Data Mining: Occupancy Detection

Carlo Paladino Giuseppe Mammoliti Piyush Tada

537650 535053 605988

1 Introduction

Nowadays, there are a lot of attentions in energy saving. For this purpose, the ability to predict if a room is occupied or not can bring essentials money savings. This prediction can prevent the risk of wastes of energy and money. The advancement of the data mining techniques joined to the depreciation of the sensor allow to predict without the use of digital cameras, so with a gain in terms of privacy of the occupants.

1.1 Data understanding

We have three different datasets, the first composed by 8143 instances, is our training dataset, second and the third, called datatest1 and datatest2 of 9752 and 2665 entries respectively, are used as test datasets we can see more details in table 1.

DATA SET	NUMBER OF RECORDS	ATTRIBUTES	OCCUPANCY DISTRIBUTION		NOTES
DATA TRAINING	8143	7	0	0.79	CLOSE DOOR
			1	0.21	
DATA TEST 1	2665	7	0	0.64	CLOSE DOOR
			1	0.36	
DATA TEST 2	9752	7	0	0.79	OPEN DOOR
			1	0.21	

Table 1: Description of the datasets.

There is one variable represented by categorical attributes and seven numerical variables among which, two are integers and the remaining five are floats. There is only a binary variable, Occupancy, which is our target variable. It has value zero if the room is defined as "not occupied" or one if is "occupied".

To lead our work, we analyze and plot the different correlations between attributes.

The figure 1 shows the correlation matrix, where we can observe how strong is the relation between different variables. The strongest correlated attributes to the Occupancy are Light and CO2 (0,91 and 0,71). As we can see from the table 13, all the correlations from the figure 1 have a valid statistical significance, because the p-value associated to every pair of attributes is less than 5%. We can also observe in the chart 38 the values of all the features over one day of 5 February 2015.

Figure 1: Correlation matrix

1.2 Creation of new variable

We created a new variable week_day with value of 0 as weekend and 1 as weekday. We can see the correlation values of this new variable with other variables in figure 1.

2 Basic Classifiers

2.1 Metrics evaluation

Before starting to analyze all the different models, we want to introduce the evaluation performances we'll use. They are accuracy, precision, recall, f-1 score and we will plot roc curve and lift chart of most interesting results.

Accuracy is defined as the number of correct predictions (true positive plus true negative) on the total number of prediction and it defines the number of predictions that are well classified. Precision is the number of true positive on the total of true positive plus false positive, it defines the percentage of results that are relevant. Recall is the number of true positive on the total of true positive and false negative and it refers to the percentage of total rel-

evant results correctly classified by the algorithm. F-1 score is the harmonic mean of precision and recall. We want to find the highest accuracy.

Then we have roc curve. It is created by plotting the true positive rate against the false positive rate at various thresholds and we will use it for the comparison on the models. The best curve will be the one higher than the others. The perfect model will be similar to an angle of 90 degrees, at the top-left of the matrix. Auc is defined as the area under the curve.

The other tool we'll use is lift chart. It represents the expected number of positives we would predict if we did not have a model but simply selected cases at random. It provides a benchmark against which we can see performance of the model. A lift chart graphically represents the improvement that a mining model provides when compared against a random guess, and measures the change in terms of a lift score.

2.2 K-nn

Nearest-neighbour classifier is an instance-based learning technique for classification and regression. It's a lazy learner, do not require model building and makes predictions based on local information.

We observe how the results changed with different values of the following parameters:

1. K (number of nearest neighbours), we tested from k = 2 to k = 200 with steps of 1

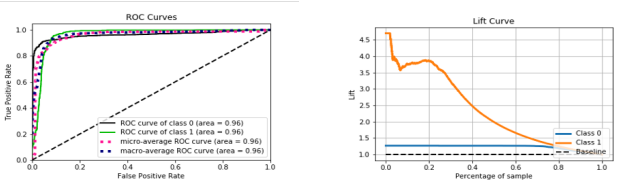
   This is the most important parameter of k_nn, it regulates the tradeoff between overfitting and underfitting (k too low, for example k=1, overfits the training set, while k too high, for example k=n.training records, underfits the data).

2. Metric, the metric for the distance function, we used "minkowski" distance that with p = 2.

3. P, is the power parameter for the minkowski distance (as said before, p = 2 is equivalent to the standard euclidean metric.

4. Weight uniform, all the points in the neighborhood are weighted equally.

We apply on kneighborsclassifier the random search (randomizedsearchcv) to find the best number of neighbours. It finds as best result k = 3, performance of that model is shown in table 14.



(a) Roc Curve (b) Lift Chart

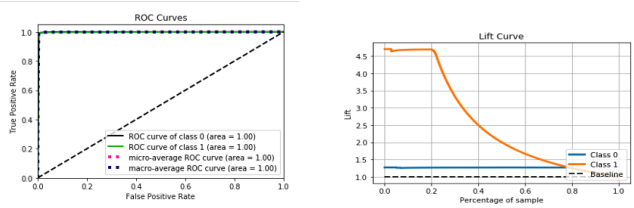
Figure 2: Chart for K-NN,training set

2.3 Naïve bayes

Naïve bayes (NB) is a probabilistic framework for solving a classification problem. Based on bayes theorem that try to predict an event A, assuming that an event B has occurred. It's assumed that the presence of a particular feature in a class is unrelated to the presence of any other feature.

We observe how the results change with different models: Gaussian and Categorical.

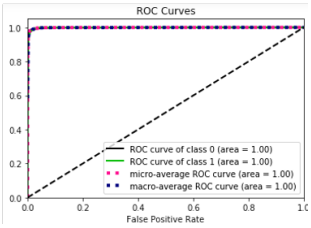
In gaussian NB the likelihood of the features is assumed to be gaussian. Is a continuous probability distribution used to describe random variables with real values which tend to concentrate around a single average value. You can obsereve lift and ROC of method in figure 3



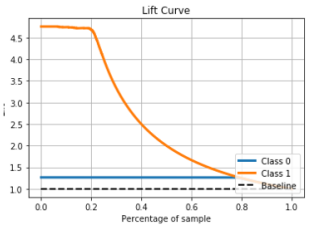
(a) Roc Curve Gaussian,training set (b) Lift Chart for GaussianNB, training set

Figure 3: Chart for GaussianNB,training set

Categorical NB is a classifier for categorical features. It's suitable for classification with discrete features that are categorically distributed. The categories of each feature are drawn from a categorical distribution. The table 14 shows the most important configurations of parameters with gaussiannb and categoricalnb package from sklearn.



(a) Roc Curve Categorical,training set



(b) Lift Chart for Categorical, training set

Figure 4: Chart for Categorical,training set

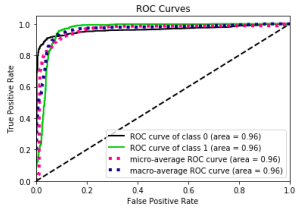
Both models were able to predict the target variable with high accuracy above 94% have a look at the full results in table 14.

2.4 Logistic regression

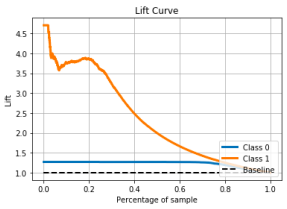
Logistic regression is a binary classification algorithm, it tries to find the best hyperplane in k-dimensional space that separates the 2 classes, minimizing logistic loss.

With logistic regression we divide the data set thought a straight line in case of multidimensions. Then we classify the data in two categories.

We have trained the model using different numbers of attributes.



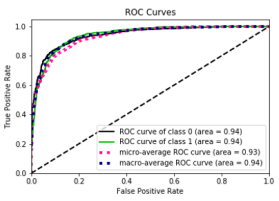
(a) Roc Curve logistic regression ,training set



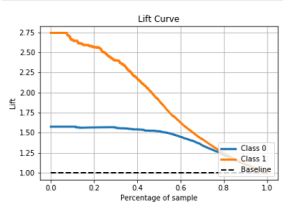
(b) Lift Chart for logistic regression, training set

Figure 5: Chart for logistic regression,training set all elements

The table 14 recap the results we reached. We get best performance using only two attributes as compare to most of them. In particular we notice that the best results in terms of accuracy for all the datasets are given by the logistic regression between co2 and light, as we can observe in figure 6.



(a) Roc Curve logistic regression ,training set



(b) Lift Chart for logistic regression, training set

Figure 6: Chart for logistic regression,training set co2 and light

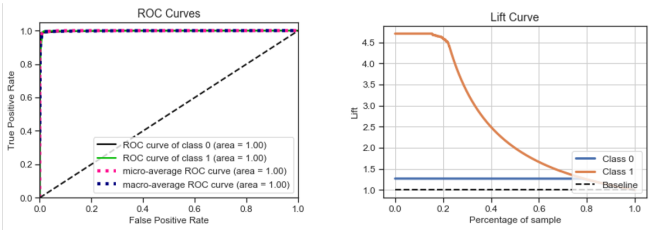
2.5 Decision tree

Decision three is one of the most popular and powerful classification model, it's a flowchart in the shape of a tree, where each internal node represents a test on the specifically attribute, each branch it's an outcome of the test, and each leaf node belongs to a class.

We decided to do several tests using as splitting measures both Gini Index and Entropy Information, the fundamental difference is that Gini return the probability of a random sample being classified incorrectly, the Entropy is a measures of impurity and controls how the Decision Tree decides to split the data. Additionally we used as splitter methodology best (that select the most relevant features for split) and random (that select random the features to split). To avoid the risk of overfitting and to validate our training set we used cross-validation.

At the beginning we applied to the training set the Decision Tree model with "standard" value(1) for the max depth (that is the max number of level of the tree), for the minimum sample split(specifies the minimum of samples required to split and internal node) and for the minimum sample leaf(specifies the minimum number of samples required to be at a leaf node), to observe the fit of decision tree with the training set and its performance with two test sets.

After the positive feedback from the above experiment we wanted to find optimum parameters for Gini and Entropy, so we used a random search from sklearn.model_selectio,n RandomizedSearchCV (i.e technique where random combinations of the hyperparameters are tested to find best possible values).



(a) Roc Curve (b) Lift Chart

Figure 7: Chart for decision tree,training set

The random search return us a "list" of possible optimal combinations of max depth, minimum sample leaf and minimum sample split, we tried a lot of these different combinations both for Gini (best and random splitter) and Entropy (best and random splitter), and we have reported the best in the table 14.

The best result is achieved with Gini Index, random splitter, max depth of 15, min sample split of 10 and minimum sample leaf of 30. For more detailed results you can look at table 14.

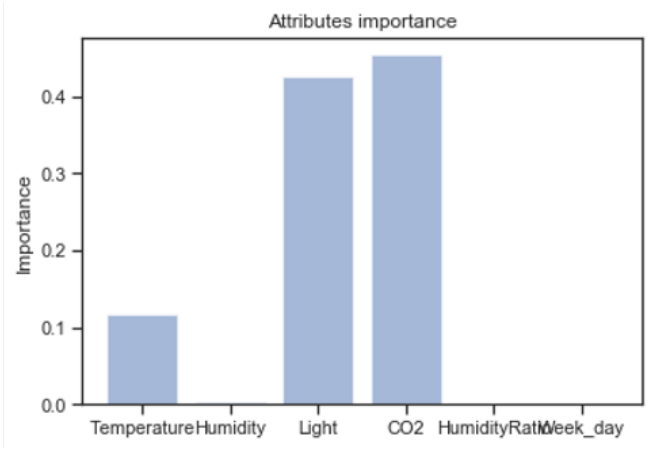


Figure 8: Importance of the attributes for the decision tree

3 Dimensionality reduction

The Principal Component Analysis (PCA) is a feature projection dimensionality reduction technique, the feature projection techniques like PCA or SVD (Singular Value Decomposition) transform the data space in a smaller dimensional space. "The goal of PCA is to find a new set of dimensions (attributes) that better captures the variability of the data. More specifically, the first dimension is chosen to capture as much of the variability as possible. The second dimension is orthogonal to the first, and, subject to that constraint, captures as much of the remaining variability as possible, and so on" (Pang-Ning Tan, Introduction to Data Mining).

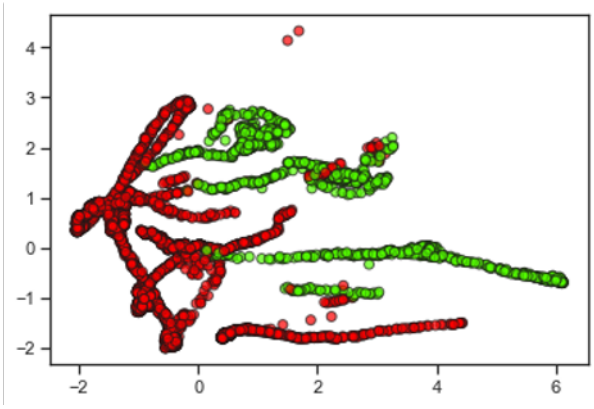


Figure 9: Two dimension plot after PCA

To understand if it's possible to reduce the dimensionality of the datasets and at the same time keep the same results as the original datasets for the model, we have applied different techniques for dimensionality reduction like variance threshold, univariate feature selection, recursive feature elimination.

After the dimensionality reduction for all the different techniques we used the same two classifications model to compare the results. The models that we have used for it are Decision Tree Entropy,random. With max depth of 3, minimum sample split of 20,minimum sample leaf of 1 and the Decision Tree Gini, random, with max depth of 15, minimum sample split 10 of and minimum sample leaf of 30.

PCA technique returned us best results, if we consider both training and two different tests.

If we consider only training we have best results with the recursive feature elimination,with an accuracy of 0.9843, but this result is misleading because for test one the results are not good, while having an accuracy of 0.7110 the recall and f1 are respectively of 0.21 and 0.35, which are sub-par.

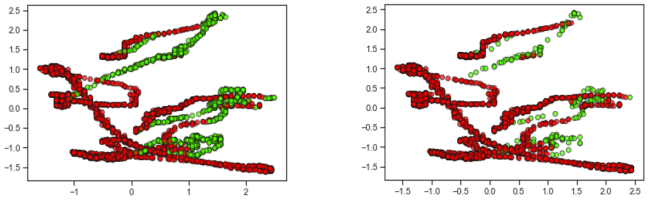
We can see the result after the application of the decision tree in table 15. The results show us that after the PCA the model doesn't suffer a significant loss in the scores of accuracy, precision, recall and f1. In conclusion we can see that in this use case PCA was able to perform at the optimal levels.

4 Unbalanced data sets

Often we have to do deal with unbalanced data sets, where one of two classes is dominant, which can lead

to wrong predictions. We need to solve this problem before we apply any algorithm because it can lead to malfunction of the algorithm when trying to predict the target class.

In our case, we can see from the table 1 we have fairly balanced datasets. To see if the unbalancing of the training datasets can lead to a wrong prediction of the target class, we unbalanced the training set with a distribution of 96% for class 0 and a distribution of 4% for the class 1.



(a) In the training set before the unbalancing

(b) After the unbalancing

Figure 10: Distribution of the Occupancy class

In figure 10a we can see a visual distribution of the two class before the displacement instead in figure 10b after the displacement. We wanted to do several test with different techniques to manage the problem which derives from the imbalance of the datasets to understand which one best suited the case.

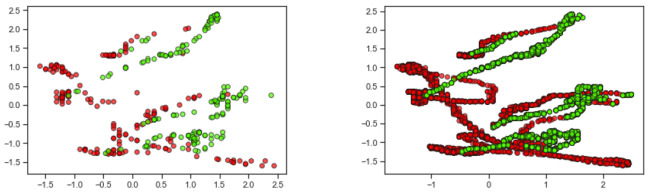
At first, we tried with the technique of Class Weight where we go to assign a greater weight to the less numerous class, with a weight of 1 for the class 0 and of 10 for the class 1, this did not lead to a significant improvement in results as we can see from the table 16, especially in the data set 1 which is the most afflicted from the problem of the imbalance of the training set.

We later tried some techniques of under-sampling and oversampling. The undersampling consists in reducing the number of the most numerous class, we tried random undersampling and condensed nearest neighbor, the random undersampling consist of randomly deleting samples from the majority class, the main problem with this techniques is that it could delete some samples of the majority class that are important.

	Distribution		Total
Imbalanced dataset (96% 4%)	0	6414	6681
	1	267	
Random Undersampling	0	187	374
	1	187	
Random Oversampling	0	4489	8978
	1	4489	
Smote	0	4489	8978
	1	4489	

Table 2: Data set distribution.

For the oversampling, that consists of increasing the number of the less numerous class, we used the Random Oversampling and the Smote techniques. The Random Oversampling, choose randomly some sample from the minority class and duplicate it into the datasets, instead the Smote use the interpolation to oversampling the minority class.



(a) Random Undersampler

(b) Random OverSampling

Figure 11: Effect of sampling

In table 16 we have a summary of how the use of these techniques has affected the use of the classification model, in this case the decision tree. From the scores of the table 16 obtained by the decision tree we can see how the Class weight techniques and the two oversampling techniques that we have used (random over sampling and smote) return the best result.

5 Conclusions

In this research, our goal was to classify/predict if an office room, in a building, was occupied or not. Mining our data, we notice that there are some features, more relevant than others, that help us to predict/classify the occupation of the room (i.e. Light and CO2).

We have trained different classification models on the data training set and tested on the two test sets (test set 1, with closed doors, and test set 2, with open doors). All the models were judged using accuracy, precision, recall, f1 and then roc-auc curve

and lift chart were plotted. We can notice from the table 14 and looking at the single roc-auc curve and lift chart for each single model, that all the different classifier used return excellent results. Only the logistic regression, that used only light and co2 to predict the target class, show us worst results compared to the other models, looking at the roc curve and lift chart.

To understand if was possible to increase the computational speed of the different algorithms used, we tried some different dimensionality reduction techniques. As we said in paragraph 1.3, we have obtained the best results from the Principal Component Analysis, so we can conclude, that having available this feature, it's possible to reduce the input size without loss in terms of scores.

At the end of this first part of the research, we have unbalanced the dataset, with a distribution for the Occupancy : 96% of 0 and 4% of 1, to understand how this can affect the ability of the different classifier to handle with a small number of records belonging to one class. The two different configurations of the decision tree that we have used to classify the target class, having the datasets unbalanced, show us a loss in terms of scores, particularly in test set 1. So, we test different techniques to manage this type of datasets and we conclude that in this case the two oversampling techniques used are the best to handle the unbalanced dataset, as we can see in table 16.

6 Linear regression

In statistics, linear regression is a linear approach to modeling the relationship between a scalar response and one or more explanatory variables. The case of one explanatory variable is called simple linear regression. For more than one explanatory variable, the process is called multiple linear regression.

Linear regression is the simplest and most widely used statistical technique for predictive modeling. It basically gives us an equation, where we have our features as independent variables, on which our target variable is dependent upon.

6.1 How to measure results of linear regression

Metrics used for measuring the results of linear regression.

R2 (r-squared) The close to 1 it is the better then it says that the value us used for prediction is moving with the final value that we wanted.

MSE (mean squared error) It calulate the distace of value from the mean. An MSE of zero, meaning that the estimator Z predicts observations of the parameter X with perfect accuracy, is the ideal, but is typically not possible.

MAE (mean absolute error) is basically the absolute error —x-x— so lower the value the better

In this scenario, we are trying to predict one variable on the basis of other Temperature Humidity we used temperature to find the value of the humidity for example, in table 12 we can see the different experiments. We only got good results in the case of the humidity and humaidity ratio as to be expected become one is derived from the other.

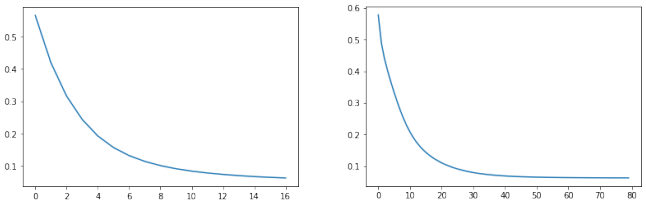
7 Advanced Classifiers

7.1 Neural networks

Neural networks are a set of algorithms that are vaguely modeled on human brain. In this kind of models we have input layer, some hidden layers and output layer. If we have one hidden layer we are talking about a perceptron, if we have two or more hidden layer we are talking about a neural networks or multi-layer perceptron.

7.1.1 Perceptron

Preceptron is a linear classifier, it is formed by four parts that are: input values, weights and bias, net sum and an activation function.



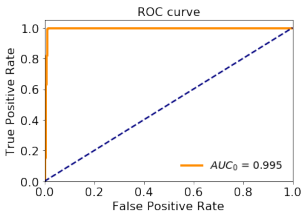
(a) Loss curve for Earlystopping models

(b) Lift Chart for Overfitting, training set

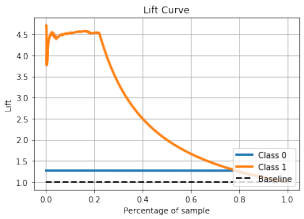
Figure 12: Loss curve for models

For activation function we have used ReLu and Tanh. During the training phase the weight parameters are adjusted until the outputs of the perceptron become consistent with the true outputs of the training examples in according to a learning rate, that is a parameter with value between 0 and 1 used to control the amount of adjustment made in each

iteration. If is close to 0 the new weight is mostly influenced by the value of the old weight, if it is close to 1, then the new weight is mostly influenced by the current adjustment.



(a) Roc Curve Preceptron ,training set



(b) Lift Chart for Preceptron, training set

Figure 13: Chart for Preceptron,training set

To achieve good results, we have tried a lot of different combinations for the parameters. Then we've used the grid search to search for the best parameters. In table 5 we can see the scores of the three best combinations of parameters that we have found and is the best combinations that we have found with grid search.

At first we haven't used any regularization techniques, when we found acceptable results we applied the early stopping technique, that is a regularization technique that mitigate the risk of overfitting, the results in table 5 are the results of the model using the early stopping. How we can see in fig 12 (that are referred to loss curve of model with learning rate constant, activation function relu and solver adam), the loss curve without the using of the early stopping (fig. 12) it's overfitted (it does not decrease as the number of epochs increases), instead the loss curve with the use of the early stopping it's perfect fitted, it stops before the number of cryptic epochs that allows it to decrease throughout its journey. In the figure 13 we can see, respectively, the roc curve and the lift curve of this model.

[5]*this value are referred to, in order: hidden layer size, learning rate, activation function, solve and momentum.

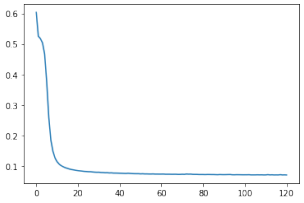
7.1.2 Multi-layer neural networks

As we already said the multi-layer neural networks are neural networks with multiple hidden layers between the input nodes and the output nodes. It can solve any type of classification task involving nonlinear decisions. We have several possible activations functions, such as: sigmoid, linear, hyperbolic tangent, sign function, soft exponential. The goal of this model is to minimize an error function, having this form:

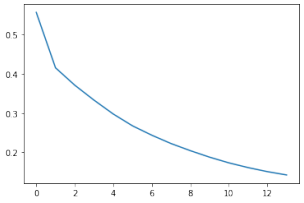
$$E = \frac{1}{2}\sum_{i=1}^{n}\left(y_i - f\left(\sum_j w_j x_{ij}\right)\right)$$

The weights are updated in the opposite direction of the gradient of the loss function:

$$w_j^{(k+1)} = w_j^{(k)} - \lambda\frac{\vartheta E}{\vartheta w_j}$$



(a) Loss curve for before Earlystopping

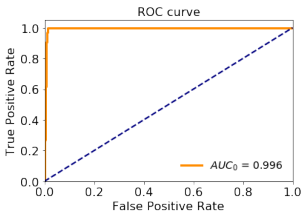


(b) Lift Chart for after earlystopping

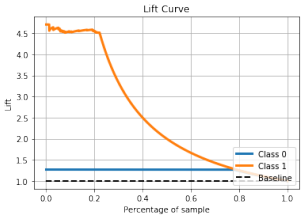
Figure 14: Loss curve for stopping

The error is computed at the output and propagated back to the input by chain rule to compute the contribution of each weight to the loss (error backpropagation).

As for the perceptron we tried in a first moment to reach the best combinations of parameters, then we have applied the early stop technique to avoid the problem of overfitting, we can see in the table 5, the better combination of hyperparameters that we have found it's with four hidden layer with dimension of: 256,128,64,32,costant,identity,adam,0.9. In figures 14 we can see how the loss curve change before and after use of the early stop technique. In figure 15 we have the roc and lift curve of the same model.



(a) Roc Curve Multi-layer ,training set



(b) Lift Chart for Multi-layer, training set

Figure 15: Chart for Multi-layer,training set

7.1.3 Deep neural networks

A deep neural network is a neural network with multiple layers between the input and output layer.

Techniques	Loss	Accuracy
Early stopping	0.067245	0.983217
L2 regularization	0.243971	0.990995
Dropout	0.117646	0.982399

Table 4: Scores for different regularization technique's

Autoencoder is a particular neural network with configuration that has been designed to allow the computer to self-learn, it takes a high-level input, encode it to make it to a lower lever and decode it to return at an understandable input to a human. This is possible for the machine by passing the input into a coding function and decoding it (rebuilding it) through another function, with minimum loss.

Encoder function: $h = f(x)$

Decoder function: $r = g(h)$

One of the biggest problems in using the models is that the backpropagation across multiple levels of hidden layers can cause the so-called gradient vanishing. One of the main causes of the gradient vanishing is the presence of classic nonlinear activation functions, such as the hyperbolic tangent or the logistic function, in order to try to avoid this we used like activation function the relu function.

We built first model of dnn with three level of hidden layers, with sizes of 128-64-1 using as activation function relu and optimizer adam. Then we train and test this model with two different configuration of epochs (defines the of number times that the learning algorithm will work through the entire training dataset) and batch size (defines the number of samples to work through before updating the internal model parameters) how we can see from the table 5, after a third model used to validate the data we made a comparison between three different regularization techniques: early stopping, l2 and dropout. We can see the scores of the three different techniques in table 4, since early stopping got the best scores, we decided to use it as technique to prevent overfitting. We can see the results on the first configuration after the applications of the early stopping in table 5.

Therefore we decided to apply randomized search to optimize the hyperparameters and built a second model with three level of hidden layers with size of 32-32-1 and retested to compare with previous model, the scores that we obtained are in table 5.

7.2 Ensemble methods

In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone.

For the project we are using random forest. First we plotted the feature importance figure 16. According to the chart, light and CO2 are the most important feature for the purpose of random forest.

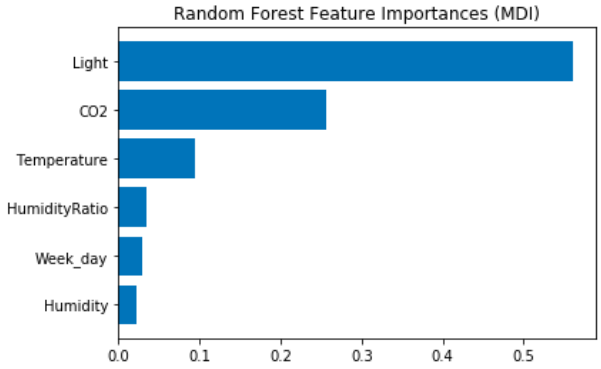
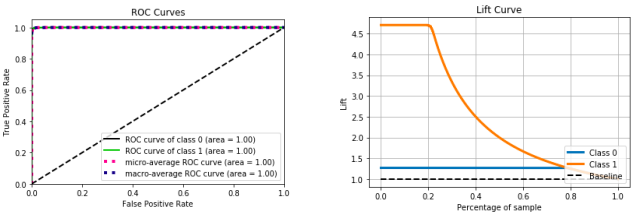


Figure 16: Random Forest feature importance

7.3 Random forest

A random forest is a meta estimator that fits a number of decision tree classifiers on various subsamples of the dataset and uses averaging to improve the predictive accuracy and control overfitting. As we can see from the table 5, we tried several different settings of this method to find if we can improve the results.



(a) Roc Curve (b) Lift Chart

Figure 17: Chart for Random Forest,training set

We can see after some iterations results in table 5,most of the results that we got are good. But only in case where we have max_depth=none, max feat =4 we see a dip in performance (0.8841) in test set 1 but if we look at the performance of test set 2 (0.9388) it is lower than training (0.9939) even in this case we cannot say that it is overfitting.

So, the conclusion from all the above experiments is simple, most of the time the predictor is able to perform well in both cases when we using training set or in case when we are doing prediction on test sets.

7.4 Cross validation

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. This is used to avoid over-fitting because at every iteration we are using different set of values in the dataset.

7.5 Tuning hyper-parameters

The next step in the process is to find out what is best parameters to use. For doing this we used two methods

1. Gridsearchcv

   Exhaustive search over specified parameter values for an estimator. It search for all the possible options of the parameters.

2. Randomizedsearchcv

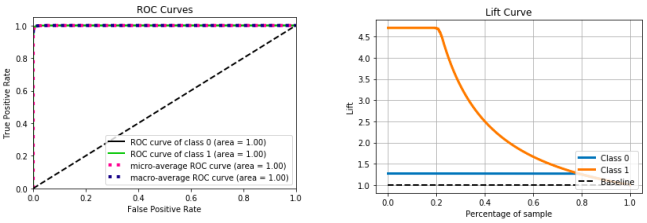
   Randomized search on hyper parameters. Here at the place of trying all the parameters it method selects them randomly.

Random search provided us with the best values as min_samples_split': 30, 'min_samples_leaf': 30, 'max_depth': 14. From table 5 one thing become clearer: we don't get any performance enhancement after hyper parameters tuning, but something interesting happened: if we have a look at recall and f1 score of test set 1, we can see that performance is reduced, but if we compare it to result of our previous test, it is better. So, we can conclude that in previous experiments, model is fitting closer to the training set but after parameter tuning. The model is more stable now.

7.6 Bagging

Bagging, often used what is considers homogeneous weak learners, learns them independently from each other in parallel and combines them following some kind of deterministic averaging process.

In the project we use three different base estimators to find which works best. In this case we are observing in table 5 that recall and f1 score of class 1 is low as compare to test set 2 and training set. Which mean that the ability of model to classify 1 in total 1 is lower than in case of 0 in test set 1. Let's look at roc curve of test set 1 in first case figure 18.



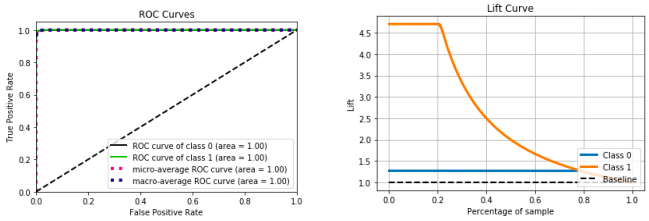
(a) Roc Curve (b) Lift Chart

Figure 18: Chart for Bagging Classifier,training set

7.7 Boosting

Boosting, that often considers homogeneous weak learners, learns them sequentially in a very adaptive way (a base model depends on the previous ones) and combines them following a deterministic strategy, but giving different weight to them.

Here we did two experiments where one with no base estimator and random forest. In both cases we observe similar results table 5 and figure 19.



(a) Roc Curve (b) Lift Chart

Figure 19: Chart for AdaBoost Classifier,training set

7.8 SVM

7.8.1 Linear svm

For implementation of linear svm we used linearsvc from sklearn.svm. We used default parameters (penalty='l2', loss='squared_hinge', *, dual=True, tol=0.0001, multi_class='ovr', fit_intercept=True, intercept_scaling=1, class_weight=None, verbose=0, random_state=None, max_iter=1000) of the library for model with exception of value of c, where we tried different values [0.1, 1, 10]. To do that, we used gridsearchcv from

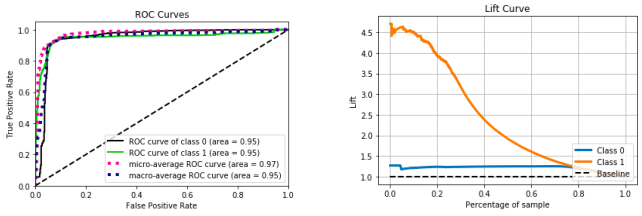
sklearn.model_selection. When we fit the following model, we got the value of c=10 as the best candidate, in the table 5 we can see how the model performs on the two testsets that we have.

As we can see, the model is performing above 94% accuracy in all the cases. It is not possible to make lift and roc curve for this model because it does not provide us with predict_proba which is required to plot the charts.

7.8.2 Nonlinear svm

For implementation of nonlinear svm we used the following settings as base parameters (gamma='auto', random_state= 45, probability=true,c=0.1). Then, we looped over different values of kernel ['linear', 'rbf', 'poly'] to determine the best model.

As we can it is performing very well in table 5, the interesting thing to note here is that class 0 has four times the values of 1 even then model is good at predicting those values. From the results we can see that all the models performed well with only exception in testset 1 with kernel = poly, where the there is dip in performance. You can see the charts 20 for the same.



(a) Roc Curve (b) Lift Chart

Figure 20: Chart for Non Linear SVM,training set

7.9 Conclusions

In this part, we have trained "advanced" classifier models as neural (single and multilayer) and deep neural networks, ensemble methods (random forest, bagging, boosting), and linear and nonlinear support vector machines. In this situation too, the models presented, have high scores in terms of accuracy, precision, recall, f1 score, as we can notice looking at table 5. But, if we analyze more in detail, we have observed that deep neural networks don't perform as well as the other models. Looking at their scores, particularly, the scores of the testset1, they obtain the lowest results. The fact that all the other models have really good scores, is supported by the graphical evidence of their Roc curves and Lift charts.

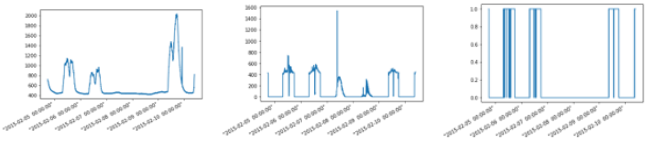
Models	Parameters	Class	Training set				Test set 1				Test set 2			
			Acc	Prec	Rec	F1	Acc	Prec	Rec	F1	Acc	Prec	Rec	F1
Nn, Single layer	(32),constant,relu,adam,0.9*	0	0.9902	1.00	0.99	0.99	0.9448	0.94	0.97	0.96	0.9738	1.00	0.97	0.98
		1		0.96	0.99	0.98		0.95	0.90	0.92		0.90	0.99	0.94
Nn, Single layer (grid search)	(256),constant, Tanh,adam,0.9	0	0.9906	1.00	0.99	0.99	0.9594	0.96	0.97	0.97	0.9681	1.00	0.96	0.98
		1		0.96	1.00	0.98		0.95	0.94	0.94		0.87	1.00	0.93
Nn, Multi layer	(256,128,64,32), Adaptive,logistic,adam,0.9	0	0.9906	1.00	0.99	0.99	0.9482	0.95	0.97	0.96	0.9762	1.00	0.97	0.98
		1		0.96	1.00	0.98		0.95	0.91	0.93		0.90	1.00	0.95
Nn, Multi layer	(256,128,64,32), Costant,identity, Adam,0.9	0	0.9910	1.00	0.99	0.99	0.9752	0.99	0.97	0.98	0.9520	1.00	0.94	0.97
		1		0.96	1.00	0.98		0.95	0.99	0.97		0.82	1.00	0.97
Dnn, Conf. 1	Epochs=1500, Batch$_size$ = 1000000, $Loss$ = 0.0206	0	0.9329	0.92	1.00	0.96	0.8675	0.83	0.99	0.90	0.9127	0.90	0.99	0.95
		1		1.00	0.68	0.81		0.97	0.65	0.78		0.96	0.61	0.75
Dnn, Conf. 2	Epochs=2000 Batch$_size$ = 100000, $Loss$ = 0.0211	0	0.9406	0.93	1.00	0.96	0.8615	0.83	0.98	0.90	0.9385	0.93	1.00	0.96
		1		1.00	0.72	0.84		0.94	0.66	0.78		0.98	0.72	0.83
Dnn, Conf. 1 Validat.	Epochs=1500, Batch$_size$ = 1000000, $Loss$ = 0.0296	0	0.9218	0.91	1.00	0.95	0.8431	0.81	0.98	0.89	0.9385	0.93	1.00	0.96
		1		1.00	0.63	0.78		0.94	0.61	0.74		0.98	0.72	0.83
Dnn, Conf. 1 Early stopping	Epochs=1500, Batch$_size$ = 1000000$Loss$ = 0.0391	0	0.9361	0.93	1.00	0.96	0.8503	0.82	0.98	0.89	0.9385	0.93	1.00	0.96
		1		0.99	0.70	0.82		0.94	0.63	0.75		0.98	0.72	0.83
Dnn, Conf. 1	Epochs=1500, Batch$_size$ = 1000000, $Loss$ = 0.0265	0	0.9321	0.92	1.00	0.96	0.8953	0.87	0.98	0.92	0.9509	0.95	1.00	0.97
		1		1.00	0.68	0.81		0.95	0.75	0.84		0.98	0.78	0.87
Dnn, Conf. 2	Epochs=2000 Batch$_size$ = 100000, 0.0307	0	0.9247	0.91	1.00	0.95	0.8885	0.86	0.99	0.92	0.9587	0.95	1.00	0.97
		1		1.00	0.65	0.79		0.98	0.71	0.82		0.98	0.82	0.89
Dnn, Conf. 1 Validat.	Epochs=1500, Batch$_size$ = 1000000, 0.0297	0	0.9214	0.91	1.00	0.95	0.8578	0.83	098	0.90	0.9183	0.91	1.00	0.95
		1		1.00	0.63	0.77		0.95	0.65	0.77		0.97	0.63	0.76
Dnn, Conf. 1 Early Stop.	Epochs=1500, Batch$_size$ = 1000000, 0.0271	0	0.8968	0.88	1.00	0.94	0.8402	0.80	0.99	0.89	0.9219	0.91	1.00	0.95
		1		1.00	0.52	0.68		0.98	0.57	0.72		0.98	0.64	0.77
Random forest	N$_e ss$ = 100, $gini$, max_depth = $none$, $max feat$ = $log2$	0	0.9930	0.99	0.99	0.99	0.9928	0.91	0.98	0.95	0.9532	0.95	0.99	0.97
		1		0.99	0.98	0.98		0.96	0.84	0.89		0.98	0.79	0.88
Random forest	N$_e ss$ = 100, $gini$, max_depth = 4, $max feat$ = $log2$	0	0.9898	0.99	0.98	0.99	0.9163	0.89	0.99	0.94	0.9858	0.99	0.99	0.99
		1		0.96	0.99	0.98		0.98	0.79	0.87		0.98	0.95	0.97
Bagging	Svc(c=1000) N.est=100	0	0.9922	0.99	1.00	1.00	0.8747	0.84	1.00	0.91	0.9358	0.93	1.00	0.96
		1		0.99	0.97	0.98		0.99	0.66	0.79		0.98	0.71	0.82
Bagging	Rfclass N.est=100	0	0.9922	0.99	1.00	1.00	0.87	0.84	1.00	0.91	0.9358	0.93	1.00	0.96
		1		0.99	0.97	0.98		0.99	0.66	0.79		0.98	0.71	0.82
Boosting	,none N.estim=100	0	0.9943	0.99	0.99	0.99	0.9201	0.90	0.98	0.94	0.94	0.94	0.98	0.96
		1		0.99	0.99	0.99		0.96	0.82	0.88		0.92	0.78	0.84
Boosting	Rfclass N:estim=100	0	0.9943	0.99	0.99	0.99	0.9208	0.90	0.99	0.94	0.95	0.94	0.99	0.97
		1		0.98	0.98	0.98		0.98	0.80	0.88		0.98	0.77	0.87
Linear svm	c =10	0	0.9906	1.00	0.99	0.99	0.9557	0.95	0.98	0.97	0.9801	0.98	0.99	0.99
		1		0.96	1.00	0.98		0.97	0.91	0.94		0.98	0.93	0.95
Linear svm	C=100	0	0.9902	1,00	0,99	0,99	0.96	0.95	0.98	0.97	0.9693	0.97	0.99	0.98
		1		0.96	1,00	0.98		0.97	0.91	0.94		0.98	0.87	0.92
Nonlinear svm	(kernel ='linear')	0	0.9902	1.00	0.99	0.99	0.9790	1.00	0.97	0.98	0.9928	1.00	0.99	1.00
		1		0.96	1.00	0.98		0.95	1.00	0.97		0.97	1.00	0.98
Nonlinear svm	(kernel='rbf')	0	0.9906	1.00	0.99	0.99	0.9779	1.00	0.97	0.98	0.9936	1.00	0.99	1.00
		1		0.96	1.00	0.98		0.95	1.00	0.97		0.98	0.99	0.98

Table 5: Compilations of different classification results

8 Time Series Analysis

8.1 Data Preparation

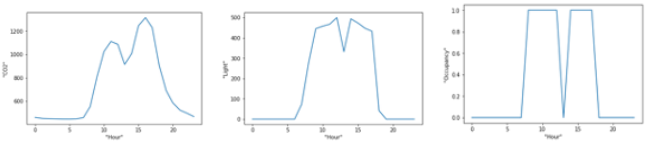
Before working on the time series, itself, we prepare and exploit the temporal informations. We use as features Light and CO2, because are the most correlated to the Occupancy, as we saw in the first phase of the report. So, we have created two different datasets, one with date, CO2 and Occupancy, the other with date, Light and Occupancy. To have a visual representation of the data, we plotted the time series of Light, CO2 and Occupancy, as we can see from the figure 21.



(a) CO2 (b) Light (c) Occupancy

Figure 21: Time series all day

Looking at the three pictures, we notice that in the working day (05/02-06/0-09/02-10/02) the features have higher values for all the two features. In the Occupancy time series, we can see that in the week end the class is always zero, while in the working day it assumes class zero or one, according to the different hours of the day.



(a) CO2 (b) Light (c) Occupancy

Figure 22: Value for hour for the average working day

In figure 22 we plotted the average for hour of the working days, for Light and CO2. In figure 22c, instead, we can see the predominant class of Occupancy for hour. Looking at the Occupancy plot we can say that the room is occupied, often, from 8 to 12, and from 14 to 17. The Light and CO2 time series have a similar path, in fact, from 8 to 12, and from 14 to 17 both CO2 and Light show us higher values than the other hours. We decided to use hours as temporal split, because the most significant variations happen in this time-frame.

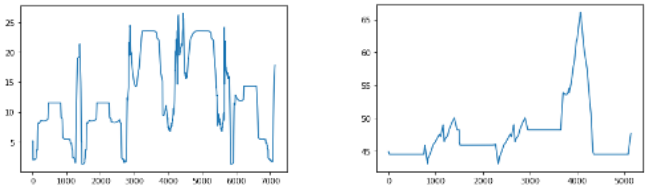
8.2 Analyze datasets

8.2.1 Motif and Anomaly Detection

Our goal is to detect repeated patterns with a typical shape that are called motifs and patterns that are different from the typical shape, anomalies. We plot the matrix profile of the TS with our feature Light and CO2. Then, apply the matrix with stomp, with the logarithmic mov difference and in the end stomp with, as timeseries values, the results of the logarithmic mov difference. We use different time windows to compare the results.

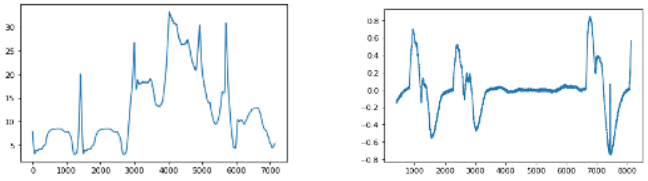
The best results came from Stomp for CO2 and Light too. For Light, we find really interesting results with a time window of w = 1000 (fig.23a). We start noticing some repetitive patters, but we decide to go deeper. We try another time window, w = 3000 (fig.23b) and we notice that there is a repetitive pattern and then a peak.

For CO2, we did the same. We found really interesting results with a time window of w = 1000 (fig.27a). We noticed some repetitive patterns. We try another time window, w = 400 (fig.24b) and we noticed that there is a repetitive pattern and then a flat pattern. We presume this result as the effect of the weekend, when few people are inside the building and the level of CO2 decreases.



(a) Windows size=1000 (b) Windows size=3000

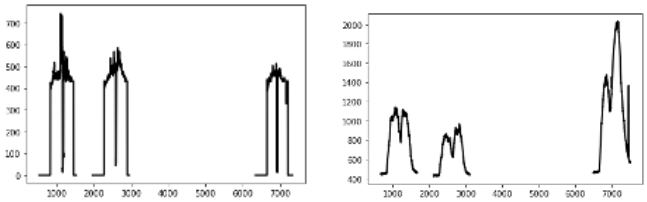
Figure 23: Light, stomp



(a) Windows size=1000 (b) Windows size=400

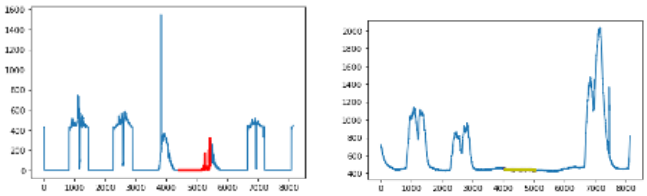
Figure 24: CO2, stomp

Figures 27 represents the motifs we find for the feature Light and CO2 during the weekend days (yellow one), saturday and sunday and the motifs for the working days (black one), in our case thursday, friday and monday. In general, there is a de-



(a) Light (b) CO2

Figure 25: Motifs detection for weekdays



(a) Light (b) CO2

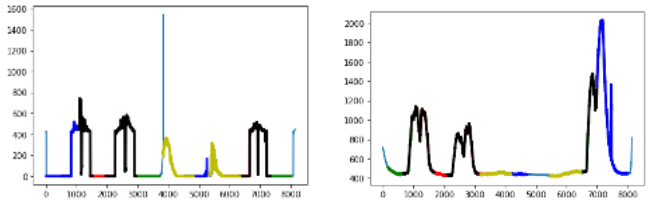
Figure 26: Anomaly detection for all days

crease of level of Light and CO2 during the weekend. In fig.25, we can notice the black pattern, that describes the motif of working days for light and CO2.

For what concert anomaly detection, the shape of the working days is basically similar, with only a higher peak before lunch of Thursday. For Light (fig.26a), using as reference the matrix profile with stomp and a time window of w = 1000, we find an anomaly (red one) and a huge peak on Saturday morning. For CO2 , the anomaly we find is the yellow one in the weekend (fig.26b).

8.2.2 Shapelet

A shapelet is a pattern/subsequence which is maximally representative of a class with respect to a given dataset of time series. Before starting to investigate which are the most representative forms (shapelets) of the two classes, both in the light and co2 time series, we created two new dataset, one for light and one for co2. In these new datasets we divided light and co2 by days, thus creating 5 new distinct features, to which we associated the most representative class by "record". Then we moved on to the analysis of the most significant shapelet for both light and co2, we can see them in the figure 28 and 29.

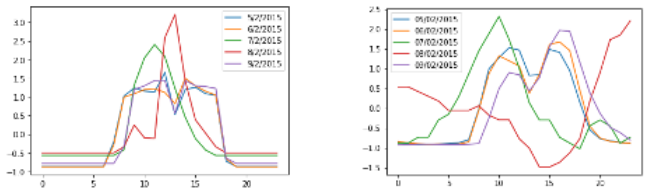


(a) Light (b) CO2

Figure 27: Motifs detection for all days

8.3 Distance and Clustering

In the dataset created, with the features light and CO2, for each day, we have computed the distance between the records for a specific day and all the others and observed how the distances change. To have a better graphical representation, in fig.30a, we can see how the time series of the single days are fitted for Light, while in the fig.30b we can see how they are in CO2. Both, are plotted after the Amplitude Scaling transformation. We can observe that the time series of the working days have a similar shape (05/02-06/02-09/02), both for Light and CO2,while, the time series for the weekend days (07/02-08/02) have different shapes compared to them and to the weekend days themselves.



(a) Light (b) Co2

Figure 30: Time series for each day

Then, in Tab.17 and Tab.18 we summarize the results for the Euclidean distance, Manhattan distance and Dynamic time warping between the records. We notice that the distance between the time series for two weekdays are lower respect to the distance between a weekday and a weekend day or between the two weekend days, for Light and CO2. These results confirmed our initial hypothesis that the working days are very similar between them and that they are different from the weekend days.

In fig. 31 we plotted the optimal path in the Dtw matrix for the days 05/02 and 09/02, we can notice that Light optimal path follows almost perfectly the diagonal, so the two time series are superimposable, instead, the optimal path for CO2 deviates slightly

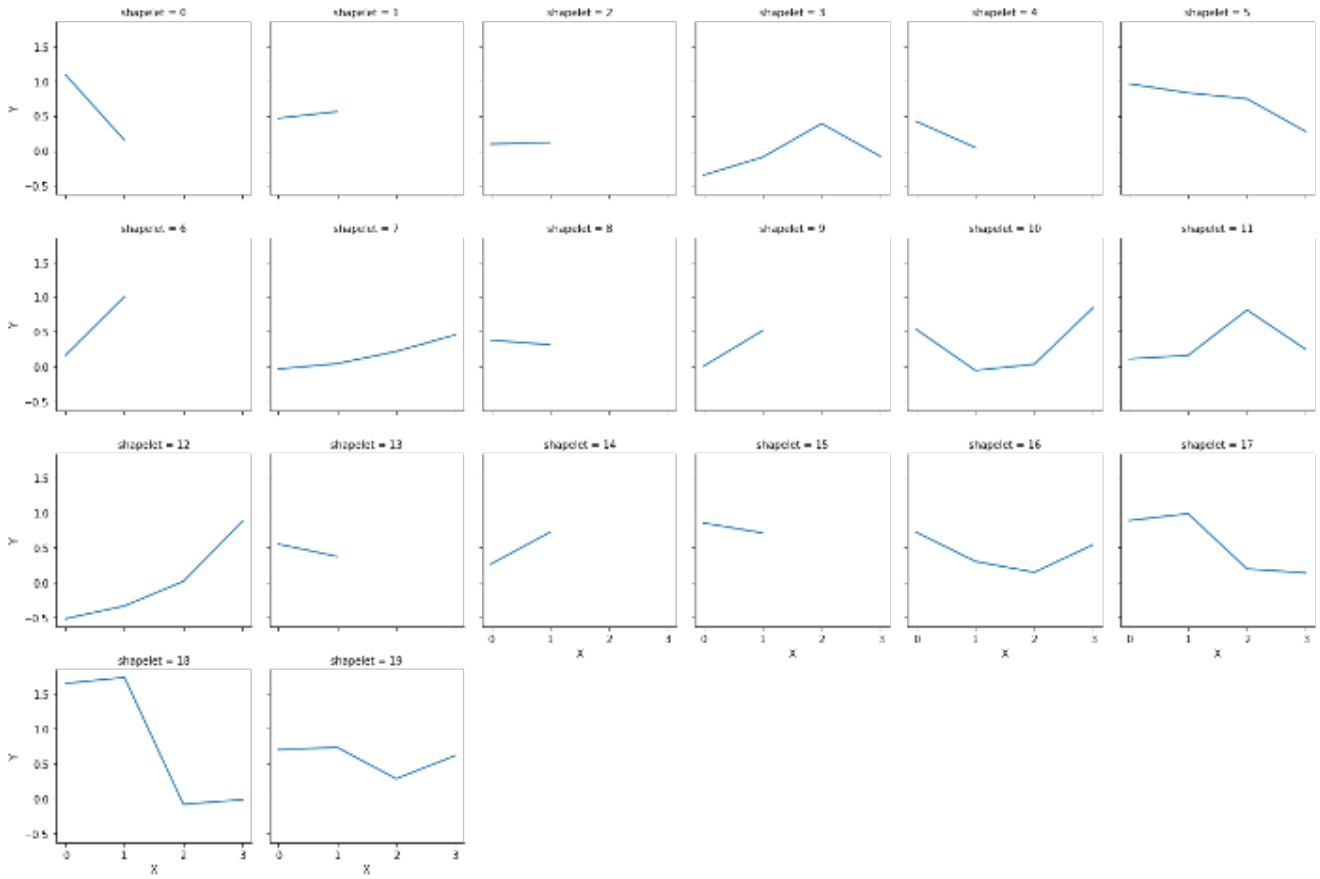


Figure 28: Shapelet CO2

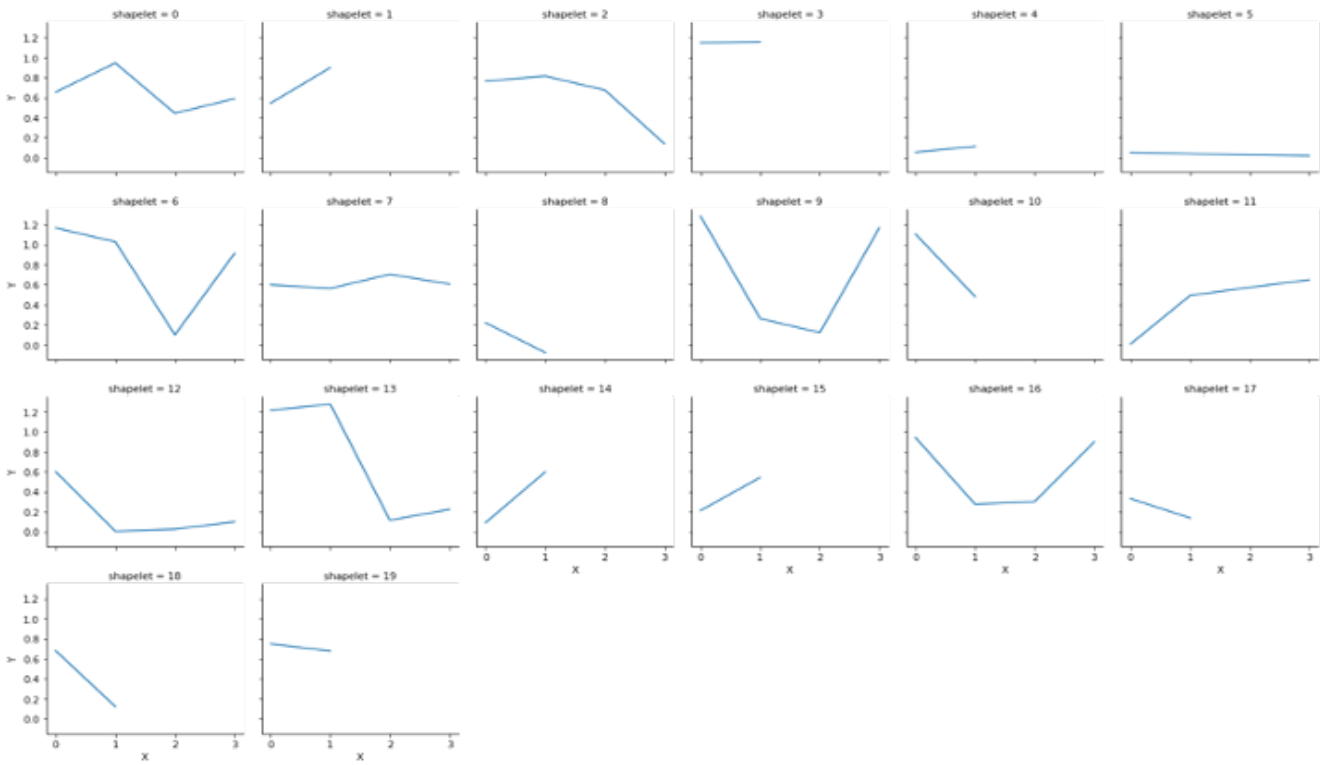
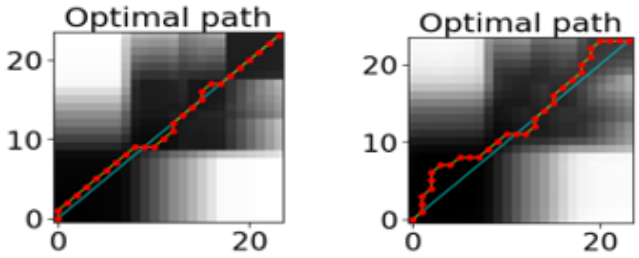


Figure 29: Shapelet Light

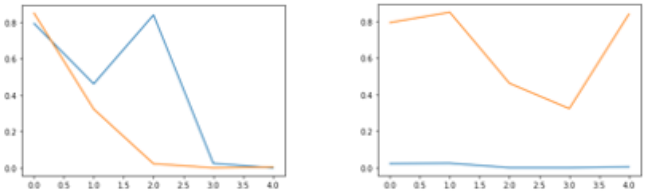
from the diagonal in its path, so we can say that the two time series are similar (as we can notice from the Dtw, Euclidean and Manhattan distance) but at different time moments.



(a) Light (b) Co2

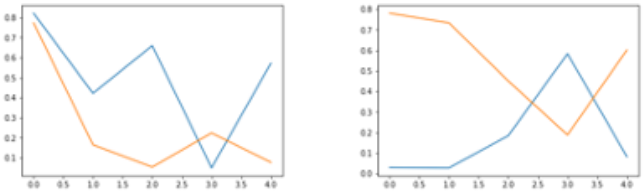
Figure 31: Dtw 5/2 -9/2

Finally, we applied two different techniques to discover the clusters, Shape based and Feature Based clustering. The results that we have obtained are summarized in the table 19. To evaluate the accuracy of clusters, we consider the inertia of them. The lowest the inertia is, the closest the points in a cluster are. The most reliable technique in this case is the Shape based, because it has an inertia of 0.17 for CO2 and of 0.13 for Light, while , the features based have an inertia of 6.69 for CO2 and of 9.52 for Light.



(a) Shape based (b) Feature based

Figure 32: Clustering of Light



(a) CO2 shape based (b) CO2 feature based

Figure 33: CO2

8.4 Forecasting

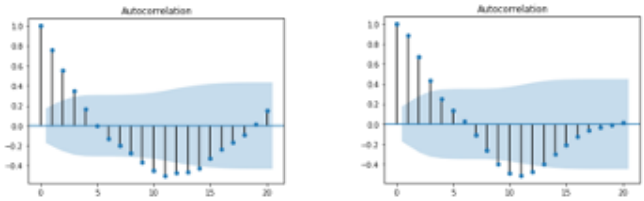
In this part we will try to forecast the evolution of the time series. We used the dataset that have a single features for the entire week, both for Light and CO2. We use as reference time split the hours. To make the time series stationary we applied the logarithmic move difference transformation Fig.37, that allow us to reach the results that we can see in table 7 and 6, where we have reported the results of the Dickey Fuller tests.

Results of Dickey-Fuller Test:	
Test Statistic	-5.934107e+00
p-value	2.342974e-07
Lags Used	1.300000e+01
Number of observations used	1.220000e+02
Critical Value (1%)	-3.485122e+00
Critical Value (5%)	-2.885538e+00
Critical Value (10%)	-2.579569e+00

Table 6: Results of Dickey-Fuller Test Light

Results of Dickey-Fuller Test:	
Test Statistic	-5.034919
p-value	0.000019
Lags Used	5.000000
Number of observations used	119.000000
Critical Value (1%)	-3.486535
Critical Value (5%)	-2.886151
Critical Value (10%)	-2.579896

Table 7: Dickey Fuller Test CO2



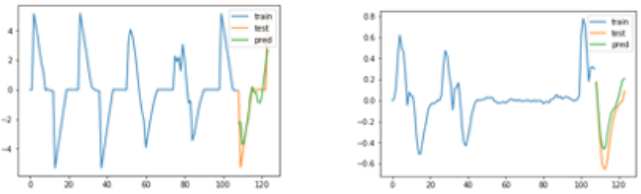
(a) Light (b) CO2

Figure 34: Autocorrelation

How we can see, since the test statistic is smaller than the three critical values the time series transformed are stationary, and with a p-value associated to this test smaller than the 5% so we can conclude that these two tests are statistically significant and reliable.

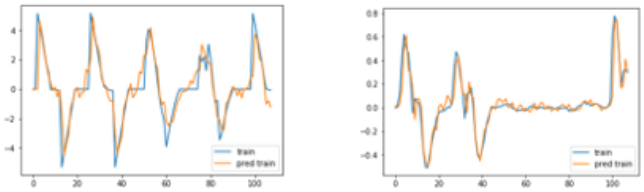
Then we plotted the autocorrelation (fig.34), how we can see there are a meaningful autocorrelation until the third lag in both the features. We tried to forecast with different techniques, i.e. exponential smoothing, arima and sarimax. The best results that we have obtained were given by the arima tech-

nique in both the features.



(a) Light (b) Co2

Figure 35: Prediction tested over the last piece of time series

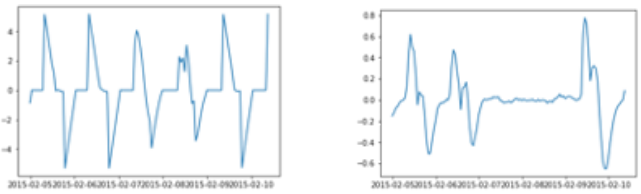


(a) Light (b) Co2

Figure 36: Prediction over the entire time series

To evaluate the forecasting of the two features over the entire time series we used the plot, fig 36 and different formulas: mae, rmse, mad, r2, mape, maxape and tape. The figure 35 are referred to Light and CO2, have been drawn by "cutting" the last piece of the two time series and going to use it as test, i.e. to evaluate the model's ability to predict that last piece of the time series, the rest of the series is used as training. In the figure 36, these too referred, respectively to Light and CO2, instead, we used the entire time series like test, to evaluate the goodness of the model trained to forecast an whole time series.

In Light we wasn't able to perform good results how we can see in the table 8 and from the two figures 35a and 36a, in particular, if we look at the figure 35a we can see that the prediction are far from the piece of the time series used like test, while the R2 are relevant the other indicator are not good like the same indicator performed by the two test of the CO2, indeed, with CO2 features, how we can see in figure 35b and 36b, we was able to obtain good results, the prediction over the last piece of time series are closely to the test and present a very similar shape. Watching at figure 36b we notice that the prediction over the entire time series, excluding some point of deviation is almost superimposable.



(a) Light (b) Co2

Figure 37: Logmovdiff transformation

Features	Test	MAE	RMSE	MAD	R2	MAPE	MAXAPE	TAPE
Light	15%	0.894	1.274	0.573	0.732	0.771	1.367	12.336
	entire	0.620	1.026	0.418	0.772	33.555	3548.83	3623.97
CO2	15%	0.126	0.145	0.132	0.828	0.895	2.572	14.326
	entire	0.050	0.085	0.028	0.859	2.174	70.398	234.772

Table 8: Score of the different index for the prediction on the test over Light and CO2 time series.

8.5 Classification

In this part we will try to train different model to label/classify the records in class one or zero of Occupancy, on the datasets divided for days with time frame of hours, both for Light and CO2. We used different approaches: features based, shape based,distance from the shapelet of the time series and CNN. For all the different approaches, we used as models KNeighbors with k=3 and a Decision tree with criterion Gini, splitter random, no maximum depth, minimum sample split of 2 and minimum sample leaf of 1. For judge the goodness of the model applied on the different approaches, we used these metrics: accuracy, precision, recall and f1. After the training of the model on the training set, we have tested the model on the two other different datasets, test 1 and test 2, properly prepared by maintaining in the datasets only light and co2 according to the training set used. In the first approach, shape based, we used the shapelet model function from "tslearn" to extract the features belonging to the two different classes, then on the shapelet found in this way, we used the two classifications model to classify the records. In the shapelet distance approach, we transformed the time series as vector of the same time series with the distance between this time series and the shapelet belonging to the two different classes. In the feature based, in a first moment we used a function "calculate_features" to extract different features, these are: average, standard deviation, variance, median, 10-25-50-75-90 percentile, skewness and kurtosis indexes. Then on the vector created by the calculation of these different features we applied

the two model KNeighbors and Decision Tree. In the Convolutional neural networks we created four layer with the filters of, respectively size: 32,64,128, dropout regularization of 0.5 and an average pooling at the end. All the models classify very well in the training set and in the test set one, instead in the test set two the scores falls, the best models seems to be the shape based.

8.6 Whole dataset classification

To be able to see the difference in terms of scores, of the metrics index used, with the univariate classification, we have used the entire dataset to try to predict the class of records. We used to do this, and to be able to confront the results with the univariate classification (see point 3.4), the same approach and models, i.e. shape based, distance with the shapelet ,feature based and CNN. If we look at the table 22, where we have summarized the results of the different classification model that we have used, we can notice that we obtained an increase of the scores for all the different metrics used. In this case the best models to classify the records are the convolutional neural networks, that reach and accuracy of 0.9877 in the training set and f 0.9677 in the test set one and of 0.9311 in the test set two.

9 SPM

In this part of the project, we will search for the combinations of events (patterns) that appear frequently in the data. We use as time split the hours. We have followed the same process for all the features. We have plotted the timeseries, and converted it in the relative numeric values. We have scaled the timeseries with 'TimeSeriesScalerMeanVariance(mu=0., std=1.)'. We have applied on it the SAX transformation with number of segments = 20 and number of symbols = 10. All the values that we have obtained are transformed in contiguous values, we have used a map_symbols function that assign to every value, for each features, a specific symbol from 0 to 10. At the end we have created different array with the remapped values with the simbols for 'Light', 'Temperature', 'CO2', 'Humidity' (table: 10).

Then we have created a list with every map symbols of the different features. On this list, we have applied Prefix Span technique to find the most frequent patterns.We used as minimum support level 3 and 4. The best 15 patterns that we have found are represented in table 9.

From the list we can observe on the left, the number of times that the path is found, on the right, the list defining the pattern. In general the best 10 paths are found 4 times.

Frequence	Patterns
4	1,1,1,1,1,1,0,0,0,0,0,0,5,5,5,5,5,5
4	1,1,1,1,1,1,5,5,5,5,5,5
4	4,4,4,4,4,4,1,1,1,1,1,1
4	4,4,4,4,4,4,1,1,1,1,1,1,0,0,0,0,0,0
4	4,4,4,4,4,4,3,3,3,3,3,3
4	4,4,4,4,4,4,6,6,6,6,6,6
4	4,4,4,4,4,4,6,6,6,6,6,6,2,2,2,2,2,2
4	6,6,6,6,6,6,0,0,0,0,0,0,5,5,5,5,5,5
4	6,6,6,6,6,6,1,1,1,1,1,1,0,0,0,0,0,0
4	6,6,6,6,6,6,1,1,1,1,1,1,1,1,1,1,1,1,1,1
4	6,6,6,6,6,6,3,3,3,3,3,3
4	6,6,6,6,6,6,5,5,5,5,5,5
4	6,6,6,6,6,6,6,6,6,6,6,6,
3	4,4,4,4,4,4,6,6,6,6,6,6,5,5,5,5,5,5
3	4,4,4,4,4,4,6,6,6,6,6,6,6,6,6,6,6,6

Table 9: Most frequent patterns.

10 Outlier Detection

"An outlier is and observation which deviates so much from the other observation as arouse suspicions that it was generated by a different mechanism"(Stephen Hawking, 1980). To know if our training set have and which are, if there are, these outliers, we used different methods, these are: dbscan,Local outlier factor, autoencoder,knn. We, obtained for all the different techniques used, a "list" of the possible outlier with an associated scores, that differ between different techniques. Then to search for the most meaningful outliers we have imposed a threshold, one for each model,according to the different measure used in the model, to be able to select only the "best" outliers found by each models. The two most "meaningful" outliers that we have found, that was found by all the techniques, are the records with index 3831 and 3832, if we look at the values for light in this two records and then at the figure of the corresponding time series of light, we can notice that this two index correspond to the big peak in the time series.

Feature	Remapped Values
Temperature	10,10,10,10,10,10,7,7,7,7,7,7,8,8,8,8,8,8,10,10,10,10,10,8,8,8,8,8,8,4,4,4,4,4,4,2,2,2,2,2,2,3,3,3,3,3,3,10,10,10,10,10,10,2,2,2,2,2,2,,1,1,1,1,1,1,0,0,0,0,0,0,2,2,2,2,2,2,1,1,1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,9,9,9,9,9,9,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
Humidity	6,6,6,6,6,6,4,4,4,4,4,4,3,3,3,3,3,3,6,6,6,6,6,6,2,2,2,2,2,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,4,4,4,4,4,4,8,8,8,8,8,8,9,9,9,9,9,9,7,7,7,7,7,7,7,7,7,7,7,7,6,6,6,6,6,6,7,7,7,7,7,7,10,10,10,10, 10,10,5,5,5,5,5,5,5,5,5,5,5,5,5,5
CO2	2,2,2,2,2,2,1,1,1,1,1,1,4,4,4,4,4,4,6,6,6,6,6,6,2,2,2,2,2,2,0,0,0,0,0,0,2,2,2,2,2,2,5,5,5,5,5,5,2,2,2,2,2,2,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,4,4,4,4,4,4,6,6,6,6,6,6,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3
Light	1,1,1,1,1,1,0,0,0,0,0,0,5,5,5,5,5,5,6,6,6,6,6,6,1,1,1,1,1,1,0,0,0,0,0,0,5,5,5,5,5,5,6,6,6,6,6,6,1,1,1,1,1,1,0,0,0,0,0,0,1,1,1,1,1,1,4,4,4,4,4,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,3,3,3,3,3,3,6,6,6,6,6,6,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2

Table 10: Remapped values for the four different attributes used for spm.

MODELS	INDEX OF THE OUTLIERS									
DBSCAN	1156	1181	1453	2606	3830	3831	3832	3833	6943	7213
LOF	1181	2606	3830	3831	3832	3833	4236	4238	4239	7178
AUTOENCODER	3831	3832	7153	7154	7160	7169	7170	7171	7172	7173
KNN	1156	1181	1453	2606	3830	3831	3832	3833	6943	7213

Table 11: Index of the most meaningful outliers

	LinearRegression()										**Ridge()**								
		Training DataSet			Test DataSet 1			Test DataSet 2			Training DataSet			Test DataSet 1			Test DataSet 2		
	Pairs	r2	mse	mae	r2	mse	mae	r2	mse	mae	r2	mse	mae	r2	mse	mae	r2	mse	mae
1	TemperatureHumidity	0.02	0.98	1	-0.223	1.223	1	0.114	0.886	1	0.02	0.98	1	-0.223	1.223	1	0.114	0.886	1
2	TemperatureLight	0.422	0.578	1	0.576	0.424	0	0.495	0.505	1	0.422	0.578	1	0.576	0.424	0	0.495	0.505	1
3	TemperatureCO2	0.313	0.687	1	0.661	0.339	0	-0.09	1.09	1	0.313	0.687	1	0.661	0.339	0	-0.09	1.09	1
4	TemperatureHumidityRatio	0.023	0.977	1	0.248	0.752	1	-0.026	1.026	1	0.023	0.977	1	0.248	0.752	1	-0.026	1.026	1
5	HumidityLight	0.001	0.999	1	0.041	0.959	1	-0.016	1.016	1	0.001	0.999	1	0.041	0.959	1	-0.016	1.016	1
6	HumidityCO2	0.193	0.807	1	0.608	0.392	1	-0.262	1.262	1	0.193	0.807	1	0.608	0.392	1	-0.262	1.262	1
7	HumidityHumidityRatio	0.912	0.088	0	0.906	0.094	0	0.777	0.223	0	0.912	0.088	0	0.906	0.094	0	0.777	0.223	0
8	LightCO2	0.441	0.559	0	0.581	0.419	0	-0.188	1.188	1	0.441	0.559	0	0.581	0.419	0	-0.188	1.188	1
9	LightHumidityRatio	0.053	0.947	1	0.266	0.734	1	0.022	0.978	1	0.053	0.947	1	0.266	0.734	1	0.022	0.978	1
10	CO2HumidityRatio	0.393	0.607	1	0.816	0.184	0	-0.338	1.338	1	0.393	0.607	1	0.816	0.184	0	-0.338	1.338	1

Table 12: Regression Results

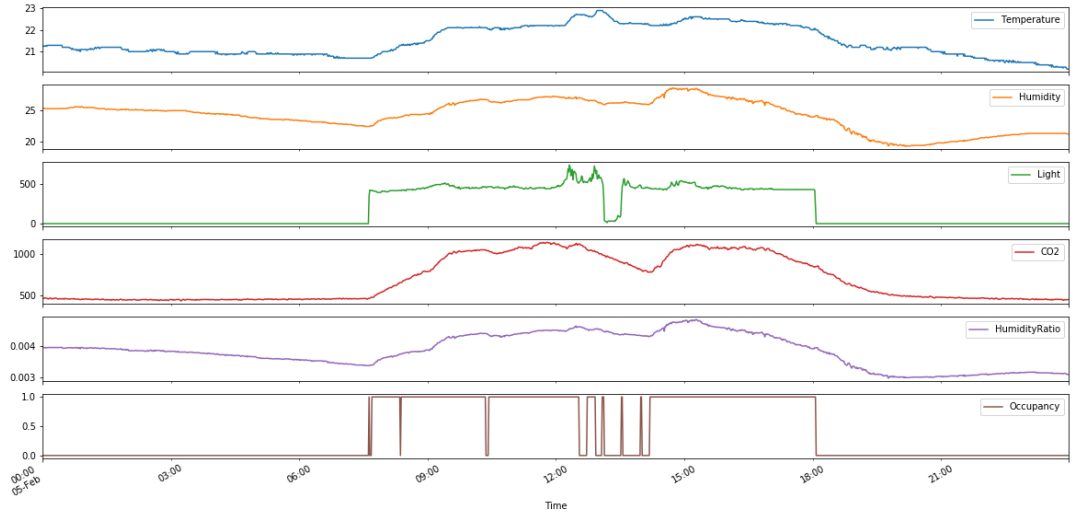


Figure 38: Measurement profiles for 1 day (morning of February 2, 2015 to February 3, 2015).

	Temperature	Humidity	Light	CO2	Humidity ratio	Week Day	Occupancy
Temperature	0.000000e+00	8.045844e-38	0.000000e+00	0.000000e+00	3.731126e-43	0.000000e+00	0.000000e+00
Humidity	8.045844e-38	0.000000e+00	6.396081e-04	0.000000e+00	0.000000e+00	8.957692e-23	1.934809e-33
Light	0.000000e+00	6.396081e-04	0.000000e+00	0.000000e+00	1.384254e-98	4.848841e-146	0.000000e+00
CO2	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	5.602596e-302	0.000000e+00
Humidity ratio	3.731126e-43	0.000000e+00	1.384254e-98	0.000000e+00	0.000000e+00	6.770917e-110	2.618870e-169
Week Day	0.000000e+00	8.957692e-23	4.848841e-146	5.602596e-302	6.770917e-110	0.000000e+00	1.698381e-284
Occupancy	0.000000e+00	1.934809e-33	0.000000e+00	0.000000e+00	2.618870e-169	1.698381e-284	0.000000e+00

Table 13: P-value associated at the correlation matrix.

Models	Class	Training set				Test set 1				Test set 2			
		Acc	Prec	Rec	F1	Acc	Prec	Rec	F1	Acc	Prec	Rec	F1
K-nn, k=3	0	0.9926	0.99	0.99	0.99	0.9594	0.96	0.96	0.96	0.9845	0.99	0.98	0.99
	1		0.98	0.98	0.98		0.94	0.94	0.94		0.95	0.97	0.96
Naïve bayes Gaussian	0	0.9905	0.99	0.98	0.99	0.9579	0.96	0.97	0.97	0.9867	0.99	0.98	0.99
	1		0.96	0.99	0.97		0.95	0.93	0.94		0.94	0.99	0.97
Naïve bayes Categorical	0	0.9826	0.99	0.99	0.99	0.9444	0.95	0.96	0.96	0.9856	0.99	0.99	0.99
	1		0.95	0.97	0.96		0.93	0.91	0.92		0.95	0.98	0.97
Logistic regression,co2-light	0	0.9898	0.99	0.99	0.99	0.9017	0.87	0.99	0.93	0.9876	0.99	0.99	0.99
	1		0.96	0.99	0.98		0.99	0.74	0.85		0.98	0.96	0.97
Logistic regression, all attributes	0	0.9881	0.99	0.99	0.99	0.8931	0.86	0.99	0.92	0.9553	0.95	0.99	0.97
	1		0.96	0.99	0.98		0.97	0.73	0.83		0.97	0.82	0.89
Dec.tree,entropy, random splitter 3,20,1*	0	0.9902	1.00	0.99	0.99	0.9549	0.95	0.98	0.96	0.9362	0.94	0.98	0.96
	1		0.96	1.00	0.98		0.96	0.92	0.94		0.90	0.78	0.84
Dec. Tree, gini, random splitter 15,10,30*	0	0.9865	0.99	0.99	0.99	0.9711	1.00	0.95	0.98	0.9617	0.99	0.96	0.98
	1		0.98	0.98	0.97		0.93	1.00	0.96		0.86	0.98	0.91

Table 14: Performance of different classification models

MODELS	ATTRIBUTES	TRAINING SET				TEST SET 1				TEST SET 2			
		ACC	PREC	REC	F1	ACC	PREC	REC	F1	ACC	PREC	REC	F1
Dec.Tree,Entropy, random splitter 3,20,1*	PCA 0.9843	0.9243	0	0.93	0.98	0.95	0.7617	0.74	0.96	0.84	0.8618	0.93	0.89	0.91
			1	0.91	0.72	0.80		0.86	0.42	0.56		0.65	0.74	0.69
Dec. Tree, Gini, random splitter 15,10,30*	PCA 0.9640	0.9820	0	0.99	0.99	0.99	0.9021	0.89	0.97	0.93	0.8444	0.96	0.84	0.90
			1	0.95	0.97	0.96		0.93	0.79	0.85		0.59	0.85	0.70

Table 15: PCA results.

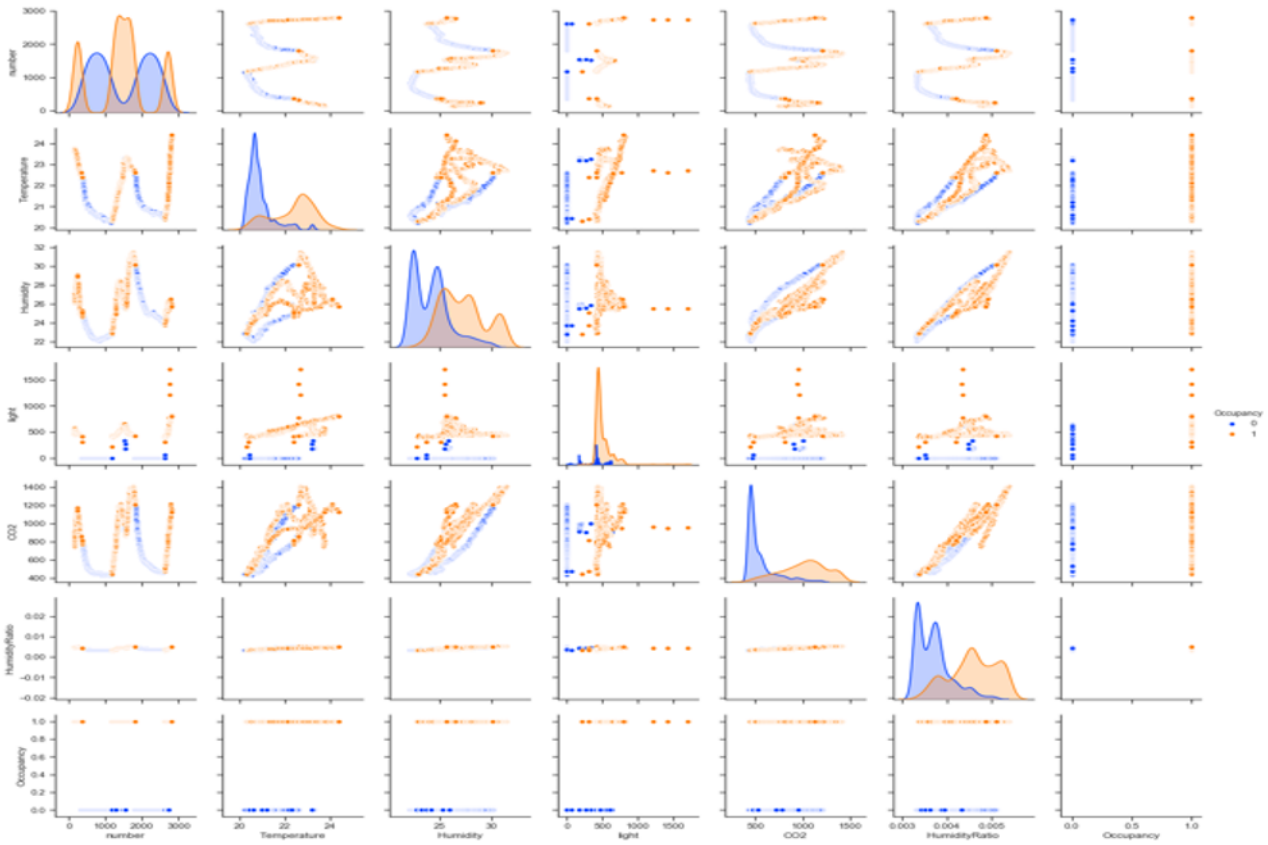


Figure 39: Pair Plot

MODELS	ATTRIBUTES	CLASS	TRAINING SET				TEST SET 1				TEST SET 2			
			ACC	PREC	REC	F1	ACC	PREC	REC	F1	ACC	PREC	REC	F1
Dec. Tree, Gini, random splitter 15,10,30*	96%-4%	0	0.9785	1.00	0.98	0.99	0.6893	0.67	1.00	0.80	0.8712	0.89	0.95	0.92
		1		0.67	0.93	0.77		0.95	0.16	0.27		0.76	0.56	0.65
Dec. Tree, Gini, random splitter 15,10,30*	Class weight	0	0.9745	1.00	0.98	0.99	0.9366	0.95	0.95	0.95	0.9156	0.92	0.97	0.95
		1		0.62	0.94	0.75		0.91	0.92	0.91		0.91	0.92	0.91
Dec. Tree, Gini, random splitter 15,10,30*	Random Undersamp..	0	0.9885	1.00	0.99	0.99	0.7295	0.70	0.99	0.82	0.9931	1.00	0.99	1.00
		1		0.78	0.99	0.87		0.96	0.27	0.42		0.98	0.99	0.98
Dec. Tree, Gini, random splitter 15,10,30*	Random Over Sampling	0	0.9885	1.00	0.99	0.99	0.9508	0.94	0.98	0.96	0.9942	1.00	0.99	1.00
		1		0.78	0.99	0.87		0.97	0.90	0.93		0.98	1.00	0.99
Dec. Tree, Gini, random splitter 15,10,30*	Smote	0	0.9885	1.00	0.99	0.99	0.8878	0.86	0.99	0.92	0.9944	1.00	0.99	1.00
		1		0.78	0.99	0.87		0.97	0.72	0.82		0.98	1.00	0.99

Table 16: Results of the decision tree, gini, random, 15,10,30 before and after the techniques to handle with imbalanced datasets

Time Series	Euclidean Distance	Manatthan Distance	Dinamic Time Warping
05/02-06/02	0.96949764365691	2.90118587373537	0.8925374460445021
05/02-07/02	4.91070963748967	18.5392865325682	1.4617681164688112
05/02-08/02	8.94309386397436	39.4612397172196	6.595252705790322
05/02-09/02	2.76914683288357	9.79287881443782	1.3345378598079805
06/02-07/02	5.36742739293713	19.2958048513574	1.475192152673260
06/02-08/02	8.98999599060598	39.3655132442521	6.54145602888618
06/02-09/02	2.22486148364910	7.39655138873138	0.9604645610649898
07/02-08/02	7.43448850927837	31.0687738778310	6.285979243907387
07/02-09/02	6.73088542405364	24.420885958181	1.0337651953953284
08/02-09/02	8.85677975762271	37.7881422184486	6.2185454093481125

Table 17: CO2 distance results

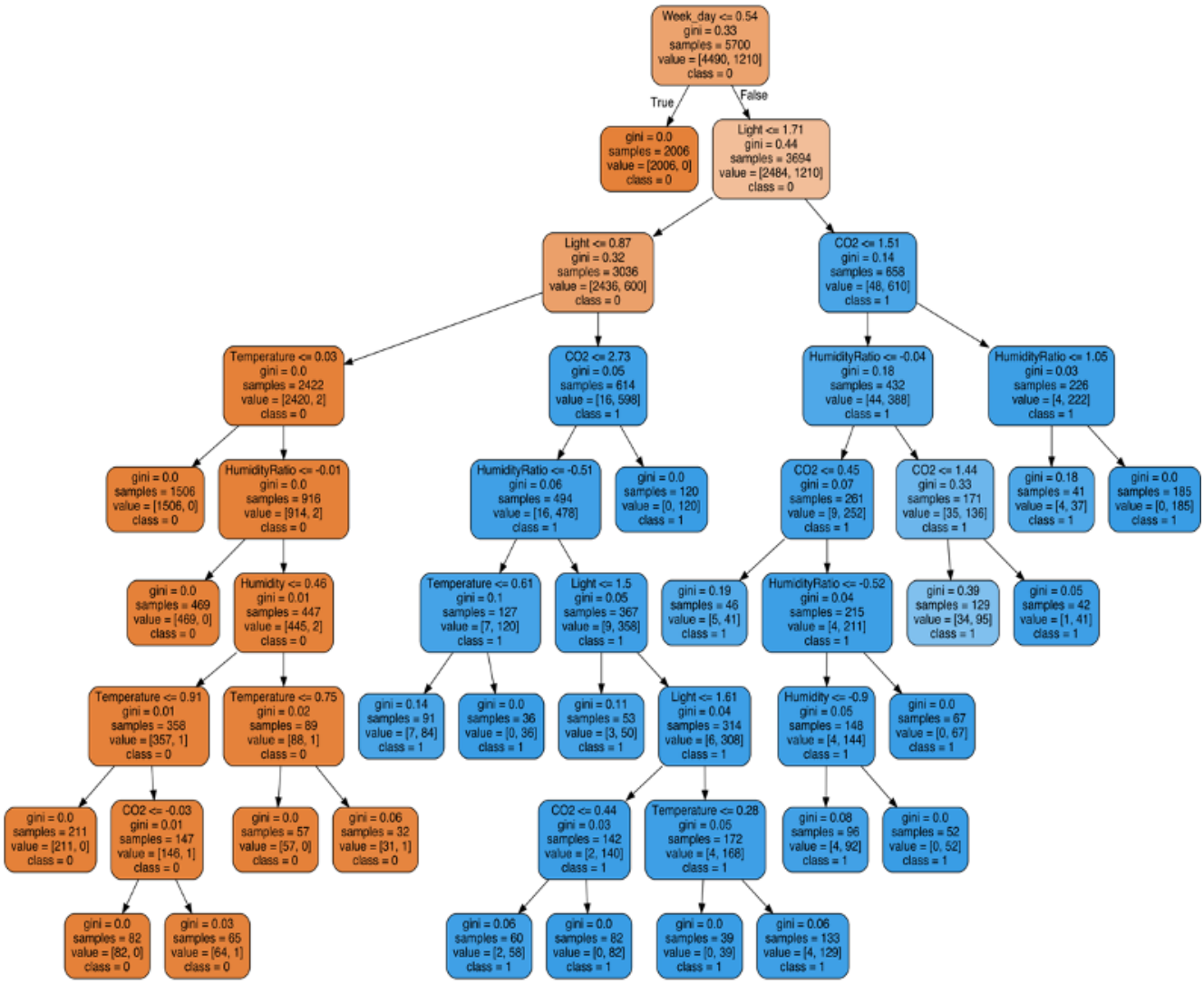


Figure 40: Visual representation of the tree referred to the try decision tree, gini with splitter

Time Series	Euclidean Distance	Manatthan Distance	Dinamic Time Warping
05/02-06/02	0.71250386198173	1.78992608928866	0.6950221709834908
05/02-07/02	3.70016116837591	14.0920170832187	1.8935556949056198
05/02-08/02	4.42972527431607	16.9164762600310	3.2720585179007284
05/02-09/02	1.58897232450579	4.34649463418315	0.6048120330013375
06/02-07/02	3.82475613244064	14.6887170981733	1.956289054362031
06/02-08/02	4.42150629059869	16.9864154082102	3.2896731481285775
06/02-09/02	1.54004660130247	4.26777490578588	0.648965048247632
07/02-08/02	4.21682724227027	11.4520246020618	1.9422903031286758
07/02-09/02	3.47389834001950	11.9793685578162	1.932606981607123
08/02-09/02	4.41578658016579	15.2866207106462	3.0210088417587846

Table 18: Light different results

Methods	TIME																							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Light, Shape based	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
Light,Feature based	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
CO2, Shape based	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
CO2,Feature based	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0

Table 19: Cluster for hour.

MODELS	ATTRIBUTES	CLASS	TRAINING SET				TEST SET 1				TEST SET 2			
			ACC	PREC	REC	F1	ACC	PREC	REC	F1	ACC	PREC	REC	F1
Shapelet Classifier	Adamax, weight=1, max$_i t$ = 1000	0	0.8946	0.91	0.96	0.94	0.8649	0.93	0.85	0.89	0.7828	0.86	0.87	0.86
		1		0.82	0.66	0.73		0.77	0.89	0.83		0.48	0.45	0.46
Shapelet Distance Classifier	KNeighbors K=3	0	0.9111	0.95	0.94	0.94	0.7745	0.87	0.75	0.81	0.6940	0.87	0.72	0.79
		1		0.77	0.82	0.80		0.65	0.81	0.72		0.36	0.61	0.46
Shapelet Distance Classifier	Decision tree Criterion=gini, splitter=rand. None,2,1	0	0.8989	0.93	0.95	0.94	0.7441	0.81	0.79	0.80	0.6895	0.86	0.73	0.79
		1		0.78	0.72	0.75		0.64	0.67	0.66		0.35	0.55	0.43
Feature Based Classifier	Decision tree Criterion=gini, splitter=rand. None,2,1	0	0.9001	0.93	0.94	0.94	0.7463	0.81	0.78	0.80	0.6928	0.86	0.73	0.79
		1		0.77	0.75	0.76		0.64	0.69	0.67		0.35	0.56	0.49
Feature Based	KNeighbors K=3	0	0.9124	0.95	0.94	0.94	0.7775	0.87	0.76	0.81	0.6961	0.87	0.72	0.79
		1		0.78	0.81	0.80		0.66	0.81	0.73		0.36	0.60	0.45
CNN	32,64,128	0	0.9218	0.98	0.92	0.95	0.8056	0.97	0.72	0.82	0.6566	0.89	0.64	0.75
		1		0.76	0.93	0.83		0.66	0.96	0.78		0.35	0.70	0.46

Table 20: Results of the models for the training set, test set 1 and 2 of Light.

MODELS	ATTRIBUTES	CLASS	TRAINING SET				TEST SET 1				TEST SET 2			
			ACC	PREC	REC	F1	ACC	PREC	REC	F1	ACC	PREC	REC	F1
Shapelet Classifier	Adamax, weight=1, max$_i t$ = 1000	0	0.9758	1.00	0.97	0.98	0.8570	0.95	0.82	0.88	0.7871	0.89	0.84	0.86
		1		0.90	0.99	0.95		0.75	0.92	0.82		0.49	0.60	0.54
Shapelet distance classifier	KNeighbors K=3	0	0.9853	0.99	0.99	0.99	0.7332	0.76	0.86	0.80	0.7936	0.87	0.87	0.87
		1		0.96	0.97	0.97		0.68	0.52	0.59		0.51	0.50	0.51
Shapelet distance classifier	Decision tree Criterion=gini, splitter=rand. None,2,1	0	0.9808	0.99	0.99	0.99	0.6470	0.66	0.91	0.77	0.8385	0.85	0.96	0.90
		1		0.96	0.95	0.96		0.54	0.22	0.32		0.72	0.38	0.49
Feature Based	Decision tree Criterion=gini, splitter=rand. None,2,1	0	0.9812	0.99	0.99	0.99	0.6447	0.66	0.91	0.77	0.8396	0.85	0.96	0.90
		1		1.00	0.67	0.80		0.54	0.18	0.27		0.73	0.38	0.50
Feature Based	KNeighbors K=3	0	0.9853	0.99	0.99	0.99	0.7336	0.76	0.86	0.80	0.7935	0.87	0.87	0.87
		1		0.96	0.97	0.97		0.68	0.52	0.59		0.51	0.50	0.51
CNN	32,64,128	0	0.9815	1.00	0.98	0.99	0.8578	0.94	0.82	0.88	0.7886	0.88	0.84	0.86
		1		0.93	0.99	0.96		0.75	0.92	0.82		0.50	0.58	0.53

Table 21: Results of the models for the training set, test set 1 and 2 of CO2.

MODELS	ATTRIBUTES	CLASS	TRAINING SET				TEST SET 1				TEST SET 2			
			ACC	PREC	REC	F1	ACC	PREC	REC	F1	ACC	PREC	REC	F1
Shapelet Classifier	Adamax, weight=1, max$_i t$ = 1000	0	0.9758	1.00	0.97	0.98	0.8570	0.95	0.82	0.88	0.7871	0.89	0.84	0.86
		1		0.90	0.99	0.95		0.75	0.92	0.82		0.49	0.60	0.54
Shapelet distance classifier	KNeighbors K=3	0	0.9853	0.99	0.99	0.99	0.7332	0.76	0.86	0.80	0.7936	0.87	0.87	0.87
		1		0.96	0.97	0.97		0.68	0.52	0.59		0.51	0.50	0.51
Shapelet distance classifier	Decision tree Criterion=gini, splitter=rand. None,2,1	0	0.9808	0.99	0.99	0.99	0.6470	0.66	0.91	0.77	0.8385	0.85	0.96	0.90
		1		0.96	0.95	0.96		0.54	0.22	0.32		0.72	0.38	0.49
Feature Based	Decision tree Criterion=gini, splitter=rand. None,2,1	0	0.9812	0.99	0.99	0.99	0.6447	0.66	0.91	0.77	0.8396	0.85	0.96	0.90
		1		1.00	0.67	0.80		0.54	0.18	0.27		0.73	0.38	0.50
Feature Based	KNeighbors K=3	0	0.9853	0.99	0.99	0.99	0.7336	0.76	0.86	0.80	0.7935	0.87	0.87	0.87
		1		0.96	0.97	0.97		0.68	0.52	0.59		0.51	0.50	0.51
CNN	32,64,128	0	0.9815	1.00	0.98	0.99	0.8578	0.94	0.82	0.88	0.7886	0.88	0.84	0.86
		1		0.93	0.99	0.96		0.75	0.92	0.82		0.50	0.58	0.53

Table 22: Results of multivariate classification with the different models.