Machine Learning & Fundamental algorithms Project

Valentin RIEU, Ugo LABBE & Mickael GAULT

	CONTRIBUTION	
	Part 1 - **Practice**	Part 2 - **Go wild !**
Valentin RIEU	31.67%	31.67%
Ugo LABBE	31.67%	31.66%
Mickael GAULT	31.66%	31.67%
AI Tools	5%	5%
Total=	100%	100%

Table 1: Work repartition

Dataset

We selected a dataset describing the occupancy of a room[1] using multiple environmental sensors like temperature, light, sound, CO2 levels and Passive Infrared (PIR) sensors. There are 17 features (excluding the Date) and 10129 records. Each record is the reception of data from the sensors every 30 seconds. The features are data recovered from 7 sensors:

- 4 of them recovering the ambient temperature, light level and sound level,
- 1 of the sensors recovering the CO2 level and the variance within the time window of the record,
- 2 of the sensors recovering passive infrared (motion detection, binary value).

The label is the number of occupants in the room, ranging from 0 to 3.

As seen in Table 2, there is some class imbalance within the dataset between the different labels. It is more common to see no occupants in the room than 3 at the same time.

Occupants in the room	0	1	2	3
Occurrences	8228	459	748	694

Table 2: Class distribution

[1] `https://archive.ics.uci.edu/dataset/864/room+occupancy+estimation`

Data Visualization

Using Principal Component Analysis, we can see on Figure 1 that with the first two Principal Components (PCs) the explained variance is 63%. If we scatter the first two PCs like in Figure 2 we can deduce that the data is not linearly separable, thus requiring the use of Support Vector Machines in order to classify the data.

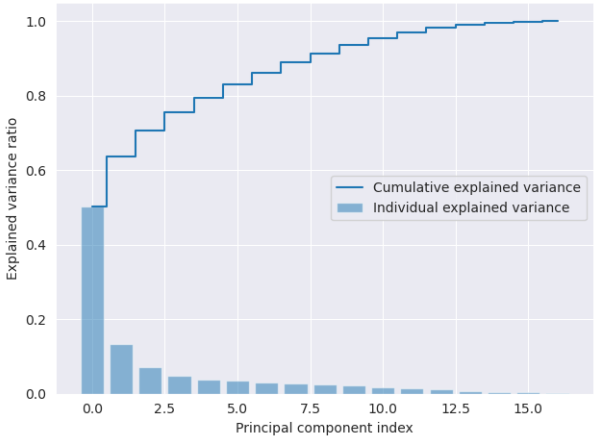


Figure 1: Explained Variance of the Principal Components.

Figure 2: Scatter of the first two Principal Components.

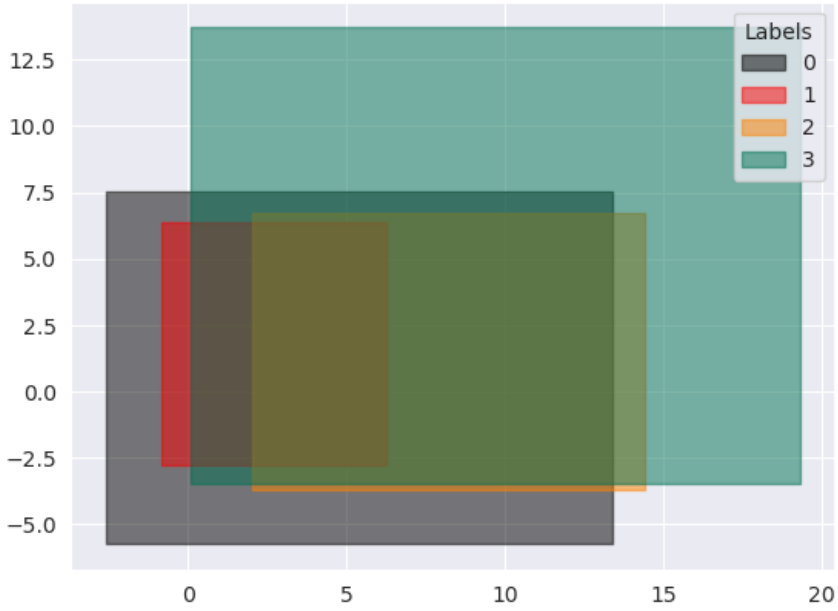


Figure 3: Region of each class point after standardization.
Overlaps between the region means that they share a similar area, i.e they are not linearly separable

Support Vector Machine

As the data is not linearly separable, to predict room occupancy we can use the Support Vector Machine (SVM) algorithm.

It works by mapping data to a high-dimensional feature space making possible categorization, even when the data are not linearly separable. Then it learns a separator, after what the data is transformed in order for the separator to be drawn into a hyperplane[2]

The advantages of the SVM algorithm are that it works well in a high dimensional plane. It is memory efficient as it only uses a subset of training points in the decision function, and it's also versatile as different hyper-parameter can be selected.

However, one of the disadvantage of this classification algorithm is that it can be prone to over-fitting if the wrong Kernel function is selected.[3]

Hyper-parameter tuning, training and results

Pre-processing

In order to have an efficient SVM it's important to ensure of the data quality. The first step is to only keep the relevant features, meaning removing the date in our case. The next one was to transform the time feature (having as format hh:mm:ss) into a correct numerical format: seconds. Fortunately, the dataset didn't have missing values, meaning no rows had to be removed.

The last step of the pre-processing was to scale the data. The Standard Scaler was chosen, making the data ranging from -1 to 1.

$$z = \frac{x - \mu}{\sigma}$$

Hyper-parameter and Training

The dataset was split into a train set of 80% and a test set of the remaining 20%

To tune the hyper-parameter of the algorithm a grid search method was used with a 5-fold cross validation to prevent over-fitting on the train set. It was done on the best regularization parameter, kernel and its coefficient in order to fit the best Support Vector Machine algorithm.

The hyper-parameters in Table 3 were tested. For the polynomial kernel, its degree was also fitted from 1 to 5.

Hyper-parameter	Values tested
Regularization parameter	0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000
Kernel	Linear, Polynomial, Radial Basis Function, Sigmoid
Kernel coefficient	0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000

Table 3: Grid Search hyper-parameters

[2] https://www.ibm.com/docs/en/spss-modeler/saas?topic=models-how-svm-works
[3] https://scikit-learn.org/stable/modules/svm.html

To keep the best hyper-parameter we used a weighted F_1 measure in order to get a fair model with regards to the class imbalance of our dataset.

The Polynomial kernel is used, its degree is 1 and its coefficient is $\frac{1}{n \times \sigma^2}$, n being the number of features. The regularization parameter is 100.

Results

The accuracy on the test dataset is of 0.994571 and its weighted F_1 measure is of 0.994572. We can understand from these score that our model is performing really well and fair with both the accuracy and the F_1 measure being really close to 1.

SVM on datasets with noise and missing values

To challenge the Support Vector Machine algorithm, datasets with missing values and noise were created, from 10 to 90% of missing values and noise from a normal distribution with $\mu = 0$ and $\sigma = 0.1$. We will be using the same hyper-parameters as the model based on the training set.

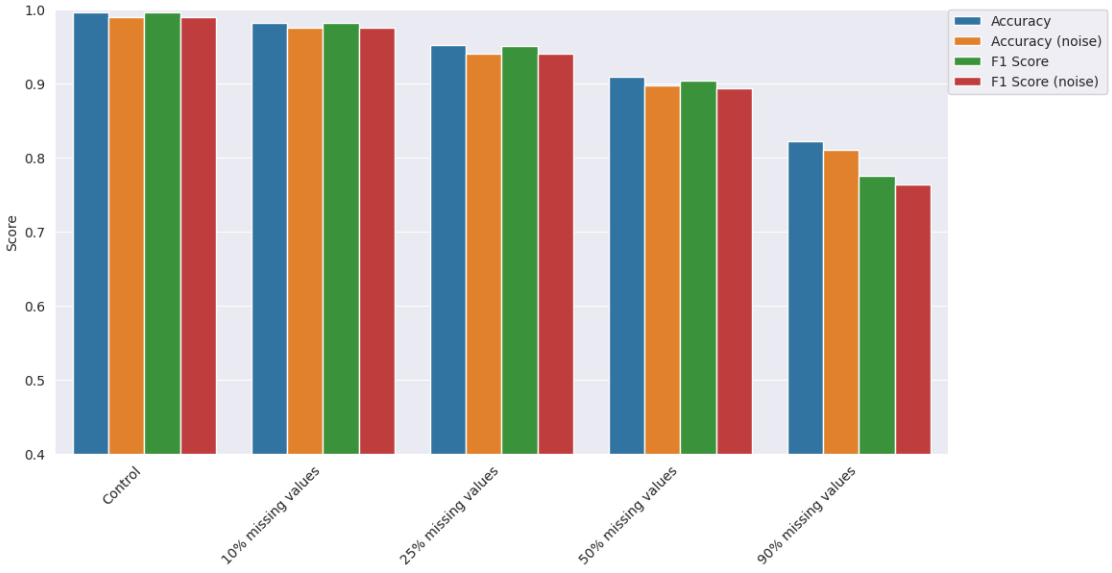
As SVM doesn't support missing values we replaced them using an Iterative Imputer that estimates each feature from all the others.[4]

Dataset	Accuracy	F_1 measure
10% missing values	0.982920	0.982831
25% missing values	0.953204	0.952422
50% missing values	0.908678	0.903238
90% missing values	0.829006	0.781520
10% missing values and noise	0.969395	0.969643
25% missing values and noise	0.941356	0.940919
50% missing values and noise	0.897917	0.893123
90% missing values and noise	0.802152	0.746215
Only noise	0.985092	0.985498

From these results, we can clearly see that missing values impact the accuracy and even more the F_1 measure, however it's only at 90% missing values that the model starts having clear worse result. At 50% the model still have a 90% accuracy and F_1 measure.

Noise also decrease the quality to predict the correct class as the F_1 measure drops much faster on the datasets with noise, its accuracy also drops by 1% or 2% more than on the non-noisy datasets

[4] `https://scikit-learn.org/stable/modules/generated/sklearn.impute.IterativeImputer.html`



Conclusion

Finally, the Support Vector Machine works well in most of the cases. It handles well missing values and noise in the datasets. Furthermore, in a short time (less than 10 min), we were able to find the best hyper-parameters to the model and get more than 99.99% of accuracy and 99.99% as f_1-measure despite the natural strong imbalance in favor of the class 0.