Ref. Ares(2018)3461245 - 29/06/2018

D3.4 FINAL USER TRACKING ALGORITHMS

Project title	**Collaborative Recommendations and Adaptive Control for Personalised Energy Saving**
Project acronym	**enCOMPASS**
Project call	**EE-07-2016-2017 Behavioural change toward energy efficiency through ICT**
Work Package	**WP3**
Lead Partner	**CERTH – Center for Research and Technology Hellas**
Contributing Partner(s)	**WVT**
Security classification	**PU (Public)**
Contractual delivery date	**June 2018 (M20)**
Actual delivery date	**June 2018 (M20)**
Version	**1.0**
Reviewers	**POLIMI** **SUPSI**

History of changes

Version	Date	Comments	Main Authors
0.1	15/05/2018	First Draft and ToC	CERTH
0.6	17/06/2018	Contribution on Sections 2, 3, 4, 5 and 6	CERTH
0.7	25/06/2018	Contribution on Section 1, Executive Summary and Conclusions	CERTH, WVT
0.8	26/06/2018	Peer review	PMI, SUPSI
0.9	28/06/2018	Quality check	PMI
1.0	28/06/2018	Ready for submission to the EC	CERTH

Disclaimer

***This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 723059*.**

Table of Contents

History of changes 2

Disclaimer 3

Table of Contents 4

List of Tables 5

List of Figures 6

List of Definition and Abbreviations 7

Executive Summary 8

1 Introduction 10

2 Machine learning – Evaluation measures 11

2.1 Machine learning technique – Decision of dominant classifier 11

2.1.1 Random Forest 11

2.2 Evaluation measures 13

3 Occupancy Inference in Indoor Environments 15

3.1 Data collection and setup for occupancy inference algorithm 15

3.1.1 Data collection and setup 15

3.1.2 Data cleansing and pre-processing 15

3.2 Machine learning based occupancy estimation approach using random forest classifier 17

4 Experimental setup and results 19

4.1 Simulation setup 19

4.2 Simulation results 19

5 Algorithm Graphical User Interface (GUI) 24

Conclusions 26

References 27

List of Tables

Table 1: Confusion matrix for occupancy inference. 14
Table 2: Database SHF. Simulation results of support vector machine with radial basis function kernel for precision, recall, accuracy and F_1-score (estimated averages) for 100 Monte Carlo iterations (highest values in bold). 19
Table 3:Database SHF. Simulation results (in %) of support vector machine with polynomial kernel for precision, recall, accuracy and F_1-score (estimated averages) for 100 Monte Carlo iterations (highest values in bold). 20
Table 4: Database SHF. Simulation results (in %) of naive Bayes, logistic regression and decision trees for precision, recall, accuracy and F_1-score (estimated averages) for 100 Monte Carlo iterations (highest values in bold). 20
Table 5: Database SHF. Simulation results (in %) of (a) random forest and (b) back-propagation network for precision, recall, accuracy and F_1-score (estimated averages) for 100 Monte Carlo iterations (highest values in bold). 21
Table 6: Database WVT. Simulation results of support vector machine with radial basis function kernel for precision, recall, accuracy and F_1-score (estimated averages) for 100 Monte Carlo iterations (highest values in bold). 22
Table 7: Database WVT. Simulation results (in %) of support vector machine with polynomial kernel for precision, recall, accuracy and F_1-score (estimated averages) for 100 Monte Carlo iterations (highest values in bold). 22
Table 8: Database WVT. Simulation results (in %) of naive Bayes, logistic regression and decision trees for precision, recall, accuracy and F_1-score (estimated averages) for 100 Monte Carlo iterations (highest values in bold). 23
Table 9: Database WVT. Simulation results (in %) of (a) random forest and (b) back-propagation network for precision, recall, accuracy and F_1-score (estimated averages) for 100 Monte Carlo iterations (highest values in bold). 23

List of Figures

Figure 1: Breiman’s random forest algorithm. 12
Figure 2: Energy consumption measurements from SHF database (MySQL Workbench view). 16
Figure 3: Dataset for selected dwelling after pre-processing (Python view). 16
Figure 4: Flow chart of proposed algorithm. 17
Figure 5:Login and successful connection windows. 24
Figure 6:Interval, dwelling selection, dwelling update command and prediction plot command. 24
Figure 7:Occupancy inference results. 25

List of Definition and Abbreviations

Abbreviation	Definition
AUC	Area under Curve
FN	False negative
FP	False positive
GA	Grand Agreement
GUI	Graphical User Interface
OI	Occupancy Inference
TN	True negative
TP	True positive

Executive Summary

Deliverable D3.4 "Final User Tracking Algorithms" is a deliverable of type "demonstrator", specified in the "amended" GA description as follows:

"Final prototype, with documentation, of the algorithms for tracking the presence and movement of users in different indoor conditions, updated after the validation in the pilots".

The present report documents the software component that constitutes the actual deliverable.

The code of the software deliverable is available in the following software repository:

ssh://certh-encompass@18.184.32.200:22/var/git/encompass-occupancy-inference.git

Access is granted upon request.

The major goal of this document is to explain the final version of the algorithms for tracking the presence of users in indoor environments developed in enCOMPASS.

Identifying building occupancy is an essential task for building analysis, though in most cases it is defined using predefined functions without use of any kind of measuring and training. Frequently in fact tools [Liao15, Mahdavi09, Yang03] analyse the building occupancy based on stochastic models (e.g. Markov chains or probabilistic distributions), but they are not accurate since the exploited occupant building usage takes into account predefined models that in most cases do not match to the actual operation of the building. The most accurate building occupancy acquisition can be automatically performed by utilizing surveillance sensors. Furthermore, building occupancy can be estimated utilizing indirect information, such energy consumption. Analysing this information, one can infer the occupancy in building with high accuracy.

Deliverable D3.4 provides a description of the final version of algorithms used for the occupancy inference in indoor environments; it illustrates:

- The environmental features and energy consumption variables used for training machine learning techniques (classifiers) for occupancy inference.
- The algorithms based on energy consumption of devices/ appliances utilizing machine learning techniques (classifiers):
    - Random Forest.
- The Graphical User Interface (GUI) developed for the visualization of results.

The main dependencies with other deliverables are as follows:

- Deliverable D3.1 "Datasets with Context Data and Energy Consumption Data": This deliverable contains the specification of the energy consumption historical data sets, collected by the utility companies, as well as the building owners of the enCOMPASS pilots.
- Deliverable D3.2 "First User Tracking Algorithms": This deliverable contains a description of the initial version of the algorithm for occupancy inference in households based on energy consumption of individual devices and appliances, along with a description of machine learning techniques used for training.

The deliverable is structured as follows:

- Section 1 is the introduction of the deliverable;
- Section 2 provides a brief description of the classification algorithms utilized in this deliverable; and the evaluation measures of classification;
- Section 3 describes the proposed algorithms for occupancy inference and the dataset pre-processing steps;

- Section 4 presents various experiments results that have been performed for the validation of the algorithms and for the selection of core classifier;
- Section 5 provides a description of the graphical user interface;
- The final two sections contain the Conclusions and References.

With respect to the version of the occupancy inference algorithms preliminarily described in D3.2, this final version adds the following improvements:

- Development of an intelligent interpolation algorithm for filling missing values, based on forward and backward passes.
- Thorough testing of the proposed algorithms on the available data, in order to detect the dominant classifier.
- Finalization of the proposed occupancy inference algorithm (both training and testing algorithms).
- Evaluation of the proposed approach.

1 Introduction

Knowing the true occupancy, the presence or the actual number of occupants of a building at any given time is fundamental for the effective management of various building operation functions, ranging from security concerns to energy savings targets, especially in complex buildings with different internal kinds of use. Occupant's locations within the building vary throughout the day, therefore it is difficult to characterize the number of people that occupy a particular space and for what duration, because human behaviour is considered stochastic in nature. In general, occupancy monitoring in buildings is of high interest, since occupancy significantly contributes to the performance of the building. Therefore, there is a need for detailed occupancy knowledge.

There are several state-of-the-art approaches regarding occupancy inference, whose performance and effectiveness depend each time on data availability and quality. A brief description of the latest state-of-the-art studies related to non-intrusive occupancy detection and occupancy estimation techniques is given in deliverable D3.2 "First User Tracking Algorithms".

This document is an accompanying document of the code developed within Task T3.2, and describes the algorithmic approaches that have been developed for occupancy detection and estimation in different indoor environments. State-of-the-art related work is presented in the next section. Final results about the performance of machine learning classifiers used for occupancy inference are provided along with a presentation of an easy-to-use graphical user interface for occupancy inference, based on simple household parameters.

2 Machine learning – Evaluation measures

In this section, the machine learning classifier that has been decided to be used for occupancy inference based on its classification performance among known classification approaches is described. Evaluation measures are also presented.

2.1 Machine learning technique – Decision of dominant classifier

In this section, we briefly describe the Random Forest classifier that has been selected as the core of occupancy inference methodology, among other classification approaches initially described in deliverable D3.2 “First User Tracking Algorithms”. In D3.2 the best known and state-of-art machine learning classification techniques, such as *support vector machine* (with polynomial and radial basis function kernels), *decision trees*, *random forest*, *naïve Bayes*, *logistic regression*, *back-propagation fully-connected neural network* and *adaboost* approach for ensemble learning, along with *hidden Markov models* and *conditional random fields*, were initially described and tested for their classification performance on energy consumption data from a domestic environment that accommodates three occupants.

The data were gathered using smart plugs (power consumption) sensors. As for the occupancy related data (for training the algorithms), they were also measured from the field using highly accurate, active infrared door counter sensors.

All the simulation and experimental results described in deliverable D3.2 “First User Tracking Algorithms” point to the fact that random forest and decision tree machine learning classifiers show a slightly higher accuracy on their classification performance, compared to the other tested classifiers. Moreover, they have managed to achieve an overall high performance (*F_1-Score*: 83.37% and 82.79%, respectively) (see Section 2.2 Evaluation measures). Thus, based on classification performance, random forest is the dominant classifier among state-of-art classification approaches regarding the specific problem of occupancy inference using energy consumption data from a domestic environment. So, it has been selected as the core of the occupancy inference algorithm described in Section 3. A brief description of random forest classifier is provided in Section 2.1.1 below.

2.1.1 Random Forest

Random forest, also known as random decision forest, is an ensemble of decision trees and each decision tree is constructed by using a random subset of the training data, while the output class is the mode of the classes decided by each decision tree. Random forest is the highest in accuracy among current classifiers, it runs efficiently on large databases and it can handle a vast amount of input variables without variable deletion [Breiman99].

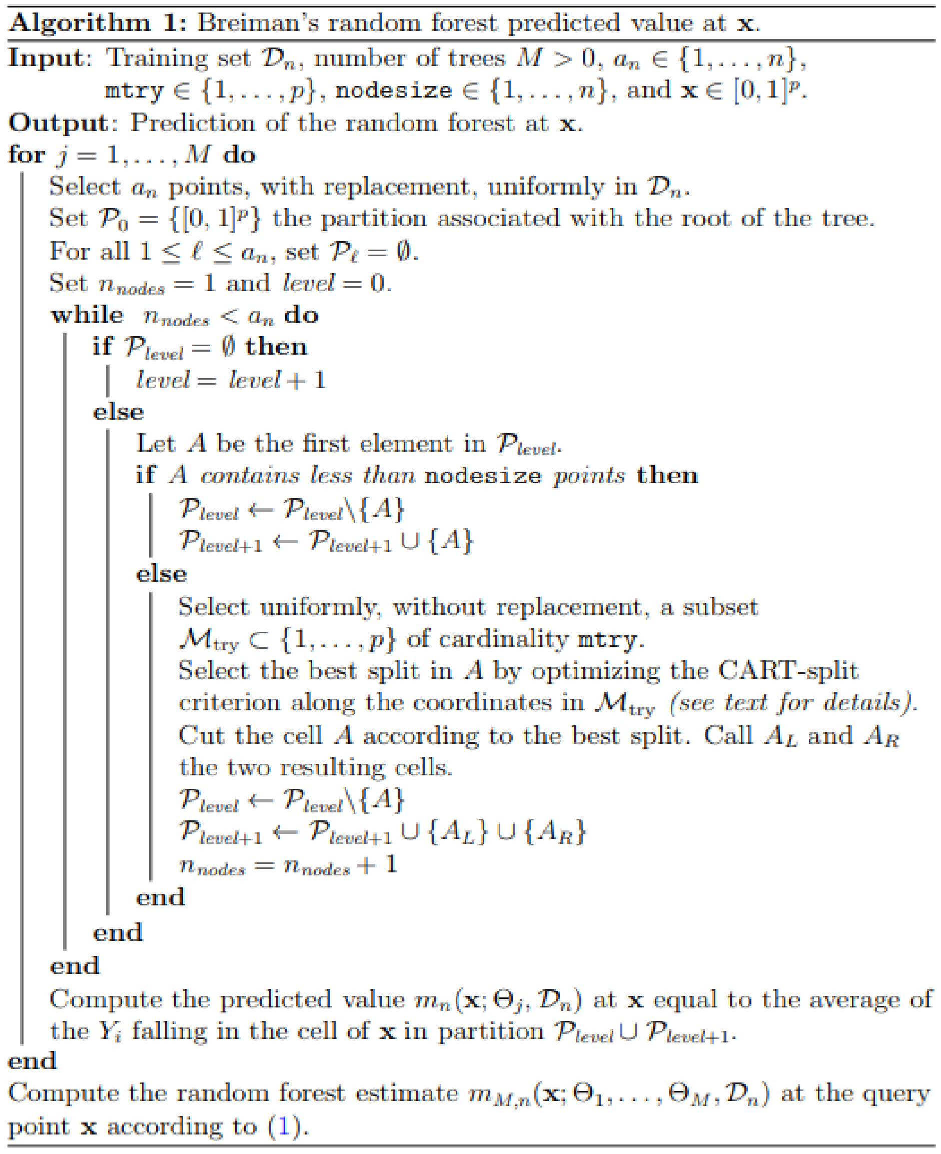
The random forest mechanism is versatile enough to deal with both supervised classification and regression tasks. However, to keep things simple, we focus in this introduction on regression analysis, and only briefly survey the classification case.

Our goal in this section is to provide a concise but mathematically precise presentation of the algorithm for building a random forest. The general framework is nonparametric regression estimation, in which an input random vector $X \in [0,1]^p$ is observed, and the goal is to predict the square integrable random response $Y \in R$ by estimating the regression function $m(x) = E[Y|X = x]$. With this aim in mind, we assume we are given a training sample $Dn = (X_1, Y_1), \dots, (Xn, Yn)$ of independent random variables distributed the same as the independent prototype pair (X, Y). The goal is to use the data set D_n to construct an estimate $m_n: [0,1]^p \to R$ of the function m. In this respect, we say that the regression function estimate m_n is (mean squared error) consistent if $E[m_n(X) - m(X)]^2 \to 0 \, asn \to \infty$ (the expectation is evaluated over X and the sample Dn).

A random forest is a predictor consisting of a collection of M randomized regression trees. For the $j-th$ tree in the family, the predicted value at the query point x is denoted by $m_n(x;\ \Theta_j;\ Dn)$ where $\Theta_1, \dots, \Theta_M$ are independent random variables, distributed the same as a generic random variable Θ and independent of Dn. In practice, the variable Θ is used to resample the training set prior to the growing of individual trees and to select the successive directions for splitting.
At this stage, we note that the trees are combined to form the (finite) forest estimate:

$$m_{\infty,n}(x; \Theta_1, \dots, \Theta_M, D_n) = \frac{1}{M}\sum_{j=1}^{M} m_n(x; \Theta_j, D_n) \tag{1}$$

In the (binary) supervised classification problem [Devroye96], the random response Y takes values in $\{0, 1\}$ and, given X, one has to guess the value of Y . A classifier or classification rule m_n is a Borel measurable function of x and Dn that attempts to estimate the label Y from x and Dn.



```
Algorithm 1: Breiman's random forest predicted value at x.
Input: Training set $\mathcal{D}_n$, number of trees $M > 0$, $a_n \in \{1, \dots, n\}$,
       mtry $\in \{1, \dots, p\}$, nodesize $\in \{1, \dots, n\}$, and $\mathbf{x} \in [0,1]^p$.
Output: Prediction of the random forest at $\mathbf{x}$.
for $j = 1, \dots, M$ do
    Select $a_n$ points, with replacement, uniformly in $\mathcal{D}_n$.
    Set $\mathcal{P}_0 = \{[0,1]^p\}$ the partition associated with the root of the tree.
    For all $1 \leq \ell \leq a_n$, set $\mathcal{P}_\ell = \emptyset$.
    Set $n_{nodes} = 1$ and $level = 0$.
    while $n_{nodes} < a_n$ do
        if $\mathcal{P}_{level} = \emptyset$ then
            $level = level + 1$
        else
            Let $A$ be the first element in $\mathcal{P}_{level}$.
            if $A$ contains less than nodesize points then
                $\mathcal{P}_{level} \leftarrow \mathcal{P}_{level} \backslash \{A\}$
                $\mathcal{P}_{level+1} \leftarrow \mathcal{P}_{level+1} \cup \{A\}$
            else
                Select uniformly, without replacement, a subset
                $\mathcal{M}_{\text{try}} \subset \{1, \dots, p\}$ of cardinality mtry.
                Select the best split in $A$ by optimizing the CART-split
                criterion along the coordinates in $\mathcal{M}_{\text{try}}$ (see text for details).
                Cut the cell $A$ according to the best split. Call $A_L$ and $A_R$
                the two resulting cells.
                $\mathcal{P}_{level} \leftarrow \mathcal{P}_{level} \backslash \{A\}$
                $\mathcal{P}_{level+1} \leftarrow \mathcal{P}_{level+1} \cup \{A_L\} \cup \{A_R\}$
                $n_{nodes} = n_{nodes} + 1$
            end
        end
    end
    Compute the predicted value $m_n(\mathbf{x}; \Theta_j, \mathcal{D}_n)$ at $\mathbf{x}$ equal to the average of
    the $Y_i$ falling in the cell of $\mathbf{x}$ in partition $\mathcal{P}_{level} \cup \mathcal{P}_{level+1}$.
end
Compute the random forest estimate $m_{M,n}(\mathbf{x}; \Theta_1, \dots, \Theta_M, \mathcal{D}_n)$ at the query
point $\mathbf{x}$ according to (1).
```

Figure 1: Breiman's random forest algorithm.

We now provide some insight on how the individual trees are constructed and how randomness kicks in. In Breiman's [Breiman99] original forests, each node of a single tree is associated with a hyperrectangular cell. At each step of the tree construction, the collection of cells forms a partition of $[0,1]^p$. The root of the tree is $[0,1]^p$ itself, and the terminal nodes (or leaves), taken together, form a partition of $[0,1]^p$. If a leaf represents region A, then the regression tree outputs on A the average of all Yi for which the corresponding Xi falls in A. Algorithm 1 describes in full detail how to compute a forest's prediction. Figure 1 provides a brief description of Breiman's random forest algorithm.

2.2 Evaluation measures

Different performance metrics are used to evaluate different machine learning algorithms. For now, we will be focusing on the ones used for classification problems. We can use classification performance metrics such as Log-Loss, Accuracy, AUC (Area under Curve) etc. For a two-class classification scenario, in order to assess our models, we use the measures of precision, recall, accuracy and F1-score, which are computed from the contents of the confusion matrix of the classification predictions (see Table 1). True positive and false positive cases are denoted as TP and FP, while true negative and false negative are denoted as TN and FN respectively. The terms associated with the confusion matrix are briefly described below:

- *True Positives (TP):* True positives are the cases when the actual class of the data point was 1(True) and the predicted is also 1(True).
- *True Negatives (TN):* True negatives are the cases when the actual class of the data point was 0(False) and the predicted is also 0(False).
- *False Positives (FP):* False positives are the cases when the actual class of the data point was 0(False) and the predicted is 1(True). False is because the model has predicted incorrectly and positive because the class predicted was a positive one.
- *False Negatives (FN):* False negatives are the cases when the actual class of the data point was 1(True) and the predicted is 0(False). False is because the model has predicted incorrectly and negative because the class predicted was a negative one.

The ideal scenario that we all want is that the model should give 0 False Positives and 0 False Negatives. But that's not the case in real life, as any model will NOT be 100% accurate most of the times.
In order to fit the classification evaluation in the occupancy detection problem, we will assign the classes' *absence*(0) and *presence*(1). *Precision* is the ratio of predicted true positive cases to the sum of true positives and false positives and is given by the equation:

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

Recall is the proportion of the true positive cases to the sum of true positives and false negatives and is given by the equation:

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

Accuracy is the fraction of the total number of predictions that were correct.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{4}$$

Precision or recall alone cannot describe a classifier's efficiency. That's why F_1-score is introduced as a combination of these two metrics. It is defined as twice the harmonic mean of precision and recall and is the metric we will be most referring to.

$$F_1 - Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{5}$$

A value closer to one means better combined precision and recall of the classifier, whereas lower values imply worst accuracy or precision or both.

Table 1: Confusion matrix for occupancy inference.

		Predicted class	
		Absence	**Presence**
Actual Class	**Absence**	*TP*	*FN*
	Presence	*FP*	*TN*

The confusion matrix is one of the most intuitive and easiest metrics used for finding the correctness and accuracy of the model. It is used for classification problems where the output can be of two or more types of classes. The form of Table 1 is similar in multi-class multi-label classification scenarios, and the calculation of precision, recall, accuracy and F_1-score is based on all the classes and then on averaging them to get a single real number measurement.

3 Occupancy Inference in Indoor Environments

In this section we present an overview of the available environmental and energy consumption data used for occupancy inference, along with the final occupancy inference methodology that have been developed in enCOMPASS. The method exploited for occupancy inference utilizes the random forest machine learning algorithm for classification, as it achieves higher classification performances among others. As a result, a model is created in advance through a training process. Furthermore, the methodology supports a two-class occupancy inference (absence / presence).

3.1 Data collection and setup for occupancy inference algorithm

3.1.1 Data collection and setup

The information retrieved from domestic environments (dwellings) is gathered with the use of smart meters (environmental and energy). The data were recorded in 15 minutes time intervals, namely 96 measurements per day per variable. The initial aggregated dataset contains *Humidity, Luminance, Temperature* and *Power consumption*. Another available variable called *Occupancy,* which counts the internal occupancy *(*0 for *absence /* 1 for *presence)* is used as ground truth for training machine learning models. These data were retrieved from three external databases. All the available data are stored in three (3) internal databases, one for each enCOMPASS pilot: SHF, SES and WVT. The data use for training and testing are retrieved automatically by the proposed algorithm.
An additional feature/variable included in data model is the *Humidity Ratio (W).* The *Humidity Ratio* is calculated using the measured temperature. The saturation pressure over liquid water $(p_{ws}\ in\ Pa)$ is calculated with:

$$\ln(p_{ws}) = \frac{C_1}{T} + C_2 + C_3T + C_4T^2 + C_5T^3 + C_6\ln(T) \tag{6}$$

where $C_1 = 5.8e + 03$, $C_2 = 1.39e + 00$, $C_3 = -4.864e - 02$, $C_4 = 4.176e - 05$, $C_5 = -1.445e - 08$, $C_6 = 6.545e + 00$. T is the absolute temperature, $K = C + 273.15$.
The *Humidity Ratio* is calculated using the following equation:

$$W = 0.622\frac{p_{ws}}{p - p_{ws}} \tag{7}$$

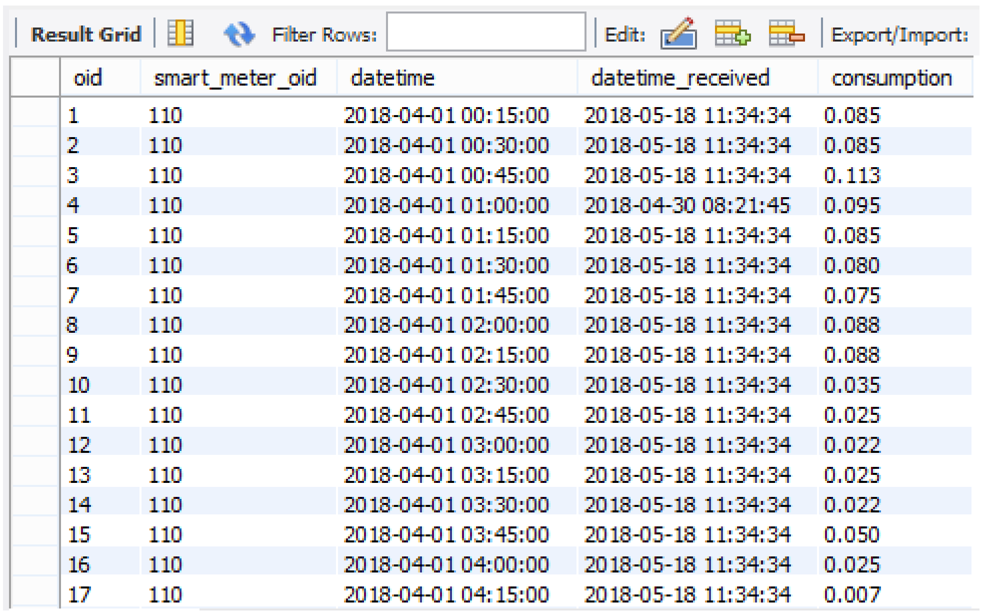
where p is taken as the standard atmospheric pressure $101.325kPa$.

3.1.2 Data cleansing and pre-processing

All the available data are pre-processed under two basic conditions:

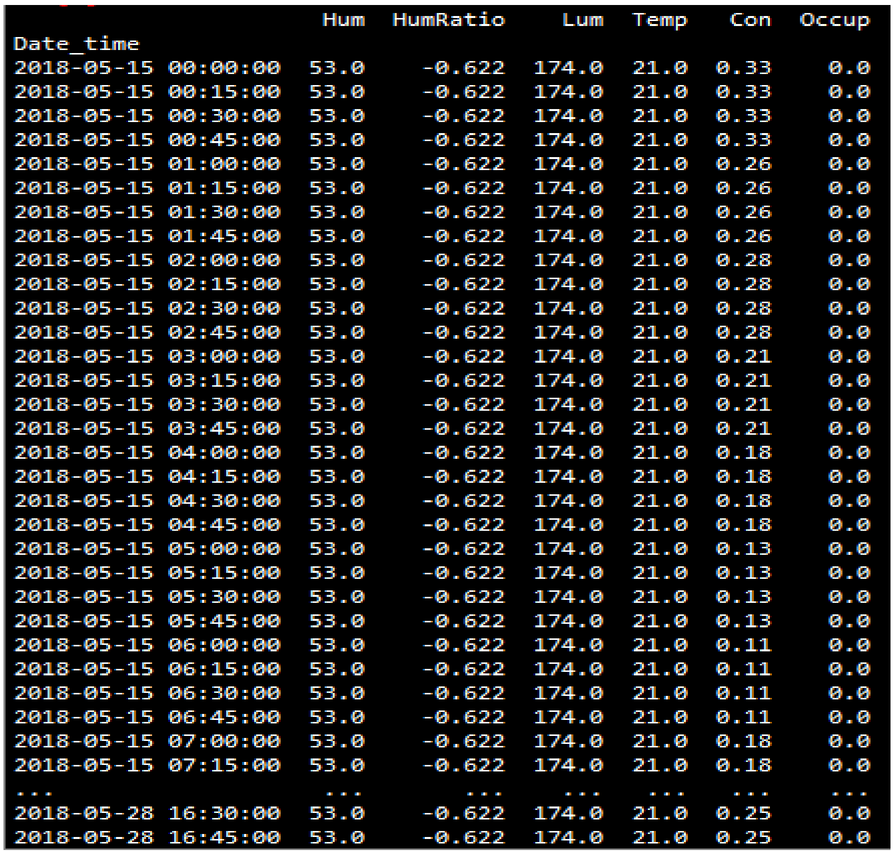
1. to retain 15 minutes time interval per feature by a pre-processing among selected dates;
2. missing values are filled using forward and backward fill.

The data processing of features takes place for those dwellings where all key features (Humidity, Luminance, Temperature and Power consumption) have at least on measurement for a selected period. This condition aims to provide a functional dataset that contains all the information that the trained model needs to perform efficiently and effectively. Below, the measurements of energy consumption per dwelling are provided in Figure 2 (MySQL Workbench 6.3 CE view). Their re-construction after preprocessing and the addition of Humidity Ratio is given in Figure 3 (Python 3.6 view). The form of the dataset for the selected dwelling for a given time period, shown in Figure 3, is given as input to the occupancy inference algorithm.



oid	smart_meter_oid	datetime	datetime_received	consumption
1	110	2018-04-01 00:15:00	2018-05-18 11:34:34	0.085
2	110	2018-04-01 00:30:00	2018-05-18 11:34:34	0.085
3	110	2018-04-01 00:45:00	2018-05-18 11:34:34	0.113
4	110	2018-04-01 01:00:00	2018-04-30 08:21:45	0.095
5	110	2018-04-01 01:15:00	2018-05-18 11:34:34	0.085
6	110	2018-04-01 01:30:00	2018-05-18 11:34:34	0.080
7	110	2018-04-01 01:45:00	2018-05-18 11:34:34	0.075
8	110	2018-04-01 02:00:00	2018-05-18 11:34:34	0.088
9	110	2018-04-01 02:15:00	2018-05-18 11:34:34	0.088
10	110	2018-04-01 02:30:00	2018-05-18 11:34:34	0.035
11	110	2018-04-01 02:45:00	2018-05-18 11:34:34	0.025
12	110	2018-04-01 03:00:00	2018-05-18 11:34:34	0.022
13	110	2018-04-01 03:15:00	2018-05-18 11:34:34	0.025
14	110	2018-04-01 03:30:00	2018-05-18 11:34:34	0.022
15	110	2018-04-01 03:45:00	2018-05-18 11:34:34	0.050
16	110	2018-04-01 04:00:00	2018-05-18 11:34:34	0.025
17	110	2018-04-01 04:15:00	2018-05-18 11:34:34	0.007

Figure 2: Energy consumption measurements from SHF database (MySQL Workbench view).



```
                      Hum  HumRatio    Lum  Temp   Con  Occup
Date_time
2018-05-15 00:00:00  53.0    -0.622  174.0  21.0  0.33    0.0
2018-05-15 00:15:00  53.0    -0.622  174.0  21.0  0.33    0.0
2018-05-15 00:30:00  53.0    -0.622  174.0  21.0  0.33    0.0
2018-05-15 00:45:00  53.0    -0.622  174.0  21.0  0.33    0.0
2018-05-15 01:00:00  53.0    -0.622  174.0  21.0  0.26    0.0
2018-05-15 01:15:00  53.0    -0.622  174.0  21.0  0.26    0.0
2018-05-15 01:30:00  53.0    -0.622  174.0  21.0  0.26    0.0
2018-05-15 01:45:00  53.0    -0.622  174.0  21.0  0.26    0.0
2018-05-15 02:00:00  53.0    -0.622  174.0  21.0  0.28    0.0
2018-05-15 02:15:00  53.0    -0.622  174.0  21.0  0.28    0.0
2018-05-15 02:30:00  53.0    -0.622  174.0  21.0  0.28    0.0
2018-05-15 02:45:00  53.0    -0.622  174.0  21.0  0.28    0.0
2018-05-15 03:00:00  53.0    -0.622  174.0  21.0  0.21    0.0
2018-05-15 03:15:00  53.0    -0.622  174.0  21.0  0.21    0.0
2018-05-15 03:30:00  53.0    -0.622  174.0  21.0  0.21    0.0
2018-05-15 03:45:00  53.0    -0.622  174.0  21.0  0.21    0.0
2018-05-15 04:00:00  53.0    -0.622  174.0  21.0  0.18    0.0
2018-05-15 04:15:00  53.0    -0.622  174.0  21.0  0.18    0.0
2018-05-15 04:30:00  53.0    -0.622  174.0  21.0  0.18    0.0
2018-05-15 04:45:00  53.0    -0.622  174.0  21.0  0.18    0.0
2018-05-15 05:00:00  53.0    -0.622  174.0  21.0  0.13    0.0
2018-05-15 05:15:00  53.0    -0.622  174.0  21.0  0.13    0.0
2018-05-15 05:30:00  53.0    -0.622  174.0  21.0  0.13    0.0
2018-05-15 05:45:00  53.0    -0.622  174.0  21.0  0.13    0.0
2018-05-15 06:00:00  53.0    -0.622  174.0  21.0  0.11    0.0
2018-05-15 06:15:00  53.0    -0.622  174.0  21.0  0.11    0.0
2018-05-15 06:30:00  53.0    -0.622  174.0  21.0  0.11    0.0
2018-05-15 06:45:00  53.0    -0.622  174.0  21.0  0.11    0.0
2018-05-15 07:00:00  53.0    -0.622  174.0  21.0  0.18    0.0
2018-05-15 07:15:00  53.0    -0.622  174.0  21.0  0.18    0.0
...                   ...       ...    ...   ...   ...    ...
2018-05-28 16:30:00  53.0    -0.622  174.0  21.0  0.25    0.0
2018-05-28 16:45:00  53.0    -0.622  174.0  21.0  0.25    0.0
```

Figure 3: Dataset for selected dwelling after pre-processing (Python view).

3.2 Machine learning based occupancy estimation approach using random forest classifier

In this section, a brief description of the updated approach for occupancy inference in dwellings is provided. Occupancy inference is based on dwellings' overall energy consumption and environmental measurements designed and implemented in previous work (see D3.2 "First User Tracking Algorithms"). The steps that the proposed algorithm follows are given below:

- Selection by the end of a time interval for occupancy inference. In the case of the pilots, by default the two previous days are automatically selected for inferring the occupancy of the pilot buildings.
- Dwelling based predictions for a given time period. The algorithm can determine which pilot buildings have measurements for:
    - Selected dates
    - Key features / variables.
- Pre-processing for the missing values (forward and backward fill of missing values).
- Execution of the Random Forest classification method, as the one with the highest classification and predictive performance among known machine learning classifiers.
- Usage of state of art metrics to determine precision, recall, accuracy and F_1-score.
- Prediction of occupancy measurements for the current interval.

A flow chart of the proposed algorithm is given in Figure 4, which shows the classifier gets as input all the environmental and overall energy consumption features in order to provide a decision about the occupancy inference for:

a) a selected interval
b) selected pilot building

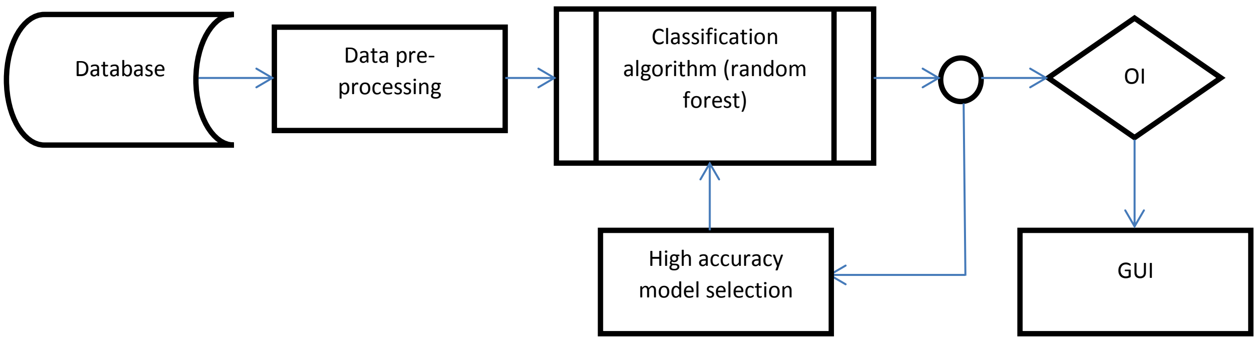

Database
Data pre-processing
Classification algorithm (random forest)
OI
High accuracy model selection
GUI

Figure 4: Flow chart of proposed algorithm.

More specifically, the data stored in the database are pre-processed and cleaned, so to improve data quality and overall classifier performance and effectiveness. The pre-processing operation steps are:

c) check dwellings have at least one recording for each feature for a given time period,
d) check time intervals between features recording are 15 minutes,
e) fill missing data (backward and forward fill),
f) calculate additional feature (humidity ratio).

After the pre-processing step, several random forest models are trained and the one with higher accuracy is selected as the one that will be used for occupancy inference (OI). The results are visualized with a functional and easy-to-use graphical user interface (GUI), briefly described in Section 5.

4 Experimental setup and results

In this section, we present performance evaluation results of the machine learning classifiers used for occupancy inference proving the dominance of random forest classifier compared to other machine learning classifiers (support vector machine (with polynomial and radial basis function kernels), decision trees, random forest, naïve Bayes, logistic regression, back-propagation fully-connected neural network). The simulation analysis is made on data from SHF, SES and WVT databases. Simulation results show again, as for the data from domestic environment (see D3.2 "First User Tracking Algorithms"), that random forest achieves the highest predictive performance, compared to other tested classifiers.

4.1 Simulation setup

Our main objective is to find the predictive model that is more efficient on occupancy inference based on energy consumption data. To that end, our simulation schema is based on the application of all tested classifiers. For cross-validation of our results, we generate a training set and a testing set, in a percent of 70% and 30%, respectively of the tested dataset. We generate 100 Monte Carlo iterations for different parameter scenarios in each classifier. For support vector machine with polynomial kernel, θ takes the values $\theta = (start = 10, end = 40, step = 10)$ and the polynomial degree takes the values $p = (2,6,1)$. For support vector machine with radial basis function kernel σ varies $\theta = (0.001,0.01,0.1)$ and the constant C as $C = (100,1000)$. The classic back-propagation full connected network has a single hidden layer and the number of neurons varies as $n = (100,200,10)$. The random forest has an ensemble of $estimators\ = (20,100,20)$ decision trees. Other classifiers tested here are naïve Bayes, logistic regression and decision trees, which have been used utilizing default parameters from python *sklearn* library. The combination of all values of parameters and a size of 100 Monte Carlo iterations for each case leads to a total of 3900 tested cases. For the definition of machine learning classifiers parameters, see D3.2 "First User Tracking Algorithms".

4.2 Simulation results

In this section, the simulation results of all machine learning classifiers tested for occupancy inference on data from SHF database are presented.

Table 2: Database SHF. Simulation results of support vector machine with radial basis function kernel for precision, recall, accuracy and F_1-score (estimated averages) for 100 Monte Carlo iterations (highest values in bold).

C	σ	**Precision (%)**	**Recall (%)**	**Accuracy (%)**	**F_1-score (%)**
100	0.001	89.75	88.76	87.91	89.25
100	0.01	**89.88**	**89.01**	**87.98**	**89.44**
100	0.1	89.77	88.65	87.34	89.21
1000	0.001	89.65	88.39	87.94	89.02
1000	0.01	88.93	87.29	86.18	88.10
1000	0.1	88.28	87.67	86.19	87.97

Table 3: Database SHF. Simulation results (in %) of support vector machine with polynomial kernel for precision, recall, accuracy and F_1-score (estimated averages) for 100 Monte Carlo iterations (highest values in bold).

p	θ	Precision (%)	Recall (%)	Accuracy (%)	F_1-score (%)
2	10	93.77	92.26	89.33	93.01
2	20	93.77	92.26	89.33	93.01
2	30	93.77	92.26	89.33	93.01
2	40	93.77	92.26	89.33	**93.01**
3	10	**95.47**	88.39	87.84	91.79
3	20	92.49	87.42	84.86	89.88
3	30	92.43	88.19	87.88	90.26
3	40	92.93	88.61	87.83	90.72
4	10	89.98	89.13	86.29	89.55
4	20	89.62	88.42	87.67	89.02
4	30	88.84	87.48	86.72	88.15
4	40	88.34	87.51	86.73	87.92
5	10	68.21	**95.26**	91.25	79.50
5	20	78.21	88.02	90.38	82.83
5	30	91.23	87.95	91.86	89.56
5	40	91.38	87.88	91.85	89.60
6	10	91.23	87.86	91.83	89.51
6	20	91.47	87.98	**92.34**	89.69
6	30	91.34	88.01	92.01	89.64
6	40	91.72	88.41	91.05	90.03

Table 4: Database SHF. Simulation results (in %) of naive Bayes, logistic regression and decision trees for precision, recall, accuracy and F_1-score (estimated averages) for 100 Monte Carlo iterations (highest values in bold).

Metric	Naïve Bayes	Logistic regression	Decision Tree
Precision (%)	83.08	80.23	94.09
Recall (%)	**92.59**	**94.28**	93.62
Accuracy (%)	80.65	78.66	**97.88**
F_1-score (%)	87.58	86.69	93.82

Table 5: Database SHF. Simulation results (in %) of (a) random forest and (b) back-propagation network for precision, recall, accuracy and F_1-score (estimated averages) for 100 Monte Carlo iterations (highest values in bold).

(a)

Estimators	Precision (%)	Recall (%)	Accuracy (%)	F_1-score (%)
20	94.37	95.96	92.80	95.16
40	**95.35**	96.63	94.04	95.99
60	94.44	97.31	93.80	95.85
80	94.74	96.97	93.75	95.84
100	94.77	**97.64**	**94.29**	**96.19**

(b)

Neurons	Precision (%)	Recall (%)	Accuracy (%)	F_1-score (%)
100	74.79	83.28	76.24	78.81
120	75.93	84.31	75.87	79.90
140	75.21	83.33	**76.71**	79.06
160	76.07	**84.52**	76.33	**80.07**
180	**76.12**	84.03	76.58	79.88

From simulation results presented in Table 2, Table 3, Table 4, and Table 5, one can see that random forest classifier has the dominant predictive performance compared to other tested machine learning classifiers. More specifically, the highest accuracy and F_1-score (87.98% and 89.44%, respectively) for support vector machine (with radial basis function kernel) takes place when $C = 100, \sigma = 0.01$. For support vector machine (with polynomial kernel), the highest accuracy and F_1-score (92.34% and 93.01%, respectively) takes place when $p = 6, \theta = 20$ and $p = 2, \theta = 40$. Naive Bayes, logistic regression and decision trees have 80.65%, 78.66% and 97.88%, respectively, accuracy and 87.58%, 86.69% and 93.82%, respectively, F_1-score. For random forest classifier, the highest accuracy and F_1-score (94.29% and 96.19%, respectively) takes place when the internal decision trees of the forest are $estimators = 100$. As for back-propagation fully connected neural network the highest accuracy and F_1-score (76.71% and 80.07%, respectively) takes place when layer neurons are $n = 140$ and $n = 160$, respectively.

Similar simulation results (not reported for the sake of conciseness) are computed for SES database, where again, random forest is the dominant classifier compared to others. In the specific case of the WVT database, the algorithm has been tested with energy consumption, temperature and humidity information, and results are presented below.

Table 6: Database WVT. Simulation results of support vector machine with radial basis function kernel for precision, recall, accuracy and F_1-score (estimated averages) for 100 Monte Carlo iterations (highest values in bold).

C	σ	Precision (%)	Recall (%)	Accuracy (%)	F_1-score (%)
100	0.001	**100.00**	27.69	72.83	43.37
100	0.01	90.91	30.77	72.83	45.98
100	0.1	86.67	**40.00**	**75.14**	**54.74**

Table 7: Database WVT. Simulation results (in %) of support vector machine with polynomial kernel for precision, recall, accuracy and F_1-score (estimated averages) for 100 Monte Carlo iterations (highest values in bold).

p	θ	Precision (%)	Recall (%)	Accuracy (%)	F_1-score (%)
2	10	79.17	32.76	74.57	46.34
2	20	78.57	37.93	75.72	51.16
2	30	79.31	39.66	76.30	52.87
2	40	79.31	39.66	76.30	52.87
3	10	80.65	43.10	**77.46**	56.18
3	20	80.65	43.10	**77.46**	56.18
3	30	78.12	43.10	76.88	55.56
3	40	**80.65**	43.10	**77.46**	56.18
4	10	62.22	48.28	72.83	54.37
4	20	57.63	58.62	71.68	58.12
4	30	50.00	51.72	66.47	50.85
4	40	54.84	58.62	69.94	56.67
5	10	65.38	58.62	75.72	**61.82**
5	20	58.06	**62.07**	72.25	60.00
5	30	68.42	54.17	70.52	60.47
5	40	67.31	55.21	68.92	60.67
6	10	59.70	57.97	67.63	58.82
6	20	61.14	56.98	66.10	58.99
6	30	62.05	55.27	66.32	58.46
6	40	62.76	56.74	68.86	59.60

Table 8: Database WVT. Simulation results (in %) of naive Bayes, logistic regression and decision trees for precision, recall, accuracy and F_1-score (estimated averages) for 100 Monte Carlo iterations (highest values in bold).

Metric	**Naïve Bayes**	**Logistic regression**	**Decision Tree**
Precision (%)	46.83	**100.00**	68.18
Recall (%)	**80.82**	33.82	66.18
Accuracy (%)	53.18	73.99	**74.57**
F_1-score (%)	59.30	50.55	**67.16**

Table 9: Database WVT. Simulation results (in %) of (a) random forest and (b) back-propagation network for precision, recall, accuracy and F_1-score (estimated averages) for 100 Monte Carlo iterations (highest values in bold).

(a)

Estimators	**Precision (%)**	**Recall (%)**	**Accuracy (%)**	**F_1-score (%)**
20	67.21	62.12	73.99	64.75
40	72.41	62.69	76.30	67.20
60	71.43	58.82	74.57	64.52
80	**74.44**	62.36	**76.88**	68.25
100	73.33	**64.71**	**76.88**	**68.75**

(b)

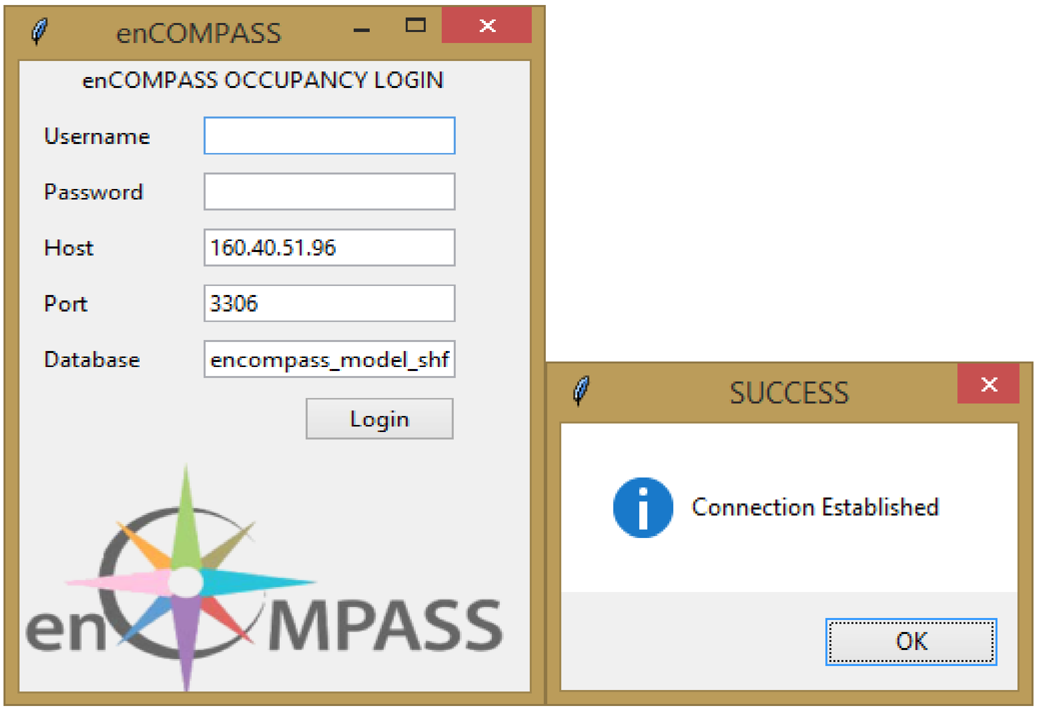
Neurons	**Precision (%)**	**Recall (%)**	**Accuracy (%)**	**F_1-score (%)**
100	90.44	24.64	**69.36**	39.08
120	77.78	**30.43**	68.79	**43.75**
140	75.00	**30.43**	68.21	43.30
160	**94.44**	24.64	**69.36**	39.08
180	**94.44**	24.64	**69.36**	39.08

From simulation results presented in Table 6, Table 7, Table 8 and Table 9, one can see that the random forest classifier again achieves the highest predictive performance compared to other tested classifiers. The maximum F_1-score achieved for support vector machine with radial basis function kernel is 54.74%, for support vector machine with polynomial kernel is 61.82%, for back-propagation network is 43.75%, for naive Bayes, logistic regression and decision tree is 59.30%, 50.55% and 67.16%, respectively, while for random forest is 68.75%.

5 Algorithm Graphical User Interface (GUI)

In this section, a simple graphical user interface of the occupancy inference algorithm is presented, which has been developed for demonstration purposes.

In Figure 5 the database login window with the parameters of Host, Port and Database selection is shown. The end-user can select among *encompass_model_shf*, *encompass_model_ses* and *encompass_model_wvt,* so as to work with the data (features and dwellings) stored in SHF, SES and WVT databases, respectively. After login, a message of successful connection pops up.

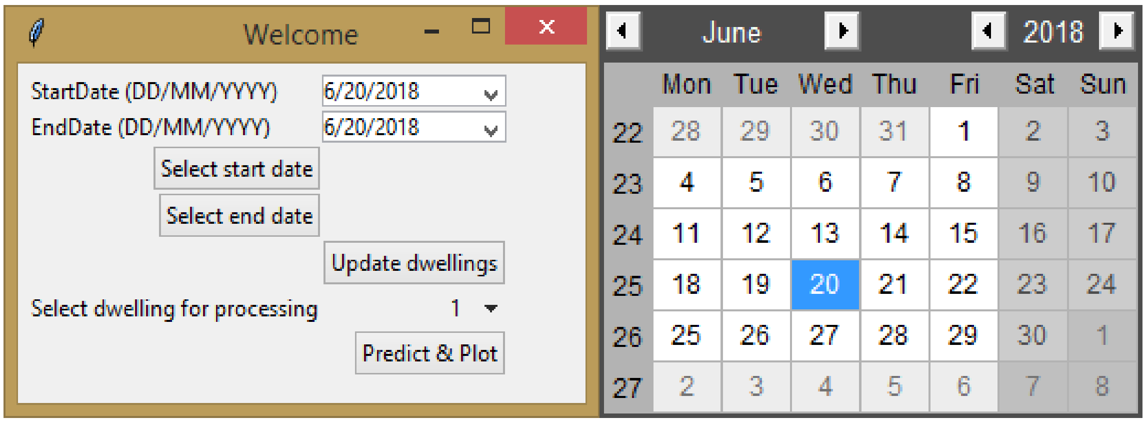


Username	
Password	
Host	160.40.51.96
Port	3306
Database	encompass_model_shf

Connection Established

Figure 5: Login and successful connection windows.

The window shown in Figure 6 pops up after successful connection, where the selection of analysis time period takes place. The end-user must choose the start date (StartDate) and the end date (EndDate) and then select the “Select start date” and “Select end date” commands so as to verify selected dates.



StartDate (DD/MM/YYYY)	6/20/2018
EndDate (DD/MM/YYYY)	6/20/2018
Select dwelling for processing	1

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
22	28	29	30	31	1	2	3
23	4	5	6	7	8	9	10
24	11	12	13	14	15	16	17
25	18	19	20	21	22	23	24
26	25	26	27	28	29	30	1
27	2	3	4	5	6	7	8

Figure 6: Interval, dwelling selection, dwelling update command and prediction plot command.

Moreover, the end-user must select the Update dwelling command, so that the algorithm can detect the dwellings with at least one record for Humidity, Luminance, Temperature and Power consumption in the selected time interval.

After the update of dwellings, the end-user can select the dwelling of interest so as to proceed with the analysis. Finally, the end-user must press the Predict & Plot button and the occupancy inference results are provided (see Figure 7).

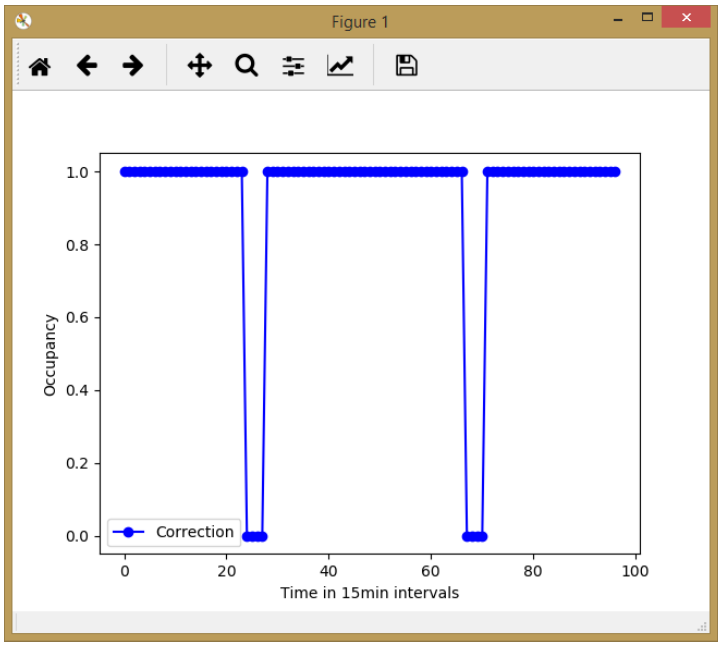


Figure 7: Occupancy inference results.

In case of a different time interval selection, the process after login must be repeated with the same steps.

Conclusions

The present document described the final version of an algorithm for occupancy inference, based on the initial version described in D3.2 “First User Tracking Algorithms”. The algorithm is dwelling based and aims to predict dwellings’ overall occupancy per 15 minutes in a day. The inputs of the model are environmental measurements (temperature, humidity, luminance, humidity ratio) and overall energy consumption of the dwelling. Except humidity ratio, which is calculated, all other environmental measurements and energy consumption are provided by smart meters. The proposed methodology utilizes random forest machine learning classifiers as the core model trainer, based on current results on SHF, SES and WVT internal databases and previous simulation results where the input features were household appliances (oven, hood, washing machine, TV, hair dryer, fridge) (see D3.2 “First User Tracking Algorithms”). In both cases, random forest classifier has better or slightly better predictive performances compared to other tested machine learning classifiers.
The algorithm is also provided with a visualization framework with a functional and easy-to-use graphical user interface. Finally, part of this work has been published and presented at the International Conference on Energy Science and Electrical Engineering (ICESEE’17) [Vafeiadis17].

References

- [Breiman99] L. Breiman, “Random forests – random features”, Technical Report 567, Statistics Department, University of California, Berkeley, https://www.stat.berkeley.edu/~breiman/random-forests.pdf
- [Devroye96] L. Devroye, L. Gyorfi , and G. Lugosi. A Probabilistic Theory of Pattern Recognition. Springer, New York, 1996
- [Hotelling33] H. Hotelling, “Analysis of a complex of statistical variables into principal components”, Journal of Educational Psychology, vol. 24 pp. 417–441, 498–520, 1933
- [Liao15] C. Liao, P. Barooah, C.L.C. Liao, “An integrated approach to occupancy modeling and estimation in commercial buildings”, In: American Control Conference(ACC’15), pp. 3130-3135, 2015
- [Mahdavi09] A. Mahdavi, C. Proglhof, “Toward empirically-based models of peoples presence and actions in buildings”, In 11th International IBPSA Conference, pp. 537-544, 2009
- [Pearson01] K. Pearson, “On lines and planes of closest fit to systems of points in space”, Philosophical Magazine, vol. 6(2), pp. 559–572, 1901
- [Sha03] F. Sha, F. Pereira, “Shallow parsing with conditional random fields”, In Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL '03), Association for Computational Linguistics, Edmonton, Canada, pp. 134-141, 2003, http://dx.doi.org/10.3115/1073445.1073473
- [Vafeiadis17] T. Vafeiadis, S. Zikos, G. Stauropoulos, D. Ioannidis, S. Krinidis, D. Tzovaras, K. Moustakas, “Machine learning based occupancy detection via the use of smart meters”, International Conference on Energy Science and Electrical Engineering (ICESEE’17), Budapest, Hungury, 20-22 Oct. 2017.
- [Wang06] S. Wang, A. Quattoni, L.P. Morency, D. Demirdjian and T. Darrell, “Hidden Conditional Random Fields for Gesture Recognition”, Computer Vision and Pattern Recognition, vol. 2, pp. 1521-1527, 2006, http://dx.doi.org/10.1109/CVPR.2006.132
- [Yang03] D. Yang, H. Gonzales-Banos and L. Guibas, “Counting people in crowds with a real-time network of simple image sensors”, Proceedings of the International Conference onComputer Vision, pp. 122–129, 2003