Università degli Studi di Napoli Federico II

Corso di Laurea Magistrale in Data Science

Corso di Hardware & Software for Big Data (Mod. B)

Prof.ssa Flora Amato

Anomaly Detection Project

EDA and Modeling

Anno Accademico: 2024/2025

Componenti del gruppo:

Chiara Caricchia

Claudio Cataldo

Fabio Fontana

Alessandro Maddaloni

Indice

1 Introduction to the Problem and Methods

This project aims to detect anomalies within a time-series dataset related to the operation of industrial machinery. In a production environment, anomaly detection plays a crucial role: promptly identifying abnormal behaviors helps prevent failures, optimize maintenance, and reduce operational costs.

The report focuses on both the theoretical aspects and the practical implementation of machine learning techniques for the automatic identification of events that deviate from a behavior considered normal. These techniques find applications in several fields, including industry, finance, and fraud detection.

After an overview of the concept of anomaly in time series and the main challenges posed by the complex and variable nature of the data, the project presents an Exploratory Data Analysis designed to highlight the key features of the dataset and justify the use of two subsequent algorithms: DBSCAN, a density-based clustering method, and Isolation Forest, an approach that leverages decision trees to efficiently isolate anomalous points, even in large datasets.

Both models were applied to the dataset and compared in terms of mechanism, parameters, and performance to assess their effectiveness in detecting anomalies within the analyzed context.

1.1 Anomalies

The definition of an anomaly varies depending on the application context, which also influences the approach adopted for its detection. One of the most widely cited definitions is that proposed by Hawkins (1980), according to which an anomaly is an observation that deviates so significantly from the others that it raises suspicions of having been generated by a different process than the norm [2].

This perspective implies that anomalous observations not only statistically differ from the rest of the data but may also be associated with different probability distributions compared to those describing normal data. Consequently, a good anomaly detection model should be able to quantify such deviation, assigning each point a probability of being considered anomalous.

Types of anomalies

The classification of anomalies can vary depending on the context and literature, but one of the most recognized is that proposed by Chandola et al. [1], which distinguishes three main categories:

- **Point anomalies:** occur when a single observation significantly deviates from the rest of the data. In an industrial context, for example, a sudden spike in the temperature of a machine—without changes in operating conditions—might indicate a point anomaly potentially related to an imminent fault.
- **Contextual anomalies:** a data point appears anomalous only when analyzed in the context in which it occurs. A value may be normal in absolute terms but inconsistent with the season, operating cycle, or time of day. For instance, a high temperature might be normal during a heating phase but anomalous during a cooling phase.
- **Collective anomalies:** involve a set of observations that appear normal individually but together constitute anomalous behavior. This often occurs in time series data: a sequence of regular vibrations that occurs out of phase or at unexpected times may indicate a collective anomaly linked to a cyclic malfunction.

2 EDA

Exploratory Data Analysis (EDA) represents the first step in understanding the dataset and preparing it for modeling. In particular, the structure of the data, the distribution of variables, the relationships among them, and their temporal behavior were evaluated using graphical and statistical techniques.

2.1 Dataset Description

The dataset consists of 499 observations and 7 columns. Its structure is as follows:

- `Finestra`: an integer identifier representing the temporal sequence of observations. It is not a real timestamp, but rather a numbered window that indicates the chronological order in which the data was recorded [int64].
- `batterie`, `inverter`, `conversione`, `scaricatori`, `controllo`, `raffreddamento`: six continuous numerical variables generated by an autoencoder, which summarize the condition of the respective subsystems of the plant [float64].

Since the six variables are derived from an automatic transformation using a neural network (autoencoder), they are not explicitly associated with physical units, and the exact criteria by which they were computed are unknown. However, through a supporting JSON file, it was possible to trace back the original variables that fed the autoencoder. Each subsystem included several measurements, such as voltages, currents, temperatures, binary states, and other indicators. Specifically:

- `batterie`: battery voltage, current, temperature, and state of charge;
- `inverter`: temperature, efficiency, and bypass status;
- `raffreddamento`: fan speed and ambient temperature;
- `conversione`: output voltage/current, overload;
- `scaricatori` and `controllo`: UPS status (Uninterruptible Power Supply), overvoltages, and remaining autonomy.

The dataset contains no missing values, and each row represents the overall state of the system at a given time window.

2.2 Univariate Analysis

As a first step, time series plots were generated for each variable. This preliminary visualization is useful to get an initial impression of the data trends and to identify any

"visually obvious" anomalies, such as sudden spikes, level shifts, or trend changes. This is purely a qualitative observation, useful for starting the exploration of the dataset.

In the time series plots, a **moving average** with a window size of 12 is also shown in red. This helps smooth local fluctuations and highlights the general behavior of the variable over time, making it easier to detect potential anomalous deviations from the trend.

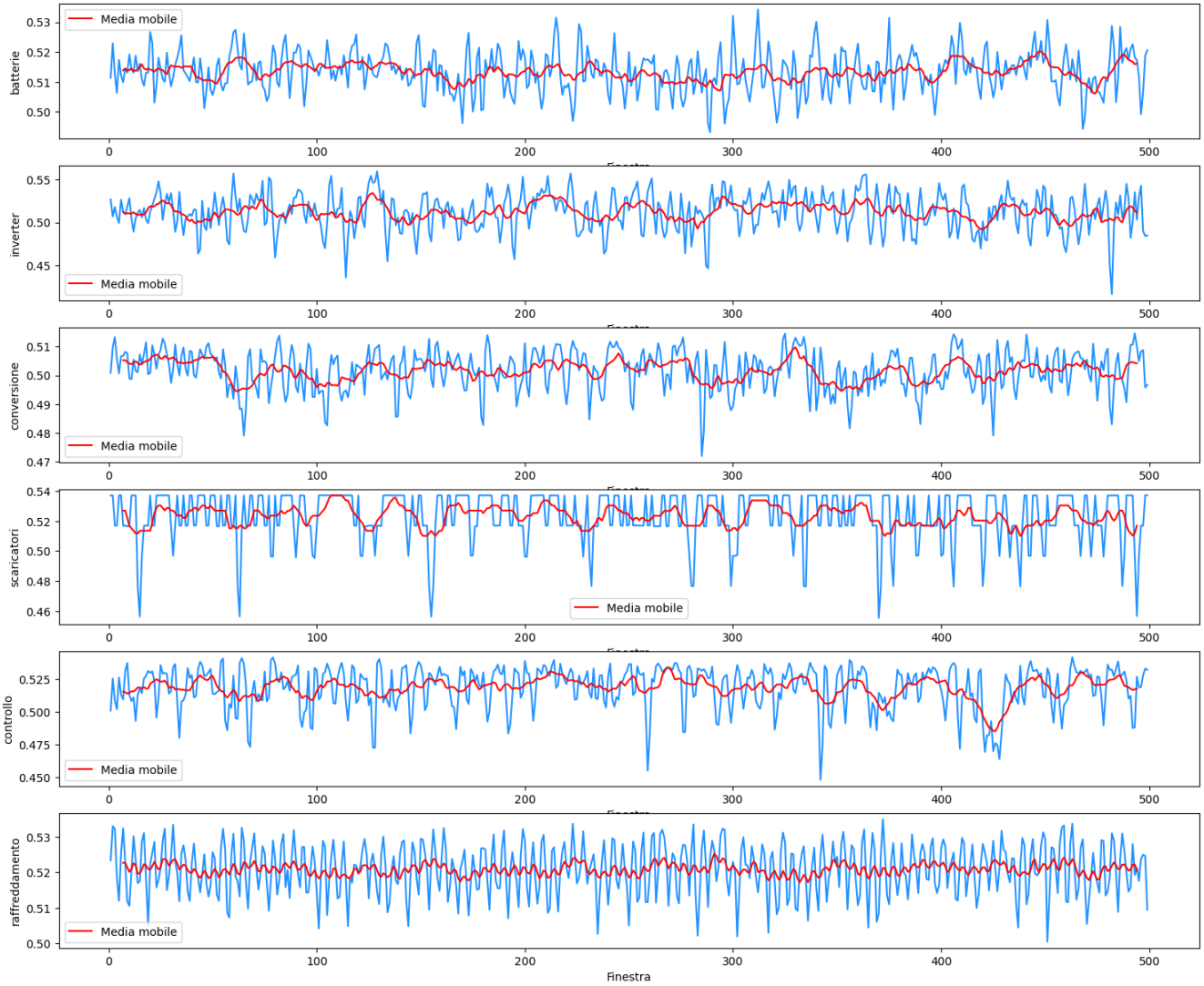

Figura 1: Time trend of the variables with moving average (window = 12)

Different trends can be observed among the variables: for example, `scaricatori` and `controllo` show sharper and more frequent variations, while `batterie` and `raffreddamento` appear more stable over time, suggesting a high likelihood of stationarity. The presence of sudden peaks and drops may already indicate potentially anomalous events, to be further investigated in the subsequent analysis stages.

Next, boxplots were generated for each variable, with the goal of identifying potential outliers in a simple and direct way. Although this technique does not take into account the temporal structure or the correlation between variables, it provides a quick view of the distributions and abnormal tails.

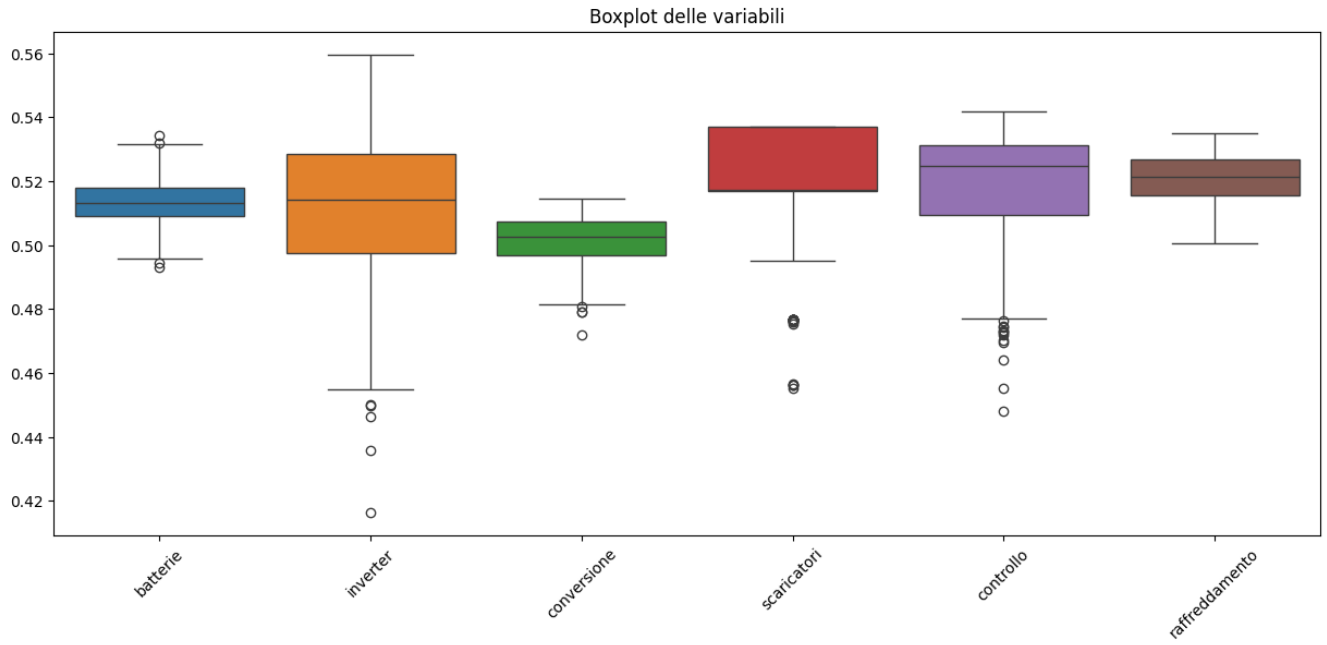

Figura 2: Boxplot of the variable `controllo`

It is observed that:

- The variable `inverter` shows greater dispersion compared to the others, with several outliers below the first quartile.
- `controllo` and `scaricatori` also display anomalous values, particularly in the lower part of the distribution.
- The variables `batterie`, `conversione`, and `raffreddamento` appear more stable, with a narrower interquartile range and fewer outliers.

Finally, for each variable, histograms and corresponding Q-Q plots were generated. Histograms allow for an intuitive visualization of data distribution, helping to identify possible skewness, heavy tails, or abnormal concentrations. Q-Q plots (Quantile-Quantile) are instead useful to compare the observed distribution with a theoretical normal distribution: any deviations from the expected quantiles highlight the non-normality of the variable.

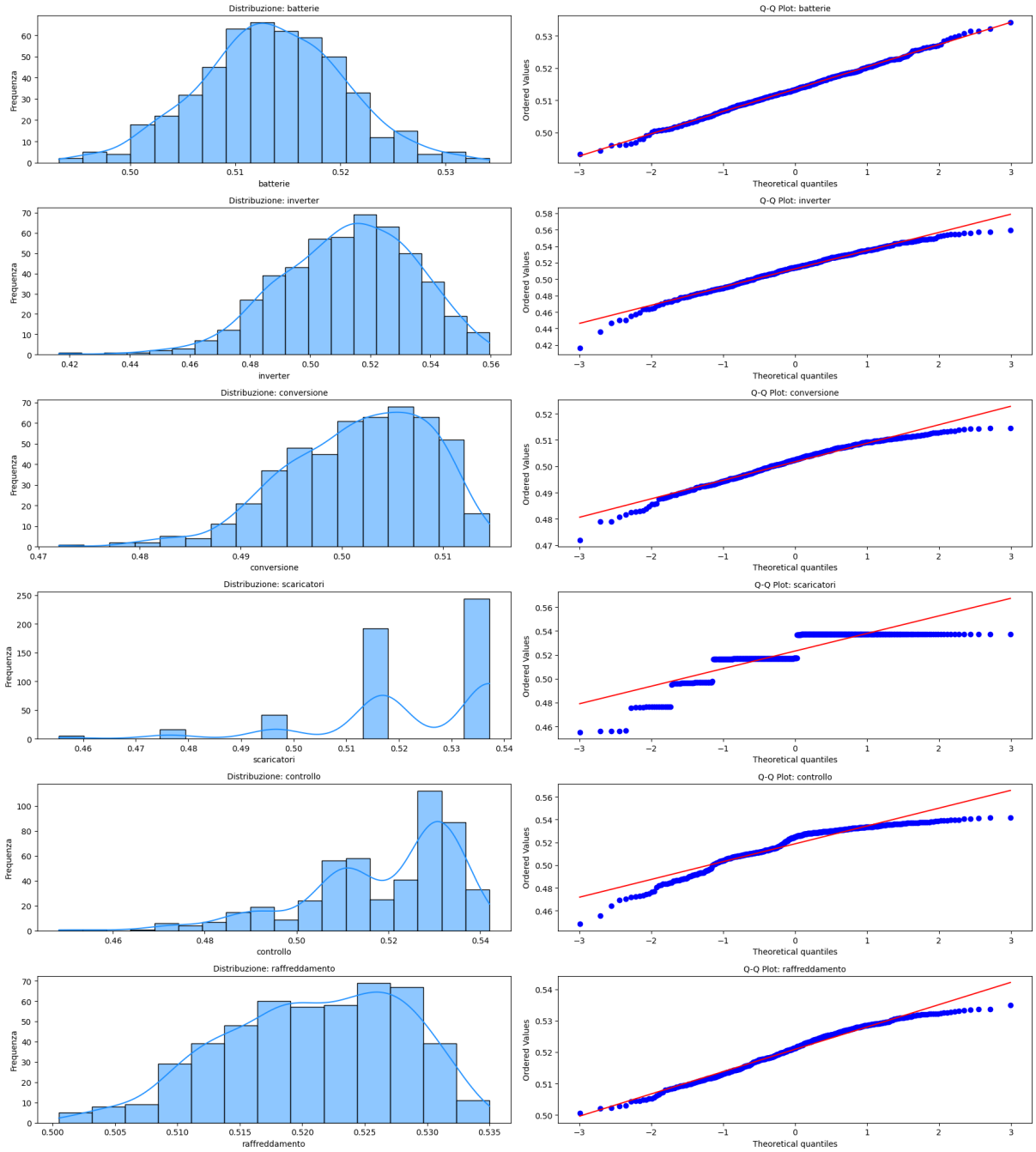

Figura 3: Boxplot of the variables

It is observed that the variables `batterie`, `inverter`, `conversione`, and `raffreddamento` exhibit behavior that is relatively close to a normal distribution, albeit with some deviations in the tails. The variables `controllo` and especially `scaricatori`, on the other hand, display less regular patterns.

In particular, `scaricatori` shows atypical behavior: the histogram highlights the presence of only a few distinct values, grouped into well-separated bins (5 in total). This suggests a discrete or strongly quantized distribution. To verify this, the `unique()` command in Python was used to directly list the distinct values present in the variable and confirm its non-continuous nature.

It was found that, although formally a continuous variable (type float64), it takes on only a limited number of distinct values, very close to one another.

More precisely, the unique values are concentrated in a few well-defined numerical intervals. By grouping the values according to the first two decimal digits, five main groups can be identified: approximately 0.45, 0.47, 0.49, 0.51, and 0.53. This observation suggests that the variable, although not explicitly categorical, behaves in a quasi-quantized manner.

This behavior is most likely due to the nature of the original variables used to feed the autoencoder. In particular, the `scaricatori` subsystem was composed of two parent variables: `num_sovratensioni` (discrete) and `stato_scaricatori` (binary). The combination of a binary variable and an integer variable with low variability may have resulted in a compressed and heavily aggregated set of values in the output of the autoencoder.

By examining the Q-Q plots, it is possible to assess that the variables `batterie`, `inverter`, and `raffreddamento` exhibit an overall regular behavior, with only slight deviations in the tails.

In the case of the variable `scaricatori`, the Q-Q plot reveals a strongly non-linear pattern: the points do not follow the expected diagonal alignment that would indicate normality, but instead form horizontal or broken segments, typical of discrete or heavily aggregated distributions. This behavior confirms the observations made in the histogram and the list of unique values, highlighting how this variable significantly deviates from a Gaussian distribution.

Normality test: Shapiro-Wilk To support the observations made with histograms and Q-Q plots, the **Shapiro-Wilk test** was applied. This test checks the null hypothesis that a variable follows a normal distribution. The test was implemented in Python using the `scipy.stats` library.

The analysis showed that only the variable `batterie` had a high p-value ($p = 0.912$), which does not lead to rejection of the null hypothesis of normality. All other variables had very low p-values, below the threshold of 0.05, indicating a marked deviation from normality. In particular, the variable `scaricatori` displayed a strongly non-normal distribution, in line with the previously observed graphical evidence.

These results confirm the non-Gaussian nature of the data and justify the use of non-parametric techniques for anomaly detection.

Stationarity Analysis: ADF and KPSS Tests To assess the stationarity of the dataset variables, two complementary statistical tests were applied: the **Augmented Dickey-Fuller (ADF) test** and the **KPSS test (Kwiatkowski-Phillips-Schmidt-Shin)**. Both tests were performed on the six variables, excluding the `Finestra` column, using the `statsmodels` library in Python.

The ADF test checks the null hypothesis of *non-stationarity*, while the KPSS test assumes as its null hypothesis that the series is *stationary*. The results showed that all variables simultaneously satisfied both criteria: the ADF test returned extremely low p-values for all variables (all < 0.001), while the KPSS test returned p-values equal to 0.1, confirming the absence of evidence against stationarity.

This suggests that, at least from a statistical point of view, the six variables can be considered **stationary**. This is particularly useful as it simplifies time series analysis and allows the application of models that assume this property without requiring preliminary transformations.

Autocorrelation and Partial Autocorrelation: ACF and PACF To analyze the temporal structure of the variables, the graphs of the **Autocorrelation Function (ACF)** and the **Partial Autocorrelation Function (PACF)** were computed for each time series, up to the tenth lag. These tools help to understand the degree of dependency of each observation on its past values.

The **ACF** measures the correlation between a variable and its lagged values, including both direct and indirect effects (i.e., also mediated by intermediate lags). The **PACF**, on the other hand, measures the correlation between the variable and a specific lag, removing the influence of previous lags. In other words, while the ACF shows how a variable is generally correlated with its past, the PACF highlights only the direct and independent relationship with each lag.

In the graphs, each bar represents the correlation coefficient for a given lag. The blue bands indicate the confidence interval: bars that exceed these limits are considered statistically significant.

ACF and PACF of the variables:

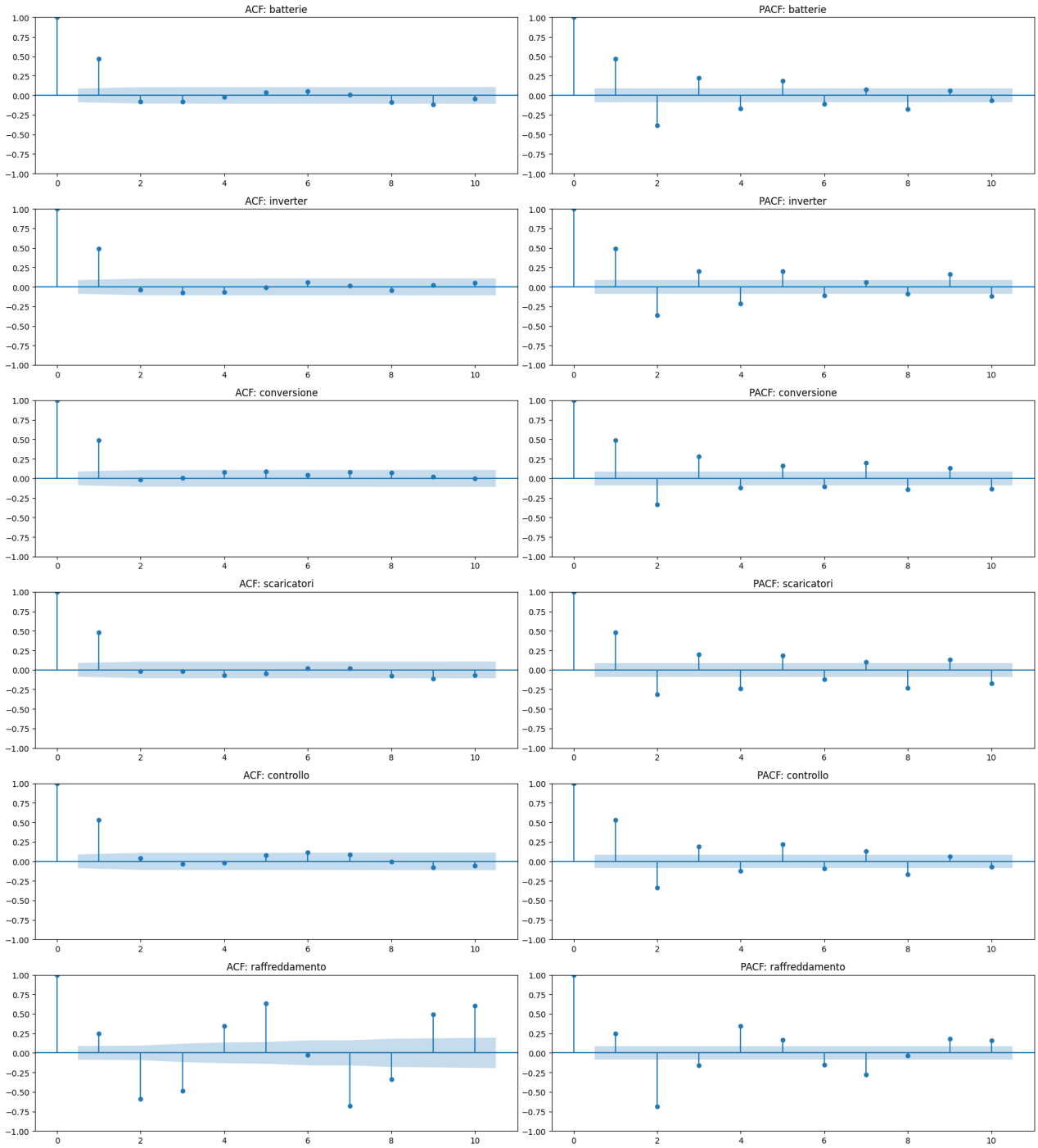

Figura 4: Autocorrelation (ACF) and Partial Autocorrelation (PACF) for the six variables.

From the analysis of the graphs, we observe that:

- The ACF of all variables shows a significant correlation only at the first lag, followed by values close to zero: this indicates that the observations are weakly dependent over time, in line with the results of the stationarity tests.
- The **PACF confirms** these results for most variables: here too, significant correlation is visible almost exclusively at lag 1.

- An exception is represented by `raffreddamento`, which in the PACF shows more than one significant lag (up to lag 3), suggesting a slightly more complex autoregressive structure or possible low-frequency seasonality.
- Also for `scaricatori` and `controllo`, the PACF highlights greater instability compared to other variables, with some significant peaks at later lags, but without a clear pattern.

Overall, the ACF/PACF analysis strengthens the idea that the dataset variables have a limited temporal component and confirms the validity of models that do not assume complex temporal dependencies.

2.3 Multivariate Analysis

To investigate the relationships between variables:

- A correlation matrix was constructed;
- Pairwise scatter plots were analyzed;

Pearson and Spearman Correlations To understand the relationships between the dataset variables, two types of correlation coefficients were computed:

- The **Pearson correlation coefficient** measures the linear relationship between two quantitative variables. It ranges from -1 to $+1$: values close to 0 indicate no linear relationship, while values near $+1$ or -1 indicate a strong positive or negative correlation, respectively. It is sensitive to the scale of the data and ideally requires the variables to be normally distributed.
- The **Spearman correlation coefficient**, on the other hand, measures the *monotonic* relationship — that is, the tendency for one variable to increase (or decrease) as the other increases. It is based on the ranks of the data rather than the absolute values, making it more robust than Pearson in the presence of non-normal distributions or nonlinear relationships.

Both coefficients were calculated for all pairs of variables in the dataset. The results were visually represented using pair grids (PairGrid), where:

- The upper triangle displays the correlation coefficients;
- The diagonal contains the univariate distributions;
- The lower triangle shows the corresponding scatter plots.

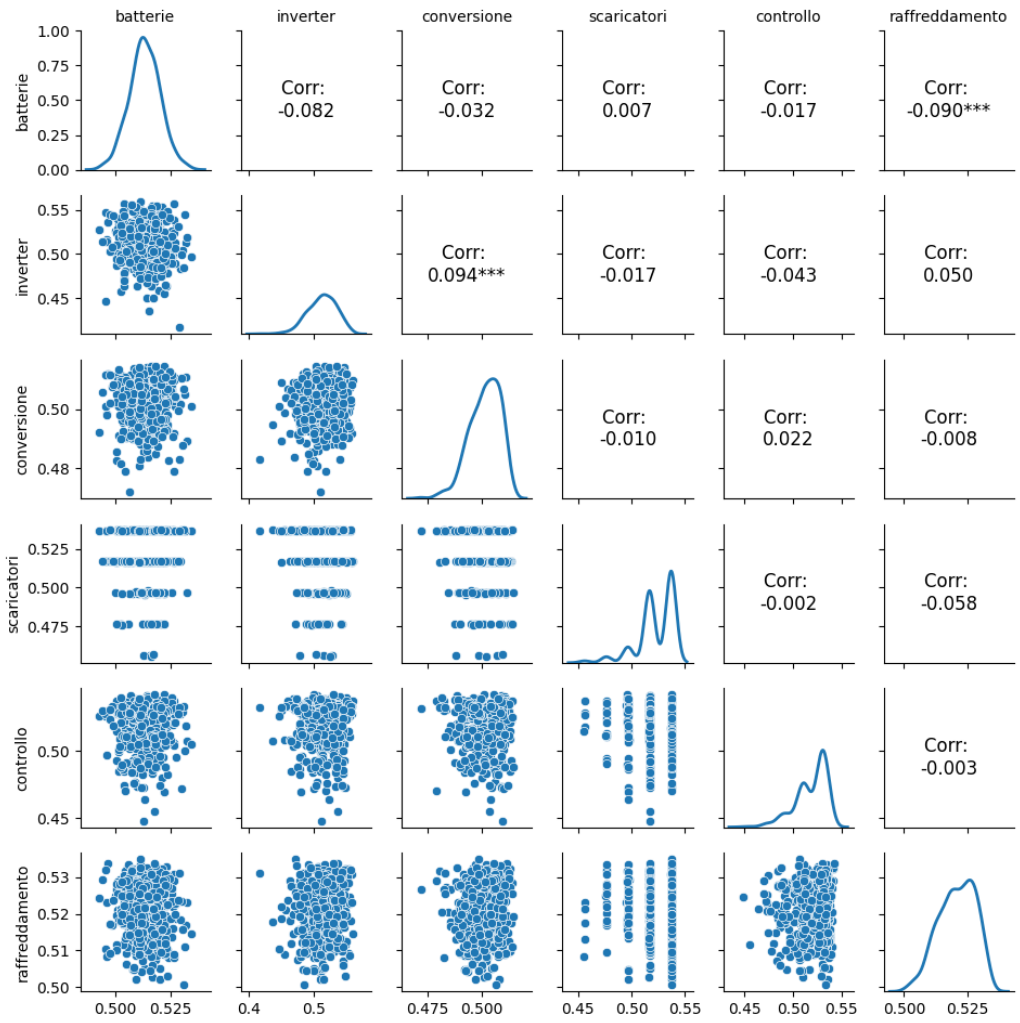

Figura 5: Pearson correlation among the variables.

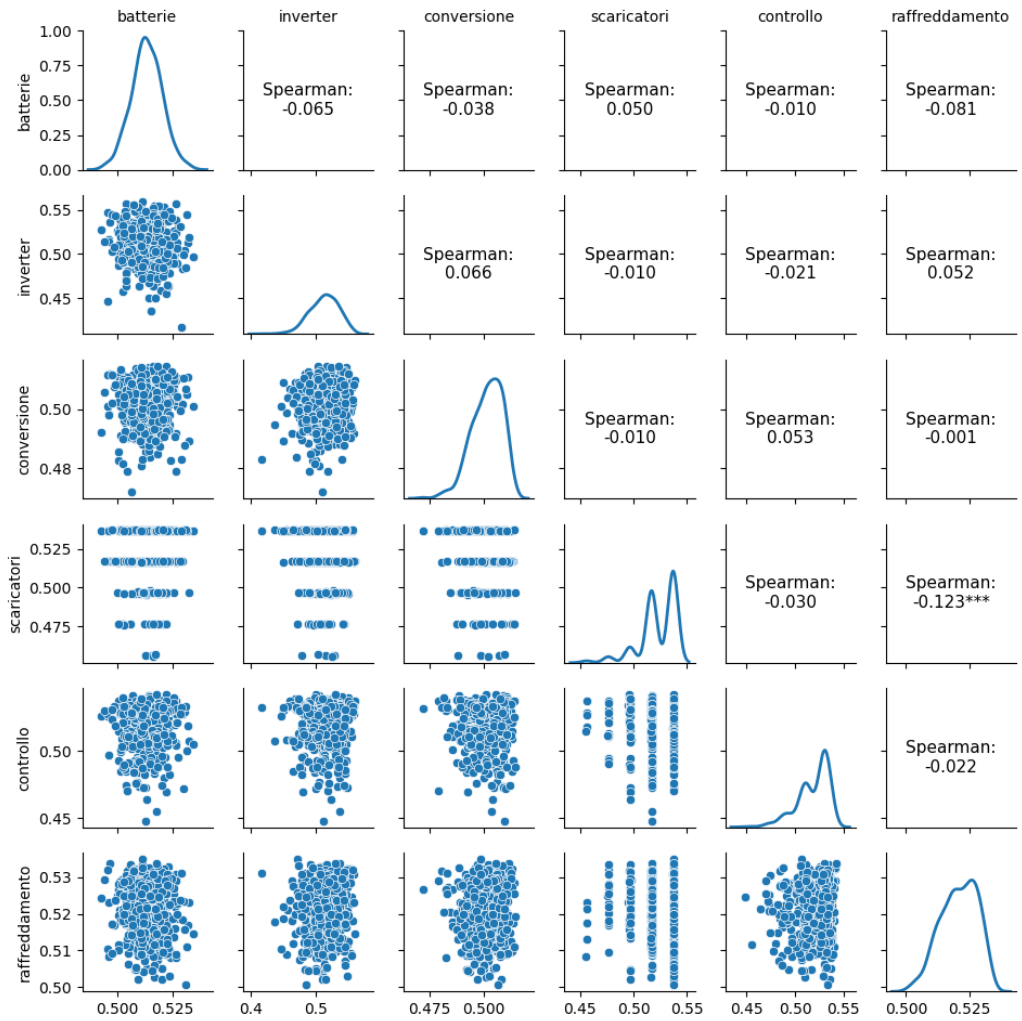

Figura 6: Spearman correlation among the variables.

Comparison of Results The analysis reveals that:

- The **Pearson** coefficients are all very close to 0, suggesting a lack of strong linear relationships between the variables. The only two statistically significant, though weak, correlations are:
  - `inverter`–`conversione`: $r = 0.094$
  - `batterie`–`raffreddamento`: $r = -0.090$
- The **Spearman** coefficients confirm the low presence of dependencies between variables. The only significant monotonic relationship identified is between:
  - `conversione`–`raffreddamento`: $r = -0.123$
- The variable `scaricatori`, consistent with its quasi-discrete nature, shows very low correlations with all others, according to both Pearson and Spearman.

2.4 Conclusions of the Exploratory Data Analysis (EDA)

The exploratory analysis provided a comprehensive overview of the dataset, highlighting its key characteristics from both a statistical and temporal perspective.

From the univariate analysis, the heterogeneity of the distributions emerged clearly: some variables showed relatively regular trends, while others (such as `scaricatori` or `controllo`) exhibited concentrations on discrete values or visible anomalies, also confirmed by the presence of outliers in the boxplots. The analysis of quantiles and normality tests revealed that most variables do not follow a normal distribution, justifying the use of non-parametric methods in the subsequent models.

From a multivariate perspective, linear correlation between variables was generally low, according to both the Pearson and Spearman coefficients. The absence of strong correlations suggests that the variables are not redundant and that each provides an independent informational contribution. This is an advantage in the context of anomaly detection, as it ensures that models — such as DBSCAN and Isolation Forest — can benefit from the diversity of signals without being distorted by strong linear dependencies.

Moreover, the presence of only mild monotonic correlations confirms the observations made in the previous analyses: the relationships between variables are complex, weak, or nonlinear, and therefore justify the adoption of robust, non-parametric techniques.

Overall, the EDA confirmed the multidimensional and nonlinear nature of the dataset, outlining the need for flexible approaches capable of capturing latent structures without overly rigid assumptions. The insights gathered will guide the selection and tuning of anomaly detection models in the following sections.

3 Anomaly Detection models

With our data, since we have no prior knowledge of which observations are anomalous, and therefore no target variable, we must face the problem of detecting machinery anomalies as an unsupervised one.

Among unsupervised methods, we can distinguish several approaches:

- *Statistical-based*, which seeks to identify the underlying model that generated the data, flagging as anomalous any observations that deviate from this process.
- *Distance-based*, based on the idea that an outlier is "far" from the majority of points.
- *Density-based*, which assumes that anomalies lie in regions of low local density.
- *Clustering-based*, where the data are first grouped into clusters and anomalies are then identified as: points not assigned to any cluster, points far from the centroid of the nearest cluster, or points belonging to unusually small clusters.

- *Isolation-based*, developed specifically for anomaly-detection problems, which treats as anomalous the points that can be isolated easily from the others.

However, because testing all of these methods to identify anomalies would go beyond the scope of this course, we have chosen to adopt, among these approaches, the two that appeared most suitable for an unsupervised multivariate problem:

- **DBSCAN**, cluster based
- **Isolation Forest**, isolation based.

3.1 DBSCAN

DBSCAN Algorithm: Theory and Implementation

Theoretical Introduction **DBSCAN** (Density-Based Spatial Clustering of Applications with Noise) is an unsupervised clustering algorithm that identifies groups of points based on local density, without requiring the number of clusters to be specified a priori. It is particularly effective in identifying arbitrarily shaped structures and in detecting outliers, which are labeled as noise points.

The algorithm relies on two key parameters:

- `eps` (ε): the maximum distance within which two points are considered neighbors;
- `minPts`: the minimum number of points required to form a dense cluster.

DBSCAN proceeds by identifying, for each point, the neighbors within a distance of (ε). If the number of neighbors is at least equal to **minPts**, the point is considered a "core" point, and a cluster expansion process is triggered, including all density-reachable points. Points that do not satisfy these criteria and are not reachable from any cluster are classified as outliers.

In our case, the algorithm was implemented in Python using the **Scikit-learn** library (class '`sklearn.cluster.DBSCAN`'). This implementation computes the distance matrix (Euclidean or Mahalanobis, depending on the case), and builds clusters based on density-connectivity logic.

Hyperparameter Optimization

Parameter Selection via Elbow Method To determine the optimal values for the parameters ε and *minPts*, a graphical approach based on the **Elbow Method** was adopted, adapted to the context of density-based clustering.

Specifically, for a given configuration of *minPts* (i.e., the minimum number of required neighbors), the distance from each point to its k-th nearest neighbor was calculated. By

sorting these distances in ascending order and plotting them, a characteristic curve is obtained with an inflection point (the "elbow"), which represents a natural threshold beyond which points are too far apart to belong to the same cluster. The value of ε corresponding to this elbow is therefore considered an optimal candidate.

In our study, the following results emerged:

- `Euclidean Distance`: the optimal range for ε was found to be between 0.015 and 0.025, with an optimal value of minPts equal to 5;
- `Mahalanobis Distance`: the optimal range for ε was between 1.00 and 2.00, with an optimal minPts equal to 4.

Parameter Selection via Grid Search Based on the ranges identified with the Elbow Method, a **Grid Search** procedure was implemented to systematically explore combinations of (ε) and *minPts*. For each parameter combination, the following steps were carried out:

1. Computation of the distance matrix (Euclidean or Mahalanobis);
2. Application of the DBSCAN algorithm with the precomputed distance matrix;
3. Evaluation of clustering performance through:
    - *DBCV (Density-Based Clustering Validation Index)* for the Euclidean distance;
    - *Silhouette Index* for the Mahalanobis distance.

DBCV is an index specifically designed to evaluate the quality of density-based clustering. It accounts for both intra-cluster cohesion (average density within clusters) and inter-cluster separation (density at cluster borders), providing a score between -1 and 1, where higher values indicate better-defined density structures. The index is particularly suited for algorithms like DBSCAN, where cluster shapes can be non-convex and the presence of outliers is explicit [3].

The *Silhouette Index*, on the other hand, measures how well each point fits within its own cluster compared to the others. It is defined as the normalized difference between the average intra-cluster distance and the average distance to the nearest cluster. It also ranges from -1 to 1: values close to 1 indicate well-separated and compact clusters, while negative values suggest incorrect assignments. Although not specifically designed for density-based clustering, the Silhouette Index is widely used for general evaluations of clustering quality [4].

The DBCV was computed only when at least two distinct clusters were detected; otherwise, a penalizing score was assigned. In addition to the validation indices, for each configuration the following information was recorded:

- The number of clusters identified;
- The number of outliers (labeled as -1);
- The values of ε and *minPts* used.

The results were sorted according to the validation index value, and the configuration with the highest score was selected for the final application of the model.

Visualization of DBSCAN Clusters via 3D t-SNE To support the interpretation of the clustering results obtained through the DBSCAN algorithm, the dimensionality reduction technique *t-distributed Stochastic Neighbor Embedding (t-SNE)* was used.

t-SNE is a nonlinear method designed to reduce high-dimensional data while preserving local relationships between points—i.e., the closeness of similar observations—making it particularly effective for visualizing complex cluster structures.

In this study, the original data were projected into a three-dimensional space using t-SNE with 3 components, PCA initialization, perplexity=30, and a fixed random seed to ensure reproducibility.

The 3D visualization assigned each point to the corresponding DBSCAN cluster. Points labeled as anomalies by DBSCAN were highlighted in red, while regular cluster members were assigned distinct colors. The resulting plot shows the spatial distribution of clusters and anomalies, providing an intuitive representation of the separation between normal and abnormal data points, while acknowledging the approximate nature of t-SNE projections for global structures.

Remarks The combination of applied techniques — grid search, elbow method, DBCV, and Silhouette Index — enabled a more robust identification of the optimal parameters for the DBSCAN algorithm, tailoring it to the structure of the dataset.

Application of DBSCAN with Euclidean Distance

Elbow Method To support the selection of the ε parameter for DBSCAN, the elbow method was used. For each value of `minPts` (from 3 to 8), the distances to the k-th nearest neighbor were sorted. The goal is to identify the point where the curve increases sharply: this point represents a natural threshold that separates dense points from isolated ones.

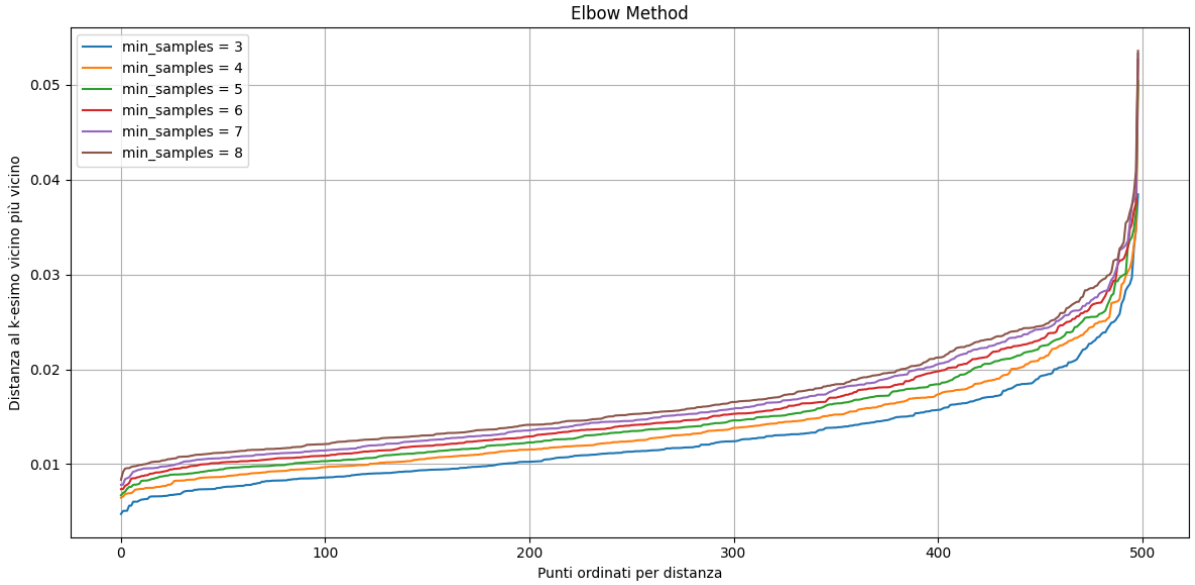

Figura 7: Elbow method for selecting ε with different values of `minPts`.

The visual analysis suggests a value of ε between 0.015 and 0.025. This range was also confirmed by the search conducted using the DBCV index, as shown in the table:

eps	min_samples	dbcv	n_clusters	n_outliers
0.021000000000000005	5.0	0.40841275743511973	2.0	32.0
0.021000000000000005	4.0	0.3194971015945837	2.0	26.0
0.022000000000000006	6.0	0.26842733738209706	2.0	29.0
0.022000000000000006	3.0	0.24657247758880324	2.0	15.0
0.015	5.0	0.18420097534217872	3.0	114.0
0.015	6.0	0.16382840202512103	4.0	123.0
0.015	4.0	0.1425344260602156	4.0	103.0
0.016	6.0	0.10923012436454081	3.0	104.0
0.017	7.0	0.10758603343490766	3.0	91.0
0.018000000000000002	6.0	0.10699336576193247	3.0	68.0

Figura 8: Grid Search result comparison using DBCV.

Clustering with Optimal Parameters Based on the grid search, the following optimal parameters were selected:

- $\varepsilon = 0.021$
- `minPts` $= 5$

The DBSCAN model was applied to the original numerical data. The results showed the presence of **3 distinct groups**:

- `cluster_0` (462 points)
- `cluster_1` (5 points)
- `anomaly` (32 points, labeled as `-1` by DBSCAN)

To graphically represent the clusters, a three-dimensional projection was used via the *t-SNE* technique applied to the original variables. The points were colored according to their cluster assignment, with red reserved for anomalies.

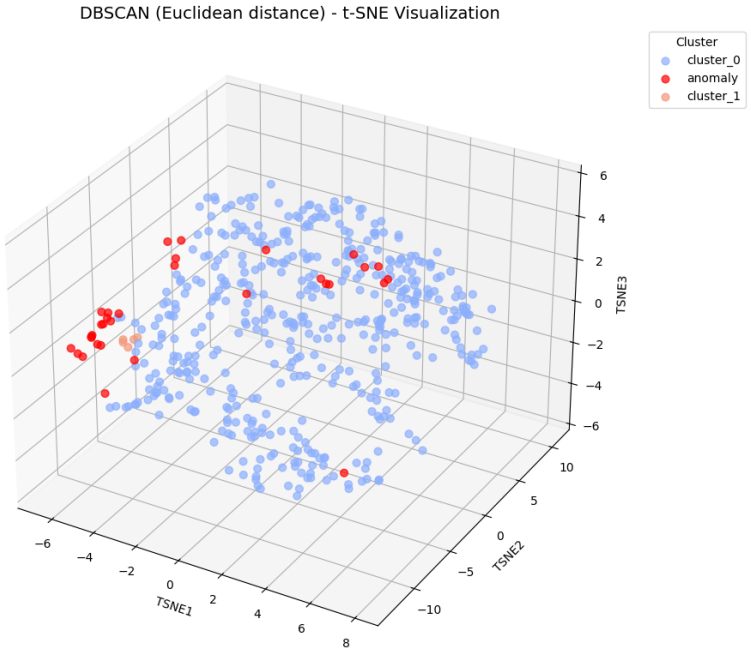

Figura 9: DBSCAN clustering with Euclidean distance — t-SNE visualization.

Remarks The model identified a large main group (`cluster_0`), a small secondary group (`cluster_1`), and a portion of anomalous points well separated from the rest. The spatial distribution shows that anomalies are located in low-density regions.

This application demonstrates how DBSCAN, supported by careful parameter selection and preliminary dimensionality reduction, is capable of effectively identifying outliers in an unsupervised context.

DBSCAN with Mahalanobis Distance

Elbow Method The elbow method was also used for the Mahalanobis metric to determine a plausible range for ε. Mahalanobis distance accounts for the correlation between variables, making it particularly suitable for multidimensional contexts with non-negligible

covariances. The inverse covariance matrix was calculated on the standardized version of the dataset, with numerical regularization to prevent instability.

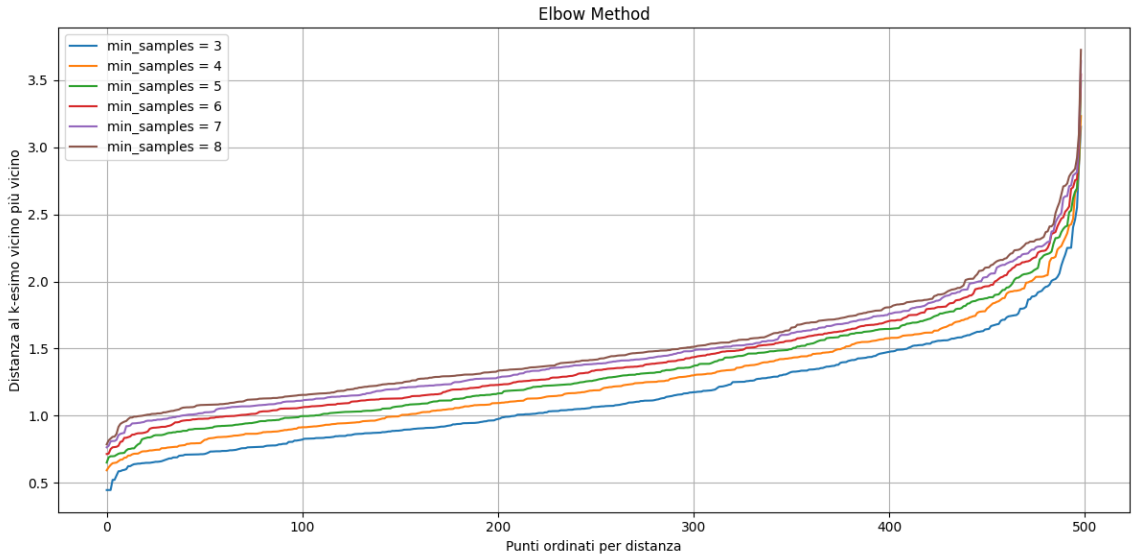

Figura 10: Elbow method with Mahalanobis distance.

Again, a visible inflection is observed around $\varepsilon \approx 1.6$. In this case, the visual analysis suggests a value of ε between 1.40 and 1.60. This range was also confirmed by the search conducted using the Silhouette index, as shown in the table:

eps	min_samples	n_clusters	silhouette_score	n_outliers
1.5959999999999839	4.0	2.0	0.2192715118606347	44.0
1.5969999999999838	4.0	2.0	0.2192715118606347	44.0
1.5929999999999842	4.0	2.0	0.21562470594338817	46.0
1.5889999999999846	4.0	2.0	0.21562470594338817	46.0
1.594999999999984	4.0	2.0	0.21562470594338817	46.0
1.593999999999984	4.0	2.0	0.21562470594338817	46.0
1.5899999999999845	4.0	2.0	0.21562470594338817	46.0
1.5909999999999844	4.0	2.0	0.21562470594338817	46.0
1.5919999999999843	4.0	2.0	0.21562470594338817	46.0
1.5969999999999838	3.0	2.0	0.21343775539946017	35.0

Figura 11: Grid Search result comparison using Silhouette index.

Clustering with Optimal Parameters Based on the Silhouette index and visual inspection, the optimal parameters were chosen as:

- $\varepsilon = 1.596$

- `minPts` = 4

The DBSCAN model, applied using Mahalanobis distance, produced the following results:

- `cluster_0`: 451 points
- `cluster_1`: 4 points
- `anomaly_mah`: 44 points

Here as well, the model identified a well-defined main group (cluster 0), alongside a small secondary cluster (cluster 1) and a portion of anomalous points clearly separated from the rest of the distribution. The projection into observation space shows that anomalies are located in low-density regions, with some being close to the main cluster, suggesting the presence of "borderline" anomalies that are difficult to distinguish using methods less sensitive to local density.

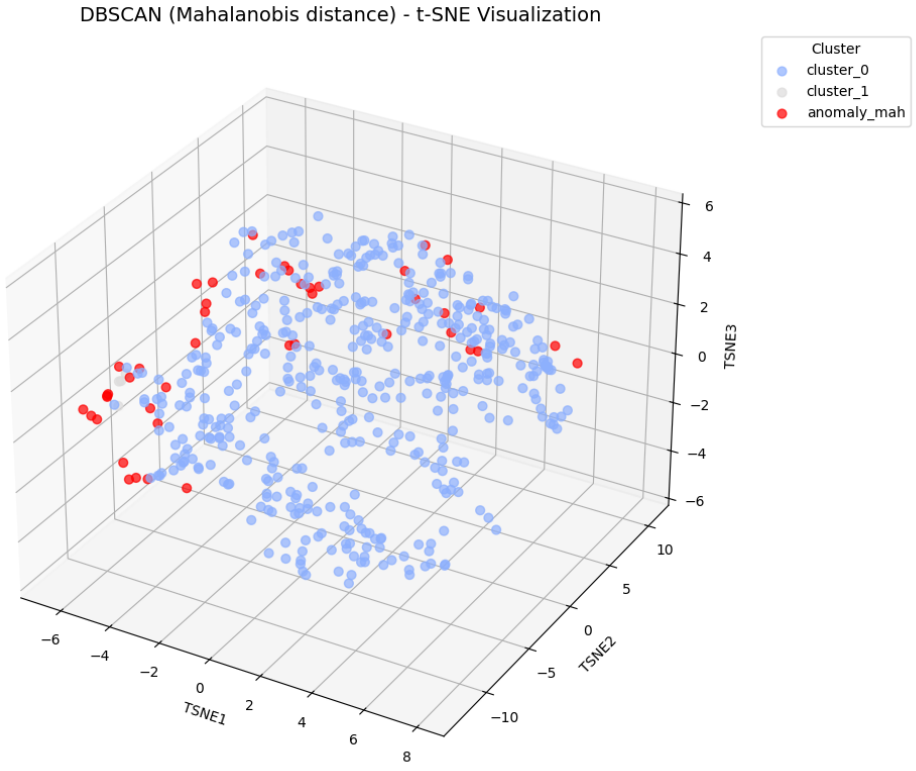

Figura 12: DBSCAN clustering with Mahalanobis distance — t-SNE Visualization.

Observations The DBSCAN algorithm, applied using both Euclidean and Mahalanobis distances, produced partially overlapping sets of outliers. Specifically, 21 observations were classified as anomalous by both metrics, indicating a common core of robust anomalies regardless of the chosen distance. However, the discrepancies observed suggest that the two distances capture complementary aspects of the data structure.

Mahalanobis distance takes into account the variance and correlation among variables, heavily penalizing points that deviate in directions with low variance (where little variability is expected), and being more permissive along directions with high variance. Consequently, even geometrically "close" points to the central distribution may be classified as anomalous if they lie in statistically unusual directions. In contrast, Euclidean distance assumes independence and orthogonality between variables, assessing dissimilarity in terms of "raw" distance without considering the covariance structure. This may lead to an underestimation of anomalies in directions with high variance or strong correlation.

Comparison Between Euclidean and Mahalanobis Distance

Numerical Comparison of Outliers The DBSCAN models, applied respectively with Euclidean and Mahalanobis distances, identified **32 outliers** and **44 outliers**, with **21 points** being classified as anomalous by both metrics.

- Outliers with Euclidean distance: 32
- Outliers with Mahalanobis distance: 44
- Common outliers: 21

Visual Comparison in the t-SNE Space To visualize the differences between the two metric configurations, the clustering results were compared in the three-dimensional space obtained via t-SNE.

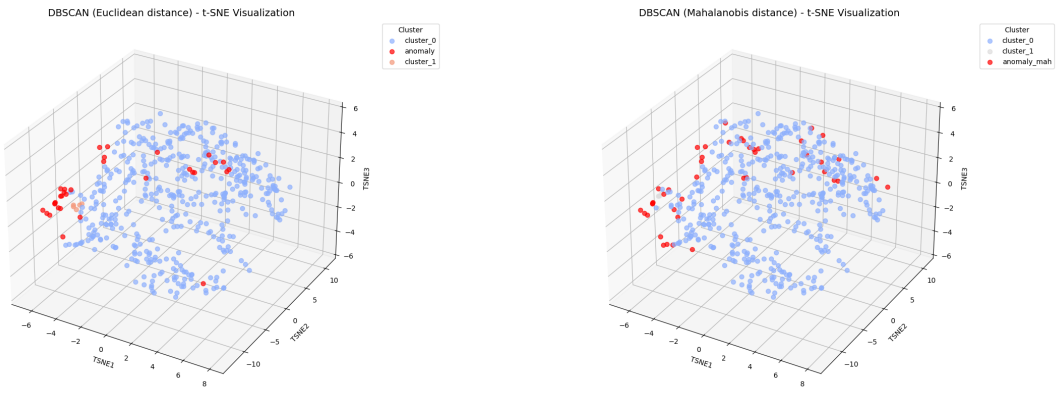

Figura 13: Comparison between clustering results obtained using DBSCAN with Euclidean distance (left) and Mahalanobis distance (right).

Both visualizations show good separation between the main cluster and the anomalous points; however, the spatial distribution of outliers differs significantly between the two metrics. In particular, Mahalanobis distance demonstrates greater sensitivity to statistically atypical deviations, even when these do not appear as geometrically pronounced. This behavior results in a higher anomaly detection capability in the presence of correlated or unevenly scaled variables, compared to Euclidean distance.

Temporal Distribution of Anomalies The results obtained from the DBSCAN models were projected onto the time series of the observed variables to identify the behavior of the different subsystems corresponding to the anomalous windows. It is important to highlight that the same anomalous points, determined through multivariate clustering, were represented across each variable to assess their local evolution.

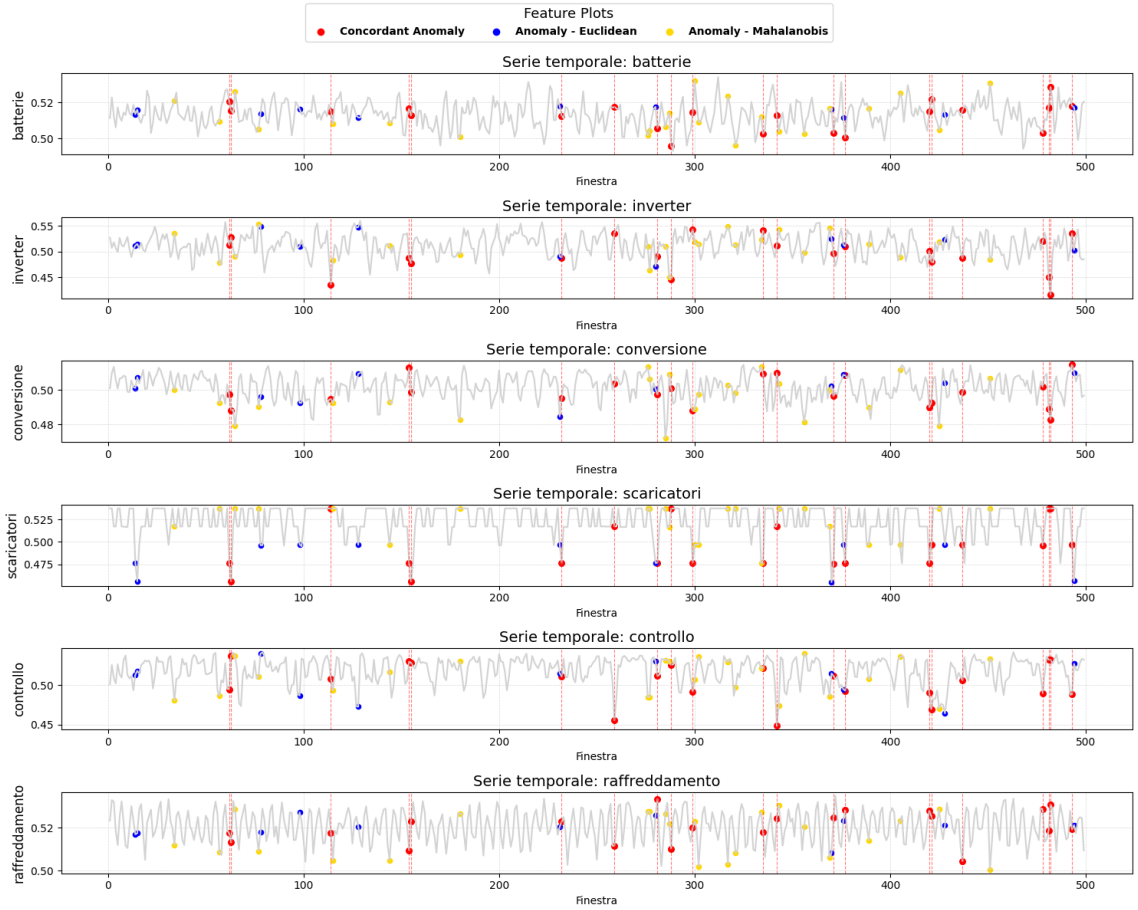

Figura 14: Anomalies detected within the original time series for each variable.

Several relevant patterns emerge from the visualization:

- The shared anomalies (red) are distributed throughout the entire time interval and often correspond to clearly visible deviations. Moreover, they frequently occur in pairs (within the time ranges around 50, 150, 420, 480).
- Anomalies exclusive to each method appear to depend on the variable and the internal structure of the data:
  - **Euclidean Anomalies** (blue): The model based on Euclidean distance often detects anomalies at the occurrence of sharp peaks or clear extreme values. This is consistent with the fact that Euclidean distance primarily measures direct deviation from centroids, without accounting for the internal variance of the variables.

– **Mahalanobis Anomalies** (yellow): The anomalies identified exclusively with Mahalanobis distance are visually less obvious but frequently appear near seemingly normal regions or subtle modulations. This is justified by the fact that Mahalanobis penalizes statistically unusual deviations, even if they are geometrically small, especially in low-variance directions.

Conclusions This representation highlights how multivariate clustering allows for the identification of anomalies even when individual variables do not exhibit apparent deviations, leveraging the synergy between dimensions. The overlap between Euclidean and Mahalanobis methods offers a more complete view, distinguishing between visually detectable anomalies and those that are statistically significant but less intuitive.

3.2 Isolation Forest

Isolation Forest, introduced by Liu et al. (2008), is an unsupervised ensemble algorithm that isolates observations using tree-based structures known as *isolation trees*.

The assumption underlying this model is that, because anomalies are few and distinct, they can be isolated from the rest of the data more easily. From a practical point of view, this goal is achieved by the isolation tree, which at each node splits the data in an attempt to isolate the extreme values of a randomly chosen variable. By aggregating the results of all the trees, and assuming a tree with n nodes, an anomaly score for each observation x is computed as:

$$s(x, n) = 2^{-\frac{E[h(x)]}{c(n)}}$$

where:

- $h(x)$ is the length of the path that an observation travels within a tree to reach a terminal node; its average is taken across all trees,
- $c(n)$ is the theoretical average value of $h(x)$ in a random tree with n nodes.

Thus, by using the anomaly score, we can make the following assessment:

- if instances return s very close to 1, then they are definitely anomalies,
- if instances have s much smaller than 0.5, then they are quite safe to be regarded as normal instances,
- if all the instances return $s \approx 0.5$, then the entire sample does not really have any distinct anomaly.

However, when applying Isolation Forest, we must also specify a rule that allows the model to translate the anomaly score in a anomaly dummy variable. That is the

contamination, which represents the percentage of observations that will be considered outliers.

Unfortunately, in an unsupervised setting like this there is no way to know which is the best value for the contamination, as it should require prior knowledge on the estimated percentage of outliers points in the data. In a real setting, this should be discussed with the area of the company that deals with the UPS, but in this case we have no way to access such informations, so we will set a placeholder value of 5% anomalies in the data.

3.3 Results

After fitting the model to our data, we obtain the distribution of anomaly scores for all observations. The location of zero on this scale is determined by the contamination parameter, because the threshold is set to separate anomalies from normal points.

As the histogram shows (Figure 1), most of the scores lie between -0.15 and -0.05, while the right-hand tail deviates from this trend. The anomaly threshold therefore isolates a distinct portion of the distribution where a clear knee point is visible. A different contamination value could be chosen to change the sensitivity of the model.

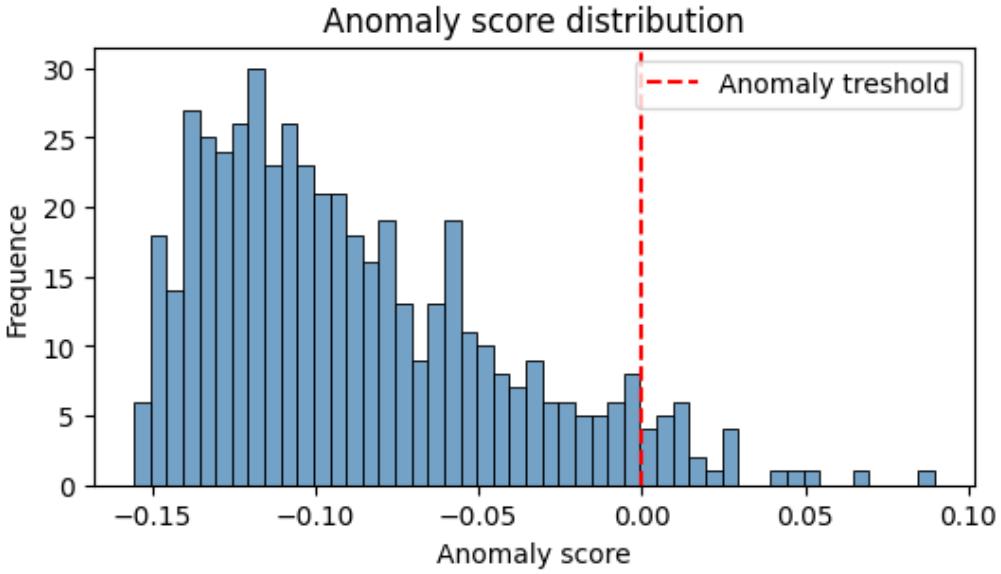

Figura 15: Anomaly score distribution with a red dotted line to separate anomalies.

Isolation Forest also provides feature-importance estimates (Figure 2), revealing that the variables contribute similarly to anomaly detection: *controllo* is the most influential feature, while *scaricatori* is the least.

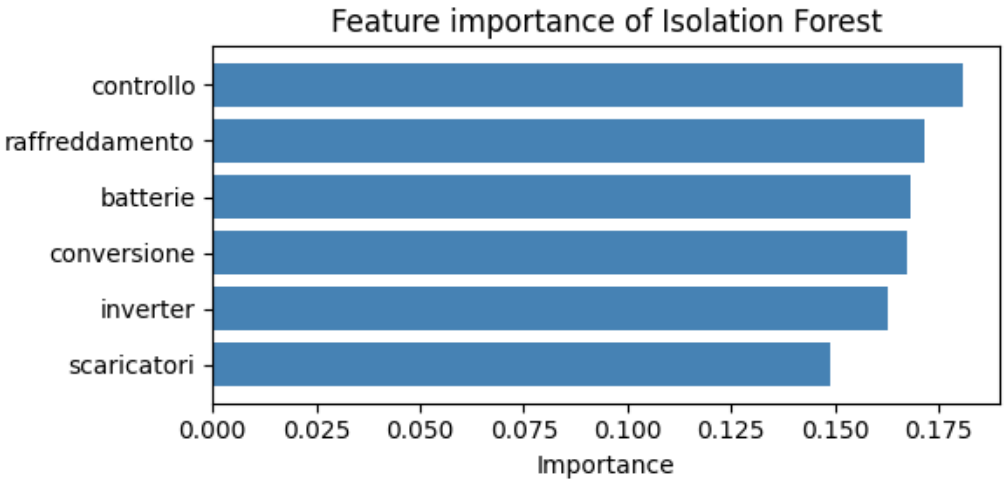

Figura 16: Feature importance in Isolation Forest

One further analysis can be done investigating the anomaly scores of the observations flagged as outlier by the two clustering methods previously applied.

Looking first at the DBSCAN model that uses Euclidean distance (Figure 3), roughly half of its flagged points are also classified as outliers by Isolation Forest. However, some of the "most anomalous" points, with the highest score, are missed by this DBscan model.

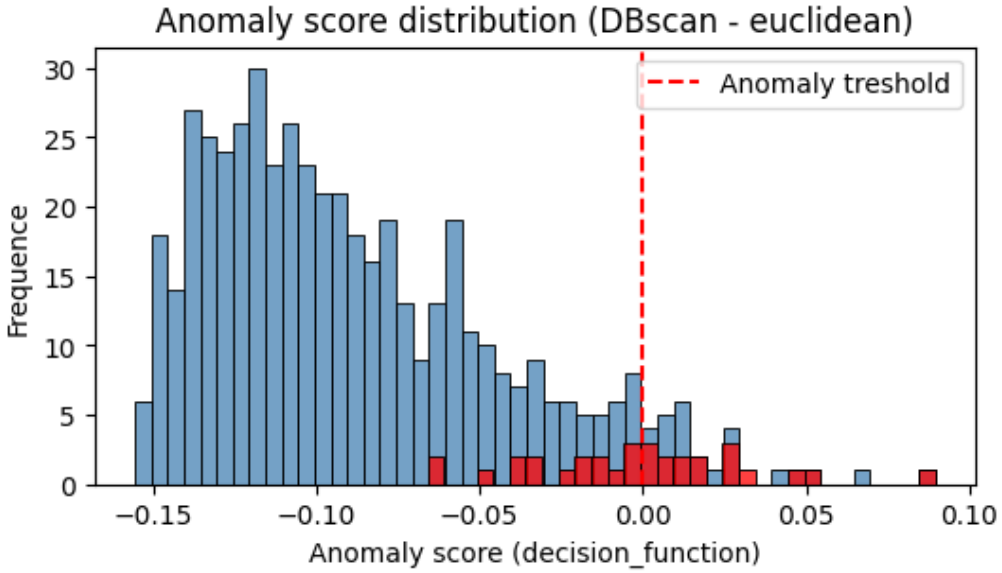

Figura 17: Anomaly score distribution of outliers detected by DBscan with euclidean distance (in red)

On the other hand, the DBSCAN variant based on Mahalanobis distance (Figure 4) identifies a larger set of outliers. Although nearly half of these were not flagged by Isolation Forest, many of the most extreme cases are detected by both models, indicating partial agreement on the most atypical observations.

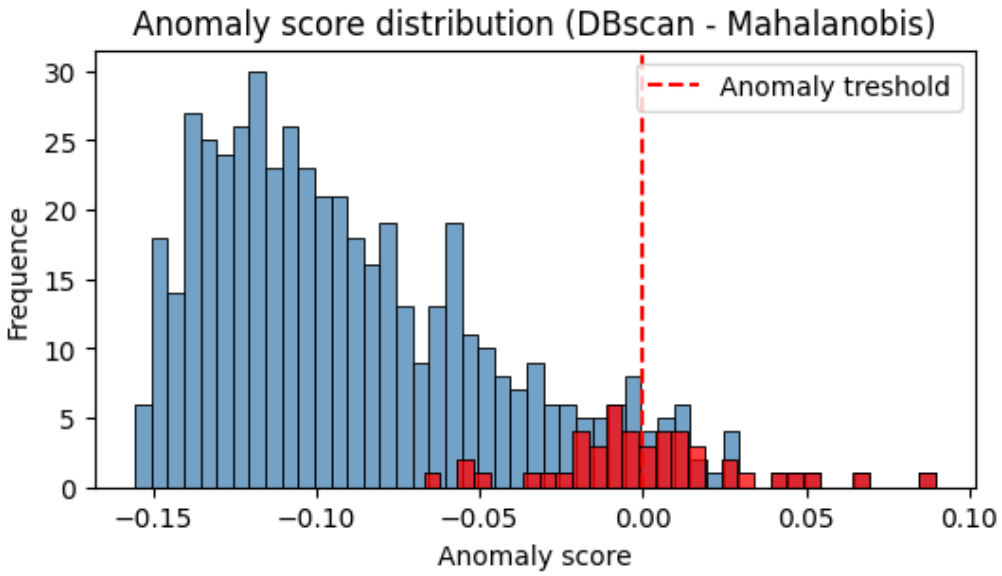

Figura 18: Anomaly score distribution of outliers detected by DBscan with Mahalanobis distance (in red)

4 Conclusion

In conclusion, since this is an unsupervised anomaly-detection problem, we cannot claim to have nailed down the moments when the machines actually malfunction. What we can say is that we have identified the points that deviate most strongly from each machine's normal operating behaviour.

The two models flag outliers in different ways:

- Isolation Forest isolates observations that are "easy to separate" from the rest of the data, assigning each one an anomaly score;
- DBSCAN labels as outliers the points that do not belong to any cluster of sufficient density. In particular, the Mahalanobis-distance variant is more sensitive in regions where variables are correlated.

When comparing the two models, the absence of a target variable means we cannot state which model is best, but several differences emerge:

- The most "extreme" outliers according to the Isolation Forest anomaly score are almost always flagged by DBSCAN as well, especially when Mahalanobis distance is used;
- Roughly half of the outliers detected by DBSCAN are also identified by Isolation Forest.

Beyond the straightforward feature-importance ranking provided by Isolation Forest, it would be worthwhile to undertake a root-cause analysis to understand why the detected deviations occur. In an actual industrial setting, this would involve:

- working with domain experts to validate the highest-priority anomalies;
- leveraging these insights to plan targeted preventive maintenance.

In this way, the outcomes of the unsupervised analysis become a concrete starting point for improving the machines' reliability.

Riferimenti bibliografici

[1] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):1–58, 2009.

[2] D. M. Hawkins. *Identification of Outliers*, volume 11. Springer, 1980.

[3] Davoud Moulavi, Pedro A. Jaskowiak, Ricardo J. G. B. Campello, Arthur Zimek, and Jörg Sander. Density-based clustering validation. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 839–847. SIAM, 2014.

[4] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.