_test_labels = y_test.map({1: 'WALKING', 2:'WALKING_UPSTAIRS',3:'WALKI
NG_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
test.sample()
```

```
D:\installed\Anaconda3\lib\site-packages\pandas\io\parsers.py:678: User
Warning: Duplicate names specified. This will raise an error in the fut
ure.
  return _read(filepath_or_buffer, kwds)
```

Out[0]:

| | tBodyAcc-mean()-X | tBodyAcc-mean()-Y | tBodyAcc-mean()-Z | tBodyAcc-std()-X | tBodyAcc-std()-Y | tBodyAcc-std()-Z | tBodyAcc-mad()-X |
|---|---|---|---|---|---|---|---|
| **2261** | 0.279196 | -0.018261 | -0.103376 | -0.996955 | -0.982959 | -0.988239 | -0.9972 |

1 rows × 564 columns

In [0]:
```
test.shape
```

```
Out[0]: (2947, 564)
```

# Data Cleaning

## 1. Check for Duplicates

```
In [0]: print('No of duplicates in train: {}'.format(sum(train.duplicated())))
        print('No of duplicates in test : {}'.format(sum(test.duplicated())))
```

```
No of duplicates in train: 0
No of duplicates in test : 0
```

## 2. Checking for NaN/null values

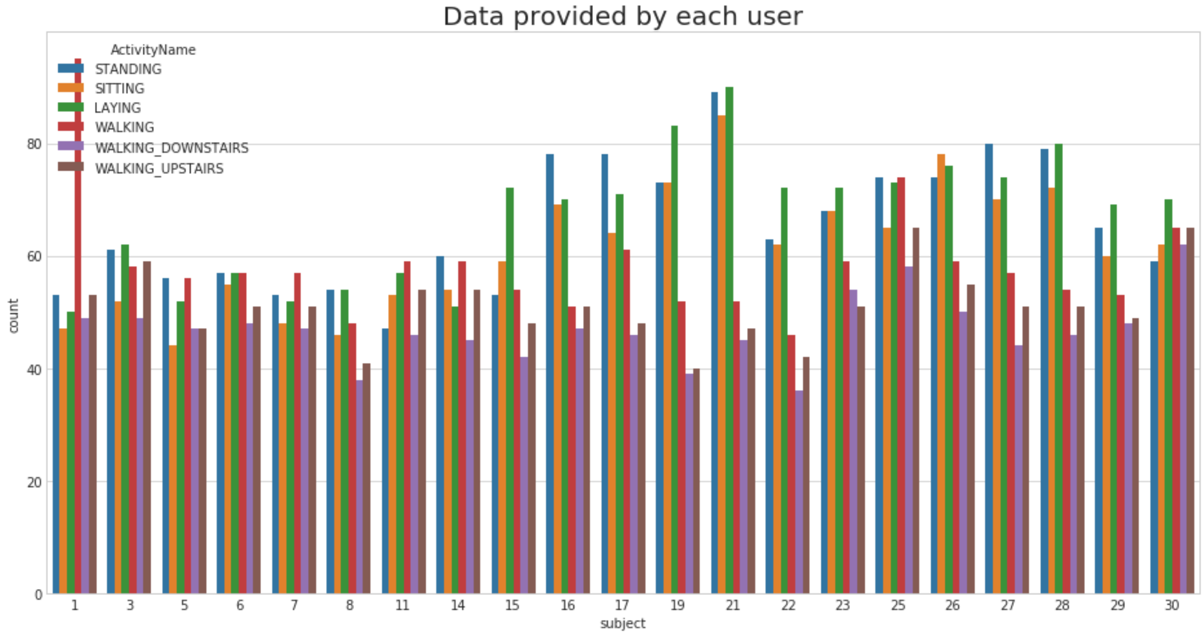
```
In [0]: print('We have {} NaN/Null values in train'.format(train.isnull().value
        s.sum()))
        print('We have {} NaN/Null values in test'.format(test.isnull().values.
        sum()))
```

## 3. Check for data imbalance

```
In [0]: import matplotlib.pyplot as plt
        import seaborn as sns
```

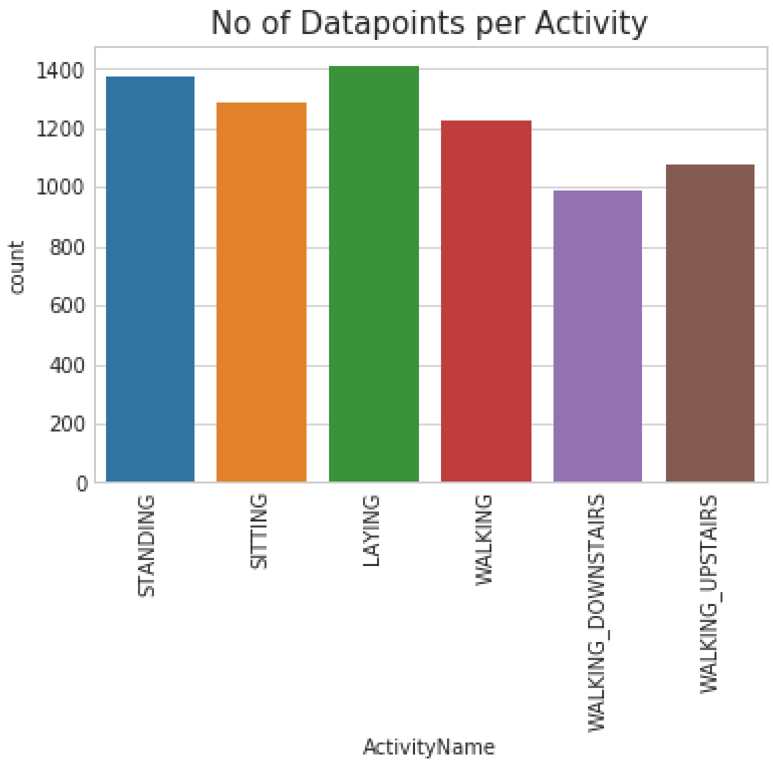
```
sns.set_style('whitegrid')
plt.rcParams['font.family'] = 'Dejavu Sans'
```

In [0]:
```
plt.figure(figsize=(16,8))
plt.title('Data provided by each user', fontsize=20)
sns.countplot(x='subject',hue='ActivityName', data = train)
plt.show()
```



> We have got almost same number of reading from all the subjects

In [0]:
```
plt.title('No of Datapoints per Activity', fontsize=15)
sns.countplot(train.ActivityName)
plt.xticks(rotation=90)
plt.show()
```



### Observation

> Our data is well balanced (almost)

## 4. Changing feature names

In [0]:

```
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

test.columns
```

Out[0]:

```
Index(['tBodyAccmeanX', 'tBodyAccmeanY', 'tBodyAccmeanZ', 'tBodyAccstd
X',
       'tBodyAccstdY', 'tBodyAccstdZ', 'tBodyAccmadX', 'tBodyAccmadY',
       'tBodyAccmadZ', 'tBodyAccmaxX',
       ...
       'angletBodyAccMeangravity', 'angletBodyAccJerkMeangravityMean',
       'angletBodyGyroMeangravityMean', 'angletBodyGyroJerkMeangravityM
ean',
       'angleXgravityMean', 'angleYgravityMean', 'angleZgravityMean',
       'subject', 'Activity', 'ActivityName'],
      dtype='object', length=564)
```

## 5. Save this dataframe in a csv files

In [0]:

```
train.to_csv('UCI_HAR_Dataset/csv_files/train.csv', index=False)
test.to_csv('UCI_HAR_Dataset/csv_files/test.csv', index=False)
```

# Exploratory Data Analysis

"***Without domain knowledge EDA has no meaning, without EDA a problem has no soul.***"

## 1. Featuring Engineering from Domain Knowledge

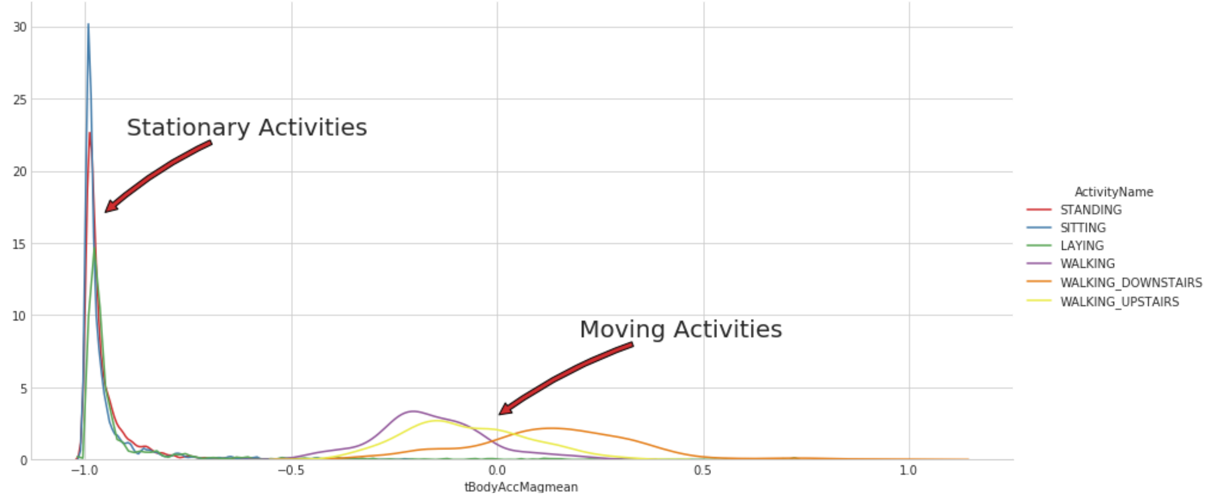
- **Static and Dynamic Activities**

- In static activities (sit, stand, lie down) motion information will not be very useful.
- In the dynamic activities (Walking, WalkingUpstairs,WalkingDownstairs) motion info will be significant.

### 2. Stationary and Moving activities are completely different

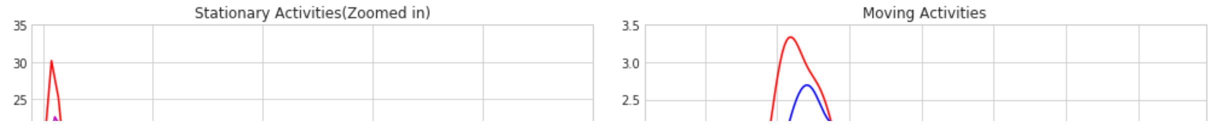
In [0]:

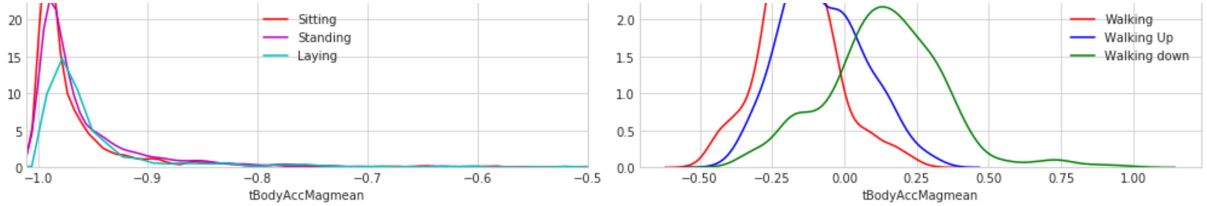
```
sns.set_palette("Set1", desat=0.80)
facetgrid = sns.FacetGrid(train, hue='ActivityName', size=6,aspect=2)
facetgrid.map(sns.distplot,'tBodyAccMagmean', hist=False)\
    .add_legend()
plt.annotate("Stationary Activities", xy=(-0.956,17), xytext=(-0.9, 23
), size=20,\
            va='center', ha='left',\
            arrowprops=dict(arrowstyle="simple",connectionstyle="arc3,r
ad=0.1"))

plt.annotate("Moving Activities", xy=(0,3), xytext=(0.2, 9), size=20,\
            va='center', ha='left',\
            arrowprops=dict(arrowstyle="simple",connectionstyle="arc3,r
ad=0.1"))
plt.show()
```



In [0]:

```
# for plotting purposes taking datapoints of each activity to a different dataframe
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=(14,7))
plt.subplot(2,2,1)
plt.title('Stationary Activities(Zoomed in)')
sns.distplot(df4['tBodyAccMagmean'],color = 'r',hist = False, label = 'Sitting')
sns.distplot(df5['tBodyAccMagmean'],color = 'm',hist = False,label = 'Standing')
sns.distplot(df6['tBodyAccMagmean'],color = 'c',hist = False, label = 'Laying')
plt.axis([-1.01, -0.5, 0, 35])
plt.legend(loc='center')

plt.subplot(2,2,2)
plt.title('Moving Activities')
sns.distplot(df1['tBodyAccMagmean'],color = 'red',hist = False, label = 'Walking')
sns.distplot(df2['tBodyAccMagmean'],color = 'blue',hist = False,label = 'Walking Up')
sns.distplot(df3['tBodyAccMagmean'],color = 'green',hist = False, label = 'Walking down')
plt.legend(loc='center right')


plt.tight_layout()
plt.show()
```

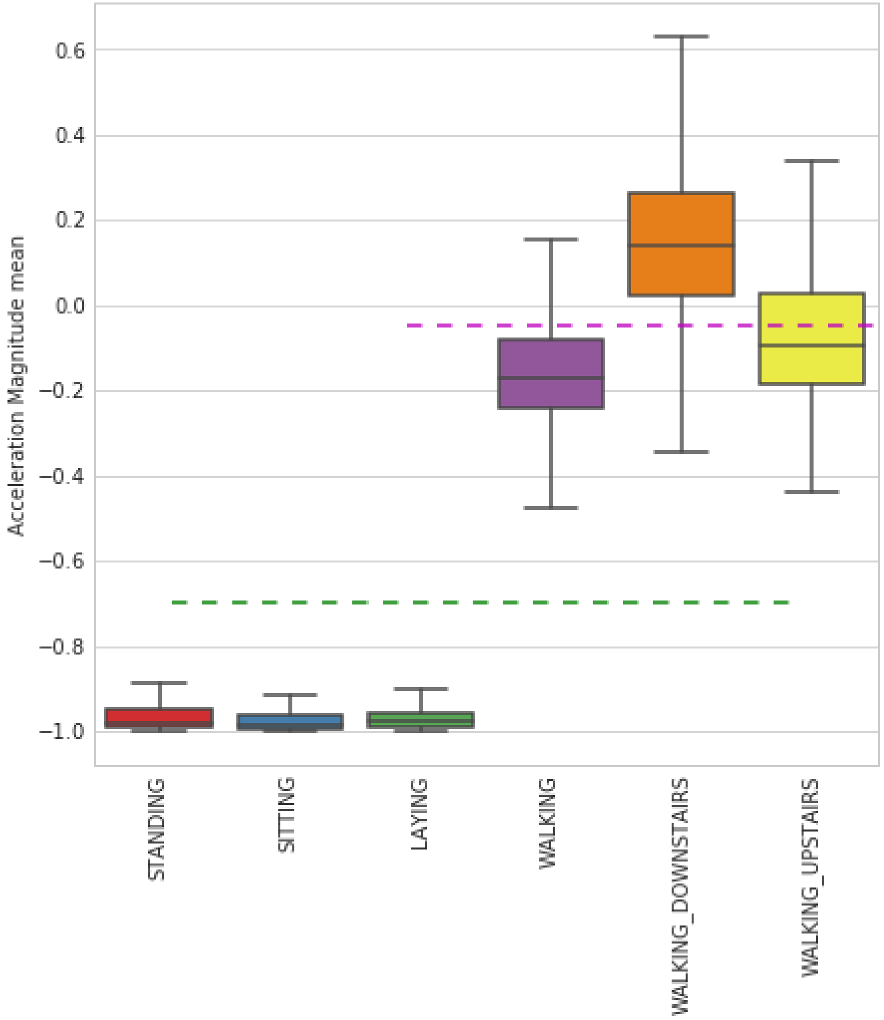




### 3. Magnitude of an acceleration can saperate it well

In [0]:

```
plt.figure(figsize=(7,7))
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=90)
plt.show()
```



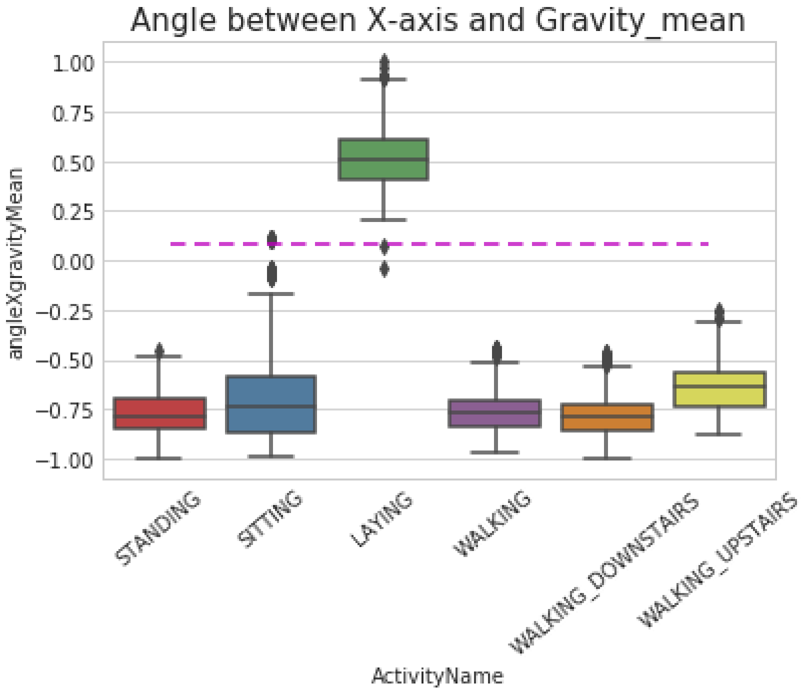
**Observations**:

- If tAccMean is < -0.8 then the Activities are either Standing or Sitting or Laying.
- If tAccMean is > -0.6 then the Activities are either Walking or WalkingDownstairs or WalkingUpstairs.
- If tAccMean > 0.0 then the Activity is WalkingDownstairs.

- We can classify 75% the Acitivity labels with some errors.

## 4. Position of GravityAccelerationComponants also matters

```
In [0]: sns.boxplot(x='ActivityName', y='angleXgravityMean', data=train)
        plt.axhline(y=0.08, xmin=0.1, xmax=0.9,c='m',dashes=(5,3))
        plt.title('Angle between X-axis and Gravity_mean', fontsize=15)
        plt.xticks(rotation = 40)
        plt.show()
```

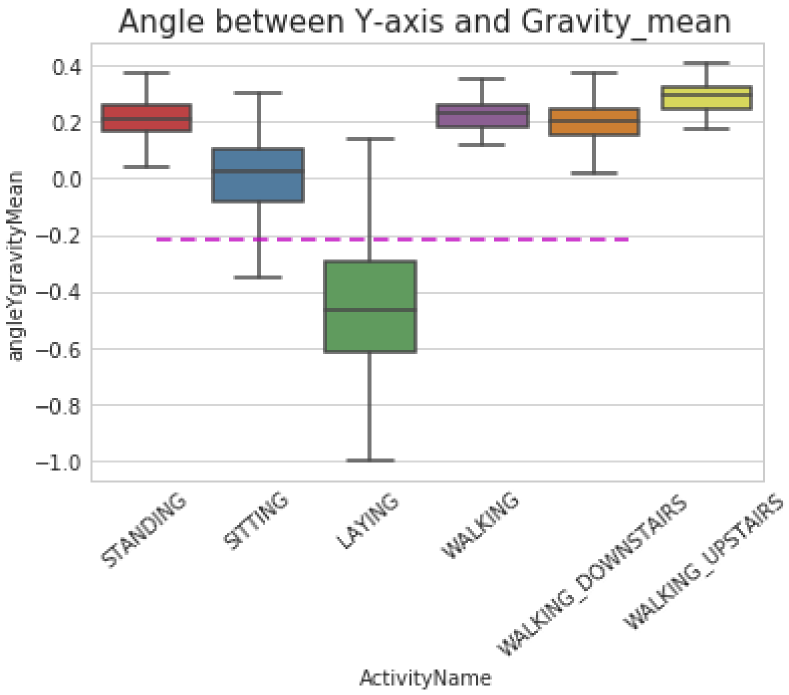


**Observations**:

- If angleX,gravityMean > 0 then Activity is Laying.
- We can classify all datapoints belonging to Laying activity with just a single if else statement.

In [0]:

```
sns.boxplot(x='ActivityName', y='angleYgravityMean', data = train, show
fliers=False)
plt.title('Angle between Y-axis and Gravity_mean', fontsize=15)
plt.xticks(rotation = 40)
plt.axhline(y=-0.22, xmin=0.1, xmax=0.8, dashes=(5,3), c='m')
plt.show()
```



# Apply t-sne on the data

In [0]:

```
import numpy as np
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
import seaborn as sns
```

In [0]:

```
# performs t-sne with different perplexity values and their repective plots..

def perform_tsne(X_data, y_data, perplexities, n_iter=1000, img_name_prefix='t-sne'):

    for index,perplexity in enumerate(perplexities):
        # perform t-sne
        print('\nperforming tsne with perplexity {} and with {} iterations at max'.format(perplexity, n_iter))
        X_reduced = TSNE(verbose=2, perplexity=perplexity).fit_transform(X_data)
        print('Done..')

        # prepare the data for seaborn
        print('Creating plot for this t-sne visualization..')
        df = pd.DataFrame({'x':X_reduced[:,0], 'y':X_reduced[:,1] ,'label':y_data})

        # draw the plot in appropriate place in the grid
        sns.lmplot(data=df, x='x', y='y', 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))
        img_name = img_name_prefix + '_perp_{}_iter_{}.png'.format(perplexity, n_iter)
        print('saving this plot as image in present working directory...')
        plt.savefig(img_name)
        plt.show()
        print('Done')
```

In [0]:

```
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])
```

```
performing tsne with perplexity 2 and with 1000 iterations at max
```

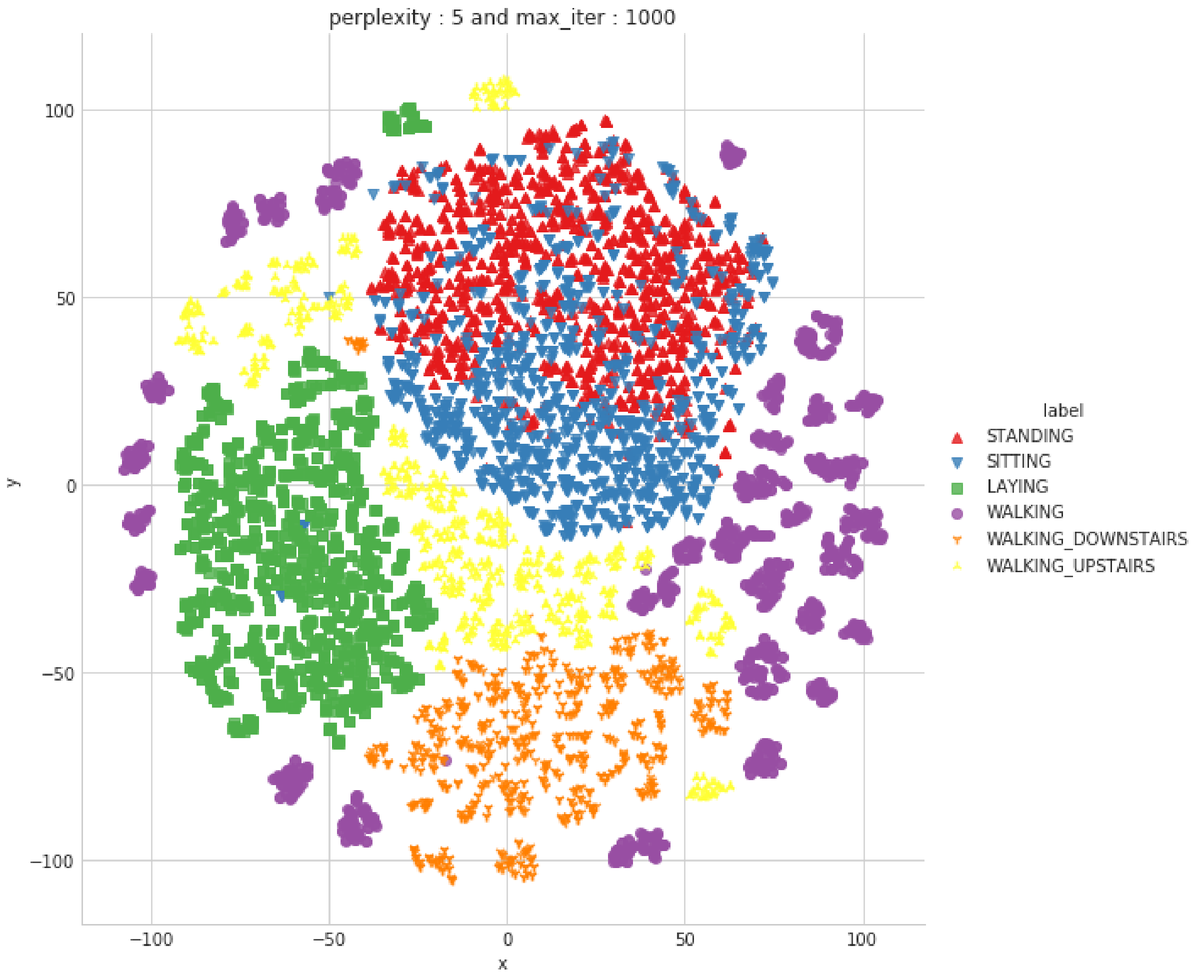
```
[t-SNE] Computing 7 nearest neighbors...
[t-SNE] Indexed 7352 samples in 0.426s...
[t-SNE] Computed neighbors for 7352 samples in 72.001s...
[t-SNE] Computed conditional probabilities for sample 1000 / 7352
[t-SNE] Computed conditional probabilities for sample 2000 / 7352
[t-SNE] Computed conditional 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: 0.635855
[t-SNE] Computed conditional probabilities in 0.071s
[t-SNE] Iteration 50: error = 124.8017578, gradient norm = 0.0253939 (5
0 iterations in 16.625s)
[t-SNE] Iteration 100: error = 107.2019501, gradient norm = 0.0284782
(50 iterations in 9.735s)
[t-SNE] Iteration 150: error = 100.9872894, gradient norm = 0.0185151
(50 iterations in 5.346s)
[t-SNE] Iteration 200: error = 97.6054382, gradient norm = 0.0142084 (5
0 iterations in 7.013s)
[t-SNE] Iteration 250: error = 95.3084183, gradient norm = 0.0132592 (5
0 iterations in 5.703s)
[t-SNE] KL divergence after 250 iterations with early exaggeration: 95.
308418
[t-SNE] Iteration 300: error = 4.1209540, gradient norm = 0.0015668 (50
iterations in 7.156s)
[t-SNE] Iteration 350: error = 3.2113254, gradient norm = 0.0009953 (50
iterations in 8.022s)
[t-SNE] Iteration 400: error = 2.7819963, gradient norm = 0.0007203 (50
iterations in 9.419s)
[t-SNE] Iteration 450: error = 2.5178111, gradient norm = 0.0005655 (50
iterations in 9.370s)
[t-SNE] Iteration 500: error = 2.3341548, gradient norm = 0.0004804 (50
iterations in 7.681s)
[t-SNE] Iteration 550: error = 2.1961622, gradient norm = 0.0004183 (50
iterations in 7.097s)
[t-SNE] Iteration 600: error = 2.0867445, gradient norm = 0.0003664 (50
iterations in 9.274s)
```

```
[t-SNE] Iteration 650: error = 1.9967778, gradient norm = 0.0003279 (50
iterations in 7.697s)
[t-SNE] Iteration 700: error = 1.9210005, gradient norm = 0.0002984 (50
iterations in 8.174s)
[t-SNE] Iteration 750: error = 1.8558111, gradient norm = 0.0002776 (50
iterations in 9.747s)
[t-SNE] Iteration 800: error = 1.7989457, gradient norm = 0.0002569 (50
iterations in 8.687s)
[t-SNE] Iteration 850: error = 1.7490212, gradient norm = 0.0002394 (50
iterations in 8.407s)
[t-SNE] Iteration 900: error = 1.7043383, gradient norm = 0.0002224 (50
iterations in 8.351s)
[t-SNE] Iteration 950: error = 1.6641431, gradient norm = 0.0002098 (50
iterations in 7.841s)
[t-SNE] Iteration 1000: error = 1.6279151, gradient norm = 0.0001989 (5
0 iterations in 5.623s)
[t-SNE] Error after 1000 iterations: 1.627915
Done..
Creating plot for this t-sne visualization..
saving this plot as image in present working directory...
```

```
Done

performing tsne with perplexity 5 and with 1000 iterations at max
[t-SNE] Computing 16 nearest neighbors...
[t-SNE] Indexed 7352 samples in 0.263s...
[t-SNE] Computed neighbors for 7352 samples in 48.983s...
[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 0.122s
[t-SNE] Iteration 50: error = 114.1862640, gradient norm = 0.0184120 (5
0 iterations in 55.655s)
[t-SNE] Iteration 100: error = 97.6535568, gradient norm = 0.0174309 (5
0 iterations in 12.580s)
[t-SNE] Iteration 150: error = 93.1900101, gradient norm = 0.0101048 (5
0 iterations in 9.180s)
[t-SNE] Iteration 200: error = 91.2315445, gradient norm = 0.0074560 (5
0 iterations in 10.340s)
[t-SNE] Iteration 250: error = 90.0714417, gradient norm = 0.0057667 (5
0 iterations in 9.458s)
[t-SNE] KL divergence after 250 iterations with early exaggeration: 90.
071442
[t-SNE] Iteration 300: error = 3.5796804, gradient norm = 0.0014691 (50
iterations in 8.718s)
[t-SNE] Iteration 350: error = 2.8173938, gradient norm = 0.0007508 (50
iterations in 10.180s)
[t-SNE] Iteration 400: error = 2.4344938, gradient norm = 0.0005251 (50
iterations in 10.506s)
[t-SNE] Iteration 450: error = 2.2156141, gradient norm = 0.0004069 (50
iterations in 10.072s)
[t-SNE] Iteration 500: error = 2.0703306, gradient norm = 0.0003340 (50
iterations in 10.511s)
[t-SNE] Iteration 550: error = 1.9646366, gradient norm = 0.0002816 (50
iterations in 9.792s)
[t-SNE] Iteration 600: error = 1.8835558, gradient norm = 0.0002471 (50
iterations in 9.098s)
[t-SNE] Iteration 650: error = 1.8184001, gradient norm = 0.0002184 (50
iterations in 8.656s)
[t-SNE] Iteration 700: error = 1.7647167, gradient norm = 0.0001961 (50
iterations in 9.063s)
[t-SNE] Iteration 750: error = 1.7193680, gradient norm = 0.0001796 (50
iterations in 9.754s)
```

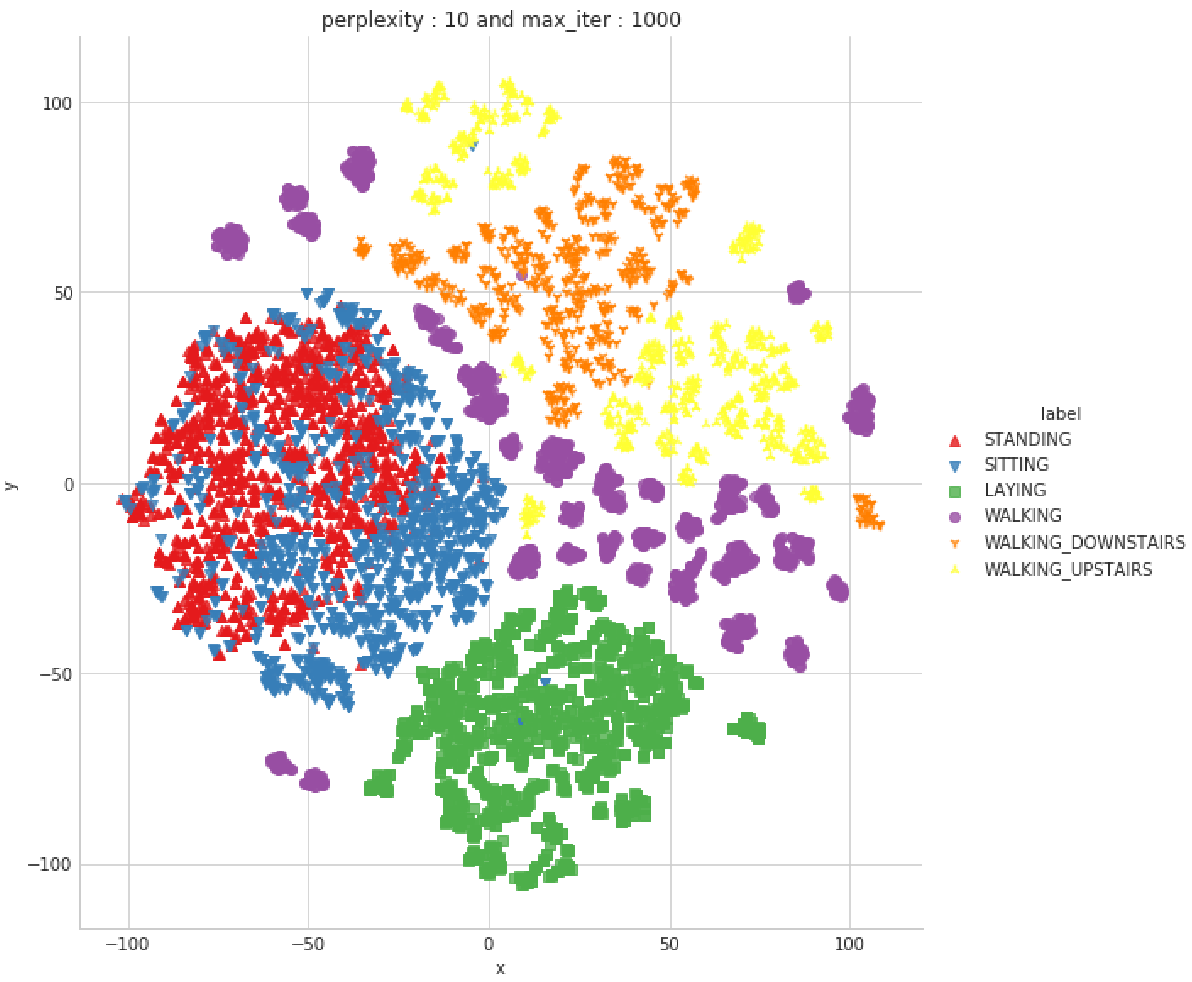
```
[t-SNE] Iteration 800: error = 1.6803776, gradient norm = 0.0001655 (50
iterations in 9.540s)
[t-SNE] Iteration 850: error = 1.6465144, gradient norm = 0.0001538 (50
iterations in 9.953s)
[t-SNE] Iteration 900: error = 1.6166563, gradient norm = 0.0001421 (50
iterations in 10.270s)
[t-SNE] Iteration 950: error = 1.5901035, gradient norm = 0.0001335 (50
iterations in 6.609s)
[t-SNE] Iteration 1000: error = 1.5664237, gradient norm = 0.0001257 (5
0 iterations in 8.553s)
[t-SNE] Error after 1000 iterations: 1.566424
Done..
Creating plot for this t-sne visualization..
saving this plot as image in present working directory...
```



```
Done

performing tsne with perplexity 10 and with 1000 iterations at max
[t-SNE] Computing 31 nearest neighbors...
[t-SNE] Indexed 7352 samples in 0.410s...
[t-SNE] Computed neighbors for 7352 samples in 64.801s...
[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 0.214s
[t-SNE] Iteration 50: error = 106.0169220, gradient norm = 0.0194293 (5
0 iterations in 24.550s)
[t-SNE] Iteration 100: error = 90.3036194, gradient norm = 0.0097653 (5
0 iterations in 11.936s)
[t-SNE] Iteration 150: error = 87.3132935, gradient norm = 0.0053059 (5
0 iterations in 11.246s)
[t-SNE] Iteration 200: error = 86.1169128, gradient norm = 0.0035844 (5
0 iterations in 11.864s)
[t-SNE] Iteration 250: error = 85.4133606, gradient norm = 0.0029100 (5
0 iterations in 11.944s)
[t-SNE] KL divergence after 250 iterations with early exaggeration: 85.
413361
[t-SNE] Iteration 300: error = 3.1394315, gradient norm = 0.0013976 (50
iterations in 11.742s)
[t-SNE] Iteration 350: error = 2.4929206, gradient norm = 0.0006466 (50
iterations in 11.627s)
[t-SNE] Iteration 400: error = 2.1733041, gradient norm = 0.0004230 (50
iterations in 11.846s)
[t-SNE] Iteration 450: error = 1.9884514, gradient norm = 0.0003124 (50
iterations in 11.405s)
[t-SNE] Iteration 500: error = 1.8702440, gradient norm = 0.0002514 (50
iterations in 11.320s)
[t-SNE] Iteration 550: error = 1.7870129, gradient norm = 0.0002107 (50
iterations in 12.009s)
[t-SNE] Iteration 600: error = 1.7246909, gradient norm = 0.0001824 (50
iterations in 10.632s)
[t-SNE] Iteration 650: error = 1.6758548, gradient norm = 0.0001590 (50
iterations in 11.270s)
[t-SNE] Iteration 700: error = 1.6361949, gradient norm = 0.0001451 (50
iterations in 12.072s)
[t-SNE] Iteration 750: error = 1.6034756, gradient norm = 0.0001305 (50
iterations in 11.607s)
```

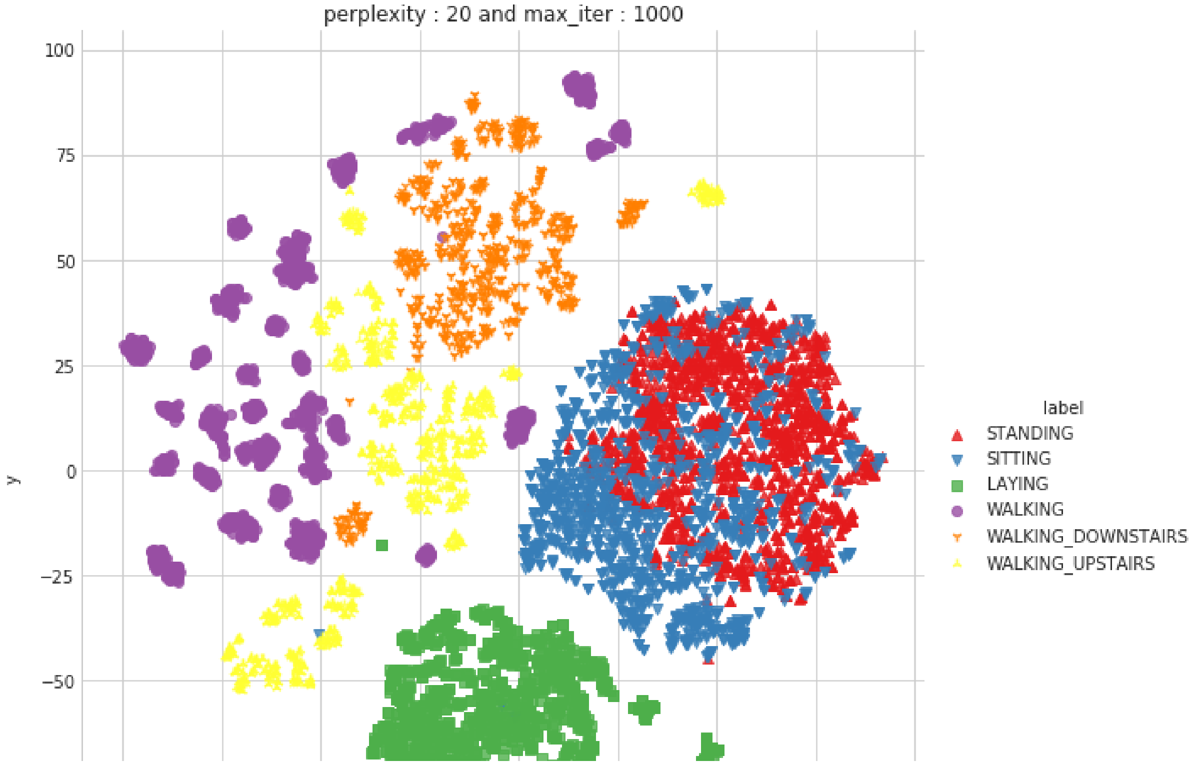
```
[t-SNE] Iteration 800: error = 1.5761518, gradient norm = 0.0001188 (50
iterations in 9.409s)
[t-SNE] Iteration 850: error = 1.5527289, gradient norm = 0.0001113 (50
iterations in 8.309s)
[t-SNE] Iteration 900: error = 1.5328671, gradient norm = 0.0001021 (50
iterations in 9.433s)
[t-SNE] Iteration 950: error = 1.5152045, gradient norm = 0.0000974 (50
iterations in 11.488s)
[t-SNE] Iteration 1000: error = 1.4999681, gradient norm = 0.0000933 (5
0 iterations in 10.593s)
[t-SNE] Error after 1000 iterations: 1.499968
Done..
Creating plot for this t-sne visualization..
saving this plot as image in present working directory...
```



```
Done

performing tsne with perplexity 20 and with 1000 iterations at max
[t-SNE] Computing 61 nearest neighbors...
[t-SNE] Indexed 7352 samples in 0.425s...
[t-SNE] Computed neighbors for 7352 samples in 61.792s...
[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
```

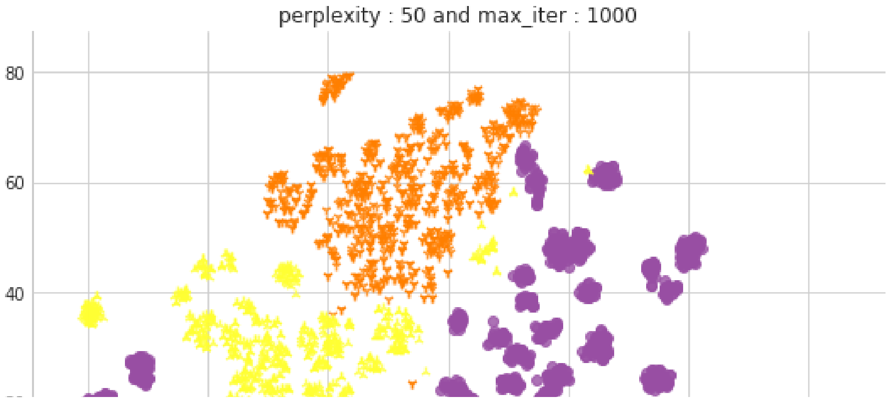
```
[t-SNE] Computed conditional probabilities for sample 4000 / 7352
[t-SNE] Computed conditional probabilities for sample 5000 / 7352
[t-SNE] Computed conditional probabilities for sample 6000 / 7352
[t-SNE] Computed conditional probabilities for sample 7000 / 7352
[t-SNE] Computed conditional probabilities for sample 7352 / 7352
[t-SNE] Mean sigma: 1.274335
[t-SNE] Computed conditional probabilities in 0.355s
[t-SNE] Iteration 50: error = 97.5202179, gradient norm = 0.0223863 (50
iterations in 21.168s)
[t-SNE] Iteration 100: error = 83.9500732, gradient norm = 0.0059110 (5
0 iterations in 17.306s)
[t-SNE] Iteration 150: error = 81.8804779, gradient norm = 0.0035797 (5
0 iterations in 14.258s)
[t-SNE] Iteration 200: error = 81.1615143, gradient norm = 0.0022536 (5
0 iterations in 14.130s)
[t-SNE] Iteration 250: error = 80.7704086, gradient norm = 0.0018108 (5
0 iterations in 15.340s)

[t-SNE] KL divergence after 250 iterations with early exaggeration: 80.
770409
[t-SNE] Iteration 300: error = 2.6957574, gradient norm = 0.0012993 (50
iterations in 13.605s)
[t-SNE] Iteration 350: error = 2.1637220, gradient norm = 0.0005765 (50
iterations in 13.248s)
[t-SNE] Iteration 400: error = 1.9143614, gradient norm = 0.0003474 (50
iterations in 14.774s)
[t-SNE] Iteration 450: error = 1.7684202, gradient norm = 0.0002458 (50
iterations in 15.502s)
[t-SNE] Iteration 500: error = 1.6744757, gradient norm = 0.0001923 (50
iterations in 14.808s)
[t-SNE] Iteration 550: error = 1.6101606, gradient norm = 0.0001575 (50
iterations in 14.043s)
[t-SNE] Iteration 600: error = 1.5641028, gradient norm = 0.0001344 (50
iterations in 15.769s)
[t-SNE] Iteration 650: error = 1.5291905, gradient norm = 0.0001182 (50
iterations in 15.834s)
[t-SNE] Iteration 700: error = 1.5024391, gradient norm = 0.0001055 (50
iterations in 15.398s)
[t-SNE] Iteration 750: error = 1.4809053, gradient norm = 0.0000965 (50
iterations in 14.594s)
[t-SNE] Iteration 800: error = 1.4631859, gradient norm = 0.0000884 (50
```

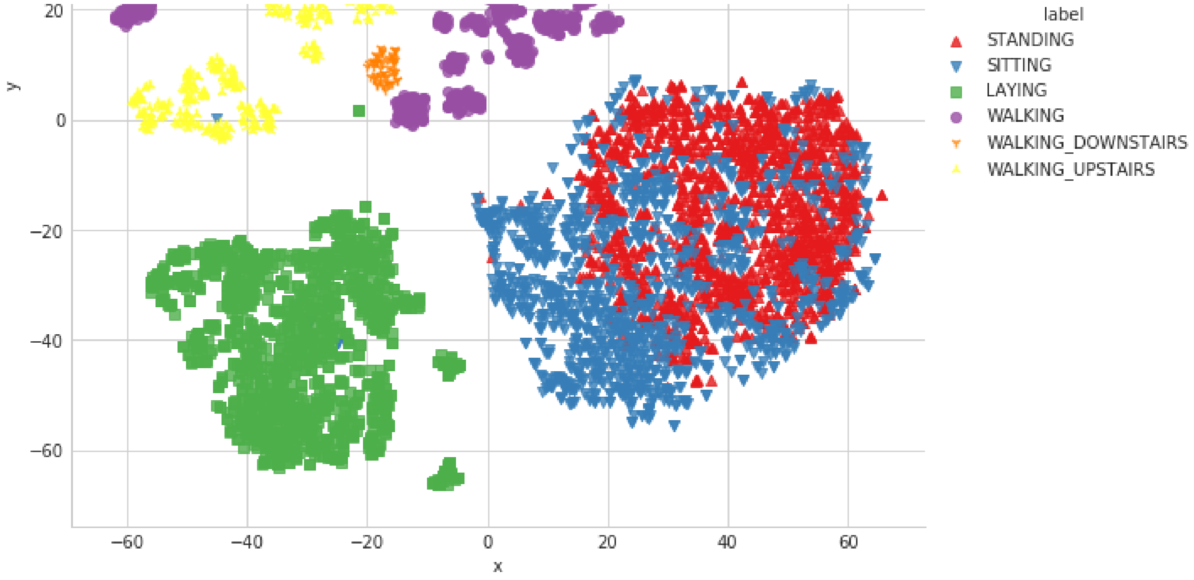
```
iterations in 15.025s)
[t-SNE] Iteration 850: error = 1.4486470, gradient norm = 0.0000832 (50
iterations in 14.060s)
[t-SNE] Iteration 900: error = 1.4367288, gradient norm = 0.0000804 (50
iterations in 12.389s)
[t-SNE] Iteration 950: error = 1.4270191, gradient norm = 0.0000761 (50
iterations in 10.392s)
[t-SNE] Iteration 1000: error = 1.4189968, gradient norm = 0.0000787 (5
0 iterations in 12.355s)
[t-SNE] Error after 1000 iterations: 1.418997
Done..
Creating plot for this t-sne visualization..
saving this plot as image in present working directory...
```



```
Done

performing tsne with perplexity 50 and with 1000 iterations at max
[t-SNE] Computing 151 nearest neighbors...
[t-SNE] Indexed 7352 samples in 0.376s...
[t-SNE] Computed neighbors for 7352 samples in 73.164s...
[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 0.844s
[t-SNE] Iteration 50: error = 86.1525574, gradient norm = 0.0242986 (50
iterations in 36.249s)
[t-SNE] Iteration 100: error = 75.9874649, gradient norm = 0.0061005 (5
0 iterations in 30.453s)
[t-SNE] Iteration 150: error = 74.7072296, gradient norm = 0.0024708 (5
0 iterations in 28.461s)
[t-SNE] Iteration 200: error = 74.2736282, gradient norm = 0.0018644 (5
0 iterations in 27.735s)
[t-SNE] Iteration 250: error = 74.0722427, gradient norm = 0.0014078 (5
0 iterations in 26.835s)
[t-SNE] KL divergence after 250 iterations with early exaggeration: 74.
072243
[t-SNE] Iteration 300: error = 2.1539080, gradient norm = 0.0011796 (50
iterations in 25.445s)
[t-SNE] Iteration 350: error = 1.7567128, gradient norm = 0.0004845 (50
iterations in 21.282s)
[t-SNE] Iteration 400: error = 1.5888531, gradient norm = 0.0002798 (50
iterations in 21.015s)
[t-SNE] Iteration 450: error = 1.4956820, gradient norm = 0.0001894 (50
iterations in 23.332s)
```

```
[t-SNE] Iteration 500: error = 1.4359720, gradient norm = 0.0001420 (50
iterations in 23.083s)
[t-SNE] Iteration 550: error = 1.3947564, gradient norm = 0.0001117 (50
iterations in 19.626s)
[t-SNE] Iteration 600: error = 1.3653858, gradient norm = 0.0000949 (50
iterations in 22.752s)
[t-SNE] Iteration 650: error = 1.3441534, gradient norm = 0.0000814 (50
iterations in 23.972s)
[t-SNE] Iteration 700: error = 1.3284039, gradient norm = 0.0000742 (50
iterations in 20.636s)
[t-SNE] Iteration 750: error = 1.3171139, gradient norm = 0.0000700 (50
iterations in 20.407s)
[t-SNE] Iteration 800: error = 1.3085558, gradient norm = 0.0000657 (50
iterations in 24.951s)
[t-SNE] Iteration 850: error = 1.3017821, gradient norm = 0.0000603 (50
iterations in 24.719s)

[t-SNE] Iteration 900: error = 1.2962619, gradient norm = 0.0000586 (50
iterations in 24.500s)
[t-SNE] Iteration 950: error = 1.2914882, gradient norm = 0.0000573 (50
iterations in 24.132s)
[t-SNE] Iteration 1000: error = 1.2874244, gradient norm = 0.0000546 (5
0 iterations in 22.840s)
[t-SNE] Error after 1000 iterations: 1.287424
Done..
Creating plot for this t-sne visualization..
saving this plot as image in present working directory...
```





```
Done
```

In [0]:
```
import numpy as np
import pandas as pd
```

## Obtain the train and test data

In [0]:
```
train = pd.read_csv('UCI_HAR_dataset/csv_files/train.csv')
test = pd.read_csv('UCI_HAR_dataset/csv_files/test.csv')
print(train.shape, test.shape)
```

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

In [0]:

```
train.head(3)
```

Out[0]:

|  | tBodyAccmeanX | tBodyAccmeanY | tBodyAccmeanZ | tBodyAccstdX | tBodyAccstdY | tB |
|---|---|---|---|---|---|---|
| **0** | 0.288585 | -0.020294 | -0.132905 | -0.995279 | -0.983111 | -0. |
| **1** | 0.278419 | -0.016411 | -0.123520 | -0.998245 | -0.975300 | -0. |
| **2** | 0.279653 | -0.019467 | -0.113462 | -0.995380 | -0.967187 | -0. |

3 rows × 564 columns

In [0]:

```
# get X_train and y_train from csv files
X_train = train.drop(['subject', 'Activity', 'ActivityName'], axis=1)
y_train = train.ActivityName
```

In [0]:

```
# get X_test and y_test from test csv file
X_test = test.drop(['subject', 'Activity', 'ActivityName'], axis=1)
y_test = test.ActivityName
```

In [0]:

```
print('X_train and y_train : ({},{})'.format(X_train.shape, y_train.sha
pe))
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,))
```

# Let's model with our data

### Labels that are useful in plotting confusion matrix

In [0]:

```
labels=['LAYING', 'SITTING','STANDING','WALKING','WALKING_DOWNSTAIRS',
'WALKING_UPSTAIRS']
```

### Function to plot the confusion matrix

In [0]:

```
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 [0]:

```
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()

    # time at which model starts training
    train_start_time = datetime.now()
    print('training the model..')
    model.fit(X_train, y_train)
    print('Done \n \n')
    train_end_time = datetime.now()
    results['training_time'] =  train_end_time - train_start_time
    print('training_time(HH:MM:SS.ms) - {}\n\n'.format(results['trainin
g_time']))


    # predict test data
    print('Predicting test data')
    test_start_time = datetime.now()
    y_pred = model.predict(X_test)
    test_end_time = datetime.now()
    print('Done \n \n')
    results['testing_time'] = test_end_time - test_start_time
    print('testing time(HH:MM:SS:ms) - {}\n\n'.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, tit
le='Normalized confusion matrix', cmap = cm_cmap)
    plt.show()

    # get classification report
    print('-------------------------')
    print('| Classifiction Report |')
    print('-------------------------')
    classification_report = metrics.classification_report(y_test, y_pre
d)
    # 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 [0]:

```
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_))
```

# 1. Logistic Regression with Grid Search

In [0]:

```
from sklearn import linear_model
from sklearn import metrics

from sklearn.model_selection import GridSearchCV
```

In [0]:

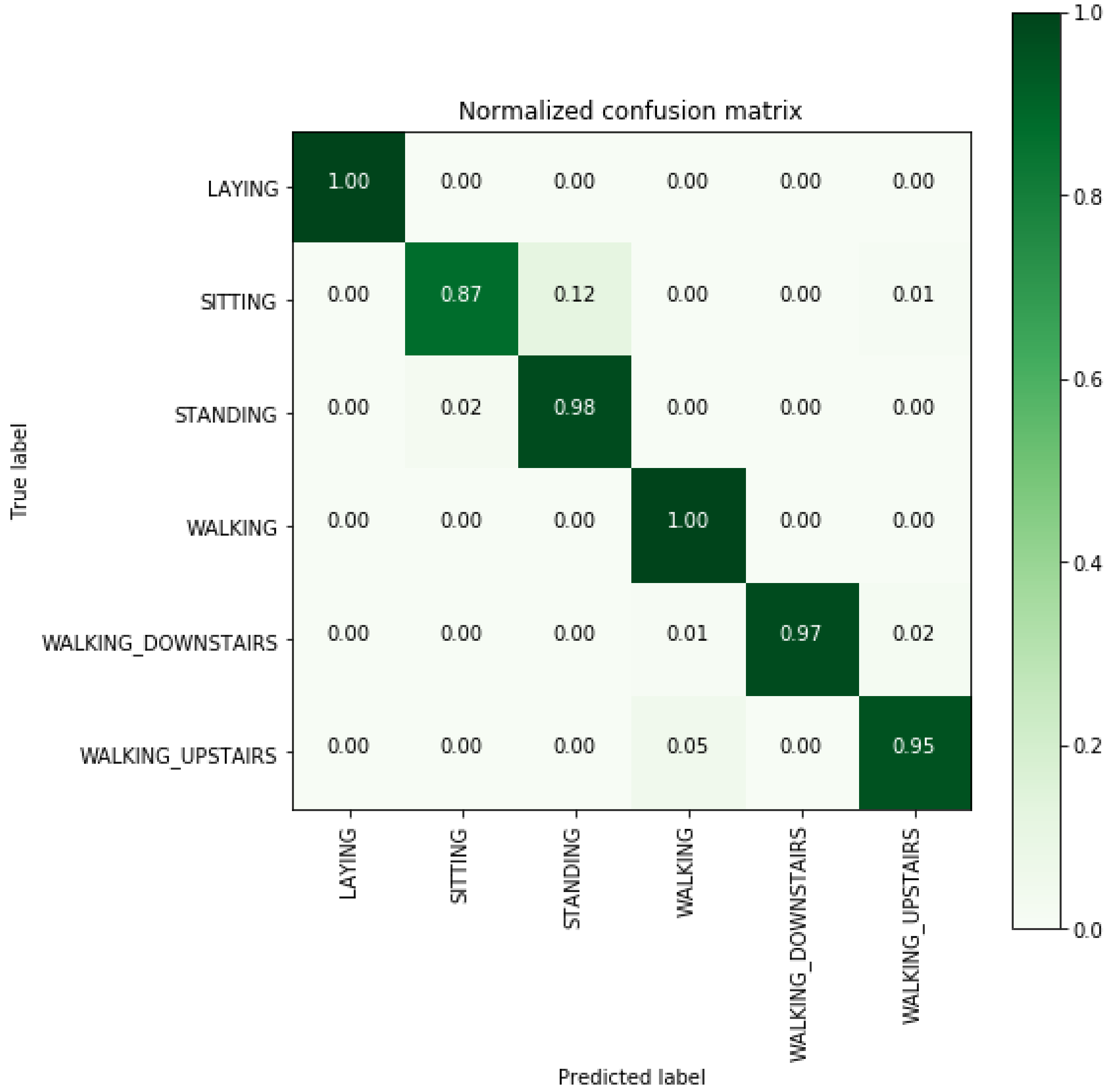
```
# start Grid search
parameters = {'C':[0.01, 0.1, 1, 10, 20, 30], 'penalty':['l2','l1']}
log_reg = linear_model.LogisticRegression()
log_reg_grid = GridSearchCV(log_reg, param_grid=parameters, cv=3, verbo
se=1, n_jobs=-1)
log_reg_grid_results =  perform_model(log_reg_grid, X_train, y_train, X
_test,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

```
training the model..
Fitting 3 folds for each of 12 candidates, totalling 36 fits
```

```
[Parallel(n_jobs=-1)]: Done  36 out of  36 | elapsed:  1.2min finished
```

```
Done


training_time(HH:MM:SS.ms) - 0:01:25.843810


Predicting test data
Done
```

```
testing time(HH:MM:SS:ms) - 0:00:00.009192


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

    0.9626739056667798


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

 [[537   0   0   0   0   0]
 [  1 428  58   0   0   4]
 [  0  12 519   1   0   0]
 [  0   0   0 495   1   0]
 [  0   0   0   3 409   8]
 [  0   0   0  22   0 449]]
```

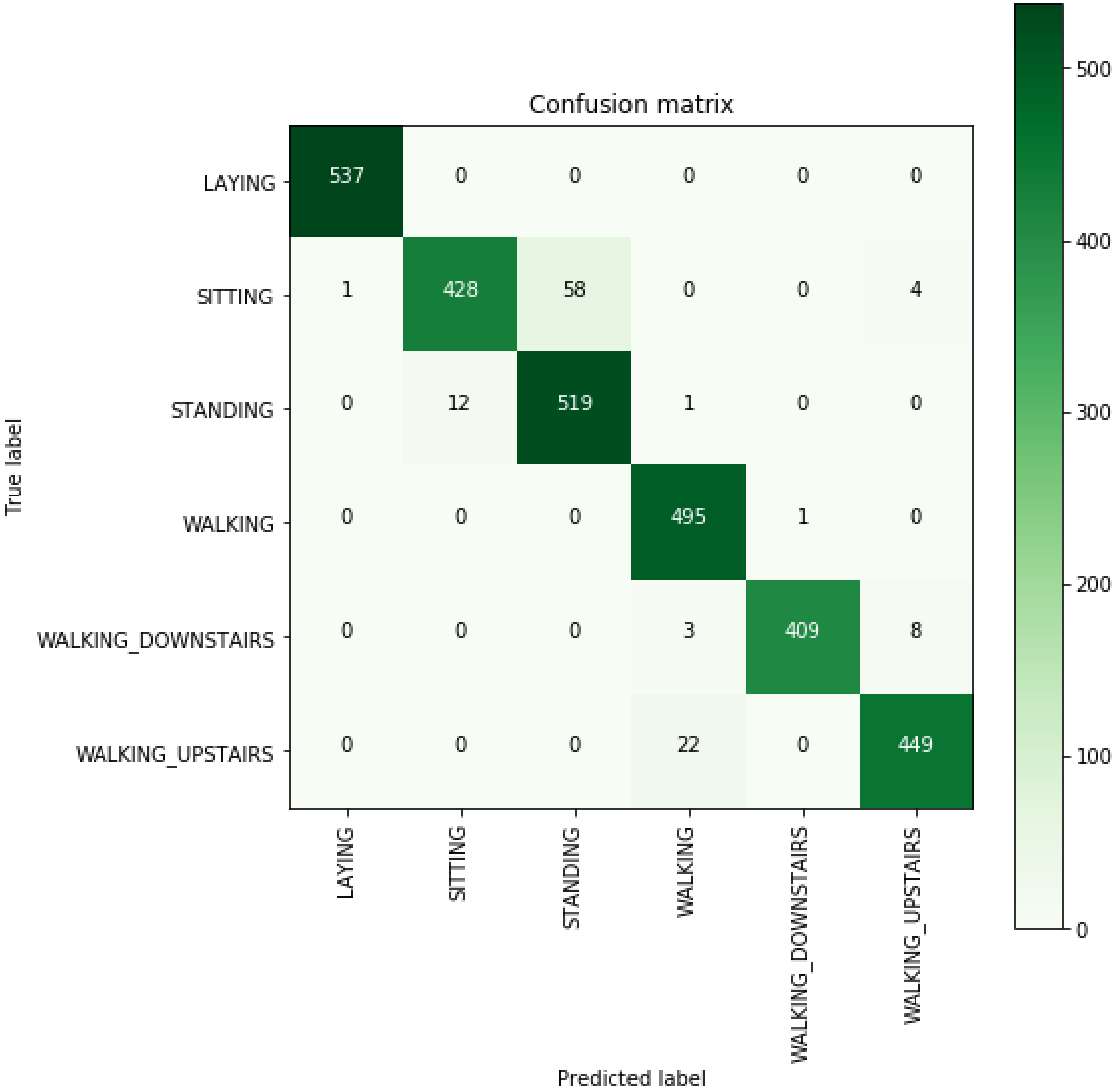


```
-------------------------
| Classifiction Report |
-------------------------
                    precision    recall  f1-score   support

            LAYING       1.00      1.00      1.00       537
```

```
           SITTING       0.97      0.87      0.92       491
          STANDING       0.90      0.98      0.94       532
           WALKING       0.95      1.00      0.97       496
WALKING_DOWNSTAIRS       1.00      0.97      0.99       420
  WALKING_UPSTAIRS       0.97      0.95      0.96       471

       avg / total       0.96      0.96      0.96      2947
```

In [0]:

```
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()
```



|  | LAYING | SITTING | STANDING | WALKING | WALKING_DOWNSTAIRS | WALKING_UPSTAIRS |
|---|---|---|---|---|---|---|
| LAYING | 537 | 0 | 0 | 0 | 0 | 0 |
| SITTING | 1 | 428 | 58 | 0 | 0 | 4 |
| STANDING | 0 | 12 | 519 | 1 | 0 | 0 |
| WALKING | 0 | 0 | 0 | 495 | 1 | 0 |
| WALKING_DOWNSTAIRS | 0 | 0 | 0 | 3 | 409 | 8 |
| WALKING_UPSTAIRS | 0 | 0 | 0 | 22 | 0 | 449 |

In [0]:
```python
# 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_int
ercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=
1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.00
01,
          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
```

## 2. Linear SVC with GridSearch

In [0]:

```python
from sklearn.svm import LinearSVC
```

In [0]:

```python
parameters = {'C':[0.125, 0.5, 1, 2, 8, 16]}
```

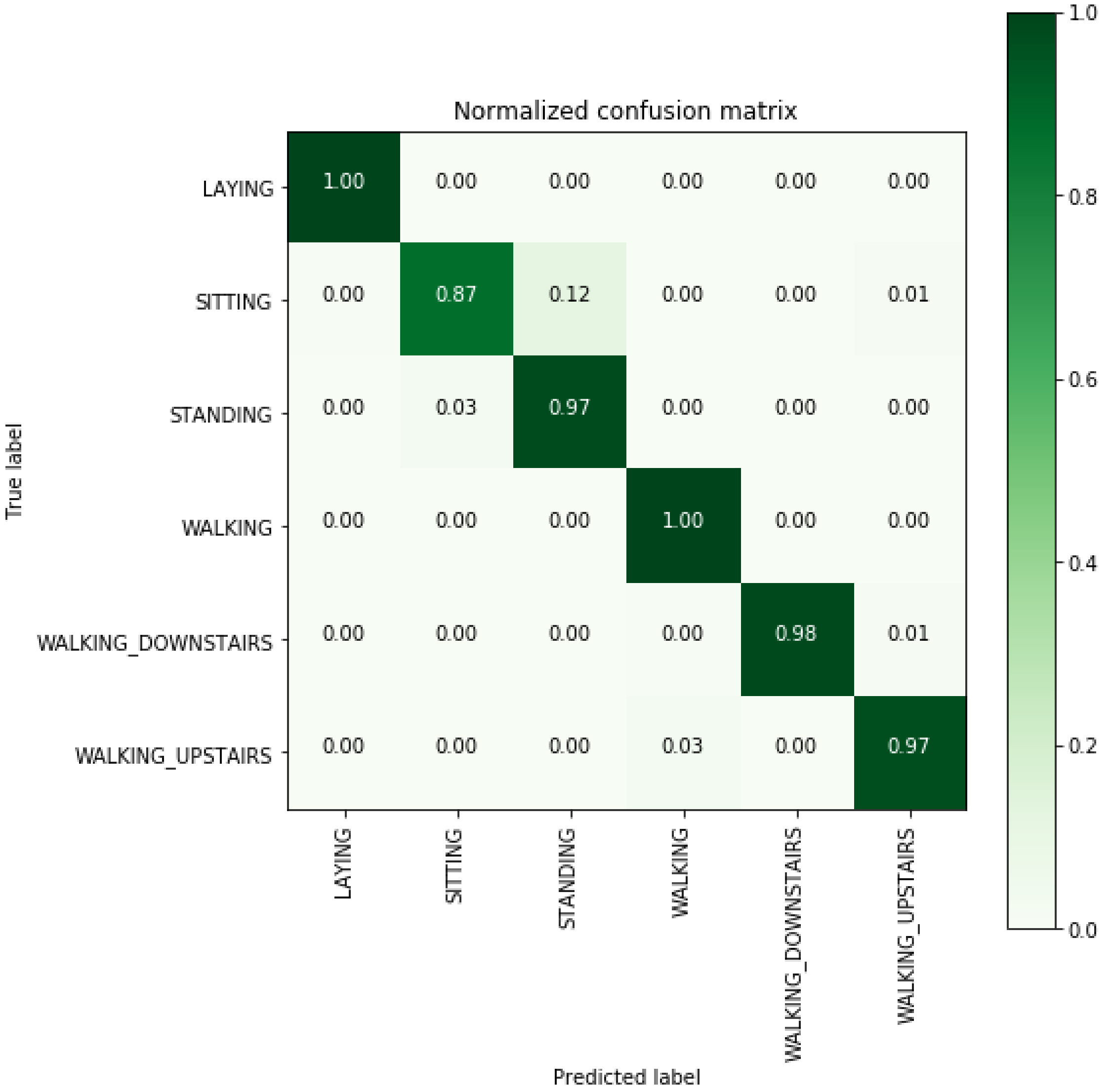
```
lr_svc = LinearSVC(tol=0.00005)
lr_svc_grid = GridSearchCV(lr_svc, param_grid=parameters, n_jobs=-1, ve
rbose=1)
lr_svc_grid_results = perform_model(lr_svc_grid, X_train, y_train, X_te
st, y_test, class_labels=labels)
```

```
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
```

```
                    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 [0]:

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

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

        LinearSVC(C=8, class_weight=None, dual=True, fit_intercept=Tru
e,
     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
```

> We can choose ***Logistic regression*** or ***Linear SVC*** or ***rbf SVM***.

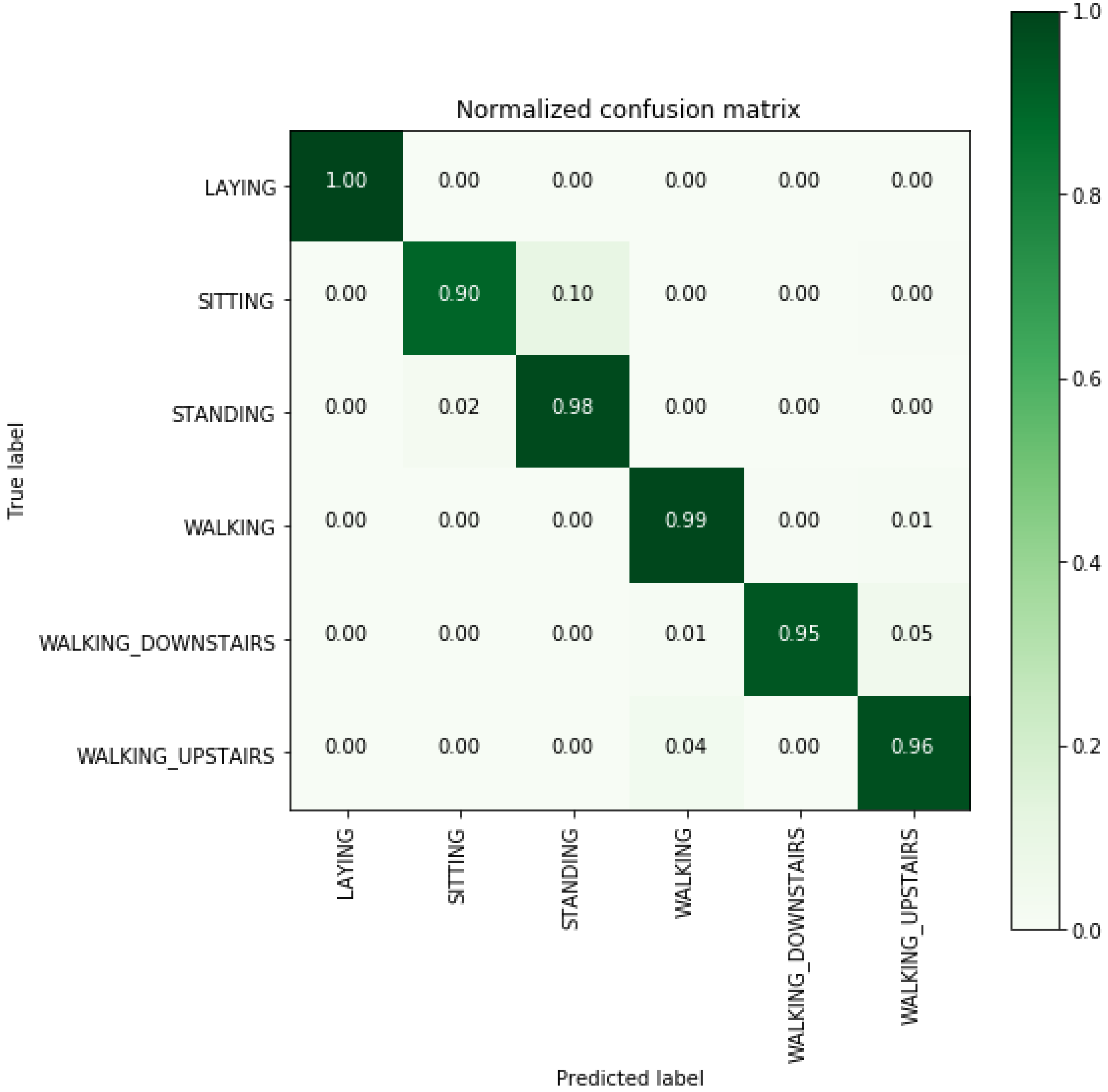
# 3. Kernel SVM with GridSearch

In [0]:

```
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=labels)
```

```
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 [0]:

```
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='rb
f',
  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
```

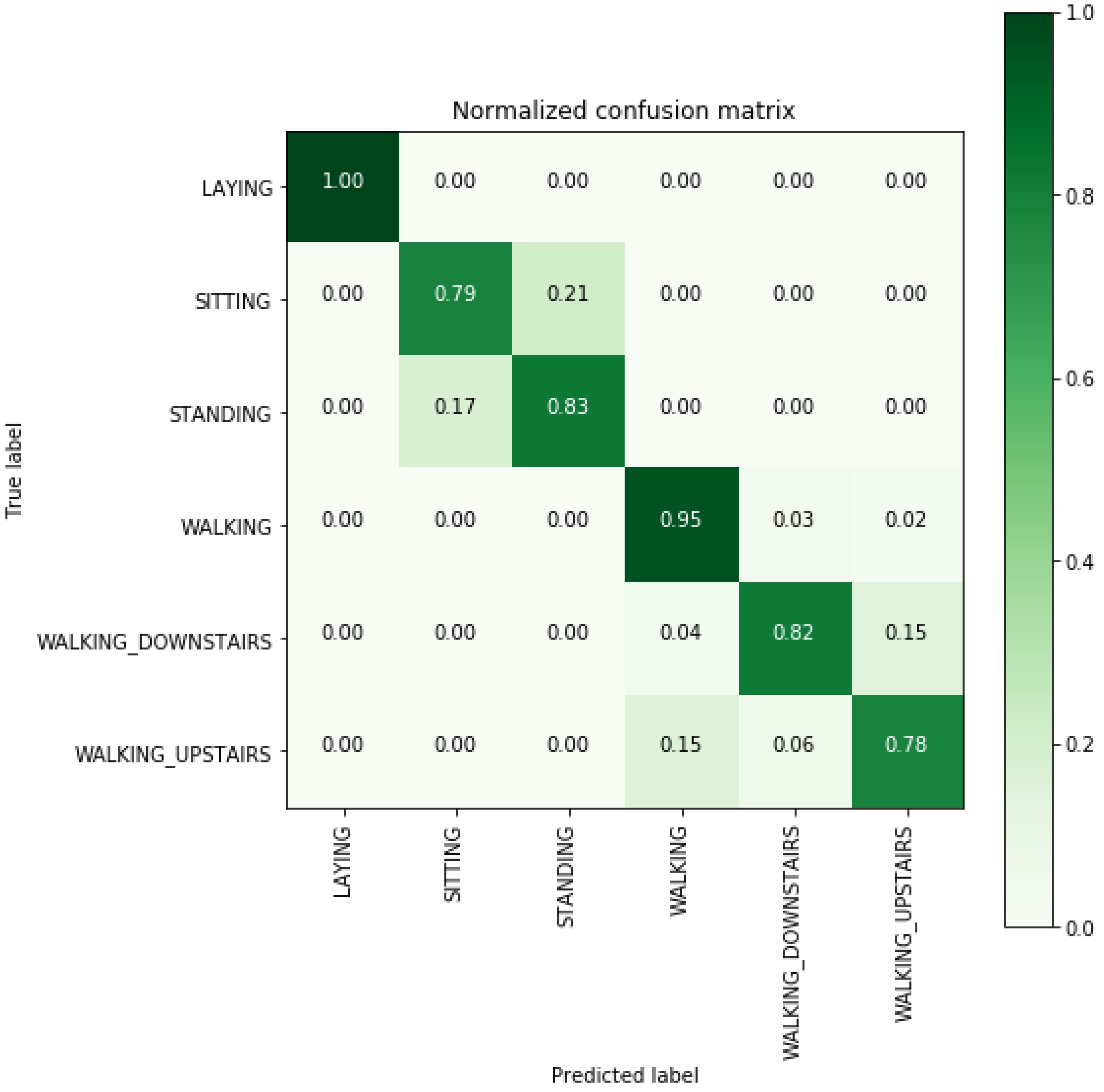
# 4. Decision Trees with GridSearchCV

In [0]:

```python
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_te
st, 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', m
ax_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
```

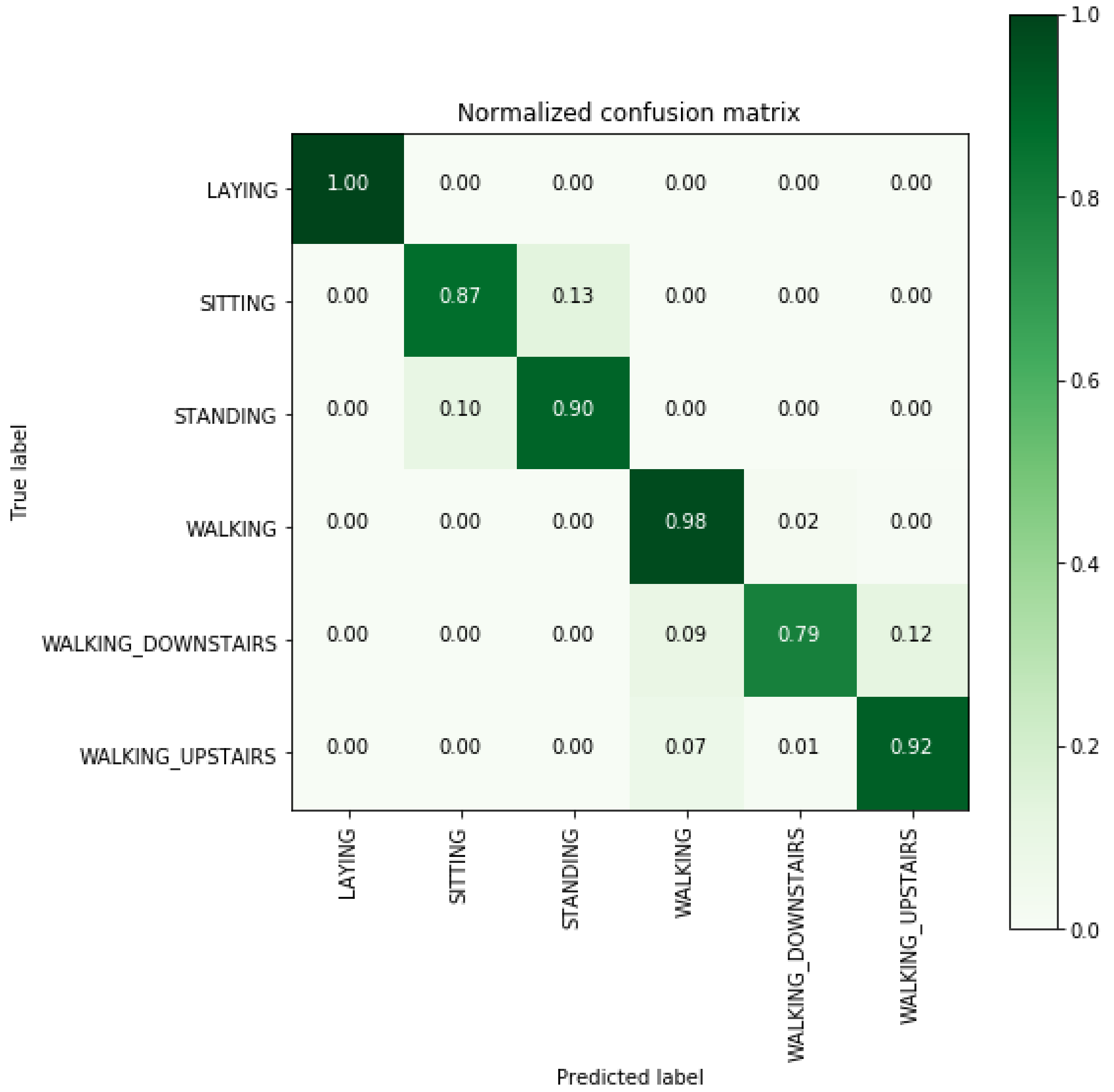
## 5. Random Forest Classifier with GridSearch

In [0]:

```
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
```

```
            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, crite
rion='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 :
```

```
Average Cross validate scores of best estimator :

0.9141730141458106
```

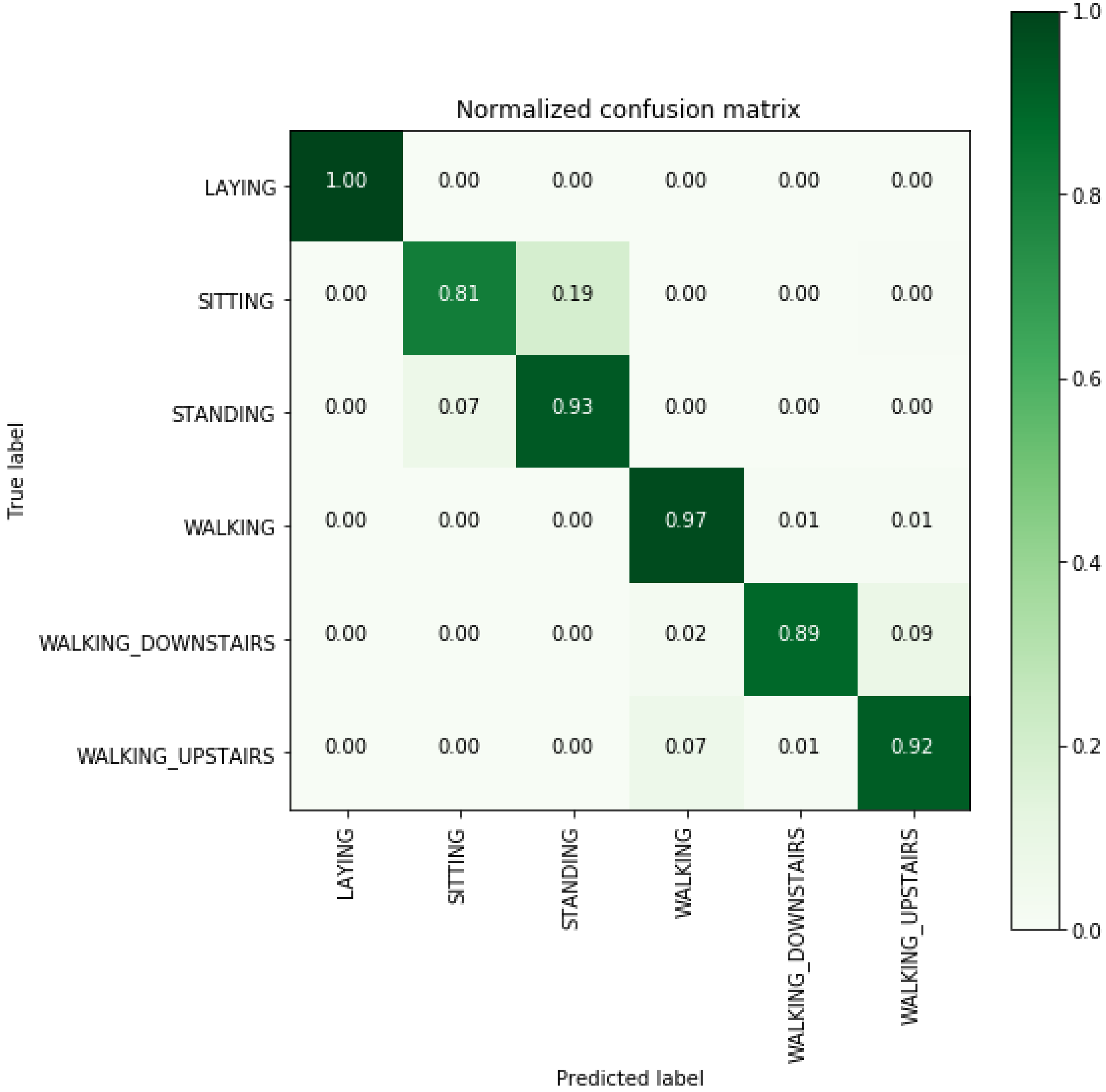
## 6. Gradient Boosted Decision Trees With GridSearch

In [0]:

```
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]]
```



|  | LAYING | SITTING | STANDING | WALKING | WALKING_DOWNSTAIRS | WALKING_UPSTAIRS |
|---|---|---|---|---|---|---|
| LAYING | 1.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| SITTING | 0.00 | 0.81 | 0.19 | 0.00 | 0.00 | 0.00 |
| STANDING | 0.00 | 0.07 | 0.93 | 0.00 | 0.00 | 0.00 |
| WALKING | 0.00 | 0.00 | 0.00 | 0.97 | 0.01 | 0.01 |
| WALKING_DOWNSTAIRS | 0.00 | 0.00 | 0.00 | 0.02 | 0.89 | 0.09 |
| WALKING_UPSTAIRS | 0.00 | 0.00 | 0.00 | 0.07 | 0.01 | 0.92 |

```
-------------------------
| 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
```

In [0]:
```
import pandas as pd
import numpy as np
```

In [0]:
```
# 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',
}

# Utility function to print the confusion matrix
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=['Pr
ed'])
```

## Data

In [0]:
```
# Data directory
DATADIR = 'UCI_HAR_Dataset'
```

In [0]:
```
# Raw data signals
```

```
# Signals are from Accelerometer and Gyroscope
# The signals are in x,y,z directions
# Sensor signals are filtered to have only body acceleration
# excluding the acceleration due to gravity
# Triaxial acceleration from the accelerometer is total acceleration
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 [0]:

```
# 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'UCI_HAR_Dataset/{subset}/Inertial Signals/{signal}_{subset}.txt'
        signals_data.append(
            _read_csv(filename).as_matrix()
        )

    # 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 [0]:

```
def load_y(subset):
    """
    The objective that we are trying to predict is a integer, from 1 to 6,
    that represents a human activity. We return a binary representation of
    every sample objective as a 6 bits vector using One Hot Encoding
    (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.html)
    """
    filename = f'UCI_HAR_Dataset/{subset}/y_{subset}.txt'
    y = _read_csv(filename)[0]

    return pd.get_dummies(y).as_matrix()
```

In [0]:

```
def load_data():
    """
    Obtain the dataset from multiple files.
    Returns: X_train, X_test, y_train, y_test
    """
    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 [0]:

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

In [0]:

```
# Importing tensorflow
np.random.seed(42)
import tensorflow as tf
tf.set_random_seed(42)
```

In [0]:

```
# Configuring a session
session_conf = tf.ConfigProto(
    intra_op_parallelism_threads=1,
    inter_op_parallelism_threads=1
)
```

In [0]:

```
# Import Keras
from keras import backend as K
sess = tf.Session(graph=tf.get_default_graph(), config=session_conf)
K.set_session(sess)
```

```
Using TensorFlow backend.
```

In [0]:

```
# Importing libraries
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers.core import Dense, Dropout
```

In [0]:

```
# Initializing parameters
epochs = 25
batch_size = 16
n_hidden =[30,50]
```

In [0]:

```
# Utility function to count the number of classes
def _count_classes(y):
    return len(set([tuple(category) for category in y]))
```

In [0]:

```
# Loading the train and test data
X_train, X_test, Y_train, Y_test = load_data()
```

In [0]:

```
# Install the PyDrive wrapper & import libraries.
# This only needs to be done once per notebook.
!pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

# Authenticate and create the PyDrive client.
# This only needs to be done once per notebook.
auth.authenticate_user()
gauth = GoogleAuth()
```

```
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)

# Download a file based on its file ID.
#
# A file ID looks like: laggVyWshwcyP6kEI-y_W3P8D26sz
listed = drive.ListFile().GetList()
for file in listed:
    print('title {}, id {}'.format(file['title'], file['id']))
```

In [0]:

```
download = drive.CreateFile({'id': '1YJG5KLOCG-AbqGd47ODcByPMejsz-lTR'
})
download.GetContentFile('X_TEST.pkl')
#https://drive.google.com/open?id=1YJG5KLOCG-AbqGd47ODcByPMejsz-lTR
```

In [0]:

```
download = drive.CreateFile({'id': '1Qq8eUm6jJDjpi6MSkGL_9rZny3El1yBr'
})
download.GetContentFile('X_TRAIN.pkl')
#https://drive.google.com/open?id=1Qq8eUm6jJDjpi6MSkGL_9rZny3El1yBr
```

In [0]:

```
download = drive.CreateFile({'id': '1drmq9jDPfw-EfeYfwUOgavSp4Uzeyhiu'
})
download.GetContentFile('Y_TEST.pkl')
#https://drive.google.com/open?id=1drmq9jDPfw-EfeYfwUOgavSp4Uzeyhiu
```

In [0]:

```
download = drive.CreateFile({'id': '1tVKv9b_SGPu-c7ScdZTtMrL8DrOz6Sez'
})
download.GetContentFile('Y_TRAIN.pkl')
#https://drive.google.com/open?id=1tVKv9b_SGPu-c7ScdZTtMrL8DrOz6Sez
```

In [0]:

```
import pickle

###Extract from file
with open("X_TRAIN.pkl","rb") as f:
    X_train = pickle.load(f)

###Extract from file
```

```
with open("X_TEST.pkl","rb") as f:
    X_test = pickle.load(f)


###Extract from file
with open("Y_TEST.pkl","rb") as f:
    Y_test = pickle.load(f)


###Extract from file
with open("Y_TRAIN.pkl","rb") as f:
    Y_train = pickle.load(f)
```

In [0]:

```
timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = 6

print(timesteps)
print(input_dim)
print(len(X_train))
```

```
128
9
7352
```

In [0]:

```
for i in n_hidden:
    print(i)
```

```
30
50
```

- Defining the Architecture of LSTM

# DEEP LEARNING

# N-HIDDEN=30,50

In [0]:

```
for i in n_hidden:#(30 and 50)
    # Initiliazing the sequential model
    model = Sequential()
    # Configuring the parameters
    model.add(LSTM(i, input_shape=(timesteps, input_dim)))
    # Adding a dropout layer
    model.add(Dropout(0.5))
    # Adding a dense output layer with sigmoid activation
    model.add(Dense(n_classes, activation='sigmoid'))
    model.summary()

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

    # Training the model
    model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)

    # Confusion Matrix
    print(confusion_matrix(Y_test, model.predict(X_test)))

    score = model.evaluate(X_test, Y_test)

    list=[]
    list.append(score)
    print(list)
```

```
WARNING: Logging before flag parsing goes to stderr.
W0903 06:18:22.929164 139936869062528 deprecation_wrapper.py:119] From
/usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backen
d.py:66: The name tf.get_default_graph is deprecated. Please use tf.com
pat.v1.get_default_graph instead.
```

```
W0903 06:18:22.940048 139936869062528 deprecation_wrapper.py:119] From
/usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backen
d.py:541: The name tf.placeholder is deprecated. Please use tf.compat.v
1.placeholder instead.

W0903 06:18:22.951153 139936869062528 deprecation_wrapper.py:119] From
/usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backen
d.py:4432: The name tf.random_uniform is deprecated. Please use tf.rand
om.uniform instead.

W0903 06:18:23.384014 139936869062528 deprecation_wrapper.py:119] From
/usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backen
d.py:148: The name tf.placeholder_with_default is deprecated. Please us
e tf.compat.v1.placeholder_with_default instead.

W0903 06:18:23.391802 139936869062528 deprecation.py:506] From /usr/loc
al/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:373
3: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob i
s deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate =
1 - keep_prob`.
W0903 06:18:23.421131 139936869062528 deprecation_wrapper.py:119] From
/usr/local/lib/python3.6/dist-packages/keras/optimizers.py:793: The nam
e tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optim
izer instead.

W0903 06:18:23.438022 139936869062528 deprecation_wrapper.py:119] From
/usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backen
d.py:3576: The name tf.log is deprecated. Please use tf.math.log instea
d.

W0903 06:18:23.554906 139936869062528 deprecation.py:323] From /usr/loc
al/lib/python3.6/dist-packages/tensorflow/python/ops/math_grad.py:1250:
add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array
_ops) is deprecated and will be removed in a future version.


Instructions for updating:
```

```
Use tf.where in 2.0, which has the same broadcast rule as np.where
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 30)                4800
_________________________________________________________________
dropout_1 (Dropout)          (None, 30)                0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 186
=================================================================
Total params: 4,986
Trainable params: 4,986
Non-trainable params: 0
_________________________________________________________________
Train on 7352 samples, validate on 2947 samples
Epoch 1/25
7352/7352 [==============================] - 93s 13ms/step - loss: 1.30
72 - acc: 0.4499 - val_loss: 1.0075 - val_acc: 0.5182
Epoch 2/25
7352/7352 [==============================] - 90s 12ms/step - loss: 0.92
83 - acc: 0.5471 - val_loss: 0.8425 - val_acc: 0.6067
Epoch 3/25
7352/7352 [==============================] - 90s 12ms/step - loss: 0.79
73 - acc: 0.6023 - val_loss: 0.7585 - val_acc: 0.6281
Epoch 4/25
7352/7352 [==============================] - 91s 12ms/step - loss: 0.74
05 - acc: 0.6748 - val_loss: 0.7687 - val_acc: 0.6783
Epoch 5/25
7352/7352 [==============================] - 90s 12ms/step - loss: 0.63
90 - acc: 0.7380 - val_loss: 0.5709 - val_acc: 0.7323
Epoch 6/25
7352/7352 [==============================] - 90s 12ms/step - loss: 0.54
27 - acc: 0.7726 - val_loss: 0.5818 - val_acc: 0.7435
Epoch 7/25
7352/7352 [==============================] - 91s 12ms/step - loss: 0.48
31 - acc: 0.8016 - val_loss: 0.5420 - val_acc: 0.7631
Epoch 8/25
7352/7352 [==============================] - 91s 12ms/step - loss: 0.43
```

```
96 - acc: 0.8464 - val_loss: 0.5268 - val_acc: 0.8314
Epoch 9/25
7352/7352 [==============================] - 91s 12ms/step - loss: 0.40
18 - acc: 0.8760 - val_loss: 0.5084 - val_acc: 0.8453
Epoch 10/25
7352/7352 [==============================] - 91s 12ms/step - loss: 0.33
74 - acc: 0.9067 - val_loss: 0.4317 - val_acc: 0.8622
Epoch 11/25
7352/7352 [==============================] - 91s 12ms/step - loss: 0.29
90 - acc: 0.9120 - val_loss: 0.3753 - val_acc: 0.8795
Epoch 12/25
7352/7352 [==============================] - 91s 12ms/step - loss: 0.27
79 - acc: 0.9170 - val_loss: 0.3683 - val_acc: 0.8826
Epoch 13/25
7352/7352 [==============================] - 91s 12ms/step - loss: 0.26
52 - acc: 0.9203 - val_loss: 0.3991 - val_acc: 0.8829
Epoch 14/25
7352/7352 [==============================] - 92s 12ms/step - loss: 0.24
42 - acc: 0.9249 - val_loss: 0.3788 - val_acc: 0.8921
Epoch 15/25
7352/7352 [==============================] - 91s 12ms/step - loss: 0.23
86 - acc: 0.9267 - val_loss: 0.5145 - val_acc: 0.8711
Epoch 16/25
7352/7352 [==============================] - 91s 12ms/step - loss: 0.22
55 - acc: 0.9298 - val_loss: 0.3603 - val_acc: 0.8924
Epoch 17/25
7352/7352 [==============================] - 91s 12ms/step - loss: 0.20
58 - acc: 0.9370 - val_loss: 0.3801 - val_acc: 0.8904
Epoch 18/25
7352/7352 [==============================] - 91s 12ms/step - loss: 0.20
29 - acc: 0.9336 - val_loss: 0.4204 - val_acc: 0.8901
Epoch 19/25
7352/7352 [==============================] - 90s 12ms/step - loss: 0.21
54 - acc: 0.9334 - val_loss: 0.5807 - val_acc: 0.8643
Epoch 20/25
7352/7352 [==============================] - 90s 12ms/step - loss: 0.19
18 - acc: 0.9388 - val_loss: 0.4082 - val_acc: 0.9016
Epoch 21/25

7352/7352 [==============================] - 91s 12ms/step - loss: 0.18
```

```
10 - acc: 0.9425 - val_loss: 0.4189 - val_acc: 0.8975
Epoch 22/25
7352/7352 [==============================] - 90s 12ms/step - loss: 0.20
48 - acc: 0.9421 - val_loss: 0.4531 - val_acc: 0.8982
Epoch 23/25
7352/7352 [==============================] - 90s 12ms/step - loss: 0.19
14 - acc: 0.9362 - val_loss: 0.3639 - val_acc: 0.9009
Epoch 24/25
7352/7352 [==============================] - 91s 12ms/step - loss: 0.16
85 - acc: 0.9437 - val_loss: 0.3534 - val_acc: 0.9016
Epoch 25/25
7352/7352 [==============================] - 90s 12ms/step - loss: 0.18
16 - acc: 0.9440 - val_loss: 0.3404 - val_acc: 0.9043

-----------------------------------------------------------------------
----
NameError                                 Traceback (most recent call l
ast)
<ipython-input-29-106868b458e9> in <module>()
     23 
     24     # Confusion Matrix
---> 25     print(confusion_matrix(Y_test, model.predict(X_test)))
     26 
     27     score = model.evaluate(X_test, Y_test)

NameError: name 'confusion_matrix' is not defined
```

## BATCH SIZE=20, H_HIDDEN=50

In [0]:

```
batch_size = [20]

# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(50, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
```

```
model.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])

for i in batch_size:
    # Training the model
    model.fit(X_train,
          Y_train,
          batch_size=i,
          validation_data=(X_test, Y_test),
          epochs=30)

    # Confusion Matrix
    print(confusion_matrix(Y_test, model.predict(X_test)))

    score = model.evaluate(X_test, Y_test)

    list=[]
    list.append(score)
    print(list)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_8 (LSTM)                (None, 50)                12000     
_________________________________________________________________
dropout_7 (Dropout)          (None, 50)                0         
_________________________________________________________________
dense_7 (Dense)              (None, 6)                 306       
=================================================================
Total params: 12,306
Trainable params: 12,306
Non-trainable params: 0
_________________________________________________________________
Train on 7352 samples, validate on 2947 samples
```

```
Epoch 1/30
7352/7352 [==============================] - 74s 10ms/step - loss: 1.34
31 - acc: 0.4056 - val_loss: 1.1828 - val_acc: 0.5049
Epoch 2/30
7352/7352 [==============================] - 72s 10ms/step - loss: 1.00
37 - acc: 0.5652 - val_loss: 0.9445 - val_acc: 0.6468
Epoch 3/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.79
32 - acc: 0.6574 - val_loss: 0.8565 - val_acc: 0.6804
Epoch 4/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.65
49 - acc: 0.7248 - val_loss: 0.7232 - val_acc: 0.7143
Epoch 5/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.59
74 - acc: 0.7520 - val_loss: 0.7075 - val_acc: 0.6926
Epoch 6/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.53
72 - acc: 0.7968 - val_loss: 0.6315 - val_acc: 0.7845
Epoch 7/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.49
87 - acc: 0.8502 - val_loss: 0.5195 - val_acc: 0.8300
Epoch 8/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.34
65 - acc: 0.8938 - val_loss: 0.4937 - val_acc: 0.8476
Epoch 9/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.30
41 - acc: 0.9063 - val_loss: 0.5530 - val_acc: 0.8354
Epoch 10/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.28
69 - acc: 0.9095 - val_loss: 0.6340 - val_acc: 0.8565
Epoch 11/30
7352/7352 [==============================] - 71s 10ms/step - loss: 0.24
49 - acc: 0.9196 - val_loss: 0.4410 - val_acc: 0.8629
Epoch 12/30
7352/7352 [==============================] - 71s 10ms/step - loss: 0.22
28 - acc: 0.9241 - val_loss: 0.9497 - val_acc: 0.7577
Epoch 13/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.21
95 - acc: 0.9263 - val_loss: 0.3575 - val_acc: 0.8839
```

```
Epoch 14/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.18
33 - acc: 0.9374 - val_loss: 0.3619 - val_acc: 0.9006
Epoch 15/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.19
32 - acc: 0.9363 - val_loss: 0.5092 - val_acc: 0.8907
Epoch 16/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.17
46 - acc: 0.9407 - val_loss: 0.4385 - val_acc: 0.8965
Epoch 17/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.16
80 - acc: 0.9412 - val_loss: 0.3242 - val_acc: 0.9030
Epoch 18/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.17
16 - acc: 0.9399 - val_loss: 0.4171 - val_acc: 0.9040
Epoch 19/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.17
39 - acc: 0.9415 - val_loss: 0.4158 - val_acc: 0.8955
Epoch 20/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.16
57 - acc: 0.9403 - val_loss: 1.0578 - val_acc: 0.8331
Epoch 21/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.13
81 - acc: 0.9461 - val_loss: 0.4731 - val_acc: 0.9030
Epoch 22/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.14
00 - acc: 0.9508 - val_loss: 0.4512 - val_acc: 0.8979
Epoch 23/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.14
77 - acc: 0.9455 - val_loss: 0.4751 - val_acc: 0.9009
Epoch 24/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.14
80 - acc: 0.9498 - val_loss: 0.5375 - val_acc: 0.8931
Epoch 25/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.16
46 - acc: 0.9448 - val_loss: 0.3878 - val_acc: 0.9125
Epoch 26/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.14
66 - acc: 0.9478 - val_loss: 0.4435 - val_acc: 0.9023
```

```
Epoch 27/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.13
26 - acc: 0.9512 - val_loss: 0.4770 - val_acc: 0.8975
Epoch 28/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.15
91 - acc: 0.9490 - val_loss: 0.3613 - val_acc: 0.9108
Epoch 29/30
7352/7352 [==============================] - 72s 10ms/step - loss: 0.15
19 - acc: 0.9426 - val_loss: 0.3667 - val_acc: 0.9118
Epoch 30/30
7352/7352 [==============================] - 73s 10ms/step - loss: 0.13
74 - acc: 0.9520 - val_loss: 0.5077 - val_acc: 0.9043
Pred                LAYING  SITTING  ...  WALKING_DOWNSTAIRS  WALKING_U
PSTAIRS
True                                 ...                                

LAYING                 535        0  ...                   0            
      2
SITTING                  5      398  ...                   0            
      3
STANDING                 0      106  ...                   0            
      0
WALKING                  0        0  ...                  33            
      1
WALKING_DOWNSTAIRS       0        0  ...                 419            
      0
WALKING_UPSTAIRS         2        0  ...                   6            
    429

[6 rows x 6 columns]
2947/2947 [==============================] - 6s 2ms/step
[[0.5076741776222898, 0.9043094672548354]]
```

## bias_initializer='zeros' AND N_HIDDEN=50

In [0]:

```
for i in n_hidden: #50
    # Initiliazing the sequential model
```

```
    model = Sequential()
    # Configuring the parameters
    model.add(LSTM(i, input_shape=(timesteps, input_dim)))
    # Adding a dropout layer
    model.add(Dropout(0.5))
    # Adding a dense output layer with sigmoid activation
    model.add(Dense(n_classes, activation='sigmoid',bias_initializer='z
eros'))
    model.summary()

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

    # Training the model
    model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)

    # Confusion Matrix
    print(confusion_matrix(Y_test, model.predict(X_test)))

    score = model.evaluate(X_test, Y_test)

    list=[]
    list.append(score)
    print(list)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_2 (LSTM)                (None, 30)                4800      
_________________________________________________________________
dropout_2 (Dropout)          (None, 30)                0         
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 186       
=================================================================
```

```
Total params: 4,986
Trainable params: 4,986
Non-trainable params: 0
_________________________________________________________________
WARNING:tensorflow:From D:\python\lib\site-packages\tensorflow\python\o
ps\math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 7352 samples, validate on 2947 samples
Epoch 1/25
7352/7352 [==============================] - 59s 8ms/step - loss: 1.409
8 - acc: 0.3658 - val_loss: 1.3462 - val_acc: 0.3848
Epoch 2/25
7352/7352 [==============================] - 53s 7ms/step - loss: 1.165
4 - acc: 0.4667 - val_loss: 1.1091 - val_acc: 0.4282
Epoch 3/25
7352/7352 [==============================] - 55s 7ms/step - loss: 1.036
2 - acc: 0.5080 - val_loss: 1.0004 - val_acc: 0.5114
Epoch 4/25
7352/7352 [==============================] - 54s 7ms/step - loss: 0.895
0 - acc: 0.5977 - val_loss: 0.9265 - val_acc: 0.5847
Epoch 5/25
7352/7352 [==============================] - 55s 7ms/step - loss: 0.833
8 - acc: 0.6235 - val_loss: 0.8451 - val_acc: 0.6060
Epoch 6/25
7352/7352 [==============================] - 52s 7ms/step - loss: 0.716
2 - acc: 0.6532 - val_loss: 0.8258 - val_acc: 0.5925
Epoch 7/25
7352/7352 [==============================] - 51s 7ms/step - loss: 0.666
8 - acc: 0.6736 - val_loss: 0.7662 - val_acc: 0.5969
Epoch 8/25
7352/7352 [==============================] - 50s 7ms/step - loss: 0.611
5 - acc: 0.7193 - val_loss: 0.6973 - val_acc: 0.6804
Epoch 9/25
7352/7352 [==============================] - 50s 7ms/step - loss: 0.587
3 - acc: 0.7844 - val_loss: 0.6832 - val_acc: 0.7886
Epoch 10/25
7352/7352 [==============================] - 51s 7ms/step - loss: 0.582
```

```
5 - acc: 0.8044 - val_loss: 0.7368 - val_acc: 0.7455
Epoch 11/25
7352/7352 [==============================] - 52s 7ms/step - loss: 0.513
3 - acc: 0.8414 - val_loss: 0.7064 - val_acc: 0.7852
Epoch 12/25
7352/7352 [==============================] - 50s 7ms/step - loss: 0.449
4 - acc: 0.8628 - val_loss: 0.7961 - val_acc: 0.7720
Epoch 13/25
7352/7352 [==============================] - 50s 7ms/step - loss: 0.414
7 - acc: 0.8717 - val_loss: 0.5192 - val_acc: 0.8402
Epoch 14/25
7352/7352 [==============================] - 50s 7ms/step - loss: 0.366
0 - acc: 0.8977 - val_loss: 0.4593 - val_acc: 0.8599
Epoch 15/25
7352/7352 [==============================] - 49s 7ms/step - loss: 0.341
6 - acc: 0.9074 - val_loss: 0.5251 - val_acc: 0.8497
Epoch 16/25
7352/7352 [==============================] - 50s 7ms/step - loss: 0.350
0 - acc: 0.8983 - val_loss: 0.4581 - val_acc: 0.8656
Epoch 17/25
7352/7352 [==============================] - 49s 7ms/step - loss: 0.285
3 - acc: 0.9180 - val_loss: 0.5698 - val_acc: 0.8351
Epoch 18/25
7352/7352 [==============================] - 49s 7ms/step - loss: 0.283
8 - acc: 0.9170 - val_loss: 0.6899 - val_acc: 0.8490
Epoch 19/25
7352/7352 [==============================] - 50s 7ms/step - loss: 0.283
3 - acc: 0.9151 - val_loss: 0.4440 - val_acc: 0.8731
Epoch 20/25
7352/7352 [==============================] - 50s 7ms/step - loss: 0.222
8 - acc: 0.9305 - val_loss: 0.3845 - val_acc: 0.8765
Epoch 21/25
7352/7352 [==============================] - 49s 7ms/step - loss: 0.237
0 - acc: 0.9358 - val_loss: 0.4509 - val_acc: 0.8551
Epoch 22/25
7352/7352 [==============================] - 49s 7ms/step - loss: 0.321
6 - acc: 0.9086 - val_loss: 0.4348 - val_acc: 0.8775
Epoch 23/25
7352/7352 [==============================] - 49s 7ms/step - loss: 0.324
```

```
9 - acc: 0.9123 - val_loss: 0.5224 - val_acc: 0.8405
Epoch 24/25
7352/7352 [==============================] - 49s 7ms/step - loss: 0.253
0 - acc: 0.9274 - val_loss: 0.4417 - val_acc: 0.8863
Epoch 25/25
7352/7352 [==============================] - 49s 7ms/step - loss: 0.281
4 - acc: 0.9170 - val_loss: 0.4307 - val_acc: 0.8785
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTA
IRS  \
True

LAYING                 510        0         0        0
  0
SITTING                  0      395        92        0
  0
STANDING                 0      109       419        4
  0
WALKING                  0        2         2      463
13
WALKING_DOWNSTAIRS       0        3         0        9
399
WALKING_UPSTAIRS         0        3         4       59
  2

Pred                WALKING_UPSTAIRS
True
LAYING                            27
SITTING                            4
STANDING                           0
WALKING                           16
WALKING_DOWNSTAIRS                 9
WALKING_UPSTAIRS                 403
2947/2947 [==============================] - 2s 753us/step
[[0.4306884481576987, 0.8785205293518833]]
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 50)                12000

_________________________________________________________________
```

```
dropout_3 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 306
=================================================================
Total params: 12,306
Trainable params: 12,306
Non-trainable params: 0
_________________________________________________________________
Train on 7352 samples, validate on 2947 samples
Epoch 1/25
7352/7352 [==============================] - 60s 8ms/step - loss: 1.327
4 - acc: 0.4211 - val_loss: 1.6261 - val_acc: 0.3756
Epoch 2/25
7352/7352 [==============================] - 70s 9ms/step - loss: 0.986
7 - acc: 0.5717 - val_loss: 0.9573 - val_acc: 0.5650
Epoch 3/25
7352/7352 [==============================] - 59s 8ms/step - loss: 0.751
0 - acc: 0.6778 - val_loss: 0.7629 - val_acc: 0.6709
Epoch 4/25
7352/7352 [==============================] - 59s 8ms/step - loss: 0.627
3 - acc: 0.7360 - val_loss: 0.6046 - val_acc: 0.7648
Epoch 5/25
7352/7352 [==============================] - 61s 8ms/step - loss: 0.527
2 - acc: 0.7750 - val_loss: 0.5351 - val_acc: 0.7520
Epoch 6/25
7352/7352 [==============================] - 59s 8ms/step - loss: 0.476
5 - acc: 0.7982 - val_loss: 0.5932 - val_acc: 0.7689
Epoch 7/25
7352/7352 [==============================] - 58s 8ms/step - loss: 0.453
5 - acc: 0.8288 - val_loss: 0.4325 - val_acc: 0.8483
Epoch 8/25
7352/7352 [==============================] - 59s 8ms/step - loss: 0.367
4 - acc: 0.8868 - val_loss: 0.3828 - val_acc: 0.8717
Epoch 9/25
7352/7352 [==============================] - 58s 8ms/step - loss: 0.270
8 - acc: 0.9104 - val_loss: 0.4445 - val_acc: 0.8768
Epoch 10/25
7352/7352 [==============================] - 58s 8ms/step - loss: 0.244
7 - acc: 0.9206 - val_loss: 0.3423 - val_acc: 0.8873
```

```
Epoch 11/25
7352/7352 [==============================] - 58s 8ms/step - loss: 0.222
6 - acc: 0.9272 - val_loss: 0.2954 - val_acc: 0.9060
Epoch 12/25
7352/7352 [==============================] - 58s 8ms/step - loss: 0.192
0 - acc: 0.9317 - val_loss: 0.3025 - val_acc: 0.8945
Epoch 13/25
7352/7352 [==============================] - 58s 8ms/step - loss: 0.211
0 - acc: 0.9332 - val_loss: 0.3361 - val_acc: 0.8795
Epoch 14/25
7352/7352 [==============================] - 58s 8ms/step - loss: 0.171
0 - acc: 0.9393 - val_loss: 0.2092 - val_acc: 0.9175
Epoch 15/25
7352/7352 [==============================] - 58s 8ms/step - loss: 0.182
3 - acc: 0.9362 - val_loss: 0.2753 - val_acc: 0.9043
Epoch 16/25
7352/7352 [==============================] - 56s 8ms/step - loss: 0.164
7 - acc: 0.9414 - val_loss: 0.3146 - val_acc: 0.8996
Epoch 17/25
7352/7352 [==============================] - 56s 8ms/step - loss: 0.153
2 - acc: 0.9444 - val_loss: 0.2999 - val_acc: 0.9074
Epoch 18/25
7352/7352 [==============================] - 56s 8ms/step - loss: 0.168
7 - acc: 0.9440 - val_loss: 0.3796 - val_acc: 0.9141
Epoch 19/25
7352/7352 [==============================] - 55s 7ms/step - loss: 0.155
2 - acc: 0.9460 - val_loss: 0.5702 - val_acc: 0.8941
Epoch 20/25
7352/7352 [==============================] - 55s 8ms/step - loss: 0.153
2 - acc: 0.9472 - val_loss: 0.3448 - val_acc: 0.9158
Epoch 21/25
7352/7352 [==============================] - 56s 8ms/step - loss: 0.163
6 - acc: 0.9438 - val_loss: 0.3222 - val_acc: 0.9138
Epoch 22/25
7352/7352 [==============================] - 55s 7ms/step - loss: 0.160
2 - acc: 0.9448 - val_loss: 0.2972 - val_acc: 0.9148
Epoch 23/25
7352/7352 [==============================] - 55s 8ms/step - loss: 0.153
1 - acc: 0.9450 - val_loss: 0.4835 - val_acc: 0.8948
```

```
Epoch 24/25
7352/7352 [==============================] - 55s 8ms/step - loss: 0.139
3 - acc: 0.9494 - val_loss: 0.3466 - val_acc: 0.9226
Epoch 25/25
7352/7352 [==============================] - 57s 8ms/step - loss: 0.162
9 - acc: 0.9445 - val_loss: 0.2939 - val_acc: 0.9209
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTA
IRS  \
True

LAYING                 537        0         0        0
  0
SITTING                  6      403        81        0
  0
STANDING                 0       92       440        0
  0
WALKING                  0        0         0      470
  4
WALKING_DOWNSTAIRS       0        0         0        4
403
WALKING_UPSTAIRS         0        0         1        8
  1

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            1
STANDING                           0
WALKING                           22
WALKING_DOWNSTAIRS                13
WALKING_UPSTAIRS                 461
2947/2947 [==============================] - 3s 861us/step
[[0.2939286813193747, 0.9209365456396336]]
```

## N_HIDDEN=50 AND glorot_normal_INITIALIZER

In [0]:

```
for i in n_hidden: #50
    # Initiliazing the sequential model
    model = Sequential()
    # Configuring the parameters
    model.add(LSTM(i, input_shape=(timesteps, input_dim)))
    # Adding a dropout layer
    model.add(Dropout(0.5))
    # Adding a dense output layer with sigmoid activation
    model.add(Dense(n_classes, activation='sigmoid',kernel_initializer=
tf.keras.initializers.glorot_normal(seed=None)))
    model.summary()

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

    # Training the model
    model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)

    # Confusion Matrix
    print(confusion_matrix(Y_test, model.predict(X_test)))

    score = model.evaluate(X_test, Y_test)

    list=[]
    list.append(score)
    print(list)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_7 (LSTM)                (None, 50)                12000     
_________________________________________________________________
dropout_7 (Dropout)          (None, 50)                0         
```

```
_________________________________________________________________
dense_4 (Dense)              (None, 6)                 306
=================================================================
Total params: 12,306
Trainable params: 12,306
Non-trainable params: 0

_________________________________________________________________
WARNING:tensorflow:From D:\python\lib\site-packages\tensorflow\python\o
ps\math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 7352 samples, validate on 2947 samples
Epoch 1/25
7352/7352 [==============================] - 78s 11ms/step - loss: 1.20
63 - acc: 0.4955 - val_loss: 1.0278 - val_acc: 0.5816
Epoch 2/25
7352/7352 [==============================] - 56s 8ms/step - loss: 0.834
7 - acc: 0.6310 - val_loss: 0.8318 - val_acc: 0.6081
Epoch 3/25
7352/7352 [==============================] - 57s 8ms/step - loss: 0.744
3 - acc: 0.6522 - val_loss: 0.7702 - val_acc: 0.6288
Epoch 4/25
7352/7352 [==============================] - 58s 8ms/step - loss: 0.668
8 - acc: 0.6997 - val_loss: 0.7439 - val_acc: 0.7167
Epoch 5/25
7352/7352 [==============================] - 58s 8ms/step - loss: 0.574
7 - acc: 0.7598 - val_loss: 0.5803 - val_acc: 0.7676
Epoch 6/25
7352/7352 [==============================] - 57s 8ms/step - loss: 0.492
7 - acc: 0.7964 - val_loss: 0.6355 - val_acc: 0.7621
Epoch 7/25
7352/7352 [==============================] - 57s 8ms/step - loss: 0.423
2 - acc: 0.8377 - val_loss: 0.5341 - val_acc: 0.8117
Epoch 8/25
7352/7352 [==============================] - 60s 8ms/step - loss: 0.356
6 - acc: 0.8868 - val_loss: 0.4469 - val_acc: 0.8588
Epoch 9/25
7352/7352 [==============================] - 56s 8ms/step - loss: 0.276
```

```
6 - acc: 0.9166 - val_loss: 0.6427 - val_acc: 0.8432
Epoch 10/25
7352/7352 [==============================] - 56s 8ms/step - loss: 0.243
1 - acc: 0.9230 - val_loss: 0.4285 - val_acc: 0.8585
Epoch 11/25
7352/7352 [==============================] - 57s 8ms/step - loss: 0.238
3 - acc: 0.9259 - val_loss: 0.4382 - val_acc: 0.8731
Epoch 12/25
7352/7352 [==============================] - 57s 8ms/step - loss: 0.214
1 - acc: 0.9347 - val_loss: 0.3263 - val_acc: 0.8924
Epoch 13/25
7352/7352 [==============================] - 56s 8ms/step - loss: 0.212
6 - acc: 0.9308 - val_loss: 0.4035 - val_acc: 0.8901
Epoch 14/25
7352/7352 [==============================] - 56s 8ms/step - loss: 0.223
2 - acc: 0.9336 - val_loss: 0.3298 - val_acc: 0.8938
Epoch 15/25
7352/7352 [==============================] - 57s 8ms/step - loss: 0.215
9 - acc: 0.9323 - val_loss: 0.4234 - val_acc: 0.8850
Epoch 16/25
7352/7352 [==============================] - 60s 8ms/step - loss: 0.197
4 - acc: 0.9343 - val_loss: 0.3042 - val_acc: 0.8975
Epoch 17/25
7352/7352 [==============================] - 58s 8ms/step - loss: 0.169
1 - acc: 0.9397 - val_loss: 0.3703 - val_acc: 0.8795
Epoch 18/25
7352/7352 [==============================] - 58s 8ms/step - loss: 0.181
2 - acc: 0.9395 - val_loss: 0.3291 - val_acc: 0.8911
Epoch 19/25
7352/7352 [==============================] - 58s 8ms/step - loss: 0.164
4 - acc: 0.9445 - val_loss: 0.3480 - val_acc: 0.8887
Epoch 20/25
7352/7352 [==============================] - 57s 8ms/step - loss: 0.154
0 - acc: 0.9427 - val_loss: 0.3574 - val_acc: 0.9135
Epoch 21/25
7352/7352 [==============================] - 58s 8ms/step - loss: 0.153
7 - acc: 0.9455 - val_loss: 0.4280 - val_acc: 0.9033
Epoch 22/25
7352/7352 [==============================] - 57s 8ms/step - loss: 0.162
```

```
3 - acc: 0.9453 - val_loss: 0.3307 - val_acc: 0.9026
Epoch 23/25
7352/7352 [==============================] - 58s 8ms/step - loss: 0.151
0 - acc: 0.9460 - val_loss: 0.4289 - val_acc: 0.8853
Epoch 24/25
7352/7352 [==============================] - 57s 8ms/step - loss: 0.145
3 - acc: 0.9478 - val_loss: 0.4368 - val_acc: 0.8975
Epoch 25/25
7352/7352 [==============================] - 58s 8ms/step - loss: 0.154
2 - acc: 0.9470 - val_loss: 0.4257 - val_acc: 0.9074
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTA
IRS  \
True

LAYING                 510        0        27        0
  0
SITTING                  0      368       120        1
  0
STANDING                 0       57       473        2
  0
WALKING                  0        0         0      453
32
WALKING_DOWNSTAIRS       0        0         0        4
414
WALKING_UPSTAIRS         0        0         0       11
  4

Pred                WALKING_UPSTAIRS
True
LAYING                             0
SITTING                            2
STANDING                           0
WALKING                           11
WALKING_DOWNSTAIRS                 2
WALKING_UPSTAIRS                 456
2947/2947 [==============================] - 3s 1ms/step
[[0.4256824295249125, 0.9073634204275535]]
```

# LSTM WITH 2-LAYERS

In [0]:

```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(50, return_sequences=True,input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(.5))
model.add(LSTM(50))
model.add(Dropout(.5))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()

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

# Training the model
model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)

# Confusion Matrix
print(confusion_matrix(Y_test, model.predict(X_test)))

score = model.evaluate(X_test, Y_test)

list=[]
list.append(score)
print(list)
```

```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_16 (LSTM)               (None, 128, 50)           12000
_________________________________________________________________
dropout_9 (Dropout)          (None, 128, 50)           0
_________________________________________________________________
lstm_17 (LSTM)               (None, 50)                20200
_________________________________________________________________
dropout_10 (Dropout)         (None, 50)                0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 306
=================================================================
Total params: 32,506
Trainable params: 32,506
Non-trainable params: 0
_________________________________________________________________
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_16 (LSTM)               (None, 128, 50)           12000
_________________________________________________________________
dropout_9 (Dropout)          (None, 128, 50)           0
_________________________________________________________________
lstm_17 (LSTM)               (None, 50)                20200
_________________________________________________________________
dropout_10 (Dropout)         (None, 50)                0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 306
=================================================================
Total params: 32,506
Trainable params: 32,506
Non-trainable params: 0
_________________________________________________________________
Train on 7352 samples, validate on 2947 samples
Epoch 1/25
Train on 7352 samples, validate on 2947 samples
Epoch 1/25
7352/7352 [==============================] - 236s 32ms/step - loss: 1.1
162 - acc: 0.5326 - val_loss: 0.9879 - val_acc: 0.6023
```

```
7352/7352 [==============================] - 236s 32ms/step - loss: 1.1
162 - acc: 0.5326 - val_loss: 0.9879 - val_acc: 0.6023
Epoch 2/25
Epoch 2/25
7352/7352 [==============================] - 147s 20ms/step - loss: 0.8
107 - acc: 0.6485 - val_loss: 0.8378 - val_acc: 0.6040
7352/7352 [==============================] - 147s 20ms/step - loss: 0.8
107 - acc: 0.6485 - val_loss: 0.8378 - val_acc: 0.6040
Epoch 3/25
Epoch 3/25
7352/7352 [==============================] - 149s 20ms/step - loss: 0.6
691 - acc: 0.6957 - val_loss: 0.6748 - val_acc: 0.7441
7352/7352 [==============================] - 149s 20ms/step - loss: 0.6
691 - acc: 0.6957 - val_loss: 0.6748 - val_acc: 0.7441
Epoch 4/25
Epoch 4/25
7352/7352 [==============================] - 160s 22ms/step - loss: 0.5
420 - acc: 0.7731 - val_loss: 0.6125 - val_acc: 0.7981
7352/7352 [==============================] - 160s 22ms/step - loss: 0.5
420 - acc: 0.7731 - val_loss: 0.6125 - val_acc: 0.7981
Epoch 5/25
Epoch 5/25
7352/7352 [==============================] - 146s 20ms/step - loss: 0.3
740 - acc: 0.8758 - val_loss: 0.6913 - val_acc: 0.7981
7352/7352 [==============================] - 146s 20ms/step - loss: 0.3
740 - acc: 0.8758 - val_loss: 0.6913 - val_acc: 0.7981
Epoch 6/25
Epoch 6/25
7352/7352 [==============================] - 152s 21ms/step - loss: 0.2
496 - acc: 0.9240 - val_loss: 0.6027 - val_acc: 0.8643
7352/7352 [==============================] - 152s 21ms/step - loss: 0.2
496 - acc: 0.9240 - val_loss: 0.6027 - val_acc: 0.8643
Epoch 7/25
Epoch 7/25
7352/7352 [==============================] - 154s 21ms/step - loss: 0.2
250 - acc: 0.9310 - val_loss: 0.3931 - val_acc: 0.8965
7352/7352 [==============================] - 154s 21ms/step - loss: 0.2
250 - acc: 0.9310 - val_loss: 0.3931 - val_acc: 0.8965
Epoch 8/25
```

```
Epoch 8/25
7352/7352 [==============================] - 152s 21ms/step - loss: 0.1
856 - acc: 0.9376 - val_loss: 0.3672 - val_acc: 0.8904
7352/7352 [==============================] - 152s 21ms/step - loss: 0.1
856 - acc: 0.9376 - val_loss: 0.3672 - val_acc: 0.8904
Epoch 9/25
Epoch 9/25
7352/7352 [==============================] - 144s 20ms/step - loss: 0.1
922 - acc: 0.9353 - val_loss: 0.4644 - val_acc: 0.8775
7352/7352 [==============================] - 144s 20ms/step - loss: 0.1
922 - acc: 0.9353 - val_loss: 0.4644 - val_acc: 0.8775
Epoch 10/25
Epoch 10/25
7352/7352 [==============================] - 141s 19ms/step - loss: 0.1
834 - acc: 0.9353 - val_loss: 0.4322 - val_acc: 0.8809
7352/7352 [==============================] - 141s 19ms/step - loss: 0.1
834 - acc: 0.9353 - val_loss: 0.4322 - val_acc: 0.8809
Epoch 11/25
Epoch 11/25
7352/7352 [==============================] - 143s 20ms/step - loss: 0.1
639 - acc: 0.9421 - val_loss: 0.3816 - val_acc: 0.8955
7352/7352 [==============================] - 143s 20ms/step - loss: 0.1
639 - acc: 0.9421 - val_loss: 0.3816 - val_acc: 0.8955
Epoch 12/25
Epoch 12/25
7352/7352 [==============================] - 146s 20ms/step - loss: 0.1
558 - acc: 0.9501 - val_loss: 0.3931 - val_acc: 0.9002
7352/7352 [==============================] - 146s 20ms/step - loss: 0.1
558 - acc: 0.9501 - val_loss: 0.3931 - val_acc: 0.9002
Epoch 13/25
Epoch 13/25
7352/7352 [==============================] - 148s 20ms/step - loss: 0.1
516 - acc: 0.9497 - val_loss: 0.3951 - val_acc: 0.9009
7352/7352 [==============================] - 148s 20ms/step - loss: 0.1
516 - acc: 0.9497 - val_loss: 0.3951 - val_acc: 0.9009
Epoch 14/25
Epoch 14/25
7352/7352 [==============================] - 152s 21ms/step - loss: 0.1
467 - acc: 0.9479 - val_loss: 0.5332 - val_acc: 0.8816
```

```
7352/7352 [==============================] - 152s 21ms/step - loss: 0.1
467 - acc: 0.9479 - val_loss: 0.5332 - val_acc: 0.8816
Epoch 15/25
Epoch 15/25
7352/7352 [==============================] - 149s 20ms/step - loss: 0.1
440 - acc: 0.9490 - val_loss: 0.4251 - val_acc: 0.8955
7352/7352 [==============================] - 149s 20ms/step - loss: 0.1
440 - acc: 0.9490 - val_loss: 0.4251 - val_acc: 0.8955
Epoch 16/25
Epoch 16/25
7352/7352 [==============================] - 158s 21ms/step - loss: 0.1
683 - acc: 0.9456 - val_loss: 0.4568 - val_acc: 0.9016
7352/7352 [==============================] - 158s 21ms/step - loss: 0.1
683 - acc: 0.9456 - val_loss: 0.4568 - val_acc: 0.9016
Epoch 17/25
Epoch 17/25
7352/7352 [==============================] - 142s 19ms/step - loss: 0.1
586 - acc: 0.9467 - val_loss: 0.3375 - val_acc: 0.8948
7352/7352 [==============================] - 142s 19ms/step - loss: 0.1
586 - acc: 0.9467 - val_loss: 0.3375 - val_acc: 0.8948
Epoch 18/25
Epoch 18/25
7352/7352 [==============================] - 147s 20ms/step - loss: 0.1
600 - acc: 0.9506 - val_loss: 0.6291 - val_acc: 0.8901
7352/7352 [==============================] - 147s 20ms/step - loss: 0.1
600 - acc: 0.9506 - val_loss: 0.6291 - val_acc: 0.8901
Epoch 19/25
Epoch 19/25
7352/7352 [==============================] - 157s 21ms/step - loss: 0.1
359 - acc: 0.9509 - val_loss: 0.4796 - val_acc: 0.9080
7352/7352 [==============================] - 157s 21ms/step - loss: 0.1
359 - acc: 0.9509 - val_loss: 0.4796 - val_acc: 0.9080
Epoch 20/25
Epoch 20/25
7352/7352 [==============================] - 166s 23ms/step - loss: 0.1
481 - acc: 0.9475 - val_loss: 0.3207 - val_acc: 0.9162
7352/7352 [==============================] - 166s 23ms/step - loss: 0.1
481 - acc: 0.9475 - val_loss: 0.3207 - val_acc: 0.9162
Epoch 21/25
```

```
Epoch 21/25
7352/7352 [==============================] - 155s 21ms/step - loss: 0.1
410 - acc: 0.9499 - val_loss: 0.3656 - val_acc: 0.9070
7352/7352 [==============================] - 155s 21ms/step - loss: 0.1
410 - acc: 0.9499 - val_loss: 0.3656 - val_acc: 0.9070
Epoch 22/25
Epoch 22/25
7352/7352 [==============================] - 137s 19ms/step - loss: 0.1
356 - acc: 0.9504 - val_loss: 0.4651 - val_acc: 0.9033
7352/7352 [==============================] - 137s 19ms/step - loss: 0.1
356 - acc: 0.9504 - val_loss: 0.4651 - val_acc: 0.9033
Epoch 23/25
Epoch 23/25
7352/7352 [==============================] - 153s 21ms/step - loss: 0.1
428 - acc: 0.9494 - val_loss: 0.4361 - val_acc: 0.8962
7352/7352 [==============================] - 153s 21ms/step - loss: 0.1
428 - acc: 0.9494 - val_loss: 0.4361 - val_acc: 0.8962
Epoch 24/25
Epoch 24/25
7352/7352 [==============================] - 155s 21ms/step - loss: 0.1
389 - acc: 0.9512 - val_loss: 0.3655 - val_acc: 0.9118
7352/7352 [==============================] - 155s 21ms/step - loss: 0.1
389 - acc: 0.9512 - val_loss: 0.3655 - val_acc: 0.9118
Epoch 25/25
Epoch 25/25
7352/7352 [==============================] - 168s 23ms/step - loss: 0.1
382 - acc: 0.9513 - val_loss: 0.3222 - val_acc: 0.9141
7352/7352 [==============================] - 168s 23ms/step - loss: 0.1
382 - acc: 0.9513 - val_loss: 0.3222 - val_acc: 0.9141
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTA
IRS  \
True

LAYING                 509        0        27        0
  1
SITTING                  1      380       108        0
  1
STANDING                 0       62       470        0
  0
```

```
WALKING                    0        1         1      484
  5
WALKING_DOWNSTAIRS         0        0         0        3
407
WALKING_UPSTAIRS           0        1         2       10
14

Pred                  WALKING_UPSTAIRS
True
LAYING                               0
SITTING                              1
STANDING                             0
WALKING                              5
WALKING_DOWNSTAIRS                  10
WALKING_UPSTAIRS                   444
  32/2947 [..............................] - ETA: 8sPred
LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS  \
True

LAYING                   509        0        27        0
  1
SITTING                    1      380       108        0
  1
STANDING                   0       62       470        0
  0
WALKING                    0        1         1      484
  5
WALKING_DOWNSTAIRS         0        0         0        3
407
WALKING_UPSTAIRS           0        1         2       10
14

Pred                  WALKING_UPSTAIRS
True
LAYING                               0
SITTING                              1
STANDING                             0
WALKING                              5
WALKING_DOWNSTAIRS                  10
```

```
WALKING_UPSTAIRS                  444
2947/2947 [==============================] - 12s 4ms/step
2947/2947 [==============================] - 12s 4ms/step
[[0.32220571241385143, 0.9141499830335935]]
[[0.32220571241385143, 0.9141499830335935]]
```

# CNN

In [0]:

```
# Initiliazing the sequential model
model = Sequential()
model.add(Conv1D(filters=300, kernel_size=15, padding='same', activation='sigmoid', input_shape=(timesteps, input_dim)))
model.add(MaxPooling1D(pool_size=15))
#model.add(Conv1D(filters=70, kernel_size=15, padding='same', activation='sigmoid', input_shape=(timesteps, input_dim)))

#model.add(MaxPooling1D(pool_size=9))
#model.add(Dropout(0.7))
#Configuring the parameters
#model.add(LSTM(50, input_shape=(timesteps, input_dim)))
#model.add(Dropout(0.6))
# Adding a dropout layer
# Adding a dense output layer with sigmoid activation
model.add(Flatten())
model.add(Dense(n_classes, activation='sigmoid'))

model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv1d_22 (Conv1D)           (None, 128, 300)          40800     
_________________________________________________________________
max_pooling1d_21 (MaxPooling (None, 8, 300)            0         
_________________________________________________________________
flatten_21 (Flatten)         (None, 2400)              0         
_________________________________________________________________
```

```
dense_21 (Dense)             (None, 6)                 14406     
=================================================================
Total params: 55,206
Trainable params: 55,206
Non-trainable params: 0
_________________________________________________________________
```

```
In [0]: # Compiling the model
        model.compile(loss='categorical_crossentropy',
                      optimizer='adamax',
                      metrics=['accuracy'])
```

```
In [0]: # Training the model
        model.fit(X_train,
                  Y_train,
                  batch_size=10,
                  validation_data=(X_test, Y_test),
                  epochs=50)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/50
7352/7352 [==============================] - 4s 542us/step - loss: 1.22
71 - acc: 0.4486 - val_loss: 0.5731 - val_acc: 0.8517
Epoch 2/50
7352/7352 [==============================] - 3s 390us/step - loss: 0.25
75 - acc: 0.9233 - val_loss: 0.3177 - val_acc: 0.9006
Epoch 3/50
7352/7352 [==============================] - 3s 387us/step - loss: 0.16
82 - acc: 0.9421 - val_loss: 0.2805 - val_acc: 0.9067
Epoch 4/50
7352/7352 [==============================] - 3s 388us/step - loss: 0.14
23 - acc: 0.9449 - val_loss: 0.2365 - val_acc: 0.9108
Epoch 5/50
7352/7352 [==============================] - 3s 390us/step - loss: 0.13
57 - acc: 0.9465 - val_loss: 0.2707 - val_acc: 0.8979
Epoch 6/50
7352/7352 [==============================] - 3s 388us/step - loss: 0.12
68 - acc: 0.9494 - val_loss: 0.2305 - val_acc: 0.9257
Epoch 7/50
```

```
7352/7352 [==============================] - 3s 386us/step - loss: 0.12
26 - acc: 0.9484 - val_loss: 0.2421 - val_acc: 0.9125
Epoch 8/50
7352/7352 [==============================] - 3s 392us/step - loss: 0.12
13 - acc: 0.9467 - val_loss: 0.2568 - val_acc: 0.9175
Epoch 9/50
7352/7352 [==============================] - 3s 393us/step - loss: 0.11
66 - acc: 0.9506 - val_loss: 0.2266 - val_acc: 0.9206
Epoch 10/50
7352/7352 [==============================] - 3s 390us/step - loss: 0.11
50 - acc: 0.9494 - val_loss: 0.2331 - val_acc: 0.9203
Epoch 11/50
7352/7352 [==============================] - 3s 390us/step - loss: 0.11
41 - acc: 0.9478 - val_loss: 0.2182 - val_acc: 0.9247
Epoch 12/50
7352/7352 [==============================] - 3s 384us/step - loss: 0.11
15 - acc: 0.9490 - val_loss: 0.2190 - val_acc: 0.9216
Epoch 13/50
7352/7352 [==============================] - 3s 390us/step - loss: 0.11
19 - acc: 0.9490 - val_loss: 0.2322 - val_acc: 0.9199
Epoch 14/50
7352/7352 [==============================] - 3s 408us/step - loss: 0.10
92 - acc: 0.9498 - val_loss: 0.2107 - val_acc: 0.9294
Epoch 15/50
7352/7352 [==============================] - 3s 409us/step - loss: 0.10
82 - acc: 0.9505 - val_loss: 0.2201 - val_acc: 0.9199
Epoch 16/50
7352/7352 [==============================] - 3s 411us/step - loss: 0.10
91 - acc: 0.9478 - val_loss: 0.2086 - val_acc: 0.9226
Epoch 17/50
7352/7352 [==============================] - 3s 403us/step - loss: 0.10
57 - acc: 0.9484 - val_loss: 0.2032 - val_acc: 0.9301
Epoch 18/50
7352/7352 [==============================] - 3s 388us/step - loss: 0.10
51 - acc: 0.9529 - val_loss: 0.2072 - val_acc: 0.9203
Epoch 19/50
7352/7352 [==============================] - 3s 385us/step - loss: 0.10
08 - acc: 0.9531 - val_loss: 0.2413 - val_acc: 0.9281
Epoch 20/50
```

```
7352/7352 [==============================] - 3s 386us/step - loss: 0.10
28 - acc: 0.9557 - val_loss: 0.2084 - val_acc: 0.9277
Epoch 21/50
7352/7352 [==============================] - 3s 386us/step - loss: 0.10
29 - acc: 0.9531 - val_loss: 0.1996 - val_acc: 0.9287
Epoch 22/50
7352/7352 [==============================] - 3s 386us/step - loss: 0.10
03 - acc: 0.9540 - val_loss: 0.2049 - val_acc: 0.9237
Epoch 23/50
7352/7352 [==============================] - 3s 384us/step - loss: 0.09
84 - acc: 0.9547 - val_loss: 0.2078 - val_acc: 0.9226
Epoch 24/50
7352/7352 [==============================] - 3s 387us/step - loss: 0.09
93 - acc: 0.9558 - val_loss: 0.2227 - val_acc: 0.9230
Epoch 25/50
7352/7352 [==============================] - 3s 384us/step - loss: 0.09
68 - acc: 0.9566 - val_loss: 0.2097 - val_acc: 0.9179
Epoch 26/50
7352/7352 [==============================] - 3s 384us/step - loss: 0.09
53 - acc: 0.9547 - val_loss: 0.2080 - val_acc: 0.9274
Epoch 27/50
7352/7352 [==============================] - 3s 387us/step - loss: 0.09
56 - acc: 0.9569 - val_loss: 0.1902 - val_acc: 0.9301
Epoch 28/50
7352/7352 [==============================] - 3s 389us/step - loss: 0.09
47 - acc: 0.9565 - val_loss: 0.2011 - val_acc: 0.9260
Epoch 29/50
7352/7352 [==============================] - 3s 387us/step - loss: 0.09
33 - acc: 0.9573 - val_loss: 0.2121 - val_acc: 0.9216
Epoch 30/50
7352/7352 [==============================] - 3s 388us/step - loss: 0.09
17 - acc: 0.9570 - val_loss: 0.2003 - val_acc: 0.9216
Epoch 31/50
7352/7352 [==============================] - 3s 392us/step - loss: 0.09
07 - acc: 0.9589 - val_loss: 0.2169 - val_acc: 0.9189
Epoch 32/50
7352/7352 [==============================] - 3s 388us/step - loss: 0.08
84 - acc: 0.9603 - val_loss: 0.1968 - val_acc: 0.9257
Epoch 33/50
```

```
7352/7352 [==============================] - 3s 386us/step - loss: 0.08
79 - acc: 0.9603 - val_loss: 0.2100 - val_acc: 0.9264
Epoch 34/50
7352/7352 [==============================] - 3s 387us/step - loss: 0.08
67 - acc: 0.9604 - val_loss: 0.2015 - val_acc: 0.9230
Epoch 35/50
7352/7352 [==============================] - 3s 389us/step - loss: 0.08
52 - acc: 0.9611 - val_loss: 0.1967 - val_acc: 0.9260
Epoch 36/50
7352/7352 [==============================] - 3s 391us/step - loss: 0.08
34 - acc: 0.9610 - val_loss: 0.1905 - val_acc: 0.9267
Epoch 37/50
7352/7352 [==============================] - 3s 389us/step - loss: 0.08
44 - acc: 0.9612 - val_loss: 0.1909 - val_acc: 0.9253
Epoch 38/50
7352/7352 [==============================] - 3s 388us/step - loss: 0.07
99 - acc: 0.9629 - val_loss: 0.2035 - val_acc: 0.9223
Epoch 39/50
7352/7352 [==============================] - 3s 390us/step - loss: 0.07
96 - acc: 0.9633 - val_loss: 0.1967 - val_acc: 0.9328
Epoch 40/50
7352/7352 [==============================] - 3s 390us/step - loss: 0.07
92 - acc: 0.9645 - val_loss: 0.2069 - val_acc: 0.9223
Epoch 41/50
7352/7352 [==============================] - 3s 385us/step - loss: 0.07
73 - acc: 0.9656 - val_loss: 0.2352 - val_acc: 0.9203
Epoch 42/50
7352/7352 [==============================] - 3s 390us/step - loss: 0.07
76 - acc: 0.9631 - val_loss: 0.1964 - val_acc: 0.9294
Epoch 43/50
7352/7352 [==============================] - 3s 385us/step - loss: 0.07
43 - acc: 0.9675 - val_loss: 0.2191 - val_acc: 0.9203
Epoch 44/50
7352/7352 [==============================] - 3s 385us/step - loss: 0.07
47 - acc: 0.9664 - val_loss: 0.1891 - val_acc: 0.9355
Epoch 45/50
7352/7352 [==============================] - 3s 386us/step - loss: 0.07
16 - acc: 0.9664 - val_loss: 0.2050 - val_acc: 0.9253
Epoch 46/50
```

```
7352/7352 [==============================] - 3s 389us/step - loss: 0.07
33 - acc: 0.9656 - val_loss: 0.2120 - val_acc: 0.9277
Epoch 47/50
7352/7352 [==============================] - 3s 385us/step - loss: 0.07
04 - acc: 0.9682 - val_loss: 0.1898 - val_acc: 0.9308
Epoch 48/50
7352/7352 [==============================] - 3s 385us/step - loss: 0.06
96 - acc: 0.9678 - val_loss: 0.1956 - val_acc: 0.9294
Epoch 49/50
7352/7352 [==============================] - 3s 388us/step - loss: 0.06
75 - acc: 0.9682 - val_loss: 0.1868 - val_acc: 0.9277
Epoch 50/50
7352/7352 [==============================] - 3s 389us/step - loss: 0.06
82 - acc: 0.9699 - val_loss: 0.1671 - val_acc: 0.9433
```

Out[0]:

```
<keras.callbacks.History at 0x7fa9b4cecba8>
```

In [0]:

```
# Confusion Matrix
# print(confusion_matrix(Y_test, model.predict(X_test)))
conf = pd.DataFrame(confusion_matrix(Y_test, model.predict(X_test)))
conf
```

Out[0]:

| Pred | LAYING | SITTING | STANDING | WALKING | WALKING_DOWNSTA |
|---|---|---|---|---|---|
| True | | | | | |
| LAYING | 534 | 0 | 0 | 0 | 0 |
| SITTING | 0 | 420 | 67 | 2 | 0 |
| STANDING | 0 | 52 | 480 | 0 | 0 |
| WALKING | 0 | 0 | 0 | 492 | 1 |
| WALKING_DOWNSTAIRS | 0 | 0 | 0 | 1 | 406 |
| WALKING_UPSTAIRS | 0 | 12 | 0 | 1 | 10 |

In [0]:

```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 0s 77us/step
```

In [0]:

```
score
```

Out[0]:

```
[0.16710662225743034, 0.9433322022395657]
```

In [0]:

```
score_train = model.evaluate(X_train, Y_train)
score_train
```

```
7352/7352 [==============================] - 1s 96us/step
```

Out[0]:

```
[0.06354207507458427, 0.9729325353645266]
```

## 2- layers of CNN

In [0]:

```
# Initiliazing the sequential model
model = Sequential()
model.add(Conv1D(filters=300, kernel_size=15, padding='same', activatio
n='sigmoid', input_shape=(timesteps, input_dim)))
model.add(MaxPooling1D(pool_size=15))
model.add(Conv1D(filters=70, kernel_size=15, padding='same', activation
='sigmoid', input_shape=(timesteps, input_dim)))
#model.add(MaxPooling1D(pool_size=9))
#model.add(Dropout(0.7))
#Configuring the parameters
#model.add(LSTM(50, input_shape=(timesteps, input_dim)))
#model.add(Dropout(0.6))
# Adding a dropout layer
# Adding a dense output layer with sigmoid activation
model.add(Flatten())
model.add(Dense(n_classes, activation='sigmoid'))

model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
```

```
conv1d_25 (Conv1D)           (None, 128, 300)          40800
_________________________________________________________________
max_pooling1d_24 (MaxPooling (None, 8, 300)            0
_________________________________________________________________
conv1d_26 (Conv1D)           (None, 8, 70)             315070
_________________________________________________________________
flatten_22 (Flatten)         (None, 560)               0
_________________________________________________________________
dense_22 (Dense)             (None, 6)                 3366
=================================================================
Total params: 359,236
Trainable params: 359,236
Non-trainable params: 0
_________________________________________________________________
```

In [0]:

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

In [0]:

```
# Training the model
model.fit(X_train,
          Y_train,
          batch_size=10,
          validation_data=(X_test, Y_test),
          epochs=50)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/50
7352/7352 [==============================] - 5s 630us/step - loss: 1.30
05 - acc: 0.3883 - val_loss: 0.8273 - val_acc: 0.6753
Epoch 2/50
7352/7352 [==============================] - 3s 446us/step - loss: 0.46
85 - acc: 0.8134 - val_loss: 0.4067 - val_acc: 0.8677
Epoch 3/50
7352/7352 [==============================] - 3s 452us/step - loss: 0.19
04 - acc: 0.9312 - val_loss: 0.2962 - val_acc: 0.8958
Epoch 4/50
7352/7352 [==============================] - 3s 454us/step - loss: 0.16
```

```
20 - acc: 0.9353 - val_loss: 0.4242 - val_acc: 0.8273
Epoch 5/50
7352/7352 [==============================] - 3s 449us/step - loss: 0.14
03 - acc: 0.9438 - val_loss: 0.2267 - val_acc: 0.9074
Epoch 6/50
7352/7352 [==============================] - 3s 452us/step - loss: 0.13
18 - acc: 0.9461 - val_loss: 0.2307 - val_acc: 0.9135
Epoch 7/50
7352/7352 [==============================] - 3s 447us/step - loss: 0.12
42 - acc: 0.9508 - val_loss: 0.2210 - val_acc: 0.9063
Epoch 8/50
7352/7352 [==============================] - 3s 450us/step - loss: 0.12
30 - acc: 0.9493 - val_loss: 0.2101 - val_acc: 0.9108
Epoch 9/50
7352/7352 [==============================] - 3s 446us/step - loss: 0.11
68 - acc: 0.9506 - val_loss: 0.2306 - val_acc: 0.9087
Epoch 10/50
7352/7352 [==============================] - 3s 449us/step - loss: 0.11
41 - acc: 0.9533 - val_loss: 0.2078 - val_acc: 0.9141
Epoch 11/50
7352/7352 [==============================] - 3s 449us/step - loss: 0.11
29 - acc: 0.9535 - val_loss: 0.2381 - val_acc: 0.9084
Epoch 12/50
7352/7352 [==============================] - 3s 448us/step - loss: 0.11
20 - acc: 0.9532 - val_loss: 0.2005 - val_acc: 0.9209
Epoch 13/50
7352/7352 [==============================] - 3s 453us/step - loss: 0.10
88 - acc: 0.9525 - val_loss: 0.2118 - val_acc: 0.9162
Epoch 14/50
7352/7352 [==============================] - 3s 450us/step - loss: 0.10
81 - acc: 0.9539 - val_loss: 0.2998 - val_acc: 0.9043
Epoch 15/50
7352/7352 [==============================] - 3s 448us/step - loss: 0.10
51 - acc: 0.9553 - val_loss: 0.1967 - val_acc: 0.9267
Epoch 16/50
7352/7352 [==============================] - 3s 450us/step - loss: 0.10
29 - acc: 0.9567 - val_loss: 0.2115 - val_acc: 0.9192
Epoch 17/50
7352/7352 [==============================] - 3s 452us/step - loss: 0.10
```

```
25 - acc: 0.9570 - val_loss: 0.2075 - val_acc: 0.9199
Epoch 18/50
7352/7352 [==============================] - 3s 449us/step - loss: 0.10
07 - acc: 0.9562 - val_loss: 0.2361 - val_acc: 0.9186
Epoch 19/50
7352/7352 [==============================] - 3s 450us/step - loss: 0.09
95 - acc: 0.9547 - val_loss: 0.2167 - val_acc: 0.9223
Epoch 20/50
7352/7352 [==============================] - 3s 451us/step - loss: 0.09
62 - acc: 0.9563 - val_loss: 0.2410 - val_acc: 0.9135
Epoch 21/50
7352/7352 [==============================] - 3s 447us/step - loss: 0.09
57 - acc: 0.9577 - val_loss: 0.2287 - val_acc: 0.9128
Epoch 22/50
7352/7352 [==============================] - 3s 448us/step - loss: 0.09
26 - acc: 0.9597 - val_loss: 0.2431 - val_acc: 0.9141
Epoch 23/50
7352/7352 [==============================] - 3s 447us/step - loss: 0.09
08 - acc: 0.9603 - val_loss: 0.2083 - val_acc: 0.9325
Epoch 24/50
7352/7352 [==============================] - 3s 451us/step - loss: 0.09
00 - acc: 0.9606 - val_loss: 0.2402 - val_acc: 0.9182
Epoch 25/50
7352/7352 [==============================] - 3s 449us/step - loss: 0.08
65 - acc: 0.9595 - val_loss: 0.2466 - val_acc: 0.9162
Epoch 26/50
7352/7352 [==============================] - 3s 448us/step - loss: 0.08
42 - acc: 0.9629 - val_loss: 0.2298 - val_acc: 0.9182
Epoch 27/50
7352/7352 [==============================] - 3s 450us/step - loss: 0.08
54 - acc: 0.9631 - val_loss: 0.2572 - val_acc: 0.9169
Epoch 28/50
7352/7352 [==============================] - 3s 447us/step - loss: 0.08
08 - acc: 0.9604 - val_loss: 0.2706 - val_acc: 0.9189
Epoch 29/50
7352/7352 [==============================] - 3s 448us/step - loss: 0.07
85 - acc: 0.9644 - val_loss: 0.2323 - val_acc: 0.9169
Epoch 30/50
7352/7352 [==============================] - 3s 454us/step - loss: 0.07
```

```
77 - acc: 0.9622 - val_loss: 0.2699 - val_acc: 0.9179
Epoch 31/50
7352/7352 [==============================] - 3s 451us/step - loss: 0.07
50 - acc: 0.9650 - val_loss: 0.2286 - val_acc: 0.9291
Epoch 32/50
7352/7352 [==============================] - 3s 449us/step - loss: 0.07
57 - acc: 0.9635 - val_loss: 0.2430 - val_acc: 0.9230
Epoch 33/50
7352/7352 [==============================] - 3s 453us/step - loss: 0.07
46 - acc: 0.9630 - val_loss: 0.2714 - val_acc: 0.9097
Epoch 34/50
7352/7352 [==============================] - 3s 452us/step - loss: 0.07
14 - acc: 0.9674 - val_loss: 0.2597 - val_acc: 0.9206
Epoch 35/50
7352/7352 [==============================] - 3s 446us/step - loss: 0.06
86 - acc: 0.9689 - val_loss: 0.2425 - val_acc: 0.9311
Epoch 36/50
7352/7352 [==============================] - 3s 451us/step - loss: 0.06
69 - acc: 0.9680 - val_loss: 0.2777 - val_acc: 0.9141
Epoch 37/50
7352/7352 [==============================] - 3s 447us/step - loss: 0.06
71 - acc: 0.9689 - val_loss: 0.2681 - val_acc: 0.9260
Epoch 38/50
7352/7352 [==============================] - 3s 448us/step - loss: 0.06
43 - acc: 0.9708 - val_loss: 0.2755 - val_acc: 0.9230
Epoch 39/50
7352/7352 [==============================] - 3s 452us/step - loss: 0.06
26 - acc: 0.9709 - val_loss: 0.2591 - val_acc: 0.9304
Epoch 40/50
7352/7352 [==============================] - 3s 447us/step - loss: 0.06
20 - acc: 0.9721 - val_loss: 0.2838 - val_acc: 0.9213
Epoch 41/50
7352/7352 [==============================] - 3s 448us/step - loss: 0.06
08 - acc: 0.9713 - val_loss: 0.3815 - val_acc: 0.8989
Epoch 42/50
7352/7352 [==============================] - 3s 453us/step - loss: 0.05
78 - acc: 0.9713 - val_loss: 0.2855 - val_acc: 0.9253
Epoch 43/50
7352/7352 [==============================] - 3s 456us/step - loss: 0.05
```

```
83 - acc: 0.9717 - val_loss: 0.2754 - val_acc: 0.9325
Epoch 44/50
7352/7352 [==============================] - 3s 450us/step - loss: 0.05
67 - acc: 0.9744 - val_loss: 0.2670 - val_acc: 0.9226
Epoch 45/50
7352/7352 [==============================] - 3s 444us/step - loss: 0.05
58 - acc: 0.9750 - val_loss: 0.2887 - val_acc: 0.9277
Epoch 46/50
7352/7352 [==============================] - 3s 447us/step - loss: 0.05
41 - acc: 0.9710 - val_loss: 0.3079 - val_acc: 0.9226
Epoch 47/50
7352/7352 [==============================] - 3s 451us/step - loss: 0.05
23 - acc: 0.9770 - val_loss: 0.2853 - val_acc: 0.9257
Epoch 48/50
7352/7352 [==============================] - 3s 444us/step - loss: 0.05
31 - acc: 0.9763 - val_loss: 0.3183 - val_acc: 0.9186
Epoch 49/50
7352/7352 [==============================] - 3s 448us/step - loss: 0.05
06 - acc: 0.9752 - val_loss: 0.3276 - val_acc: 0.9230
Epoch 50/50
7352/7352 [==============================] - 3s 445us/step - loss: 0.05
07 - acc: 0.9777 - val_loss: 0.3183 - val_acc: 0.9216
```

Out[0]:
```
<keras.callbacks.History at 0x7fa9b45c8240>
```

In [0]:
```python
score = model.evaluate(X_test, Y_test)
print(score)

score_train = model.evaluate(X_train, Y_train)
print(score_train)
```

```
2947/2947 [==============================] - 0s 89us/step
[0.3449924494941751, 0.9192399049881235]
7352/7352 [==============================] - 1s 85us/step
[0.045094185050339956, 0.9767410228509249]
```

In [0]:
```python
# Confusion Matrix
# print(confusion_matrix(Y_test, model.predict(X_test)))
```

```
conf = pd.DataFrame(confusion_matrix(Y_test, model.predict(X_test)))
conf
```

Out[0]:

| Pred | LAYING | SITTING | STANDING | WALKING | WALKING_DOWNSTA |
|---|---|---|---|---|---|
| True | | | | | |
| **LAYING** | 510 | 0 | 0 | 0 | 0 |
| **SITTING** | 0 | 395 | 94 | 0 | 0 |
| **STANDING** | 0 | 70 | 461 | 1 | 0 |
| **WALKING** | 0 | 0 | 0 | 480 | 7 |
| **WALKING_DOWNSTAIRS** | 0 | 0 | 0 | 4 | 411 |
| **WALKING_UPSTAIRS** | 0 | 0 | 0 | 0 | 19 |

# CNN + LSTM

In [0]:

```
# Initiliazing the sequential model
model = Sequential()
model.add(Conv1D(filters=300, kernel_size=15, padding='same', activatio
n='sigmoid', input_shape=(timesteps, input_dim)))
model.add(MaxPooling1D(pool_size=15))
#model.add(Conv1D(filters=70, kernel_size=15, padding='same', activatio
n='sigmoid', input_shape=(timesteps, input_dim)))

#model.add(MaxPooling1D(pool_size=9))
#model.add(Dropout(0.7))
#Configuring the parameters
model.add(LSTM(50, input_shape=(timesteps, input_dim)))
model.add(Dropout(0.6))
# Adding a dropout layer
# Adding a dense output layer with sigmoid activation
#model.add(Flatten())
```

```python
model.add(Dense(n_classes, activation='sigmoid'))

model.summary()

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


# Training the model
model.fit(X_train,
          Y_train,
          batch_size=10,
          validation_data=(X_test, Y_test),
          epochs=50)


score = model.evaluate(X_test, Y_test)
print(score)

score_train = model.evaluate(X_train, Y_train)
print(score_train)

# Confusion Matrix
# print(confusion_matrix(Y_test, model.predict(X_test)))
conf = pd.DataFrame(confusion_matrix(Y_test, model.predict(X_test)))
conf
```

```
W0616 07:39:34.822537 140374351411072 nn_ops.py:4224] Large dropout rat
e: 0.6 (>0.5). In TensorFlow 2.x, dropout() uses dropout rate instead o
f keep_prob. Please ensure that this is intended.
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_40 (Conv1D)           (None, 128, 300)          40800
_________________________________________________________________
max_pooling1d_33 (MaxPooling (None, 8, 300)            0
_________________________________________________________________
lstm_8 (LSTM)                (None, 50)                70200
```

```
_________________________________________________________________
dropout_5 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_27 (Dense)             (None, 6)                 306
=================================================================
Total params: 111,306
Trainable params: 111,306
Non-trainable params: 0
_________________________________________________________________
Train on 7352 samples, validate on 2947 samples
Epoch 1/50
7352/7352 [==============================] - 15s 2ms/step - loss: 1.133
7 - acc: 0.5303 - val_loss: 0.7919 - val_acc: 0.6546
Epoch 2/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.784
4 - acc: 0.6234 - val_loss: 0.7166 - val_acc: 0.7204
Epoch 3/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.703
9 - acc: 0.6748 - val_loss: 0.6864 - val_acc: 0.6834
Epoch 4/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.649
9 - acc: 0.7133 - val_loss: 0.5547 - val_acc: 0.7662
Epoch 5/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.580
1 - acc: 0.7447 - val_loss: 0.4997 - val_acc: 0.7611
Epoch 6/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.475
5 - acc: 0.7920 - val_loss: 0.4990 - val_acc: 0.7516
Epoch 7/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.419
2 - acc: 0.8221 - val_loss: 0.4501 - val_acc: 0.7713
Epoch 8/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.369
7 - acc: 0.8555 - val_loss: 0.3573 - val_acc: 0.8941
Epoch 9/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.315
2 - acc: 0.8881 - val_loss: 0.3059 - val_acc: 0.8975
Epoch 10/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.258
```

```
6 - acc: 0.9246 - val_loss: 0.2724 - val_acc: 0.9094
Epoch 11/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.250
3 - acc: 0.9240 - val_loss: 0.2439 - val_acc: 0.9111
Epoch 12/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.227
7 - acc: 0.9317 - val_loss: 0.2399 - val_acc: 0.9145
Epoch 13/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.214
2 - acc: 0.9361 - val_loss: 0.2215 - val_acc: 0.9196
Epoch 14/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.201
0 - acc: 0.9366 - val_loss: 0.2228 - val_acc: 0.9216
Epoch 15/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.185
8 - acc: 0.9402 - val_loss: 0.3012 - val_acc: 0.8955
Epoch 16/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.181
3 - acc: 0.9433 - val_loss: 0.2351 - val_acc: 0.9237
Epoch 17/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.178
5 - acc: 0.9441 - val_loss: 0.2208 - val_acc: 0.9155
Epoch 18/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.168
6 - acc: 0.9410 - val_loss: 0.2250 - val_acc: 0.9291
Epoch 19/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.172
5 - acc: 0.9408 - val_loss: 0.2275 - val_acc: 0.9260
Epoch 20/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.168
1 - acc: 0.9430 - val_loss: 0.2231 - val_acc: 0.9196
Epoch 21/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.159
0 - acc: 0.9433 - val_loss: 0.2180 - val_acc: 0.9240
Epoch 22/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.147
4 - acc: 0.9474 - val_loss: 0.3544 - val_acc: 0.8816
Epoch 23/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.160
```

```
5 - acc: 0.9456 - val_loss: 0.2234 - val_acc: 0.9209
Epoch 24/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.144
8 - acc: 0.9494 - val_loss: 0.2287 - val_acc: 0.9223
Epoch 25/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.149
9 - acc: 0.9491 - val_loss: 0.2094 - val_acc: 0.9233
Epoch 26/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.151
0 - acc: 0.9471 - val_loss: 0.2226 - val_acc: 0.9267
Epoch 27/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.151
7 - acc: 0.9495 - val_loss: 0.2131 - val_acc: 0.9260
Epoch 28/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.146
4 - acc: 0.9436 - val_loss: 0.2201 - val_acc: 0.9213
Epoch 29/50
7352/7352 [==============================] - 14s 2ms/step - loss: 0.143
8 - acc: 0.9490 - val_loss: 0.2058 - val_acc: 0.9250
Epoch 30/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.139
8 - acc: 0.9494 - val_loss: 0.2356 - val_acc: 0.9270
Epoch 31/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.137
4 - acc: 0.9490 - val_loss: 0.2204 - val_acc: 0.9250
Epoch 32/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.133
4 - acc: 0.9505 - val_loss: 0.2372 - val_acc: 0.9220
Epoch 33/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.139
0 - acc: 0.9468 - val_loss: 0.2412 - val_acc: 0.9247
Epoch 34/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.140
2 - acc: 0.9487 - val_loss: 0.2264 - val_acc: 0.9253
Epoch 35/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.134
6 - acc: 0.9501 - val_loss: 0.2530 - val_acc: 0.9162
Epoch 36/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.135
```

```
4 - acc: 0.9501 - val_loss: 0.2319 - val_acc: 0.9179
Epoch 37/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.134
9 - acc: 0.9508 - val_loss: 0.2315 - val_acc: 0.9257
Epoch 38/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.130
6 - acc: 0.9532 - val_loss: 0.2304 - val_acc: 0.9274
Epoch 39/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.137
6 - acc: 0.9508 - val_loss: 0.2402 - val_acc: 0.9298
Epoch 40/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.133
3 - acc: 0.9493 - val_loss: 0.2290 - val_acc: 0.9270
Epoch 41/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.125
5 - acc: 0.9513 - val_loss: 0.2715 - val_acc: 0.9186
Epoch 42/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.129
1 - acc: 0.9535 - val_loss: 0.2591 - val_acc: 0.9152
Epoch 43/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.130
0 - acc: 0.9524 - val_loss: 0.2332 - val_acc: 0.9287
Epoch 44/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.129
8 - acc: 0.9544 - val_loss: 0.2711 - val_acc: 0.9216
Epoch 45/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.128
3 - acc: 0.9506 - val_loss: 0.2492 - val_acc: 0.9169
Epoch 46/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.126
4 - acc: 0.9508 - val_loss: 0.2511 - val_acc: 0.9226
Epoch 47/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.123
8 - acc: 0.9521 - val_loss: 0.2565 - val_acc: 0.9213
Epoch 48/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.134
6 - acc: 0.9520 - val_loss: 0.2876 - val_acc: 0.9165
Epoch 49/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.125
```

```
7 - acc: 0.9529 - val_loss: 0.2678 - val_acc: 0.9121
Epoch 50/50
7352/7352 [==============================] - 13s 2ms/step - loss: 0.121
2 - acc: 0.9529 - val_loss: 0.2423 - val_acc: 0.9277
2947/2947 [==============================] - 1s 207us/step
[0.24233421574737116, 0.9277231082456736]
7352/7352 [==============================] - 1s 203us/step
[0.09086589647518013, 0.9604189336235038]
```

Out[0]:

| **Pred** | **LAYING** | **SITTING** | **STANDING** | **WALKING** | **WALKING_DOWNSTA** |
|---|---|---|---|---|---|
| **True** | | | | | |
| **LAYING** | 534 | 0 | 0 | 0 | 0 |
| **SITTING** | 0 | 401 | 88 | 0 | 0 |
| **STANDING** | 0 | 82 | 446 | 0 | 0 |
| **WALKING** | 0 | 0 | 0 | 491 | 5 |
| **WALKING_DOWNSTAIRS** | 0 | 0 | 0 | 0 | 412 |
| **WALKING_UPSTAIRS** | 0 | 0 | 0 | 0 | 21 |

# 2 CNN layers, 3 Pooling layers and one Dropout layer

In [0]:

```
# Initiliazing the sequential model
model = Sequential()
model.add(Conv1D(filters=200, kernel_size=15, padding='same', activatio
n='sigmoid', input_shape=(timesteps, input_dim)))
model.add(MaxPooling1D(pool_size=15))
model.add(MaxPooling1D(pool_size=2))
model.add(Conv1D(filters=100, kernel_size=15, padding='same', activatio
n='sigmoid'))
```

```
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.4))
model.add(Flatten())
model.add(Dense(n_classes, activation='sigmoid', kernel_initializer='gl
orot_normal'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_9 (Conv1D)            (None, 128, 200)          27200
_________________________________________________________________
max_pooling1d_15 (MaxPooling (None, 8, 200)            0
_________________________________________________________________
max_pooling1d_16 (MaxPooling (None, 4, 200)            0
_________________________________________________________________
conv1d_10 (Conv1D)           (None, 4, 100)            300100
_________________________________________________________________
max_pooling1d_17 (MaxPooling (None, 2, 100)            0
_________________________________________________________________
dropout_2 (Dropout)          (None, 2, 100)            0
_________________________________________________________________
flatten_6 (Flatten)          (None, 200)               0
_________________________________________________________________
dense_6 (Dense)              (None, 6)                 1206
=================================================================
Total params: 328,506
Trainable params: 328,506
Non-trainable params: 0
_________________________________________________________________
```

In [0]:

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

In [0]:

```
# Training the model
history = model.fit(X_train,
          Y_train,
```

```
          batch_size=50,
          validation_data=(X_test, Y_test),
          epochs=300)
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/300
7352/7352 [==============================] - 1s 190us/step - loss: 1.44
82 - acc: 0.3702 - val_loss: 1.0487 - val_acc: 0.6620
Epoch 2/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.73
33 - acc: 0.7153 - val_loss: 0.6046 - val_acc: 0.8358
Epoch 3/300
7352/7352 [==============================] - 1s 118us/step - loss: 0.43
40 - acc: 0.8398 - val_loss: 0.4747 - val_acc: 0.8619
Epoch 4/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.30
98 - acc: 0.9023 - val_loss: 0.3850 - val_acc: 0.8911
Epoch 5/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.22
76 - acc: 0.9285 - val_loss: 0.3341 - val_acc: 0.8904
Epoch 6/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.18
48 - acc: 0.9387 - val_loss: 0.2959 - val_acc: 0.8911
Epoch 7/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.16
74 - acc: 0.9411 - val_loss: 0.2790 - val_acc: 0.8948
Epoch 8/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.15
80 - acc: 0.9400 - val_loss: 0.2723 - val_acc: 0.9077
Epoch 9/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.14
57 - acc: 0.9433 - val_loss: 0.2694 - val_acc: 0.9006
Epoch 10/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.13
85 - acc: 0.9475 - val_loss: 0.2576 - val_acc: 0.9128
Epoch 11/300
7352/7352 [==============================] - 1s 118us/step - loss: 0.13
51 - acc: 0.9478 - val_loss: 0.2488 - val_acc: 0.9046
Epoch 12/300
7352/7352 [==============================] - 1s 118us/step - loss: 0.12
```

```
77 - acc: 0.9506 - val_loss: 0.2521 - val_acc: 0.9046
Epoch 13/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.12
60 - acc: 0.9505 - val_loss: 0.2557 - val_acc: 0.9074
Epoch 14/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.12
35 - acc: 0.9508 - val_loss: 0.2537 - val_acc: 0.9067
Epoch 15/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.12
08 - acc: 0.9512 - val_loss: 0.2324 - val_acc: 0.9118
Epoch 16/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.11
73 - acc: 0.9527 - val_loss: 0.2330 - val_acc: 0.9087
Epoch 17/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.11
71 - acc: 0.9527 - val_loss: 0.2274 - val_acc: 0.9141
Epoch 18/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.11
61 - acc: 0.9516 - val_loss: 0.2248 - val_acc: 0.9121
Epoch 19/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.11
55 - acc: 0.9517 - val_loss: 0.2345 - val_acc: 0.9094
Epoch 20/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.11
81 - acc: 0.9482 - val_loss: 0.2249 - val_acc: 0.9192
Epoch 21/300
7352/7352 [==============================] - 1s 123us/step - loss: 0.11
44 - acc: 0.9510 - val_loss: 0.2154 - val_acc: 0.9172
Epoch 22/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.10
98 - acc: 0.9551 - val_loss: 0.2205 - val_acc: 0.9206
Epoch 23/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.10
89 - acc: 0.9553 - val_loss: 0.2304 - val_acc: 0.9135
Epoch 24/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.10
96 - acc: 0.9539 - val_loss: 0.2208 - val_acc: 0.9203
Epoch 25/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.10
```

```
57 - acc: 0.9535 - val_loss: 0.2108 - val_acc: 0.9189
Epoch 26/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.10
79 - acc: 0.9540 - val_loss: 0.2128 - val_acc: 0.9233
Epoch 27/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.10
63 - acc: 0.9535 - val_loss: 0.2110 - val_acc: 0.9155
Epoch 28/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.10
43 - acc: 0.9562 - val_loss: 0.2033 - val_acc: 0.9230
Epoch 29/300
7352/7352 [==============================] - 1s 123us/step - loss: 0.10
44 - acc: 0.9558 - val_loss: 0.1967 - val_acc: 0.9206
Epoch 30/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.10
22 - acc: 0.9569 - val_loss: 0.2094 - val_acc: 0.9192
Epoch 31/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.10
36 - acc: 0.9557 - val_loss: 0.2120 - val_acc: 0.9213
Epoch 32/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.10
19 - acc: 0.9548 - val_loss: 0.1923 - val_acc: 0.9270
Epoch 33/300
7352/7352 [==============================] - 1s 118us/step - loss: 0.10
17 - acc: 0.9592 - val_loss: 0.1928 - val_acc: 0.9277
Epoch 34/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.09
88 - acc: 0.9577 - val_loss: 0.1858 - val_acc: 0.9287
Epoch 35/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.10
14 - acc: 0.9570 - val_loss: 0.1860 - val_acc: 0.9267
Epoch 36/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.09
78 - acc: 0.9592 - val_loss: 0.1929 - val_acc: 0.9294
Epoch 37/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.09
83 - acc: 0.9565 - val_loss: 0.1856 - val_acc: 0.9270
Epoch 38/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.09
```

```
85 - acc: 0.9587 - val_loss: 0.1882 - val_acc: 0.9253
Epoch 39/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.09
77 - acc: 0.9588 - val_loss: 0.1853 - val_acc: 0.9267
Epoch 40/300
7352/7352 [==============================] - 1s 117us/step - loss: 0.09
53 - acc: 0.9596 - val_loss: 0.2141 - val_acc: 0.9253
Epoch 41/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.09
80 - acc: 0.9580 - val_loss: 0.1916 - val_acc: 0.9216
Epoch 42/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.09
72 - acc: 0.9589 - val_loss: 0.1927 - val_acc: 0.9291
Epoch 43/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.09
85 - acc: 0.9589 - val_loss: 0.1854 - val_acc: 0.9270
Epoch 44/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.09
71 - acc: 0.9610 - val_loss: 0.1956 - val_acc: 0.9274
Epoch 45/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.09
44 - acc: 0.9614 - val_loss: 0.1949 - val_acc: 0.9393
Epoch 46/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.09
33 - acc: 0.9606 - val_loss: 0.2004 - val_acc: 0.9189
Epoch 47/300
7352/7352 [==============================] - 1s 118us/step - loss: 0.09
47 - acc: 0.9601 - val_loss: 0.1835 - val_acc: 0.9287
Epoch 48/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.09
26 - acc: 0.9582 - val_loss: 0.1806 - val_acc: 0.9277
Epoch 49/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.08
99 - acc: 0.9626 - val_loss: 0.1836 - val_acc: 0.9308
Epoch 50/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.09
00 - acc: 0.9629 - val_loss: 0.1881 - val_acc: 0.9270
Epoch 51/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.08
```

```
90 - acc: 0.9608 - val_loss: 0.1803 - val_acc: 0.9281
Epoch 52/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.08
95 - acc: 0.9627 - val_loss: 0.1770 - val_acc: 0.9277
Epoch 53/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.08
73 - acc: 0.9630 - val_loss: 0.1745 - val_acc: 0.9294
Epoch 54/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.08
70 - acc: 0.9629 - val_loss: 0.1714 - val_acc: 0.9308
Epoch 55/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.08
79 - acc: 0.9625 - val_loss: 0.1877 - val_acc: 0.9253
Epoch 56/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.08
71 - acc: 0.9631 - val_loss: 0.1645 - val_acc: 0.9308
Epoch 57/300
7352/7352 [==============================] - 1s 118us/step - loss: 0.08
79 - acc: 0.9619 - val_loss: 0.1709 - val_acc: 0.9308
Epoch 58/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.08
73 - acc: 0.9623 - val_loss: 0.1725 - val_acc: 0.9308
Epoch 59/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.08
62 - acc: 0.9611 - val_loss: 0.1810 - val_acc: 0.9267
Epoch 60/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.08
63 - acc: 0.9641 - val_loss: 0.1858 - val_acc: 0.9264
Epoch 61/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.08
48 - acc: 0.9645 - val_loss: 0.1649 - val_acc: 0.9308
Epoch 62/300
7352/7352 [==============================] - 1s 118us/step - loss: 0.08
45 - acc: 0.9648 - val_loss: 0.1807 - val_acc: 0.9284
Epoch 63/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.08
32 - acc: 0.9663 - val_loss: 0.1642 - val_acc: 0.9382
Epoch 64/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.08
```

```
27 - acc: 0.9635 - val_loss: 0.1722 - val_acc: 0.9304
Epoch 65/300
7352/7352 [==============================] - 1s 118us/step - loss: 0.08
20 - acc: 0.9642 - val_loss: 0.1737 - val_acc: 0.9410
Epoch 66/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.08
60 - acc: 0.9649 - val_loss: 0.1742 - val_acc: 0.9308
Epoch 67/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.08
11 - acc: 0.9638 - val_loss: 0.1720 - val_acc: 0.9318
Epoch 68/300
7352/7352 [==============================] - 1s 118us/step - loss: 0.08
03 - acc: 0.9650 - val_loss: 0.1692 - val_acc: 0.9301
Epoch 69/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.08
29 - acc: 0.9649 - val_loss: 0.1679 - val_acc: 0.9335
Epoch 70/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.08
08 - acc: 0.9665 - val_loss: 0.1683 - val_acc: 0.9321
Epoch 71/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.07
82 - acc: 0.9671 - val_loss: 0.1894 - val_acc: 0.9287
Epoch 72/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.07
72 - acc: 0.9694 - val_loss: 0.1833 - val_acc: 0.9345
Epoch 73/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.07
68 - acc: 0.9693 - val_loss: 0.1762 - val_acc: 0.9311
Epoch 74/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.07
59 - acc: 0.9687 - val_loss: 0.1863 - val_acc: 0.9294
Epoch 75/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.07
60 - acc: 0.9678 - val_loss: 0.1683 - val_acc: 0.9352
Epoch 76/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.07
51 - acc: 0.9691 - val_loss: 0.1723 - val_acc: 0.9423
Epoch 77/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.07
```

```
75 - acc: 0.9663 - val_loss: 0.1743 - val_acc: 0.9311
Epoch 78/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.07
87 - acc: 0.9679 - val_loss: 0.1868 - val_acc: 0.9270
Epoch 79/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.07
59 - acc: 0.9687 - val_loss: 0.1696 - val_acc: 0.9352
Epoch 80/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.07
53 - acc: 0.9676 - val_loss: 0.1640 - val_acc: 0.9348
Epoch 81/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.07
55 - acc: 0.9687 - val_loss: 0.1765 - val_acc: 0.9325
Epoch 82/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.07
56 - acc: 0.9693 - val_loss: 0.1847 - val_acc: 0.9287
Epoch 83/300
7352/7352 [==============================] - 1s 118us/step - loss: 0.07
47 - acc: 0.9698 - val_loss: 0.1997 - val_acc: 0.9308
Epoch 84/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.07
42 - acc: 0.9697 - val_loss: 0.1743 - val_acc: 0.9365
Epoch 85/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.07
34 - acc: 0.9697 - val_loss: 0.1816 - val_acc: 0.9301
Epoch 86/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.07
45 - acc: 0.9691 - val_loss: 0.1794 - val_acc: 0.9345
Epoch 87/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.07
61 - acc: 0.9684 - val_loss: 0.1781 - val_acc: 0.9338
Epoch 88/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.07
33 - acc: 0.9694 - val_loss: 0.1774 - val_acc: 0.9342
Epoch 89/300
7352/7352 [==============================] - 1s 123us/step - loss: 0.07
23 - acc: 0.9701 - val_loss: 0.1556 - val_acc: 0.9457
Epoch 90/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.07
```

```
14 - acc: 0.9702 - val_loss: 0.2043 - val_acc: 0.9230
Epoch 91/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.07
22 - acc: 0.9684 - val_loss: 0.1710 - val_acc: 0.9437
Epoch 92/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.06
92 - acc: 0.9716 - val_loss: 0.1755 - val_acc: 0.9396
Epoch 93/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.07
08 - acc: 0.9691 - val_loss: 0.1919 - val_acc: 0.9304
Epoch 94/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.06
61 - acc: 0.9733 - val_loss: 0.1737 - val_acc: 0.9427
Epoch 95/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.06
97 - acc: 0.9701 - val_loss: 0.1671 - val_acc: 0.9454
Epoch 96/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.07
07 - acc: 0.9729 - val_loss: 0.1900 - val_acc: 0.9348
Epoch 97/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.06
37 - acc: 0.9736 - val_loss: 0.1613 - val_acc: 0.9393
Epoch 98/300
7352/7352 [==============================] - 1s 118us/step - loss: 0.06
62 - acc: 0.9717 - val_loss: 0.1860 - val_acc: 0.9362
Epoch 99/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.06
28 - acc: 0.9727 - val_loss: 0.2216 - val_acc: 0.9237
Epoch 100/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.06
66 - acc: 0.9721 - val_loss: 0.1643 - val_acc: 0.9423
Epoch 101/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.06
35 - acc: 0.9736 - val_loss: 0.1788 - val_acc: 0.9447
Epoch 102/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.06
72 - acc: 0.9716 - val_loss: 0.1733 - val_acc: 0.9467
Epoch 103/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.06
```

```
28 - acc: 0.9732 - val_loss: 0.1727 - val_acc: 0.9423
Epoch 104/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.06
16 - acc: 0.9733 - val_loss: 0.1731 - val_acc: 0.9406
Epoch 105/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.06
40 - acc: 0.9728 - val_loss: 0.1722 - val_acc: 0.9474
Epoch 106/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.06
16 - acc: 0.9737 - val_loss: 0.1770 - val_acc: 0.9416
Epoch 107/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.05
91 - acc: 0.9750 - val_loss: 0.1592 - val_acc: 0.9494
Epoch 108/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.05
91 - acc: 0.9744 - val_loss: 0.1596 - val_acc: 0.9494
Epoch 109/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.05
66 - acc: 0.9758 - val_loss: 0.1583 - val_acc: 0.9491
Epoch 110/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.05
87 - acc: 0.9759 - val_loss: 0.2098 - val_acc: 0.9291
Epoch 111/300
7352/7352 [==============================] - 1s 123us/step - loss: 0.05
79 - acc: 0.9752 - val_loss: 0.1948 - val_acc: 0.9359
Epoch 112/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.05
61 - acc: 0.9767 - val_loss: 0.1789 - val_acc: 0.9403
Epoch 113/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.05
50 - acc: 0.9771 - val_loss: 0.1410 - val_acc: 0.9576
Epoch 114/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.05
51 - acc: 0.9786 - val_loss: 0.1447 - val_acc: 0.9583
Epoch 115/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.05
70 - acc: 0.9758 - val_loss: 0.1570 - val_acc: 0.9522
Epoch 116/300
7352/7352 [==============================] - 1s 124us/step - loss: 0.05
```

```
88 - acc: 0.9755 - val_loss: 0.1693 - val_acc: 0.9528
Epoch 117/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.05
42 - acc: 0.9759 - val_loss: 0.1596 - val_acc: 0.9474
Epoch 118/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.05
07 - acc: 0.9786 - val_loss: 0.1767 - val_acc: 0.9481
Epoch 119/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.05
85 - acc: 0.9736 - val_loss: 0.1530 - val_acc: 0.9539
Epoch 120/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.05
38 - acc: 0.9786 - val_loss: 0.1826 - val_acc: 0.9321
Epoch 121/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.05
07 - acc: 0.9796 - val_loss: 0.1717 - val_acc: 0.9413
Epoch 122/300
7352/7352 [==============================] - 1s 123us/step - loss: 0.05
24 - acc: 0.9793 - val_loss: 0.1558 - val_acc: 0.9549
Epoch 123/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.05
28 - acc: 0.9778 - val_loss: 0.1537 - val_acc: 0.9522
Epoch 124/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.05
20 - acc: 0.9782 - val_loss: 0.1523 - val_acc: 0.9542
Epoch 125/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.05
02 - acc: 0.9805 - val_loss: 0.1652 - val_acc: 0.9450
Epoch 126/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.04
84 - acc: 0.9805 - val_loss: 0.1638 - val_acc: 0.9508
Epoch 127/300
7352/7352 [==============================] - 1s 118us/step - loss: 0.04
77 - acc: 0.9803 - val_loss: 0.2000 - val_acc: 0.9315
Epoch 128/300
7352/7352 [==============================] - 1s 117us/step - loss: 0.04
75 - acc: 0.9800 - val_loss: 0.1583 - val_acc: 0.9484
Epoch 129/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.04
```

```
88 - acc: 0.9799 - val_loss: 0.1812 - val_acc: 0.9437
Epoch 130/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.04
67 - acc: 0.9807 - val_loss: 0.1611 - val_acc: 0.9471
Epoch 131/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.04
39 - acc: 0.9814 - val_loss: 0.1604 - val_acc: 0.9593
Epoch 132/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.04
44 - acc: 0.9815 - val_loss: 0.1826 - val_acc: 0.9444
Epoch 133/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.04
37 - acc: 0.9829 - val_loss: 0.1566 - val_acc: 0.9566
Epoch 134/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.04
59 - acc: 0.9818 - val_loss: 0.1662 - val_acc: 0.9501
Epoch 135/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.04
34 - acc: 0.9823 - val_loss: 0.1521 - val_acc: 0.9511
Epoch 136/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.04
17 - acc: 0.9838 - val_loss: 0.1762 - val_acc: 0.9474
Epoch 137/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.04
29 - acc: 0.9823 - val_loss: 0.1675 - val_acc: 0.9454
Epoch 138/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.04
23 - acc: 0.9804 - val_loss: 0.1782 - val_acc: 0.9410
Epoch 139/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.04
36 - acc: 0.9818 - val_loss: 0.1713 - val_acc: 0.9450
Epoch 140/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.04
43 - acc: 0.9823 - val_loss: 0.1500 - val_acc: 0.9596
Epoch 141/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.04
04 - acc: 0.9844 - val_loss: 0.1755 - val_acc: 0.9444
Epoch 142/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.04
```

```
01 - acc: 0.9837 - val_loss: 0.1591 - val_acc: 0.9552
Epoch 143/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.03
98 - acc: 0.9839 - val_loss: 0.1691 - val_acc: 0.9505
Epoch 144/300
7352/7352 [==============================] - 1s 123us/step - loss: 0.03
85 - acc: 0.9853 - val_loss: 0.1688 - val_acc: 0.9498
Epoch 145/300
7352/7352 [==============================] - 1s 128us/step - loss: 0.03
99 - acc: 0.9834 - val_loss: 0.1606 - val_acc: 0.9589
Epoch 146/300
7352/7352 [==============================] - 1s 128us/step - loss: 0.03
91 - acc: 0.9837 - val_loss: 0.1628 - val_acc: 0.9572
Epoch 147/300
7352/7352 [==============================] - 1s 128us/step - loss: 0.04
27 - acc: 0.9819 - val_loss: 0.1795 - val_acc: 0.9413
Epoch 148/300
7352/7352 [==============================] - 1s 129us/step - loss: 0.04
07 - acc: 0.9826 - val_loss: 0.1626 - val_acc: 0.9471
Epoch 149/300
7352/7352 [==============================] - 1s 130us/step - loss: 0.03
87 - acc: 0.9834 - val_loss: 0.1812 - val_acc: 0.9423
Epoch 150/300
7352/7352 [==============================] - 1s 130us/step - loss: 0.03
98 - acc: 0.9833 - val_loss: 0.1361 - val_acc: 0.9684
Epoch 151/300
7352/7352 [==============================] - 1s 129us/step - loss: 0.04
34 - acc: 0.9823 - val_loss: 0.1408 - val_acc: 0.9650
Epoch 152/300
7352/7352 [==============================] - 1s 130us/step - loss: 0.04
35 - acc: 0.9841 - val_loss: 0.1928 - val_acc: 0.9457
Epoch 153/300
7352/7352 [==============================] - 1s 127us/step - loss: 0.03
86 - acc: 0.9837 - val_loss: 0.1497 - val_acc: 0.9528
Epoch 154/300
7352/7352 [==============================] - 1s 130us/step - loss: 0.04
34 - acc: 0.9814 - val_loss: 0.1892 - val_acc: 0.9413
Epoch 155/300
7352/7352 [==============================] - 1s 126us/step - loss: 0.03
```

```
80 - acc: 0.9844 - val_loss: 0.1413 - val_acc: 0.9532
Epoch 156/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.04
09 - acc: 0.9822 - val_loss: 0.1714 - val_acc: 0.9477
Epoch 157/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.04
12 - acc: 0.9826 - val_loss: 0.1800 - val_acc: 0.9416
Epoch 158/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.04
00 - acc: 0.9839 - val_loss: 0.1410 - val_acc: 0.9552
Epoch 159/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.03
98 - acc: 0.9829 - val_loss: 0.1648 - val_acc: 0.9511
Epoch 160/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.04
19 - acc: 0.9830 - val_loss: 0.1419 - val_acc: 0.9589
Epoch 161/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.03
90 - acc: 0.9856 - val_loss: 0.1490 - val_acc: 0.9545
Epoch 162/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.04
22 - acc: 0.9819 - val_loss: 0.1691 - val_acc: 0.9433
Epoch 163/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.03
64 - acc: 0.9846 - val_loss: 0.1446 - val_acc: 0.9525
Epoch 164/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.03
89 - acc: 0.9838 - val_loss: 0.1665 - val_acc: 0.9532
Epoch 165/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.04
00 - acc: 0.9835 - val_loss: 0.1467 - val_acc: 0.9505
Epoch 166/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.03
77 - acc: 0.9848 - val_loss: 0.1799 - val_acc: 0.9508
Epoch 167/300
7352/7352 [==============================] - 1s 118us/step - loss: 0.03
45 - acc: 0.9867 - val_loss: 0.1827 - val_acc: 0.9460
Epoch 168/300
7352/7352 [==============================] - 1s 117us/step - loss: 0.03
```

```
57 - acc: 0.9849 - val_loss: 0.1955 - val_acc: 0.9437
Epoch 169/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.04
19 - acc: 0.9823 - val_loss: 0.1349 - val_acc: 0.9661
Epoch 170/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.03
57 - acc: 0.9837 - val_loss: 0.1567 - val_acc: 0.9539
Epoch 171/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.03
44 - acc: 0.9856 - val_loss: 0.1440 - val_acc: 0.9555
Epoch 172/300
7352/7352 [==============================] - 1s 118us/step - loss: 0.03
65 - acc: 0.9859 - val_loss: 0.1647 - val_acc: 0.9511
Epoch 173/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.03
05 - acc: 0.9876 - val_loss: 0.1496 - val_acc: 0.9545
Epoch 174/300
7352/7352 [==============================] - 1s 118us/step - loss: 0.03
06 - acc: 0.9882 - val_loss: 0.1655 - val_acc: 0.9491
Epoch 175/300
7352/7352 [==============================] - 1s 117us/step - loss: 0.03
19 - acc: 0.9863 - val_loss: 0.1490 - val_acc: 0.9569
Epoch 176/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.03
24 - acc: 0.9876 - val_loss: 0.1596 - val_acc: 0.9545
Epoch 177/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.03
33 - acc: 0.9875 - val_loss: 0.1750 - val_acc: 0.9484
Epoch 178/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.03
33 - acc: 0.9869 - val_loss: 0.1745 - val_acc: 0.9471
Epoch 179/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.03
18 - acc: 0.9871 - val_loss: 0.1494 - val_acc: 0.9559
Epoch 180/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.03
38 - acc: 0.9846 - val_loss: 0.1533 - val_acc: 0.9542
Epoch 181/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.03
```

```
44 - acc: 0.9856 - val_loss: 0.1579 - val_acc: 0.9562
Epoch 182/300
7352/7352 [==============================] - 1s 117us/step - loss: 0.03
09 - acc: 0.9883 - val_loss: 0.1628 - val_acc: 0.9569
Epoch 183/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.03
00 - acc: 0.9879 - val_loss: 0.1408 - val_acc: 0.9637
Epoch 184/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.03
29 - acc: 0.9865 - val_loss: 0.1618 - val_acc: 0.9552
Epoch 185/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.03
02 - acc: 0.9884 - val_loss: 0.1454 - val_acc: 0.9630
Epoch 186/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.03
25 - acc: 0.9880 - val_loss: 0.2003 - val_acc: 0.9437
Epoch 187/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.03
47 - acc: 0.9857 - val_loss: 0.1314 - val_acc: 0.9647
Epoch 188/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.03
12 - acc: 0.9882 - val_loss: 0.1445 - val_acc: 0.9562
Epoch 189/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.03
14 - acc: 0.9883 - val_loss: 0.1448 - val_acc: 0.9576
Epoch 190/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.03
35 - acc: 0.9861 - val_loss: 0.1484 - val_acc: 0.9623
Epoch 191/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.03
32 - acc: 0.9867 - val_loss: 0.1533 - val_acc: 0.9555
Epoch 192/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.02
99 - acc: 0.9874 - val_loss: 0.1532 - val_acc: 0.9603
Epoch 193/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.03
24 - acc: 0.9869 - val_loss: 0.1917 - val_acc: 0.9444
Epoch 194/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.02
```

```
96 - acc: 0.9874 - val_loss: 0.1574 - val_acc: 0.9593
Epoch 195/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.03
30 - acc: 0.9878 - val_loss: 0.1492 - val_acc: 0.9555
Epoch 196/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.02
99 - acc: 0.9880 - val_loss: 0.1975 - val_acc: 0.9437
Epoch 197/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.03
21 - acc: 0.9867 - val_loss: 0.1507 - val_acc: 0.9589
Epoch 198/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.03
05 - acc: 0.9883 - val_loss: 0.1301 - val_acc: 0.9681
Epoch 199/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.03
04 - acc: 0.9886 - val_loss: 0.1579 - val_acc: 0.9569
Epoch 200/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.02
88 - acc: 0.9887 - val_loss: 0.1552 - val_acc: 0.9559
Epoch 201/300
7352/7352 [==============================] - 1s 123us/step - loss: 0.02
76 - acc: 0.9902 - val_loss: 0.1434 - val_acc: 0.9617
Epoch 202/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.02
67 - acc: 0.9894 - val_loss: 0.1489 - val_acc: 0.9566
Epoch 203/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.02
78 - acc: 0.9891 - val_loss: 0.1476 - val_acc: 0.9600
Epoch 204/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.02
83 - acc: 0.9888 - val_loss: 0.1457 - val_acc: 0.9606
Epoch 205/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.02
91 - acc: 0.9872 - val_loss: 0.1512 - val_acc: 0.9589
Epoch 206/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.02
91 - acc: 0.9872 - val_loss: 0.1566 - val_acc: 0.9562
Epoch 207/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.03
```

```
18 - acc: 0.9883 - val_loss: 0.1529 - val_acc: 0.9549
Epoch 208/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.02
56 - acc: 0.9901 - val_loss: 0.1348 - val_acc: 0.9695
Epoch 209/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.02
68 - acc: 0.9887 - val_loss: 0.1641 - val_acc: 0.9549
Epoch 210/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.02
64 - acc: 0.9908 - val_loss: 0.1391 - val_acc: 0.9654
Epoch 211/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.02
61 - acc: 0.9895 - val_loss: 0.1905 - val_acc: 0.9481
Epoch 212/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.02
80 - acc: 0.9891 - val_loss: 0.1465 - val_acc: 0.9623
Epoch 213/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.02
89 - acc: 0.9874 - val_loss: 0.2022 - val_acc: 0.9471
Epoch 214/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.02
96 - acc: 0.9883 - val_loss: 0.1316 - val_acc: 0.9671
Epoch 215/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.02
64 - acc: 0.9898 - val_loss: 0.1407 - val_acc: 0.9627
Epoch 216/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.03
34 - acc: 0.9879 - val_loss: 0.1389 - val_acc: 0.9644
Epoch 217/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.02
53 - acc: 0.9906 - val_loss: 0.1471 - val_acc: 0.9583
Epoch 218/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.02
70 - acc: 0.9895 - val_loss: 0.1554 - val_acc: 0.9569
Epoch 219/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.02
69 - acc: 0.9891 - val_loss: 0.1884 - val_acc: 0.9498
Epoch 220/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.02
```

```
54 - acc: 0.9905 - val_loss: 0.2170 - val_acc: 0.9433
Epoch 221/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.03
28 - acc: 0.9865 - val_loss: 0.1481 - val_acc: 0.9617
Epoch 222/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.03
16 - acc: 0.9887 - val_loss: 0.1462 - val_acc: 0.9606
Epoch 223/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.02
32 - acc: 0.9910 - val_loss: 0.1544 - val_acc: 0.9562
Epoch 224/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.02
27 - acc: 0.9920 - val_loss: 0.1722 - val_acc: 0.9542
Epoch 225/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.02
57 - acc: 0.9898 - val_loss: 0.1466 - val_acc: 0.9593
Epoch 226/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.02
29 - acc: 0.9897 - val_loss: 0.1970 - val_acc: 0.9494
Epoch 227/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.03
02 - acc: 0.9888 - val_loss: 0.2200 - val_acc: 0.9444
Epoch 228/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.02
16 - acc: 0.9908 - val_loss: 0.1580 - val_acc: 0.9603
Epoch 229/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.02
72 - acc: 0.9893 - val_loss: 0.1449 - val_acc: 0.9613
Epoch 230/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.02
76 - acc: 0.9880 - val_loss: 0.1358 - val_acc: 0.9623
Epoch 231/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.02
60 - acc: 0.9893 - val_loss: 0.2007 - val_acc: 0.9464
Epoch 232/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.02
50 - acc: 0.9912 - val_loss: 0.1836 - val_acc: 0.9511
Epoch 233/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.02
```

```
58 - acc: 0.9902 - val_loss: 0.1514 - val_acc: 0.9617
Epoch 234/300
7352/7352 [==============================] - 1s 118us/step - loss: 0.02
54 - acc: 0.9908 - val_loss: 0.1947 - val_acc: 0.9477
Epoch 235/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.02
41 - acc: 0.9899 - val_loss: 0.1514 - val_acc: 0.9613
Epoch 236/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.02
45 - acc: 0.9898 - val_loss: 0.1556 - val_acc: 0.9620
Epoch 237/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.02
46 - acc: 0.9901 - val_loss: 0.1289 - val_acc: 0.9678
Epoch 238/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.02
24 - acc: 0.9918 - val_loss: 0.1678 - val_acc: 0.9549
Epoch 239/300
7352/7352 [==============================] - 1s 118us/step - loss: 0.02
07 - acc: 0.9918 - val_loss: 0.1581 - val_acc: 0.9593
Epoch 240/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.02
02 - acc: 0.9924 - val_loss: 0.1624 - val_acc: 0.9593
Epoch 241/300
7352/7352 [==============================] - 1s 116us/step - loss: 0.01
93 - acc: 0.9918 - val_loss: 0.1576 - val_acc: 0.9610
Epoch 242/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.01
96 - acc: 0.9929 - val_loss: 0.1844 - val_acc: 0.9545
Epoch 243/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.02
24 - acc: 0.9910 - val_loss: 0.1409 - val_acc: 0.9637
Epoch 244/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.02
07 - acc: 0.9921 - val_loss: 0.1601 - val_acc: 0.9572
Epoch 245/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.02
34 - acc: 0.9908 - val_loss: 0.1599 - val_acc: 0.9586
Epoch 246/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.01
```

```
89 - acc: 0.9924 - val_loss: 0.1617 - val_acc: 0.9583
Epoch 247/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.01
92 - acc: 0.9936 - val_loss: 0.2374 - val_acc: 0.9444
Epoch 248/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.02
32 - acc: 0.9909 - val_loss: 0.1369 - val_acc: 0.9667
Epoch 249/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.02
01 - acc: 0.9916 - val_loss: 0.1641 - val_acc: 0.9579
Epoch 250/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.02
16 - acc: 0.9913 - val_loss: 0.1720 - val_acc: 0.9562
Epoch 251/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.02
01 - acc: 0.9928 - val_loss: 0.1703 - val_acc: 0.9603
Epoch 252/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.02
19 - acc: 0.9912 - val_loss: 0.1875 - val_acc: 0.9555
Epoch 253/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.02
22 - acc: 0.9918 - val_loss: 0.1905 - val_acc: 0.9508
Epoch 254/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.01
93 - acc: 0.9924 - val_loss: 0.2275 - val_acc: 0.9464
Epoch 255/300
7352/7352 [==============================] - 1s 118us/step - loss: 0.02
05 - acc: 0.9927 - val_loss: 0.1790 - val_acc: 0.9562
Epoch 256/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.02
11 - acc: 0.9920 - val_loss: 0.1437 - val_acc: 0.9637
Epoch 257/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.02
21 - acc: 0.9916 - val_loss: 0.1537 - val_acc: 0.9661
Epoch 258/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.02
20 - acc: 0.9913 - val_loss: 0.1390 - val_acc: 0.9650
Epoch 259/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.02
```

```
06 - acc: 0.9912 - val_loss: 0.1675 - val_acc: 0.9562
Epoch 260/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.02
00 - acc: 0.9932 - val_loss: 0.1735 - val_acc: 0.9600
Epoch 261/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.02
78 - acc: 0.9893 - val_loss: 0.1580 - val_acc: 0.9623
Epoch 262/300
7352/7352 [==============================] - 1s 117us/step - loss: 0.01
92 - acc: 0.9928 - val_loss: 0.1700 - val_acc: 0.9566
Epoch 263/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.02
02 - acc: 0.9918 - val_loss: 0.1703 - val_acc: 0.9617
Epoch 264/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.01
69 - acc: 0.9931 - val_loss: 0.1742 - val_acc: 0.9579
Epoch 265/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.02
05 - acc: 0.9918 - val_loss: 0.1898 - val_acc: 0.9528
Epoch 266/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.01
65 - acc: 0.9937 - val_loss: 0.1540 - val_acc: 0.9620
Epoch 267/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.01
90 - acc: 0.9932 - val_loss: 0.1674 - val_acc: 0.9603
Epoch 268/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.02
28 - acc: 0.9901 - val_loss: 0.2336 - val_acc: 0.9454
Epoch 269/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.02
08 - acc: 0.9921 - val_loss: 0.1613 - val_acc: 0.9600
Epoch 270/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.01
99 - acc: 0.9921 - val_loss: 0.2037 - val_acc: 0.9474
Epoch 271/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.02
01 - acc: 0.9920 - val_loss: 0.1571 - val_acc: 0.9596
Epoch 272/300
7352/7352 [==============================] - 1s 124us/step - loss: 0.01
```

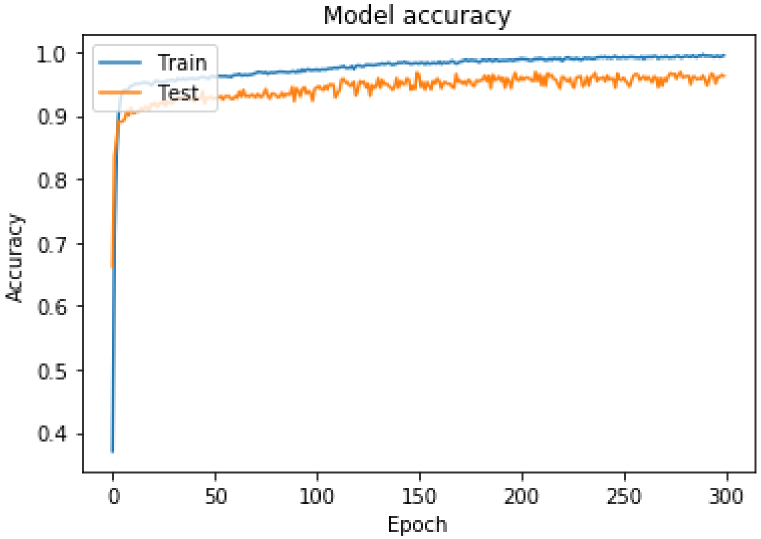
```
69 - acc: 0.9935 - val_loss: 0.2013 - val_acc: 0.9545
Epoch 273/300
7352/7352 [==============================] - 1s 124us/step - loss: 0.02
04 - acc: 0.9916 - val_loss: 0.1543 - val_acc: 0.9627
Epoch 274/300
7352/7352 [==============================] - 1s 125us/step - loss: 0.02
13 - acc: 0.9914 - val_loss: 0.1377 - val_acc: 0.9674
Epoch 275/300
7352/7352 [==============================] - 1s 127us/step - loss: 0.02
13 - acc: 0.9924 - val_loss: 0.1419 - val_acc: 0.9654
Epoch 276/300
7352/7352 [==============================] - 1s 126us/step - loss: 0.01
94 - acc: 0.9917 - val_loss: 0.1406 - val_acc: 0.9664
Epoch 277/300
7352/7352 [==============================] - 1s 124us/step - loss: 0.01
58 - acc: 0.9939 - val_loss: 0.1931 - val_acc: 0.9586
Epoch 278/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.01
67 - acc: 0.9935 - val_loss: 0.1506 - val_acc: 0.9654
Epoch 279/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.02
02 - acc: 0.9917 - val_loss: 0.1365 - val_acc: 0.9691
Epoch 280/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.01
90 - acc: 0.9920 - val_loss: 0.1658 - val_acc: 0.9606
Epoch 281/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.01
78 - acc: 0.9933 - val_loss: 0.1570 - val_acc: 0.9620
Epoch 282/300
7352/7352 [==============================] - 1s 119us/step - loss: 0.01
62 - acc: 0.9940 - val_loss: 0.1546 - val_acc: 0.9627
Epoch 283/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.01
75 - acc: 0.9929 - val_loss: 0.1720 - val_acc: 0.9627
Epoch 284/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.01
73 - acc: 0.9927 - val_loss: 0.2060 - val_acc: 0.9498
Epoch 285/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.01
```

```
54 - acc: 0.9947 - val_loss: 0.1514 - val_acc: 0.9617
Epoch 286/300
7352/7352 [==============================] - 1s 124us/step - loss: 0.01
66 - acc: 0.9928 - val_loss: 0.1656 - val_acc: 0.9617
Epoch 287/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.02
07 - acc: 0.9920 - val_loss: 0.1736 - val_acc: 0.9576
Epoch 288/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.01
55 - acc: 0.9947 - val_loss: 0.1548 - val_acc: 0.9589
Epoch 289/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.01
62 - acc: 0.9935 - val_loss: 0.1592 - val_acc: 0.9579
Epoch 290/300
7352/7352 [==============================] - 1s 120us/step - loss: 0.01
25 - acc: 0.9962 - val_loss: 0.1570 - val_acc: 0.9637
Epoch 291/300
7352/7352 [==============================] - 1s 124us/step - loss: 0.01
74 - acc: 0.9937 - val_loss: 0.1811 - val_acc: 0.9569
Epoch 292/300
7352/7352 [==============================] - 1s 123us/step - loss: 0.01
54 - acc: 0.9944 - val_loss: 0.2292 - val_acc: 0.9481
Epoch 293/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.01
59 - acc: 0.9939 - val_loss: 0.1810 - val_acc: 0.9572
Epoch 294/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.01
54 - acc: 0.9940 - val_loss: 0.1417 - val_acc: 0.9671
Epoch 295/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.01
69 - acc: 0.9924 - val_loss: 0.1630 - val_acc: 0.9634
Epoch 296/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.01
82 - acc: 0.9931 - val_loss: 0.2041 - val_acc: 0.9511
Epoch 297/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.01
55 - acc: 0.9943 - val_loss: 0.1862 - val_acc: 0.9610
Epoch 298/300
7352/7352 [==============================] - 1s 123us/step - loss: 0.01
```

```
84 - acc: 0.9922 - val_loss: 0.1728 - val_acc: 0.9613
Epoch 299/300
7352/7352 [==============================] - 1s 121us/step - loss: 0.01
50 - acc: 0.9933 - val_loss: 0.1495 - val_acc: 0.9647
Epoch 300/300
7352/7352 [==============================] - 1s 122us/step - loss: 0.01
43 - acc: 0.9947 - val_loss: 0.1655 - val_acc: 0.9630
```

In [0]:

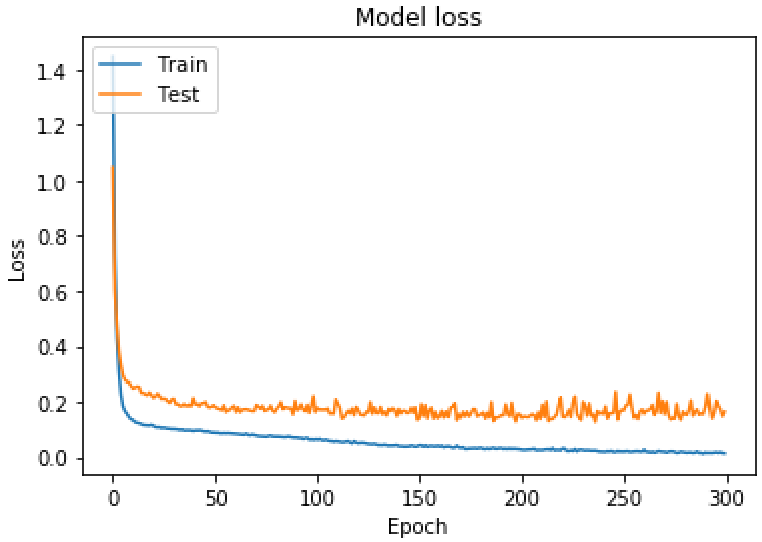
```python
import matplotlib.pyplot as plt

# Plot training & validation accuracy values
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```



In [0]:

```python
# Plot training & validation loss values
plt.plot(history.history['loss'])
```

```
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```



In [0]:

```
# Confusion Matrix
# print(confusion_matrix(Y_test, model.predict(X_test)))
conf = pd.DataFrame(confusion_matrix(Y_test, model.predict(X_test)))
conf
```

Out[0]:

| **Pred** | **LAYING** | **SITTING** | **STANDING** | **WALKING** | **WALKING_DOWNSTA** |
|---|---|---|---|---|---|
| **True** | | | | | |
| **LAYING** | 537 | 0 | 0 | 0 | 0 |
| **SITTING** | 0 | 438 | 53 | 0 | 0 |
| **STANDING** | 0 | 9 | 523 | 0 | 0 |

| Pred | LAYING | SITTING | STANDING | WALKING | WALKING_DOWNSTA |
|---|---|---|---|---|---|
| **True** | | | | | |
| **WALKING** | 0 | 3 | 0 | 480 | 13 |
| **WALKING_DOWNSTAIRS** | 0 | 0 | 0 | 0 | 405 |
| **WALKING_UPSTAIRS** | 0 | 2 | 0 | 0 | 14 |

In [0]:

```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 0s 79us/step
```

In [0]:

```
score
```

Out[0]:

```
[0.1655390887501719, 0.9630132337970818]
```

In [0]:

```
score_train = model.evaluate(X_train, Y_train)
score_train
```

```
7352/7352 [==============================] - 1s 74us/step
```

Out[0]:

```
[0.008311180752592058, 0.9970076169749728]
```

## 7. Comparing all ML models

In [0]:

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

print('Linear SVC          : {:.04}%       {:.04}% '.format(lr_svc_grid
_results['accuracy'] * 100,\
                                                        100-(lr_svc_gri
```

```
d_results['accuracy'] * 100)))

print('rbf SVM classifier  : {:.04}%      {:.04}% '.format(rbf_svm_grid
_results['accuracy'] * 100,\
                                                             100-(rbf_svm_
grid_results['accuracy'] * 100)))

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

print('Random Forest       : {:.04}%      {:.04}% '.format(rfc_grid_res
ults['accuracy'] * 100,\
                                                              100-(rfc_gri
d_results['accuracy'] * 100)))
print('GradientBoosting DT : {:.04}%      {:.04}% '.format(rfc_grid_res
ults['accuracy'] * 100,\
                                                           100-(rfc_grid_r
esults['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%
```

> We can choose ***Logistic regression*** or ***Linear SVC*** or ***rbf SVM***.

# Conclusion :

This project is to build a model that predicts the human activities such as Walking, Walking_Upstairs, Walking_Downstairs, Sitting, Standing or Laying.

These sensor signals are preprocessed by applying noise filters and then sampled in fixed-width windows(sliding windows) of 2.56 seconds each with 50% overlap. ie., each window has 128 readings.

- Jerk signals are calculated for BodyAcceleration readings.
- Fourier Transforms are made on the above time readings to obtain frequency readings.
- Now, on all the base signal readings., mean, max, mad, sma, arcoefficient, engerybands,entropy etc., are calculated for each window.
- We get a feature vector of 561 features and these features are given in the dataset.
- Each window of readings is a datapoint of 561 features.

Splited the data

Removed dublicates

Removed nan values

This is balanced dataset

Performed EDA

```
Stationary and Moving activities are completely different

Magnitude of an acceleration can saperate it well

Position of GravityAccelerationComponants also matters
```

Apply t-sne on the data

Let's model with our data

```
Linear SVC with GridSearch
```

```
Logistic Regression

Gradient Boosted

Decision Trees With GridSearch

Random Forest Classifier with GridSearch

Decision Trees with GridSearchCV

Kernel SVM with GridSearch
```

Deep learning

```
LSTM for n_hidden layer-30
-  0.8934509670851714

LSTM for n_hidden layer-50
-  0.9277231082456736

LSTM for n_hidden layer-50 using bias_initializer='zeros'
-  0.9209365456396336

LSTM for 2-layers
-  0.9141499830335935

LSTM for n_hidden layer-50 using glorot_initializer-
-  0.909365456396336

CNN  with 300 units and kernal size and max-pool-15=
  -0.9433322022395657
```

```
   2 layers of  CNN  with 300 units and kernal size and max-pool
-15= -0.9192399049881235

   CNN  +  LSTM-
     0.9277231082456736

   2 CNN layers + 3 Pooling layers + one Dropout layer
     0.9630132337970818
```