Anomaly Detection on the Hypothyroidism Dataset

Daniele Montagnani
Universita degli Studi di Milano-Bicocca
Id: 896621
Email: d.montagnani@campus.unimib.it

***Abstract*—This project report summarizes my analysis of the Hypothyroidism dataset. It discusses data pre-processing, modeling choices, the definition of a custom distance, a brief clustering analysis, and anomaly detection. It concludes with a brief reflection on unsupervised learning and data analysis.**

1. Introduction

In the following project I will study the hypothyroidism dataset. In particular I will: explore the dataset, develop a custom distance to treat data with different type of attributes, and perform an anomaly detection analysis.

2. Data Pre-processing

2.1. Dataset Exploration

As a first step in my analysis, I explored the given dataset in order to be able to decide on appropriate modelling choices. The hypothyroidism dataset contains 7200 observations across 21 variables. Of these, 15 are binary and 6 are continuous. Among observations there are no missing values.

In Figure 1, we can visualise how the dataset's variables are distributed. Notably, continuous variables are already scaled in the [0,1] range. Furthermore, they present high peaks near the mean / mode of the distribution, this suggests that there may have been some missing data which was subsequently imputed. All binary variables present an unbalanced distribution.

2.1.1. Modelling Choices. Given that binary variables present an unbalanced distribution, I treated some of them as sparse. To distinguish between sparse and categorical binary variables I set a "balance" threshold. Namely, if a variables does not have at least 20% of observations in every class I treated it as sparse. In the end, this criterion made it so that only the first binary variable was treated as categorical.

2.1.2. Pre-Processing. As some pre-processing steps were already performed on the dataset, I did not need to do many operations.

First, I obtained a one hot encoding of the first binary variable as I was going to treat it as categorical (with the

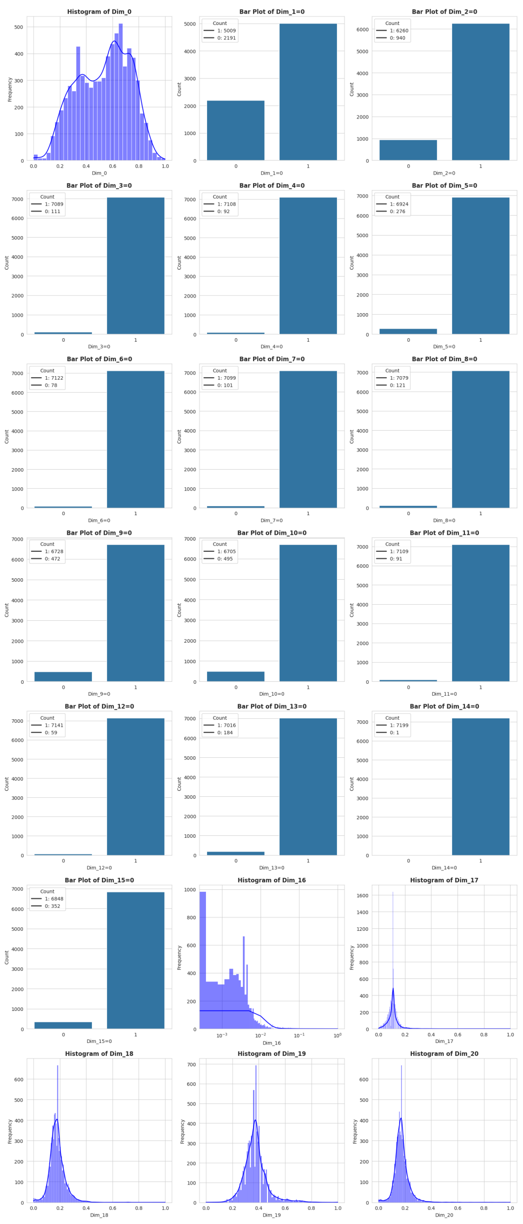

Figure 1. Visualisation of the variables' distribution

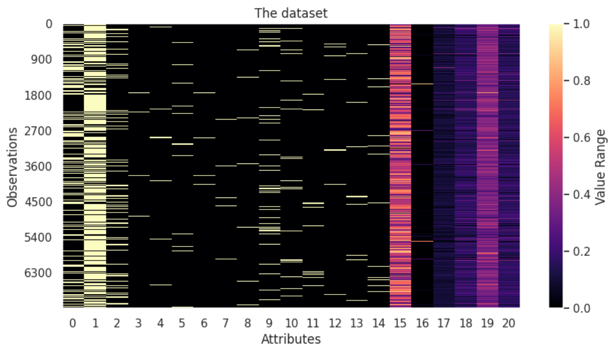

Figure 2. Heat-map of the dataset

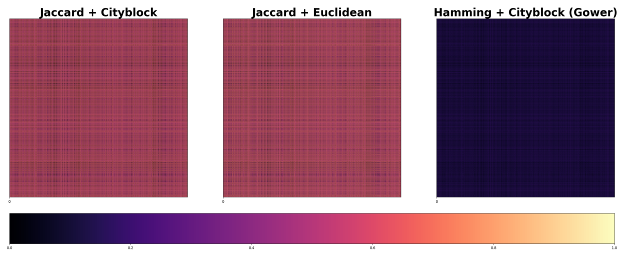

Figure 3. Comparison of composite distances

```
Jaccard + Cityblock
Mean: 0.44124299099318737
Standard Deviation: 0.2714828152660941
Min: 0.0
Max: 0.8712898421052632
25th, 50th, 75th Percentiles: [0.34613361 0.4763099  0.70427542]

Jaccard + Euclidean
Mean: 0.4539256086094625
Standard Deviation: 0.2716854661405964
Min: 0.0
Max: 0.9008808331453159
25th, 50th, 75th Percentiles: [0.34748397 0.48483049 0.71315514]

Hamming + Cityblock (Gower)
Mean: 0.10512740419391749
Standard Deviation: 0.0631685032724041
Min: 0.0
Max: 0.46613354887218045
25th, 50th, 75th Percentiles: [0.0579326  0.1100686  0.15371565]
```

Figure 4. Statistical summary of distances

Jaccard distance). Then, I flipped sparse variables so that the majority of observations had a zero value. Then again, I dropped the 14th binary observation as it had only 1 positive value. Lastly, I renamed and reordered the dataset variables to manage them more conveniently. In particular, I moved all binary variables in the first columns and I changed every variable name to reflect its type.

With the new order, the dataset can be visualised in Figure 2 as an heat-map.

3. Distance Computation

Computing the distance between data-points was not straightforward as the dataset contains mixed data. Looking at the literature, I found, in [1], the Gower distance / similarity.

Essentially, the Gower similarity is a weighted average of rescaled, continuous and binary, similarity coefficients. Usually, the continuous distance is the cityblock distance, and the binary/categorical distance is derived from the simple matching coefficients. However, since I'm treating most binary variables as sparse, I could not use the simple matching coefficient as a similarity measure. Instead, I opted for the Jaccard coefficient.

In summary, to implement my custom distance, I adapted Gower's distance to account for sparse data. In fact, I took a weighted average between a rescaled continuous distance (euclidean or cityblock) and the Jaccard distance between binary vectors. Notably, applying the Jaccard coefficient on my, one-hot encoded, categorical variable is equivalent to using the simple matching coefficient. However, to correct for double counting, the two perfectly correlated encodings have to weight for only one variable.

With the custom distance in place, I tested it on the dataset to choose between the cityblock and the euclidean continuous distance. Furthermore, I took the chance to compare its performances with those of the classic Gower distance. In Figure 3 we can see these comparisons.

Furthermore, in table 4 we can see a statistical summary of the computed distance for every approach.

Clearly, the custom distance has a superior performance with respect to the classic Gower distance on a sparse dataset. The distances' distribution has a much higher standard deviation and therefore the distance becomes more informative. We could say that it is a way to lessen the effect of "the curse of dimensionality".

On the other hand, we can see that the choice between cityblock and euclidean is not so relevant. Therefore, to leave my measure more similar to Gower distance, I will use the cityblock distance as my continuous component.

Notably, the custom distance will be used in all subsequent stages. The only exception is the elbow method to find the number of clusters in DBSCAN.

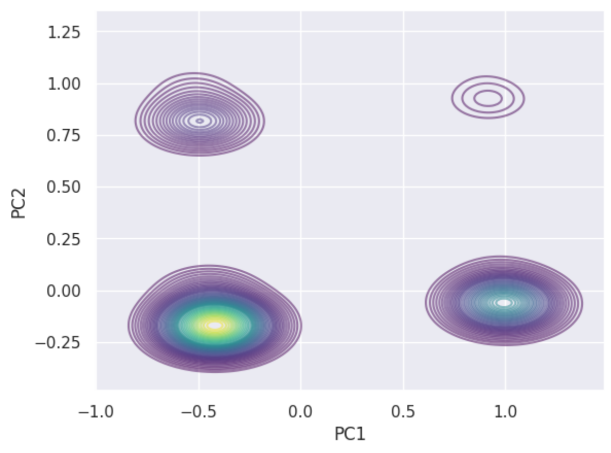

Figure 5. PCA, density estimation

4. Clustering and PCA

After having decided how to compute distances, I decided to visualise the dataset and to perform a clustering analysis. These things were done both to understand the dataset better and to prepare for some anomaly detection methods.

4.1. PCA

As a first step, I performed a PCA decomposition and plotted my data in the space spanned by the first two principal components. In figure 5 we can see the plot with density estimation. I see three natural clusters and a smaller cluster of, likely, outliers.

4.2. Clustering

4.2.1. DBSCAN. With an idea of the probable cluster structure of the dataset, I went on and performed a DBSCAN clustering analysis. To find the number of clusters, I used the elbow method as depicted in figure 6.

Then, knowing the number of clusters I applied the DBSCAN approach and found my clusters as they are plotted in Figure 7. Notably, the DBSCAN algorithm was very aggressive in finding outliers with 1674 eliminations. For the clustering found with DBSCAN, I obtain a silhouette coefficient of 0.6429 .

4.2.2. Hierarchical. As a second clustering approach, I chose to use the hierarchical method. Since the clusters seem well separated, I used the single linkage method. As we can see in figure 8, I plot the full dendogram and cut it at the optimal number of cluster by visual inspection.

Conveniently, even in this case I found 4 clusters. In figure 9 we can visualise the clusters found with the hierarchical approach. Notably, cluster 2 is very isolated and contains only 530 points. Furthermore, all of its points are flagged as outliers by DBSCAN. For these reasons, I decided

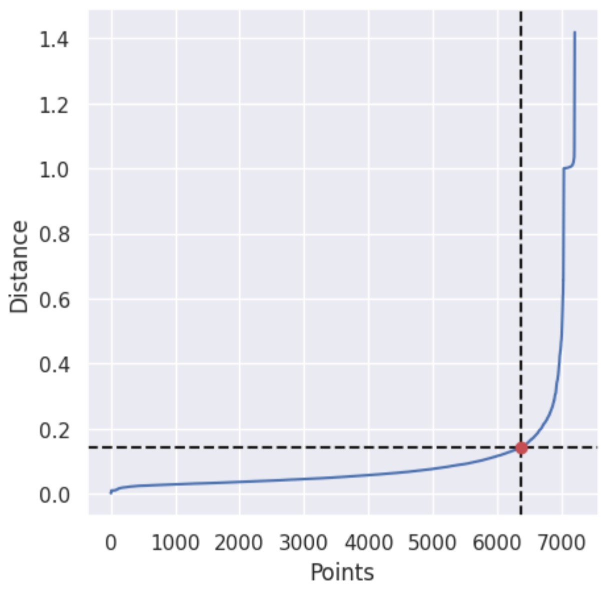

Figure 6. Enter Caption

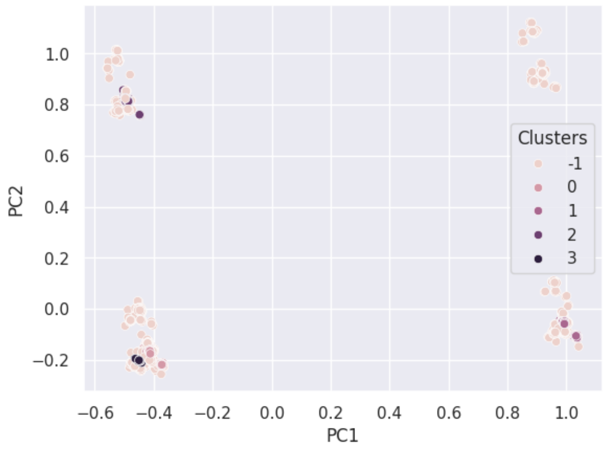

Figure 7. DBSCAN clusters, outliers denoted as -1

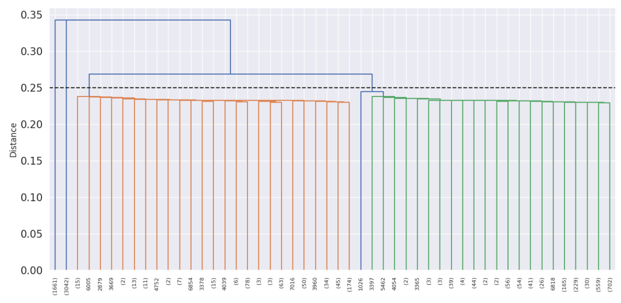

Figure 8. Dendogram

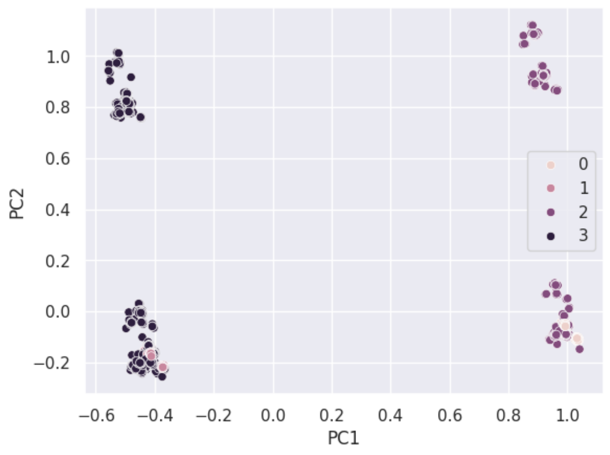

Figure 9. Hierarchical clusters

to consider its elements as outliers. With the Hierarchical clustering approach I obtained a silhouette score of 0.5734.

5. Anomaly Detection

Coming to the hearth of the project, for the anomaly detection task I decided to experiment with a variety of methods. I used 8 methods divided in 4 families. In the follwoing paragraphs I will mention them and give a brief explanation.

The first family comprises clustering based methods and uses the results of the previous section. The idea behind clustering based methods is to treat those observations that the algorithm can't cluster efficiently as outliers. Alternatively, if one finds very small clusters, all of their elements can be considered as outliers. Some algorithms, such as DBSCAN readily exclude some data points from the resulting cluster effectively flagging them as outliers. Other algorithms, such as hierarchical clustering, form comprehensive clusters, it is the analyst duty to decide which of these clusters actually represent anomalies.

The second family uses reconstruction based approaches. I chose PCA and (moderately deep) autoencoders. The rationale behind reconstruction based approaches is to force a model to learn a reduced representation of a datapoint. The reduced representation constitutes a bottleneck on the amount of information that the model can allocate to represent data. Clearly, for a model it is wiser to invest such information to represent "normal" datapoints as they will occur more often. This will make it so that models tend to struggle more when reconstruction non normal observations. For this reason, the higher the error made by the model, the higher the ratinale in considering the dataoint an outlier.

The third family uses "isolation based" methods including one class support vector machines and isolation forest. Similarly to the previous example, we get a measure of the exceptionality of a datapoint by looking at how much the model struggles to learn it. Ocsvm fit a support vector machine on a single class of normal observations discarding a proportion (determined by an hyperparameter) of outliers. Isolation forest looks at the depth at which, on average, an observation was split by a decision tree. Naturally outliers are easier to discard as they are markedly different.

The fourth family considers model free proximity based approaches: KNN and LOF. It is a model based approach in the sense that we can compute the anomaly score of an outlier simply by looking at his relation with other entities in its neighborhood. The most important hyperparameter in these approaches is the numebr of neighbors to include in a neighborhoos.

For every method, I performed an hyperparameter sweep, trying to run it with some sensible hyperparameter configurations. Then I plotted an histogram of the anomaly scores produced by every hyperparameter. We can see a couple of examples in figures 10 and 11.

Lastly, by visual inspection as explained on the course slides, I chose the well behaved ones and I saved the results in a numpy tensor for the following stage. Every anomaly score produced was normalized to lay in the range [0,1]. Some scores, for which a negative value represented normality, had their negative values raised to zero.

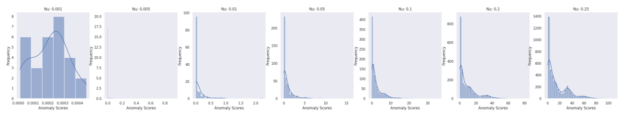

Figure 10. One class support vector machine - anomaly scores histogram

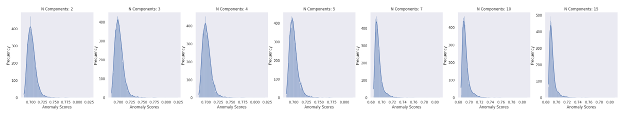

Figure 11. PCA reconstruction - anomaly scores histogram

6. Final Decision

In the final stage of my analysis, I had to combine anomaly scores produced by various methods in order to obtain a final decision on which points to consider outliers.

I decided to aggregate the scores in stages. First, for every method, I computed the average outlier score by aggregating all results produced by well-behaved hyperparameter configurations. Then, for every family, I aggregated its methods' predictions by taking a weighted average with weights reflecting the expected quality of the model (as judged by visual inspection of the histogram). Lastly, I aggregated family's scores by taking again a weighted average reflecting expected performance.

This approach produced a vector of aggregated anomaly scores. In figure 12, we can see the histogram representation. In figure 13, we can see the time series.

To identify potential outliers, I employed a threshold-based approach. Namely, from the histogram, I chose a significant percentile and considered every point beyond it as an outlier. As significant percentiles I tried both the 95th and the 93rd.

In figure 14, I visualised and compared the resulting outliers. Ultimately, I chose the 93rd percentile to classify all points in the right upper cluster as outliers.

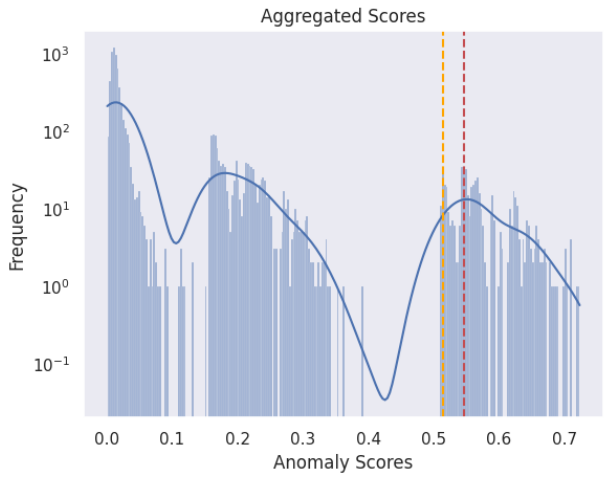

Figure 12. Aggregated scores - histogram

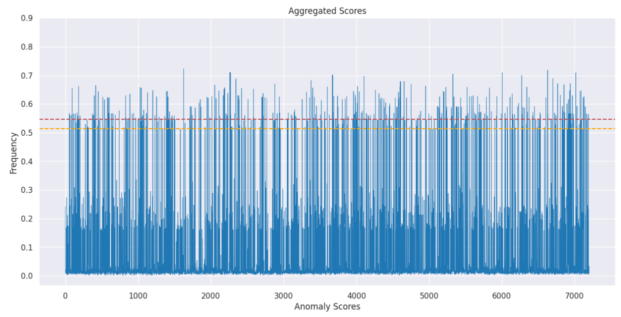

Figure 13. Aggregated scores - time series

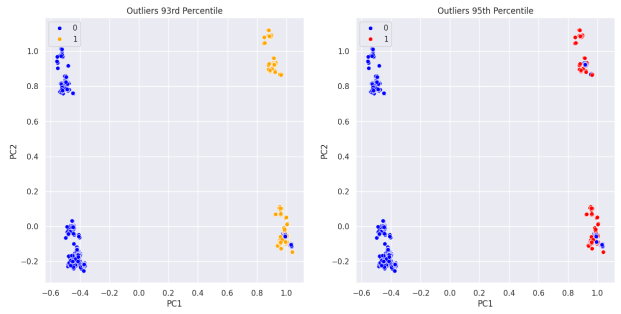

Figure 14. Outlier Comparison

Finally, I had to compute the probability that a given observation is an outlier. I already had my aggregated scores lying in [0,1] but I wasn't happy with the result. High scores were too low, and scores in the low-medium range were too high. I believe these effects were artifacts of the aggregation approach. So, to remedy and increase the confidence of my scores, I applied a distortion by squaring scores in the medium to low range, and by taking the square root of high scores. This gave me the probabilities depicted in figure 15.

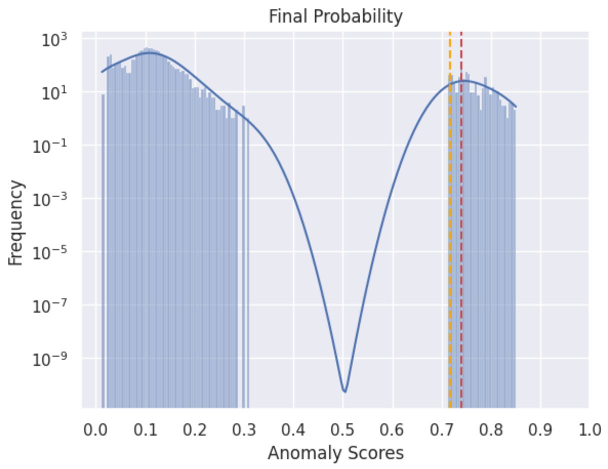

Figure 15. Final probabilities

7. Conclusions

In this project I performed a set of tasks aimed at conducting an anomaly detection analysis. From distance computation, to score aggregation, many times I had to make seemingly arbitrary choices. In the end, I chose to try different methods to probe their strengths and weaknesses. At the final stage, I combined their predictions with custom weights reflective of the expected performance of every method.

8. Disclosure

I confirm that this report is entirely original and does not contain any form of plagiarism. I did not make us of ChatGPT or any other Natural Language Processing models.

9. Take Home Message

Data analysis leaves many degrees of freedom in the hands of the analyst. In unsupervised learning this problem is exacerbated by the absence of a ground truth. These conditions make it likely that the analyst's convictions and preference influence the result. Fortunately, this can be seen both as a bug and as a feature of statistics. With subjectivity we open the door for intuition, first principles reasoning and creative thinking. And, at the end of the day, this is not something new to the foundations of statistics, rooted in the theory of decisions. As it is written in [3], one of my favorite statistics book, "there are two types of statisticians: those who know what decision problem they are solving and those who don't."

References

[1] J. C. Gower, *A General Coefficient of Similarity and Some of its Properties*, Biometrics, vol. 27, no. 4, pp. 857–871, 1971.

[2] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*, 2nd ed. [https://www.pearson.com/us/higher-education/program/Tan-Introduction-to-Data-Mining-2nd-Edition/PGM330948.html].

[3] G. Parmigiani and L. Inoue, *Decision Theory: Principles and Approaches* Chichester, England: John Wiley Sons, 2009.

[4] *scipy*: scientific computing in python [https://pypi.org/project/scipy/]

[5] *scikit-learn*: machine learning in python [https://scikit-learn.org/]

[6] *PyTorch*: An open-source machine learning framework [https://pytorch.org/].

[7] Lab Sessions: I have adapted some code and some approaches from the lab sessions.

[8] Course Slides: Apart from textbooks, I've used the slides of the course to choose and understand appropriate approaches.