Supervised vs Unsupervised Anomaly Detection

Alessandro Rosario Torrisi - s330732, Simone Sambataro - s331812,
Floralucy Cusano - s334142

1 Task 1: Dataset Charactization and Preprocessing

1.1 What are your dataset characteristics? How many categorical and numerical attributes do you have? How are your attack labels and binary label distributed?

Our train dataset has *38 numerical features* and *3 categorical features* (’*protocol_type*’,’*service*’,’*flag*’), that are protocol type, service and flag. The train partition contains a total of 18,831 connection records, with a 71.41% majority of normal traffic and 28.59% labeled as attacks. In the whole trainset, 15.51% is made of DOS attack, 12.12% is probe and the rest is made of r2l (0.96%). Out of this distribution we can see that the dataset is unbalanced. This is expected, given the well known difficulty of obtaining well characterized datasets with a substantial amount of compromised traffic.

1.2 How do you preprocess categorical and numerical data?

- We dropped duplicates and removed NULL values.
- Dropped columns: urgent, num_outbound_cmds, is_host_login, land.
  - **urgent**, **num_outbound_cmds**, and **is_host_login** are constant—every value is 0 in both train and test. A feature with zero variance carries no information and cannot help any model, so removing it speeds up training and reduces memory usage without affecting performance.
  - Hot must stay. It counts the number of “hot indicators” found in the packet payload (e.g. suspicious system call patterns, set-uid programs, etc.). The field shows real variation (0–101 in test).
  - We noticed that the feature *land* had a correlation coefficient of 0 with all the feature, and also, out of 18,831 samples, we have all 0 as values except for 1 row that has a value of 1, which is labeled as ”normal.” This corresponds to the case in which the source and destination IP are the same, which may represent a user trying to ping themselves. We decided to drop it because this feature may lead to label mismatch, making the sample appear as an anomaly among normal samples, since every other sample has 0 except for one.
- We decided to use one-hot encoding for the three different categorical features.
  - **Service**: before applying one-hot encoding to this feature, we saw that it had 11 possible values. We decided to keep only the values that had a presence of at least 1% in the dataset; the others were grouped together into the category ”others.” In the end, the services kept were: ”http,” ”private,” ”smtp,” ”domain_u,” ”other,” ”ftp_data,” ”ecr_i,” ”eco_i,” ”telnet,” ”finger,” ”ftp.”
  - **Flags**: the same reasoning applies to this category, we groupped all the features that appeared less then 1% in the dataset, and the values that we kept in the end were: ”SF,” ”S0,” ”REJ,” ”RSTO.”
  - **Protocols**: in this case, it was not necessary to reduce the number of possible values since the main values of this feature were only 2 (TCP and UDP).

Even though there are a lot of features with a high value of correlation (a lot of features have values around 0.95 with each other) we still decided to keep them since there is the possibility that through models that use Deep Learning methodology, those models might be able to catch multicollinearity between those features.

1.3 Looking at the different heatmaps, do you find any main characteristics that are strongly correlated with a specific attack?

The **standard deviation map** in Figure 7 may have the following interpretation:

- **DoS** and **Probe** have the highest σ for byte counts and connection rates, indicating bursty behavior.
- **R2L** shows almost zero variance across most fields; these attacks are rare and highly deterministic, they do not try to overwhelm the network but simply try to exploit vulnerability by sending well crafted payloads that may be hard to detect by just looking at normal metrics.

Feature Attack Correlation

There are features strongly correlated with specific attack types:

- **SYN error features** $\leftrightarrow$ **DoS:** `serror_rate` > 0.9 is rare outside DoS. Spearman $\rho \approx 0.49$ with the DoS label.
- **RST error features** $\leftrightarrow$ **Probe:** `rerror_rate` > 0.8 appears almost only in Probe rows ($\rho \approx 0.33$).
- **Login/root indicators** $\leftrightarrow$ **R2L:** non-zero `num_failed_logins` or `root_shell` are unique to R2L, with $p < 0.01$ in class-wise tests.
- **High `same_srv_rate`** $\leftrightarrow$ **Normal:** values near 1 correlate negatively with the attack flag (Spearman $\rho = -0.55$).

These features encode the *mechanistic signature* of each attack type (e.g., TCP handshake failure, repeated RSTs, login faults) making them valuable for both **explainable rules** and as input to **supervised/unsupervised models**.

2 Task 2: Shallow Anomaly Detection – Supervised vs. Unsupervised

Full indicator are present in table 2.

2.1 One-Class SVM fitted with normal data only

Considering that you are currently training only on normal data, which is a good estimate for the parameter ν? Which is the impact on training performance? Try both your estimate and the default value of ν.
Since we are training on normal data only, a sensible prior for ν lies between 0.01 % and $1-2$ %. Even a "clean" dataset may hide slight intra-class heterogeneity, so allowing a *tiny* fraction of slack points is prudent. Below we contrast the default setting ($\nu = 0.5$) with the best value found through a simple grid search aimed at maximising the F1 score.[1], which was even lower then what we expected, and it probably means that we could go even lower.

- **Default** $\nu = 0.5$**.** This choice implicitly assumes that up to half the training observations might be anomalous. On a dataset partition that should be wholly normal, the assumption is patently false: the model ends up *over rejecting* normal points and delivers mediocre validation metrics (accuracy 0.64, macro-F1 0.64).
- **Tuned** $\nu = 0.0005$**.** A grid search was performed over

  $$\nu \in \{0.0005,\, 0.001,\, 0.002,\, 0.005,\, 0.01\}, \quad \gamma \in \{10^{-4},\, 3{\times}10^{-4},\, 10^{-3},\, 3{\times}10^{-3},\, 10^{-2},\, 3{\times}10^{-2},\, 10^{-1},\, \texttt{scale}\}.$$

  For each (ν, γ) pair, a One-Class SVM with RBF kernel was fitted on the "normal only" training set, used to predict anomalies on the mixed validation set, and its F_1-score computed. The chosen ν (and γ) is the one yielding the highest validation F_1.

[1]The search ranged over $\nu \in \{0.5, 0.05, 0.01, 0.005, 0.001, 0.0005\}$; the optimum was 0.0005.

A much smaller ν—just 0.05 % of the training set—lets the decision boundary envelop almost all normal samples while still retaining enough flexibility to flag outliers. Validation performance rises around (accuracy 0.79, macro-F1 0.76 on the test set).

The related empirical cumulative distribution functions for "Normal" and "Anomalous" samples reveal a clear separation: almost all normal points have scores below about 0.1, while anomalies span from roughly 0.0 up to 0.45. This gap implies that choosing a detection threshold in the range $[0.1, 0.2]$ will recover most anomalies with minimal false alarms on normal data.

2.2 One-Class SVM on the *full* dataset

Now train the OC-SVM with both normal and anomalous data. Which model performs better, and why?

A grid search on the validation set selects an optimal value of $\nu \approx 0.33$, just above the true contamination rate. Introducing a small buffer (e.g. using $\nu + 0.002$ or $\nu + 0.01$) can absorb label noise and raises the F_1 score to about 0.73. However, this still underperforms the F_1=0.76 achieved by training the OC-SVM solely on clean traffic, indicating that the "contaminated" model is inherently weaker.

This is confirmed by the ECDF plot: the decision boundary becomes less sharply defined, causing the chosen threshold to capture a smaller fraction of anomalies than in the previous setting.

Why does performance drop?

- Support expansion. One-class SVMs learn the support of the *training* distribution. Mixing attacks with benign flows stretches that support, forcing the algorithm to accept many points that should be flagged. A large ν (30%) becomes necessary just to fit the hybrid cloud.
- Heterogeneous attacks. Our malicious set contains several families—probe scans, DoS bursts, infiltration attempts—each living on its own micro manifold. The model must envelop *all* of them, so the boundary grows ragged and loses discriminative power.
- Camouflaged traffic. Some probes imitate normal packets almost perfectly; their feature vectors sit close to legitimate ones. No single hyperplane can slice them apart without also chopping away big chunks of genuine traffic.

In short, the moment anomalies pollute the training pool, the one-class objective stops acting like an anomaly detector and starts behaving like a noisy density estimator. A fully supervised approach or, at the very least, keeping the training corpus clean—remains the surer path to high recall and precision.

2.3 OC–SVM Contamination Strategies

Plot the f1-macro score for each scenario of different percentage of anomalies in the training set. How is the increasing ratio of anomalies impacting the results

In the notebook, we experimented with three different data-splitting strategies to guard against any class-distribution bias in the malware samples. Here, for clarity, we present only the F_1 score curve obtained with the *stratified* split (Figure 1), which preserves the relative proportions of each attack type (DoS, Probe, R2L). Full results for all splitting methods are available in the notebook.

- More specifically, in the stratified case as well for the others, we can analyze this pattern: in the zero-contamination scenario the model only observes clean traffic and therefore learns a very tight decision boundary around normal points, which effectively excludes almost all future attacks while generating very few false alarms. Once a small fraction of attacks is included in training, the boundary must expand to accommodate them, causing the detector to misclassify many new attacks and sharply degrading overall performance. As more attacks are mixed into the training set, the model gradually readjusts: it learns to flex its boundary just enough to exclude the most extreme outliers, recovering some detection ability and reducing false positives. When the training set is evenly balanced between normal and attack traffic, the optimum strategy becomes to accept the majority and only reject the most extreme deviations this lowers false alarms and improves the overall score despite only moderate recall, this inflates the macro-F1, even though the attack class F1 remains dismal. Finally, when the model is trained on data that is entirely "normal" again (albeit a much larger inlier cloud), it treats most points including many future attacks as inliers; this yields very few false positives but also misses most attacks, paradoxically boosting the aggregated metric once more because normal traffic dominates the evaluation.

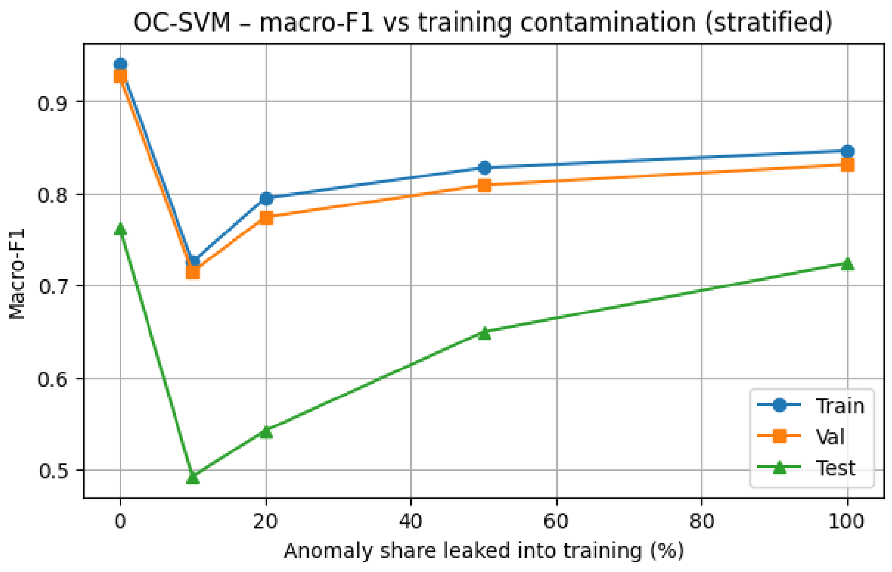


Figure 1: F1 score over different percentage of anomalies in the training dataset

Summary by contamination level:

- **0 % contamination:** All strategies use only normal data → identical macro-F1 (train/val/test = 0.7637/0.9277/0.7637).
- **10–20 % contamination:** Equal quota stratified is best. It compensates for R2L rarity by guaranteeing visibility during training.
- **50 % contamination:** Proportional stratified reflects true class distributions, yielding the best balance.
- **100 % contamination:** All strategies converge. Minor differences (e.g., random F1 = 0.7244 vs. proportional/equal = 0.7003) are due to rounding or hyperparameter noise.

2.4 One-Class SVM Model Robustness

Test the models trained with normal data only and 10 % of anomalous data on the test set. Q: Is the best-performing model in the training set also the best here? Compare the results of the best model in the test set with its results in the training set. Can it still spot the anomalies? Does it confuse normal data with anomalies?

Q: Is the best-performing model on training also the best on test?

- The **OC-SVM trained on normal-only data** is best overall, with macro-F1 = 0.94 (train) and 0.76 (test).
- The “mixed” model with 33 % anomalies scores 0.84 (train), 0.74 (test).
- The “10 % contaminated” model performs worst on test (macro-F1 = 0.60) despite scoring 0.80 on training.

Q: Can the best model still spot anomalies? Does it confuse normals?

- Yes, the “normal only” model achieves anomaly recall of 0.87 and precision of 0.80 on test.
- However, it misclassifies some normal samples (normal recall = 0.64).
- The “mixed” model is slightly worse overall, similar recall but lower precision on normal classes.
- The “10 % contaminated” model fails to generalize (anomaly recall = 0.43) and suffers low precision and accuracy.

Unsurprisingly, the OC-SVM trained solely on clean data still delivers the strongest test set performance. This makes sense: when trained only on normal samples, the model learns a tight boundary around the genuine data manifold. As soon as anomalies sneak into the training set even in small numbers and not perfectly separable the decision boundary distorts, degrading generalization and leading to more misclassifications of both normal and anomalous points, classifying only very far anomalies.

3 Task 3: Deep Anomaly Detection and Data Representation

We can insert Deep Learning into the game for either Anomaly Detection or Data Representation only.

3.1 Training and Validating the Autoencoder on Normal Data Only

We implemented a deep Autoencoder with a four-layer encoder (input $\rightarrow 128 \rightarrow 64 \rightarrow 32 \rightarrow$ *bottleneck*) mirrored by a symmetric decoder. The loss combines mean squared error on numerical features with cross-entropy on one-hot categorical blocks (via a custom `mixed_ae_loss`).

Hyperparameters (*bottleneck size*, learning rate, number of epochs) were chosen by grid search on an 80/20 split of the “only normal” data, using early stopping (patience = 10) on the validation loss. The best configuration was:

$$\text{bottleneck} = 16, \quad \text{lr} = 1 \times 10^{-3}, \quad \text{epochs} = 66$$

which yielded a minimum validation loss of 0.223 (after 44 retraining epochs on the train split).

3.2 Estimating the Reconstruction Error Threshold

With the final model trained exclusively on normal traffic, we compute per-sample reconstruction errors on the normal validation set. We then fit a robust threshold as:

$$\text{threshold} = \text{median}(\epsilon) \; + \; K \times \text{MAD}(\epsilon)$$

(Median Absolute Deviation) and we set it, as a fall back, to the 95th percentile of $\{\epsilon_i\}$ in case of MAD=0.
This thresholding strategy leverages robust statistics to adapt automatically to the spread of normal reconstruction errors, preventing a few extreme values from pulling the cutoff too far while still capturing genuine anomalies. This give us a threshold = 0.33196. We experiment this approach since it adapts to the inherent spread of ”normal” errors and avoids arbitrary percentile cutoffs.

3.3 Anomaly Detection via Reconstruction Error

Using the chosen threshold, we evaluated the model reconstruction error capabilities on:

1. **Validation (normal only)**–as above;
2. **Full training set** (mix of normals + anomalies);
3. **Test set** (unseen normals + anomalies).

Figure 2 shows the ECDF curves shifting rightwards from validation to training to test, reflecting progressively higher error mass from anomalies and unseen patterns. This, because the validation split contains only “seen” normal samples, the Autoencoder having been trained exclusively on normals—reconstructs them very accurately, so almost all errors lie at the extreme left of the ECDF. In the full training set, however, we have re-introduced anomalies alongside the normals. Those anomalous points were never seen during training, so the Autoencoder cannot reproduce them well: their reconstruction errors are much larger, stretching out the right tail of the ECDF and shifting the full train curve to the right of the validation curve. Finally, the test set includes both previously unseen normal variations and fresh anomalies. The combination of new normals and new attack patterns leads to even higher reconstruction errors on average and thus the test set ECDF sits furthest to the right, reflecting the heaviest tail of poor reconstructions.

Classification metrics:

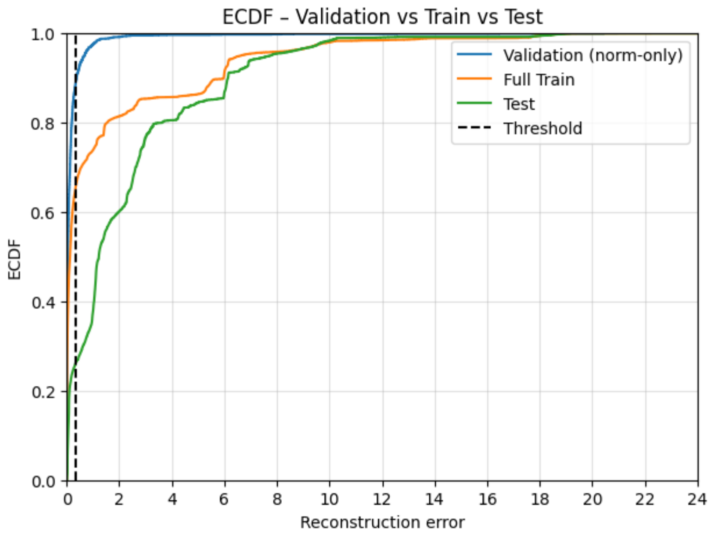


Figure 2: ECDF of reconstruction errors on validation (green), full train (blue) and test (red) sets.

- **Train (norm + anomaly)**: precision = 0.968, recall = 0.882, F_1 = 0.923 on normals; precision = 0.758, recall = 0.928, F_1 = 0.835 on anomalies; overall macro-F_1 = 0.879.
- **Test**: precision = 0.794, recall = 0.556, F_1 = 0.654 on normals; precision = 0.776, recall = 0.914, F_1 = 0.839 on anomalies; overall macro-F_1 = 0.747.

3.4 Auto-Encoder's bottleneck and OC-SVM

We first pass our normal only training data through the autoencoder's encoder to obtain a low dimensional "bottleneck" representation (here 16 D). Before fitting an OC-SVM, we standardize these embeddings to zero mean and unit variance so that the RBF kernel treats each dimension equally and isn't dominated by any one scale.

We then train a One-Class SVM on the *normalized* normal embeddings, performing a lightweight grid-search over ν and γ. The best model (here $\nu = 0.01$, $\gamma = 0.1$) is selected by maximizing F_1 on the reserved validation split. Finally, we apply this OC-SVM to three sets of embeddings, *all* training samples, the validation set, and the test set and we report precision, recall, and F_1 (along with confusion matrices).

While the OC-SVM on raw features achieves a test set anomaly F_1 of 0.84 (macro-$F_1 \approx 0.76$), the AE-OC-SVM only reaches 0.65 (macro-$F_1 \approx 0.62$). Compressing into a 16-dimensional bottleneck discards much of the discriminative structure needed to separate normal from anomalous patterns (low anomaly recall even in the train set). Although the embeddings capture a compact summary of "normal" traffic,in this case, they do not preserve the fine grained feature differences on which the OC-SVM relies, so overall detection performance degrades.

3.5 PCA and OC-SVM

Compare results with original OC-SVM and the OC-SVM trained using the Encoder embeddings. Describe the performance of the PCA-model w.r.t. the previous OC-SVMs.
The results are shown in the Table 1.

- The first thing we can notice is how PCA has better performance with respect to the model that uses the bottleneck of the encoder architecture, this because PCA captures directions of maximum variance, which may more directly separate anomalies if anomalies differ in variance from normal data. There is also the possibly that the representation that we are using is not appropriate for the data that we are using, in that case we should modify the architecture that we used to create the bottleneck.
  - One additional thing that we did in order to improve the performance of the PCA, is to apply standard scaling over the components, we did that to take into account components that have

an higher importance over the others (we have chosen 20 principal components), in this way we try to make them all appear equal.

This PCA - OC-SVM shows a slightly improvement respect to the one trained with raw feature, this maybe because in this case PCA captures a better lower dimension representation of the features, and this have permitted OC-SVM to capture more complex non-linear patterns, with respect to the raw features case.

4 Task 4: Unsupervised Anomaly Detection and Interpretation

This task focuses on anomaly detection without the use of labels. We applied clustering and visualization techniques (K-means and t-SNE) to identify and interpret anomalous patterns in the data, simulating a real world scenario where labels are not available.

4.1 K-means Clustering

After fitting K-means with 4 clusters on the full training dataset (including both normal and anomalous samples), we examined clusters to understand their quality.

- **How big are the clusters?** Regarding the cluster sizes, the distribution is quite uneven. The largest cluster is Cluster 1, containing 13589 instances. In contrast, Cluster 2 is much smaller, with only 10 instances. Clusters 0 and 3 are similar in size, with 1744 and 1604 instances respectively, but both are still considerably smaller than Cluster 1.
- **How are the attack labels distributed across the clusters? Are the clusters pure (i.e., they consist of only one attack label)?** Almost every cluster exhibits some impurity, containing a mixture of traffic types. A closer look reveals:
  - **Cluster 0**: Predominantly Probe traffic (54.15%), with the remainder split between DoS (18.15%) and Normal (27.70%).
  - **Cluster 1**: Largely Normal flows (85.18%), with small contributions from Probe (7.65%), DoS (5.97%) and R2L (1.19%).
  - **Cluster 2**: Completely composed of Normal samples (100%), albeit only 10 points—suggesting a highly specific Normal sub-pattern.
  - **Cluster 3**: Almost exclusively DoS attacks (93.20%), with negligible Normal (2.12%), Probe (4.62%) and R2L (0.06%) presence.

  Except for Cluster 2, no cluster is perfectly pure, indicating significant overlap in feature space across attack types.
- **How high is the silhouette per cluster? Is there any clusters with a lower silhouette value? If it is the case, what attack labels are present in these clusters?** As shown in Figure 3, the average silhouette value is approximately 0.47, which indicates a relatively poor overall clustering quality. More in detail:
  - Cluster 0 has the lowest value of 0.27, indicating it is the weakest and least cohesive cluster. Many of its points lie near decision boundaries: this is much less separated than the other clusters. Its mixed composition (54% Probe, 28% Normal, and 18% DoS attacks) creates ambiguity and internal confusion, as these traffic types are not well distinguished in feature space.
  - Cluster 1 has a moderate silhouette score of 0.46, but still lower than 0.5.
  - Cluster 2 is the smallest group and contains only Normal flows, yet it registers a score of 0.39. Although one would expect a pure cluster to score higher, this lower value likely reflects the internal heterogeneity among the Normal samples themselves or potentially mislabeled samples.
  - Cluster 3 has the highest silhouette score of 0.74, indicating that it is both well separated from other clusters and internally consistent. This cluster is largely composed of DoS attacks.

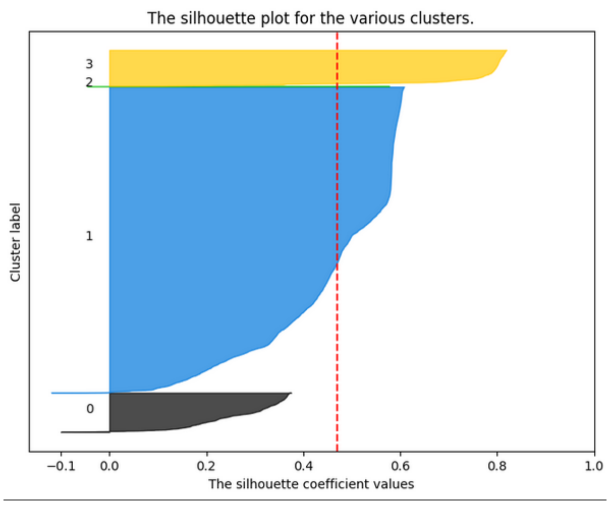


Figure 3: Silhouette plot for 4 clusters

4.2 t-SNE visualization

To visualize the clustering results, we used the t-SNE algorithm to project the high-dimensional data into a 2D space. We performed the visualization testing three different perplexity values: 10, 20, and 30. Each configuration produced noticeably different layouts, displayed in Figure 4:

- Perplexity of 10: with this configuration the clusters appear to be dense, but this level of density may be a problem for the cluster that have a smaller dimension (cluster 2), making it harder to separate from the others.
- Perplexity of 20: this configuration provided a good balance between local and global structure. All four clusters appeared as well separated and compact groups. Red is at the top, purple bottom-left, green in the center, and blue is distributed across the rest of the map, even green has a better separation (compared to the other run, but we know it may not be enough). Overlap between clusters is minimal, and the layout is visually coherent.
- Perplexity of 30: the visualization started becoming too smooth. The small green cluster at the center started merging with neighboring clusters, and the dominant blue cluster lost its boundaries and smeared into adjacent regions.

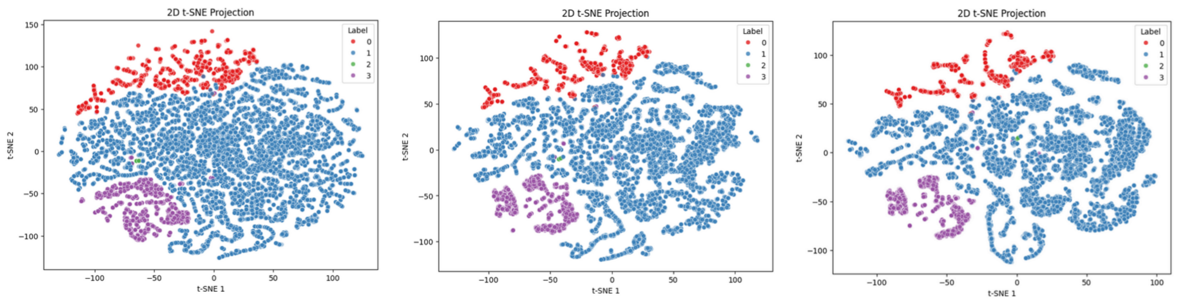


Figure 4: 2D t-SNE Projections with Perplexity Values respectively of 10, 20 and 30

Using the best perplexity value identified earlier (perplexity = 20), we applied t-SNE again, this time labeling the points based on their true attack labels instead of cluster assignments.
Can you find a difference between the two visualizations? What are the misinterpreted points? In the visualization using true attack labels, the overall structure remains similar to the cluster based one, with almost distinct groups occupying the same regions. This is shown in Figure 5. However, the boundaries appear less well defined in the visualization using true attack labels. While the general structure is preserved, the separation between some groups is more blurred, especially around the border

regions. This plot is consistent with the silhouette values computed earlier. Clusters with lower silhouette scores tend to show more label mixing in the t-SNE plot. For example, Cluster 0, which had the lowest silhouette score (0.27), is composed of a heterogeneous mix of labels: 54% Probe, 28% Normal, and 18% DoS. As expected, this cluster appears very scattered. In contrast, Cluster 3, which had the highest silhouette value (0.74), is composed almost entirely of DoS attacks: it remains compact and clearly separated from the others. R2L attacks are not clearly captured by any cluster and appear near the edges between clusters.

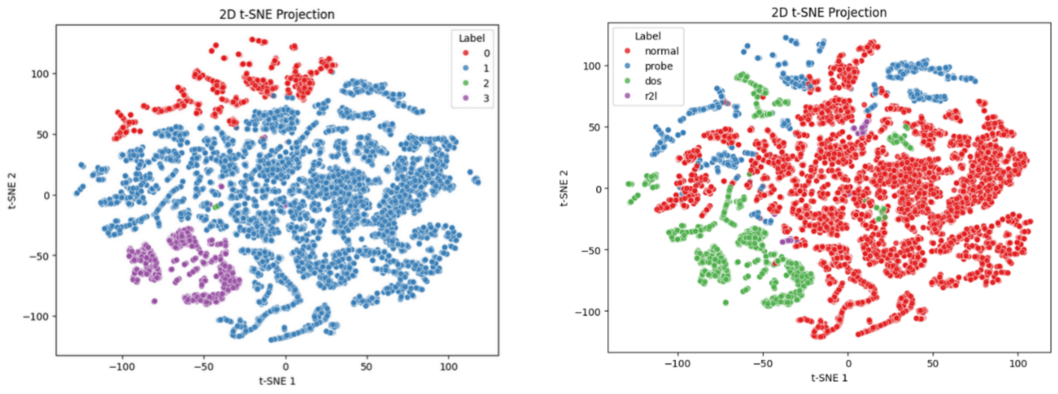


Figure 5: t-SNE Projection of clusters (on the left) compared to t-SNE Projection of true attack labels (on the right), with the best perplexity of 20

4.3 DB-Scan anomalies are anomalies?

To detect anomalies, we applied DB-Scan using the full training set (normal + anomalous samples). To estimate the min_points parameter, we used as a value the smallest K-Means cluster previously identified composed only of normal data. The number of points in this pure cluster (Cluster 2) was used as min_points (10). To determine the ϵ parameter, we used the k-distance elbow method. We computed the distance to each point's 10th nearest neighbor, sorted these distances, and plotted them. The elbow point in the curve (where the distance starts increasing rapidly) was visually selected as the optimal ϵ value (0.5).
Does the DB-Scan noise cluster (cluster -1) consist only of anomalous points (cross reference with real attack labels)? The results show that the DB-Scan noise cluster does not consist only of anomalous points. In fact, approximately 72% of the points labeled as noise are actually normal traffic. DB-Scan with a single global ϵ has erroneously labeled a large portion of normal data as noise, so it failed to isolate true anomalies.

Next, we considered the 10 largest clusters produced by DB-Scan. For each cluster, we analyzed how homogeneous it is by checking how labels (normal vs. anomalous) are distributed inside.
How are the labels distributed across these clusters, i.e. how are they composed of a single label? Several clusters were found to be composed entirely of a single class. Specifically, Cluster 4 and Cluster 10 are purely composed of DoS samples, while Cluster 1 contains only Probe samples. Clusters 3, 5, 14, and 19 are completely made up of Normal traffic. Other clusters are almost pure. For example, Cluster 0 contains 98.5% Normal points, Cluster 11 has 98.9% Normal, and Cluster 23 is composed of 95% DoS samples. Overall, DB-Scan manages to form several very clean and homogeneous clusters, especially for Normal and DoS traffic. This shows that, despite some limitations in detecting noise, the algorithm is capable of grouping together samples of the same class effectively.

To better understand the clustering result, we used t-SNE for 2D visualization with the best perplexity found earlier of 20:

- t-SNE with cluster labels: we plotted all training data using cluster IDs (limited to top 10 clusters) as colors.
- t-SNE with attack labels: we plotted the same data using the real labels (normal/anomalous).

Can you find a difference between the two visualizations? What are the misinterpreted points? By comparing both plots (Figure 6), we observed that the plot with the actual plot has more variability compared to the one of DB-SCAN.

- While the t-SNE visualization has better separation (taking account also that we are looking at just the top 10 of clusters, that may be insufficient), the actual distribution of point is more dense and it does not have well separated areas between normal and anomaly traffic.
- For sure the point that have been more **misinterpreted** are the one belonging to r2l, in the t-sne with the attack-labels we see how they do not have clear boundaries and are all sparse in the visualization, and so in the db-scan (due to their poor density) they do not appear and are present in different clusters.

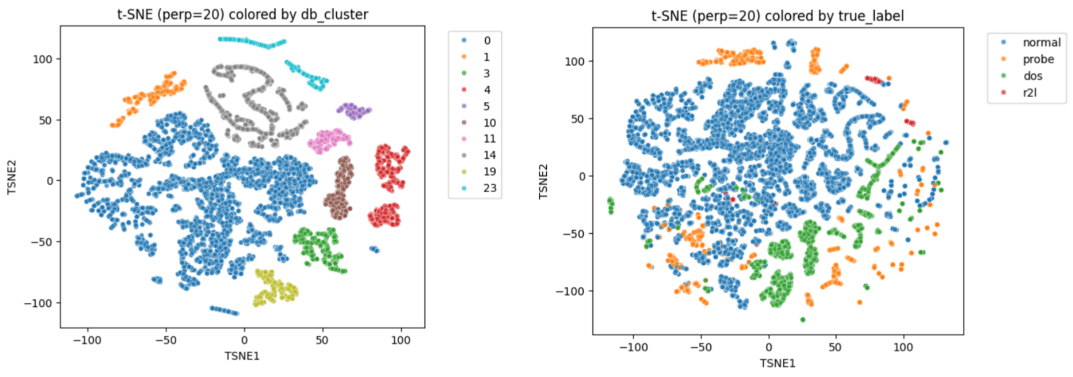


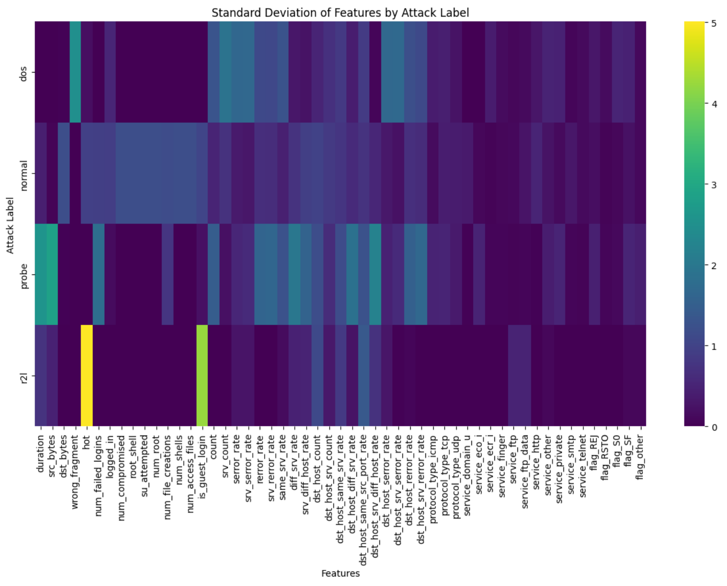
Figure 6: t-SNE Projection of DB-Scan top 10 clusters compared to t-SNE Projection of true attack labels

Why do you think that DB-Scan cannot separate the normal anomalous points well? DB-Scan struggles to clearly separate normal and anomalous traffic because of:

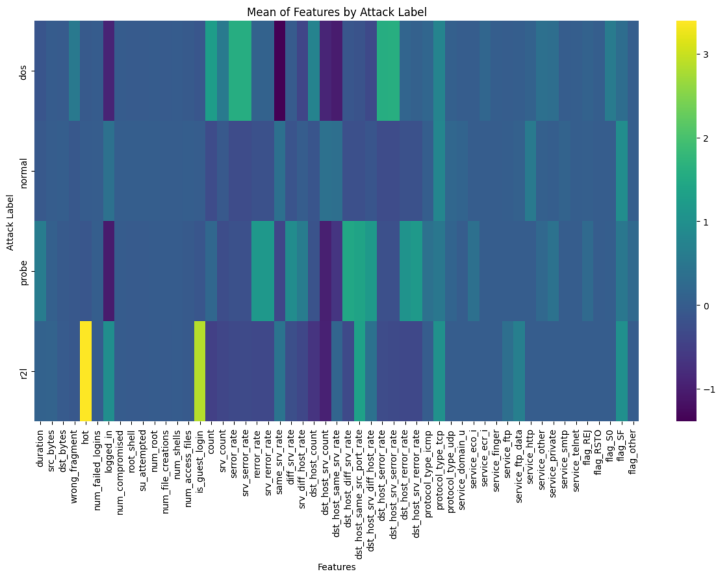
- **Single global density fails**: Normal behavior spans a wide range of local densities. Some normal bursts are as sparse as light attack probes, making it impossible for a single global ϵ to isolate all normal points without also including anomalies.
- **Heterogeneous cluster shapes**: Different attack types (e.g., DoS, Probe, R2L) form clusters with varying shapes and densities. DB-Scan uses a single ϵ, so it cannot adjust to these regional variations, resulting in either over or under clustering.
- **Mixed behavior**: Some benign samples resemble low intensity or stealthy attacks. As a result, DB-Scan may mislabel benign traffic as noise, or conversely embed some anomalous patterns inside normal clusters.

	OC-SVM on PCA				OC-SVM on AE			
Metric / Class	Precision	Recall	F1-score	Support	Precision	Recall	F1-score	Support
0	0.7145	0.7361	0.7251	2152	0.4992	0.7509	0.5997	2152
1	0.8426	0.8277	0.8351	3674	0.7885	0.5521	0.6494	3619
Accuracy	0.7939 (5826)				0.6262 (5771)			
Macro avg	0.7785	0.7819	0.7801		0.6439	0.6515	0.6246	
Weighted avg	0.7953	0.7939	0.7945		0.6806	0.6262	0.6309	

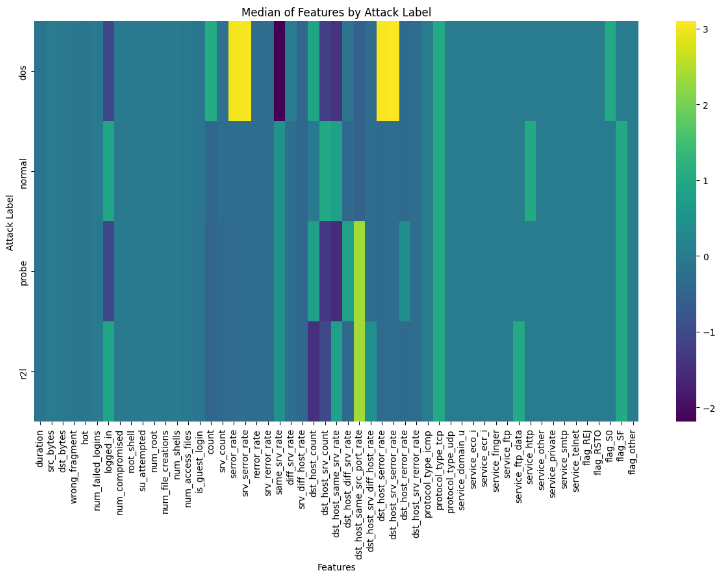
Table 1: Comparison of OC-SVM on PCA components vs. AE bottleneck (Test set results)



(a) Standard deviation grouped by label



(b) Mean features grouped by label



(c) Median features grouped by label

Figure 7: Feature statistics (standard deviation and mean on top; median below) grouped by label

Table 2: Performance comparison across OC-SVM training strategies

Model	**Split**	**Class**	**Precision**	**Recall**	**F1**	**Macro-F1**
Normal only	Training	Normal	0.95	0.99	0.97	0.94
		Anomaly	0.96	0.87	0.91	
	Validation	Normal	0.95	0.97	0.96	0.93
		Anomaly	0.93	0.87	0.90	
	Test	Normal	0.75	0.64	0.69	0.76
		Anomaly	0.80	0.88	0.84	
All data (33 %)	Training	Normal	0.93	0.87	0.90	0.84
		Anomaly	0.72	0.84	0.78	
	Validation	Normal	0.92	0.87	0.89	0.82
		Anomaly	0.71	0.81	0.76	
	Test	Normal	0.68	0.65	0.67	0.74
		Anomaly	0.80	0.82	0.81	
10 % contamination	Training	Normal	0.85	0.97	0.90	0.80
		Anomaly	0.88	0.57	0.69	
	Validation	Normal	0.84	0.97	0.90	0.78
		Anomaly	0.88	0.53	0.66	
	Test	Normal	0.44	0.76	0.56	0.60
		Anomaly	0.76	0.43	0.55	

Note: Test set accuracy — Normal only: 0.79, All data: 0.76, 10 % contaminated: 0.55.