Unsupervised Learning Approaches for Ransomware Detection using File Attribute Analysis

Reena Sajad Hyder, Tala Talaei Khoei, Reemaa Sajad Hyder, Peizhi Yan
Khoury College of Computer Sciences, Roux Institute Northeastern University, Portland ME, USA

Abstract—**Ransomware is currently a major global cybersecurity issue, with many organizations experiencing attacks, causing significant financial damage and disruption. According to recent data, nearly two-thirds of companies were hit by ransomware in the past year. So, detecting ransomware is a field with a lot of potential. A lot of techniques have been used to detect ransomware. This paper compares the performance of five different unsupervised machine learning algorithms in detecting ransomware files. The models are K Means Clustering, Density Based Clustering of Applications with Noise (DBSCAN), Gaussian Mixture Model (GMM), Hierarchical Clustering, and Autoencoder. The performance comparison is made in terms of Homogeneity, Completeness, V Measure, Calinski-Harabasz Score and Silhouette score. Our results show that DBSCAN has the highest Silhouette Score, which means it forms well-defined clusters, while the Autoencoder with K-Means has the highest Calinski-Harabasz Score, showing strong cluster separation. Both DBSCAN and Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) perform well in terms of homogeneity, and GMM provides a good balance between homogeneity and completeness with its V-Measure. Overall, autoencoder offers a promising solution for ransomware detection.**

Keywords—Machine Learning, Unsupervised Learning, Ransomware Detection, K Means Clustering, HDBSCAN, DBSCAN, GMM, Hierarchical Clustering, Autoencoder

I. Introduction

Over the past three years, ransomware has become one of the biggest cyber scams to hit businesses. Indeed, the FBI estimates that losses incurred in 2016 due to ransomware will top $1bn. Ransomware is malicious software that allows a hacker to restrict access to an individual's or company's vital information in some way and then demand some form of payment to lift the restriction. The most common form of restriction today is encryption of important data on the computer or network, which essentially lets the attacker hold user data or a system hostage [1].

The Ransomware variants are classified as Crypto and Locker ransomware. The Crypto ransomware uses a crypto algorithm to encrypt all the user data files and demand ransom to share the key for decryption. The Locker ransomware uses privilege escalation technique through various management applications and restricts the access of resources to the users. The persistence techniques are adopted by the Locker ransomware to ensure the resources are locked even after reboot[2]. Over the past two years, Symantec has identified at least 16 different ransomware variants. That is, 16 different families of malware, the majority of which are developed independently by competing criminal gangs. To be clear, each variant is not merely version 2.0, or 3.0 of the same malware, but completely separate, developed by different people and using different messages. The variants all operate in a similar manner, locking a computer screen and displaying a message purporting to be from a law enforcement agency.[3]So, it is important to detect ransomware attacks to reduce the possible damage.

This study uses the Ransomware Detection dataset which contains different attributes about ransomware files and legitimate files such as Name, Machine, SizeOfInitializedData, SizeOfUninitializedData etc. We use unsupervised machine learning models to detect if the file is a ransomware file or not. We also compare the performances of these models. The five models we have used are: K Means Clustering, DBSCAN, GMM, Hierarchical Clustering, and Autoencoder. In Autoencoder, we do dimensionality reduction using Autoencoder and then do K Means Clustering.

Several studies have been conducted to detect ransomware files using machine learning models. In [4], the authors generated the classification model using the generated vector with weights. This detection model based on six machine learning algorithms is used to classify unknown binary samples into ransomware, malware, or benign files. They use six different machine learning models (all supervised) and get an accuracy between 97% and 98.65%. This paper focused on API invocation sequences and proposed a ransomware detection method with our proposed CF-NCF and machine learning algorithms.

In [5], the authors conducted two experiments to analyze the effectiveness of applying machine learning algorithms for ransomware detection. They applied seven classification methods on CICAndMal2017 dataset in each experiment. The results show that Random Forest is the best classifier in both experiments and in general tree-based classification algorithm is effective for ransomware detection. Their findings imply that the performance of the classifiers are not varied significantly when they are trained on each family distinctly.

In [6], the authors used Random Forest and XG Boost algorithms in a framework based on feature selection to effectively classify and identify ransomware. They conducted multiple trials to assess the efficacy of the models using an arsenal of ransomware samples. The experimental findings showed that the Random Forest classifier conducted better than other classifiers, consistently attaining the greatest accuracy, F-beta, and precision scores.

In [7], the authors get the ransomware and benign binaries dataset from the Feature Generation Engine, which is the starting component for the machine learning model. Experimental results show that among eight supervised machine learning classifiers their framework could achieve an accuracy of more than 90% (96.5% on average) for seven of them for combined feature dataset and assembly level instruction dataset.

Other authors did different feature selection and applied semi-supervised classification methods to the CICAndMal 2017 dataset for analyzing the ransomware and the semi -supervised classification method using the random forest as a base classifier outperforms the various semi-supervised classification techniques for ransomware detection. They applied the framework with all the experiments on a ransomware dataset and evaluated the models' performance by a robust comparative analysis among DT, RF, NB, LR, and NN classifiers. The experimental results demonstrate that the Random Forest classifier outperformed other classifiers by achieving the highest accuracy, F -beta, and precision scores with reasonable consistency in the 10- fold cross-validation results.[8]

In this research [9], the authors collected the generate ransomware data-set from RISS of ICL machine learning online repository. In this work they proposed a Supervised Machine learning framework for detection of ransomware malware. The support vector Machine classify has proven a better performance with less root mean square error RMSE) of 0.179 and accuracy of 88.2% that outperform other algorithms. In [10], the authors used four semi-supervised learning methods were investigated namely Chopper, YATSI, Collective IBK, and Collective Wrapper on the ransomware datasets from CICAndMal2017 dataset (including 10 ransomware families). The results show that the Wrapper RF classification and Chi-squared or OneR feature selection methods are very effective in semi-supervised ransomware detection. The main limitation of this research is that the applied feature selection methods are supervised. They applied the Simplified Silhouette Filter (SSF) unsupervised feature selection, but the results were very poor.

All the above studies have decent accuracy but primarily only use supervised and semi supervised learning algorithms. This study explores the idea of using unsupervised machine learning models for detecting ransomware files. It also evaluates the performance of each of these models on diverse metrics such as Completeness, Homogeneity, V Measure, Calinski-Harabazs Score and Silhouette Score, to determine the efficacy of the models. The dataset used is also appropriately preprocessed to improve the accuracy of the models. The dataset used has already been used to detect ransomware files [11]. The author used supervised learning algorithms (Random Forest). This study, on the other hand, is focused on unsupervised learning models.

The remainder of the paper is as follows: Section II contains the materials and methods (which includes data exploration, data preprocessing and applying the models), Section III contains the results of the performance of each of the models and Section IV contains the conclusions and future scope of the study done.

Identify applicable funding agency here. If none, delete this text box.

II. MATERIALS AND METHODS

This section describes the proposed framework, the dataset description, the data preprocessing performed, feature selection, classification models and metrics.

A. Proposed Framework

The proposed framework comprises of detecting ransomware files using three different unsupervised machine learning algorithms namely K Means Clustering, DBSCAN, GMM, Hierarchical Clustering and Autoencoder with K Means. These are all clustering models and they are used for grouping data points based on their similarities. These models group the dataset into two clusters: ransomware files and legitimate files [11]. These models are evaluated and their performances are compared. They are discussed in greater detail in subsequent sections.

B. Data Description

This study makes use of the Ransomware Detection dataset [11] which contains file properties of a large number of files (some of which are ransomware and some of which are legitimate or benign). It has a number of technical attributes derived from different stages of the software development process, including source code development, compilation, linking, and final Portable Executable (PE) file creation. These attributes are collected using PE File Analysis Tools.

It comprises of 138,047 rows and 57 columns, consisting of both numerical and categorical attributes. These attributes are important in determining whether a file is legitimate or potentially malicious, such as ransomware. Table 1 has all the attributes along with their definitions.

TABLE I. ATTRIBUTES AND THEIR DEFINITIONS

Attribute	Definition
Characteristics	Flags indicating properties of the file (whether it is executable or dynamic link library)
MajorLinkerVersion	Version of the linker used to create the executable
SizeOfCode	Size of the code section
SizeOfInitializedData	Sizes of initialized data sections.
AddressOfEntryPoint	Memory address where program execution begins.
Base of Data	This is the relative virtual address (RVA) of the beginning of the data section when the executable is loaded into memory.
ImageBase	Preferred memory address for loading the executable
MajorOperatingSystem Version	Minimum version of the operating system required to run the executable
MinorImageVersion	Minor - smaller updates/ revisions
SizeOfImage	The total size of the image, including all headers and sections, when loaded into memory.
CheckSum	A checksum is a value that represents the number of bits in a transmission message
SubSystem	Specifies the environment required to execute the file, such as Windows GUI, Windows Console, or a device driver.
DLLCharacteristics	These are the characteristics defined in the PE Header for Windows executables.
SizeOStackReserve	This indicates the amount of memory reserved for the stack in bytes. The stack is used for managing function calls and local variables.

Attribute	Definition
SectionsMeanEntropy	The average entropy (randomness) of all sections in the file. Higher entropy may suggest encryption or obfuscation.
SectionsMinEntropy	The lowest entropy value among the sections in the file, indicating the section with the least randomness.
SectionsMaxEntropy	The highest entropy value among the sections in the file, indicating the section with the most randomness.
SectionsMeanRawsize	The average size (in bytes) of all sections in the file when stored on disk.
SectionsMinRawsize	The smallest size (in bytes) of any section in the file when stored on disk.
SectionMaxRawsize	The largest size (in bytes) of any section in the file when stored on disk.
SectionsMeanVirtualsize	The average size (in bytes) of all sections in memory when the file is loaded.
SectionsMinVirtualsize	The smallest size (in bytes) of any section in memory when the file is loaded.
SectionMaxVirtualsize	The largest size (in bytes) of any section in memory when the file is loaded.
ImportsNb	The total number of imported functions from all DLLs.
ExportNb	The number of functions or symbols exported by the executable, making them available for use by other executables or libraries.
ResourcesNb	The number of resources (such as icons, images, strings) embedded in the executable.
ResourcesMeanEntropy	The average entropy (randomness) of all resources, indicating how compressed or encrypted they are.
ResourcesMinEntropy	The lowest entropy among the resources, indicating the least compressed or encrypted resource.
ResourcesMaxEntropy	The highest entropy among the resources, indicating the most compressed or encrypted resource.
ResourcesMeanSize	The average size (in bytes) of all resources.
ResourcesMinSize	The smallest size (in bytes) among the resources.
ResourcesMaxSize	The largest size (in bytes) among the resources.
VersionInformationSize	The size (in bytes) of the version information structure, which holds details about the software version, such as the product name and version number.
legitimate	This value denotes whether it is a legitimate file or a ransomware file. 1 means legitimate and 0 means ransomware

C. *Data Preprocessing*

Data preprocessing is a very crucial step in running unsupervised models because it directly affects the quality of the models. The data preprocessing carried out in this study is as follows.

The first step is balancing the dataset. Before the preprocessing, the dataset has more ransomware files than legitimate files. This will skew the results, so balancing is performed. Initially, there are 96724 rows of ransomware files and 41323 rows of legitimate files. We under sample the ransomware data using the resample method and get 50:50 distribution.

Next step in preprocessing is imputation. Imputation is the process of checking null values and dealing with it. Imputation is important because missing values will lead to an unreliable model and will introduce a bias. In this dataset, there are no null values. So, there is no need to perform imputation.

Encoding is the next step in the process. But this dataset contains only one non numerical column (the .exe filename) which is not necessary for the prediction process. Following this, we check the data distribution pattern for the numerical columns by calculating the skewness of the columns. If it is normally distributed, then standardization is done. If the distribution is significantly skewed, then Yeo Johnson power transformation is done. If it doesn't fall into either of those categories (not normally distributed and requires scaling), then normalization is done. Now preprocessing is complete, and we can select features to train the models.

D. *Feature Selection*

Feature selection, as a data preprocessing strategy, has been proven to be effective and efficient in preparing data (especially high-dimensional data) for various data-mining and machine-learning problems. When data-mining and machine-learning algorithms are applied on high-dimensional data, a critical issue is known as the curse of dimensionality. It refers to the phenomenon that data become sparser in high-dimensional space, adversely affecting algorithms designed for low-dimensional space.[12]

In this study, we use correlation matrices and mutual information values of the attributes to select features. For calculating the correlation matrices, we use Pearson, Kendall and Spearman correlation methods. These are used to calculate the correlation between different predictor attributes. Mutual Information is used to calculate how closely related the predictor attributes are to the target variable. It is a non-negative value, which measures the dependency between the variables. It is equal to zero if and only if two random variables are independent ,and higher values mean higher dependency. We set threshold values and drop those attributes that are very closely related to the target variable (have very high Mutual Information Value).

E. *Classification Models*

Unsupervised pretraining, which learns a useful representation using a large amount of unlabeled data to facilitate the learning of downstream tasks, is a critical component of modern large-scale machine learning systems.[14]. Unsupervised models are helpful in discovering new patterns that supervised models may have missed. So, we have carried out clustering using five different unsupervised machine learning algorithms. They are K Means Clustering, DBSCAN, GMM, Gaussian Mixed Model, Hierarchical Clustering and Autoencoder with K Means.

We chose these models because they each offer unique strengths. K Means Clustering is a widely used method because it is fast and scalable and works well with different types of data. DBSCAN, a density-based clustering algorithm, offers robustness in identifying clusters of various shapes and handling noisy data patterns. GMM are flexible and can capture complex patterns in data. Hierarchical Clustering is easier to apply compared to other models since there are no parameters to specify. The combination of Autoencoders with K Means captures more complex structures.

The selected features are used to train the five models and the clusters are also visualized. Since unsupervised models need unlabeled data, we remove the labels and then run the models.

F. *Metrics*

The performances of the models are evaluated using five key metrics namely Completeness, Homogeneity, V-Measure, Calinski-Harabazs Score and Silhouette Score.

- Completeness: Completeness measures how well all the data points that are members of a given ground-truth class are assigned to the same cluster. A clustering result satisfies completeness if all the data points of a class are assigned to the same cluster.
- Homogeneity: Homogeneity measures how pure the clusters are, i.e., if each cluster contains only members of a single class. A clustering result satisfies homogeneity if all of its clusters contain only data points that are members of a single class.
- V-Measure: V-Measure is the harmonic mean of Homogeneity and Completeness. It provides a balance between these two scores, ensuring that clusters are homogeneous while also being complete.
- Calinski-Harabasz Score: It evaluates the ratio of the sum of between-cluster dispersion and the sum of within-cluster dispersion. A higher score indicates better-defined clusters.
- Silhouette Score: Silhouette Score measures how similar each point in one cluster is to points in its own cluster (cohesion) compared to points in other clusters (separation). The score ranges from -1 to 1, where a high score indicates that the data points are well clustered.

III. RESULTS

The ransomware dataset is cleaned and pre-processed by performing standardization, normalization and power transformation. The dataset has 57 attributes. So, feature selection is performed following data cleaning. The feature selection is done with the help of correlation matrix and mutual information. The dataset now has 34 attributes.

In this study, we apply two approaches to feature selection. First is calculating the correlation matrix with the help of Spearman, Pearson and Kendall methods and plotting heat maps for these matrices. We then set a threshold value and determine pairs of attributes that are highly correlated. We find common pairs between the three correlation matrices (Pearson, Spearman and Kendall). Of the attributes in the pairs, we drop the attribute with higher variance and retain the attribute with lower variance.

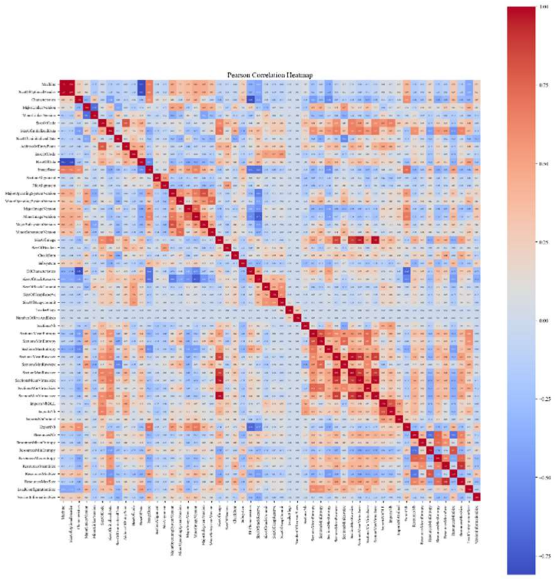

Fig 1. Pearson Correlation Coefficient Heatmap

Next, we use mutual information technique to select features. It is used to measure the dependency or shared information between two different attributes. Here, we calculate the mutual information between all the predictor attributes and the target attribute. Any attribute that is highly correlated with the target attribute is dropped. We experiment with three different values for mutual information (0.01, 0.1 and 0.25) and finally set the threshold value to 0.25 and drop all attributes that have mutual information value higher than 0.25. This process resulted in isolating 34 different attributes for running the unsupervised models.

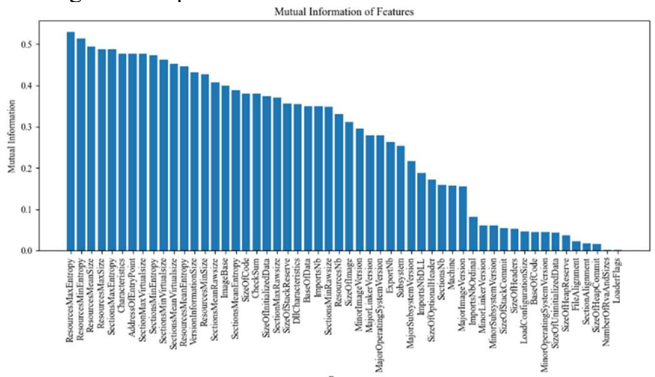

Fig 2. Mutal Information Values for all features

This data is now used to train the models. The features are extracted and the labels are removed since the unsupervised models require unlabelled data. We also store the evaluation metrics for each model in a dictionary to compare their performances. We then define a method to visualize the clusters and a method for evaluating the five performance metrics for the models.

For K means clustering, we fit the K Means Model to the dataset, predicting two clusters (each data point is assigned to one of the two clusters) and compare the results of the model with the actual labels. We also visualize the cluster and evaluate the model. The second model performs DBSCAN clustering on the dataset with the following parameters (eps = 0.5 and min_samples = 5). The performance of this model is then evaluated. The third model applies GMM Clustering to the dataset assuming two clusters. It predicts cluster labels and compares it with the actual labels. The next model performs HDBSCAN clustering on the dataset after applying Principal Component Analysis (PCA) for dimensionality reduction and standardization. It reduces the data to two components and fits the HDBSCAN model to detect clusters and compares the predicted clusters with the actual labels. The fifth model builds and trains autoencoder to perform dimensionality reduction on the given dataset. It reduces the data to 10 dimensions and then K Means Clustering is applied. The significance of using autoencoder is to compress the data, capture the key features before doing the clustering.

These are the clusters for the five models:

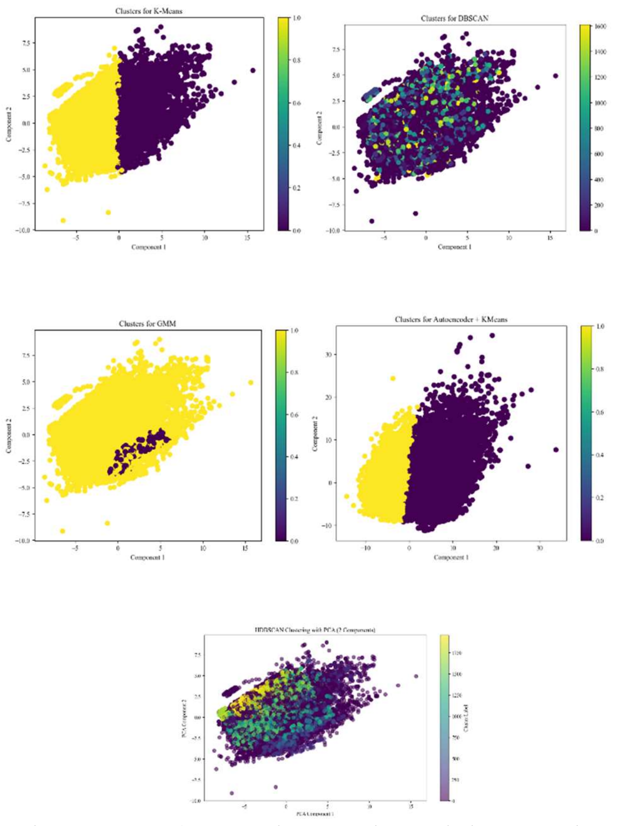

Fig 3. From L to R(K Means Cluster, Density Based Cluster, Gaussian Mixture Model Cluster, Hierarchical Cluster, Autoencoder Cluster)

The performances of all the models can be tabulated as follows:

TABLE II. PERFORMANCE METRICS

Models	**Completeness**	**Homogeneity**	**V-Measure**	**Calinski Harabasz Score**	**Silhouette Score**
K-Means	0.245713	0.244775	0.245243	26180.676728	0.252765
DBSCAN	0.126379	0.812995	0.218753	130.034945	0.394153
GMM	0.577845	0.553568	0.565446	16461.336370	0.193111
Hierarchical Clustering	0.095171	0.806326	0.170248	123.915077	0.323115
Autoencoder + K Means	**0.554106**	**0.553025**	**0.553565**	**38996.457273**	**0.324121**

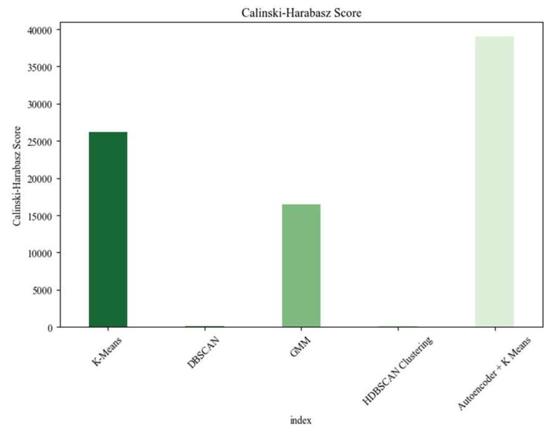

Fig 4: Comparison of Calinski Harabazs Scores of the five models

In terms of the Calinski Harabazs Score, the Autoencoder + K-Means combination achieved the highest score, showing that this approach found the most compact and well-separated clusters. K Means performed well but not as well as the Autoencoder. DBSCAN and Hierarchical Clustering had very ow scores.

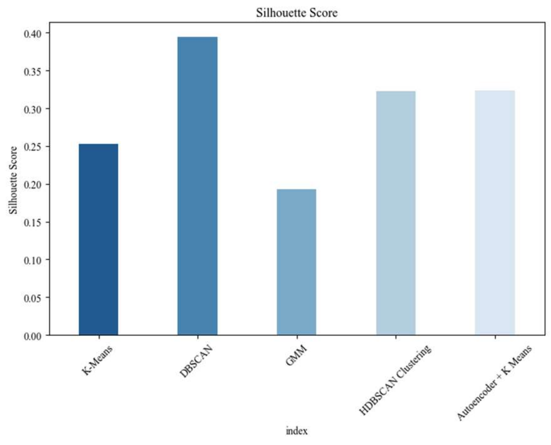

Fig 5. Comparison of Silhouette Scores of the five models

In terms of Silhouette Score, DBSCAN achieved the highest score, showing it found well-defined clusters in terms of distance between points. HDBSCAN and Autoencoder + K-Means also performed well. K- Means has a reasonable score. GMM has the lowest score, showing that the clusters are not very distinct and overlapping.

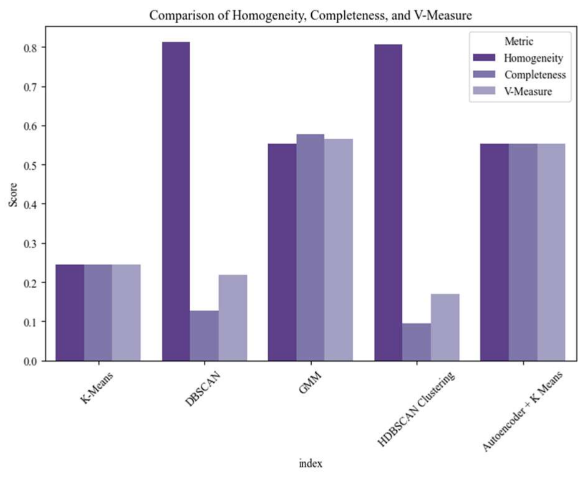

Fig 6: Comparison of Homogeneity, Completeness and V-Measure of the models

DBSCAN and HDBSCAN have high Homogeneity scores, showing that their clusters contain mostly points from the same true class. However, they fall short in Completeness, meaning they fail to group entire classes together. Autoencoder + K-Means offers a strong balance between Homogeneity and Completeness, performing well across all three metrics. This indicates that it is the best model overall in aligning with the underlying class structure. GMM performs moderately across all metrics, while K-Means struggles to form meaningful clusters in terms of class labels.

IV. CONCLUSION

This study provides a detailed comparison of five clustering algorithms namely K-Means, DBSCAN, GMM, Hierarchical Clustering, and Autoencoder + K-Means, based on a range of clustering performance metrics. The findings highlight the superior capability of the Autoencoder + K-Means model in producing well-defined, meaningful clusters. This model achieves strong Completeness, Homogeneity, and V-Measure scores, indicating balanced and coherent clusters, while also recording the highest Calinski-Harabasz score, suggesting excellent cluster separation

While the GMM model performs reasonably well, its lower Silhouette score indicates overlapping clusters, limiting its overall effectiveness. DBSCAN excels at creating compact, distinct clusters, but struggles with completeness, likely due to labeling many points as noise. K-Means delivers average performance without producing highly defined clusters, and Hierarchical Clustering shows the weakest performance.These results emphasize the potential of Autoencoder + K-Means as a powerful tool for clustering tasks, combining the dimensionality reduction of an autoencoder with the simplicity of K-Means to improve clustering quality. Future work could focus on extending this approach by experimenting with other dimensionality reduction techniques or combining autoencoders with more advanced clustering algorithms. Moreover, further exploration of model performance on larger and more complex datasets, as well as real-time clustering applications, could provide valuable insights into optimizing the trade-off between model accuracy and computational efficiency.

REFERENCES

[1] Brewer, Ross. (2016). Ransomware attacks: detection, prevention and cure. Network Security. 2016. 10.1016/S1353-4858(16)30086-1.

[2] T.R, Reshmi. (2021). Information security breaches due to ransomware attacks -A Systematic Literature Review. International Journal of Information Management Data Insights. 1. 10.1016/j.jjimei.2021.100013.

[3] O'Gorman, Gavin, and Geoff McDonald. Ransomware: A growing menace*.* Arizona, AZ, USA: Symantec Corporation, 2012. Available: https://www.01net.it/whitepaper_library/Symantec_Ransomware_Growing_Menace.pdf

[4] B. Niu, Z. Zhang, Z. Zhao, and L. Zhang, “A novel approach to measuring the robustness of complex networks,” 2021, DOI: 10.1002/cpe.5422. [Online]. Available: https://doi.org/10.1002/cpe.5422

[5] F. Noorbehbahani, F. Rasouli and M. Saberi, "Analysis of Machine Learning Techniques for Ransomware Detection," 2019 16th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC), Mashhad, Iran, 2019, pp. 128-133, Available: https://ieeexplore.ieee.org/document/8985139.

[6] A. B. Jadhav and S. M. Chikhale, “Ransomware detection and prevention using machine learning techniques,” *International Journal of Innovative Research in Science, Engineering and Technology*, vol. 12, no. 4, pp. 1738-1745, 2023. [Online]. Available: https://www.ijirset.com/upload/2023/april/169_Ransomware_NC.pdf

[7] S. Poudyal, K. P. Subedi and D. Dasgupta, "A Framework for Analyzing Ransomware using Machine Learning," *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, Bangalore, India, 2018, pp. 1692-1699, doi: 10.1109/SSCI.2018.8628743.

[8] M. Masum, M. J. Hossain Faruk, H. Shahriar, K. Qian, D. Lo and M. I. Adnan, "Ransomware Classification and Detection With Machine Learning Algorithms," *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, Las Vegas, NV, USA, 2022, pp. 0316-0322, doi: 10.1109/CCWC54503.2022.9720869.

[9] U. Adamu and I. Awan, "Ransomware Prediction Using Supervised Learning Algorithms," *2019 7th International Conference on Future Internet of Things and Cloud (FiCloud)*, Istanbul, Turkey, 2019, pp. 57-63, doi: 10.1109/FiCloud.2019.00016.

[10] F. Noorbehbahani and M. Saberi, "Ransomware Detection with Semi-Supervised Learning," *2020 10th International Conference on Computer and Knowledge Engineering (ICCKE)*, Mashhad, Iran, 2020, pp. 024-029, doi: 10.1109/ICCKE50421.2020.9303689.

[11] Mudit Mathur. "Ransomware Detection." GitHub repository, 2020. Available at: https://github.com/muditmathur2020/RansomwareDetection/tree/master

[12] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. 2017. Feature Selection: A Data Perspective. ACM Comput. Surv. 50, 6, Article 94 (November 2018), 45 pages. https://doi.org/10.1145/3136625

[13] Utkarsh Mahadeo Khaire, R. Dhanalakshmi, Stability of feature selection algorithm: A review, Journal of King Saud University - Computer and Information Sciences, Volume 34, Issue 4, 2022,Pages 1060-1073, ISSN 1319-1578, https://doi.org/10.1016/j.jksuci.2019.06.012.

[14] Lei Wang, Zongzhang Zhang, Peilin Zhao, Minjie Xu, Longbing Cao. "Online Variational Imitation Learning with Continuous Feedback." arXiv preprint arXiv:2303.01566, 2023. Available at: https://arxiv.org/abs/2303.01566

[15] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. 2017. Feature Selection: A Data Perspective. ACM Comput. Surv. 50, 6, Article 94 (November 2018), 45 pages. https://doi.org/10.114