Project Installation Guideline

Author: HoangBamberg@Gmail.Com

1. Project Description

A. Purpose of the project

The Python GUI project was developed as the programming part for fulfilling the thesis (topic "*Evaluating the Impact of Sampling on Activity Monitoring in Dairy Cattle*") in MSc. Software Systems Science program at the University of Bamberg (https://www.uni-bamberg.de/en/ma-isosysc/). This is also a research part of the FutureIoT/Rindertracking project https://www.futureiot.de/portfolio/rindertracking/

The topic is about building and evaluating Activity recognition (https://en.wikipedia.org/wiki/Activity_recognition) (Machine Learning) models from triaxial sensors data (accelerometer and gyroscope carried by objects (cattle in this case) of the study). The project, however, can be used to build Machine Learning models in recognizing human activity as well.

B. Development Language and Environment

The project is developed in Python 3.8 (64 bit) using PyCharm IDE. The required packages (dependencies) are defined in the file **requirements.txt** (located in the root folder of the Repository). In case the project is loaded by PyCharm IDE, then it is automatically detected, and a pop-up will ask for the installation (see https://www.jetbrains.com/help/pycharm/managing-dependencies.html).

Alternatively, this can be done by manual command `python -m pip install -r /requirements.txt.`

C. The project workflow

After researching the previous studies, the author proposed an implemented workflow as shown in **Figure 1**.

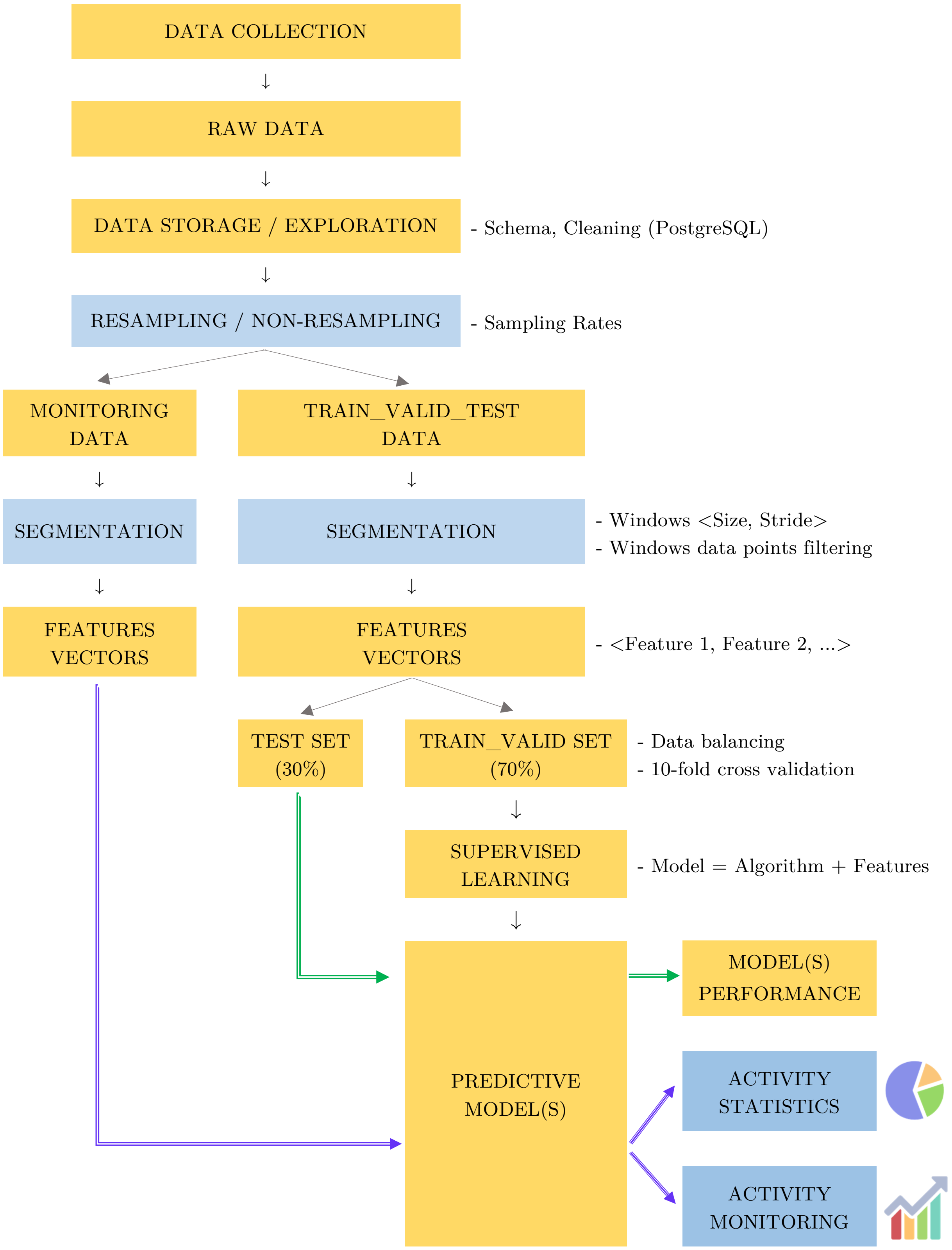


Figure 1: Implemented workflow

D. Database Management System

The data for the application come from the device Bosch BNO055 which contains two sensors (accelerometer and gyroscope) originally sampled at 10Hz.

The database management system used in the project is PostgreSQL version 9.5.17. The input data table has the structure described in **Table 1.**

The two data tables **Train_Valid_Test** and **Monitoring** must contain at least 9 columns (i.e., *cattle_id, label, gx, gy, gz, ax, ay, az, timestamp*) with data types as follow:

Column name	**Data type**	**Explanation**
cattle_id	text	ID number of the cow (e.g., Lilith/*Hanna*)
label	text	Labelled activity of cows (e.g., *Liegen)*
gx	double precision	Gyroscope x axis signal value
gy	double precision	Gyroscope y axis signal value
gz	double precision	Gyroscope z axis signal value
ax	double precision	Accelerometer x axis signal value
ay	double precision	Accelerometer y axis signal value
az	double precision	Accelerometer z axis signal value
timestamp	bigint	Timestamp in Unix Epoch time format

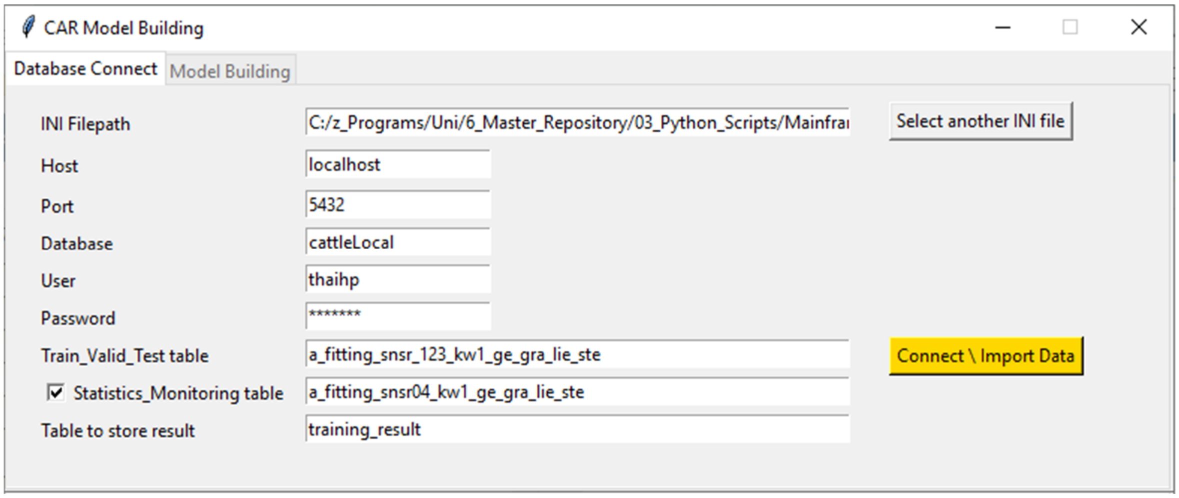
Table 1: Training / Monitoring data table structure

E. Repository Structure

Folder/File	**Purpose**
app.ini	Stores setting for GUI building
car_model_building.py	Main Python module
csv_out	Folder containing csv output files for each run
db_credentials.ini	Stores credentials and tables' names for database connection
features.py	Contains functions for calculating features
requirements.txt	Stores versions' names of the required Python packages
sample_datasets	Folder contain sample data for some cattle
user_defined_functions.py	Contains other user-defined functions

Table 2: Repository Structure

2. Step-by-step running



INI Filepath	C:/z_Programs/Uni/6_Master_Repository/03_Python_Scripts/Mainfra
Host	localhost
Port	5432
Database	cattleLocal
User	thaihp
Password	*******
Train_Valid_Test table	a_fitting_snsr_123_kw1_ge_gra_lie_ste
☑ Statistics_Monitoring table	a_fitting_snsr04_kw1_ge_gra_lie_ste
Table to store result	training_result

Figure 2: Database connection screen

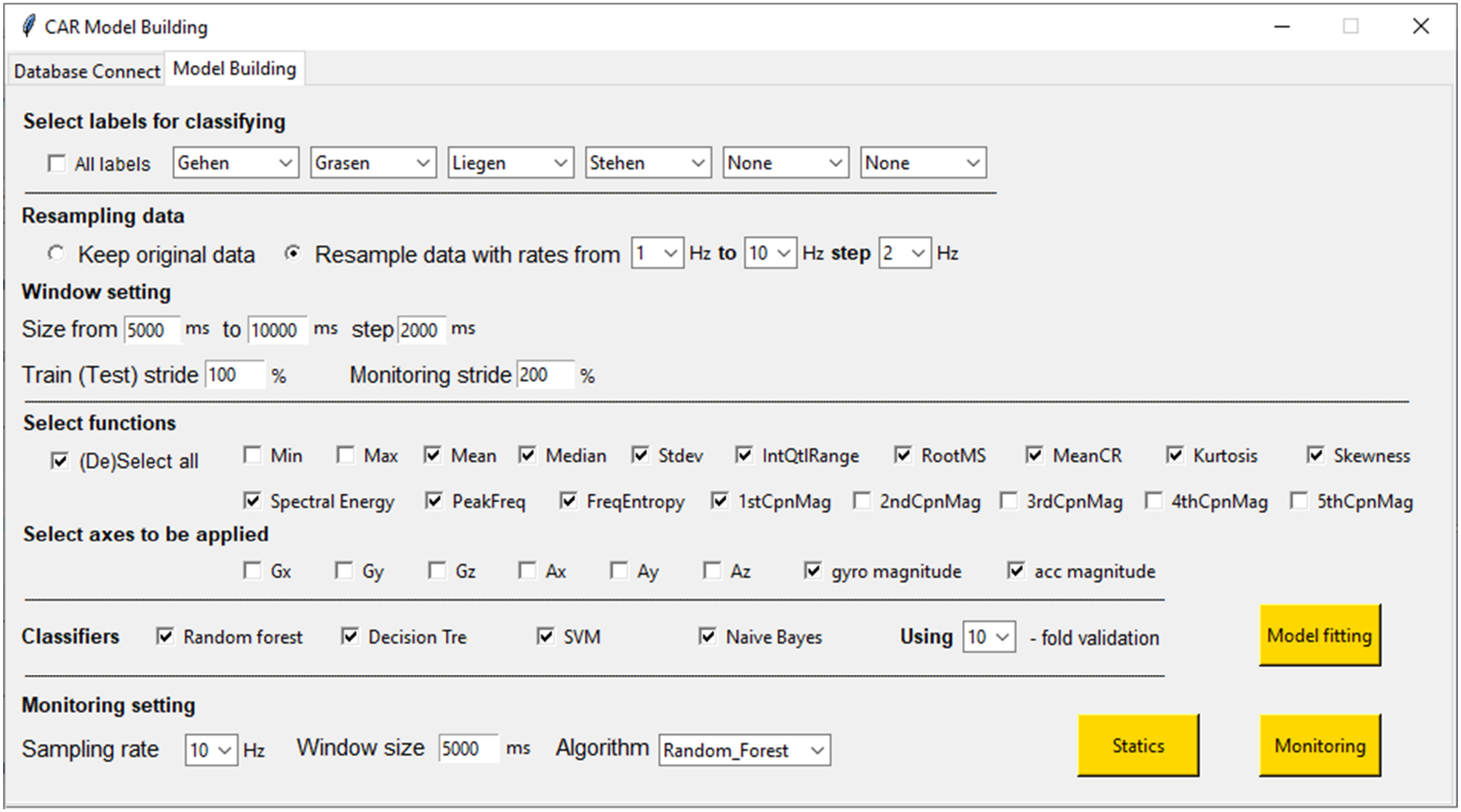
– **Ini Filepath** section stores the path to the ini file that contains the database credentials (e.g., the template is the file **db_credentials.ini** shown in **Table 2**). In the very first run if it shows a message "*Could not find a proper credentials ini file for db connection*" then please update the file **db_credentials.ini** appropriately. The fields **traintable** and **monitortable** in this ini file should refer to the tables that exist in the PosgreSQL database system.

– **Host, Port, Database, User, Password** stores database connection and user credentials which are loaded (*updated*) from (*to*) ini file in **Ini Filepath** section.

– **Train_Valid_Test table** indicates the name of table which is used for the train/valid and test phrase. This table structure meets the requirements in **Table 1**.

– **Statistics_Monitoring table** indicates the name of table which is used for the Statistics/Monitoring phrase. This table structure meets the requirements in **Table 1**. If the checkbox is unchecked, then it only runs training/validation and testing phrase.

– **Table to store result** stores the result for each experiment, it is automatically created if not exists on the database system. Every derived model will be saved into this table as a new record. The structure of this table is described in **Structure of the result table**.



CAR Model Building

Database Connect | Model Building

Select labels for classifying

□ All labels Gehen Grasen Liegen Stehen None None

Resampling data

○ Keep original data ● Resample data with rates from 1 Hz **to** 10 Hz **step** 2 Hz

Window setting

Size from 5000 ms to 10000 ms step 2000 ms

Train (Test) stride 100 % Monitoring stride 200 %

Select functions

☑ (De)Select all □ Min □ Max ☑ Mean ☑ Median ☑ Stdev ☑ IntQtlRange ☑ RootMS ☑ MeanCR ☑ Kurtosis ☑ Skewness

☑ Spectral Energy ☑ PeakFreq ☑ FreqEntropy ☑ 1stCpnMag □ 2ndCpnMag □ 3rdCpnMag □ 4thCpnMag □ 5thCpnMag

Select axes to be applied

□ Gx □ Gy □ Gz □ Ax □ Ay □ Az ☑ gyro magnitude ☑ acc magnitude

Classifiers ☑ Random forest ☑ Decision Tre ☑ SVM ☑ Naive Bayes **Using** 10 - fold validation

Monitoring setting

Sampling rate 10 Hz Window size 5000 ms Algorithm Random_Forest

Figure 3: Model building screen/tab

– **Select labels for classifying**: Select **All labels** to train the model with all labels existing in train table, otherwise it will train the model with selected labels.

– **Resampling data**: User can either choose **Keep original data** or resample the data with given sampling rates.

– **Window setting**: With the setting like in the **Figure 3**, it will run with the window sizes of 5, 7 and 9 seconds. The **Train (Test) stride** is set to 100% meaning all data in **Train_Valid_Test table** will be used for Train/Valid and Test phrase. The **Monitoring stride** option is set to 200% meaning only half of data in **Statistics_Monitoring table** will be used for Statistics/Monitoring metrics calculations.

– **Select functions** and **Select axes to be applied** help to choose features for the classifier(s) selected in the **Classifiers** section underneath. The number of features = number of functions * number of axes.

– After selecting **classifier(s)** and **Kfold** option, just click Model fitting button for running **Train_Valid_Test** phrase, the result will display on PyCharm console, log into *"Train_Test_Result.txt"* file in **csv_out** folder and update to Database.

– **Monitoring setting**: After fitting the model, user can view the Statistics and Monitoring metrics with the two buttons Statistics and Monitoring. These two buttons will be disabled if the checkbox in the **Figure 2** is unchecked.

– Because the statistics and monitoring metrics are generated under each classifier at a specific sampling rate and window size, in the **Monitoring setting** section user needs to choose these three options for viewing these metrics. It is important to select these options that fall into one combination of sampling rates and window sizes (types) selected previously right above. To be more clearly, settings in **Figure 3** enables user to train the model with sampling rates of 1, 3, 5, 7, 9 (Hz) in combination with window sizes of 5, 7, 9 (seconds) under 4 classifiers (5 * 3 * 4 = 60 combinations). As a result, in the Monitoring setting section, if user selects Sampling rate of 10Hz (which is not in the 60 combinations) then it does not show the statistics/monitoring metrics.

– The Statics function helps to see how well a model can predict unseen data. More clearly it will show a comparison between ground-truth data and the predicted data generated by the model. An example of this function is shown below

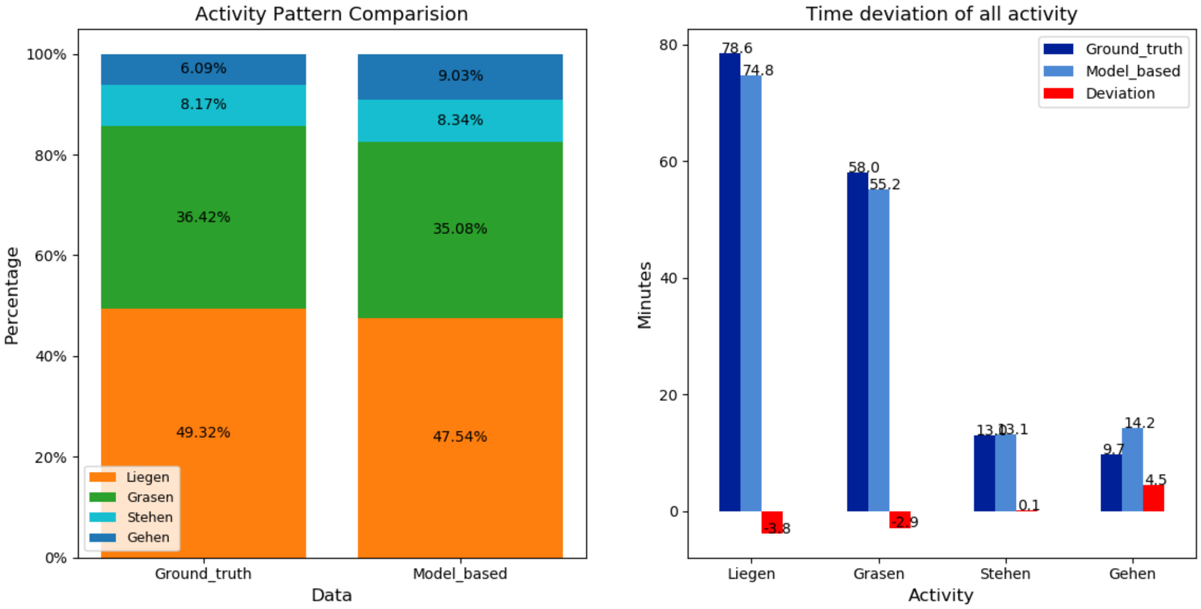


Figure 4: Activity statistics comparison between ground truth and model-based prediction

3. Other important notes

A. How to train/valid/test on multiple (mixed) cows' data and testing on another unseen cow?

To train on cow 1,2,3 and test on cow 4, just create the **Train_Valid_Test table** containing data from cow 1,2,3 then create the **Statistics_Monitoring table** containing sensor data of cow 4. Data for each cow in the **Train_Valid_Test table** distinguished by **cattle_id** field**.** The structure of these tables must meet the requirements in **Table 1**.

The result will be shown in PyCharm console, updated into database and logged into *"Train_Test_Result.txt"* file located in **csv_out** folder as described in **(C)**.

B. *How to carry Binary classification (E.g.: Lying* and *Non-lying)*

The application is originally developed for multiclass classification, so in order to carry a binary classification as between a main label (**Liegen)** and some other **Non-main labels (Gehen, Grasen, Stehen),** the following requirements should be met

+ The data in both **Train__Valid__Test table** (and **Statistics__Monitoring table**) must contain all four labels **Gehen Grasen Liegen Stehen**

+ In the **app.ini** file/section [GENERAL SETTINGS], the following variables must be set like below:

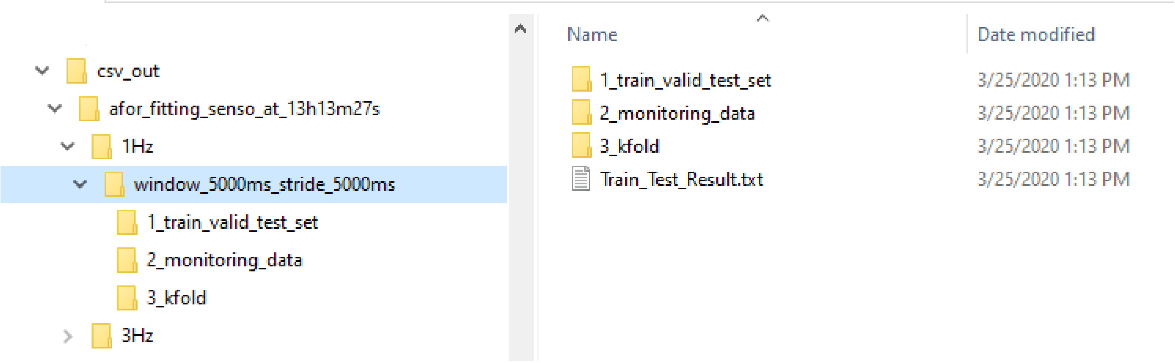
```
binarymode = 1
mainlabel = Liegen
sublabels = Gehen__Grasen__Stehen
```

Note: the sublabels should be separated by the underline (__) character. This setting can be used to build binary model to predict Rumination and Non-Rumination (Grasen, Stehen…) The number of sub labels could be 1,2,3…. The tool will automatically detect them.

C. CSV log files

With a specific sampling rate and window size (type), a text file called "Train__Test__Result.txt" will be created to store the experiment results for that configuration. This txt file is created in a sub folder of **csv__out** folder as shown Figure 5. This sub folder is created in every run (every click on Model fitting button).

Additionally, for each configuration (e.g., sampling rate, window size/stride) the data set regarding **train__valid__test** and **monitoring** can be logged into .csv file for checking of correctness of features calculation. To enable .csv files saving, in the **app.ini** file/section [GENERAL SETTINGS], just set the variable **csvsaving** to 1 (Or *csvsaving = 1*)



Name	Date modified
1_train_valid_test_set	3/25/2020 1:13 PM
2_monitoring_data	3/25/2020 1:13 PM
3_kfold	3/25/2020 1:13 PM
Train_Test_Result.txt	3/25/2020 1:13 PM

Figure 5: csv__out folder structure

4. Structure of the result table on the database

Column name	**Data Type**	**Description**
model__title	text	This column is created for the purpose of showing the model to the end-user. The default value is an empty string, when it is given a string then it will be shown to the end-user.
model__init__name	text	The initialised name of the model, after a model is trained then its name is set at the default format: [date__of__creation] __[hhmmss]__username__[Binary/Multi]__[name__of__algorithm] An example is "20200520__005954__thaihp__Binary__RandomForest"
model__binary__content	bytea	The content of the model in binary format
model__comments	text	User comments of the model
train__table	text	The name of table for training
monitor__table	text	Table for the Staticstic__Monitoring metrics
no__of__predicted__classes	integer	Number of classes to be classified
list__of__predicted__classes	text	List of classes to be classified
original__sample__rate__in__hz	integer	The original sampling rate of the training data (train__table)
no__of__original__train__data__points	integer	Number of data points in the train__table
resampled__rate__in__hz	integer	resampling rate (if user chooses resample data in Figure 3 of readme.pdf file)
no__of__resampled__train__data__points	integer	The number of data points of training data after resampling
no__of__instances__for__each__class__in__resampled__train__table	integer	Number of instances for each class in the (resampled) training data
algorithm	text	The classifier selected
no__of__functions	integer	Number of functions

list_of_functions	text	List of functions
no_of_axes	integer	Number of axes selected for the training
list_of_axes	text	List of axes selected for the training
window_size	integer	The window size (in milliseconds) in of **training/monitoring** phrase
window_stride	text	The window stride for **training** phrase **only**
k_fold	integer	K fold
accuracy_train_valid	real	Accuracy of train_valid phrase
precision_train_valid	real	Precision of train_valid phrase
recall_train_valid	real	Recall of train_valid phrase
specificity_train_valid	real	Specificity of train_valid phrase
f1_train_valid	real	F1 score of train_valid phrase
accuracy_test	real	Accuracy on Test set (30% of the training data)
precision_test	real	Precision on Test set
recall_test	real	Recall on Test set
specificity_test	real	Specificity of Test set
f1_test	real	F1 score of Test set
monitoring_window_stride	text	The window stride for monitoring phrase
accuracy_monitor	real	Accuracy on Monitoring data
precision_monitor	real	Precision on Monitoring data
recall_monitor	real	Recall on Monitoring data
specificity_monitor	real	Specificity on Monitoring data
f1_monitor	real	F1 score on Monitoring data
start_time	timestamp	Starting time of the run
end_time	timestamp	Ending time of the run
running_time_in_minutes	text	The duration of the experiment