Unsupervised Learning - Anomaly Detection on the Hypothyroidism Dataset

Simone Vaccari
Artificial Intelligence for Science and Technology
Università degli Studi di Milano-Bicocca
Id. 915222
s.vaccari10@campus.unimib.it

Davide Vettore
Artificial Intelligence for Science and Technology
Università degli Studi di Milano-Bicocca
Id. 868855
d.vettore@campus.unimib.it

***Abstract*— This study applies unsupervised anomaly detection methods to a dataset of 7,200 patient records related to hypothyroidism. By employing algorithms such as K-Nearest Neighbors, Local Outlier Factor, Density-Based Spatial Clustering of Applications with Noise, Principal Components Analysis, and One-Class SVM we aim to identify anomalies in medical data characterized by diverse attribute types. To address the limitations of individual methods, we adopt a majority vote approach, enhancing the robustness and accuracy of anomaly detection.**

***Index terms*—Hypothyroidism Dataset, Anomaly Detection, KNN, LOF, DBSCAN, PCA, One-Class SVM**

I. Introduction

Hypothyroidism, also known as underactive thyroid, is a condition that occurs when the thyroid gland produces much less thyroid hormones than the body needs, affecting its metabolism. The main symptoms include: weight gain, slowed heart rate, more sensitivity to cold, dry skin, fatigue, muscle weakness, and depression.

The most common causes of hypothyroidism involve:

- Hashimoto's disease, an autoimmune disorder;
- Inflammation of the thyroid gland;
- Surgical removal of part or all of the thyroid;

The occurrence of the disease is around 4.7% in Europe and increases to 5-6% in the USA, with a higher prevalence among older patients and women. [1]

Despite advancements in healthcare technology, medical errors continue to pose a significant challenge that can be addressed using Machine Learning and Artificial Intelligence. In particular, anomaly detection, the process of identifying data instances that are significantly different from the rest, can help recognize unusual health conditions or patient activities. This allows for early detection of diseases, which can result in improved patient outcomes. Furthermore, it can highlight inconsistencies or errors in medical records and treatment indications, giving healthcare providers the opportunity to fix these mistakes before they impact patient care. [2]

II. Dataset description

The Hypothyroidism dataset includes 7,200 data objects, each described by 21 attributes. Out of these, 15 are binary, and the other 6 are continuous.

An initial look at the data shows that none of the features have missing values. However, there are 71 duplicate observations. Most duplicates appear twice, a few appear three times, and only two observations are repeated four and five times. Since our dataset lacks an identification feature, we decide to keep the duplicates, assuming they represent equal measures rather than errors and likely belong to dense clusters.

The `Pandas` library's *describe* method [3], as shown in Table 1, provides important statistics about the distribution of our continuous features and helps us gain some insights.

	Dim0	**Dim16**	**Dim17**	**Dim18**	**Dim19**	**Dim20**
mean	0.532	0.009	0.108	0.180	0.374	0.174
std	0.197	0.043	0.042	0.060	0.088	0.056
min	0.000	0.000	0.000	0.000	0.000	0.000
25%	0.375	0.001	0.092	0.146	0.324	0.145
50%	0.563	0.003	0.109	0.176	0.370	0.170
75%	0.688	0.005	0.120	0.206	0.403	0.195
max	1.000	1.000	1.000	1.000	1.000	1.000

Table 1: Interesting statistics regarding the distribution of the continuous features obtained with the *describe* method.

We can observe that all continuous features are already normalized between 0 and 1. However, most of them are highly right-skewed. This skewness is evident from the difference between the mean and median of each feature and the significant increase in value in the last quartile. We can investigate further by examining the distribution of each continuous feature and the corresponding boxplot.

The distributions of the continuous features, visualized in Figure 1 using the `Matplotlib` [4] and `Seaborn`[5] libraries, once again illustrate how most of them exhibit highly skewed distributions, with long tails on the right side.

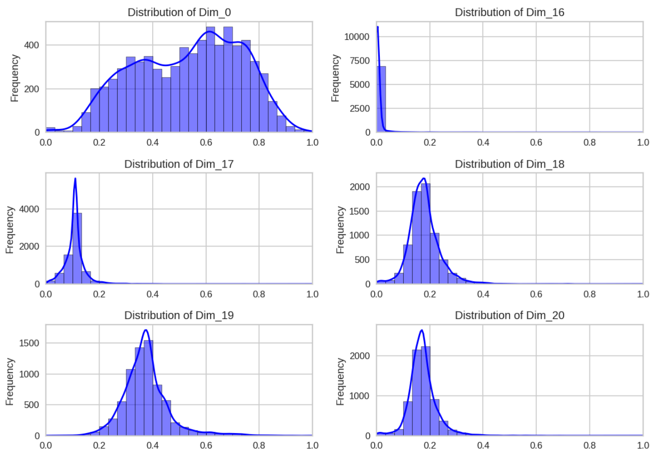

Figure 1: Distributions of continuous features.

In Figure 2, the boxes represent the quartiles of the features, while the whiskers extend to show the rest of the distribution, except for points identified as outliers based on a method that considers the inter-quartile range. This visualization highlights several outliers in all the continuous features except for *Dim_0*. In our preprocessing pipeline, we certainly don't want to remove these outliers, common practice in most machine learning tasks, as we are interested in anomaly detection.

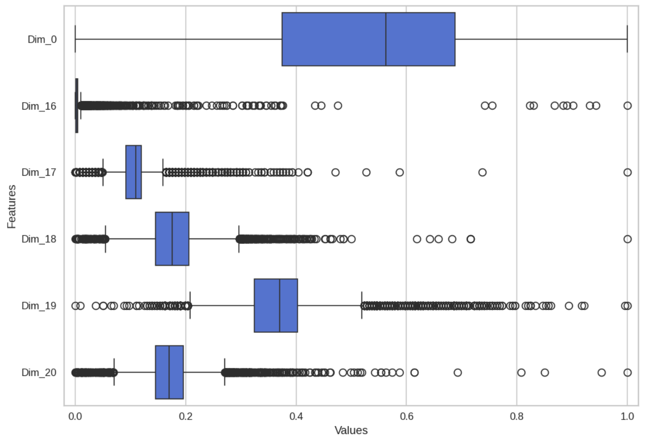

Figure 2: Boxplots of continuous features.

For our final analysis of the dataset, in Figure 3, we plot how the values of continuous features vary in 40 subsequent random observations. Once again, *Dim_0* appears to be evenly distributed, while the other features, especially *Dim_16*, seem to oscillate in the lower end of the normalized interval. From the plot, we can observe a data point whose values deviate notably from the global behavior in three out of the six continuous features. Although we are only observing a small sample of observations and a subset of features, we can already hypothesize that we are looking at an anomalous data point.

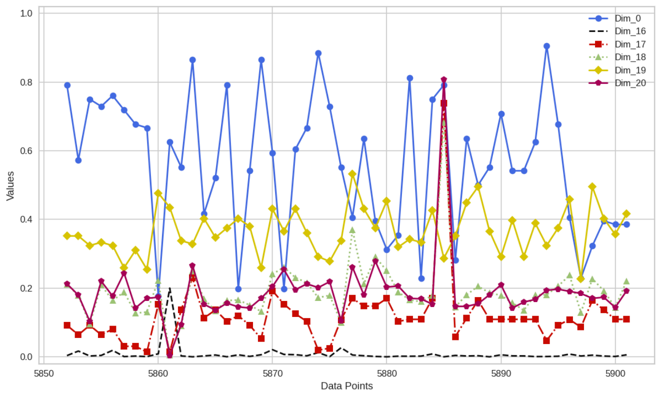

Figure 3: Values of continuous features in 40 subsequent random observations.

A. *Preprocessing Pipeline*

As mentioned earlier, the continuous features seem to have already been normalized. Another important preprocessing step involves converting the features into appropriate data types for the next steps. The continuous features, previously encoded as objects, are converted into float values, while the binary features, previously encoded as integers, are converted into boolean data types.

III. Distance measure

Some of the algorithms we apply are based on the concept of proximity, thus requiring the definition of a proximity or similarity measure. The key idea behind this is to calculate the distance between data points to identify which observations are more separated from the rest, and tag them as outliers.

Unfortunately, traditional proximity measures like Euclidean or Minkowski distance only work with numerical data. Therefore, since our dataset contains both numerical and categorical attributes, we need to define an appropriate alternative similarity measure. In this case, we decide to use Gower's distance, defined in [6], which uses both Manhattan distance for continuous variables and Dice distance for binary attributes. More specifically, the Gower's distance between two data points $\boldsymbol{x}$ and $\boldsymbol{y}$ with M attributes is defined as:

$$S(\boldsymbol{x}, \boldsymbol{y}) = \frac{\sum_{k=1}^{M} \delta_k s_k(\boldsymbol{x}, \boldsymbol{y})}{\sum_{k=1}^{M} \delta_k}$$

where $s_k(\boldsymbol{x}, \boldsymbol{y})$ is the similarity for the k-th attribute, computed using Manhattan or Dice distance depending on that attribute's type, and δ_k is a quantity equal to 1 when attribute k can be compared for $\boldsymbol{x}$ and $\boldsymbol{y}$, and 0 otherwise.

IV. Anomaly detection algorithms

In this chapter, we delve into different algorithms for anomaly detection. Building on the introduction, we explain how these algorithms work and we try to evaluate how well they find anomalies in our dataset. Each algorithm is based on specific assumptions that affect how they operate and interpret the data, and can be categorized as one of the following:

- Proximity-based algorithm;
- Clustering-based algorithm;
- Reconstruction-based algorithm;
- Other algorithms;

*A. **KNN***

The K-Nearest Neighbors (KNN) algorithm belongs to the proximity-based approaches for anomaly detection. This approach starts by computing a neighborhood for each data point and aims to identify anomalies by analyzing these neighborhoods. Proximity-based approaches include both distance-based methods, like KNN, and density-based methods, like LOF.

Key assumption: Normal data objects have close neighbors, while anomalies are data objects most distant from other data objects.

In practice, the neighborhood of each data object is computed using the scikit-learn [7] `NearestNeighbors` model fitted on the precomputed Gower distance matrix. The parameter K signifies the number of nearest neighbors to consider, and its choice is crucial as it impacts the effectiveness of anomaly detection. Ideally, in a classification task, K can be optimized by evaluating how it influences model performance or can be chosen based on some prior knowledge of the domain. However, since we don't have any ground truth about the anomaly labels or prior knowledge of the dataset, a common rule of thumb suggests setting K to the square root of the total number of data points, i.e. $\sqrt{7200} \approx 84$.

Once the neighborhood of each data object has been calculated, based on our main assumption, the anomaly score of each data point is determined by measuring the distance to the furthest data object in the neighborhood. Alternatively, we could have calculated the anomaly score by considering the average or even the median distance between each object and its neighborhood.

The next step is to determine how many observations are anomalies. To achieve this, we can plot the anomaly scores in ascending order and adopt one of the following methods to establish a threshold:

- Position the threshold where the knee point is identified using the `KneeLocator` library [8].
- Set a predetermined percentage of outliers beforehand, for example, 5%.
- Draw a threshold manually by examining the plot of sorted anomaly scores.

In Figure 4, the sorted scores and the automatically computed thresholds are shown, along with a histogram of the frequencies at which the anomaly scores occur.

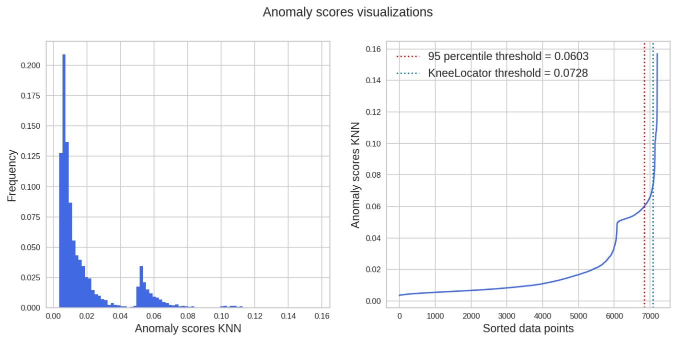

Figure 4: Frequencies and sorted KNN anomaly scores.

We decided to consider an observation an anomaly if its anomaly score falls in the top 5%. The choice of this 'contamination' parameter, along with the choice of K, is crucial for the success of the anomaly detection task. This sensitivity to parameter selection represents one of the main weaknesses of proximity-based approaches.

In Figure 5, we visualize the obtained anomalies using a comprehensive set of four scatter plots. The two plots on top show different pairs of features: Dim_0 with Dim_16 and Dim_17 with Dim_19, respectively. On the bottom, we represent the dataset with two different dimensionality reduction techniques:

- The T-distributed Stochastic Neighbor Embedding (t-SNE) plot reduces our high-dimensional data to two dimensions, making it easier to identify distinct groups and potential anomalies.
- The Principal Component Analysis (PCA) plot also reduces our data to two dimensions but aims to capture the maximum variance through the first two principal

components. In our dataset, these two principal components explain 46.75% of the total variance.

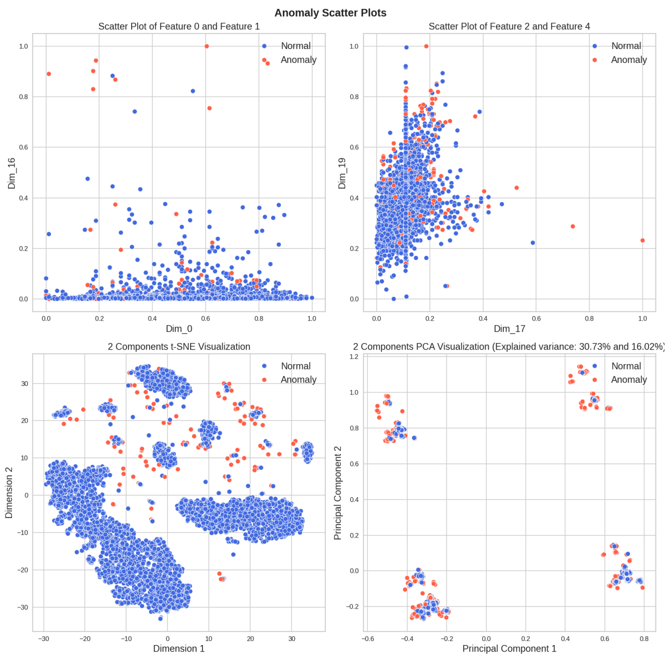

Figure 5: KNN anomaly detection visualized with scatter plots and dimensionality reduction techniques.

Continuing with dimensionality reduction, anomalies can be visualized on a 3D scatter plot using a 3-component PCA. The additional component explains an extra 9.16% of the total variance. In Figure 6, a static visualization is presented; however, using the `Plotly` library [9], interactive exploration of the dataset and the detected anomalies in 3D space is also possible.

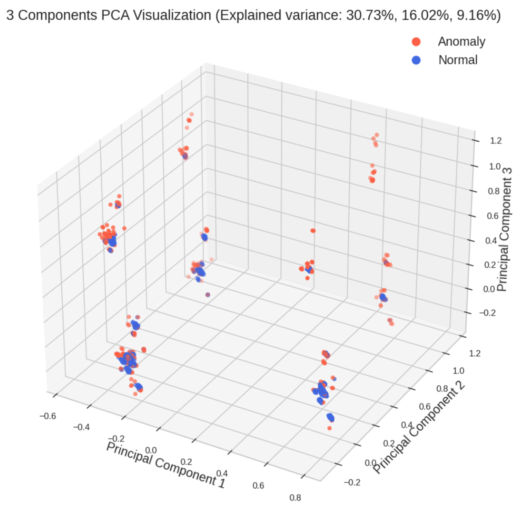

Figure 6: KNN anomaly detection scatter plot on 3-component PCA.

B. LOF

As mentioned before, the Local Outlier Factor (LOF) algorithm is part of the family of density-based anomaly detection techniques.

Key assumption: Normal data objects have close neighbors, while anomalies are data objects in low density regions.

In particular, LOF considers as outliers the samples that have a substantially lower density than their neighbors.

In order to compute the LOF score for every sample q we proceed according to the following steps [10]. First, we compute the distance d_k to q's k-th nearest neighbor. As in the case of KNN, we set $k = \sqrt{7200} \approx 84$. Then, we compute the *reachability distance* with the selected neighbors, defined as follows:

$$reachdist(p, q) = \max\{d_k, d(p, q)\} \quad \forall p \in N_k(q)$$

Subsequently, we calculate the *local reachability density* of data point q as the inverse of the average reachability distance based on the k nearest neighbors of q:

$$lrd(q) = \frac{k}{\sum_{p \in N_k(q)} reachdist_k(p,q)}$$

Finally, the LOF score is computed by comparing the lrd of the selected record with the average lrd of its k-neighbors:

$$LOF(q) = \frac{1}{k} \sum_{p \in N_k(q)} \frac{lrd(p)}{lrd(q)}$$

This process is carried out for every data object, resulting in an array of anomaly scores. In practice, we manually define a function that, given the Gower distance matrix and the chosen number of neighbors as inputs, performs all the previously described steps for all points at once, optimizing the calculations, and returns the LOF scores.

In general, points with a score higher than 1 are considered outliers, and the higher the score, the more likely the sample is to be an outlier. In our case, we proceed as before to visualize the elbow curve (Figure 7) and determine a threshold, once again choosing the data points with top 5% LOF scores as outliers.

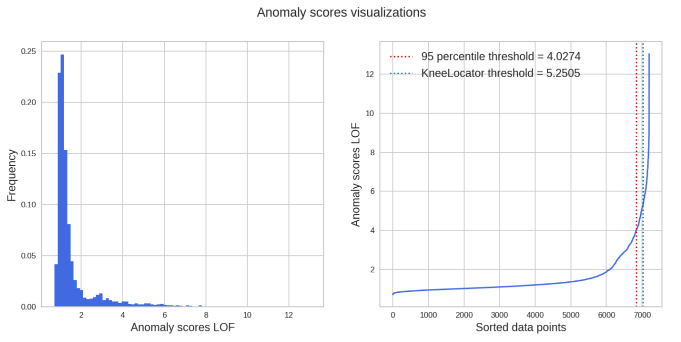

Figure 7: Frequencies and sorted LOF anomaly scores.

Once the 360 anomalies have been selected, we reproduce in Figure 8 the precedent scatter plots in order to have a visual comparison of the anomalies detected by the different algorithms.

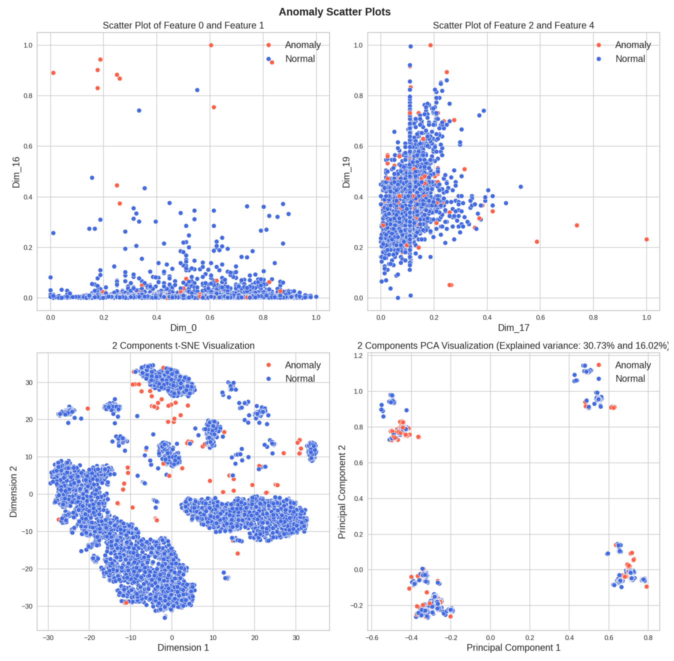

Figure 8: LOF anomaly detection visualized with scatter plots and dimensionality reduction techniques.

C. *DBSCAN*

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a clustering-based algorithm which finds core samples of high density and expands clusters from them. This works well for data which contains clusters of similar density.

Key assumption: Normal data instances belong to large and dense clusters, while anomalies do not belong to any significant cluster.

DBSCAN uses two main parameters:

- *min_samples* is the minimum number of points needed to form a cluster.
- *eps* is the maximum distance between points to consider them neighbors.

A core sample has at least *min_samples* neighbors within *eps* distance, indicating a dense region. Clusters are formed by connecting core samples and their neighbors. Samples far from any core sample are considered anomalies. Choosing the right *eps* is crucial: too small and most data isn't clustered, too large and clusters merge into one. Some heuristics for choosing this parameter involve looking for a knee in a nearest neighbor distances plot, which led us to set it to 0.05. Given the noise and sparse nature of our dataset, we set *min_samples* to $\sqrt{7200} \approx 84$ in order to look for denser clusters.

Unlike other clustering-based algorithms, such as k-means, scikit-learn's `DBSCAN` [11] can be fitted on a precomputed distance matrix. In our dataset, this algorithm detects 566 anomalies, which can be visualized in Figure 9.

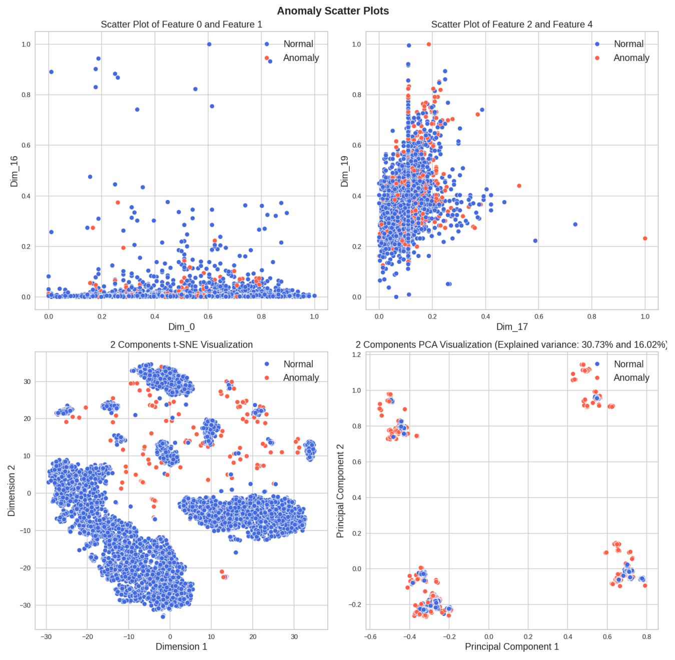

Figure 9: DBSCAN anomaly detection visualized with scatter plots and dimensionality reduction techniques.

D. *PCA*

Principal Component Analysis (PCA) is commonly used to reduce the dimensionality of data for easier exploration and analysis. We have already applied PCA to visualize our dataset in both 2D and 3D plots. However, the underlying theory of PCA also makes it suitable for using as a reconstruction-based anomaly detection technique.

Key assumption: There are patterns in the distribution of the normal class that can be captured using lower-dimensional representations. By comparing the original data with its reconstructed reduced-dimensional version, anomalies can be detected through significant differences.

The initial phase involves reducing the data to a lower number of dimensions. With scikit-learn's `PCA`, we project each data object into 7 dimensions; these principal components explain 81.6% of the total variation in the original data. The transformed data is then projected back to its original space, and the reconstruction error for each data point is calculated as the sum of the squared differences between the original and reconstructed data points:

$$RE_i = \sum_{j=1}^{d} \left(x_{ij} - \hat{x}_{ij}\right)^2$$

Anomalies can be detected by setting a threshold on the reconstruction errors, similar to other anomaly scores. Another approach involves considering an observation anomalous if the sum of squared projections onto principal components exceeds the critical value of the Chi-square distribution at a chosen significance level [12].

In Figure 10, the sorted reconstruction errors are reported. This time, we decided to manually choose a threshold at the first elbow point, which is not detected by the `KneeLocator's` default sensitivity parameter.

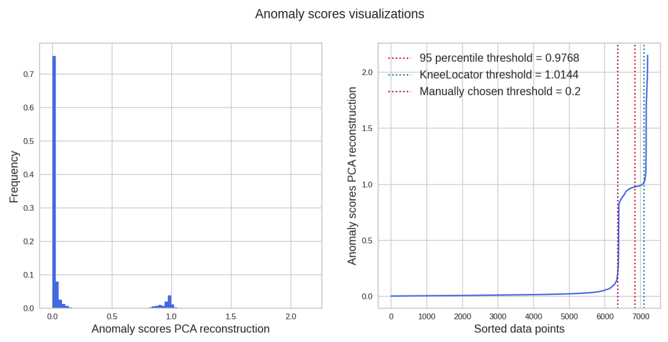

Figure 10: Frequencies and sorted reconstruction errors.

The more impactful, manually chosen threshold led to identifying 839 anomalies. These anomalies can be visualized in Figure 11.

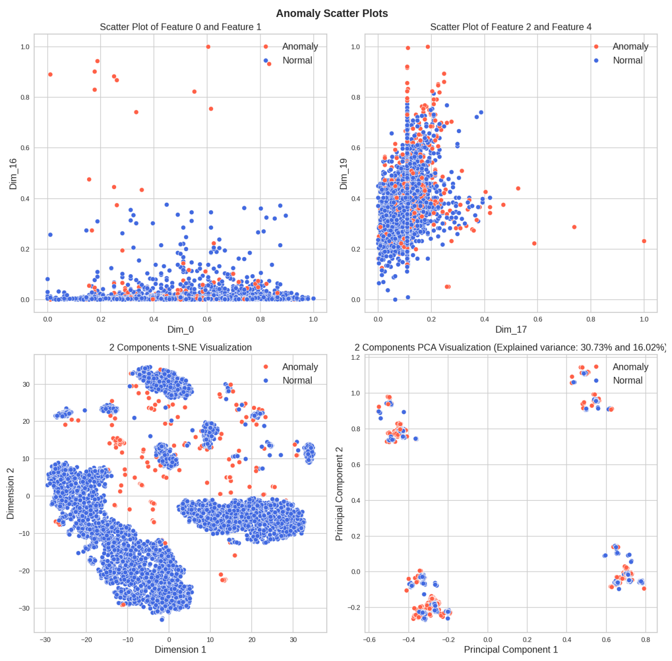

Figure 11: PCA anomaly detection visualized with scatter plots and dimensionality reduction techniques.

E. *One-Class SVM*

The last algorithm we apply does not belong to any of the categories previously mentioned, but it's inspired by the Support Vector Machines algorithms.

Key assumption: The boundary of normal data objects can be learnt using a support vector machines approach, by projecting the data to a higher dimensional space where anomalies and normal points can be linearly separated.

The data points are initially mapped to a higher dimensional space using a function Φ, in such a way that in this space the normal class can be isolated with a linear hyperplane of equation:

$$\langle \boldsymbol{w}, \Phi(\boldsymbol{x}) \rangle = b$$

In order to have all normal instances on one side, we need to impose:

$$\langle \boldsymbol{w}, \Phi(\boldsymbol{x}) \rangle > b \quad \text{if } \boldsymbol{x} \in \text{normal class}$$

$$\langle \boldsymbol{w}, \Phi(\boldsymbol{x}) \rangle < b \quad \text{if } \boldsymbol{x} \in \text{anomaly class}$$

By using a Gaussian kernel, for example, it is possible to map each point to a unit hypersphere. Then, according to the so-called origin trick, OC-SVM tries to maximize the distance of the hyperplane from the origin $\left(\frac{b}{\|\boldsymbol{w}\|}\right)$ while minimizing the distance of points lying on the "anomaly side" of the hyperplane, with the introduction of the slack variables

ξ_i. Ultimately, it all boils down to solving the following optimization problem, as indicated in [13]:

$$\begin{aligned} &\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \tfrac{1}{2}\ \|\boldsymbol{w}\|^2 - b + \tfrac{1}{n\nu} \textstyle\sum_{i=1}^{n} \xi_i \\ &\langle \boldsymbol{w}, \Phi(\boldsymbol{x})\rangle \geq b - \xi_i \qquad && i = 1, ..., n \\ &\xi_i \geq 0 && i = 1, ..., n \end{aligned}$$

where n is the number of observations, and ν is a crucial hyperparameter that controls the proportion of outliers allowed. It sets an upper bound on the fraction of training errors and a lower bound on the fraction of support vectors, and typically ranges between 0 and 1, where lower values imply a stricter margin and may capture fewer outliers, while higher values are more permissive.

The algorithm is very sensitive to the choice of this parameter, which results quite difficult, but at the same time, thanks to kernel functions, it can capture complex non-linear relationships between features, making it suitable for detecting anomalies in high-dimensional datasets such as ours.

By using the `OneClassSVM` class of scikit-learn with a value of ν set to 0.05 we detect 359 anomalies shown in Figure 12.

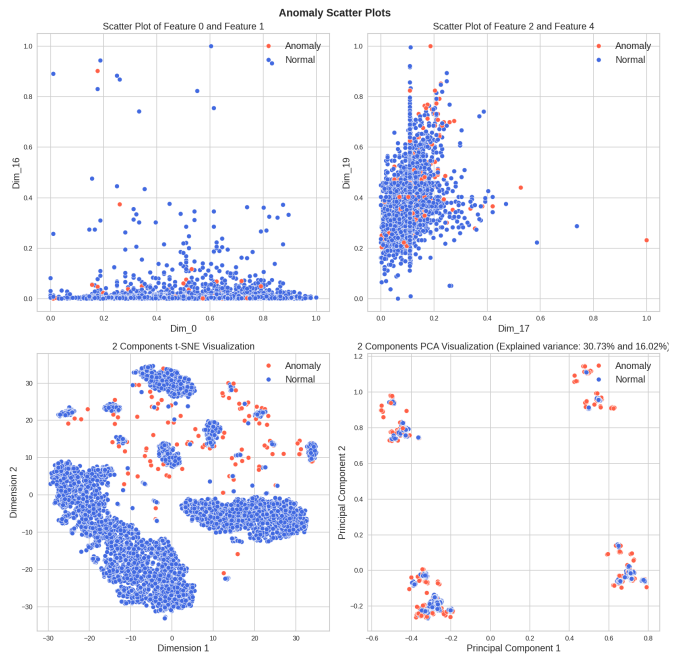

Figure 12: OC-SVM anomaly detection visualized with scatter plots and dimensionality reduction techniques.

V. Majority Vote

Since we don't have ground truth labels for our data, we can consider all the algorithms discussed above and combine their results using a majority vote approach. We count how many algorithms flag each observation as an anomaly; if at least three out of the five algorithms agree, the observation is labeled as an anomaly. The variation in the number of detected anomalies when changing the threshold level of agreement can be seen in Figure 13. This method helps balance the weaknesses of individual models and gives a more reliable way to identify anomalies.

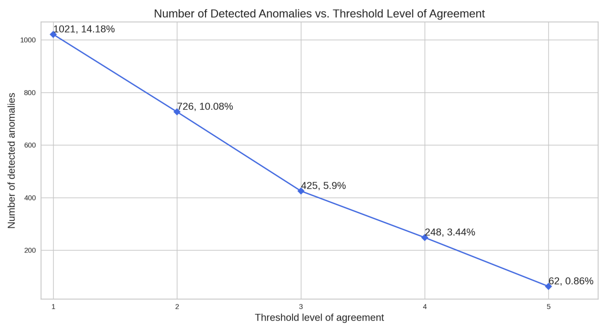

Figure 13: Variation in the number of detected anomalies while changing the threshold level of agreement.

This approach led to identifying 425 anomalies, corresponding to 5.9% of the entire dataset. These anomalies can be visualized in Figure 14.

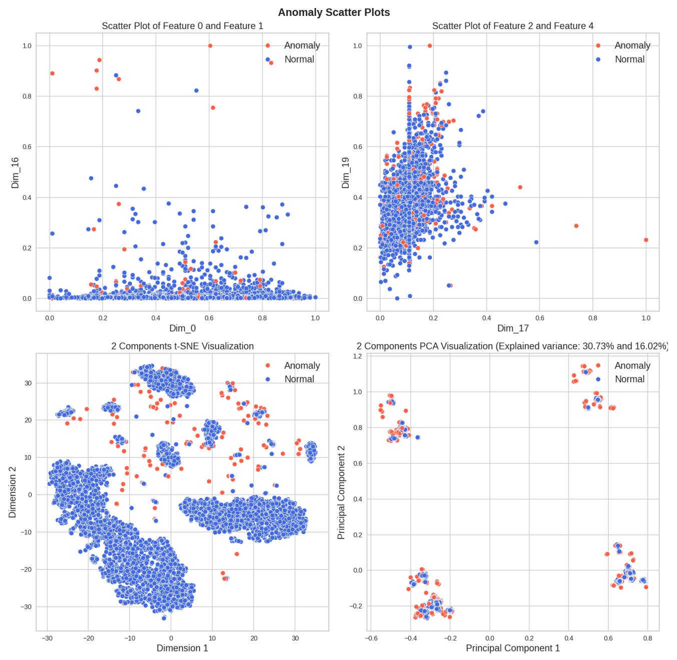

Figure 14: Majority vote anomaly detection visualized with scatter plots and dimensionality reduction techniques.

A. *Coherence Check*

Visually inspecting the anomaly detection scatter plots from different algorithms and observing how the number of detected anomalies changes in the majority vote approach highlights disagreements among the algorithms over some data objects. We can check the coherence between the algorithms using the Adjusted Rand Index (ARI), a similarity measure widely used in machine learning to evaluate and compare clustering algorithms. While the basic Rand Index (RI) measures the agreement between two clusterings by considering all pairs of samples, it doesn't account for random agreements. ARI adjusts for this by comparing the RI to the expected value of the RI for random clusterings. Scikit-learn's `adjusted_rand_score` ranges from -0.5 (discordant clusterings) to 1 (identical clusters), with values close to 0.0 representing random labeling. The ARI computed pairwise between the algorithms (Table 2) shows a wide variation in levels of agreement.

	KNN	**LOF**	**DBSCAN**	**PCA**	**OCSVM**	**MV**
KNN	1	0.219	0.655	0.374	0.702	0.780
LOF		1	0.294	0.455	0.268	0.423
DBSCAN			1	0.498	0.587	0.729
PCA				1	0.390	0.513
OCSVM					1	0.788
MV						1

Table 2: Adjusted Rand Index computed pairwise between the algorithms.

While the majority vote is quite coherent with all the employed techniques, algorithms like LOF and PCA seem to disagree to some extent with the other approaches. A strong coherence can be seen pairwise between KNN, OC-SVM, and DBSCAN. However, it's important to remember that different algorithms have different assumptions and mechanisms for identifying anomalies. While KNN-based methods might detect anomalies based on distance or density metrics, PCA might detect anomalies based on variance, and OC-SVM might use hyperplane separation.

The different nature of anomalies seen by each algorithm convinces us to choose the majority vote approach as the best model, as it combines the strengths of all the algorithms.

B. *Conversion to probabilities*

The final task involves estimating the probability of each data object being an anomaly. A logistic regression model is trained on the data using anomaly labels extracted from the majority vote model. Once the relationship is learned, the model uses `predict_proba` to generate probabilities for each data point.

VI. Conclusions

In this study we conducted an analysis on a medical dataset related to hypothyroidism with the purpose of identifying anomalies in patient data. After an initial thorough examination of attribute types and necessary preprocessing steps, we applied multiple anomaly detection algorithms, each producing a possible outcome.

These algorithms followed different approaches, each presenting some limitations, such as sensitivity to specific hyperparameters or reduced effectiveness when handling high-dimensional data. Moreover, our own approach in conducting the detection varied from case to case: for some methods we established a predefined threshold to identify anomalies, while for others, after a more keen analysis, we considered it appropriate to adjust such threshold based on deeper insights gained from the data.

At all times, the unsupervised settings of the problem, thus the impossibility of comparing our results with ground truth labels, posed a difficult challenge in terms of evaluating the results of individual techniques. Therefore, we decided to combine the various detections together through a majority vote approach, aiming to mitigate the drawbacks of the individual procedures and achieve a more robust anomaly detection by leveraging collective decisions.

We confirm that this report is entirely original and free from plagiarism. The research were conducted exclusively by the authors, without the use of ChatGPT or any other automated language models. All sources we used are properly cited in the references.

References

[1] S. N. B. M. F. Mendes D Alves C, "Prevalence of Undiagnosed Hypothyroidism in Europe: A Systematic Review and Meta-Analysis.," *Eur Thyroid J.*, vol. 8, pp. 130–143, 2019.

[2] K. D. H. B. e. a. Šabić E., "Healthcare and anomaly detection: using machine learning to predict anomalies in heart rate data.," *AI & Soc*, vol. 36, pp. 149–158, 2021.

[3] T. pandas development team, "pandas-dev/pandas: Pandas." [Online]. Available: https://doi.org/10.5281/zenodo.3509134

[4] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007, doi: 10.1109/MCSE.2007.55.

[5] M. L. Waskom, "seaborn: statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021–3022, 2021, doi: 10.21105/joss.03021.

[6] J. C. Gower, "A General Coefficient of Similarity and Some of Its Properties.," *Biometrics*, vol. 27, pp. 857–871, 1971.

[7] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[8] V. Satopaa, J. Albrecht, D. Irwin, and B. Raghavan, "Finding a "Kneedle" in a Haystack: Detecting Knee Points in System Behavior," 2011.

[9] P. T. Inc., “Collaborative data science.” [Online]. Available: https://plot.ly/

[10] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “LOF: identifying density-based local outliers,” in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, in SIGMOD '00. Dallas, Texas, USA: Association for Computing Machinery, 2000, pp. 93–104. doi: 10.1145/342009.335388.

[11] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, and others, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *kdd*, 1996, pp. 226–231.

[12] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang, “A novel anomaly detection scheme based on principal component classifier,” in *Proceedings of the IEEE foundations and new directions of data mining workshop*, 2003, pp. 172–179.

[13] B. Schölkopf, R. C. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, “Support Vector Method for Novelty Detection,” in *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. Müller, Eds., MIT Press, 1999, p. . [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/1999/file/8725fb777f25776ffa9076e44fcfd776-Paper.pdf