UNIVERSITÀ DI PISA

DATA MINING PROJECT
Occupancy Detection

Marco Dalla Zanna (603549)

Andrea Cardia (540370)

Giulia Ferro (597681)

A.Y. 2019/2020

Summary

1 Data Understanding

The aim of this project is to detect the occupancy of an office room. The data provided to reach this goal are from three different data sets: data training, data test and data test2, containing respectively 8143, 2665 and 9752 records. These data sets include information about the occupancy of an office monitored during the month of February in 2015. Every minute, the sensors in the room recorded data about light, temperature expressed in Celsius degrees, percentage of humidity, CO_2 level and humidity ratio which is derived from the temperature and the relative humidity. These are the five numeric continuous variables of the data sets to which is added the feature *'date'* that provides the timestamp of every sensors readings with the following format:'Year-Month-Day Hours:Minutes:Seconds'. The last feature is the target class *'Occupancy'*. It is a boolean variable that takes on the value 1 if the room is occupied and 0 otherwise.
The data sets don't have missing values and looking at their temporal windows it can be seen that the data refer to three different but consecutive periods. The observations of the test set starts from the 2015-02-02 at 14:19:00 to the 2015-02-04 at 10:43:00, then the information in the training begin from the 2015-02-04 at 17:51:00 until the 2015-02-10 at 09:33:00. Thirdly, the time window in the test set2 goes from 2015-02-11 at 14:48:00 to 2015-02-18 at 09:19:00. So there are two temporal gaps between the observations of the data sets. They can be seen in the following graphs which shows the distribution of the *'Light'* values and the spaces that break the line highlight the interruptions of the sensor readings.

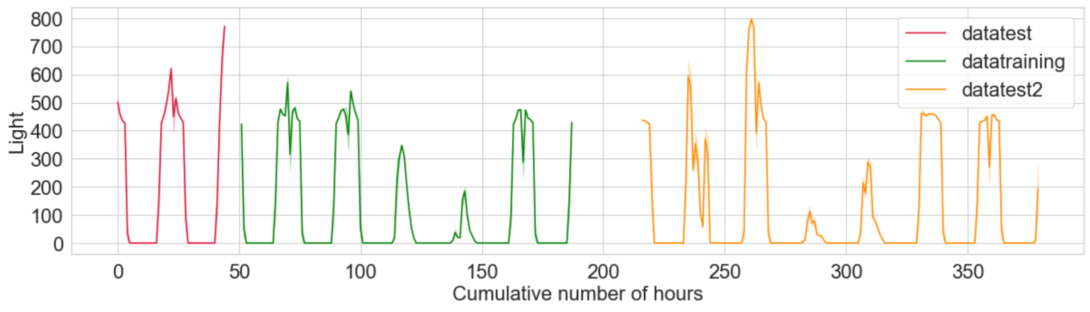


Figure 1: Time gaps

These temporal gaps are quite short-lived and the data detections belong to the same month, namely February 2015. That's why the three data sets are merged together.
The data preparation step starts by converting the data sets from a TXT to a CSV format and proceeds by cleaning the labels of the variables by removing the quotes from them and also from the feature *'date'* and the index. This last element is also transformed from a string to an integer value.
After the labels cleaning, the attention is focused on the definition of new variables. From the feature *'date'* the month and the year can be ignored because they have always the same values. From the *'date'* variable are created the following new features:

- Three numeric variables are defined from the breakdown of the date:
  - *'day'* represents the day number.
  - *'hour'* represents the hour of the day.
  - *'minute'* represents the minutes of the corresponding hour.
- *'day_minute'* containing the number of minutes elapsed by the beginning of each day. In this way, the specific minute during which is made the detection by the sensor can be precisely identify. The formula used to create this feature is the following: '$day\_minute' = (hour * 60) + minute$. This value ranges from 0 to 1440, which are the total minutes in a day. For example the first observation in the data test is taken at 14:19:00, so the respectively value of *'day_minute'* will be equal to $(14 * 60) + 19 = 859$.

- *'cumulative_hour'* represents the incremental number of hours elapsed from the beginning to the end of the tracking. To do so, the formula used is: $cumulative\_hour : (day * 24 + hours) - 113$. The range of the observations goes from 0 to 136 which is the total monitoring hours.
- *'cumulative_minute'* follows the same idea of *'cumulative_hour'*. It includes the incremental number of minutes elapsed from the beginning to the end of the tracking. As said before, the observations are made one every minute so the total number of minutes is equal to the number of records which is 20560. The formula is: $cumulative\_minute = cumulative\_hour * 60 + day\_minutes$.
- *'weekend'* is the last new variable obtained from the column *'day'*. It has the function to establish if the considered day is a working or non-working day. The feature is a boolean one, so it assumes value 0 for the days that goes from Monday to Friday and 1 when the days are Saturday or Sunday, during which the office was always empty.

The preparation task ends with the split of the data set into the training and the test set that will be used for the analysis. The split is made according the variable *'date'* and it selects the temporal window that goes from the first detection on the 02/02/2015 at 14:19 until the 14/02/2015 at 02:31, as training set and the remaining data, that continue until 18/02/2015 at 09:19, as test set. Therefore the training set represents the 70% of the data and on the other hand, the test set contains the remaining 30% of data.

2 Classification Task

Training set and test set are already available for the classification task from the data preparation step. Before the implementation of the algorithm all the variables not suited for the analysis were removed. This variables is *'date'*. When needed, the data is normalized with the Z-score distribution.
An important aspect that has to be analyzed is the role of the variables in the classifications and their connections with the target class. The goal of the analysis is to detect if the room is occupied, that's why in the following classification models, the features that aren't directly modified by the presence of an individual in the room are removed. So, in order to create models that can generalise the detection of the room occupancy, none of the variables with temporal information are used because they aren't directly modify by the presence of an individual in the room.
Another aspect that will be explored is how the performances change with combinations of only the attributes *'Temperature'*, *'CO_2'*, *'Humidity'* and *'HumidityRatio'* to check which features are essential for the classification task. These data are more reliable indicators because their values are directly modified by the presence of an individual in any generic room. In this way, they can help to create a more general predictive model that could be applied in different contexts. In some configurations also *'Light'* is used as attribute even if it isn't strictly influenced by the occupancy but it is more linked with the working hours schedule of the office. Moreover, in the following sections it can be seen that all the classifiers rely a lot on this variable to reach the best performances.

2.1 K-Nearest Neighbors

The first classifier implemented is K-Nearest Neighbors. The number of values in the training is 14392, from this value the estimated number of neighbours should be around 120. The performances of the algorithm are tested with the number of neighbors, in range 1 to 200, analyzing the trend of accuracy and F-measure which it can be seen in the following graphs.

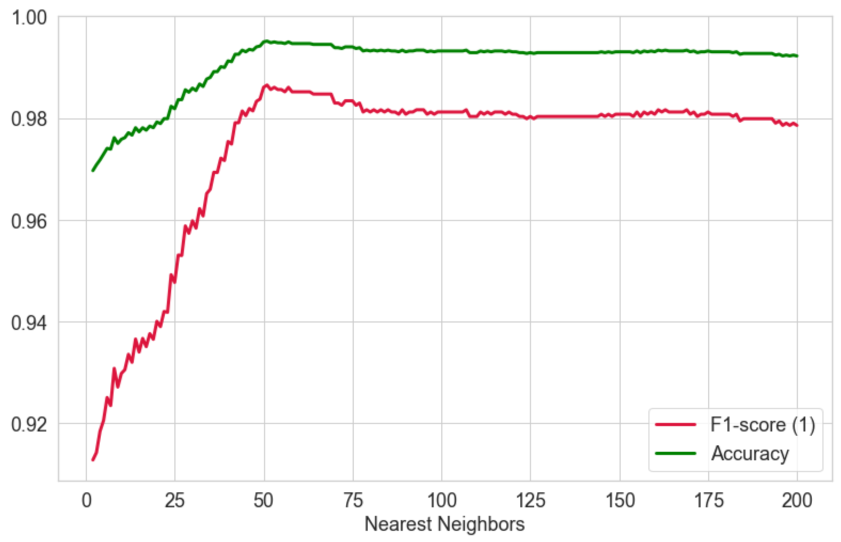


Figure 2: K-Nearest Neighbors performances

The results show high values in both metrics until a number of neighbours equals to 50. The accuracy ranges from 96% with one neighbor to 99% with 50, but considering that the distribution of the target class is slightly imbalanced, the F1-measure is a more reliable metric for the performance evaluation. In the figure, the red line represents the F1-measure for the instances that are labeled with 'Occupancy' equal to 1 (occupied room). This last one is the minority class but despite this, the F-score shows good results starting with a value of 90% with just a single neighbor until reaching a value of 99% with 50 neighbors. After 50 neighbours the performances decrease. These are the results given by all the variables as input. To check if these performances are heavily influenced by the temporal information, like already explained in the introduction of this section, the algorithm is tested changing the combination of features given as input. For every variables configuration is used the grid search to select the best parameters and the cross validation, where the best number of folds ranges from 3 to 50 selecting the one with the highest F-measure. The results are collected in Table 1. Due to space issues, in this table, features names are mapped by shortening their names with their starting character. For example, 'H' stands for *'Humidity'*, while 'HR' for *'HumidiRatio'*.
In all the configurations as metric is used the euclidean distance and all the points are weighted equally. In the table it can be seen that the best performance is given by the attributes *'Temperature'*, *'Light'* and *'CO_2'*. All the configurations with the attribute *'Light'* have an accuracy higher than 95% and a very high recall (always around 98-99%), while the level of precision for the minority class is still high but lower than the recall. This means that the classificators can correctly classify almost the totality of occupied rooms as occupied, but the lower precision is a consequence of the fact that not all the rooms classified as occupied are really occupied. The attribute *'Light'* allows to reach the best performances, but as already said, it is heavily linked with the working hours schedule and it isn't directly influenced by the occupancy level of the room. Removing the attribute *'Light'*, the best performances are given by the information about the temperature of the office and the level of humidity. The performances get worst by removing the attribute *'Light'*, but in this way the result comes from a model that is applicable in different contexts because it takes as input variables that directly depend on the presence of a person.

Attributes	**Neighbors**	**Cross Validation**			**Test performances**			
		Cv	*Accuracy*	F_1	*Accuracy*	F_1[1]	*Precision[1]*	*Recall[1]*
{T, L}	47	31	98.2%	97.9%	98.5%	95.8%	92.1%	99.9%
{T, C}	49	13	86.7%	82.4%	78.7%	48.8%	42.7%	56.9%
{T, H}	42	35	80.2%	74.0%	89.0%	73.9%	64.0%	87.3%
{T, HR}	49	46	81.3%	74.6%	90.3%	76.3%	67.7%	87.4%
{L, C}	31	43	98.7%	98.3%	98.4%	95.7%	91.8%	99.9%
{L, H}	49	45	98.5%	98.2%	98.0%	94.7%	90.1%	99.9%
{L, HR}	49	45	98.5%	98.5%	97.6%	93.7%	88.2%	99.9%
{C, H}	50	8	85.0%	80.4%	70.8%	37.0%	30.0%	48.2%
{C, HR}	50	43	84.8%	79.6%	70.0%	29.0%	25.0%	34.5%
{H, HR}	31	39	81.7%	75.3%	86.2%	67.5%	58.1%	80.4%
{T, L, C}	**31**	**43**	**98.1%**	**97.7%**	**98.9%**	**97.0%**	**94.3%**	**99.9%**
{T, L, H}	45	5	98.3%	97.8%	98.1%	95.0%	90.5%	99.9%
{T, L, HR}	23	5	98.1%	97.5%	98.1%	95.0%	90.6%	99.9%
{T, C, H}	50	47	84.3%	79.3%	75.5%	51.3%	39.7%	72.5%
{T, C, HR}	43	47	84.5%	79.8%	72.9%	48.4%	36.6%	71.4%
{T, H, HR}	45	46	81.3%	74.6%	90.0%	76.1%	66.2%	89.6%
{L, C, H}	34	43	98.3%	97.8%	96.8%	91.7%	84.8%	99.9%
{L, C, HR}	30	13	98.3%	97.8%	97.5%	93.3%	87.6%	99.9%
{L, H, HR}	49	45	98.4%	98.1%	97.8%	93.9%	88.6%	99.9%
{C, H, HR}	48	20	85.2%	81.3%	71.2%	36.0%	29.8%	45.5%
{T, L, C, H}	46	43	97.9%	97.4%	98.6%	96.2%	93.2%	99.3%
{T, L, C, HR}	44	44	98.1%	97.6%	97.7%	93.8%	90.0%	98.0%
{T, L, H, HR}	45	5	98.4%	97.9%	97.8%	94.1%	89.0%	99.9%
{T, C, H, HR}	50	49	84.6%	79.8%	74.7%	49.0%	38.2%	68.4%
{L, C, H, HR}	30	43	97.9%	97.3%	96.8%	91.6%	84.7%	99.8%
{T, L, C, H, HR}	45	5	98.0%	97.4%	97.8%	94.1%	89.9%	98.8%

Table 1: Nearest Neighbor Performances

2.2 Naive Bayes Classifier

The Naive Bayes algorithm is at first implemented with all the variables selected for the classification. The results present an accuracy of 89% and a F1-score of the minority class of 76%, that comes from a precision of 62% and a recall of 100%. Like with the KNN algorithm, the variables with the temporal information are removed and the Naive Bayes classifier is tested with different combinations of the following attributes: *'Temperature'*, *'Light'*, *'CO_2'*, *'Humidity'* and *'HumidityRatio'*. It's also used the cross validation technique with the number of folds that ranges from 3 to 49. The following table shows some of the results.

Attributes	**Cross Validation**			**Test performances**			
	Cv	*Accuracy*	F_1	*Accuracy*	F_1[1]	*Precision[1]*	*Recall[1]*
{H, HR}	5	77.3%	59.5%	51.7%	27.4%	18.7%	51.1%
{C, HR}	6	86.3%	78.0%	78.7%	62.5%	45.5%	100%
{T, H}	3	83.6%	74.6%	83.1%	43.3%	53.6%	36.3%
{C, H}	4	85.8%	77.0%	79.0%	62.9%	45.9%	100%
{T, L, H, HR}	46	96.0%	95.6%	97.8%	94.1%	89.0%	100%
{T, L}	48	95.8%	95.6%	97.9%	94.3%	89.3%	100%
{T, L, HR}	48	95.8%	95.6%	97.9%	94.3%	89.3%	100%
{L, H}	3	97.3%	96.5%	98.1%	94.8%	90.2%	100%
{L, H, HR}	3	97.3%	96.5%	98.1%	94.8%	90.2%	100%
{L, HR}	41	96.7%	96.4%	98.1%	95.0%	90.4%	100%

Table 2: Naive Bayes Classifier

As it can be seen from the table, the attribute *'Light'* is essential to reach high values of recall and at the same time, a good level of precision for the minority class. In some configurations, the model reaches a recall equal to 100%, meaning that every time the room is occupied (target class is equal to 1), the classifier is able to correctly classify every instance. When *'Light'* is not used by the classifier, high level of recall can still be reached, but in these cases the precision decreases significantly. For example, keeping the recall equal to 100%, the highest value of precision that can be reach is 45.9% and is given by the attributes 'CO_2' and *'Humidity'*. This level of precision implies that every time that the classifier predict the room as occupied, classifying instances to class 1, only the 45.9% of the predictions are correct.
Another interesting thing is that with this method, the attribute *'Humidity Ratio'* doesn't improve the performances of the classification. For example, when it is added to *'Temperature'* and *'Light'* or to *'Light'* and *'Humidity'*, the performances remain exactly the same so it doesn't bring any additional information gain.

2.3 Logistic Regression

The Logistic Regression is tested with every attributes using Ridge normalization. The following table shows the performances of the model and the logarithmic loss.

Attributes	**Accuracy**	**F_1**	**Precision[1]**	**Recall[1]**	**log error**
'Temperature'	82.6%	28.0%	53%	19%	6.01
'Humidity'	82.2%	0.0%	0%	0%	6.14
'Light'	**99.2%**	**97.9%**	**96%**	**100%**	**0.26**
'CO_2'	76.0%	60.0%	43%	100%	8.29
'Humidity Ratio'	82.2%	0.0%	0%	0%	6.14
'Hour'	82.2%	0.0%	0%	0%	6.14
'Minute'	82.2%	0.0%	0%	0%	6.14
'Day Minute'	82.2%	0.0%	0%	0%	6.14
'Weekend'	82.2%	0.0%	0%	0%	6.14

Table 3: Logistic Regression Performances

Like in the previous classification algorithms, the best performance is given by the feature *'Light'* which presents the highest F1-score and the lowest logarithmic loss.

The two following graphs show how, with the *'Light'* attribute, the algorithm can successfully divide the observations assigning all the records with a light value lower than 350 to the target class 0 and the remaining to the minority class. On the other hand, the other features can't correctly classify the records and that's why they present higher errors and lower (if not null) value of F1-score. For example, in Figure 8 with the attribute *'CO_2'*, the Logistic Regression misclassified as class 1 a lot of records when the 'CO_2' level is higher than 800.

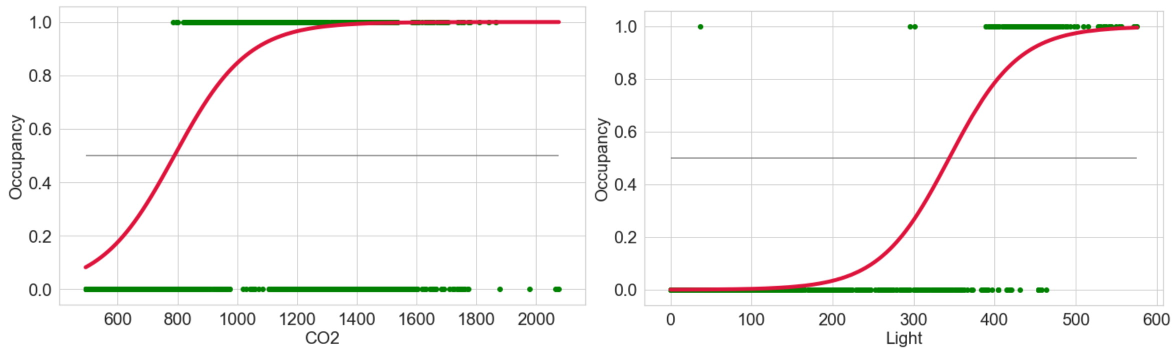


Figure 3: Graphical representations of *'CO_2'* and *'Light'* performances

2.4 Decision Tree Classifier

For the Decision Tree algorithm the following parameters are tested with the grid search:

- As splitting criterion are used Gini index or Entropy.
- The minimum number of values to make a split is tried on the following numbers: 10, 15, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 130, 150, 170, 190 and 210.
- The minimum number such that a node can be a leaf takes the following values: 10, 15, 20, 30, 35, 40, 45, 50, 55, 60, 65, 70, 80 and 100.

The results of the classification with the attributes *'Temperature', 'Light', 'CO_2', 'Humidity'* and *'HumidityRatio'* show an accuracy of 95% and for the target class *'Occupacy'* equal to 1 an F-measure of 86% coming from a precision level of 90% and a recall of 83%. The best parameters selected by the grid search from the list above are: Entropy as splitting criterion, a number of minimum samples leaf equal to 10 and minimum sample spit of 100.
Also this classifier is tested with different variables combinations. The best performance is given by the features *'Light'*, *'CO_2'*, and as input, suggested by the grid search, Gini index as split criterion, minimum samples leaf of 20 and minimum samples split of 150. The model has an accuracy of 99%, regarding the target class equal to 1, an F-score of 99%, that comes from precision and recall level of 98%. As it can be seen in the Decision tree figure, the use of the *'Light'* attribute helps the classifier to make defined splits with low values of Gini index, which means low probability of missclassifying an observation.

Attributes	**Cross Validation**			**Test performances**			
	Cv	*Accuracy*	F_1	*Accuracy*	$F_1[1]$	*Precision[1]*	*Recall[1]*
{C, HR}	41	84.0%	79.0%	65.2%	21.8%	18.2%	27.3%
{T, HR}	49	80.5%	74.0%	91.0%	78.0%	68.8%	90.1%
{T, L, C, H}	43	98.4%	98.1%	95.3%	86.2%	89.7%	83.0%
{T, C, H, HR}	34	86.3%	83.1%	82.0%	64.5%	49.6%	92.3%
{L, C}	**37**	**98.6%**	**98.3%**	**99.7%**	**99.1%**	**98.4%**	**99.9%**

Table 4: Decision Tree Classifier performance

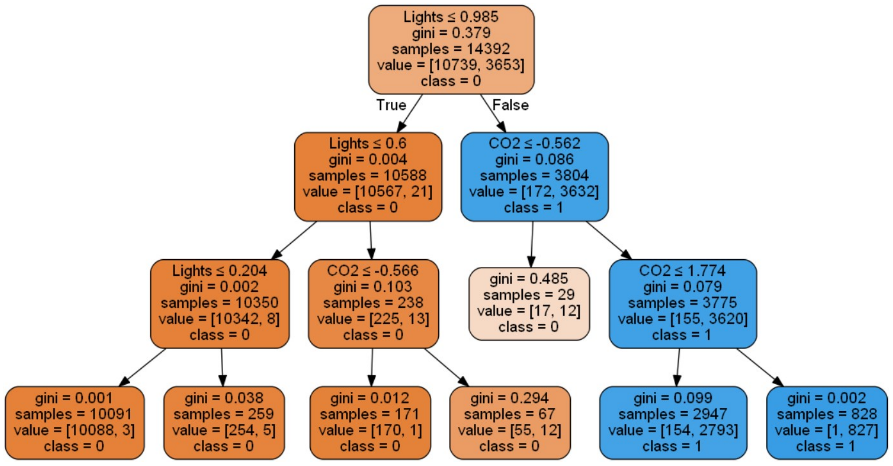
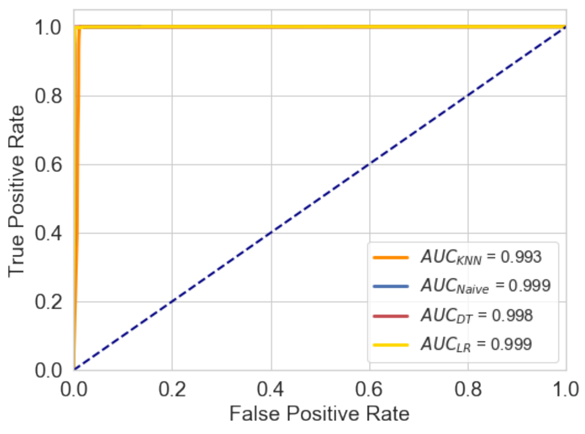


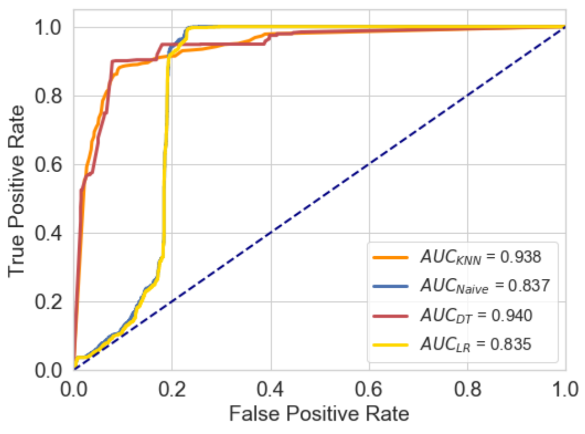
Figure 4: Decision Tree

3 Classificators Summary

To have an overall view of the performances given by classification models just implemented, in the following graph there are the ROC curves of every classifier. On the left graph there are the ROC curves that come from the classifiers with the best F1-score of the minority class. On the right side, there are the ROC curves from the implementation with the best F1-score of the minority class without the *'Light'* attribute. The comparison between these two graphs highlight the crucial information gain given by the variable *'Light'*.



(a) ROC curves with attribute *Light*



(b) ROC curves without attribute *Light*

Figure 5: ROC curves

An interesting factor is that the best model on the left graph, that are Naive Bayes and Logistic regression, are the worst in the right graph. This means that the performances of the other two models, that are K-nearest Neighbor and Decision Tree, are less affected by the absence of *'Light'*. In the following figures there are the Lift charts of the same models presented with the ROC curves. Also these graphs are used to make the comparison between the performances of the models with and without the attribute *'Light'* to highlight the importance of this variable in the classifications.

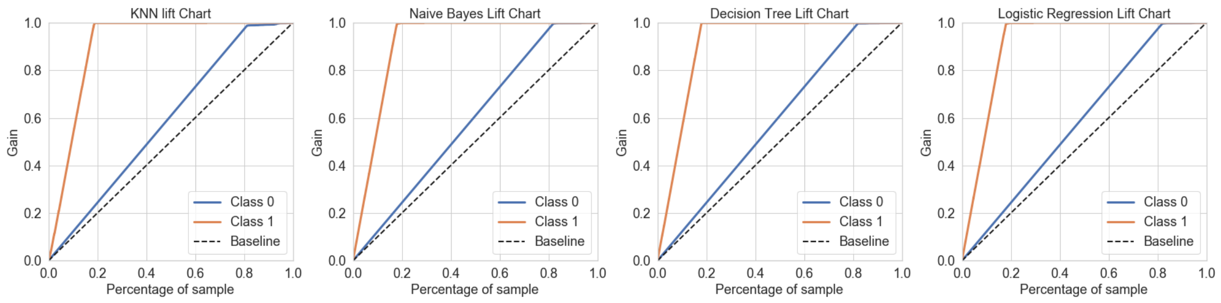


Figure 6: Lift charts with the attribute Light

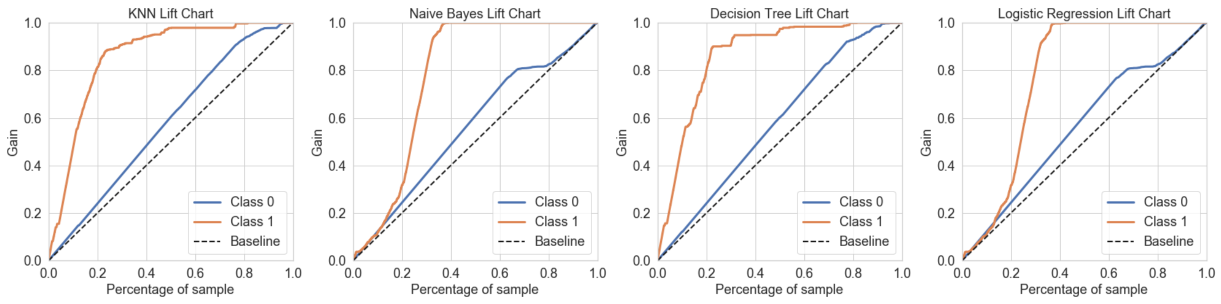


Figure 7: Lift charts without the attribute Light

4 Dimensionality Reduction

An attempt of dimensionality reduction of the data set through features elimination has already been made during the previous classifications task where the algorithms were implemented with different compositions of variables. The selection of variables is a way to reduce the dimensions of the data set that are employed in the analysis and it was done to understand how the performances of the classification methods change with different features combinations.

4.1 PCA

Another possible way to reduce the dimensionality is through features extraction techniques like principal components analysis. After the data normalization with the Z-score distribution, the PCA returns the values of the nine principal components that represent the new nine dimensions of the data set. The following graph shows the values of the principal components and therefore the quantity of the variance (information gain) that can be caught even by reducing the data set to that number of dimensions.

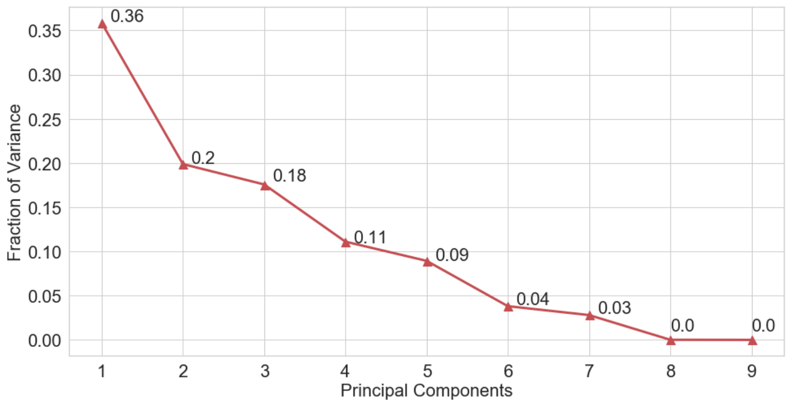


Figure 8: PCA

The first part of the graph doesn't present a very rapid decrease, which means that more principal component are needed to extract a reasonable quantity of variance that is needed to have a good level of information. For example, the first four principal components return the 84.42% of the data variability, choosing only these first four dimensions will cause a loss of the remaining 15.58% of variability which represents information gain that is left out from the analysis. Using only the first three principal components to implement a KNN model, the performances decrease. For example, the value of the F1-score for the minority class changes from 97% to 72% especially due to the reduction of the precision that goes from 97% to 59%. In addition to dimensionality reduction, the PCA can be useful for data visualization purposes. Selecting the first two principal component is equal to consider the two dimensions with the higher variability of the data. The same concept goes with 3 principal components. Doing so, the data can be represented by the following graphs.

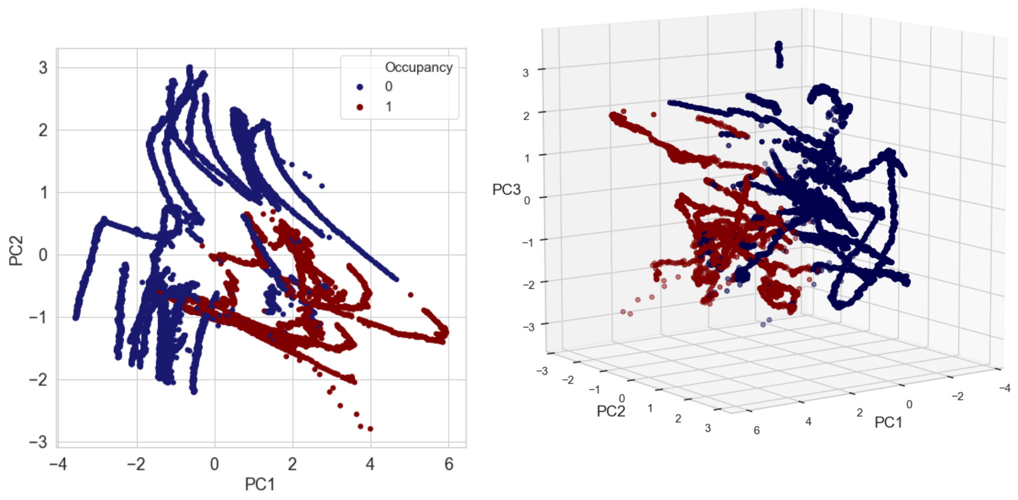
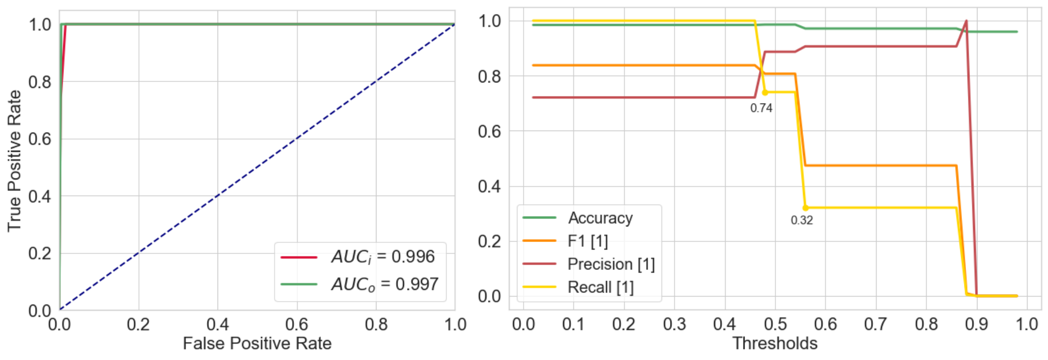


Figure 9: PCA graphical representations

5 Classification with imbalanced data set

In the training set, the initial composition of the target variable *'Occupancy'* was 10739 samples belonging to class 0 (74.62%) and 3653 belonging to class 1 (25.38%). The respective frequency on the test set was 5071 for class 0 (82.12%) and 1097 for class 1 (17.79%). The imbalanced version of these data sets were created by removing records from the minority class, respectively 3205 records from the training and 885 from the test. Doing so, the distribution of the classes presents 4% of values belonging to class 1 and the remaining 96% to class 0. The total number of samples is 11187 for the training and 5283 for the test.
After data normalization, the performances of the Decision Tree algorithm are tested in both datesets versions. In both cases the input variables given to the classifier are: *'Temperature'*, *'Light'*, *'CO_2'*, *'Humidity'* and *'HumidityRatio'*. The parameters used are the best ones indicated by the grid search, that are the number of minimum samples leaf equals to 100 and minimum samples split equals to 10. The results from the original data set are an accuracy of 99.5% and a F1-score of 98.6%. Instead, for the imbalanced data set the accuracy is 98.5% and the F1-score is 79.8%. As expected, the performances decrease in the imbalanced version of the data set. This can also be seen from the following graph, where the ROC curve of the original data set is higher. The AUC value in the original data set is a bit higher, which means that in the imbalance version, the model struggles more in the classification but despite this, the performances are still high.



(a) Imbalanced and original data set ROC curves

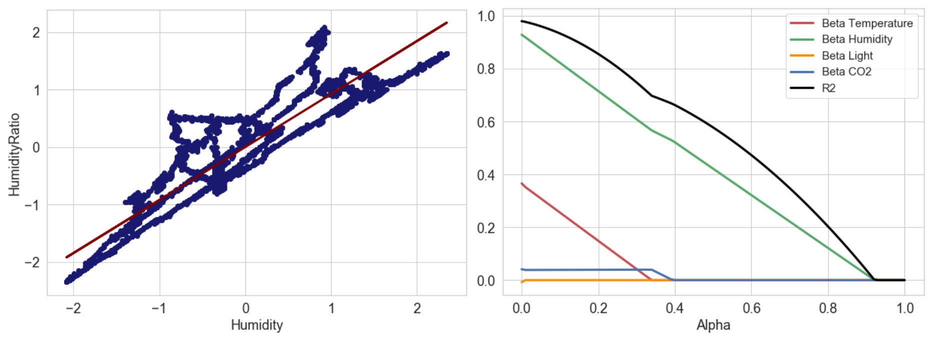
(b) Influences of different thresholds

Figure 10

To improve the performances in the imbalanced data set we used a decision threshold ranging from 0 to 1 with step of 0.2. The classification results improve with a threshold lower than 0.48. However, beyond this value, the model starts to increasingly misclassify real value of 0 as 1. This can be seen through the sharp decrease of recall with a threshold equals to 0.56, and later on equals to 0.88, when the model classify all instances as 0.

6 Simple & Multiple Linear Regression

After the data normalization with Z-score distribution, the linear regression is tested with all the configurations between the variables without temporal information. The performances were significantly low. The only model with sufficient results comes from the regression between *'Humidity'* and *'HumidityRatio'* that it can be seen in the left graph below.



(a) Simple Regression

(b) Multiple Regression Beta's with different levels of Ridge regularization

Figure 11

For the multiple linear regression the feature *'HumidityRatio'* is set as dependent variable and the results are following: the coefficient of determination is 0.98, the mean squared error is equal to 0.02 and the Mean absolute error is 0.111. Two different data regularization methods are tested on this regression model. With the Ridge regularization the performances don't change. As shown in Figure 9, with the Lasso regularization, different values of alpha are tried, to see how variables are penalised and how the performance changes as penalisation changes. The best regression is obtained with no penalisation and alpha equals to 0, but when alpha gradually increases, the performances decrease in an almost linear trend.

7 Support Vector Machines

7.1 Linear SVM

After data normalization, the Linear SVM model is implemented with different combinations of variables. The parameters tested in the grid search are values of C ranging from 0.01 to 3, with step 0.01. The best model uses *'Temperature'*, *'Light'* and *'CO_2'* as attributes, and the value of C is equal to 0.06. This model performs with an accuracy of 99.1%, a F1-score, with respect to the minority, of 97.6%, a precision of 95.4% and a recall of 100%.

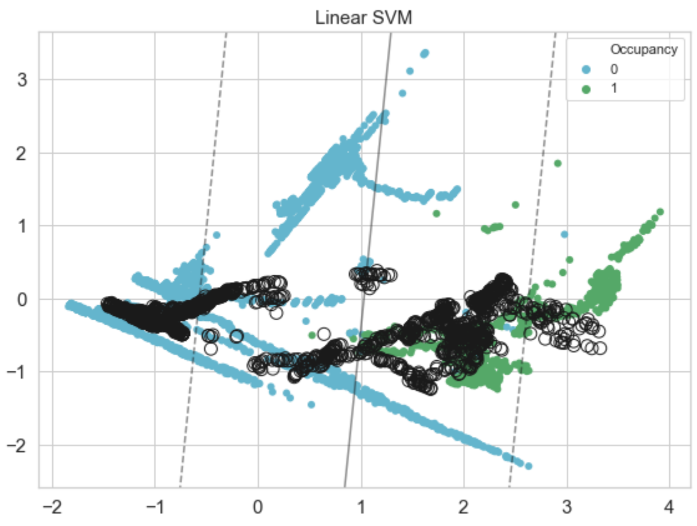


Figure 12: Linear SVM

From the Figure 12 it can be seen that the linear model struggles to divide the instances, that's why the non linear version of SVM is implemented in the following section.

7.2 Non Linear SVM

Trying to improve the performances, the non linear SVM model is implemented in the same way as the linear SVM, with the only exception that kernels are added as parameters in the grid search. Kernels tested are polynomial, rbf and sigmoid. The best model found with non linear SVM is the one that uses *'Temperature'*, *'Light'* and *'CO_2'* as attributes, the kernel is rbf and the C parameter is equal to 0.06 as parameter, like the linear counterpart. However, the performances are slightly higher: accuracy is now 99.17%, f1-score 97.72%, precision 96% and recall of 100%.

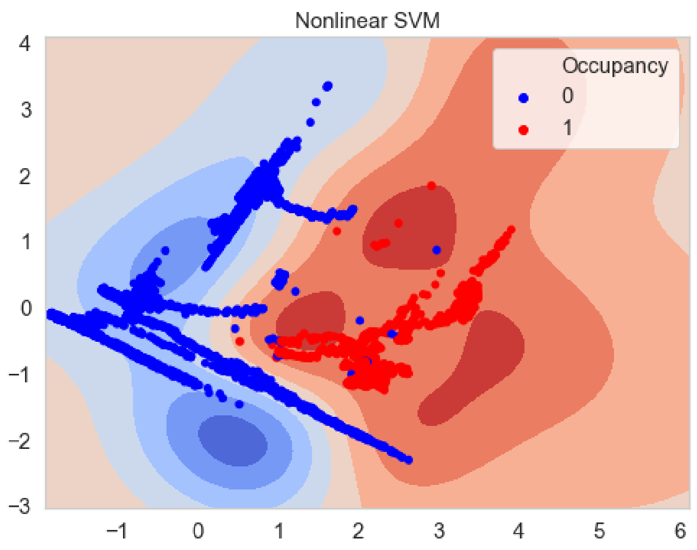


Figure 13: Non Linear SVM

8 Neural Network

8.1 Multi Layer Perceptron

After data normalization, the Neural Network model is tested with all the configurations (that are 4320) of the following parameters:

- As activation functions are tested: Identity, Logistic, Hyperbolic Tangent and Linear.
- As number of input in the layers the following values: (50,), (100,), (50, 100,), (50, 150,), (50, 100, 150,), (50, 100, 200,)
- As solvers are tested L-BFGS, the Stochastic gradient descent and Adam.
- As learning rate are tried: constant, invscaling and adaptive.
- The momentum value takes on the values 0, 0.3, 0.6 and 0.9.

The results of the combinations are ordered to find the best F1-score for the minority class. The best model is the first in the Table 5. Since this model has an high loss value respect to the others, it is also tested on the training set with the same parameters and the results are way lower that the one on the test set, which means that the model is underfitting the training data. To solve this problem, the best configurations of parameters already found are retested on the training to find a model with high and balanced performances. The results show that only the first model is underfitting the training data while the others present good performances in both training and test set. The best performances on the training and test set are given by 12 models, one of which can be seen in second row of Table 5. In these 12 models only two parameters change: the learning rate changes between constant, invscaling and adaptive and the momentum values are 0, 0.3, 0.6 and 0.9. Even with different configurations of these parameters the models have the same results.

Parameters					**Performances**				
Activation	*Hidden Layers*	*Solver*	*Learning rate*	*Momentum*	*Accuracy*	*F1[1]*	*Precision[1]*	*Recall[1]*	*Loss*
Relu	(50, 100, 200)	Sgd	Invscaling	0.9	99.6%	98.8%	99%	99%	0.36940
Identity	(50, 150)	Adam	Constant	0	99.5%	98.5%	97%	100%	0.065651

Table 5: Neural Network Implementation

The following loss curves show the difference between the model that has underfitting and the model with high performances in both training and test. The difference between these two loss values is the reason why the model is also tested on the training.

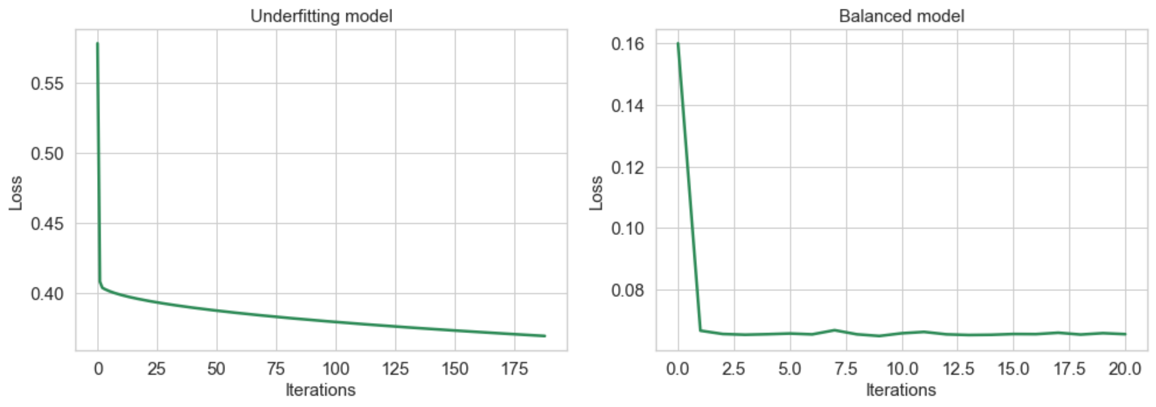


Figure 14: Loss curve Neural Network

8.2 Deep Neural Network

After data normalization, the implementation of the Deep Neural Network starts by testing different models with a total of five layers. The nodes for the input layer are chosen from the number of variables, which are five, plus one additional as bias. For the activation functions are tested all the combinations between Softsign, Linear, Relu, Tahn, Softplus and Sigmoid in every layer except the output layer in which is always used the Sigmoid function. The total number of models is 625. Giving the high number of parameters configurations, an early stopping criterion is chosen. The maximum number of epochs is 30 but the building of the model stops when the value of the loss doesn't decrease for 2 consecutive epochs. Different weights are chosen to help the model with the slightly imbalanced classes, for the target class equals to 0 the weight is set to 0.85, while for the minority class is set to 0.15.

Models	**Parameters**										
	Input Layer		*Hidden Layers 1*		*Hidden Layer 2*		*Hidden Layer 3*		*Output Layer*		*Solver*
	Nodes	*Activation*	*Nodes*	*Activation*	*Nodes*	*Activation*	*Nodes*	*Activation*	*Nodes*	*Activation*	
Model 1	6	Softsign	9	Linear	12	Linear	3	Linear	1	Sigmoid	Adam
Model 2	6	Linear	9	Relu	12	Tanh	3	Tanh	1	Sigmoid	Adam
Model 3	6	Softsign	9	Linear	12	Linear	3	Tanh	1	Sigmoid	Adam
Model 4	6	Tanh	9	Tanh	12	Relu	3	Softsign	1	Sigmoid	Adam

Table 6: Deep Neural Network characteristics

Models	**Training performance**					**Test performance**				
	Accuracy	*F1 [1]*	*Precision[1]*	*Recall[1]*	*Loss*	*Accuracy*	*F1 [1]*	*Precision[1]*	*Recall[1]*	*Loss*
Model 1	96.8%	93.5%	96.1%	91.1%	0.0261	99.7%	99.1%	98.3%	99.8%	0.0616
Model 2	98.6%	97.4%	95.6%	99.2%	0.0264	99.4%	98.4%	96.9%	99.9%	0.0519
Model 3	98.6%	97.4%	95.6%	99.2%	0.0265	99.4%	98.4%	96.9%	99.9%	0.0519
Model 4	98.6%	97.4%	95.6%	99.3%	0.0262	99.4%	98.4%	96.9%	99.9%	0.0519

Table 7: Deep Neural Network results

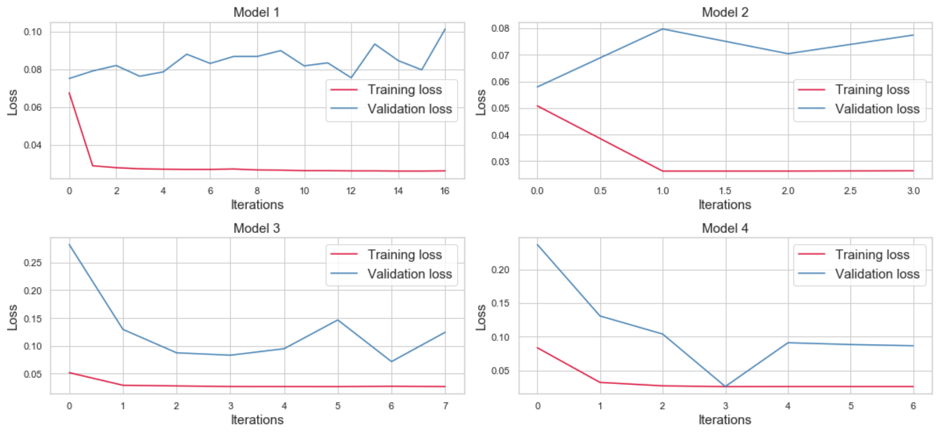


Figure 15: Loss curve DNN

9 Ensemble methods

9.1 Random Forest

Since Random Forest algorithm tries different combinations of features when building trees, no preliminary attribute selections were made. Therefore the implementation is made using all the attributes except the temporal information and the best parameters are found by the random search. The parameters that return the best model in terms of performances are: maximum depth set to 4, maximum number features set to 3, minimum sample leaf set to 20, minimum sample split set to 100 and the number of estimators equals to 100.
The results return an accuracy level of 99.51% and an F1-score of 98.65% for the minority class, coming from a precision of 97.59% and a recall of 99.73%. In the following graphs it can be seen the features that have more importance in the training and in the test set.

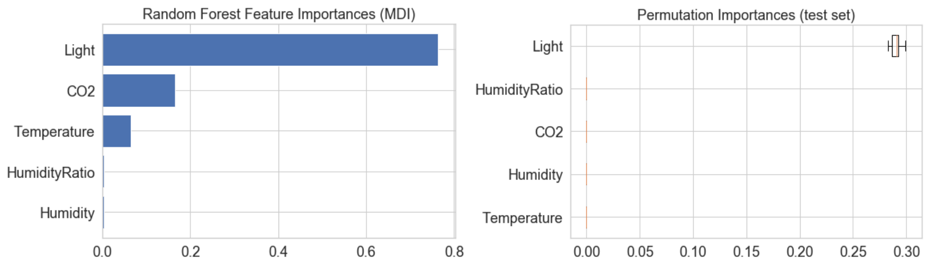


Figure 16: Features Importance

Like in the previous algorithms, the *'Light'* attribute brings the main contribution to the model implementation.

9.2 Bagging

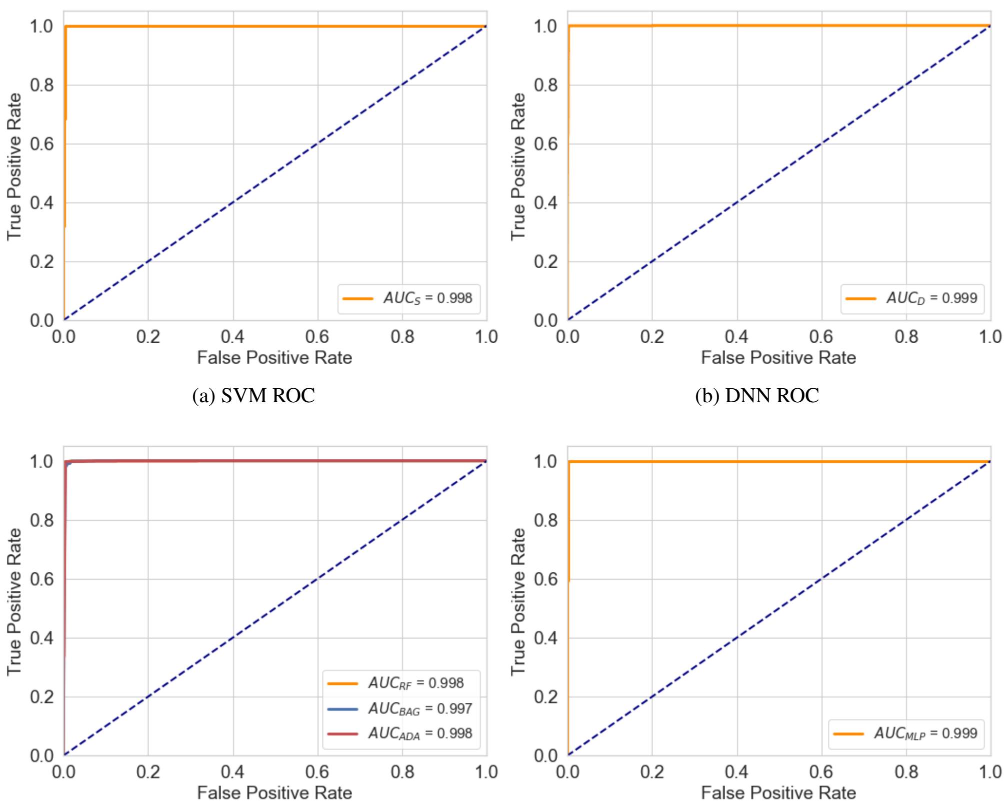
Before testing the Bagging method, the performances of the Decision Tree used as base classifier in the ensemble method are checked. The results of the Decision Tree with depth equals to 1 are very high and the ensemble method that use this classifier can't improve the performances due to the fact that the high accuracy of the base classifiers results prevents the application of the wisdom of the crowd principle that is the key aspect of ensemble methods.
To check if the Bagging method can improve the results, the classifiers are tested without the feature with the highest importance which is *'Light'*. The performances of the simple classifier without *'Light'* are: an accuracy of 45.62%, a F1-score of 40%, a precision 25% and recall 100% for the minority class. The Bagging method with the Decision Tree and a number of estimators equals to 18 gives the following results: the accuracy is 60%, the F1-score is 39%, precision of 27% and recall of 72%. Comparing the results, the ensemble method is able to improve the accuracy of the base classifier.
With the same principle explained above, also the Linear Support Vector Machine is used as base classifier with and without the variable *'Light'*. With all the attributes the ensemble doesn't improve the performances, so the method is tested without *'Light'*. The performances of the SVM with C equals to 1000 are: an accuracy of 67.96%, a F1-score of 53%, a precision 36% and recall 100% for the minority class. These performances are improved by the Bagging with 10 base classifiers and the results are the following: an accuracy of 71.33%, a F1-score of 55%, a precision 38% and recall 100% for the minority class.
Keeping all the variables, the Bagging method is also tested on another ensemble classifier which is the Random Forest. As base classifier are tested different Random Forest with different configurations of variables. Bagging improves the performances of all the models except the best model found by the Random Forest where the results remain the same.

9.3 AdaBoost

The same pattern followed with Bagging is tested here. Also AdaBoost can't improve the performances with the attribute *'Light'*, but it can improve the one without this attributes. This ensemble method is tested with a Decision Tree as base classifier with a number of estimators equals to 100 and presents an accuracy level of 64.77%, a F1-score of 42%, a precision of 30% and recall of 72% for the minority class.
Looking at the performances of the two ensemble both can improve the results without the attribute *'Light'* but the highest improvement is made by AdaBoost.

10 Classificators performances

In this section are presented the ROC curves and the Lift charts of the best model obtained before.



(a) SVM ROC

(b) DNN ROC

(c) Ensemble ROC

(d) MLP ROC

Figure 17: ROC curves

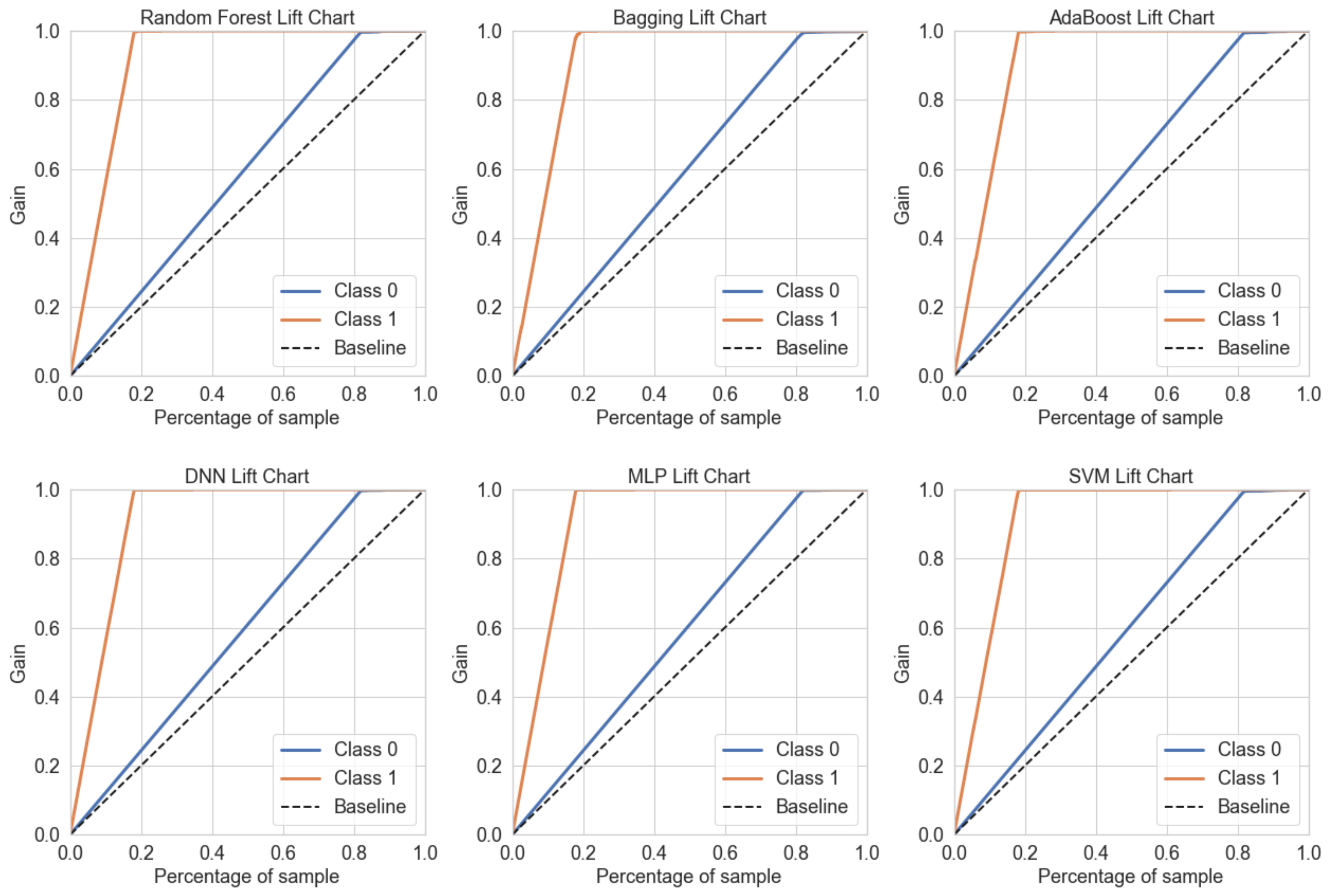


Figure 18: Lift charts

11 Time series

Time Series Analysis tasks are performed with three different data preparations. For clustering, 12 time series are created for the whole data set, each one with one day length. The selected days are: Tuesday 3^{th}, from Thursday 5^{th} to Sunday 9^{th} and from Thursday 12^{th} to Tuesday 17^{th}. These 12 days are chosen because they have a complete 24-hour observation. The resulting data set is composed by 12 time series of length 1440, which are the number of minutes in a day. The values of the target class are obtained by the values of the attribute *'weekend'* created in the data preparation task, which for this purpose has the meaning of *'Occupancy'*. For time series Classification, a new data set is obtained by selecting the same days from time series clustering, and dividing them into four six-hours time series each. From this subdivision is created a data set with 48 time series with length 360 minutes. Values of *'Occupancy'* are obtained by selecting the majority class in the six-hour time range. The data set created for time series forecasting is obtained from the file 'datatraining.csv', and time units frequency is converted from minutes to hours, resulting in a unique time series of length 126 hours.

11.1 Clustering & Similarity

For the clustering task, were implemented algorithms with different metrics, applied respectively on the variables 'CO_2' and '*Humidity*'. The algorithms were tested on the original and normalized data in both cases without noise. The aim of the analysis was to verify if the algorithms are able to identify clusters that can divide the working days from the weekends relying on the shape of the time series. After checking the compositions of the clusters, this goal was better reached by the non normalized data. The results can be seen in Table 8 which shows the structure of the clusters and the inertia values. For the implementation of the K-Means algorithm, the number of clusters was set initially equal to two to check if the division between working and not working days was achievable. After that, the number of clusters was increased to three to understand which time series have an unusual pattern that keeps the algorithm from generating the expected clusters compositions. The best configuration is given by the feature '*Humidity*', with three clusters and the Dynamic Time Warping as metric (see Table 8). In this case, changing the number of clusters allows to correctly identify three non working days that are put together in the same cluster (label 2).
The DBSCAN algorithm identifies only one cluster dividing it from the other time series that are considered noise and assigning them the label -1. In this case the compressed versions of the time series are used, and the pairwise distances as metric. With both variables the value of sample in the neighborhood is set to 4 and the radius is equal to 0.999485 for 'CO_2' and 0.9995 '*Humidity*'. With both features the algorithm identifies three time series as noise and among these the only one that is always assigned to the label -1 is Tuesday 3^{rd}, which is highlighted in green in Table 8.

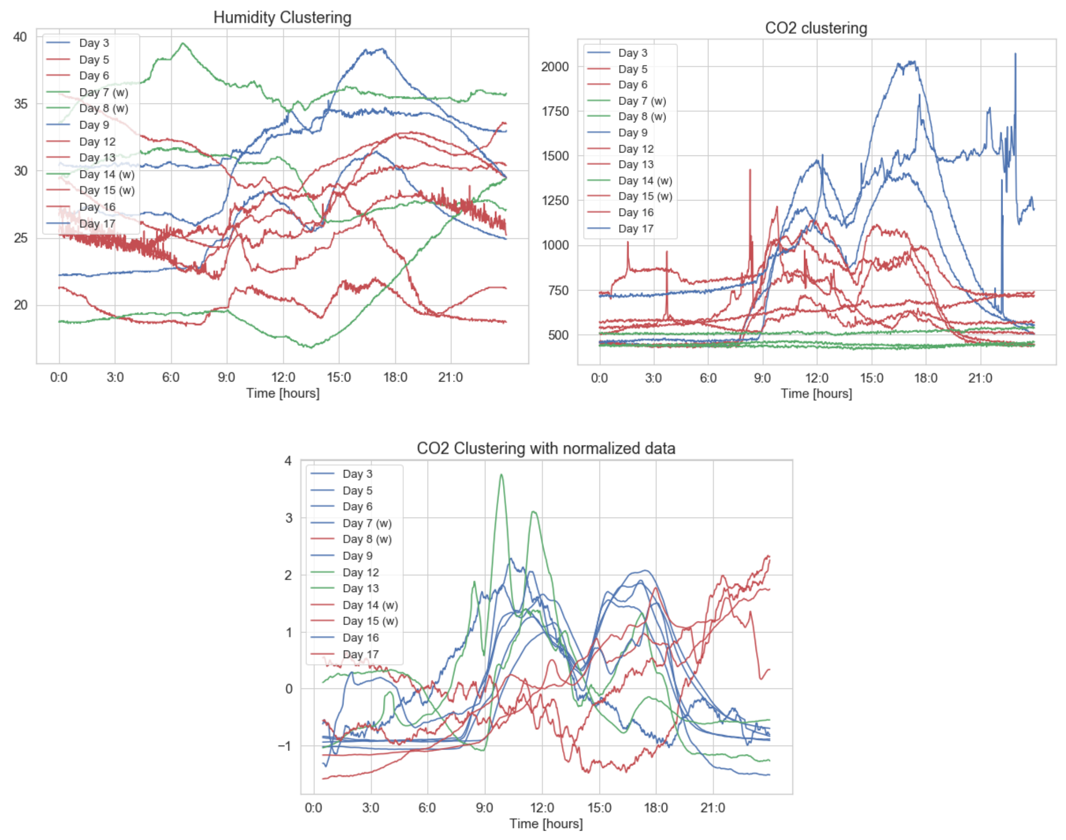


Figure 19: Time series clusters with K-means (metric = DTW, K = 3)

	Humidity Clustering Configuration							**CO2 Clustering Configuration**						
	KM Euclid		*KM DTW*		*KM Feat*		*DBSCAN*	*KM Euclid*		*KM DTW*		*KM Feat*		*DBSCAN*
	K=2	*K=3*	*K=2*	*K=3*	*K=2*	*K=3*		*K=2*	*K=3*	*K=2*	*K=3*	*K=2*	*K=3*	
Cluster Labels														
Tue 3rd	0	0	0	0	1	0	-1	0	0	0	2	0	2	-1
Thu 5th	1	1	1	1	1	0	0	0	2	0	2	0	2	0
Fri 6th	1	1	1	1	1	1	0	0	2	0	2	0	2	0
Sat 7th	1	2	1	2	1	1	0	0	2	0	2	0	2	0
Sun 8th	1	2	1	2	1	1	-1	1	0	0	0	1	0	0
Mon 9th	0	0	0	0	0	2	0	1	0	1	0	1	0	0
Thu 12th	1	1	1	1	1	1	0	0	2	0	2	0	2	0
Fri 13th	1	2	1	1	1	1	0	0	0	0	0	0	0	0
Sat 14th	1	2	1	2	1	1	0	1	1	1	1	1	1	0
Sun 15th	1	1	1	1	1	1	0	1	1	1	1	1	0	-1
Mon 16th	1	1	1	1	1	1	0	0	0	0	0	1	0	-1
Tue 17th	0	0	0	0	1	0	-1	1	1	1	0	1	0	0
Inertia	16632	11877	3267	1039	1473	981	-	42609813	29201154	8627495	2746691	$1.9x10^{10}$	$1.9x10^{10}$	-

Table 8: Time Series Clustering

The three graphs in the Figure 19 show the best clusters configuration with the attributes *'Humidity'* and the attribute 'CO_2'. One of the 'CO_2' graph shows the original data and the other shows the normalized version with the same parameters as input. In these two versions the compositions of the clusters change but the best distinction between working and not working days is given by the not normalized configuration.

11.2 Forecasting

Before starting the Forecasting Task, the time series stationarity is tested through the analysis of the Autocorrelation and Partial Autocorrelation, and by performing the Dickey-Fuller test. As the figures below shows, the transformed time series does not show a particular trend, while a daily seasonality is present. Also the Dickey-Fuller test statistics, which is -4.71 for the transformed time series, confirms that it is stationary with a confidence of 99%, whose critical value is -3.48.

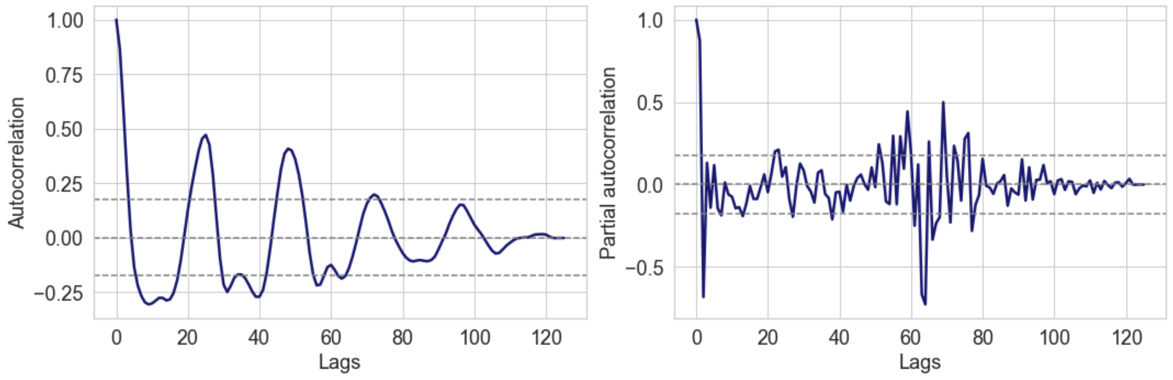


Figure 20: 'CO_2' Autocorrelation and Partial Autocorrelation Function

To test the forecasting algorithms, we split the data set into a training set composed by approximately four days (108 hours), and a test set composed by approximately one day (29 hours). Time Series Forecasting is started with Simple Exponential Smoothing and the Holt's Linear Trend, which both resulted in very poor performances. R^2 score are -0.03 and -0.01 respectively. However this was expected since the time series has no particular trend but a quite clear seasonality. In fact, better performances are obtained with the Holt-Winter Seasonal method. Different parameters are tested, and the best performance is obtained with seasonal periods set to 48, trend and seasonal set to 'additive'. This model returned an R^2 of 0.726, an MAE of 0.011 and an RSME of 0.014. The performances of Arima and Seasonal Arima models are in line with the models above. When Arima model starts to forecast future values of 'CO_2', it simply adds a straight line at value 0. In fact its best scores are 0.083 of R^2, 0.019 of MAE and 0.026 of RMSE, found with p, d, q parameters set to 7, 1, 1. However, when adding the parameters to model also the seasonality of the time series, performances improve significantly. After testing various parameters, the best performances are reached with the same parameters for modeling the level and trend part of the time series, and 6, 1, 3, for p, d, and q respectively, to model the seasonal part, with a seasonality set to 24. The score of this model is 0.76 of R^2, 0.011 of MAE and 0.013 of RMSE.

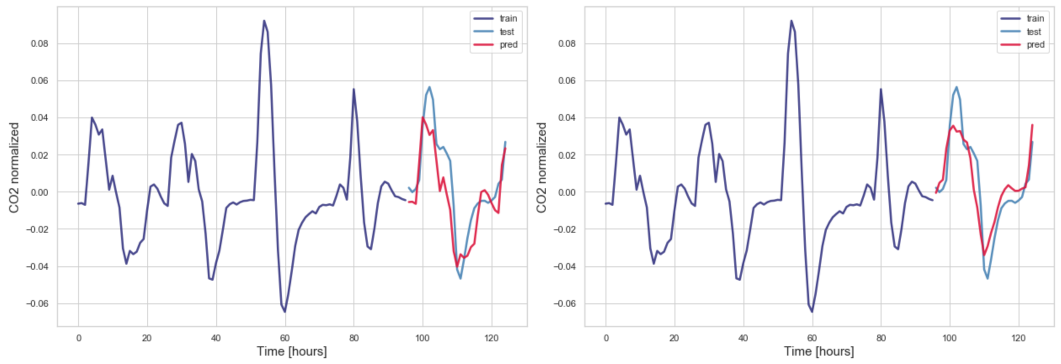


Figure 21: Simple Exponential smoothing (left) and Arima model (right) modelling seasonality

11.3 Shapelets & Motifs

11.3.1 Shapelet classification

For the classification task with the shapelets the 48 time series are divided in 33 for the training and 15 for the test. For this task, 'CO_2' is the chosen variable. After different parameters setting, the best results are found with the following parameters: the number of shapelets is 6 (3 of length 108 and 3 of length 116), the optimizer used is adam and the value of the weights is 0.01. With this configuration the model reaches an accuracy of 93.33% and for the minority class an F1-score of 85.71%, coming from a precision of 100% and a recall of 75%.

Real values	0	1	1	0	0	0	0	0	1	0	0	0	0	1	0
Predicted values	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0

Table 9: Shapelets Classification on Test set

In the following graphs it can be seen the shapes of the shapelets that better represent the classes of the time series.

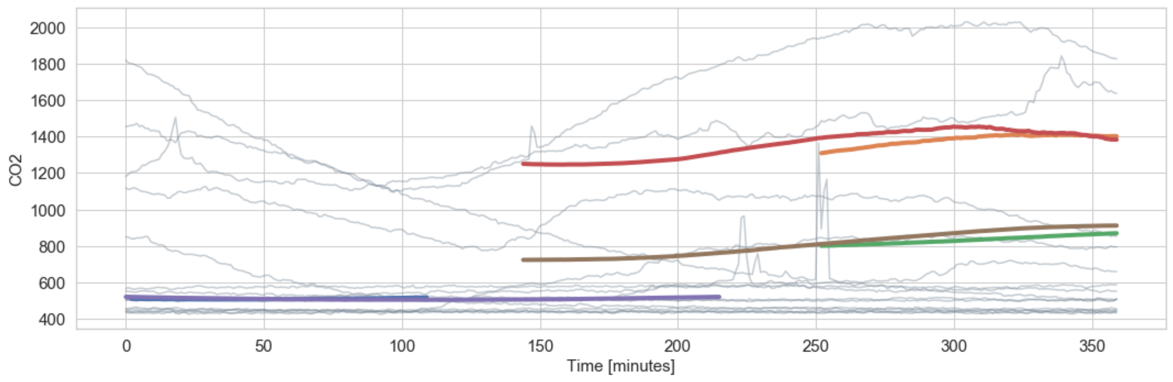


Figure 22: Shapelets on a 6-hours time series

In the next graphs it can be seen another type of classification with 4 shapelets made from the 12 daily times series.

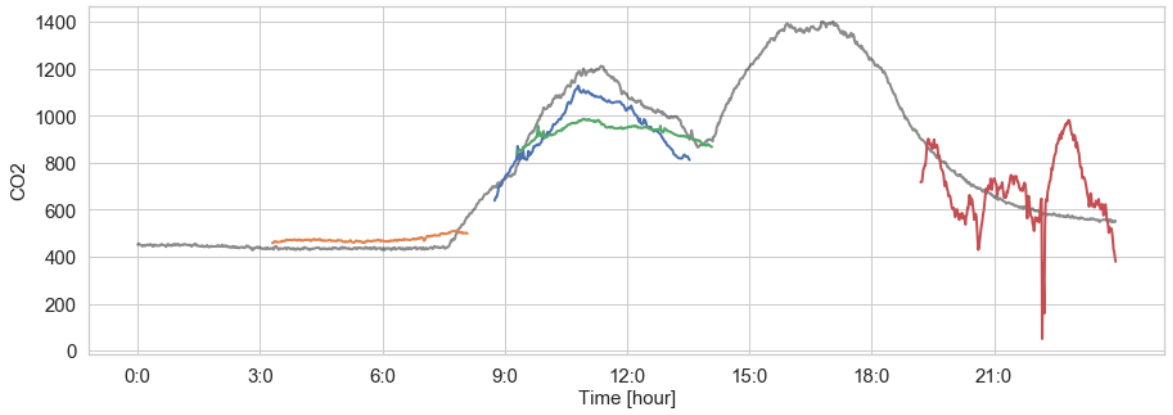


Figure 23: Shapelets on a daily time series

11.3.2 Shapelet-distance classification

The classification task is also performed with the implementation of two classification algorithm that use the distance between shapelets and time series. The first classifier is the Decision Tree and the results return an F1-score of 67% for the minority class, with a precision and recall of 71%. The second classifier is the KNN algorithm, which returns a F1-score of 67%, a precision of 80% and a recall of 57%. Both classifiers have a good level accuracy (86%), but the performances are lower respect to the one obtained without using the shapelets distances as metrics.

11.3.3 Motif

For the motif discovery, after the determination of the matrix profile, different number of motif is tested. The algorithm found 4 motifs that are highlight in the graph with different colours. From the figure below can be seen that the majority of the motifs (3 out of 4) are in the first half of the graph. This is due to the fact that the first part of the time series presents more regular patterns so it is more easy to find representative subseries. In the second half, the time series loses its regularities that's why it is more difficult to find motifs. In fact the only motif found in the second part is the one in blue. The changes in the CO_2 levels between the first half and the second could be due to the fact that on the second one the door of the office was closed so the CO_2 variations followed a more regular pattern.

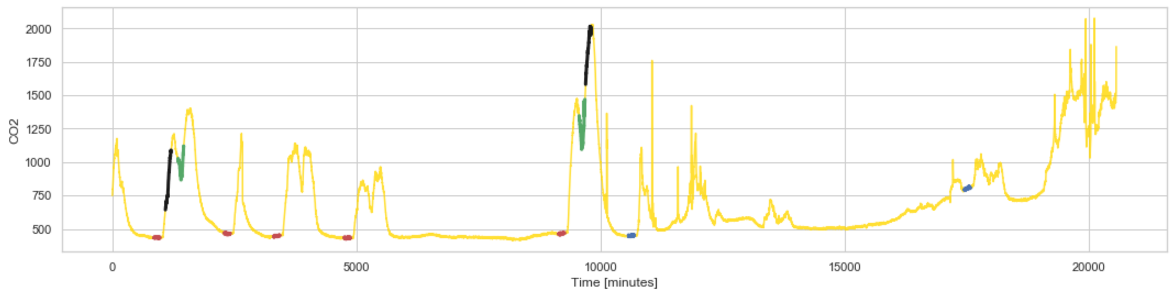


Figure 24: Motif

11.4 Classification

We proceeded with the classification with respect to the 13 features of the TSs of the attribute CO_2 (avg, std, var, med, 10p, 25p, 50p, 75p, 90p, iqr, cov, skw, kur) obtaining a list of 48 TSs (one for each interval of 6 hours) composed by 13 values (matching to the features above), that was splitted into a training set that contained 33 and a test set that contained 15. The classificators used are those shown in Table 10 with the corresponding results.

Real values	0	1	1	0	0	0	0	0	1	0	0	0	0	1	0
DT	1	1	0	0	0	0	0	0	1	0	0	0	0	1	0
KNN	1	1	1	0	1	0	0	0	1	0	0	0	0	1	0
CNN	1	1	1	0	0	0	0	0	1	0	0	0	0	1	0
LSTM	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0

Table 10: Time Series Classification

The model are constructed as follows:

- DT: min_samples_split=6, min_samples_leaf=1, max_depth=1; **[Accuracy = 86.7%, F1-Score = 75.0%]**.
- KNN: metric='dtw_sakoechiba', n_neighbors=3, weights='distance'; **[Accuracy = 86.7%, F1-Score = 80.0%]**.
- CNN: We constructed the Convolutional Neural Network with two convolutional layers: the first one with 16 filters, kernel_size = 8, and relu as activation function; the second one with 64 filters, kernel_size=3

and relu as activation function; the two layers were interspersed with a relu activation function and a Dropout of 0.3, while the output layer had a sigmoid as activation function. The model was compiled with adam optimizer and with metrics = ['accuracy']. **[Accuracy = 93.3%, F1-Score = 88.9%]**.

- LSTM: This classifier was constructed with two layers of size 4 and 16 respectively interspersed with a relu activation function and a Dropout of 0.2. From the second layers to the output layer the activation function was a sigmoid and a Droput of 0.3. The model was compiled with adam optimizer, with metrics = ['accuracy'] and loss = 'sparse_categorical_crossentropy'. **[Accuracy = 93.3%, F1-Score = 85.7%]**.

It is to observe that the last 10 values over the total 15 are always correctly predicted; this is, perhaps, explained by the clear diversity of these TS from the 5 initials. It seems that with the only attributes CO_2 is very difficult, with the few data available, to build a classifier that predicted even the most particular trends. However, there is no record of the test set that is always incorrectly predicted by all the classifiers as a whole. To be able to correctly classify all the TSs the classification is implemented with the multivariate data set containing all the variables in the following section.

11.4.1 Classification with multivariate dataset

The classification with the multivariate dataset is implemented with the CNN algorithm. The network is implemented with an input layer with 8 kernels and 16 filters, the only hidden layer has 5 kernels and 32 filters. To find the best model are tested all the combinations between the following activation functions: relu, linear, sigmoid, softsign, softplus and tanh. The only function that is kept fixed is the one in the output layer which is sigmoid. The solver is always Adam, but the other parameter that is tested with different values is the learning rate that takes on the values 0.01, 0.001, and 0.0001. The total number of models is 75 and 34 of them give an accuracy level of 100%. Some of these models are presented in Table 11.

Models	**Parameters**									
	Input Layer			*Hidden Layer 1*			*Output Layer*		*Solver*	*Learning rate*
	Kernel	*Filter*	*Activation*	*Kernel*	*Filter*	*Activation*	*Nodes*	*Activation*		
Model 1	8	16	Relu	5	32	Relu	2	Sigmoid	Adam	0.01
Model 2	8	16	Tanh	5	32	Tanh	2	Sigmoid	Adam	0.01
Model 3	8	16	Linear	5	32	Linear	2	Sigmoid	Adam	0.01
Model 4	8	16	Tanh	5	32	Softplus	2	Sigmoid	Adam	0.001

Table 11: CNN with multivariate dataset

12 Sequential Pattern Mining

The Sequential Pattern Mining task is implemented by selecting the variable 'CO_2' and by defining 12 time series (which are the number of complete available days) of 24 hours each, where every observation is the mean of the 'CO_2' values in that hour. After the data normalization is made the SAX transformation with a number of segments equals to 24 and a number of symbols equals to 20.

TUE 03	B	B	B	B	B	B	B	B	H	N	S	S	R	O	R	T	T	T	R	M	I	G	E	E
THU 05	B	B	B	B	B	B	B	B	I	P	Q	R	R	O	O	R	R	P	J	D	C	B	B	B
FRI 06	B	B	B	A	A	A	A	B	F	K	M	L	L	H	K	O	O	N	H	C	B	B	B	B
SAT 07	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B
SUN 08	B	B	B	B	B	B	A	A	B	A	A	A	A	A	A	A	A	A	A	B	B	B	B	B
MON 09	B	B	B	B	B	B	B	B	C	M	S	T	T	S	T	T	T	T	T	S	N	I	G	D
THU 12	C	D	E	F	E	E	F	J	L	R	M	L	K	I	F	E	E	J	F	E	E	E	E	E
FRI 13	E	E	E	E	E	E	D	D	D	E	F	I	G	E	E	E	F	F	E	C	C	C	C	C
SAT 14	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	D	D	D	D	D	D	D
SUN 15	D	D	D	E	E	E	E	F	F	G	G	G	G	G	H	H	H	H	H	H	I	I	J	J
MON 16	J	M	N	N	M	L	L	L	M	Q	P	Q	O	N	O	N	O	P	O	K	J	J	J	I
TUE 17	I	I	J	J	J	J	K	K	L	O	P	R	T	S	T	T	T	T	T	T	T	T	T	S

Table 12: CO_2 trend with SAX transformation

In Table 12 the letters correspond, in lexicographical order, to the interval generated by the SAX transformation. The first letter A identifies the lowest level of CO_2 and the last letter T represents the highest one.
The non-working days have low levels of CO_2, and they can be identified in Table 12 by the fact that in these days there are no letters greater than 'J' (the middle letter of the sequence [A,..., T]). More precisely, the first weekend presents only values from the first two lower level A and B, and the second weekend presents a more unusual pattern with higher values of CO_2 but it is still represented by the letters indicating the lower range of levels. Another pattern that can be highlight is how from the trend of CO_2 level can be derived the working hour schedule. Especially in the first four working days, where the office door was open, it can be seen that every day starts with low CO_2 level which is equal to B. This level increases with the beginning of the working hour (around 8:00 am), until the lunch break where there is a slight decrease, for example it goes from level R to O or from L to H. In the afternoon, with the restarting of the working schedule, the level increases and then it later decreases (around 06:00 pm) when the working day is over. As expected, when during the day is reached very high values of CO_2, more time is needed to reestablish lower level.
From Thursday 12^{th} is more difficult to find a regular pattern, but in general the levels are way higher than the previous days. The two most irregular days are Sunday 15^{th} where the office should be closed but the level of CO_2 still increases, and the other one is Tuesday 17^{th} where the second half of the day presents the two highest level of observations identified with the letters T and S.
From the symbols can be also noticed the gap of the time windows. In every time series there is a continuity between the level of CO_2 at the end of a day and at the beginning of the following one. This pattern isn't respected only during the two time gap: the first between Tuesday 3^{th} and Thursday 5^{th} where the level skips from a level E to B, and the second between Monday 9^{th} and Thursday 12^{th} where the level D is followed by C. The following list shows the top 30 closed frequent patterns extracted from the time series with their support:

- support = 8: ['I']
- support = 6: ['B', 'B', 'B', 'B', 'B', 'B', 'B', 'B'], ['C'] , ['D']
- support = 4: ['E', 'E'] , ['F'] , ['I', 'G'] , ['K'] , ['L'] , ['M', 'I'] , ['N'] , ['R']
- support = 5: ['G'] , ['J'] , ['M'] , ['O']
- support = 3: ['D', 'E', 'E', 'E', 'E'], ['D', 'E', 'E', 'E', 'E', 'F'], ['D', 'E', 'E', 'E', 'F', 'F']['D', 'E', 'F', 'G'] , ['D', 'E', 'F', 'I'] , ['D', 'E', 'G'] , ['I', 'G', 'E', 'E'] , ['I', 'J'] , ['M', 'I', 'G'] , ['M', 'K'] , ['M', 'L', 'K'] , ['O', 'O'] , ['O', 'P'] , ['O', 'R']

The frequent patterns with the single letter give us only information about the frequency of that precise level. For example the first frequent pattern shows that only the level I is reached by 8 day of a total of 12. The longest frequent pattern with a support equals to 6 shows that half of the 12 time series maintain for 8 hours per day a value of CO_2 equals to the level B, which is one of the lowest level. This pattern can be found only in the first half of the time series, from Tuesday 3^{th} to Monday 9^{th}, where the office door should have been closed.

13 Outlier Detection

Before the implementation of specific methods for the outliers detection, the distributions of the different variables with their respective anomaly values, is analyzed with the following boxplots. From these graphs two interesting factors are that on one side, the distributions of the variables *'Humidity'* and *'HumidityRatio'* don't show any outliers, on the other side, the variable with the highest number of outliers is CO_2. To better explore this aspect, in the following section, these anomalies are highlight on the time series with the CO_2 variable.

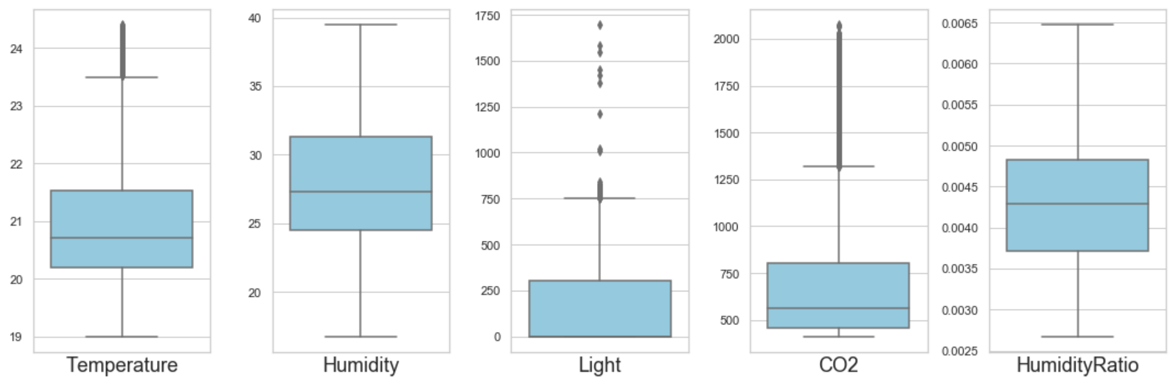


Figure 25: Boxplot of the attributes

The boxplots are very straightforward informative tool to check the outliers distributions but, to overcome their limit of representing variables one by one, specific outliers detection methods are tested.

13.1 Outlier detection methods

After the data normalization, for the outlier detection task three different algorithms are tested: the first algorithm is DBSCAN which is a density-based method, the second is LOF which is a distance-based method and the third one is ABOD which is an angle-based method. Each method is implemented to find the top 1% outliers.
The parameters used for DBSCAN are: number of neighbours equal to 4 and the radius equal to 13.11. The results of this method are discussed in the next paragraph.
The number of neighbours for the LOF method is set to 25 while for the ABOD method it is set to 11. Figure 26 shows the distribution of the outliers in both methods.

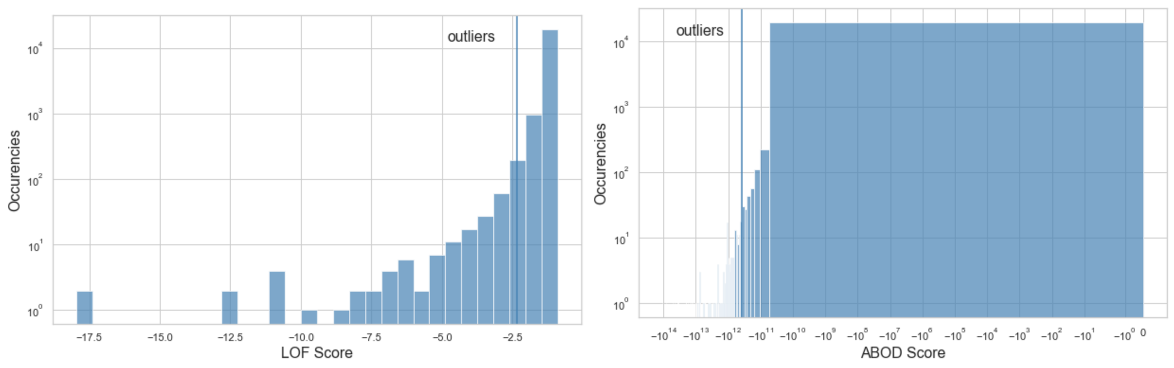


Figure 26: LOF and ABOD outlier's scores

In the previous graphs the LOF graph has a semilogarithmic scale while the ABOD graph has a logarithmic scale on both the axis.

13.2 Comparison between outlier detection methods

To check if the three methods selected more or less the same values as outliers, two different graphs are chosen for each method to compare the performances. For one type of graph the PCA is used to represent the data set in two dimensions highlighting the outliers in red. The other graphs show the positions of the outliers in the complete time series with the CO_2 variable, highlighting them in red.
From the graphs with the time series it seems that the LOF and the ABOD algorithm select almost the same outliers, so their results seem to be more similar between each other respect to the DBSCAN method. Each method detects 207 data points, 57 of them are detected by all three methods, which means that the methods share the 27% of the results. These common outliers are highlight in blue in the graphs. The time series graph (Figure 28) shows that the majority of the common outliers are found on the second part of the time series, from Tuesday 10^{th}. Several outliers are especially found during the CO_2 peaks on Thursday 12^{th} and during the night between Tuesday 17^{th} and Wednesday 18^{th} which follows a very unusual pattern. The only three days that don't have common outliers are all non working days.

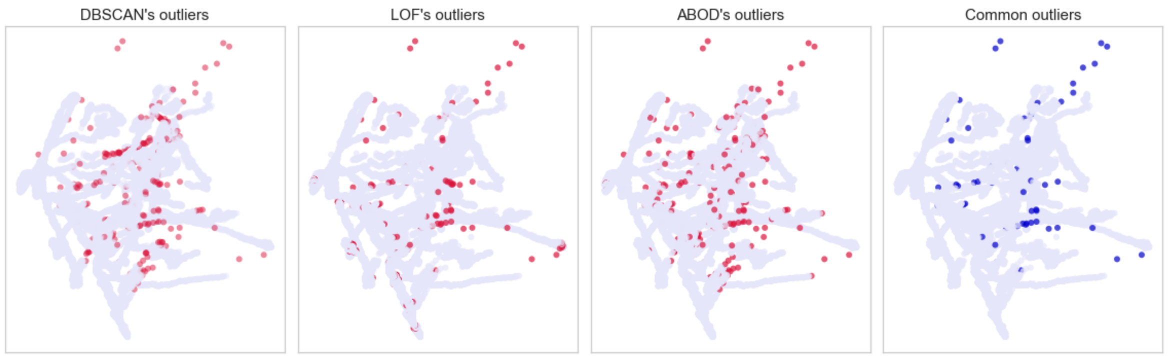


Figure 27: Outliers representation in PCA domain

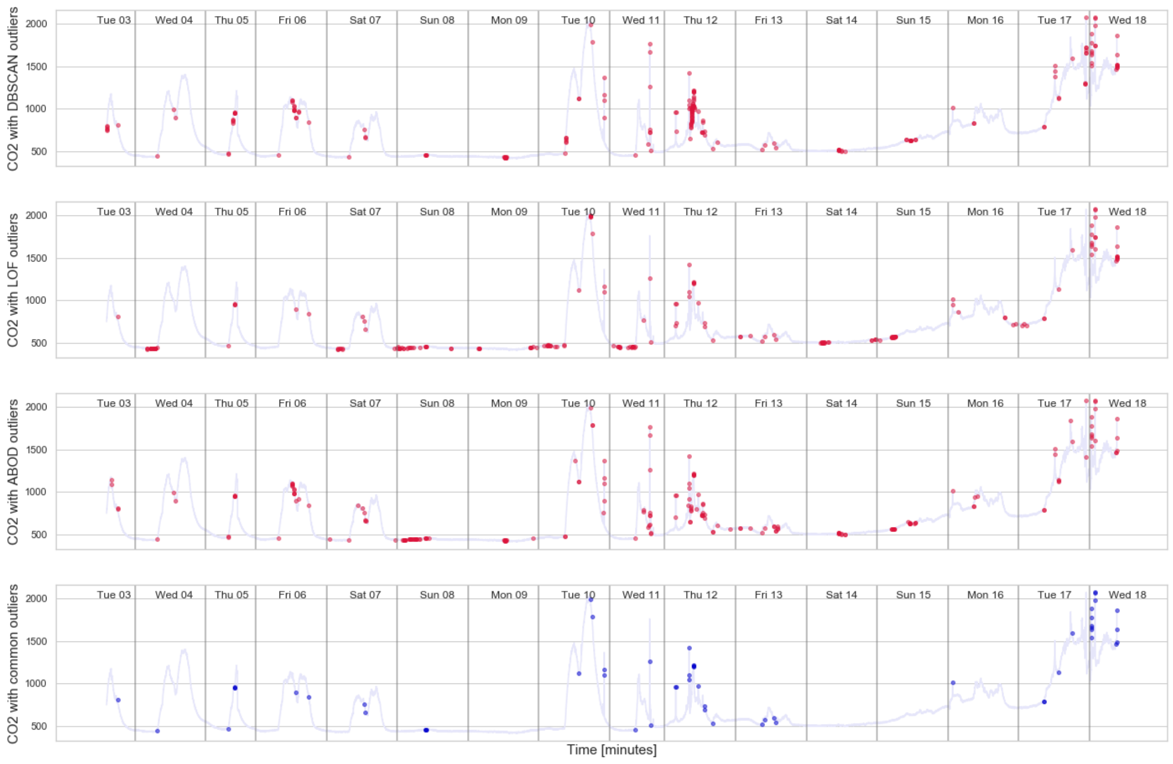


Figure 28: Outliers representation in time domain

14 Conclusions

Like already explained during the introduction, the variable *'Light'* is resulted to be crucial for every algorithm tested due to it strong correlation with the working schedule which helped the classificators to classify the room as occupied or not. Anyway, the limited amount of data has not made possible such an efficient classification of the target class using the remaining attributes with respect to using also *'Light'*. To implement more general models which are independent from the working hours, the more reliable variable is considered to be 'CO_2' because it is directly influenced by an individual in a room.
The more complex classificators has performed on average slightly better with respect to the more elementary ones. However, since the design of these models is more difficult, we expected higher results in the classification to justify this complexity. This performance issue is probably due to two reasons: thanks to the easy data set the elementary classifiers rapidly reach high performances with a simple parameter testing. Secondly, these complex classifiers usually works better when they are fed with a large amount of data. Unfortunately, this was not the case with a data set of 20,000 records and 5 features.