Politecnico di Torino

AI and Cybersecurity

Project 4: Supervised vs Unsupervised Anomaly Detection

With shallow and deep learning methodologies

Angelo Barbera - s326432, Giorgia Moscato - s324647, Alessandro Genova - s330893

Table Of Contents

1 Task 1: Dataset Characterization and Preprocessing

1.1 Explore the dataset

Q: What are your dataset characteristics? How many categorical and numerical attributes do you have? How are your attack labels and binary_label distributed?
The train dataset consists of samples of network connections of three protocols: **TCP** (about 14000 samples), **UDP** (about 3000 samples) and **ICMP** (about 1800 samples). Each network connection is related to a service, mostly **HTTP**. There are **39** numerical attributes and **4** categorical attributes. The label and binary_label distribution are shown in Fig. 1 and Fig. 2.

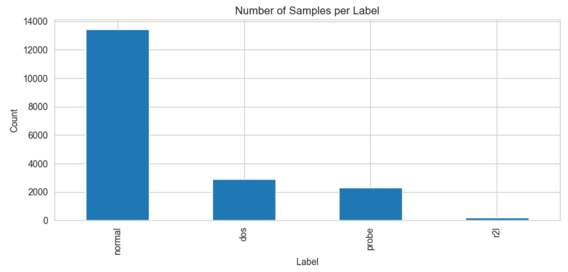


Figure 1: Label distribution

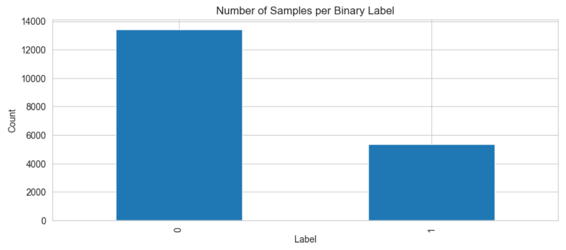


Figure 2: Binary label distribution

1.2 Preprocessing

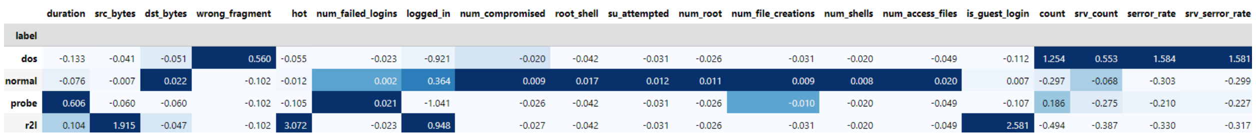
Q: How do you preprocess categorical and numerical data?
We dropped the features `urgent`, `num_outbound_cmds`, `is_host_login` because they are always equal to 0 and the feature `land` because it is always 0 with the exception of one sample. These features do not add any information since they have the same values for all the samples. We checked for the presence of duplicates, missing and negative values but there are none. We analysed the `flag` and `service` attributes, setting to `other` the values which are present in the samples with a percentage lower than 1% to reduce the dimensionality of the data when performing one-hot-encoding. We analysed the features using the Pearson correlation matrix to remove potential unnecessary features, but the results are better without dropping the high correlated features, so we decided to keep these features. We split the train dataset in **80%** for training and **20%** for validation using stratified sampling based on `label` attribute. Then, we processed the categorical attributes using one-hot-encoding, while we standardized the numerical attributes.

1.3 Study your data from a domain expert perspective

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

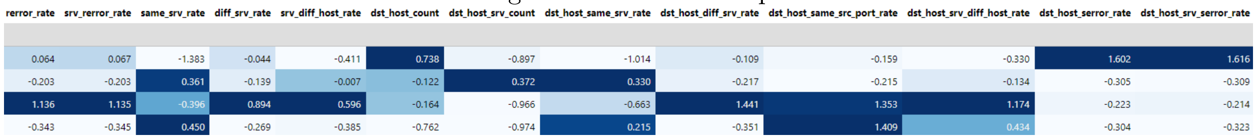
- **Denial of service:** the features with an high mean and standard deviation values are: `count`, `srv_count`, `serror_rate`, `srv_serror_rate`, `dst_host_serror_rate`, `dst_host_srv_serror_rate`, `flag S0` which shows an high number of connections and SYN errors which are common for SYN flood attack. Also the `wrong_fragment` feature is correlated with this attack due to its high standard deviation.
- **Probe:** the features with an high mean and standard deviation values are: `duration`, `rerror_rate`, `srv_rerror_rate`, `same_srv_rate`, `diff_srv_rate`, `srv_diff_host_rate`, `dst_host_diff_srv_rate`, `dst_host_same_src_port_rate`, `dst_host_srv_diff_host_rate`, `dst_host_rerror_rate`, `dst_host_srv_rerror_rate`, `flag REJ`, `flag RSTR`. Also the `count` and `num_failed_logins` features are correlated with this attack due to their high standard deviation compared to the other classes. The values of these features shows an high number of REJ errors which are typical for scanning attacks.
- **R2L:** the features with an high mean and standard deviation values are: `src_bytes`, `hot`, `is_guest_login`, `dst_host_same_src_port_rate`, `service_ftp`, `service_ftp_data`, `flag SF`. Also the `logged_in` and `protocol_type_tcp` features are correlated with this attack due to their high mean. The values of

these features shown an high number of successful login, an high use of ftp and tcp and an high number of data bytes from source to destination, which are commonly observed in attacks where the attacker sends packet to a machine to gain local access and privileges.



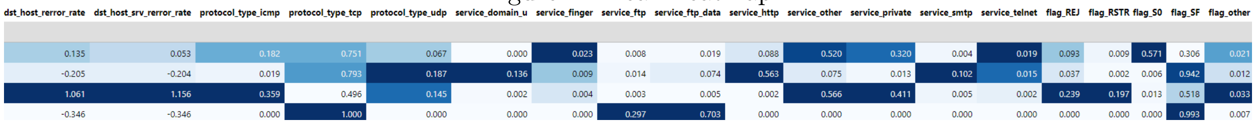
label	duration	src_bytes	dst_bytes	wrong_fragment	hot	num_failed_logins	logged_in	num_compromised	root_shell	su_attempted	num_root	num_file_creations	num_shells	num_access_files	is_guest_login	count	srv_count	serror_rate	srv_serror_rate
dos	-0.133	-0.041	-0.051	0.560	-0.055	-0.023	-0.921	-0.020	-0.042	-0.031	-0.026	-0.031	-0.020	-0.049	-0.112	1.254	0.553	1.584	1.581
normal	-0.076	-0.007	0.022	-0.102	-0.012	0.002	0.364	0.009	0.017	0.012	0.011	0.009	0.008	0.020	0.007	-0.297	-0.068	-0.303	-0.299
probe	0.606	-0.060	-0.060	-0.102	-0.105	0.021	-1.041	-0.026	-0.042	-0.031	-0.026	-0.010	-0.020	-0.049	-0.107	0.186	-0.275	-0.210	-0.227
r2l	0.104	1.915	-0.047	-0.102	3.072	-0.023	0.948	-0.027	-0.042	-0.031	-0.026	-0.031	-0.020	-0.049	2.581	-0.494	-0.387	-0.330	-0.317

Figure 3: Mean heatmap



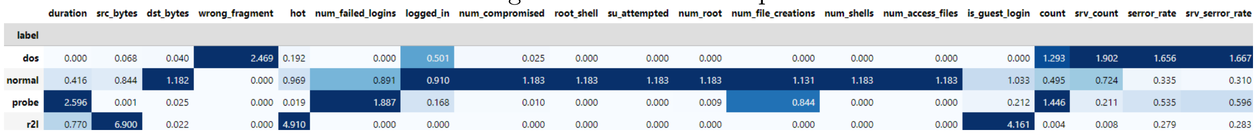
rerror_rate	srv_rerror_rate	same_srv_rate	diff_srv_rate	srv_diff_host_rate	dst_host_count	dst_host_srv_count	dst_host_same_srv_rate	dst_host_diff_srv_rate	dst_host_same_src_port_rate	dst_host_srv_diff_host_rate	dst_host_serror_rate	dst_host_srv_serror_rate
0.064	0.067	-1.383	-0.044	-0.411	0.738	-0.897	-1.014	-0.109	-0.159	-0.330	1.602	1.616
-0.203	-0.203	0.361	-0.139	-0.007	-0.122	0.372	0.330	-0.217	-0.215	-0.134	-0.305	-0.309
1.136	1.135	-0.396	0.894	0.596	-0.164	-0.966	-0.663	1.441	1.353	1.174	-0.223	-0.214
-0.343	-0.345	0.450	-0.269	-0.385	-0.762	-0.974	0.215	-0.351	1.409	0.434	-0.304	-0.323

Figure 4: Mean heatmap



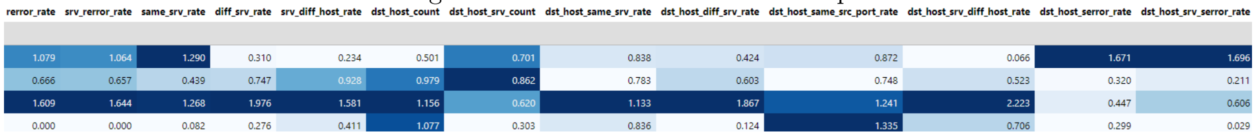
dst_host_rerror_rate	dst_host_srv_rerror_rate	protocol_type_icmp	protocol_type_tcp	protocol_type_udp	service_domain_u	service_finger	service_ftp	service_ftp_data	service_http	service_other	service_private	service_smtp	service_telnet	flag_REJ	flag_RSTR	flag_S0	flag_SF	flag_other
0.135	0.053	0.182	0.751	0.067	0.000	0.023	0.008	0.019	0.088	0.520	0.320	0.004	0.019	0.093	0.009	0.571	0.306	0.021
-0.205	-0.204	0.019	0.793	0.187	0.136	0.009	0.014	0.074	0.563	0.075	0.013	0.102	0.015	0.037	0.002	0.006	0.942	0.012
1.061	1.156	0.359	0.496	0.145	0.002	0.004	0.003	0.005	0.002	0.566	0.411	0.005	0.002	0.239	0.197	0.013	0.518	0.033
-0.346	-0.346	0.000	1.000	0.000	0.000	0.000	0.297	0.703	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.993	0.007

Figure 5: Mean heatmap



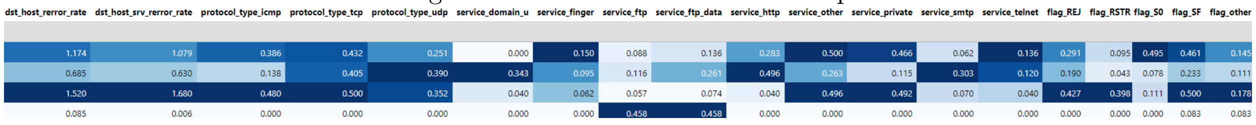
label	duration	src_bytes	dst_bytes	wrong_fragment	hot	num_failed_logins	logged_in	num_compromised	root_shell	su_attempted	num_root	num_file_creations	num_shells	num_access_files	is_guest_login	count	srv_count	serror_rate	srv_serror_rate
dos	0.000	0.068	0.040	2.469	0.192	0.000	0.501	0.025	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.293	1.902	1.656	1.667
normal	0.416	0.844	1.182	0.000	0.969	0.891	0.910	1.183	1.183	1.183	1.183	1.131	1.183	1.183	1.033	0.495	0.724	0.335	0.310
probe	2.596	0.001	0.025	0.000	0.019	1.887	0.168	0.010	0.000	0.000	0.009	0.844	0.000	0.000	0.212	1.446	0.211	0.535	0.596
r2l	0.770	6.900	0.022	0.000	4.910	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	4.161	0.004	0.008	0.279	0.283

Figure 6: Standard deviation heatmap



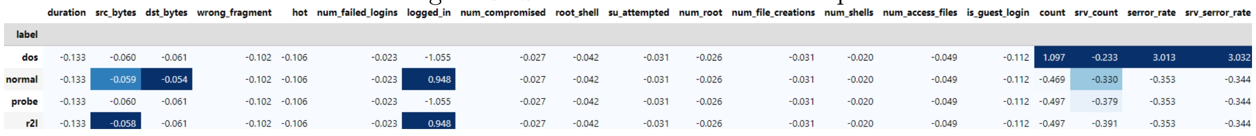
rerror_rate	srv_rerror_rate	same_srv_rate	diff_srv_rate	srv_diff_host_rate	dst_host_count	dst_host_srv_count	dst_host_same_srv_rate	dst_host_diff_srv_rate	dst_host_same_src_port_rate	dst_host_srv_diff_host_rate	dst_host_serror_rate	dst_host_srv_serror_rate
1.079	1.064	1.290	0.310	0.234	0.501	0.701	0.838	0.424	0.872	0.066	1.671	1.696
0.666	0.657	0.439	0.747	0.928	0.979	0.862	0.783	0.603	0.748	0.523	0.320	0.211
1.609	1.644	1.268	1.976	1.581	1.156	0.620	1.133	1.867	1.241	2.223	0.447	0.606
0.000	0.000	0.082	0.276	0.411	1.077	0.303	0.836	0.124	1.335	0.706	0.299	0.029

Figure 7: Standard deviation heatmap



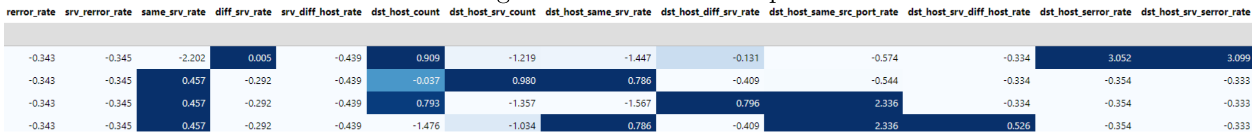
dst_host_rerror_rate	dst_host_srv_rerror_rate	protocol_type_icmp	protocol_type_tcp	protocol_type_udp	service_domain_u	service_finger	service_ftp	service_ftp_data	service_http	service_other	service_private	service_smtp	service_telnet	flag_REJ	flag_RSTR	flag_S0	flag_SF	flag_other
1.174	1.079	0.386	0.432	0.251	0.000	0.150	0.088	0.136	0.283	0.500	0.466	0.062	0.136	0.291	0.095	0.495	0.461	0.145
0.685	0.630	0.138	0.405	0.390	0.343	0.095	0.116	0.261	0.496	0.263	0.115	0.303	0.120	0.190	0.043	0.078	0.233	0.111
1.520	1.680	0.480	0.500	0.352	0.040	0.062	0.057	0.074	0.040	0.496	0.492	0.070	0.040	0.427	0.398	0.111	0.500	0.178
0.085	0.006	0.000	0.000	0.000	0.000	0.000	0.458	0.458	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.083	0.083

Figure 8: Standard deviation heatmap



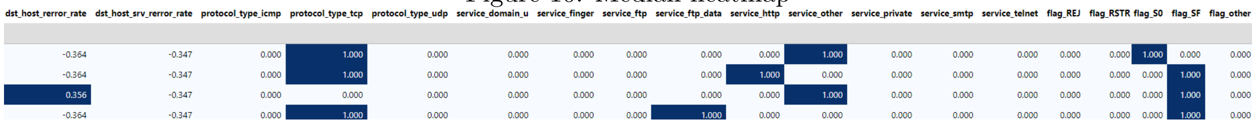
label	duration	src_bytes	dst_bytes	wrong_fragment	hot	num_failed_logins	logged_in	num_compromised	root_shell	su_attempted	num_root	num_file_creations	num_shells	num_access_files	is_guest_login	count	srv_count	serror_rate	srv_serror_rate
dos	-0.133	-0.060	-0.061	-0.102	-0.106	-0.023	-1.055	-0.027	-0.042	-0.031	-0.026	-0.031	-0.020	-0.049	-0.112	1.097	-0.233	3.013	3.032
normal	-0.133	-0.059	-0.054	-0.102	-0.106	-0.023	0.948	-0.027	-0.042	-0.031	-0.026	-0.031	-0.020	-0.049	-0.112	-0.469	-0.330	-0.353	-0.344
probe	-0.133	-0.060	-0.061	-0.102	-0.106	-0.023	-1.055	-0.027	-0.042	-0.031	-0.026	-0.031	-0.020	-0.049	-0.112	-0.497	-0.379	-0.353	-0.344
r2l	-0.133	-0.058	-0.061	-0.102	-0.106	-0.023	0.948	-0.027	-0.042	-0.031	-0.026	-0.031	-0.020	-0.049	-0.112	-0.497	-0.391	-0.353	-0.344

Figure 9: Median heatmap



rerror_rate	srv_rerror_rate	same_srv_rate	diff_srv_rate	srv_diff_host_rate	dst_host_count	dst_host_srv_count	dst_host_same_srv_rate	dst_host_diff_srv_rate	dst_host_same_src_port_rate	dst_host_srv_diff_host_rate	dst_host_serror_rate	dst_host_srv_serror_rate
-0.343	-0.345	-2.202	0.005	-0.439	0.909	-1.219	-1.447	-0.131	-0.574	-0.334	3.052	3.099
-0.343	-0.345	0.457	-0.292	-0.439	-0.037	0.980	0.786	-0.409	-0.544	-0.334	-0.354	-0.333
-0.343	-0.345	0.457	-0.292	-0.439	0.793	-1.357	-1.567	0.796	2.336	-0.334	-0.354	-0.333
-0.343	-0.345	0.457	-0.292	-0.439	-1.476	-1.034	0.786	-0.409	2.336	0.526	-0.354	-0.333

Figure 10: Median heatmap



dst_host_rerror_rate	dst_host_srv_rerror_rate	protocol_type_icmp	protocol_type_tcp	protocol_type_udp	service_domain_u	service_finger	service_ftp	service_ftp_data	service_http	service_other	service_private	service_smtp	service_telnet	flag_REJ	flag_RSTR	flag_S0	flag_SF	flag_other
-0.364	-0.347	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
-0.364	-0.347	0.000	1.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000
0.356	-0.347	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000
-0.364	-0.347	0.000	1.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000

Figure 11: Median heatmap

2 Task 2: Shallow Anomaly Detection - Supervised vs Unsupervised

2.1 One-Class SVM with Normal data only

Q: Considering that you are currently training only on normal data, which is a good estimate for the parameter `nu`? Which is the impact on the training performance?
Even if we are training one-class SVM only on normal data, 0 is not a good estimate for `nu` because normal data always contains errors, so there is always a percentage of anomalous data that must be considered. With the default value of `nu` (0.5) and `gamma` ('scale') we obtained an accuracy of **64%** on validation set.
We estimated the parameter `nu` trying different values and evaluating the F1 score on the validation set. With the estimated `nu` (0.001) and `gamma` (0.1) we obtained an accuracy of **95%** on validation set.
We can observe that using the default values the performance of the classifier is very poor, as shown by the low F1 and recall score. With the estimated value the performances greatly improve. The performance is shown in Tab. 1

Type	**Default `nu`**				**Estimated `nu`**			
	Precision	**Recall**	**F1**	**Support**	**Precision**	**Recall**	**F1**	**Support**
Normal	0.99	0.50	0.67	2690	0.95	0.97	0.96	2690
Anomaly	0.44	0.99	0.61	1077	0.93	0.88	0.90	1077
Macro avg	0.72	0.75	0.64	3767	0.94	0.93	0.93	3767
Weighted	0.83	0.64	0.65	3767	0.95	0.95	0.95	3767

Table 1: Validation set performance with normal data only

2.2 One-Class SVM with all data

We trained One-Class SVM with both normal and anomalous data, the percentage of anomalous samples is **28.58%**, hence `nu` is 0.28. We estimated the parameter `gamma` (1e-05) trying different values and evaluating the F1 score on the validation set.

Q: Which model performs better? Why do you think it is the case?
We achieved an accuracy of **87%** on validation set, the performance is shown in Tab. 2. This model perform slightly worse than the model trained only on normal data, as shown by the F1 score. This can be explained considering that with only normal data the model learns the boundaries of the normal samples, while using also anomalous data the model also need to consider that the training data contains anomalous samples that must not be included inside the boundaries of normal data, resulting in worst boundaries.

Type	**Precision**	**Recall**	**F1-score**	**Support**
Normal	0.92	0.90	0.91	2690
Anomaly	0.77	0.79	0.78	1077
Macro avg	0.84	0.85	0.84	3767
Weighted avg	0.87	0.87	0.87	3767

Table 2: Validation set performances with anomalies+normal data

2.3 One-Class SVM with normal traffic and some anomalies

We trained one-class SVM with an increasing subsample of the anomalous classes (0%, 10%, 20%, 50%, 100%) of anomalies. The `nu` parameter is estimated as the ratio between the anomalous samples and the total number of samples. The `gamma` parameter is equal to **0.1** in the case of only normal data and **1e-5** in the other cases.

Q: Plot the f1-macro score for each scenario. How is the increasing ratio of anomalies impacting the results?
As shown in Fig. 12, the macro F1 score reaches its maximum value when one-class SVM is trained using only normal data, it reaches the lowest value for anomaly percentage of **10%** then it increases and remains stable for anomaly percentage of **50%** and **100%**. This plot shows that the best model is obtained when the data boundaries are learned using only normal data than including also anomalous ones in the train set.

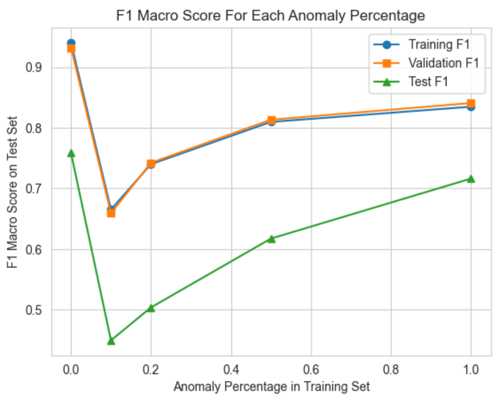


Figure 12: F1-macro plot for each anomaly percentage

2.4 One-Class SVM model robustness

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?
The best model in the training set is the model trained with normal data only, with an accuracy of **95%** and the performance is shown in Tab. 3. The same model is also the best one in the test set, with an accuracy of **78%**. With respect to the results in the training set all the metrics are lower, however the macro-f1 is **76%** so the performance of the model are good. Validation set performance (estimated nu values) is already reported in Tab. 1 .

Type	Training set				Test set			
	Precision	Recall	F1	Support	Precision	Recall	F1	Support
Normal	0.95	0.99	0.97	10758	0.74	0.64	0.68	2152
Anomaly	0.96	0.87	0.91	4306	0.80	0.87	0.83	3674
Macro avg	0.96	0.93	0.94	15064	0.77	0.75	0.76	5826
Weighted avg	0.95	0.95	0.95	15064	0.78	0.78	0.78	5826

Table 3: Training and test sets performance with normal data only

3 Task 3: Deep Anomaly Detection and Data Representation

The core premise is that an Autoencoder trained on normal network traffic yields low reconstruction errors for normal patterns and higher errors for unseen anomalies. The Autoencoder will employ a mixed loss function, combining Mean Squared Error for numerical features and Cross-Entropy for categorical ones, with a threshold applied to the reconstruction error to identify outliers.

3.1 Estimate the Reconstruction Error Threshold

Q: How did you choose the threshold? Which is its value?
We determine an effective reconstruction error threshold for anomaly detection that involved the two following approaches:

- **Initial Estimation with Normal Validation Data**: Initially, we estimated a threshold based on the **normal-only** validation set, selecting the 95th percentile of reconstruction errors as **0.4012**. This choice was motivated by the fact that the Autoencoder was trained exclusively on normal data, prompting us to set a threshold near the maximum reconstruction error for normal patterns to ensure it reflects the upper limit of expected normal behavior.

- **Evaluation and Refinement with Entire Validation Data**: To refine the initial threshold, we tested various percentiles of reconstruction errors on the **entire** validation set (including normal and anomalous data). We plotted metrics, as shown in Fig. 13, with one line tracking a custom score averaging Precision (Normal), avoiding misclassified anomalies as benign, and Recall (Anomaly), ratio of detected anomalies to actuals. Another line shows the average F1-score that we can consider to avoid excessive false positives. With the optimal threshold set at the 70th percentile (**0.3090**), the mean of the two scores at that point is 0.9112. This moderate threshold minimizes false alarms and benign errors.

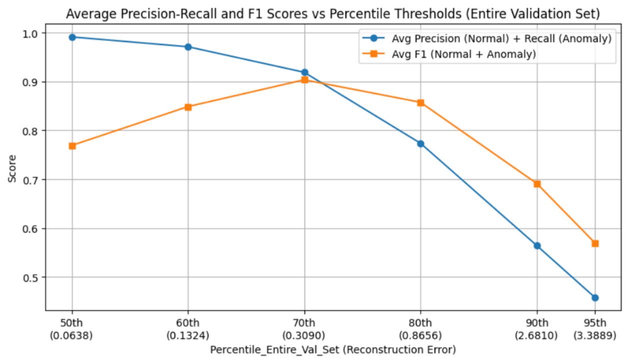


Figure 13: Average F1, Recall, and Precision vs. Percentile Thresholds

The initial 95th percentile (**0.4012**) from the normal-only set was plausible but enhanced by this analysis. So we have set **0.3090** (70th percentile of entire validation set reconstruction errors) as final threshold.

3.2 Anomaly Detection with reconstruction error

Q: Plot and report the ECDF of the reconstruction errors for each point i) in the validation set; ii) in the full training set; iii) in the test set. Why are the reconstruction errors higher on the full training set than on the validation one? And why are the reconstruction errors in the test set even higher?
The ECDFs of reconstruction errors, shown in Fig. 14, Fig. 15 and Fig. 16, reveal the following mean values:

- **Validation Set (only normal data)** mean error = 0.2081.
- **Full Training Set (normal + anomalies)** mean error = 0.7250.
- **Test Set (normal + anomalies)** mean error = 1.9379.

Because the validation set in this scenario contains only normal traffic, and the AE was trained only on normal patterns, it produces the lowest reconstruction errors. The full training set includes anomalous samples, which the AE cannot reconstruct well, so its overall reconstruction errors rise. The test set shows the highest reconstruction errors both because it not only contains anomalies, but also because its traffic distribution differs from the training domain, introducing novel patterns that further challenge the autoencoder's

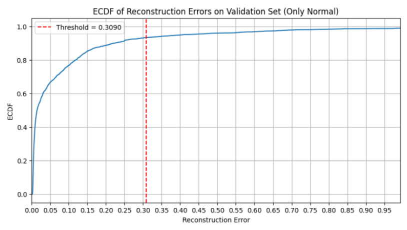


Figure 14: ECDF Validation Set

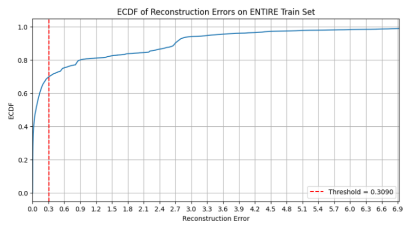


Figure 15: ECDF Full Training Set

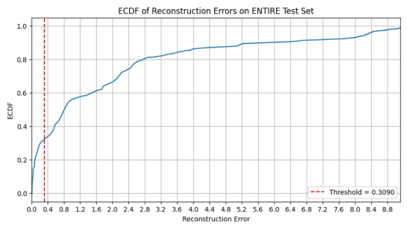


Figure 16: ECDF Test Set

reconstruction.
The Tab. 4 shows the classification performance on the test set. The accuracy is **75%**. Using raw reconstruction error as the anomaly score (thresholded) generates reasonable detection.

Class	Precision	Recall	F1-score	Support
Normal	0.69	0.60	0.64	2152
Anomaly	0.78	0.84	0.81	3674
Macro Avg	0.74	0.72	0.73	5826
Weighted Avg	0.75	0.75	0.75	5826

Table 4: Classification performance on test set using reconstruction error as anomaly score

3.3 Auto-Encoder's bottleneck and OC-SVM

The bottleneck embeddings extracted by the AE's encoder represent an abstract, compressed feature set, potentially enhancing the separability of normal and anomalous patterns. A One-Class SVM with parameters `kernel='rbf'`, `gamma=0.1`, and `nu=0.001` (optimized from Task 2 on normal data) was trained on these embeddings from the normal training set. The test set performance is shown in Tab. 5:

Class	Precision	Recall	F1-score	Support
Normal	0.67	0.67	0.67	2152
Anomaly	0.81	0.80	0.80	3674
Macro Avg	0.74	0.74	0.74	5826
Weighted Avg	0.75	0.75	0.75	5826

Table 5: Classification performance on test set using OC-SVM on Autoencoder bottleneck embeddings

With an accuracy of **75%** and a macro-average F1-score of **74%**, the OC-SVM on bottleneck embeddings delivers performance comparable to using reconstruction error on the same embeddings.

3.4 PCA and OC-SVM

PCA was applied to the normal training data, retaining **21** components to explain approximately **95% of the variance**, determined via an elbow point analysis of explained variance shown in Fig. 17. An OC-SVM with parameters `kernel='rbf'`, `gamma=0.1`, and `nu=0.001` (params optimized for normal data only training like we saw in Task 2) was then trained on these PCA-transformed features. The test set performance is shown in Tab. 6. With an accuracy of **77%** and a macro-average F1-score of **76%**, the PCA+OC-SVM pipeline slightly outperforms the autoencoder-based methods. In other words, on this dataset, a shallow method like simple linear projection retaining 95% of the variance combined with a well-tuned one-class SVM, can match, or even exceed, the more complex deep non-linear autoencoder.

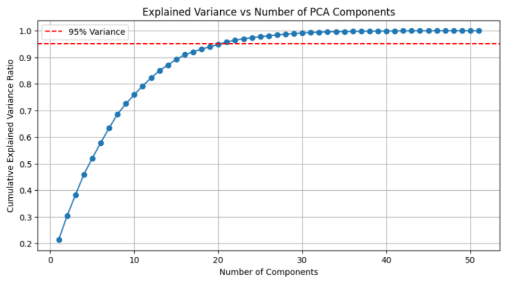


Figure 17: Explained variance with respect to the number of PCA components

Class	Precision	Recall	F1-score	Support
Normal	0.66	0.77	0.71	2152
Anomaly	0.85	0.77	0.81	3674
Macro Avg	0.76	0.77	0.76	5826
Weighted Avg	0.78	0.77	0.77	5826

Table 6: Classification performance on test set using OC-SVM on PCA components

3.5 Performance Analysis and Comparison

Q: Compare results with the best original OC-SVM. Describe the performance and where the model performs better or worst w.r.t. the original OC-SVM.
Q: 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.
Two additional metrics are reported in Tab. 8 to offer a clearer evaluation: the average between precision on normal data and recall on anomalies (**Avg Prec+Rec**) reflects a model's ability to detect true threats while minimizing false negatives, a key concern in cybersecurity contexts. The **Avg F1** score serves as a balanced overall performance indicator: a high Avg Prec+Rec but low Avg F1 would imply too many anomaly false positives, which can overwhelm monitoring systems or reduce trust in the alerts. Including both helps assess the trade-off between sensitivity to anomalies and general classification balance.
The original OC-SVM on raw features is the best architecture. The AE reconstruction error approach tends to misclassify more normal traffic, while the AE bottleneck combined with OC-SVM shows slightly better balance but does not surpass the baseline. PCA+OC-SVM gives decent results. So we've seen that even shallow methods without deep non-linear models can be surprisingly effective on this dataset, possibly due to clear statistical separability or low intrinsic complexity of the data. The classification performance of all the architectures is shown in Tab. 7.

Class	Precision	Recall	F1-score	Support
OC-SVM on Raw Features				
Normal	0.74	0.64	0.68	2152
Anomaly	0.80	0.87	0.83	3674
AE Reconstruction Error				
Normal	0.69	0.60	0.64	2152
Anomaly	0.78	0.84	0.81	3674
AE Bottleneck + OC-SVM				
Normal	0.67	0.67	0.67	2152
Anomaly	0.81	0.80	0.80	3674
PCA + OC-SVM				
Normal	0.66	0.77	0.71	2152
Anomaly	0.85	0.77	0.81	3674

Table 7: Classification performance on the test set

Method	Accuracy	Avg Prec+Rec	Avg F1
OC-SVM on Raw Features	0.78	0.80	0.75
AE Reconstruction Error	0.75	0.76	0.73
AE Bottleneck + OC-SVM	0.75	0.73	0.74
PCA + OC-SVM	0.77	0.72	0.76

Table 8: Classification performance on the test set

4 Task 4: Unsupervised Anomaly Detection and Interpretation

4.1 K-means cluster interpretation

Q: How big are the clusters? How are the attack labels distributed across the clusters? Are the clusters pure (i.e., they consist of only one attack label)?
The cluster sizes are as follows: **cluster 0** has 12110 points, **cluster 1** has 1522 points, **cluster 2** has 7 points, and **cluster 3** has 1425 points.

Cluster	dos	normal	probe	r2l
0	0.06	0.85	0.08	0.01
1	0.17	0.29	0.54	0.00
2	0.00	1.00	0.00	0.00
3	0.93	0.02	0.05	0.00

Table 9: Normalized label distribution per cluster

From Tab. 9, we see that: **Cluster 0** is mainly normal, with minor proportions of probe and DoS; **Cluster 1** more than half of the samples are probe attacks, the remaining points are split between normal and DoS, while r2l is not present; **Cluster 2** is pure, contains only normal traffic; **Cluster 3** is highly composed of DoS attacks with minimal presence of normal and probe traffic.

Q: 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?
We computed both the overall average silhouette score (≈ **0.46**) and the per cluster scores.

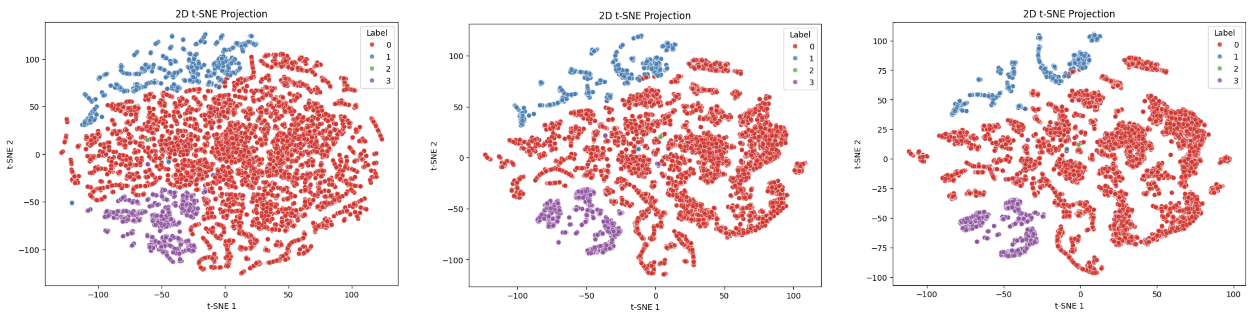
Cluster	Silhouette Score
0	0.45
1	0.27
2	0.48
3	0.75

Table 10: Silhouette scores per cluster

From Tab. 10 we can see that **cluster 1** is the one with the lowest silhouette, this means that its points have low cohesion (the cluster is not very dense around a central point) and poor separation (some points in this cluster lie very close to points in other clusters).

We can see from Tab. 9 that the cluster 1 is composed of samples of three different classes, so the points aren't grouped around a single behaviour. In other words since Probe, Normal, and DoS samples lie close to each other, K-means can't separate well them, which explains the low silhouette score.

Use the t-SNE algorithm to obtain a 2D visualization of your points.
We plot t-SNE using all training data with three different values of perplexity (10, 25 and 40).



(a) perplexity = 10 (b) perplexity = 25 (c) perplexity = 40

Figure 18: t-SNE visualization

From the plots shown in Fig. 18, we observed that with:

- **Perplexity = 10**: many of the points are mixed together and the clusters are not well separated. The red points spread across the plot and overlap points belonging to other clusters, reducing class separation. The reason is that a low perplexity value, like 10, makes t-SNE focus too much on very local relationships between points, it does not capture the global structure well, so the clusters look unclear.
- **Perplexity = 25**: when we increase perplexity, t-SNE considers a broader neighbourhood for each point and this usually gives a better balance between local and global structure. Indeed this value of perplexity is a good compromise, gives the clearest visualization, reduces intra-cluster distance while increasing separation between clusters.
- **Perplexity = 40**: points are packed tightly and overlap. For example, some purple points near the centre are hidden under red points, making the purple cluster look merged with the red.

After finding that perplexity = 25 is the optimal value, we plotted t-SNE on all training data with the attack label.

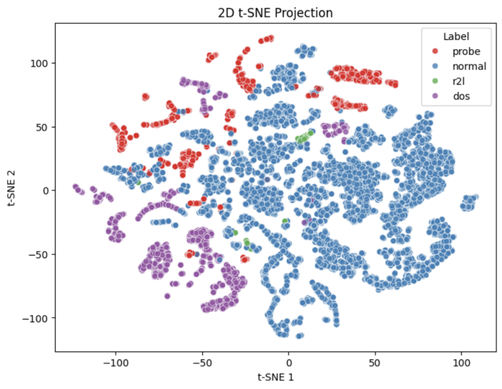


Figure 19: t-SNE with the attack label

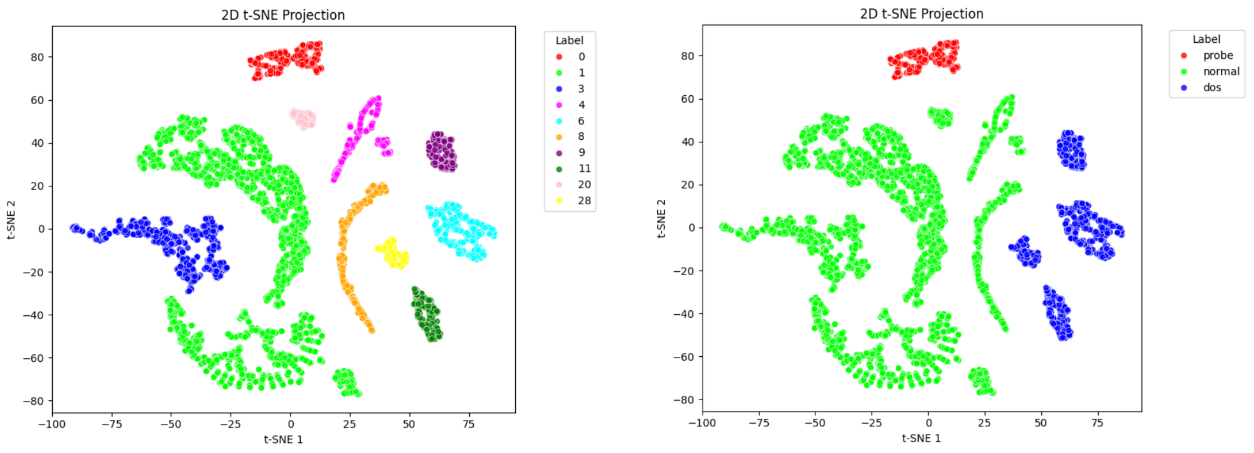
Q: Can you find a difference between the two visualizations? What are the misinterpreted points?
Yes, if we compare the plot in Fig. 18b with the one in Fig. 19 we can see that K-Means misclassified several points. The largest cluster (cluster 0) is dominated by normal traffic, but the algorithm misclassified a significant number of DoS and probe attacks into this cluster, incorrectly grouping them with the legitimate data. K-Means successfully grouped most DoS attacks into cluster 3, but it misclassified other DoS points that were located closer to cluster 0 and 1. The r2l (green) points are very few and scattered throughout the projection, making them difficult for K-Means to cluster successfully.

4.2 DB-Scan anomalies are anomalies?

Q: Does the DB-Scan noise cluster (cluster -1) consist only of anomalous points (cross-reference with real attack labels)?
No, the DBSCAN noise cluster is not pure anomaly, 70% of the points DBSCAN labels as noise are actually normal according to the ground truth. This misclassification of benign records as noise indicates DBSCAN is not reliable as an anomaly detector in this case.

Q: Consider the 10 largest clusters by size - DO NOT consider cluster -1. How are the labels distributed across these clusters, i.e. how are they composed of a single label?
The 10 largest clusters are pure, each of these clusters is composed entirely of a single label.

Q: Comparing t-SNE Visualizations. Can you find a difference between the two visualizations? What are the misinterpreted points?
Looking at Fig. 20 we can assess that DBSCAN incorrectly divided what should be a single group of normal data into five different clusters (1, 3, 4, 8 and 20). A similar issue occurred with the DoS attacks, instead of grouping all the DoS points into a single cluster, DBSCAN fragmented them into four separate clusters (6, 9, 11 and 28). The issue with the normal and DoS data is that they have different densities throughout their distribution and with a single fixed ϵ value, because of the lower-density zones, DBSCAN separated the data into multiple clusters.
Despite the issues of over-clustering, the isolated cluster that DBSCAN labeled as Cluster 0 is a perfect match for the red group representing the probe attacks in the plot with the original labels. This demonstrates that the algorithm in this case identified a dense and distinct group of anomalies and separated them from the rest of the data.



(a) t-SNE with DBSCAN cluster IDs (b) t-SNE with the original Attack Labels

Figure 20: t-SNE visualization

Q: Why do you think that DB-Scan cannot separate the normal anomalous points well?
There is the curse of dimensionality problem. High dimensionality makes it difficult for DBSCAN to form meaningful clusters because it causes the distances between all points to become very similar. This makes the concept of a 'dense neighbourhood', which the algorithm relies on, unstable. DBSCAN's behavior depends on the choice of epsilon and it's hard in this case to choose a single epsilon value that can accurately separate dense groups from sparse ones.