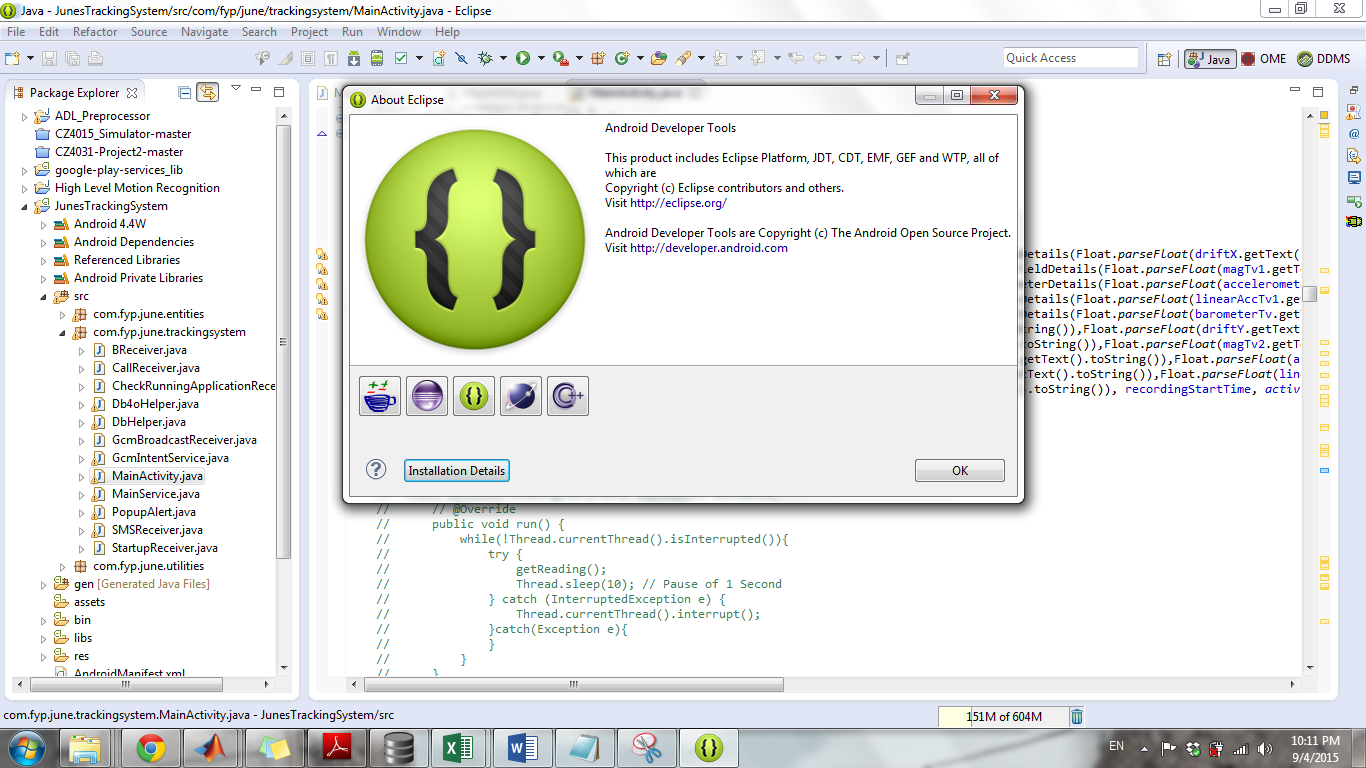
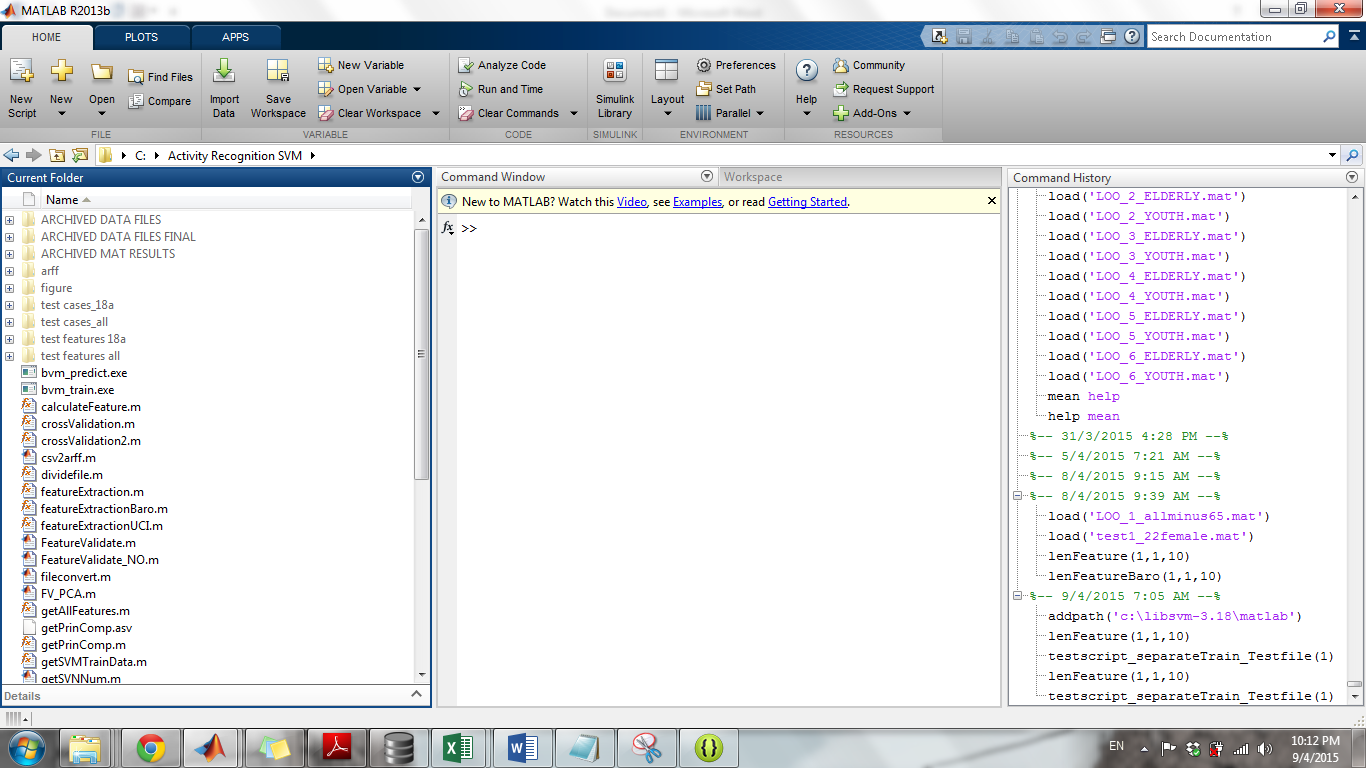
# Set Up Instructions

* You will need these two IDEs:
  + **Eclipse with android SDK**
  + **Matlab 2013 and above**
  + Remember to install **libsvm** correctly (requires compilation- refer to WangDi for help on that)

1. Data Collection Android App
   1. Copy-paste the **TrackingSystem\_SQLITE** folder into the Eclipse workbench directory
   2. Find the apk file and transfer it to your phone for download
   3. OR you can run and emulate it on ur phone through eclipse
2. Java Pre-processing component
   1. Copy-past the **ADL\_preprocessor** folder into the Eclipse workbench
3. Matlab SVM model
   1. Copy paste the **Activity Recognition SVM** folder into C:/ drive
   2. Run Matlab and change the path to C:/Activity Recognition SVM
   3. Make sure **libSVM** is installed to C:/ drive
   4. Run this command “addpath('c:\libsvm-3.18\matlab')” before starting to execute anything in Matlab
4. Java High level motion recognition processor
   1. Copy paste the **High Level Motion Recognition** folder into eclipse workbench directory

# Data Collection Using Android App

Activity Tag

Data collection period in milliseconds

Button to press when start recording

1. Key in data collection period
2. Key in Activity Tag (used in this project)

1 Lying

2 Sit

3 Stand

4 Walk

5 Run

6.1 Fall Down- stand

6.2 Fall Down- sit

6.3 Fall Down- walk

6.4 Fall Down- run

7 Sit-Lying

8 Lying-Sit

9 Sit-Stand

10 Stand-Sit

11 Walk-Sit

12 Sit-Walk

13 Walk-Lying

14 Lying-Walk

15 Walk-Stand

16 Stand-Walk

17 Stand-Lying

18 Lying-Stand

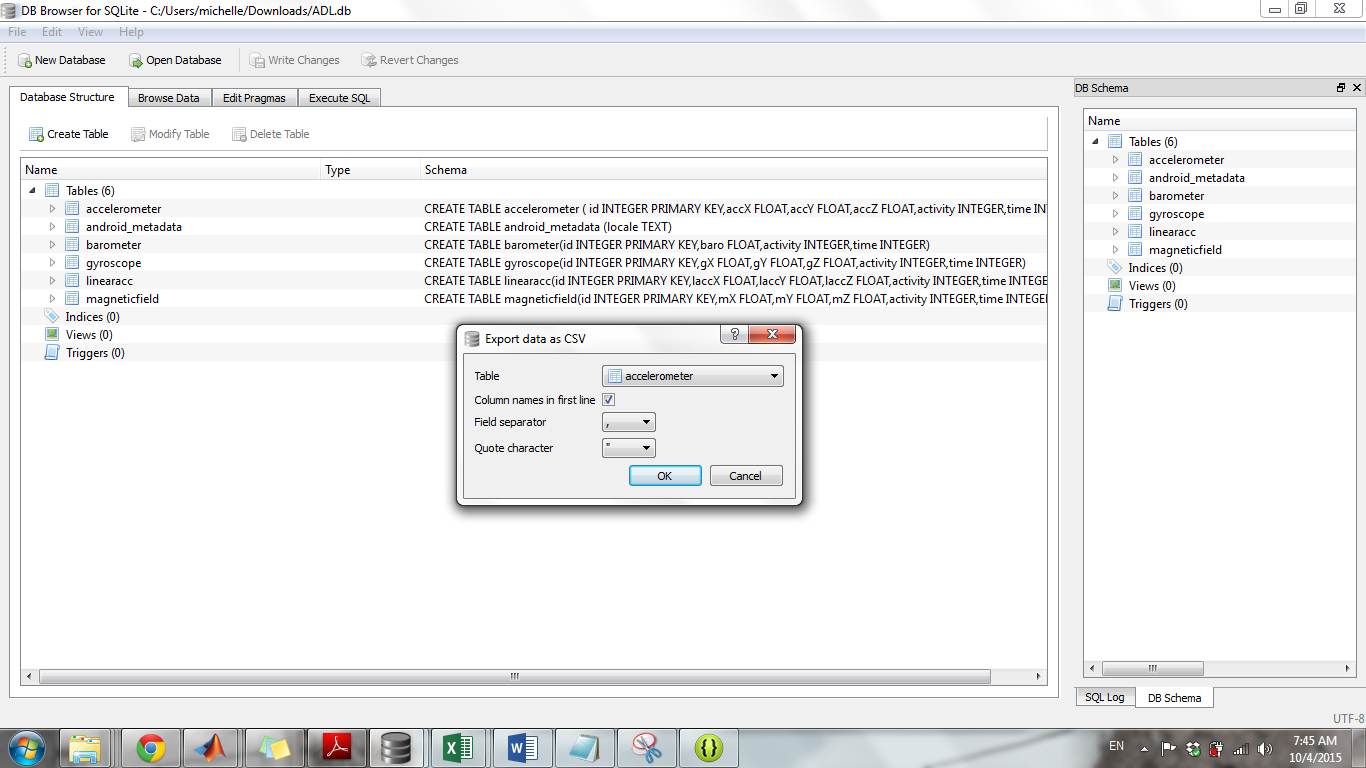
19 Stand-Run

20 Run-Stand

21 Walk-Run

22 Run-Walk

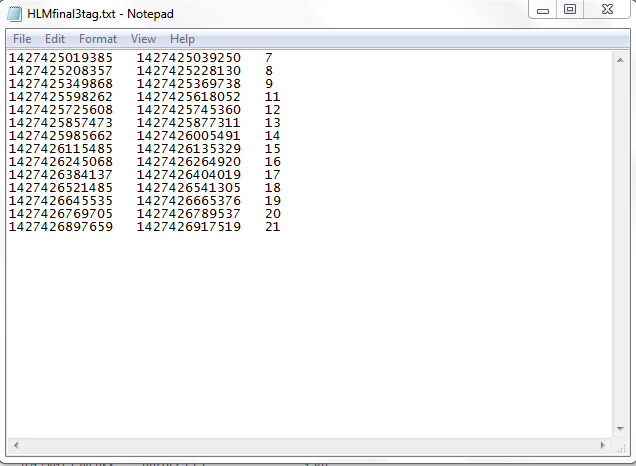
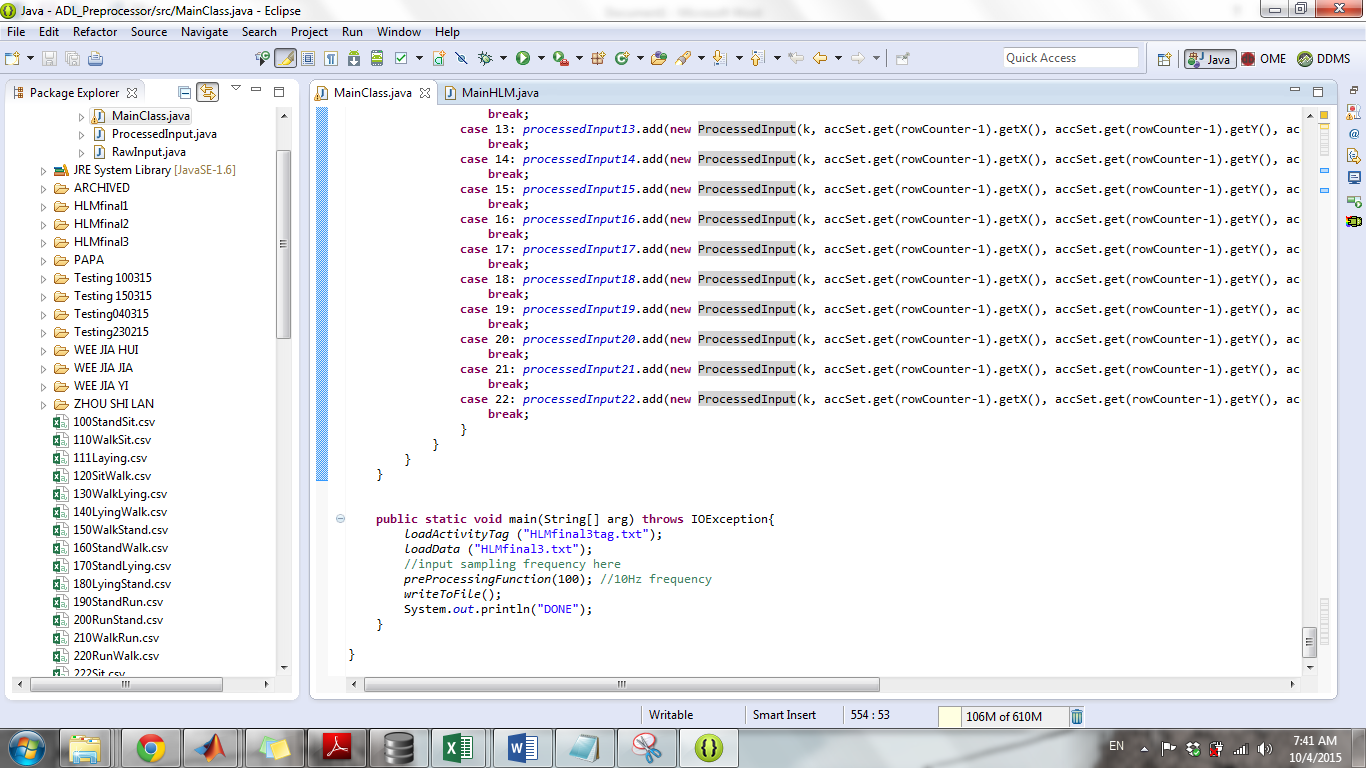
* + Note that other benchmark dataset uses different labelling (information available in dataset folder)

1. Press “Start Recording” button
2. Sound Signal #1: indicate that button press has been received (how many seconds after…)
3. Sound Signal #2: start performing activity
4. Sound Signal #3: actual recording started
5. Sound Signal #4: recording stopped
6. Locate the app database:
   * Data>Data>com.fyp.june.trackingsystem>databases>ADL.db
   * Copy-paste it to storage>extSDCard (from here when you plug in ur phone to the computer you can see the file in the “Card” folder.
   * Note: if you are using a Samsung phone, you need to root your phone before you can do this. HTC phone can work without rooting.
7. Open the db file using any SQLite browser
8. Export the accelerometer and barometer readings into 2 excel files

# Data Pre-processing Using Java Program

Ensure that the ADL\_Preprocessor folder is in Eclipse workbench directory

**Low Level Motion Recognition Processing**

1. Open the Accelerometer exported csv file
2. Using Excel’s filter function, manually create activity tag file:
   * Take the first and last timestamp of each action in the **Accelerometer file**
   * Create a text file and save it to the ADL\_Preprocessor main folder (which is residing in eclipse workbench)
3. Click “Save As”- To save Accelerometer csv file as a text file in the ADL Preprocessor folder
4. In the Java program, fill in the file names to process and set the sampling frequency in the main function

Key in tag file and main data file to be processed here

Set sampling frequency here

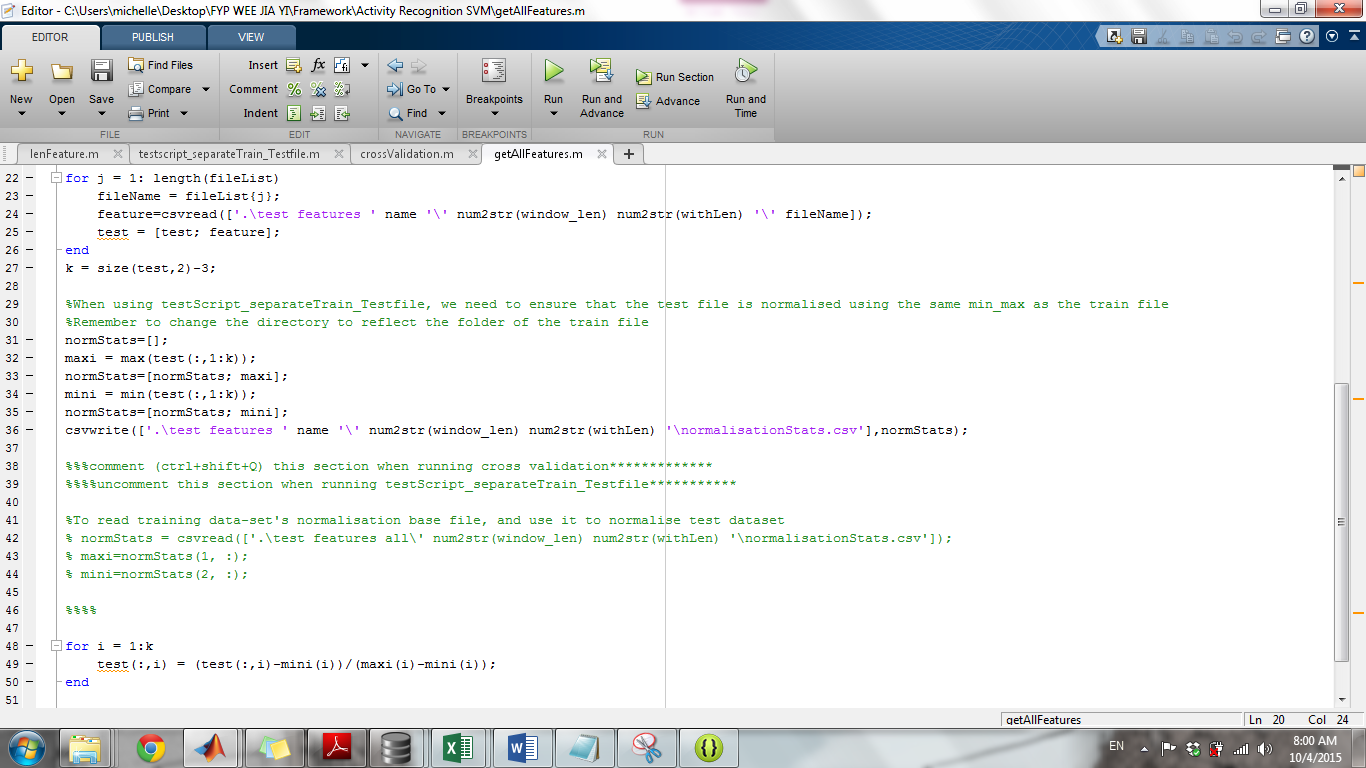
**High Level Motion Transition Recognition Processing**

1. Open **the Accelerometer exported csv file**
2. Using Excel’s filter function, manually create activity tag file:
   * Take the first and last timestamp of each action in the **Accelerometer file**
   * Create a text file and save it in the ADL\_Preprocessor main folder (which is residing in eclipse workbench)
3. Open the **Barometer exported csv file**
   * Check against the newly created activity tag file
   * Change the first and last timestamp of each activity to reflect what is in the tag file
4. Click “Save As”- To save Accelerometer and Barometer csv files as a text files in the ADL Preprocessor folder
5. Fill in the file names to process and set the sampling frequency in the main function

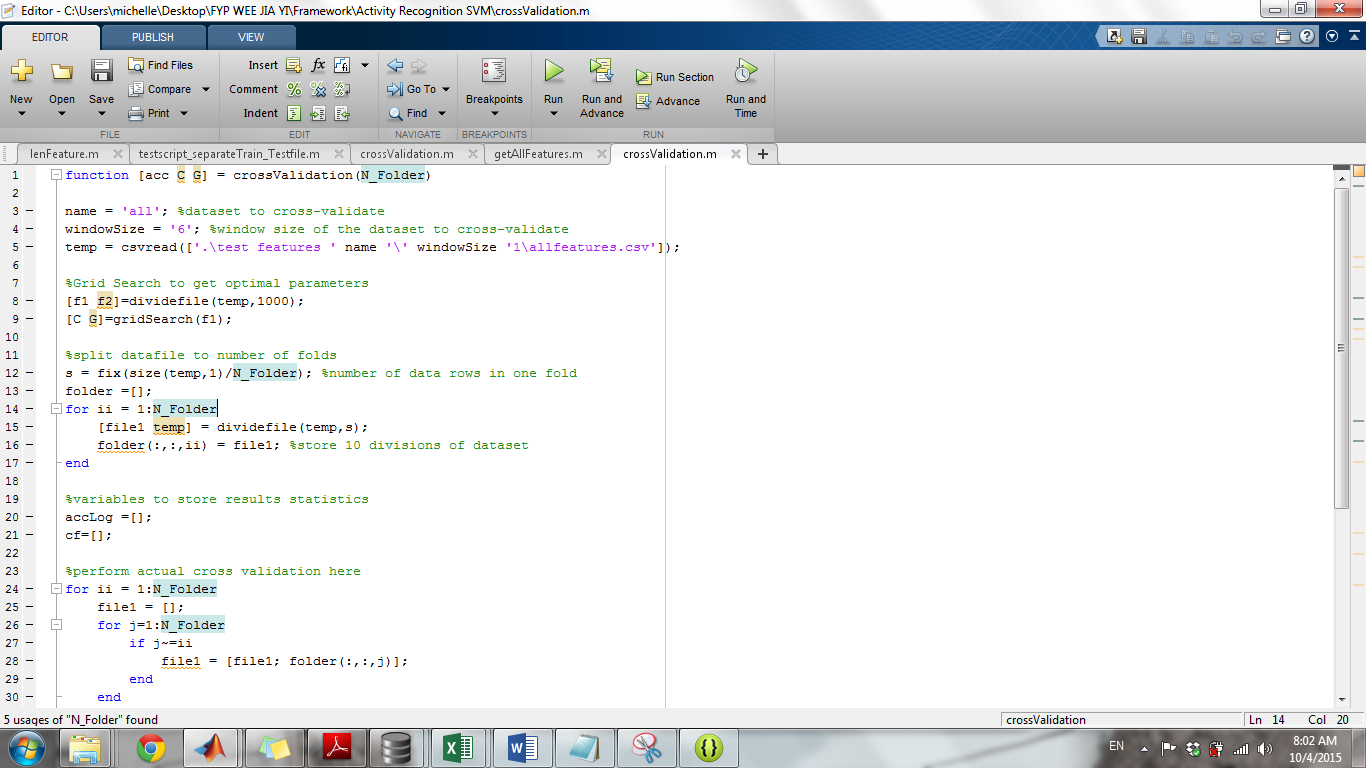
# Matlab SVM

* Remember to change Matlab working directory to the **Activity Recognition SVM folder**
* Addpath for libsvm

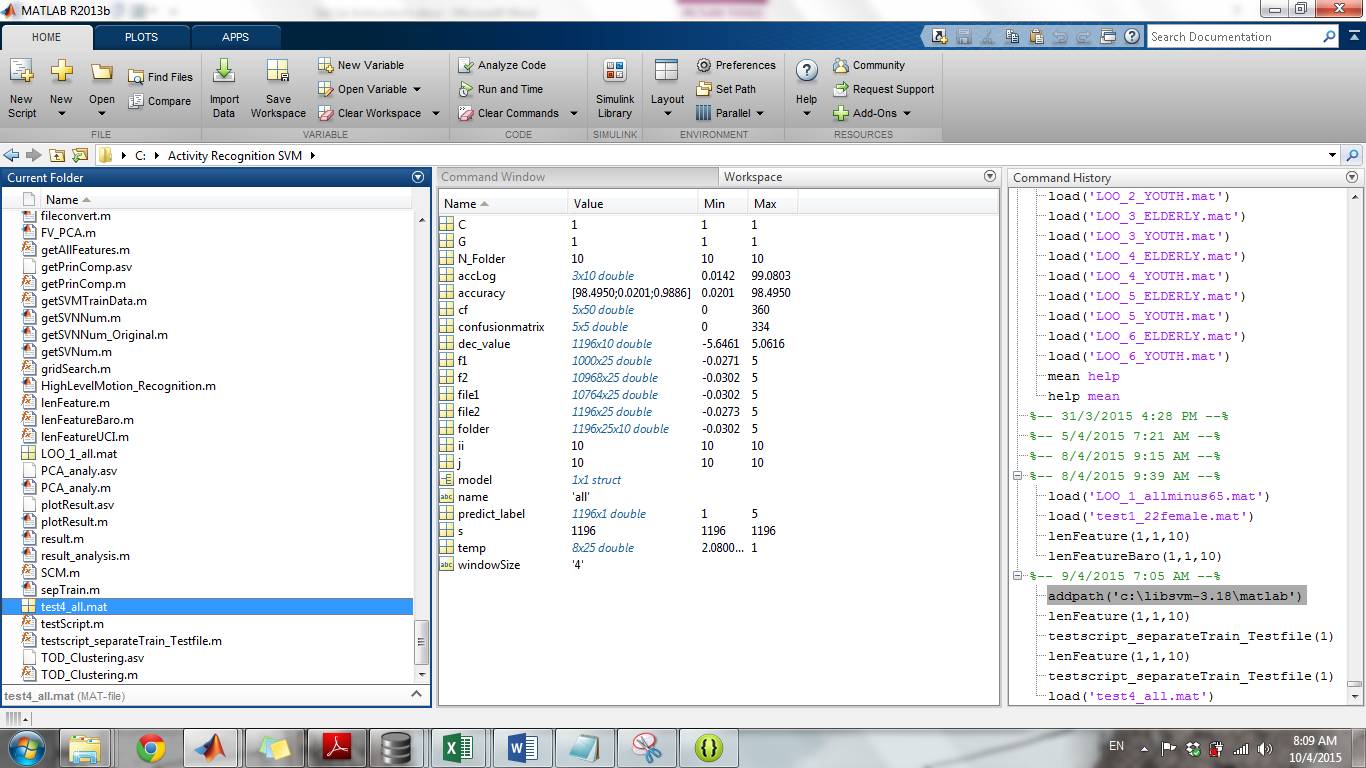
**Low Level Motion Recognition Testing**

1. Create a new folder “test cases\_XX” files into **Activity Recognition SVM folder**
2. Copy Paste pre-processed activity csv files into the new folder
   * Ensure that the csv files are properly formatted (text to columns)
3. **If dataset is a training data, comment the codes in the green area and uncomment the codes in the red area.**
4. **If dataset is testing data, uncomment the codes in the green area and comment the codes in the read area.**
   * Ensure that the normalisationStats.csv file is read from the correct training data folder (e.g test features all)
5. Run “***lenFeature(frame\_len, withLen, frequency)***” to extract the features
   * EX: If your pre-processing frequency is 10Hz, you need to extract features using 1 second frame size and you intend to use the magnitude dimension

>> lenFeature(1,1,10)

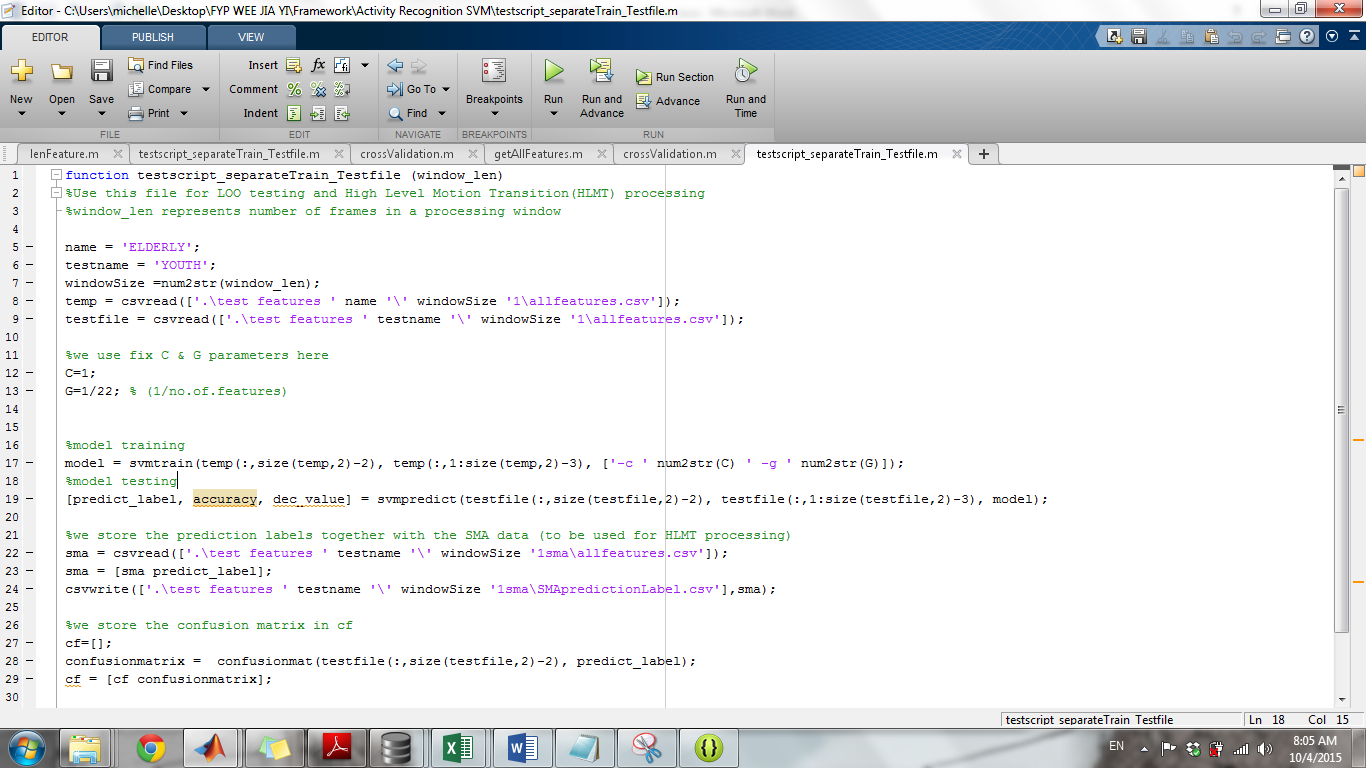
1. To do crossvalidation:
   * Run ***crossValidation(no. of folds)***

Set dataset name and frame size to cross validate

* + To obtain the results
    - Go the matlab workspace and locate the file “test(windowsize)\_dataset’s name”

Can get the 10 confusion matrices here

1. To do Leave One User/Group Out
   * Run ***testscript\_separateTrain\_Testfile (window\_len)***



Set the training dataset and testing dataset here

1. To get the test results:
   * Go to the Matlab workspace and find the file “LOO(training data’s name)\_(windowSize)\_(testing data’s name)”

**High Level Motion Transition Recognition**

1. Unlike for low level motion testing, pre-processed files has to be separated into individual folders. One for each motion transition.
2. Perform feature extraction as per mentioned above for acceleration file (e.g. test cases\_7a) by running lenFeature
3. Run ***testscript\_separateTrain\_Testfile (window\_len)***
   * This will generate a predictionLabel csv file
4. Perform feature extraction for barometer readings (e.g. test cases\_7b)
   * Run lenFeatureBarometer
   * Manually combine the barometer feature file (in test features \_**b** file ) and SMApredictionLabel file (in test features **\_a** file)
5. Save this file as a text file to the **High Level Motion Recognition** folder in eclipse workbench directory

# 3. High Level Motion Recognition

1. Ensure that the ADL\_Preprocessor folder is in Eclipse workbench directory
2. Re-Name and SaveAs the combined SMA, PredictionLabel and Barometer file as input.txt
3. You can change the value of the SMA and barometer threshold values in the Java program
   1. Do a search for “SMA threshold” or “Barometer threshold” to locate where the values can be adjusted
4. Change file name to process in the main function
5. After running the program,
   1. You should see the output in the command window
   2. Also there will be an output file to reflect the transition predictions