# Pattern Recognition Chain

Activity monitoring - Building a complete pipeline for sensor-based activity recognition

Activity Monitoring is one of the popular application fields of Ubiquitous Computing which detects a specific set of activities using sensors worn by a subject (person) and provides data in real time. It has multiple applications in the fields of surveillance, gaming, remote control, assistive living etc. In this report, we are going to present the working and results of a project done on activity recognition classification problems using the Cognitive Village (CogAge) dataset. The classifier used for this problem is Support Vector Machine (SVM) and the model is trained to recognize 55 behavioral activities, out of 61 activities available in CogAge dataset. Complete data of three devices, smartphone, smartwatch, and smart glasses, is used. Model is evaluated by two evaluation metrics: accuracy and average F1-score.

**Description of Implemented Pattern Recognition Chain (PRC):**

**Data Acquisition:**

Cognitive Village (CogAge) dataset is used that was fully checked when extracted.

**Pre-processing:**

Pre-processing techniques used on raw data: (in order to make it valuable for the model) Segmentation can be performed But I have Choose the Following below.

**1-** **Normalization:**

In this project normalization of data is done channel wise. Mean and standard deviation of each channel of all examples is taken, then mean is subtracted from each value of respective channel and divided by standard deviation of that channel.

**normal\_value = (x - mean) / std**

**2- Up-sampling:**

After normalization we adopted Up-sampling in pre-processing. Provided data was coming from different sensors of different frequency, the number of data values of different sensors were not the same i.e., Accelerometer (2284, 800, 3) and Magnetometer (2284, 200, 3). So, to make these values same we have to up sample the data of sensors that have lower frequency. Technique used here for up-sampling is interpolation.

**Segmentation:**

Segmentation was already done in the CogAge dataset provided.

**Feature Extraction:**

**These Features are Choose from the Document Provided.**

Following features are extracted to train the model:

1. Min
2. Max
3. Mean
4. Standard Deviation
5. Range
6. Median
7. Mode
8. First Mode
9. Variance
10. Energy
11. Dissimilarity

**Classification:**

**Classifier Used:**

Classifier used for training, in the space of features:

* SVM
* Logistic Regression ( Optional for Compariosn)

**Classifier Evaluation:**

Evaluation metrics used to evaluate model:

* Accuracy
* Average F1-score
* Confusion Matrix

**Results Obtained by Method:**

**For training data:**

* Maximum Accuracy achieved —------ 64.92 %

**For testing data:**

On setting following values of variables:

* window -size: 13
* Stride-size: 9
* Window-features: 7

Evaluation measures obtained in SVM:

* Accuracy —------ 22.9 %
* Average F1-score —------- 0.2168
* Training Accuracy : 0.649246

**Structure of Python Code:**

1. Python code is Written in Jupiter Note Book, and loading the data.

2. Performing normalization on data and storing in arrays.

3. Then the above 11 features are computed for each channel.

4. Then each sensor’s data is concatenated to make a complete example.

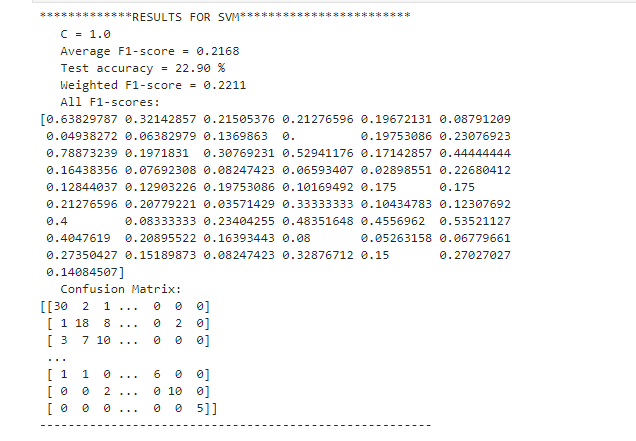
5. Then whole training data is given to SVM (built-in library function) to train the model.

6. At the end, accuracy and average F1-score are computed on testing data using confusion metrics.

**Note**: Step 2 to 4 are performed for both training and testing data.

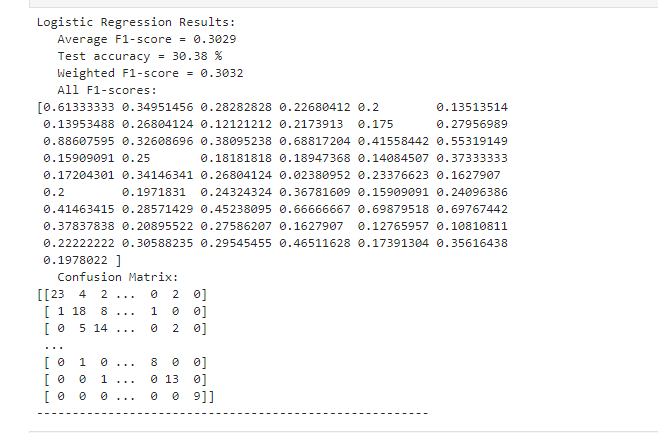
# Experiments:

**SVM vs LOGISTIC REGRESSION**



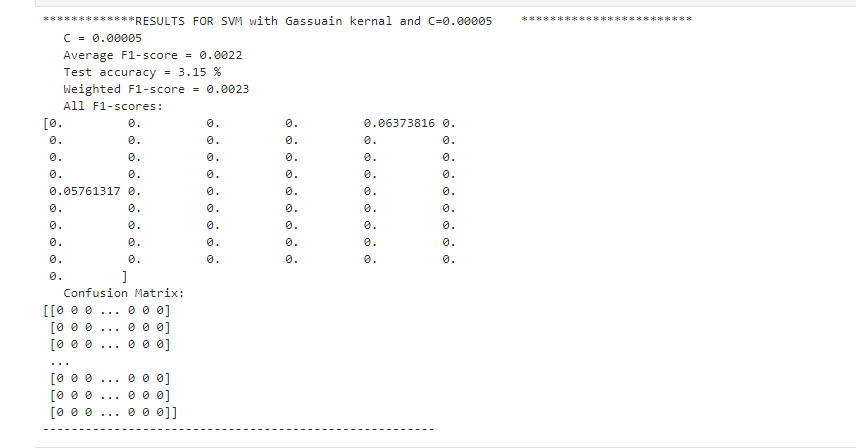
**For the Sake of Compariosn I have Also implemented The LOGISTIC REGRESSION MODEL to the normalized Training and Testing Data.**

* **We Performed the Feature Scaling on the testing and training examples and pass to the Logistic Regression Model and Results are Given Below.**



**EXPERIMENTS WITH C-VALUE AND KERNALS:**

* **C=0.0005 (SMALLER) and KERNAL=GAUSSIAN**



* **C=100000 (LARGER) and KERNAL=GAUSSIAN**

