Case study: NAML user workflow for 2013 driving data

In this section we will outline the workflow of using NAML with a real fNIRS dataset collected from a distracted driving experiment aimed at determining low and high memory workload. The experimental data is exported into a csv. The following specifies the steps needed to run and use the tool effectively.

1) *Set up an environment for the tool to run*
   In order to run NAML on your computer you will need:
   a) **linux** or **mac** operating system
   b) **python 3**
   c) Installed **sktime** library
   If you want to further gain further information on installing sktime or setting up sktime you are welcome to use the getting started with sktime guide.

2) *Download the repository*
   The next step is to download the repository from the following GitHub link:
   https://github.com/erins/NAML
   The following picture shows the GitHub repository. Click the green download button on the upper right hand corner.

   The downloaded repository has two main folders. The "examples" folder contains sktime tutorials and the "scripts". The scripts file on this computer contains four main items needed to run the tool
   1. `naml.py`
   2. `configuration_checker.py`
   3. `data` folder
   4. `configurationFile` folder

   Using the terminal navigate to script folder within the downloaded folder and type the following commands that will make the python files executable :

   ```
   chmod a+x naml.py
   chmod a+x configuration_checker.py
   ```

3) *Explore the data in the csv file*
   Under the data folder you will find a csv file labeled 2013e.csv. The data has multiple columns including: name, event, channel, start time, end time
   - name represents the participant. There about thirty participants in this dataset.
   - event is the type of event the participant is performing. For this dataset we have two event, 0 back and 2 back tests. This data could be indicative of high and low memory workload, respectively.
   - channel represents the sensor the data is being received from. This dataset has sixteen different sensors.

- start time and end time are the start and end time for a particular event done by a participant
- The remaining columns represent the data points collected by the machine over time at the sampling frequency

Several columns represent the same event but with different sensors. One way to distinguish between the different events is to look at the start time or end time

4) *Set up the configuration file*

The following picture shows an example configuration file that can be found under `configFiles`.

```
{
 "filePath":"../scripts/data/2013e.csv",
 "loggingEnabled": true,
 "targetCol": "event",
 "percentTrain":0.5,
 "jobs": [
    {"method": "SHAPELET_TRANSFORM"
    },
    {"method": "UNIVARIATE_TRANSFORMATION",
     "classifier": "TSF_CLF"
    },
    {"method": "COLUMN_ENSEMBLE",
     "ensembleInfo":[{
                      "classifier": "TSF_CLF",
                      "columnNum":1
                     },
                     {
                      "classifier": "TSF_CLF",
                      "columnNum":0
                     }]
    }
  ]
}
```

a) filePath , a string, defines the path to the data you will be using
b) loggingEnabled, true or false, determines whether the output of the run will be logged onto a text file
c) targetCol is a string that specifies which column the data will be classified into
d) percentTrain is a float value that determines the percent of data that will be trained on
e) jobs is a list of jobs that will be run. Within each job you must specify a method. If the type of method is univariate transformation, you can also specify a classifier.

5) *Check correctness of the configuration file*

On the command line type the following command:

```
./configuration_checker.py  ../scripts/configFiles/example_config.json
```

This script is aimed at helping you detect any errors with configuration file. You can also choose to run python files in the following manner.

```
python configuration_checker.py  ../scripts/configFiles/example_config.json
```

6) Run the classification jobs on the data
   On the command line type the following command:
   `./naml.py ../scripts/configFiles/example_config.json`

7) Review logged output
   In your local directory navigate to the logs folder. In this folder you will find a .txt file that will show the percent accuracy and the time it took for the model to run.