

Unsupervised Anomaly Detection on PE File Headers

Using K-Means Clustering and DBSCAN

CYSE 420 — Applied Machine Learning in Cybersecurity
Assignment 05 | Carl Lochstampfor | Fall 2026

1. Introduction and Dataset

This report documents a step-by-step implementation of two unsupervised anomaly detection methods, K-Means clustering and DBSCAN, applied to static malware detection on Windows Portable Executable (PE) header data. Unlike the supervised classifiers built in Assignment 03 on the same dataset, neither model receives the class label during fitting or parameter selection. Each model instead learns the structure of the data on its own and assigns every file an anomaly score; the label is introduced only afterward to measure how well that score corresponds to maliciousness. The report follows the working order of the analysis: preprocessing, parameter tuning, results for each model, and comparison.

The dataset, `file_pe.csv`, contains 19,611 samples described by 79 columns: a file identifier, 77 numeric PE header features, and a binary `Malware` label. No values are missing. As delivered, the class distribution is 14,599 malicious to 5,012 benign, or 74.4 percent malicious. Following the assignment instructions, malicious files (label 1) are treated as outliers and clean files (label 0) as normal data.

Label discipline. Every parameter rule in Section 3 was fixed before any result was examined, and each depends only on unlabeled quantities: silhouette scores, distance percentiles, and feature dimensionality. The `Malware` column appears solely in the evaluation sections and in the cluster composition tables, both of which follow fitting. Selecting a parameter by comparing candidate values against the labels and then reporting performance on those same labels would inflate every figure in this report, so that practice was excluded by construction rather than avoided informally.

2. Step One: Data Preprocessing

2.1 Removal of an identifier-based label leak

The `Name` column was removed before any modeling. As established in Assignment 03, benign samples carry genuine Windows filenames while every malicious sample carries a `VirusShare_` prefix, so the column separates the classes perfectly and would leak the label into an ostensibly unsupervised procedure. The assignment instruction to cluster on feature columns only reinforces this: an identifier is not a feature.

2.2 Duplicate records

With the identifier removed, 2,980 duplicate rows appeared, of which 2,934 were malicious. These represent files with different hashes but byte-identical PE headers, the expected signature of repacking and polymorphism. Duplicates were removed, leaving 16,631 samples at 70.1 percent malicious (11,665 malicious, 4,966 benign). For distance-based methods, retaining them would also distort density estimates: identical rows sit at distance zero from one another and would inflate the apparent density of repacked families.

2.3 Constant features

Eight of the 77 features take a single value across the entire dataset and were removed, leaving 69: `e_magic` and seven `Section` statistics that are uniformly zero. A constant feature contributes an identical term to every pairwise distance and therefore cannot affect any clustering outcome.

2.4 Two evaluation sets and the contamination problem

Clustering-based anomaly detection rests on the assumption that anomalies are rare and lie apart from dense regions of normal data. In this dataset that assumption is inverted: after deduplication, the designated anomaly class constitutes 70.1 percent of the records. Malware is the dense majority, and a detector that flags sparse, isolated points cannot be expected to recover it.

Rather than discard the instruction or the assumption, the analysis was run twice on two evaluation sets, with an identical pipeline and identical parameter rules:

Set A is the full deduplicated dataset (16,631 records, 70.1 percent malicious), exactly as instructed. **Set B** retains all 4,966 benign files and a random sample of malicious files reduced to five percent contamination (261 malicious files, 5,227 records total), the regime in which unsupervised anomaly detection is normally evaluated. The five percent figure and the sampling seed were fixed before any model was run, so the subsampling is a stated design choice rather than a tuned parameter. The contrast between the two sets is treated as a result in its own right in Section 6.

2.5 Transformation of the feature space

Three transformations were applied to each set independently. First, a `log1p` transform compressed the heavy right skew of the header fields: 51 of the 69 features have absolute skewness above 10, and `ImageBase` reaches roughly 4.3 billion. Because every value in the data is non-negative, the transform is defined throughout. Without it, a handful of large-magnitude fields would dominate every Euclidean distance. Second, the features were standardized to zero mean and unit variance, which is a precondition for both algorithms: DBSCAN in particular interprets its `eps` parameter as a radius in the units of the feature space. Third, Principal Component Analysis retained the components explaining 95 percent of variance, reducing 69 features to 37 components for Set A and 26 for Set B, which mitigates the distance concentration that degrades neighborhood methods in high dimensions.

No train/test split was used. Neither model consumes labels during fitting, and no quantity derived from the labels is used to select parameters, so there is no information that a held-out partition would protect. All scaling and PCA parameters are estimated from unlabeled feature data.

3. Step Two: Parameter Tuning

3.1 K-Means: choosing K

The selection rule, fixed before results were examined, was to take the value of K with the highest silhouette score over the range 2 to 10, with the elbow plot serving as a supporting check. Both quantities are computed from the feature space alone. Silhouette peaked sharply at $K = 2$ for both sets (0.443 for Set A, 0.503 for Set B), and the inertia curves decline smoothly with no competing bend, so $K = 2$ was adopted throughout.

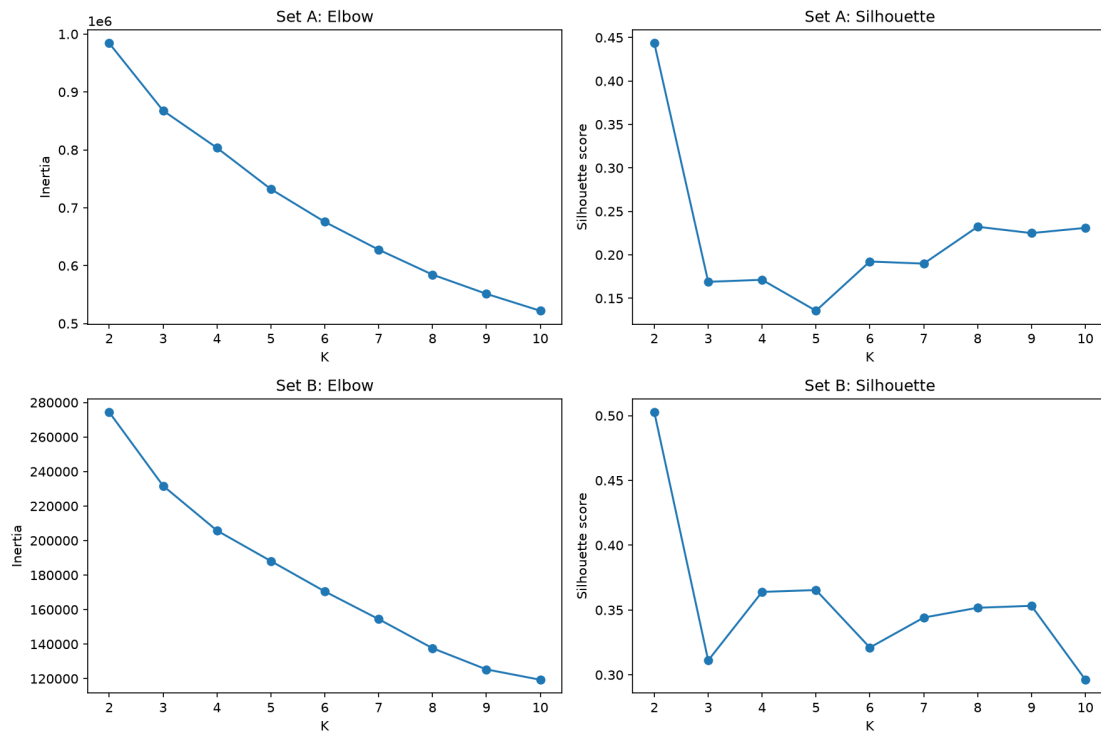


Figure 1. Inertia (left) and silhouette score (right) across K for Set A (top) and Set B (bottom). Silhouette identifies K = 2 in both cases; the elbow curves show no sharper bend elsewhere.

A practical consideration reinforces the low value: at K of five or more on Set B, K-Means begins isolating single points as their own clusters. A point that is its own centroid has a centroid distance of zero and would therefore be scored as maximally normal, defeating the anomaly score described next.

3.2 K-Means: anomaly score and flag threshold

K-Means produces cluster assignments, not anomaly scores, so a score was derived as the Euclidean distance from each point to its nearest centroid: points that sit far from every centroid belong to no compact group. This continuous score is what the ROC analysis consumes. For the confusion matrices, a binary flag is required, and the rule fixed in advance was to flag the five percent of points with the largest distances, the 95th percentile. The resulting thresholds were 12.875 for Set A (832 files flagged) and 10.553 for Set B (262 files flagged).

Applying the same five percent rule to Set A, where 70.1 percent of files are malicious, necessarily caps recall near seven percent. Setting the rate to match the known contamination would import label knowledge into a detector that is supposed to be blind to it, so the uniform rule was kept and the consequence is reported as a finding.

3.3 DBSCAN: choosing min_samples and eps

DBSCAN requires two parameters. For min_samples, the analysis used the standard heuristic of twice the dimensionality of the data, giving 74 for Set A and 52 for Set B. For eps, the k-distance method was applied: for every point, the distance to its min_samples-th nearest neighbor is computed and the sorted values are plotted. Points in dense regions need only a short reach to gather enough neighbors, so the curve is flat before rising steeply where sparse points begin. To keep the choice reproducible and consistent with the K-Means threshold, eps was set to the 95th percentile of the k-distance values: 7.592 for Set A and 6.783 for Set B. Figure 2 confirms that this value falls in the bend of each curve, which is the region manual inspection targets.

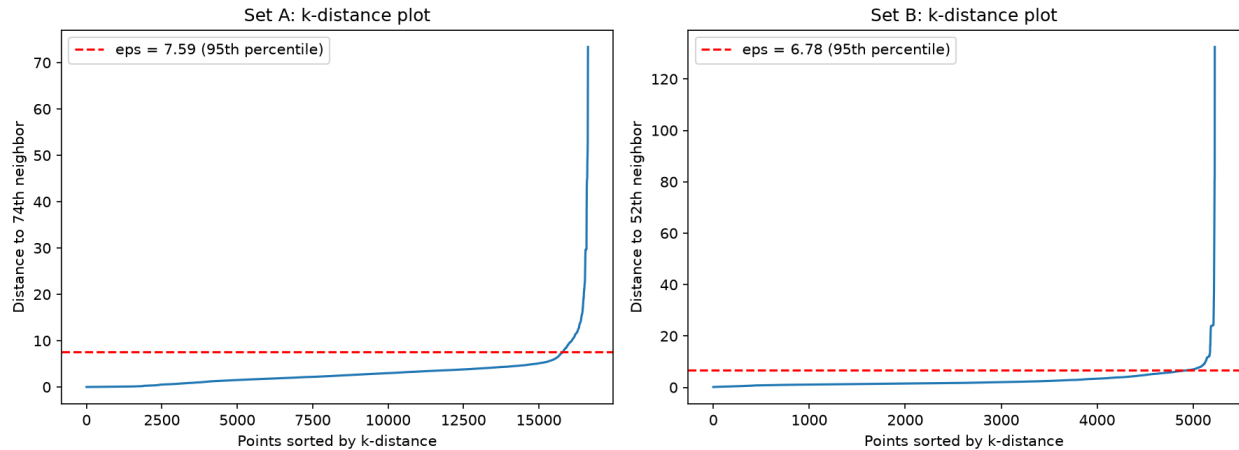


Figure 2. Sorted k -distance curves with the selected ϵ marked. Set A (left) uses the 74th nearest neighbor, Set B (right) the 52nd. Both thresholds land in the bend where the curve turns upward.

3.4 DBSCAN: constructing an anomaly score

DBSCAN emits only cluster labels and a noise marker; it has no `decision_function`, no `predict` method for new data, and no native continuous score. Because ROC-AUC requires a ranking rather than a binary outcome, the k -distance value computed in Section 3.3 was reused as the anomaly score: a point that must reach far to find `min_samples` neighbors is, by DBSCAN's own definition of density, more anomalous. The two outputs agree in direction: points DBSCAN labels as noise have mean k -distance 17.32 against 2.50 for clustered points in Set A, and 20.15 against 2.36 in Set B, a roughly sevenfold separation in both cases. Confusion matrices for DBSCAN use its own noise labels; ROC curves use the k -distance score.

4. Step Three: K-Means Results

With $K = 2$, Set A divided into clusters of 15,576 and 1,055 records and Set B into 4,929 and 298. Cross-tabulating the clusters against the labels after fitting shows that neither partition corresponds to the malicious/benign division. In Set B the smaller cluster is entirely benign (298 benign, 0 malicious), and all 261 malicious files fall inside the large cluster; in Set A the smaller cluster is 88 percent benign (930 benign, 125 malicious). What K-Means separates is a distinct subpopulation of benign files, not malware.

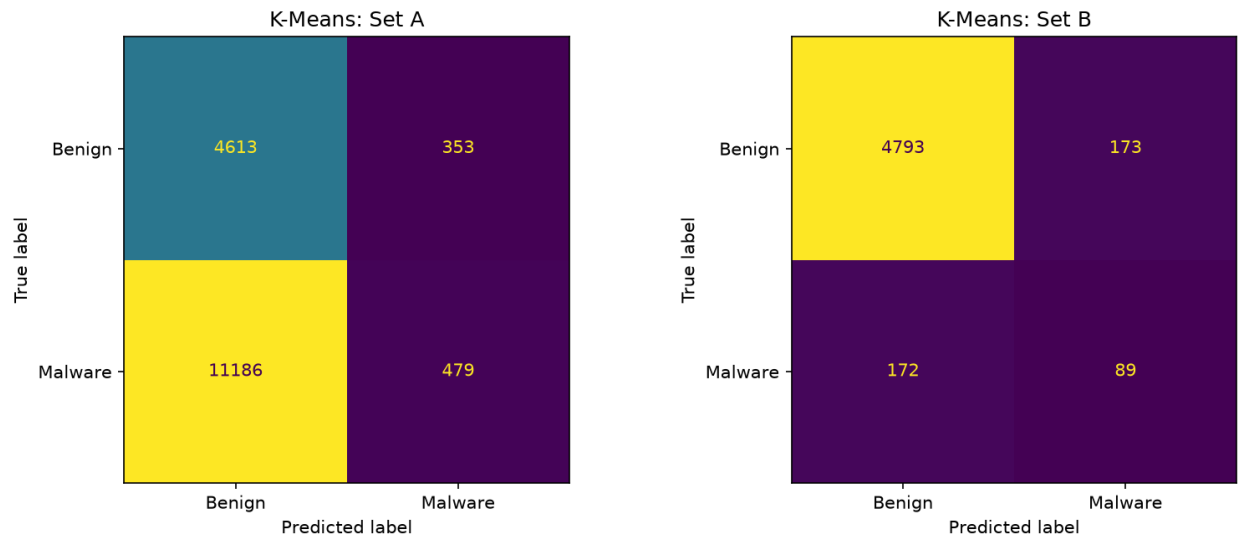


Figure 3. K-Means confusion matrices at the fixed five percent flag rate. Cells read [true negative, false positive; false negative, true positive].

On Set A the detector flagged 832 files, of which 479 were malicious: recall 4.1 percent at 57.6 percent precision, against a 70.1 percent base rate. Flagging files at random would have been more precise. On Set B the same procedure flagged 262 files, of which 89 were malicious: recall 34.1 percent at 34.0 percent precision and a false positive rate of 3.5 percent. Accuracy is unhelpful on both sets and is reported here only to be set aside: Set B's 93.4 percent trails the 95.0 percent obtainable by labeling every file benign, and Set A's 30.6 percent trails the 70.1 percent obtainable by labeling every file malicious.

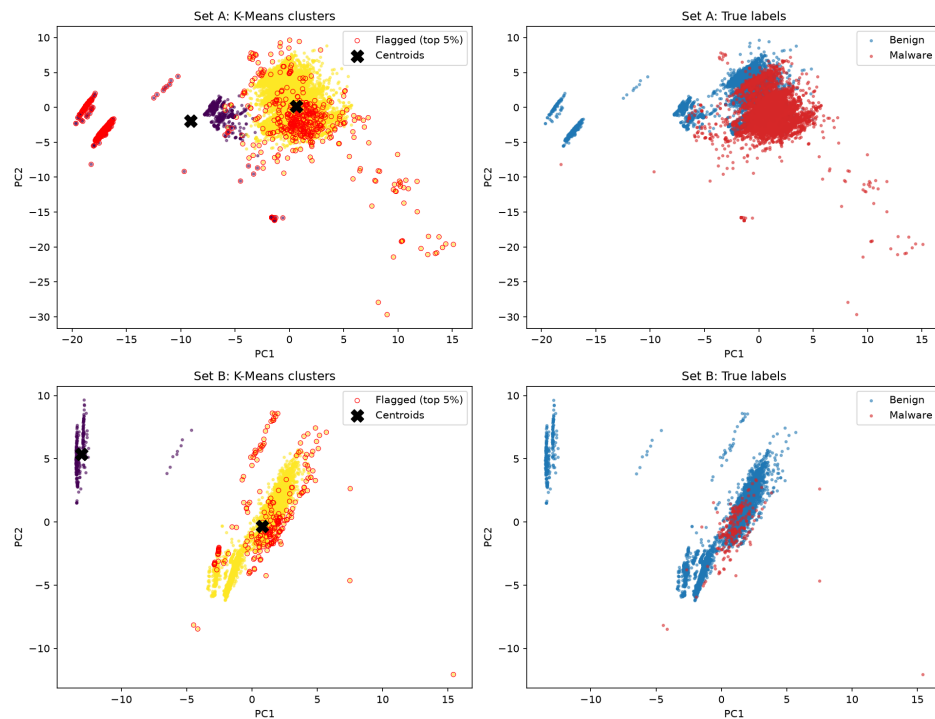


Figure 4. K-Means clusters with centroids and flagged points (left) against true labels (right), projected onto the first two principal components. The projection retains 24.6 percent of variance for Set A and 33.0 percent for Set B, so apparent overlap understates the separation available to the models.

Figure 4 explains the Set A result directly. The benign subpopulation occupies the narrow strands at the left of the projection, far from the centroid of the main mass, so centroid distance flags benign files preferentially. Malware occupies the dense center and is thereby scored as normal. In Set B the malicious files lie inside the principal benign band rather than outside it, which is why two thirds of them escape a rule that flags the band's extremes.

5. Step Four: DBSCAN Results

DBSCAN found three clusters in each set, labeling 603 points as noise in Set A (3.6 percent) and 115 in Set B (2.2 percent). Both figures fall below the five percent implied by the `eps` percentile because a point with a long `k`-distance is not automatically noise: if it lies within `eps` of a core point it is absorbed as a border point. DBSCAN therefore operates at a more conservative flag rate than K-Means, a difference that matters when reading the confusion matrices.

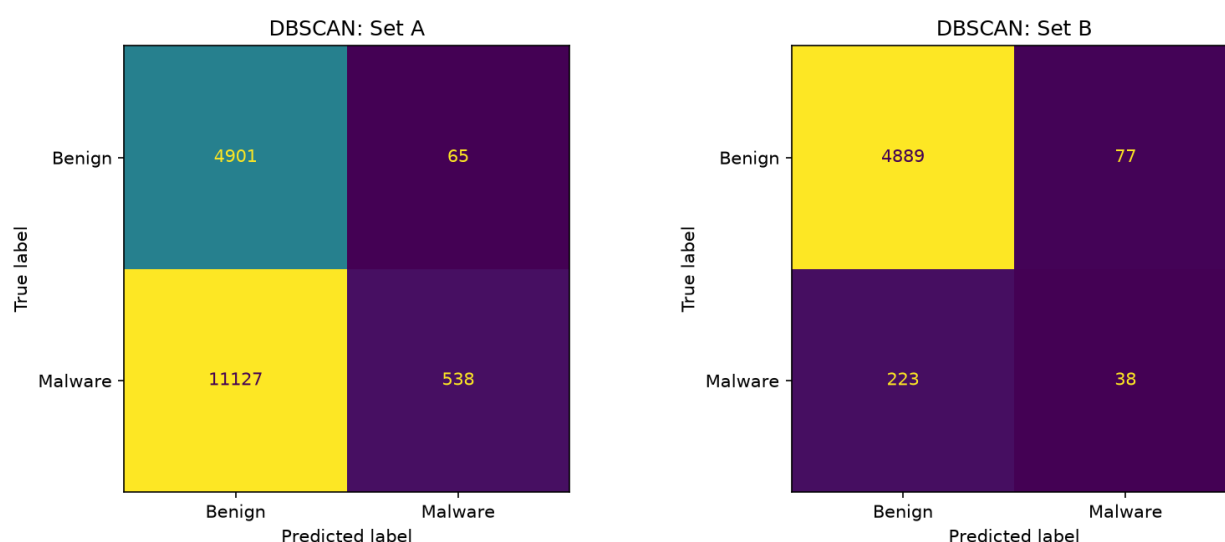


Figure 5. DBSCAN confusion matrices using the algorithm's own noise labels as the positive class.

Cluster	Set A benign	Set A malicious	Set B benign	Set B malicious
Noise (-1)	65	538	77	38
Cluster 0	4,611	11,027	4,552	203
Cluster 1	0	100	290	0
Cluster 2	290	0	47	20

Table 1. Composition of DBSCAN clusters by true label, computed after fitting.

Table 1 contains the most informative structure recovered in this assignment. Three findings follow from it. First, a 290-record cluster that is 100 percent benign appears in both sets; it is the same subpopulation K-Means isolated in Section 4, identified independently by a density-based method. Because DBSCAN recognizes this group as dense in its own right rather than distant from a centroid, it produces only 65 false positives on Set A where K-Means produces 353. Second, Set A contains a 100-record cluster that is entirely malicious: a group of files whose headers are similar enough to form a dense region, plausibly a single family or packer. Density-based detection cannot flag such a group, since its members are each other's neighbors. Malware that arrives in volume is invisible to this class of method by construction. Third, in Set B, 203 of the 261 malicious

files sit inside the principal benign cluster, corroborating the K-Means conclusion that most malicious headers are structurally unremarkable.

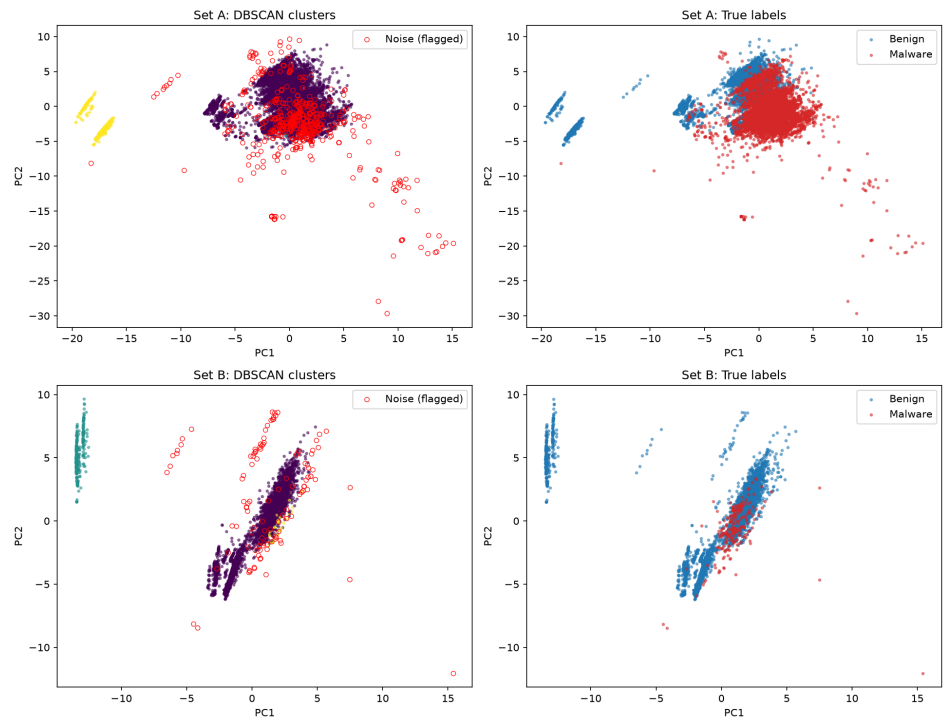


Figure 6. DBSCAN clusters and noise points (left) against true labels (right) on the same two-component projection used in Figure 4.

6. Model Evaluation and Comparison

Model	Set	ROC-AUC	Flagged	Precision	Recall	FPR
K-Means	Set A	0.350	832	0.576	0.041	0.071
K-Means	Set B	0.794	262	0.340	0.341	0.035
DBSCAN	Set A	0.518	603	0.892	0.046	0.013
DBSCAN	Set B	0.910	115	0.330	0.146	0.016

Table 2. Detection performance. ROC-AUC uses the continuous anomaly scores; the remaining columns describe each model at its own operating point.

Contamination dominates every other factor. Holding features, algorithms, and parameter rules constant and changing only the proportion of malicious files raised K-Means from 0.350 to 0.794 and DBSCAN from 0.518 to 0.910. The Set A figure for K-Means is the more instructive number: an AUC of 0.350 is not merely weak but inverted, meaning the score ranks benign files as more anomalous than malicious ones. That is the rarity assumption failing in a measurable way rather than an implementation fault, and it was left uncorrected deliberately. Reversing the sign of the score to obtain 0.650 would be a choice made after inspecting the labels, which is precisely the practice this report avoids.

DBSCAN ranks anomalies better than K-Means on both sets. It achieves a higher AUC and a materially lower false positive rate (1.3 and 1.6 percent against 7.1 and 3.5 percent). The reason is visible in Section 5: density is a more faithful notion of normality than proximity to a centroid when the normal class is not a single

compact blob. On Set A, DBSCAN is additionally the only configuration with precision above the base rate, at 89.2 percent, meaning the small set of points it does flag are genuinely malicious outliers.

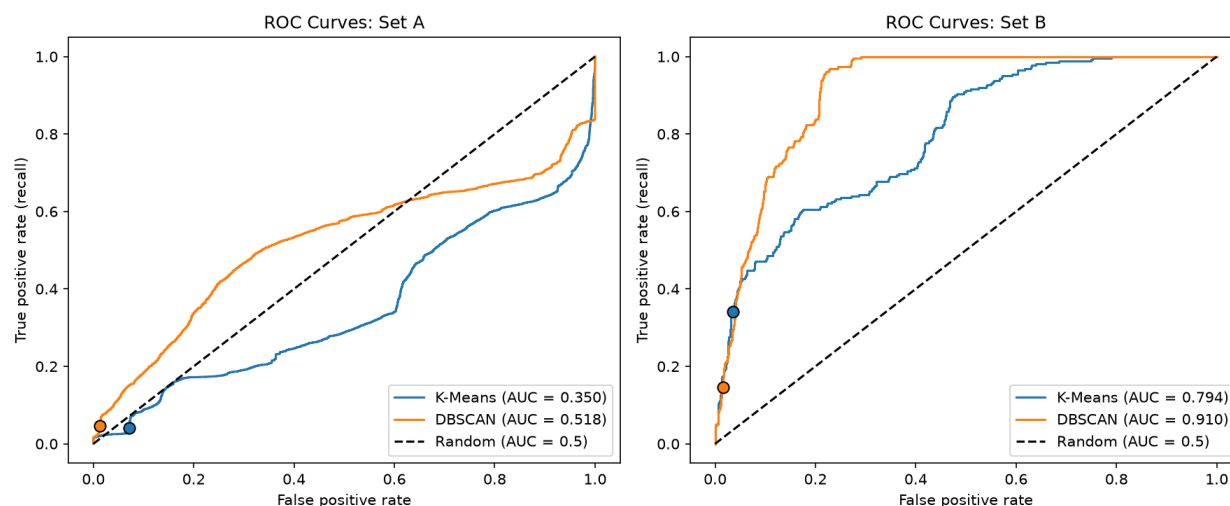


Figure 7. ROC curves for both models on each set. The filled circles mark each model's actual operating point (its false positive rate and recall from Table 2), which lies on the curve for K-Means and near it for DBSCAN, whose noise labels are not a pure threshold on k -distance.

Ranking quality and flag rate are separate properties. On Set B, DBSCAN ranks better (0.910 against 0.794) yet catches fewer malicious files at its own operating point (14.6 percent against 34.1 percent), because its noise rule flags 2.2 percent of files where the K-Means rule flags 5.0 percent. Figure 7 resolves the apparent contradiction: below roughly four percent false positive rate the two curves are nearly indistinguishable, and DBSCAN's advantage opens up between five and twenty percent, by which point it has recovered all 261 malicious files. Which model is preferable therefore depends on the alert budget. A team able to tolerate a ten percent false positive rate should prefer DBSCAN with a threshold applied to the k -distance score; a team confined to a very low alert rate would find the two nearly equivalent.

Set A remains near random for both models even where the ROC curve is informative. DBSCAN's curve on Set A rises above the diagonal at low false positive rates before crossing below it, and the 0.518 summary averages those two regions away. The practical reading is that the very top of the DBSCAN ranking is usable on Set A while the rest of it is not.

7. Comparison with the Supervised Baseline

Model	Data	ROC-AUC	Precision	Recall	FPR
kNN (k = 3), supervised	Assignment 03 test set	0.984	0.981	0.984	0.045
Decision Tree (depth 7), supervised	Assignment 03 test set	0.964	0.978	0.974	0.052
K-Means, unsupervised	Set A	0.350	0.576	0.041	0.071
DBSCAN, unsupervised	Set A	0.518	0.892	0.046	0.013
K-Means, unsupervised	Set B	0.794	0.340	0.341	0.035
DBSCAN, unsupervised	Set B	0.910	0.330	0.146	0.016

Table 3. Unsupervised results against the supervised classifiers built on the same PE header data in Assignment 03.

Three qualifications apply to this comparison. The supervised models were scored on a 4,990-record held-out test set rather than on the full data, so Set A is the closer match in composition at approximately 70 percent malicious. The supervised models were trained on labels the clustering models never saw, which is the source of the gap rather than an incidental difference. And the Assignment 03 hyperparameters were themselves selected on the test set, which makes those figures modestly optimistic.

Even granting all three, the gap is not close: 0.984 against 0.518 on data of comparable composition. The conclusion is not that clustering is a poor technique but that it answers a different question. Supervised classification requires a labeled corpus of known malware; clustering requires none and can operate where labels do not exist, at a cost in detection performance that this experiment quantifies.

8. Conclusions

- 1. The proportion of anomalies governed the outcome more than the choice of algorithm.** Reducing malicious files from 70.1 percent to 5.0 percent, with everything else held fixed, moved K-Means from 0.350 to 0.794 and DBSCAN from 0.518 to 0.910 in ROC-AUC.
- 2. Density-based scoring outperformed centroid distance.** DBSCAN achieved a higher AUC and a lower false positive rate on both sets, and was the only configuration to exceed the base rate in precision on Set A.
- 3. A model's ranking quality and its default flag rate are distinct.** DBSCAN ranked better on Set B while catching less malware at its own operating point, because its noise criterion is conservative. Applying an explicit threshold to the k-distance score would recover the difference.
- 4. Small clusters are not anomalies.** Both algorithms independently isolated the same 290-record benign subpopulation, and the K-Means minority cluster in Set B was entirely benign. Treating cluster size as a proxy for maliciousness would have produced a detector with zero true positives.
- 5. Grouped malware is invisible to density-based detection.** The 100-record, wholly malicious cluster in Set A satisfies the definition of a dense region and is therefore classified as normal, a structural limitation rather than a tuning problem.
- 6. Supervised classification on identical data remains far stronger,** which positions these methods as tools for unlabeled settings rather than as replacements for a trained classifier.

9. Limitations

Set B rests on a single random draw of 261 malicious files at one contamination level; a different draw or a different level would shift the figures, and no sensitivity analysis across levels was performed. The parameter rules were selected for being label-free and reproducible, not for being optimal, and a labeled validation set would almost certainly yield better values for K , eps , and the flag threshold, at the cost of the independence that motivates the exercise. Several retained features (`Machine`, `Magic`, `Subsystem`, `Characteristics`, `DllCharacteristics`) are identifier codes or bit flags for which numeric distance has no natural meaning, yet they contribute to every distance computed. The two-component projections in Figures 4 and 6 retain only 24.6 and 33.0 percent of variance and are illustrative rather than evidentiary. Finally, as noted in Assignment 03, benign and malicious samples originate from different collection sources, so an unknown portion of the structure these models detect may reflect provenance rather than executable behavior.