Anomaly Detection using Clustering

Team Griffin - Ivan Kutovoi, Dominic Cunningham, Jack Tumulty

Abstract

This paper provides details about the process of creating clustering anomaly detectors. It goes into detail about the various values and parameters used for the models and the reduction algorithm. In the end, some potential improvements are given.

The latest version of the project has DBScan (Density-Based Spatial Clustering of Applications with Noiseand) and K-Means algorithms. Figures 1 and 2 show the final clusters. The results are the following:

Metric / Algorithm	DBScan (e=0.0172, m=2)	K-Means (d=16, k = 76)
False Negatives	**0**	**0-1**
False Positives out of 8000 samples	661	~550
Accuracy	91.74%	92.7% +- 0.3%
True Positive Rate	**100%**	**99.98% +- 0.02%**
False Positive Rate	16.53%	14.4% +- 0.5%
F1	92.37%	93.5% +- 0.5%

Note: error margin was not calculated but approximated based on 20 samples.

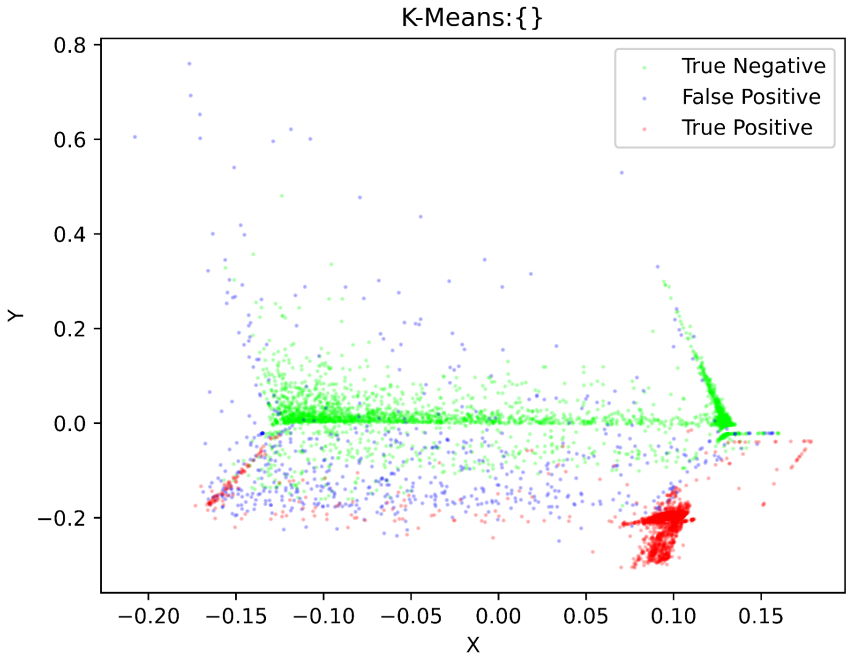

Fig 1. K-Means, k = 79.

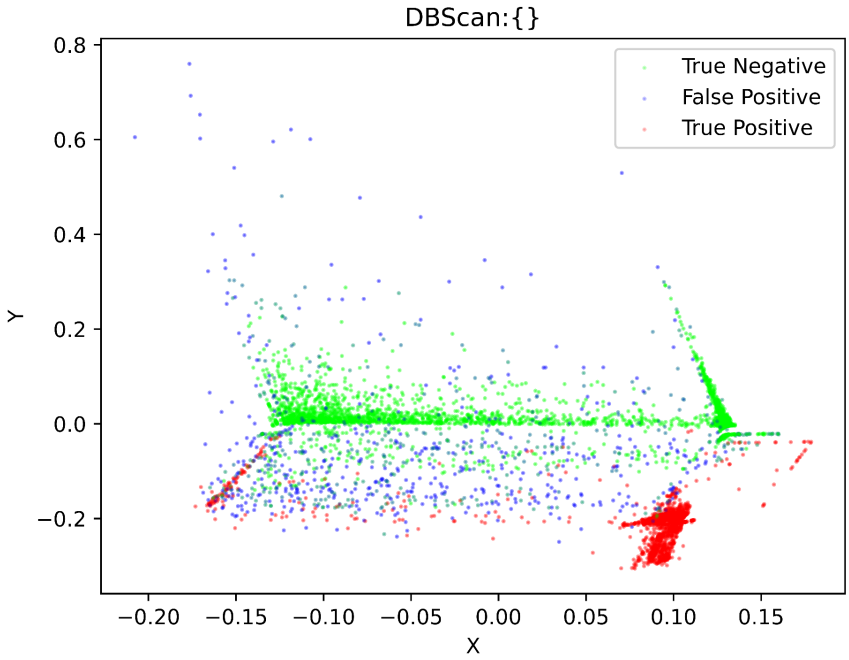

Fig 2. DBScan, e = 0.0172, min = 2.

Dimension Reduction

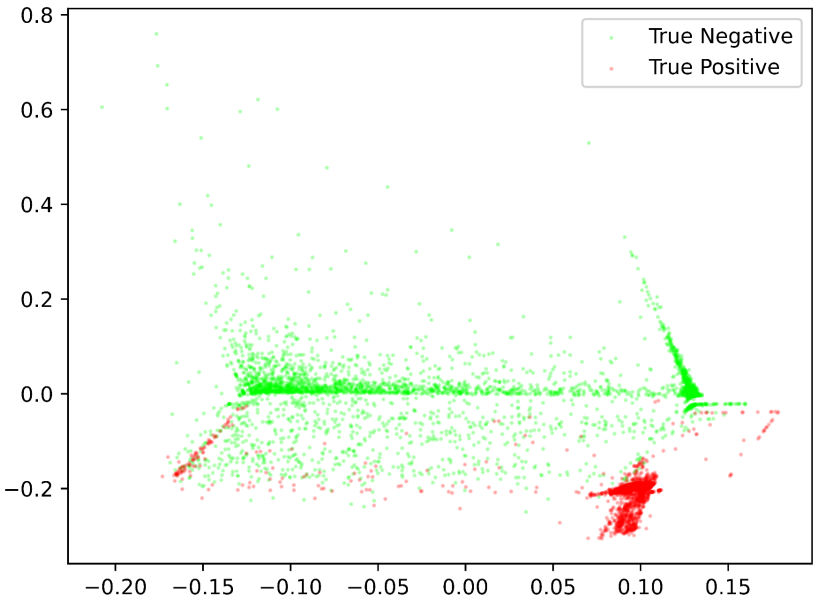

Fig 3. All the data points.

Figure 3 shows all the data points, with red being attack data and green being normal data. Two discernible clusters can be identified for each normal data and attack data. They are shown on figure 4.

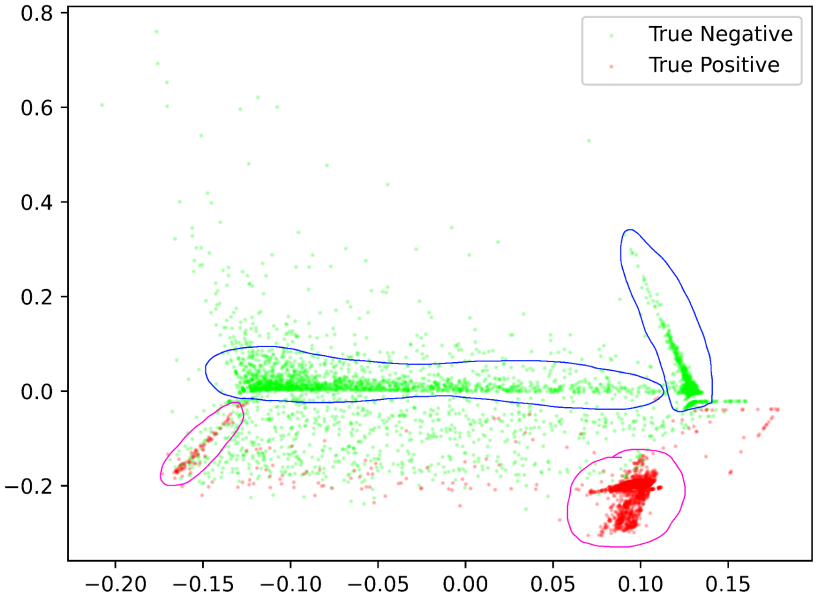

Fig 4. Clusters.

A linear decision boundary cannot be drawn for these samples. Below are two ways that one could draw it. However, both ways leave out a lot of outliers. One of them would result in many false positives, the other - many false negatives. These ways are shown on figures 5 and 6.

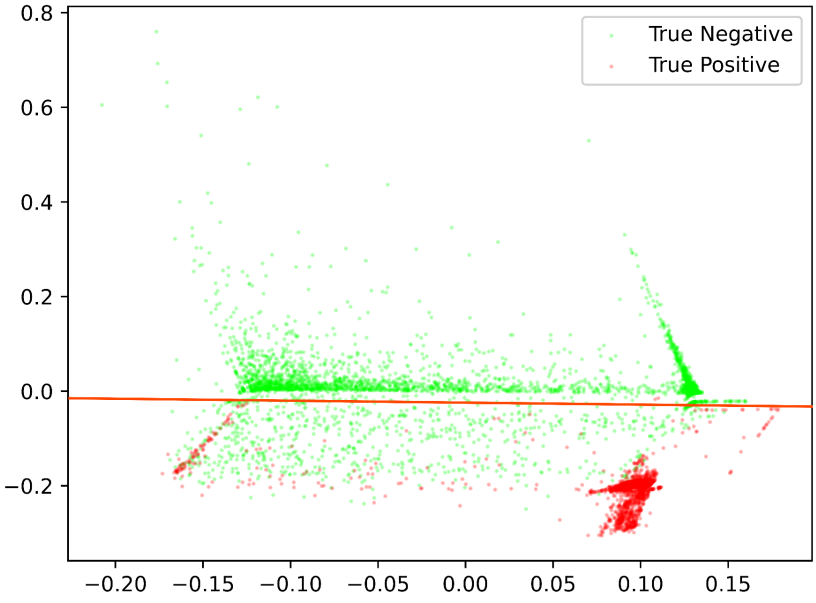

Fig 5. Linear decision resulting in false positives

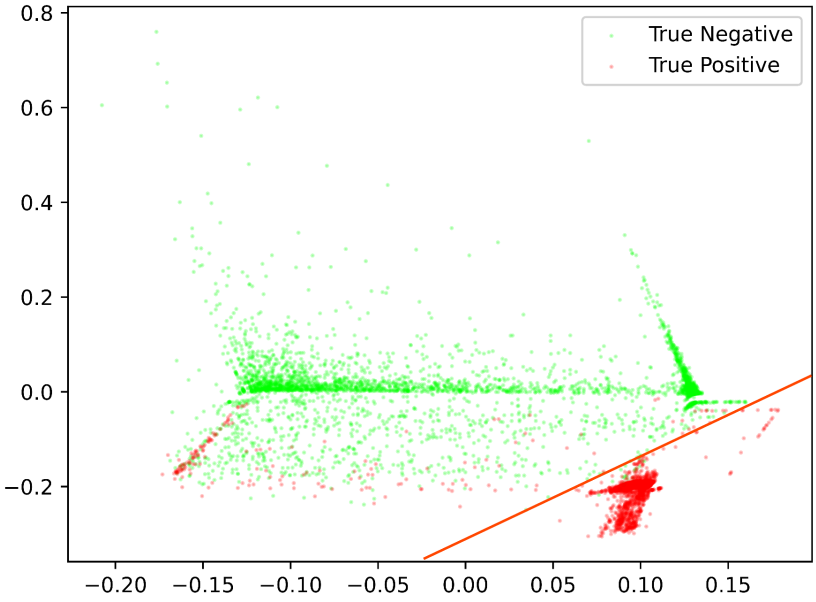

Fig 6. Linear decision resulting in false negatives

A curved decision boundary would work much better. However, it would still result in many false positives and some false negatives.

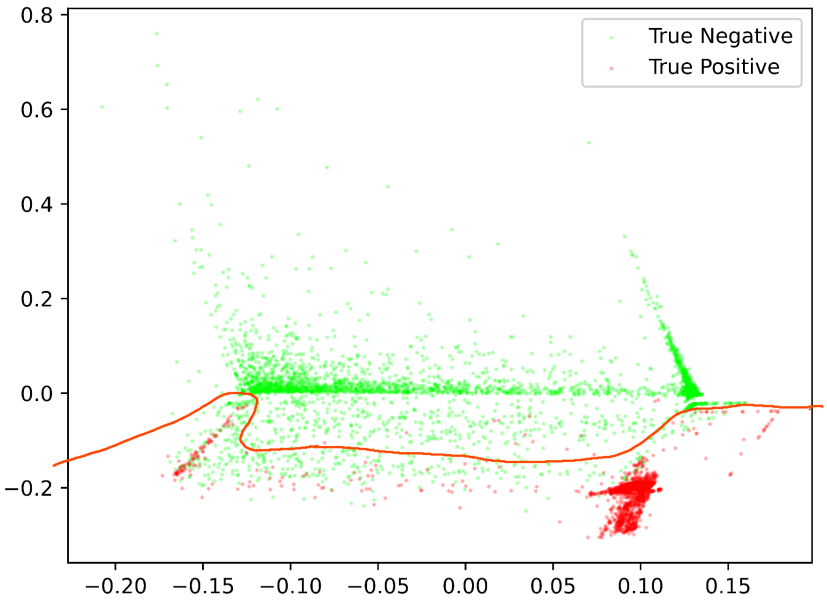

Fig 7. Curved decision boundary.

Hyperparameter Tuning

It was decided that the simplest and most effective way of determining the best hyperparameter values would be brute-force. An algorithm was implemented that was used to test different values for all the parameters that the model takes.

Every parameter was tested, including more specific ones, such as K-Means threshold. However, only the major parameters were found to be worth tuning. These are epsilon and minimum sample count for DBScan and k for K-Means.

Below are several graphs that show the differences in performance of the models using different parameters. Green points represent true negatives, red - true positives, blue - false positives, black - false negatives. It is clearly visible that one model resulted in more false positives, while the other had some false negatives.

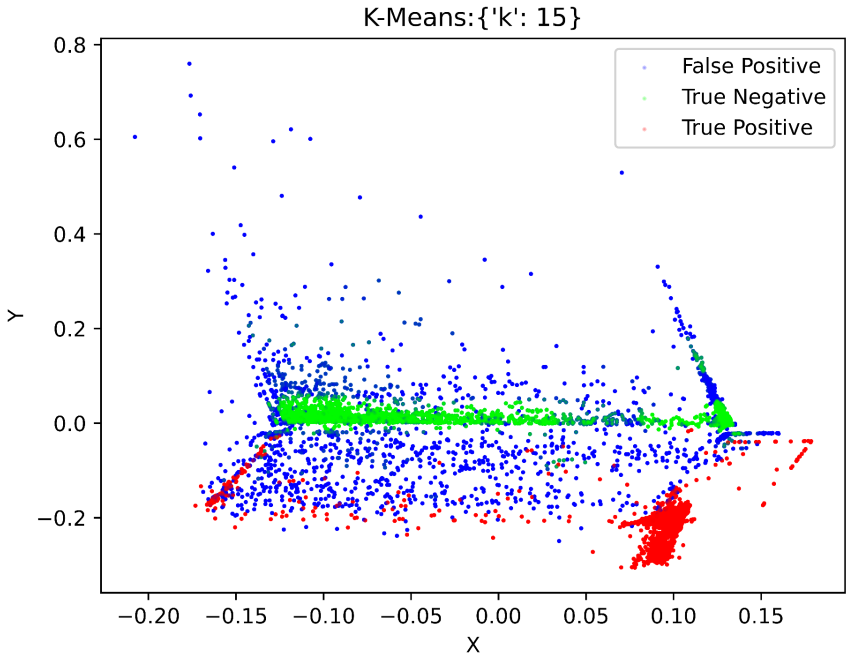

Fig 8. K-Means result with k = 15.

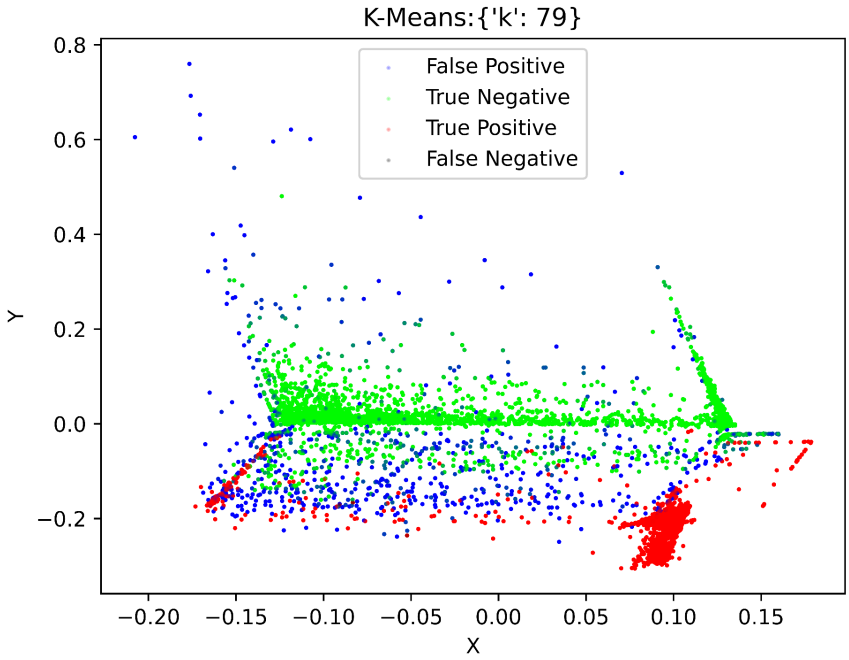

Fig 9. K-Means result with k = 79.

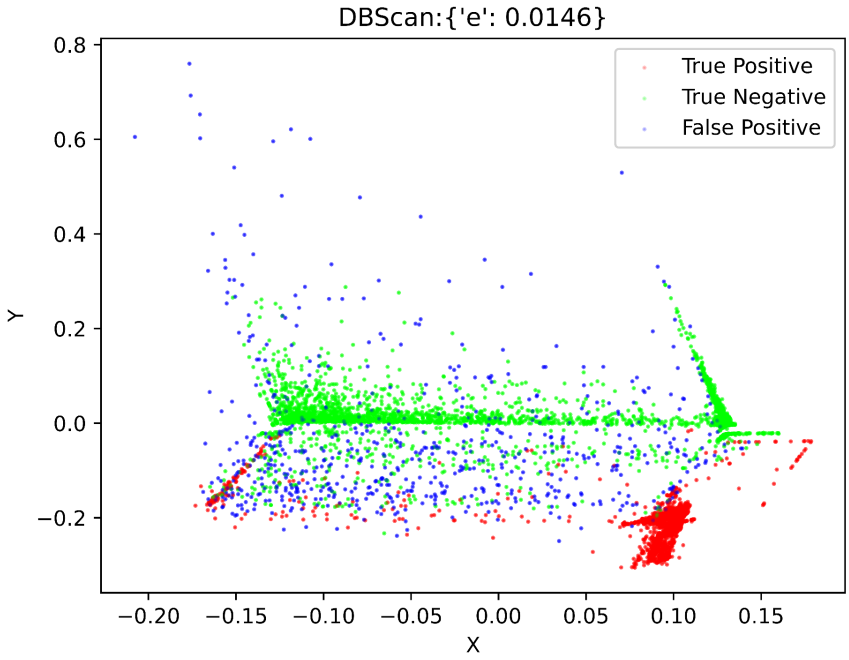

Fig 10. DBScan with e = 0.0172 and min = 2

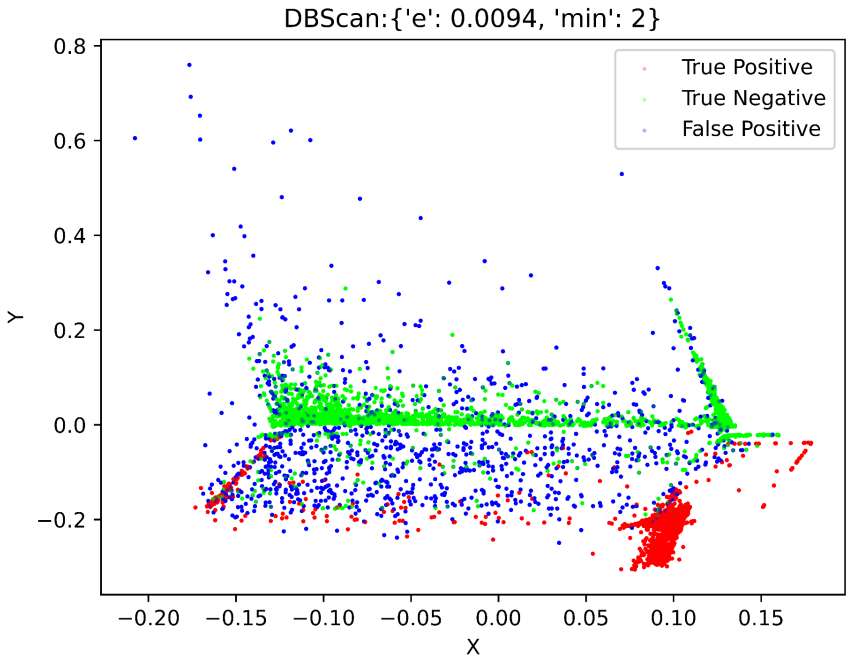

Fig 11. DBScan with e = 0.0094 and min = 2

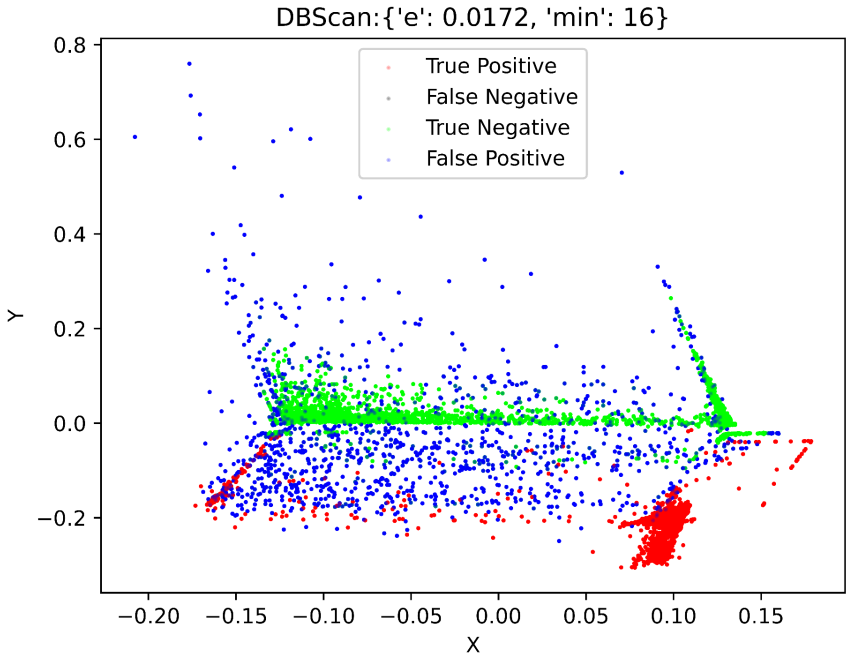

Fig 12. DBScan with e = 0.0172 and min = 16

Another value that was tested is the dimension that the data was reduced to. The initial data has 41 features. It was found that reducing that to 18 features yields the best results.

Multiple results could be considered "the best". The final ones that were kept are provided at the start of the report. The metrics that were used when deciding which hyperparameters to keep are described in more detail in the "Performance Analysis" section.

K-Means had two results that could be applicable in different situations. The one that was used (k=79) provides a lower false positive rate of about 14% but incorrectly clusters very few samples, resulting in about 0.02% of false negatives. With a lower k in the range from 10 to 15, the algorithm performs much worse in terms of false positives, having a false positive rate of about 40%. However, it always has a perfect 100% false negative rate. If false negatives are absolutely critical, this lower value should be preferred.

DBScan only showed one "best". It was quickly found that only minimum = 2 results in a false negative rate of 0% - any other minimum brought some false negatives. Initially, before brute-force was ran, all values except epsilon were hand-picked. Epsilon was calculated using the elbow plot method - the K-Distance graph was created by

determining the distance matrix for all points in the training data. The matrix was sorted by distance for each point. The k-distance data was then sorted in an ascending order to be plotted with each point on the X-axis and epsilon values on the Y-axis. The epsilon value was determined by looking at the "elbow" of the graph. Using this method, the best epsilon value was determined to be 0.015, which was incredibly close to the final best value of 0.0146.

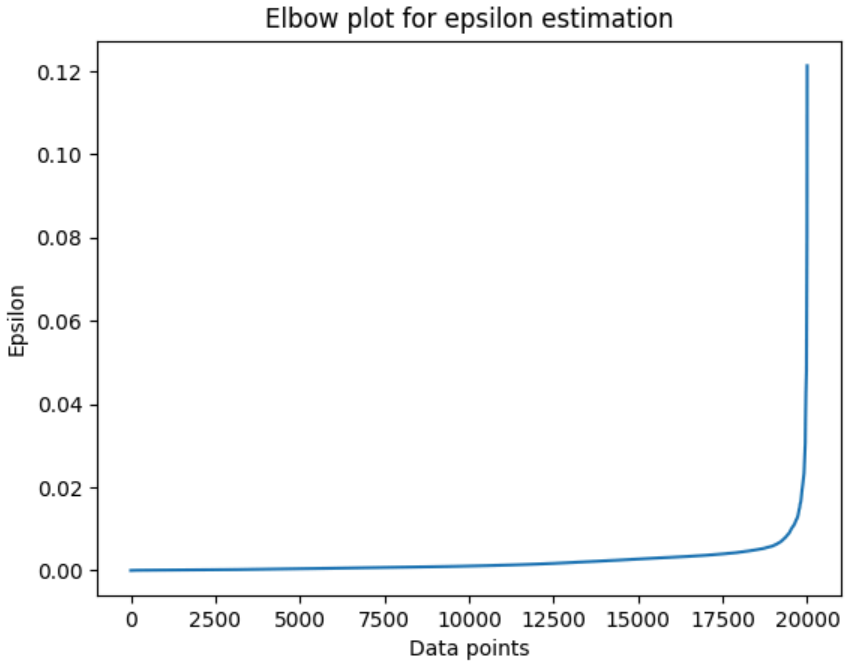

Fig 13. Elbow plot for epsilon value estimation

It is also interesting to note that DBScan always performs the same with the same hyperparameter values. K-Means, on the other hand, has a small margin of error. This is due to the fact that K-Means contains some randomness in the form of initial centroids.

Clustering Algorithms

The DBScan clustering algorithm is sensitive to the epsilon and minimum neighbor samples hyperparameters. The epsilon parameter defines the maximum distance from a sample point to another point for that point to be considered a neighbor. The minimum neighbor samples parameter defines the minimum number of neighbors a point must have to be added to a cluster. Appendix A contains a small part of the table showing brute-force tests iterating over both parameters. '*DBScan:e=0.0001,min=40:100.0*' - this is a single row from the table. The last value (in this case, 100.0) represents the score that the model got. The score was calculated using the following formula:

FPR_Percentage + False_Negative_Count * 1000. The lower the score is, the better a model performs.

Both algorithms perform well. A detailed overview on how a model's score was determined was provided above. The model needs to be selected based on corporate goals. If false negatives are absolutely detrimental and must be avoided at all costs, DBScan is the model of choice. If a small amount of false negatives is allowed, K-Means performs better because it results in less false positives.

Performance Analysis

Since the testing data was separated into normal and attack, it was possible to determine whether a sample was true positive, true negative, false positive or false negative. These values, in turn, enable further calculations of true positive rate, false positive rate, accuracy and F1 score.

When searching for the best hyperparameter values, false positive rate and false negative count were considered the most important values. Since the dataset contained relatively few samples, it was decided directly looking at the false negative count was more important than the true positive rate. However, this report talks about true positive rate instead of direct false negative count because the rate shows more information for larger datasets.

True positive rate is the most important metric for intrusion detection systems. Any missed anomaly may result in huge system breaches and other very negative outcomes. This score needs to be as high as possible - high values represent lower false negative counts.

False positive rate is less important but still very desirable. This metric represents false positive counts. A system identifying too much benign data as malicious results in fatigue. The security team needs to investigate every alert - if there are too many alerts, they become meaningless.

It is quite hard to tell if a clustering algorithm has overfit or underfit issues. It is not possible to make a learning curve, which is usually the main way of determining these issues. However, according to Matilda Sarah, it is somewhat possible to detect these issues based on the clusters. Highly separated clusters, clusters of very unusual shapes or incredibly small clusters may mean overfitting. None of these are visible on the graphs shown in figures 1 and 2. Underfitting may be identified by generalized clusters or poor separation between clusters. This might be the case for the models from this project. As seen on the graphs, the models incorrectly cluster a lot of normal data.

Appendix A

DBScan:e=0.0001,min=40:100.0
DBScan:e=0.0001,min=50:100.0
DBScan:e=0.0001,min=60:100.0
DBScan:e=0.0001,min=70:100.0
DBScan:e=0.0001,min=80:100.0
DBScan:e=0.0001,min=90:100.0
DBScan:e=0.0011,min=40:100.0
DBScan:e=0.0011,min=50:100.0
DBScan:e=0.0011,min=60:100.0
DBScan:e=0.0011,min=70:100.0
DBScan:e=0.0011,min=80:100.0
DBScan:e=0.0011,min=90:100.0
DBScan:e=0.0021,min=40:97.39999999999999
DBScan:e=0.0021,min=50:98.32499999999999
DBScan:e=0.0021,min=60:98.875
DBScan:e=0.0021,min=70:99.1
DBScan:e=0.0021,min=80:99.55000000000001
DBScan:e=0.0021,min=90:99.75
DBScan:e=0.0031,min=40:93.45
DBScan:e=0.0031,min=50:94.15
DBScan:e=0.0031,min=60:94.975
DBScan:e=0.0031,min=70:95.875
DBScan:e=0.0031,min=80:97.02499999999999
DBScan:e=0.005,min=40:84.65
DBScan:e=0.005,min=60:88.775
DBScan:e=0.005,min=80:91.07499999999999
DBScan:e=0.005,min=100:92.77499999999999
DBScan:e=0.006,min=40:79.525
DBScan:e=0.006,min=60:84.425
DBScan:e=0.006,min=80:87.35000000000001
DBScan:e=0.006,min=100:89.55
DBScan:e=0.007,min=40:75.64999999999999
DBScan:e=0.007,min=60:80.375
DBScan:e=0.007,min=80:83.35000000000001
DBScan:e=0.0149,min=2:1016.15
DBScan:e=0.015,min=2:1015.975
DBScan:e=0.0147,min=3:19.725
DBScan:e=0.0148,min=3:19.475
DBScan:e=0.0149,min=3:19.3
DBScan:e=0.017,min=3:1016.575
DBScan:e=0.017,min=5:1021.35
DBScan:e=0.017,min=2:1013.725
DBScan:e=0.0171,min=2:1013.65
DBScan:e=0.0172,min=2:1013.475

References

A Comprehensive Guide to Cluster Analysis: Applications, Best Practices and Resources. Matilda Sarah. https://www.displayr.com/understanding-cluster-analysis-a-comprehensive-guide/#interpreting-and-visualizing-cluster-analysis-results