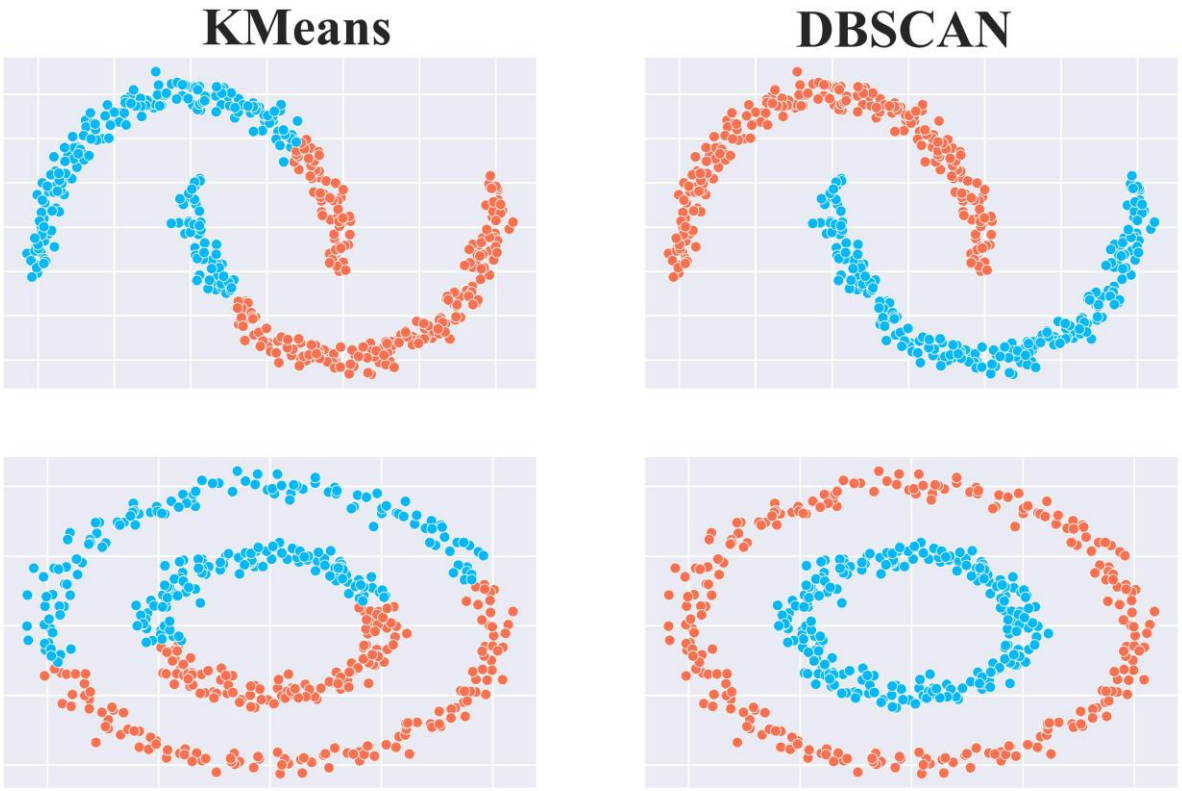
KMEANS, DBSCAN CLUSTERING



MUNAL BARAILI
900006725

ABSTRACT

This project applies unsupervised clustering to a cleaned numeric subset of the KDD Cup ’99 dataset to detect anomalous (attack) traffic without using labels during training. After fixing feature inconsistencies, removing multicollinearity, and scaling, K-Means with K = 2 produced the most meaningful separation, achieving an 84% attack-detection rate at a 6% false-alarm rate. DBSCAN, in contrast, struggled in the high-dimensional feature space. Visual analysis using PCA, t-SNE, and radar plots confirms that differences in byte counts, and host/service request behaviour drive the cluster separation.

Contents

Introduction

Network environments rarely provide labelled data in real time, so anomaly detection often relies on unsupervised learning. This project evaluates how well K-Means and DBSCAN can discover underlying structure in network flows and separate normal from attack behaviour without supervision. Labelled evaluation is only used afterwards to benchmark the cluster quality.

Objectives

Prepare a fully numeric, cleaned dataset for clustering.

Apply K-Means and DBSCAN to identify natural traffic groupings and compare the model performance.

Compare cluster assignments with true labels to estimate detection performance

Analyse feature differences across clusters to understand behavioural drivers

Dataset, Pre-processing

Data Overview

Activity columns consist of normal or DOS or PROBE or R2L or U2R but for this project, Activity is normal, and rest mapped as attack.

In overview, the dataset consists of high dimensional with initial feature count to 41.

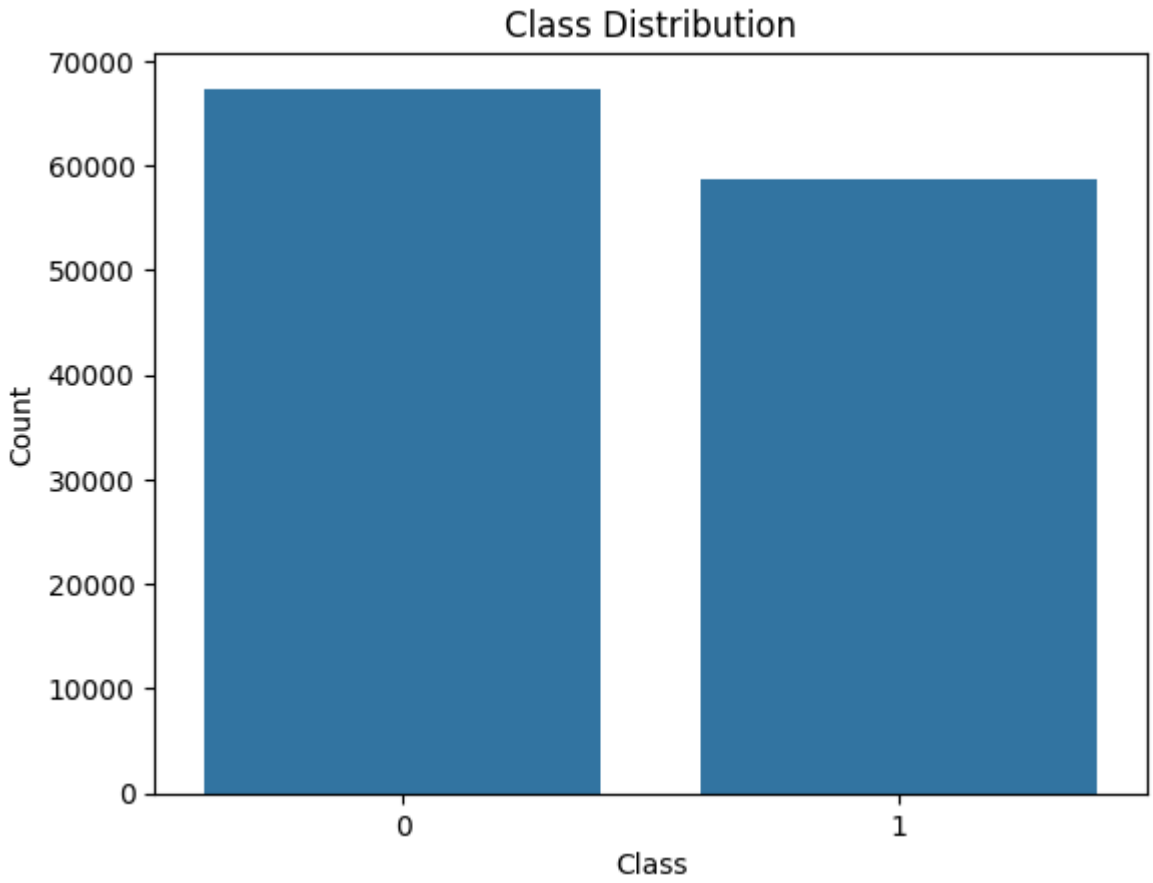
Rows with a binary label (0 = normal, 1 = attack).

BINARY CLASSIFICATION (NORMAL (0), REST (1))

```
df['class'] = df['attack'].apply(lambda x: 'normal' if x == 'normal' else 'attack')
```

```
# map class to 0 and 1
df['class'] = df['class'].map({'normal': 0, 'attack': 1})
```

Dataset appears balanced, but balance is irrelevant for clustering — models rely on spatial structure, not label counts.



0 = Normal and 1 = Attack

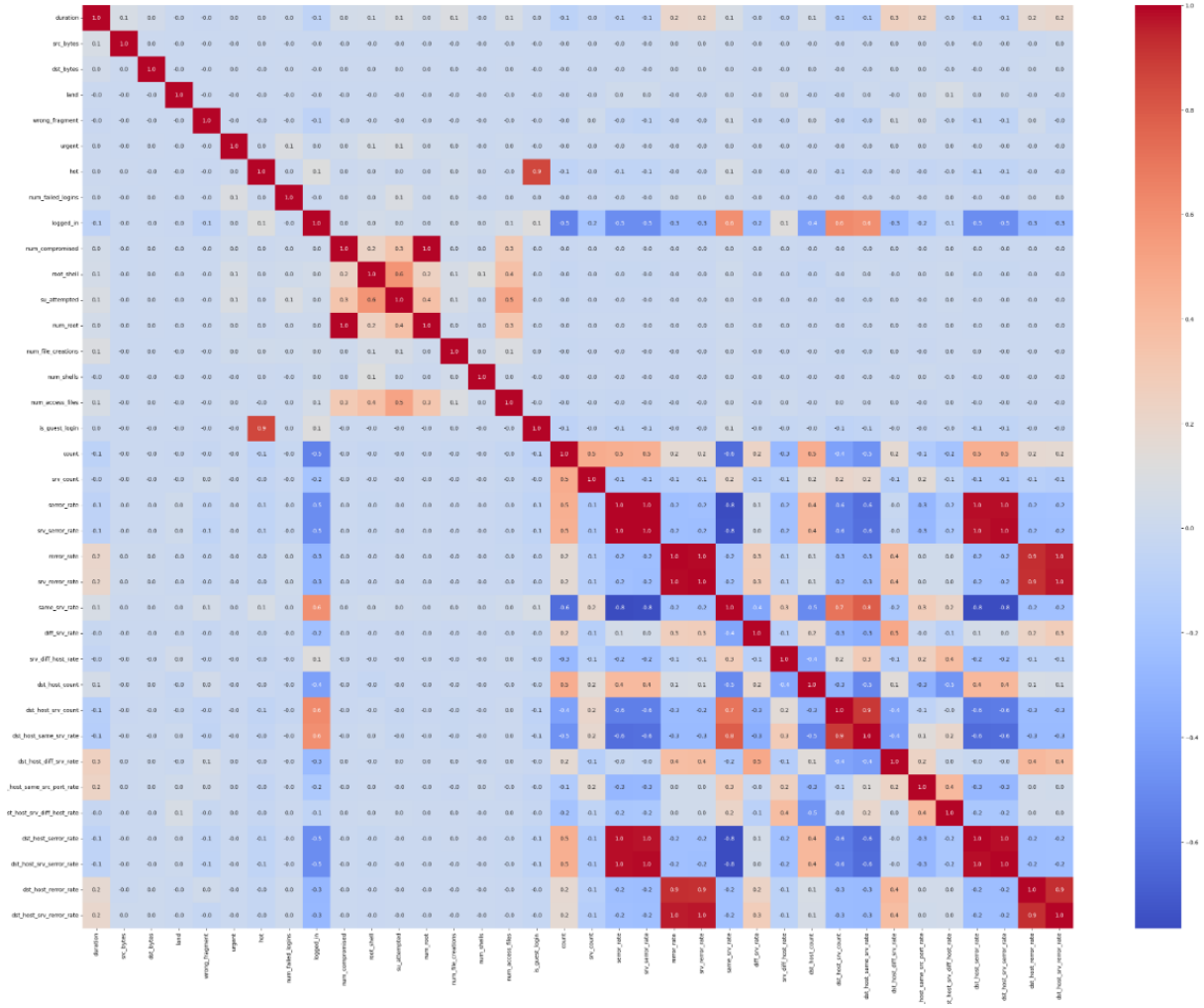
Processing Steps

Converted all features to numeric; fixed su_attempted (0,1,2 → binary)

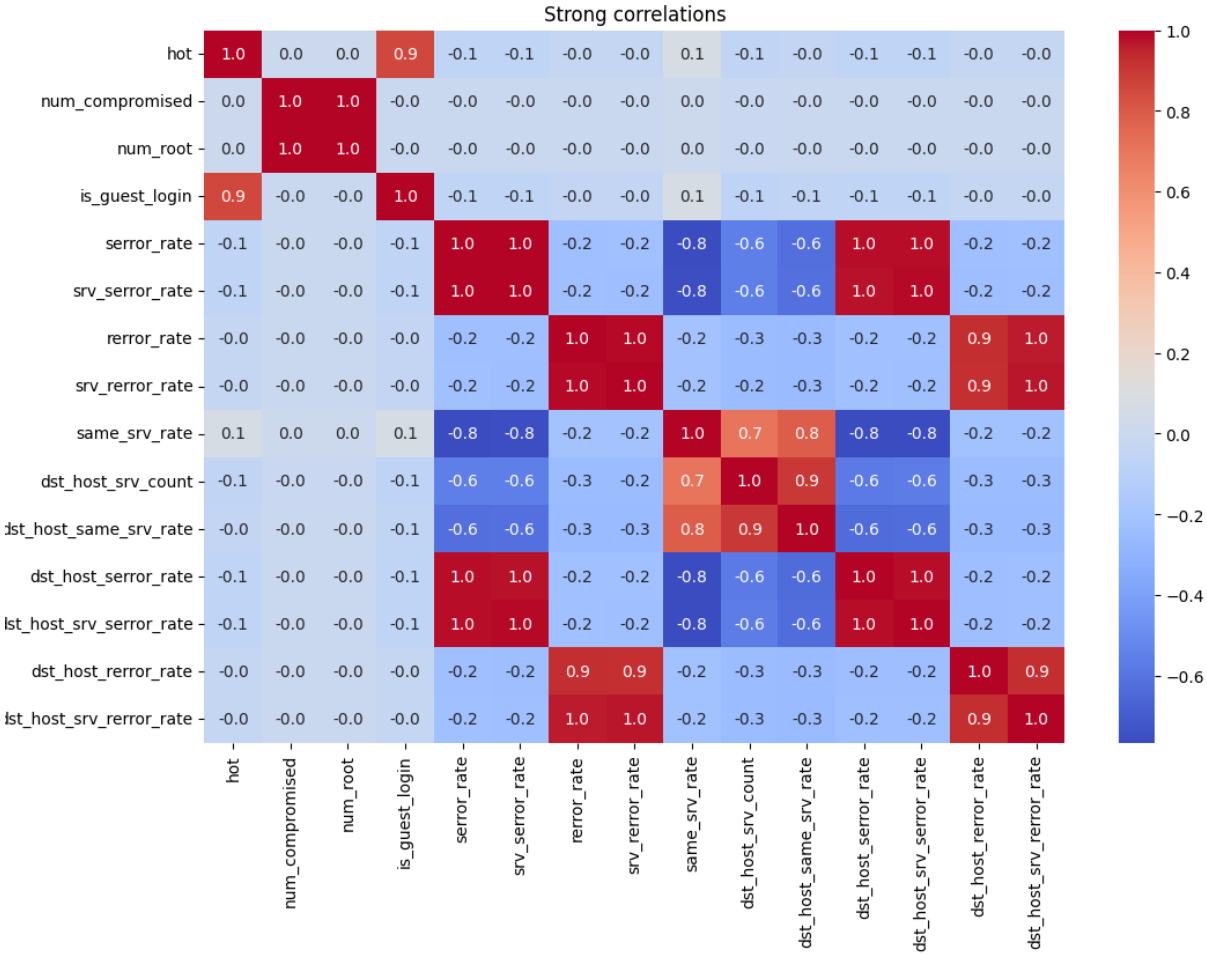
su_attempted	count
0	125893
1	80

dtype: int64

Removed features with correlation > 0.7 to address multicollinearity



Initial Features correlation map.



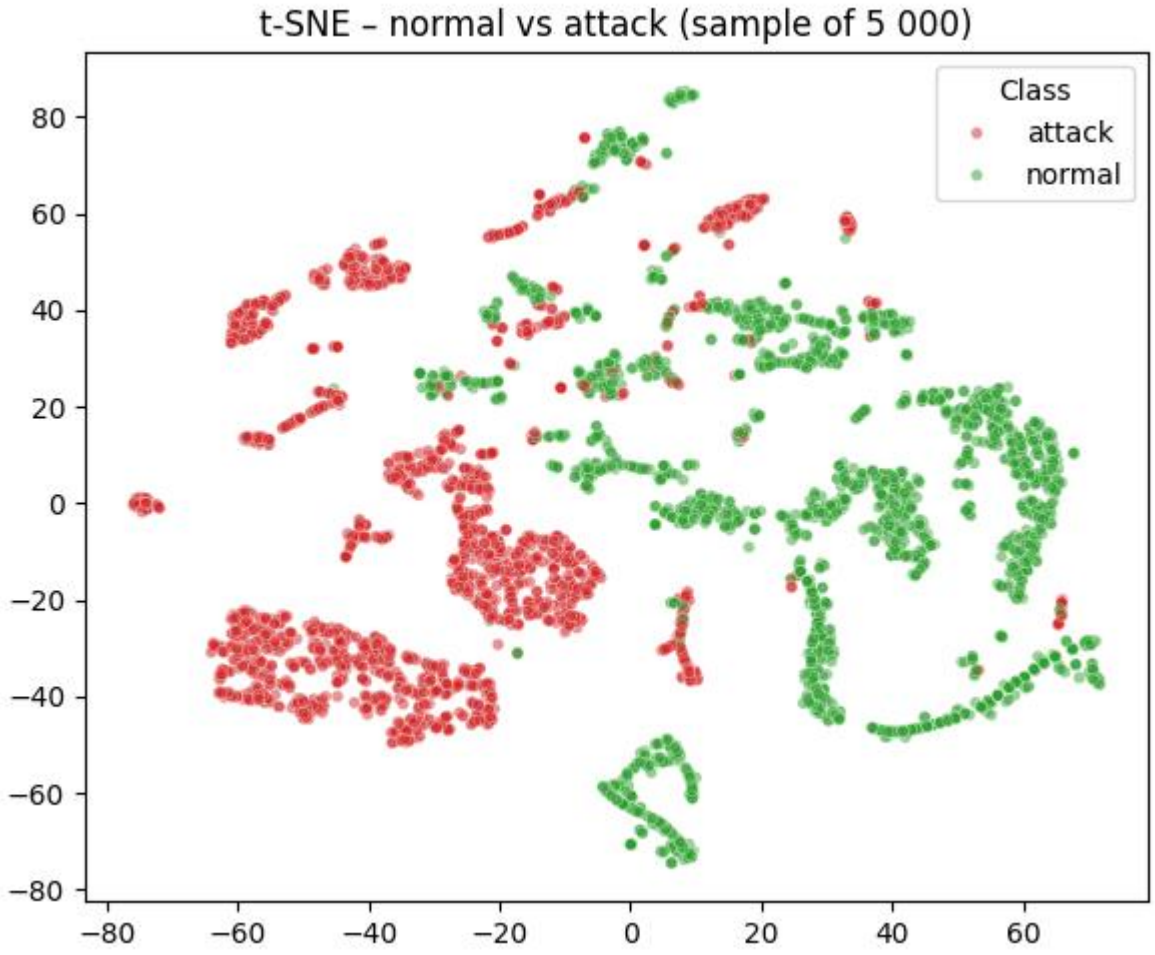
a mask for values above the threshold or below the negative threshold, excluding self-correlation

All numeric attributes were standard-scaled to normalise their ranges, ensuring no single feature dominates the Euclidean distance metric used by K-Means.

Prepared a clean matrix suitable for distance-based clustering

Exploratory Analysis

t-SNE (5,000 sample) showed clear natural separation between normal and attack regions.



The attack traffic sticks together in these tight little blobs, while the normal points are more spread out across the space. This is a good sign because it means the dataset really does have some structure that separates the two behaviours, even without labels.

Correlation map revealed tightly coupled error-rate and host-rate features.

Methodology

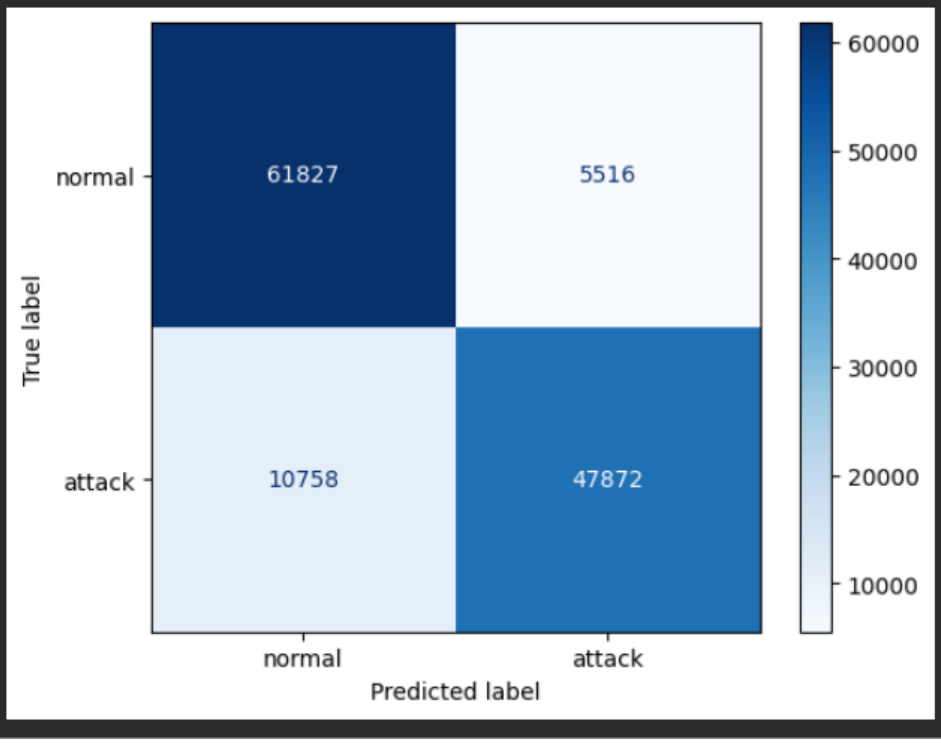
K-Means

Silhouette scores increased with higher K, but real-label evaluation showed that K = 7 over-split the data and reduced detection performance.

Although K = 7 produced a moderate silhouette score (~0.40), the clusters were scattered and lacked functional separation between normal and attack traffic. Because the task emphasises broad behavioural grouping, K = 2 was further assessed. Benchmarking indicated K = 7 achieved 81 % attack recall with 8 % false alarms, while K = 2 improved this to 84 % recall and 6 % false alarms, making the two-cluster structure more consistent with the

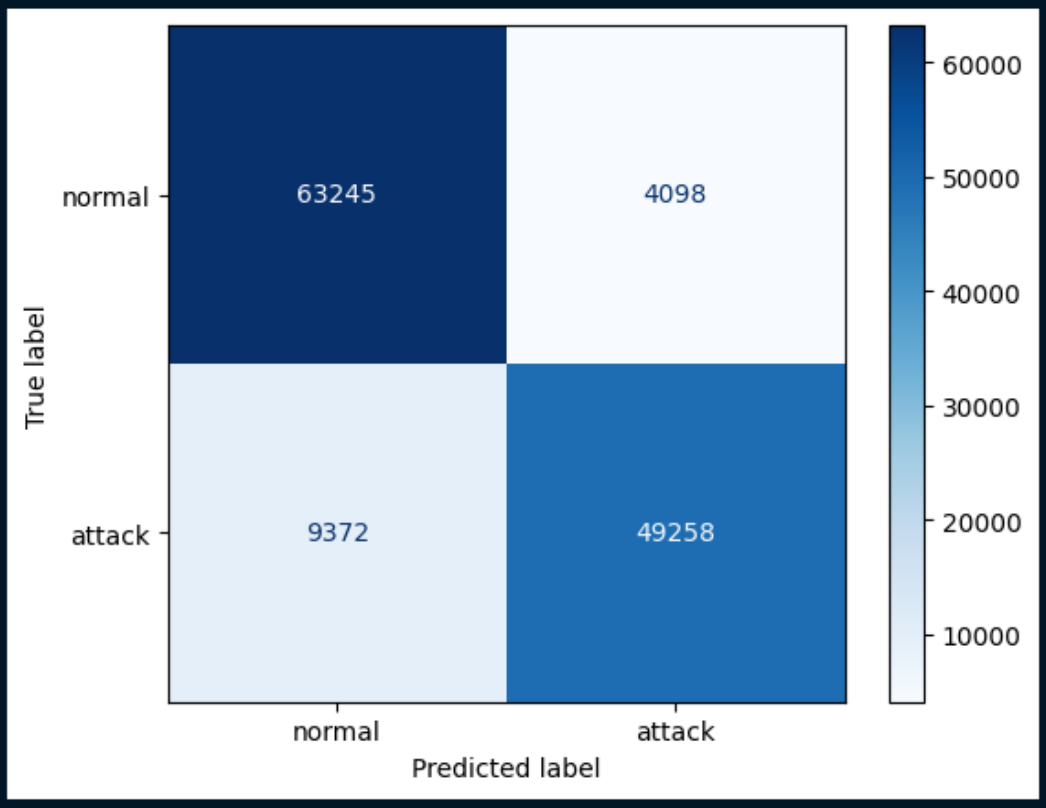
detection goal.

```
Confusion matrix (rows = true, cols = pred):
 [[61827  5516]
 [10758 47872]]

Detection rate (recall on attacks): 81.651%
False-alarm rate (normal mis-flagged): 8.191%

Classification report:
              precision    recall  f1-score   support

      normal       0.85      0.92      0.88     67343
      attack       0.90      0.82      0.85     58630

    accuracy                           0.87    125973
   macro avg       0.87      0.87      0.87    125973
weighted avg       0.87      0.87      0.87    125973
```



Findings when K = 7

```
Confusion matrix (rows = true, cols = pred):
 [[63245  4098]
 [ 9372 49258]]

Detection rate (recall on attacks): 84.015%
False-alarm rate (normal mis-flagged): 6.085%

Classification report:
              precision    recall  f1-score   support

      normal       0.87      0.94      0.90     67343
      attack       0.92      0.84      0.88     58630

    accuracy                           0.89    125973
   macro avg       0.90      0.89      0.89    125973
weighted avg       0.90      0.89      0.89    125973
```



findings when k = 2

K = 2 produced the strongest alignment with real behaviours.

Parameters: n_clusters= 2, init= 'k-means++', n_init='auto'.

DBSCAN

Tested multiple eps/min_samples settings on a 20k sample.

	eps	min_samples	detect	false_alarm	noise_%
2	0.3	15	0.209	0.357	28.8
1	0.3	10	0.165	0.303	23.9
5	0.5	15	0.131	0.212	17.5
4	0.5	10	0.099	0.186	14.5
0	0.3	5	0.097	0.222	16.3
8	0.7	15	0.091	0.146	12.0
7	0.7	10	0.067	0.117	9.3
3	0.5	5	0.053	0.128	9.3
6	0.7	5	0.031	0.081	5.8

Detection remained low (≤ 20%) and false-alarms high due to the curse of dimensionality: the dataset lacks clear density pockets for DBSCAN to exploit.

DBSCAN really didn't perform as well as K-Means on this dataset. No matter how tuned eps and min_samples, the detection rate stayed low, and the false-alarm rate stayed relatively high. The main reason is that DBSCAN needs well-separated dense clusters to work properly, but our dataset is high-dimensional, and the points don't form clean "density pockets."

Results

Cluster Composition (K = 2)

Cluster 0 = attack-heavy (92% attacks)

Cluster 1 = normal-heavy (87% normal)

	cluster	size	normal	attack	pct_normal
0	0	53356	4098	49258	7.7
1	1	72617	63245	9372	87.1

Shows a natural two-group structure.

Note: They are not fully 100% so expect some false-Alarm Rate and Affect in Detection Rate.

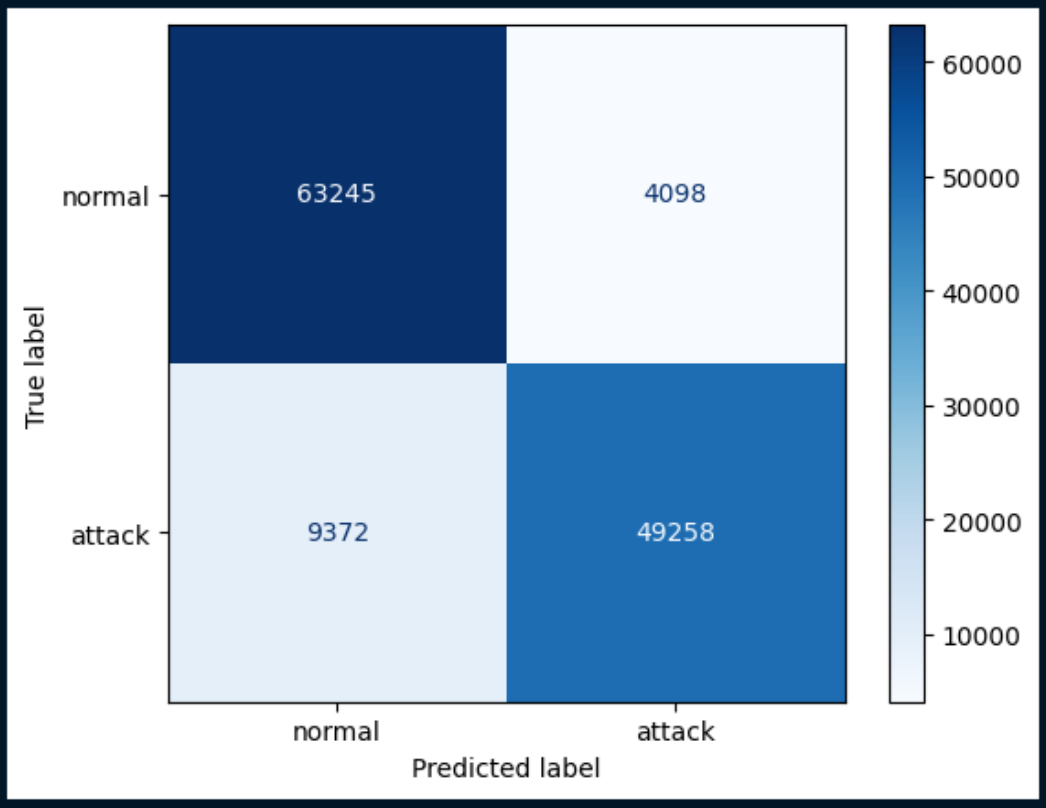
Labelled Evaluation (Benchmarking Only)

Confusion matrix gave:

- TN = 63,245
- FP = 4,098
- FN = 9,372
- TP = 49,258

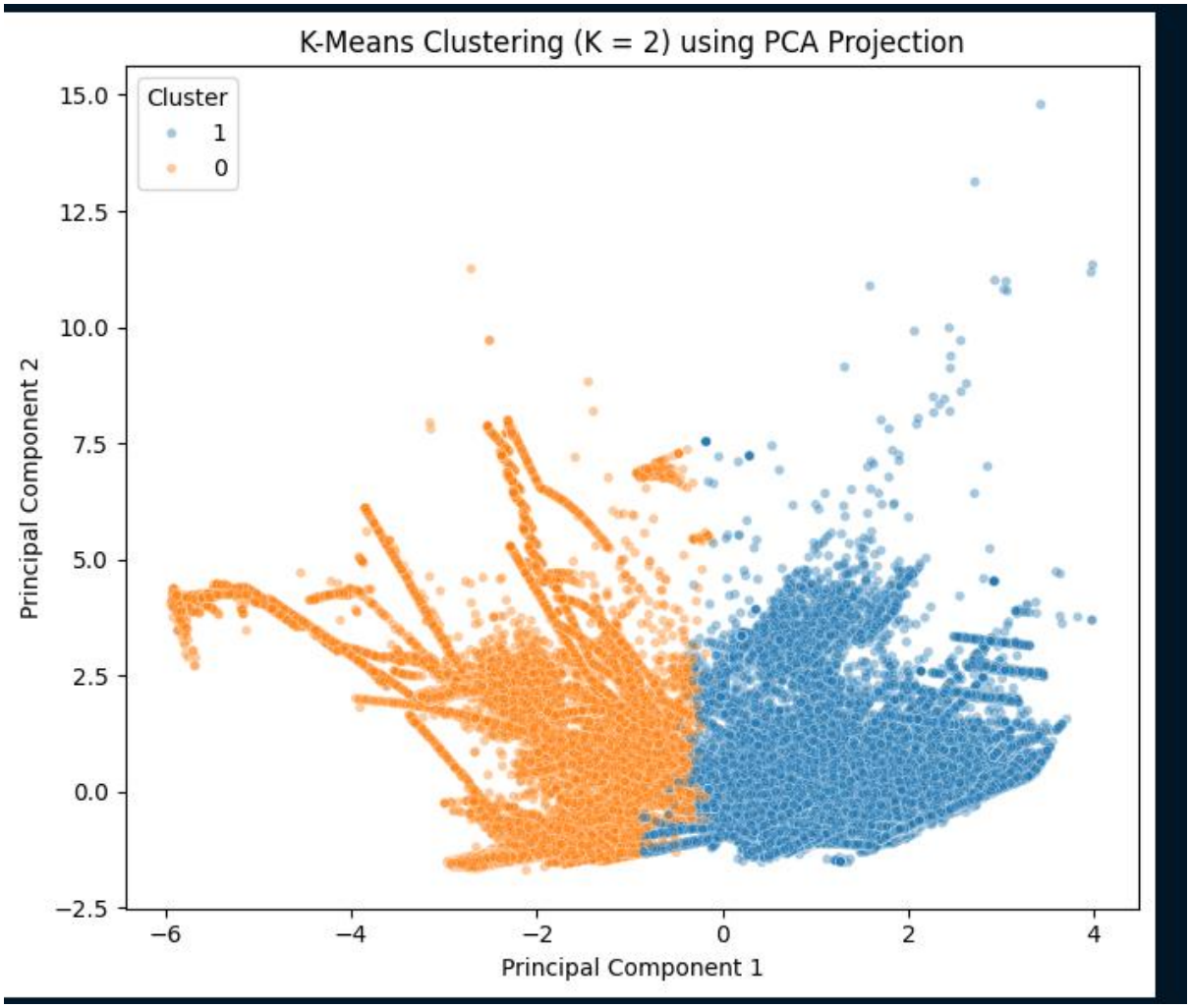
Performance: Detection Rate: 84% False-Alarm Rate: 6% Accuracy: 89%

```
Confusion matrix (rows = true, cols = pred):
 [[63245  4098]
 [ 9372 49258]]

Detection rate (recall on attacks): 84.015%
False-alarm rate (normal mis-flagged): 6.085%

Classification report:
              precision    recall  f1-score   support

      normal       0.87      0.94      0.90     67343
      attack       0.92      0.84      0.88     58630

    accuracy                           0.89    125973
   macro avg       0.90      0.89      0.89    125973
weighted avg       0.90      0.89      0.89    125973
```



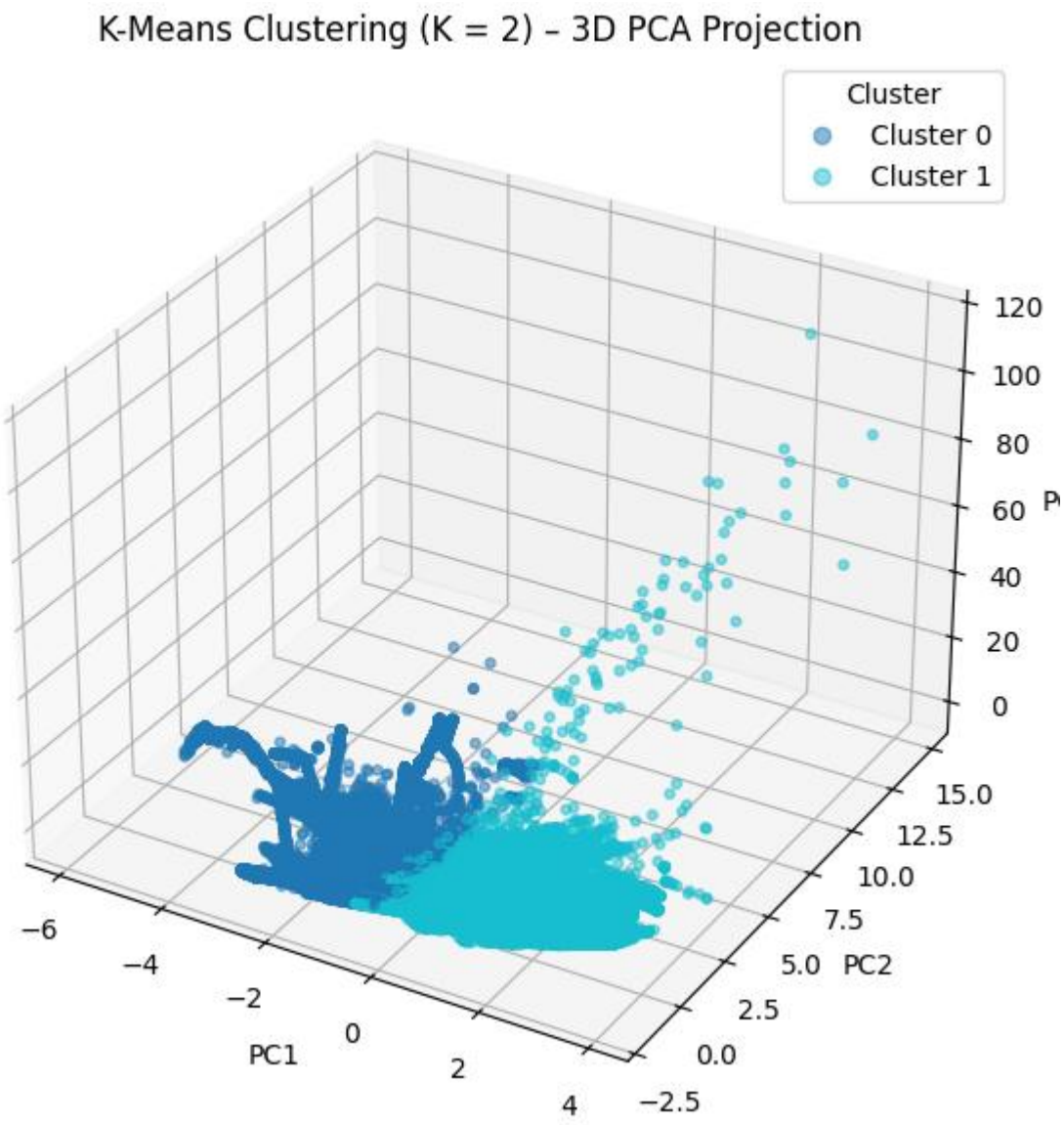
Visual Insights

PCA/t-SNE show two clean behavioural clusters.

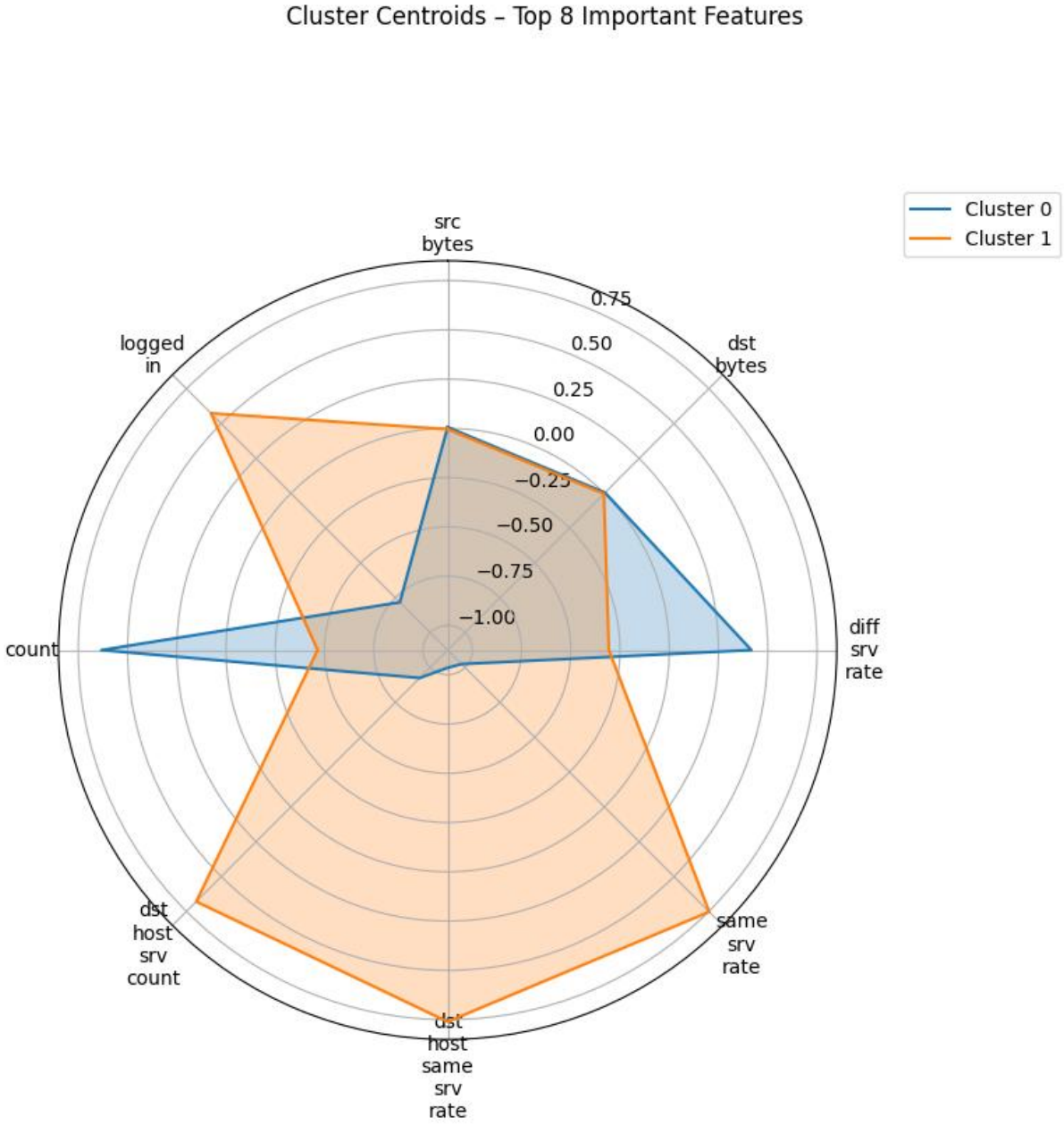


overall the split is clean and consistent

3D PCA (Plotly) reveals even clearer separation.



Radar plot highlights feature differences: attack flows peak strongly in host/service request rates.



High-Level Behaviour Inferred from the Clusters

The cluster patterns suggest that normal traffic tends to distribute its activity across many services and hosts, producing low values in repetition-based metrics (*same_srv_rate*, *dst_host_same_srv_rate*, *srv_count*). Flows in this group usually interact with different endpoints, transfer moderate byte volumes, and avoid tight loops of repeated requests.

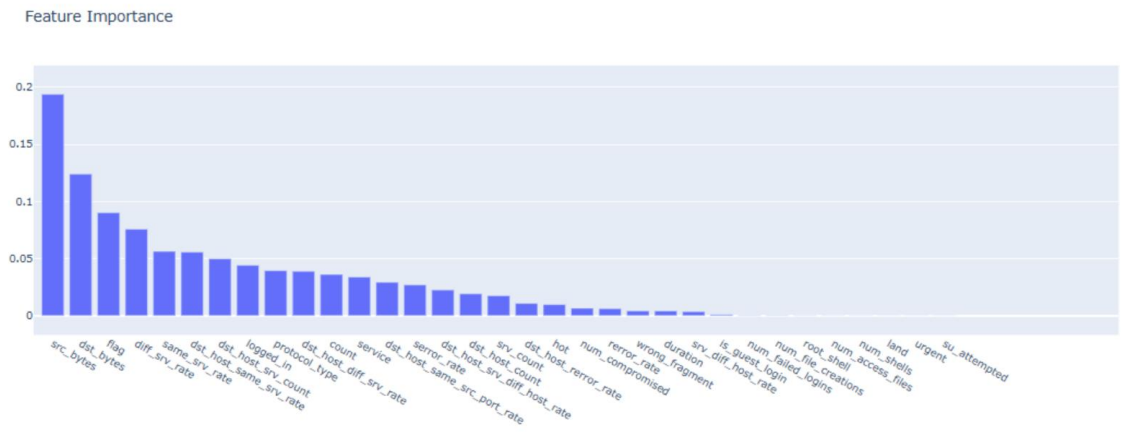
In contrast, attack traffic concentrates heavily on a small set of services or hosts. This produces sharply elevated *same_srv_rate*, *dst_host_same_srv_rate*, and *srv_count* values, reflecting repeated hits to the same target or service — a behaviour typical of scanning, probing, or automated attack routines. These flows also generate higher host-level activity footprints, indicating rapid bursts of similar requests within a short time window.

Taken together, the dataset shows that benign connections look broad, diverse, and lightly distributed, while malicious connections appear focused, repetitive, and service-intensive. This distinction forms the basis of the clustering separation observed.

Discussion

K-Means with K = 2 ended up matching the dataset's natural structure the best — the traffic only shows two dominant behavioural patterns: "normal-ish" and "attack-ish." Even though K = 7 gave a higher silhouette score, the real-label evaluation showed the opposite: detection dropped, false-alarms increased, and the clusters became tiny and meaningless. So, ignored the silhouette trick and chose the value that worked in practice. DBSCAN also didn't perform well because the dataset is high-dimensional and tightly packed — there aren't clear density regions for it to discover, so it mostly marked large chunks as noise. The radar plot helped explain why K = 2 makes sense: one cluster spikes aggressively on features like dst_host_count, dst_host_srv_count, and same_srv_rate, which are classic high-volume attack behaviours, while the other cluster stays low on those but higher on "normal" traffic indicators. Those opposite feature shapes showed that the two clusters truly behave differently, not randomly.

Random Forest was used just in for feature importance analysis for radar plot features.



Conclusion

Unsupervised K-Means (K = 2) successfully isolates attack behaviour in the dataset with strong detection and manageable false-alarms. PCA, t-SNE, and radar plots all confirm the separation. DBSCAN was unsuitable due to dimensionality limitations. Future work should explore dimensionality reduction before density-based clustering and add temporal/session-level features for more robust anomaly detection.