

CAS Thesis in Advanced Machine Learning

Application of various Machine Learning models on real cybersecurity data
(BETH dataset)

Jamel Belgacem - Papa Moryba Kouate

14.06.2024

Contents

1	Introduction	2
2	Data analysis	3
2.1	Features	3
2.2	Correlation matrix	12
2.3	Event Frequency	13
3	Data preparation	14
3.1	Numerical data transformation	14
3.2	StackAddresses	14
3.3	Args	14
3.4	Ordinal encoding	14
3.5	Scaling	15
3.6	Smote	15
3.7	Shapelet discovery method	15
4	Models	16
4.1	Dense neural network	16
4.1.1	Model 1	16
4.1.2	Model 2	18
4.1.3	Model 3	20
4.1.4	Model 4	22
4.2	Convolutional neural network	24
4.2.1	Model 5	24
4.2.2	Model 6	26
4.3	LSTM neural network	28
4.3.1	Model 7	28
4.3.2	Model 8	30
4.3.3	Model 9	32
4.4	Transformer	34
4.5	Decision Tree Classifier with Shapelet Discovery method	36
5	Results	37
6	Discussion	37
7	References	39

1 Introduction

The BETH dataset addresses a critical need in cybersecurity research: the availability of real-world, labeled data for anomaly detection. Unlike synthetic datasets, BETH captures genuine host activity and attacks, making it a valuable resource for developing robust machine learning models [1].

The scale, diversity, and structured heterogeneity of BETH dataset makes it an invaluable resource for advancing anomaly detection techniques and enhancing the robustness of machine learning models in the cybersecurity domain.

Size and Composition of the dataset:

- BETH comprises over eight million data points collected from 23 hosts.
- Each host records both benign activity (normal behavior) and, at most, one attack.
- The dataset is diverse, reflecting various types of network traffic and system events.

Structured Heterogeneity of the dataset:

- BETH's features are highly structured but heterogeneous.
- This diversity mirrors the complexity of real-world cybersecurity data.
- Features include network traffic statistics, system logs, and process-level information.

Scale and realism of the BETH dataset:

- BETH is one of the largest publicly available cybersecurity datasets.
- It captures contemporary host behavior, including modern attacks.
- Researchers can use BETH to study the impact of scale on anomaly detection algorithms.

Behavioral Diversity:

- The dataset covers a wide range of activities, from routine tasks to malicious actions.
- Hosts exhibit different patterns, making BETH suitable for behavioral analysis.

Robustness Benchmarking:

- BETH enables evaluating the robustness of machine learning models.
- Researchers can assess how well their algorithms generalize to unseen attacks.
- It serves as a benchmark for novel anomaly detection techniques.

2 Data analysis

The Beth dataset represents more than 8 millions events collected over 23 honeypots, only nearly 1 million of it will be used on this project. Data are already divided into training, valadating and testing dataset (60% / 20% /20%).

2.1 Features

Each of this dataset (training, valadating and testing dataset) has features showed in the below table. We have also created charts that provide visual insight into the frequency distribution of various entities (like host names, process names, threadId, etc.) using logarithmic scaling and color gradients to highlight differences in counts.

FEATURE	TYPE	DESCRIPTION
TIMESTAMP	FLOAT	SECONDS SINCE SYSTEM BOOT
PROCESSID*	INT	INTEGER LABEL FOR THE PROCESS SPAWNING THIS LOG
THREADID	INT	INTEGER LABEL FOR THE THREAD SPAWNING THIS LOG
PARENTPROCESSID*	INT	PARENT'S INTEGER LABEL FOR THE PROCESS SPAWNING THIS LOG
USERID*	INT	LOGIN INTEGER ID OF USER SPAWNING THIS LOG
MOUNTNAMESPACE*	INT (LONG)	SET MOUNTING RESTRICTIONS THIS PROCESS LOG WORKS WITHIN
PROCESSNAME	STRING	STRING COMMAND EXECUTED
HOSTNAME	STRING	NAME OF HOST SERVER
EVENTID*	INT	ID FOR THE EVENT GENERATING THIS LOG
EVENTNAME	STRING	NAME OF THE EVENT GENERATING THIS LOG
ARGSNUM*	INT	LENGTH OF ARGS
RETURNVALUE*	INT	VALUE RETURNED FROM THIS EVENT LOG (USUALLY 0)
STACKADDRESSES	LIST OF INT	MEMORY VALUES RELEVANT TO THE PROCESS
ARGS	LIST OF DICTIONARIES	LIST OF ARGUMENTS PASSED TO THIS PROCESS
SUS	INT (0 OR 1)	BINARY LABEL AS A SUSPICIOUS EVENT (1 IS SUSPICIOUS, 0 IS NOT)
EVIL	INT (0 OR 1)	BINARY AS A KNOWN MALICIOUS EVENT (0 IS BENIGN, 1 IS NOT)

Figure 1: Features

- timestamp: time in seconds since system boot (float)
 - processId: id of the process spawning this log (integer)
- The bar chart shows the top 20 process IDs (PIDs) based on their count. The process labeled “Else” has the highest count, far exceeding the counts of any individual Process ID. PID 159 has the second-highest count, noticeably higher than the rest of the process IDs, though much lower than “Else”. Each colored bar represents a distinct process ID, with colors transitioning from dark purple for higher counts to lighter green for lower counts.

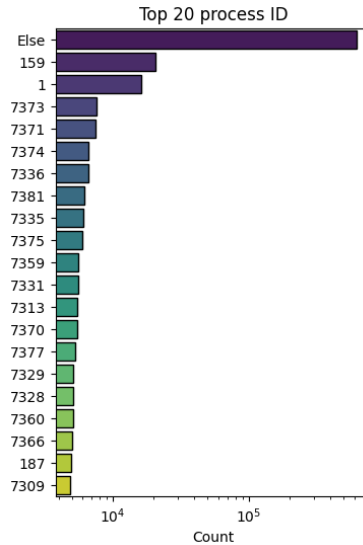


Figure 2: Process ID

- threadId: id of the thread (integer)

The thread IDs “159”, “1”, “7373”, “7371”, and “7374” follow in decreasing order. The counts are plotted on a logarithmic scale.

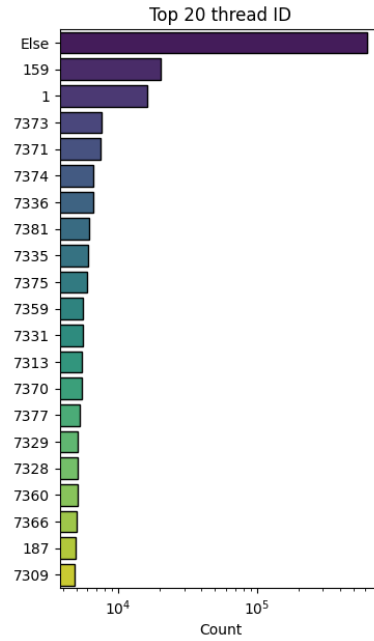


Figure 3: Thread ID

- parentProcessId: parent process id (integer) Parent process ID “187” has the highest count, nearing 100,000 occurrences. Other IDs such as “7099”, “1”, “1469”, “188”, and “1336” show decreasing counts. The counts are plotted on a logarithmic scale.

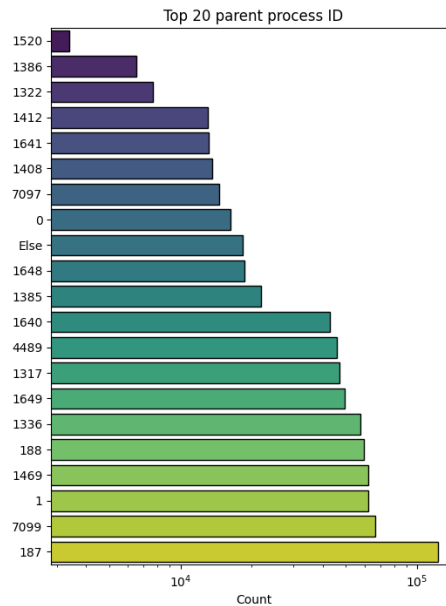


Figure 4: Parent process ID

- `userId`: login integer id (integer)
 - The dataset is highly imbalanced with respect to user IDs, particularly dominated by user ID “0.”
 - The presence of a few other user IDs with significantly lower frequencies indicates that the data might be heavily skewed towards certain users.

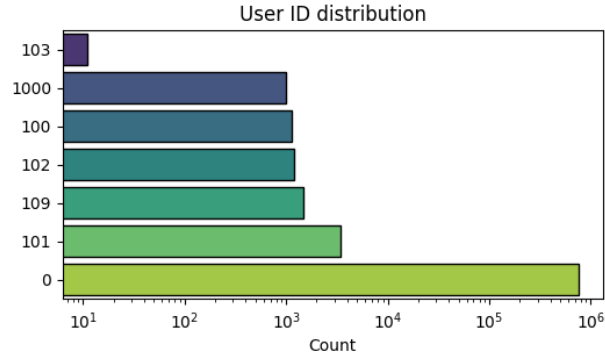


Figure 5: User ID

- `mountNamespace`: Set mounting restrictions this process log (integer)

The mount namespace with the label “4026531840” has the highest count, reaching more than 100,000 occurrences. Other labels such as “4026532217”, “4026532232”, “4026532229”, “4026532231”, and “4026532288” follow in descending order.

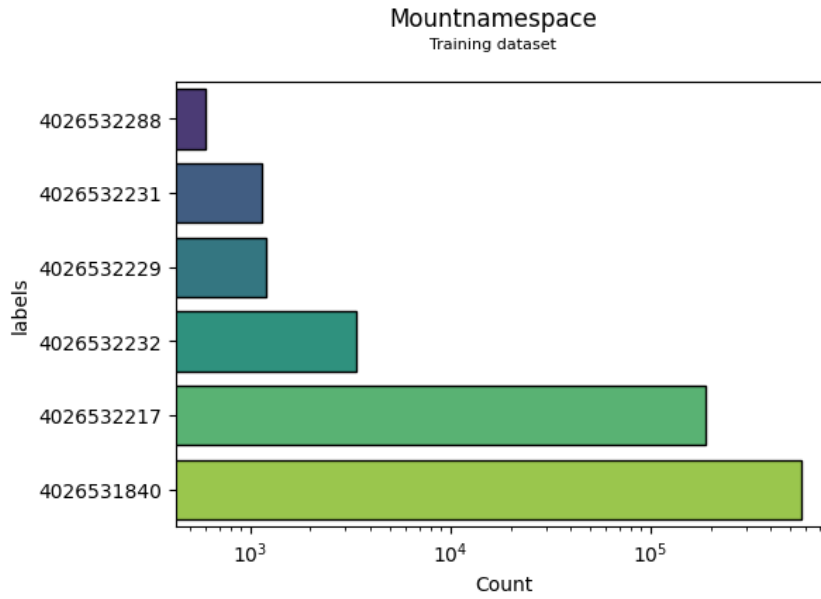


Figure 6: Mount name space

- processName: command executed (string)

The dataset is dominated by a few process names, with “ps” being the most frequent.

Other process names such as “systemd-udevd”, “sshd”, “systemd-journal”, and “systemd” follow in decreasing order. A large number of processes are included, some with very small counts compared to “ps”.

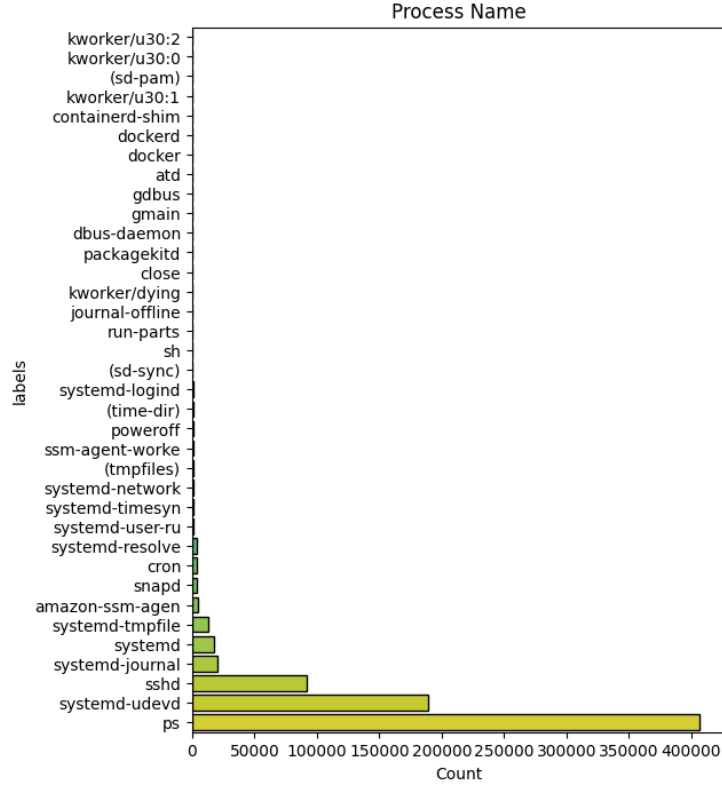


Figure 7: Process name

- hostName: host server (string)

The hostname “ubuntu” has the highest count, nearly reaching 200,000 occurrences. Other hostnames such as “ip-10-100-1-57”, “ip-10-100-1-120”, “ip-10-100-1-28”, “ip-10-100-1-55”, and “ip-10-100-1-173” show a descending count, with the least count observed for “ip-10-100-1-79”.

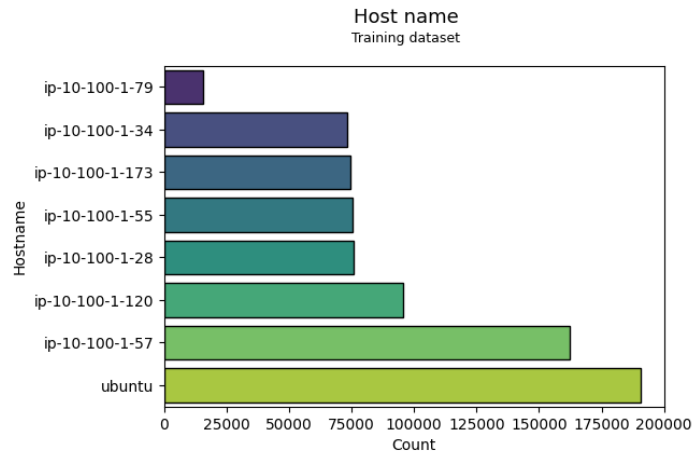


Figure 8: Host name

- eventId and eventName: id of the event generating this log (integer)
Event ID (or Event Name) “3” has the highest count, exceeding (10^5). Following IDs include “257”, “1005”, “4”, “21”, “1008”, and many others, with gradually fewer occurrences.

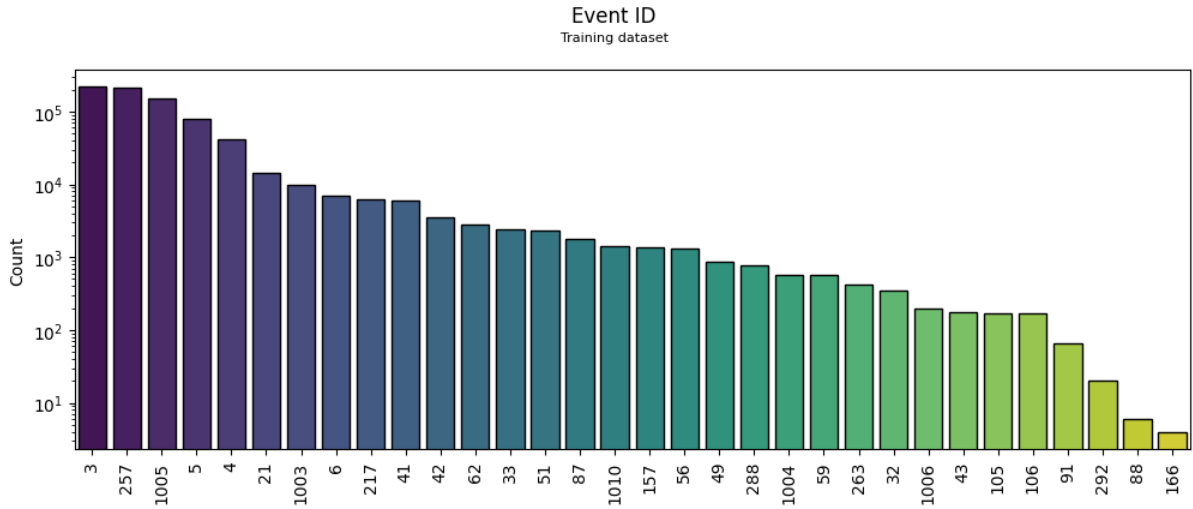


Figure 9: Event ID

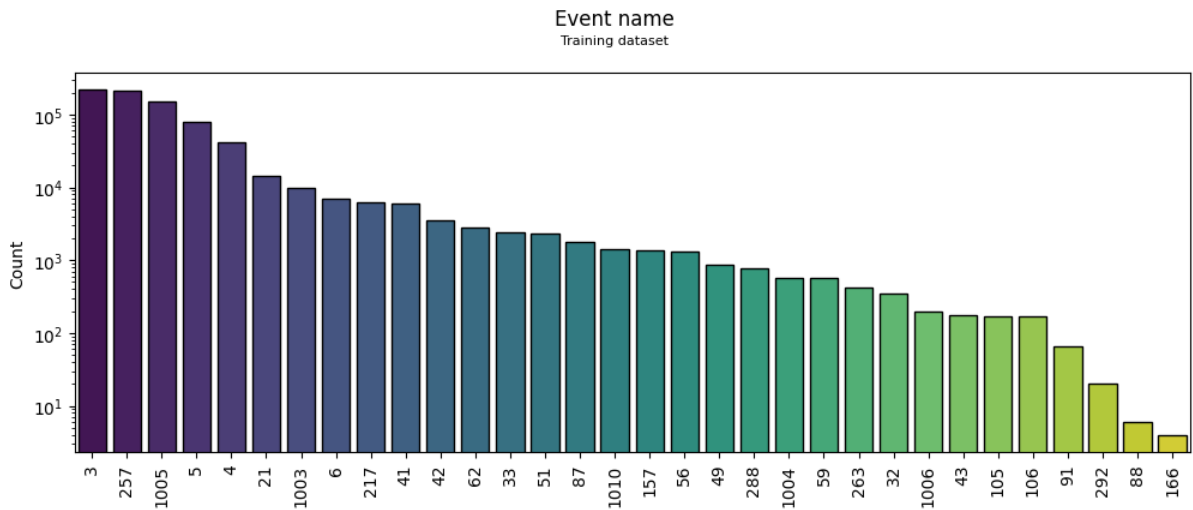


Figure 10: Event name

- `returnValue`: value returned from this event log (integer)

The `returnValue` of 0 is overwhelmingly dominant, with a count exceeding 500,000. This suggests that the majority of processes or functions in the dataset complete successfully without errors (assuming 0 indicates success).

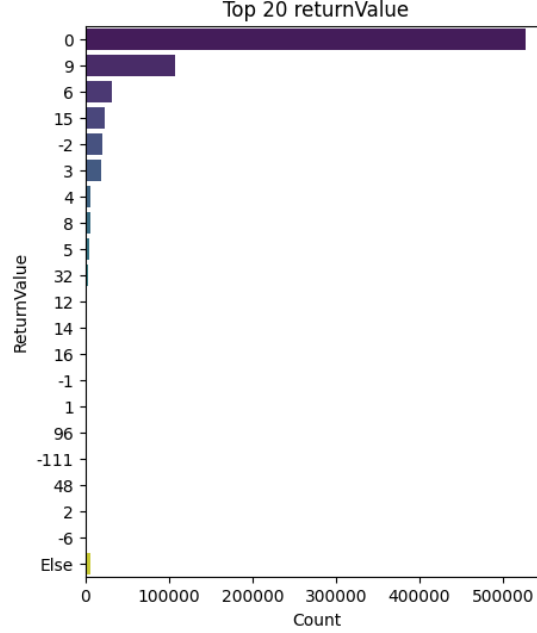


Figure 11: Return value

- `stackAddresses`: memory values relevant to the process (list of integer)
 - All three datasets (training, validation, and testing) have a similar distribution of stack addresses list lengths.
 - The most frequent stack addresses list length is 2, followed by 1 and 0.
 - There is a significant drop in frequency as the stack addresses list length increases beyond 3, with very few instances having lengths greater than 10.
 - The stack address “2048.0” is by far the most common, with a count exceeding 40,000.
 - The plot indicates a highly skewed distribution with a few stack addresses being extremely common and many others being relatively rare.

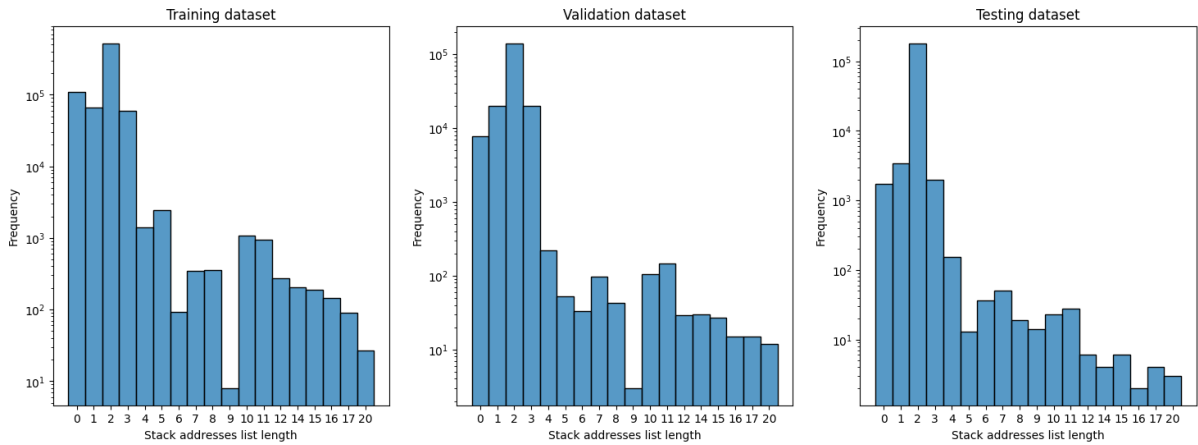


Figure 12: Stack length name

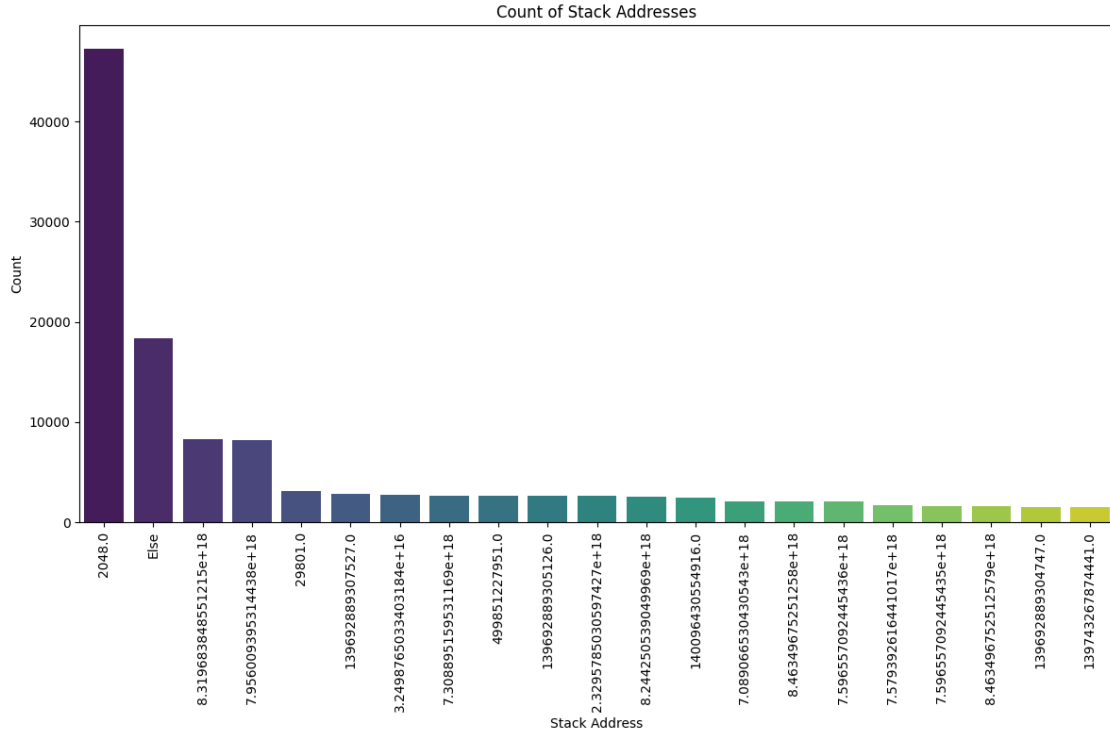


Figure 13: Stack addresses

- argsNum: number of arguments (integer)

ArgNum indicates the number of dictionaries that contains each Timestamp. And in the plot below you can observe that the most common numbers of arguments for each timestamp are 1, 2, and 4, with 4 being the most dominant by a significant margin. 0, 3, or 5 arguments is relatively rare, with zero and five arguments being the least common. The data shows a non-linear distribution with clear peaks at 1, 2, and 4 arguments.

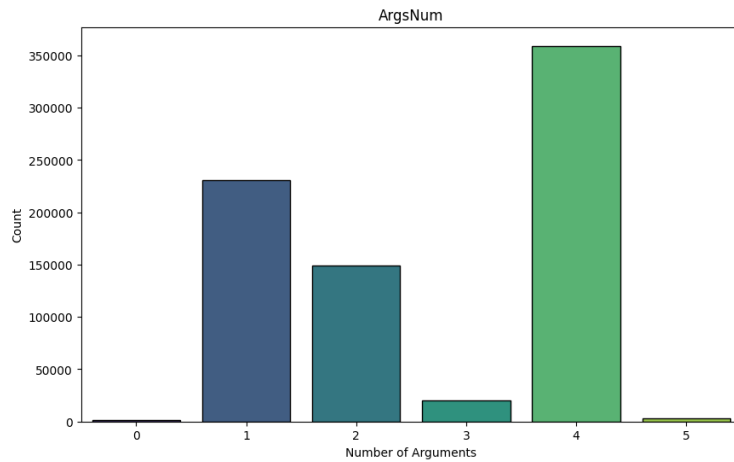


Figure 14: Args

- args: arguments passed to this process (list of dictionaries)

- sus: This is an integer label where 0 indicates non-suspicious activity and 1 indicates suspicious activity. We want to develop a model that can accurately classify and identify suspicious activities based on this labeling system.
 - The training and validation datasets are highly imbalanced with a large majority of “Unsuspicious” instances.
 - The testing dataset has a reversed imbalance, with “Suspicious” instances being the majority.
 - This imbalance across datasets suggests the need for careful handling during model training and evaluation to ensure that the model generalizes well and performs adequately on both classes. Techniques like oversampling the minority class, undersampling the majority class, or using class weights may be necessary during training.

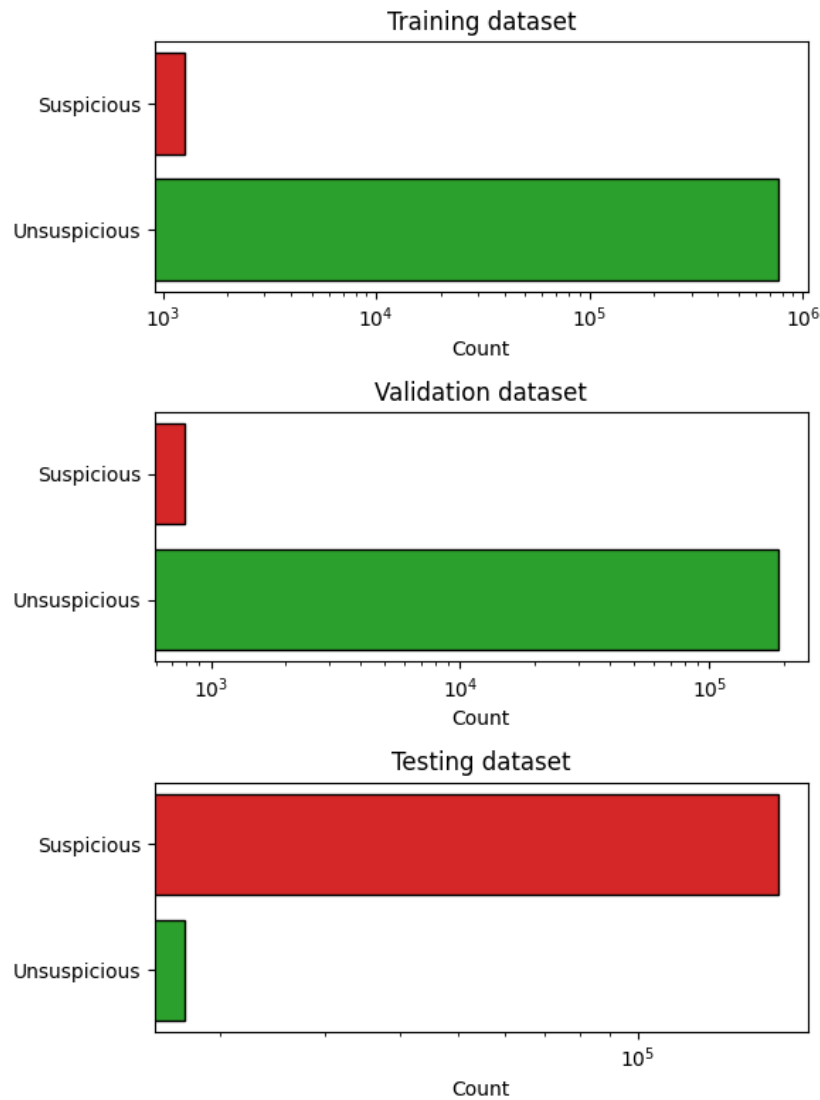


Figure 15: Sus label

- evil: This is an integer label where 0 indicates non-malicious activity and 1 indicates malicious activity. This label was not chosen for classification because the training and validation datasets do not contain any instances of the malicious class.

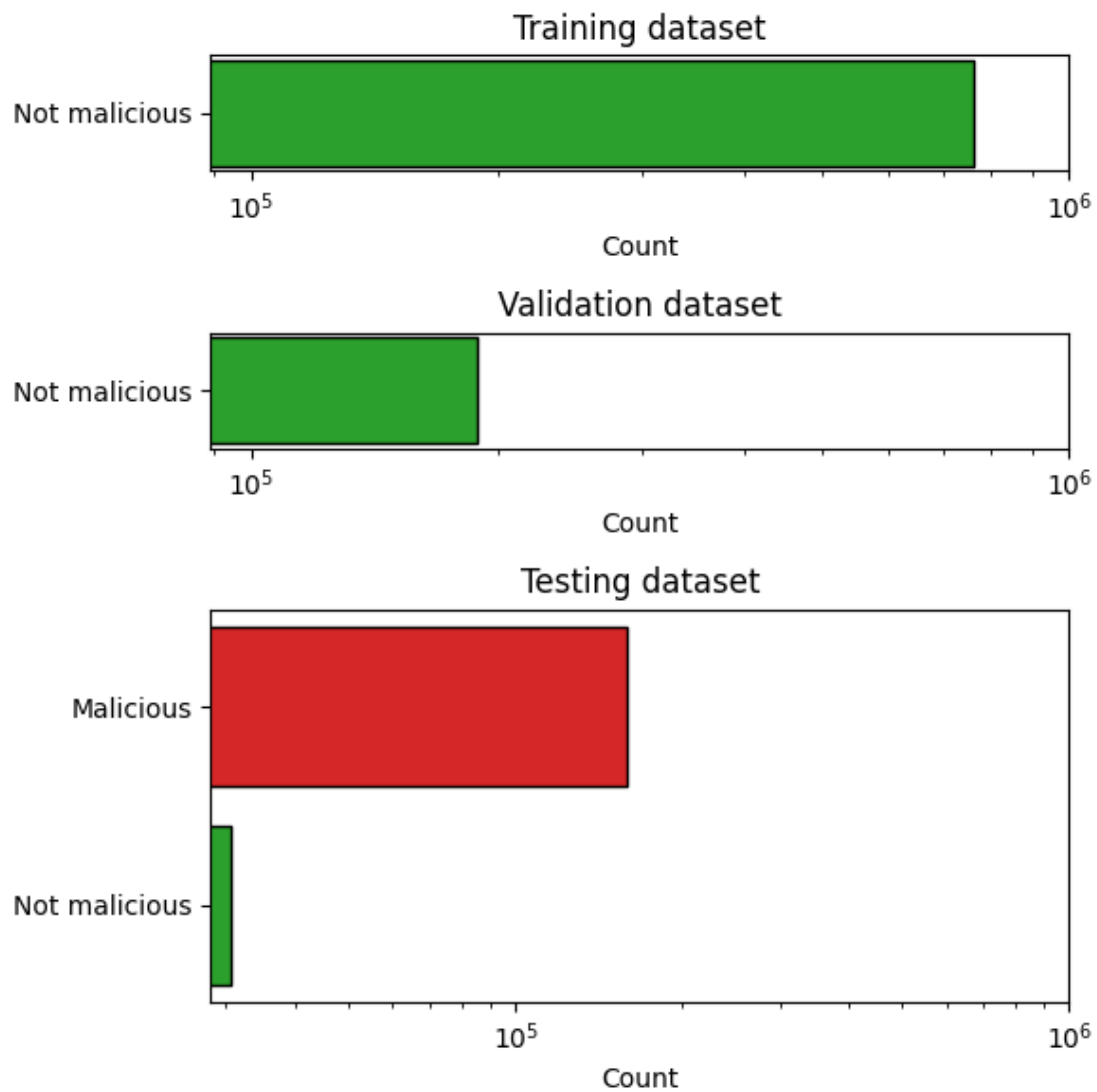


Figure 16: Evil label

2.2 Correlation matrix

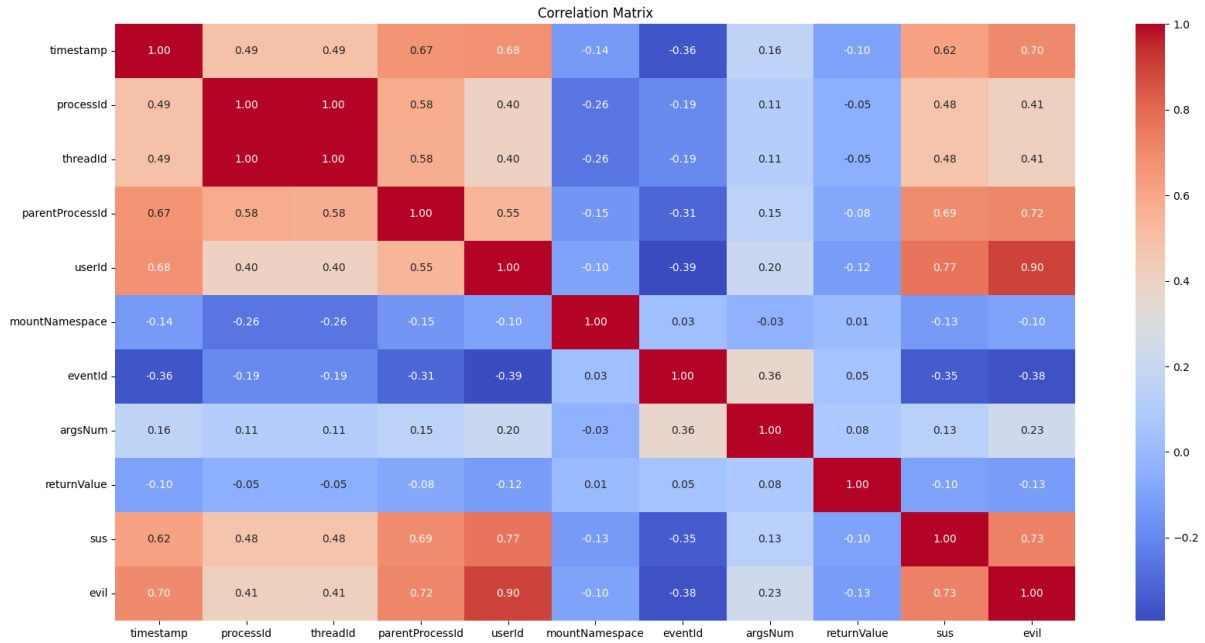


Figure 17: Correlation matrix

- Strong Positive Correlations:
 - processId and threadId: They have a correlation of 1.00, indicating they are perfectly correlated. This makes sense as threadId is often associated with processId.
 - parentProcessId and userId: With a correlation of 0.55, it suggests a moderate positive relationship. Likely because parent processes are tied to user accounts.
 - sus and userId: This has a high correlation of 0.77, suggesting that suspicious activity (sus) is strongly linked with specific user IDs.
 - evil and userId: This shows a very strong positive correlation of 0.90, indicating that ‘evil’ actions are highly associated with certain user IDs.
 - sus and evil: With a correlation of 0.73, it indicates that actions labeled as suspicious are strongly correlated with those labeled as evil.
- Moderate Positive Correlations:
 - timestamp and userId: A correlation of 0.68 suggests that timestamps are moderately positively related to user IDs, possibly indicating certain users are more active at certain times.
 - parentProcessId and timestamp: With 0.67, it shows a moderate positive relationship.
 - sus and parentProcessId: This correlation is 0.69, indicating that suspicious activities are moderately correlated with parent processes.
 - evil and parentProcessId: Correlation of 0.72, indicating a strong association between evil actions and parent processes.
- Negative Correlations:
 - mountNamespace with processId, threadId, parentProcessId: These are moderately negatively correlated (around -0.26), indicating that certain process/thread IDs and their parent processes

are less likely to have specific mountNamespace values.

- eventId with timestamp, userId, sus, evil: Negative correlations, especially -0.36 with timestamp and -0.39 with userId, suggest that certain events are less likely to happen at certain times or for certain users.
- eventId with sus and evil: Both are negatively correlated (around -0.35 to -0.38), indicating that particular events are less associated with suspicious and evil activities.
- Low/No Correlation:
 - argsNum and other variables: Mostly low correlations, suggesting that the number of arguments has little to no linear relationship with the other features.
 - returnValue and other variables: Low correlations overall, indicating the return value of processes is largely independent of other features.
- Interpreting Specific Pairs:
 - timestamp and sus/evil: These have correlations of 0.62 and 0.70, respectively. This suggests that the timing of events is significantly associated with suspicious and evil activities.
 - mountNamespace: Shows mostly weak correlations with other features, suggesting that mountNamespace values are relatively independent of other variables.

2.3 Event Frequency

The following chart shows the entire frequency of suspicious and not suspicious event:

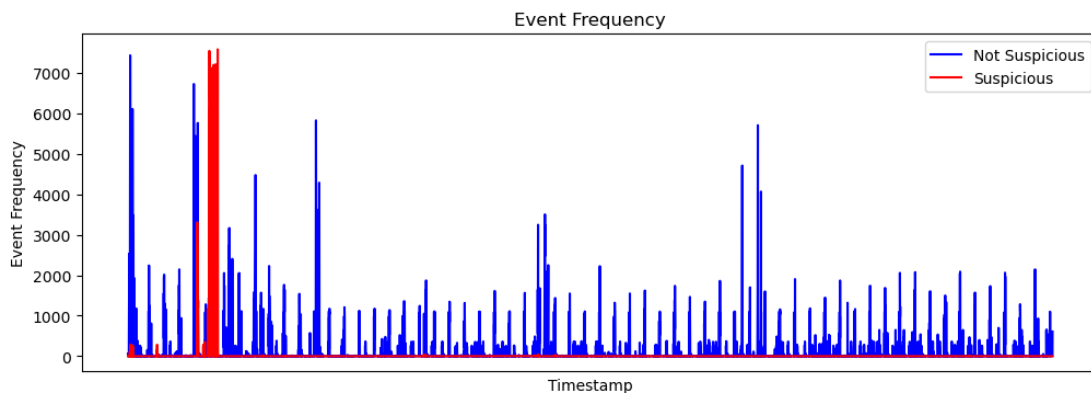


Figure 18: Suspicious and not suspicious event

- Event Frequency: The y-axis represents the frequency of events, ranging from 0 to over 7000.
- Timestamp: The x-axis represents the timestamps when the events occurred.
- Not Suspicious Events: Represented by blue lines. The frequency of these events is higher and more variable, with several spikes reaching high values, particularly towards the left side of the chart.
- Suspicious Events: Represented by red lines. These events are less frequent and usually have lower values compared to the “Not Suspicious” events. There are a few noticeable spikes in the red lines, indicating higher frequencies of suspicious events at certain timestamps.

Overall, the chart shows that “Not Suspicious” events occur more frequently and with higher peaks compared to “Suspicious” events, which occur less often and with lower peaks.

3 Data preparation

3.1 Numerical data transformation

As advised by the authors of the beth dataset's paper, we applied these transformation: - ProcessId and ParentprocessId: 0 if it is [0,1,2] otherwise 1 - UserId: 0 if id is less than 1000 otherwise 1 - MountNamespace: 0 if it is equal to 4026531840 otherwise 1 - ReturnValue: 0 if it is 0, 1 if it is positif and 2 if it is negatif

3.2 StackAddresses

Stackaddresses sf a list of numerics with a maximum of 20 elements. We created 20 new columns named "stack_1", "stack_2", etc. in each dataset, and assigns each element from the list to its respective new column.

3.3 Args

Args column contains a list of maximum 5 dictionaries, each disctionary contains three elements ({'name': 'dev', 'type': 'dev_t', 'value': 211812353}). We created 15 new columns in each datset, and assigns each element from the dictionaries to its respective new column.

3.4 Ordinal encoding

Ordinal encoding is a technique for converting categorical data, where variables have distinct labels or categories, into numerical form suitable for machine learning algorithms. It assigns a unique integer value to each category based on its order or rank. As our approach is for an unsupervised model, we used ordinal encoder to handle new classes not present in the training dataset.

Ordinal encoder will assign -1 value to unknown classes (labels not present in the training dataset)

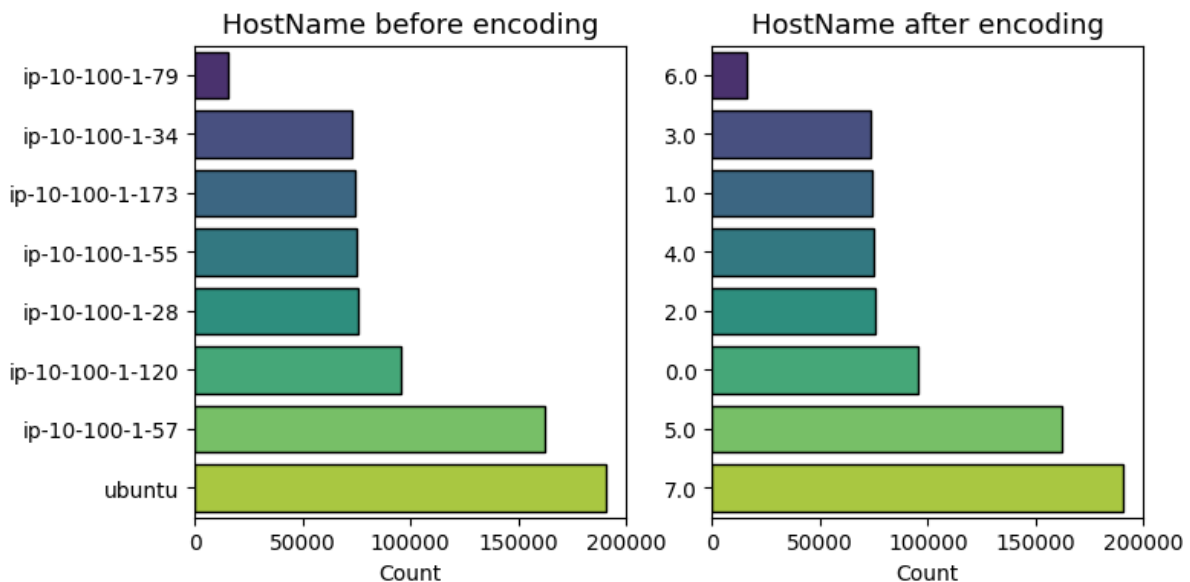


Figure 19: Ordinal encoder

3.5 Scaling

Numerical features are scaled to similar range as they have different scales. Since we used ordinal encoding for categorical features, scaling is not necessary. Ordinal encoding preserves the order of the categories, but the assigned values don't necessarily reflect their magnitude.

3.6 Smote

Dealing with unbalanced data can be tricky, most of the machine learning model will give good results for big classes and poor performance on the minority although, as it is our case, minority class is more important. To balance that, we tried to use Smote library combined as it is advised with random undersampling for the majority class. SMOTE (Synthetic Minority Oversampling TEchnique) [2] works by interpolating new instances along line segments joining existing minority class instances.

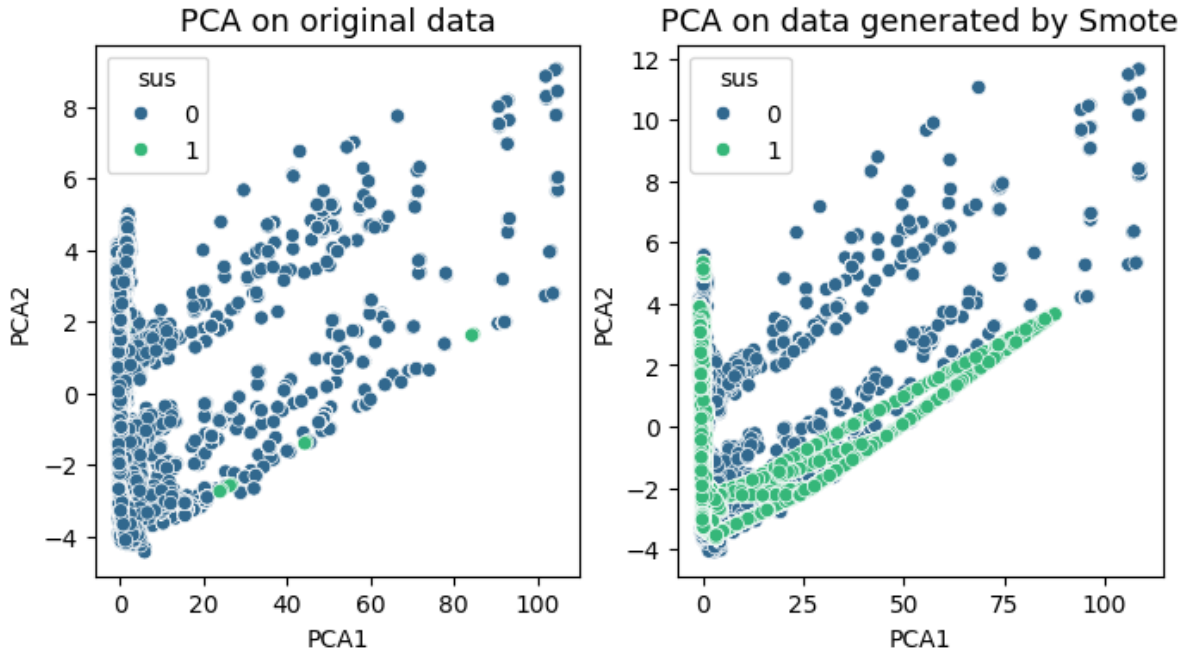


Figure 20: Smote

3.7 Shapelet discovery method

Shapelet discovery is a technique used in time series analysis to identify discriminative subpatterns, known as shapelets, within a set of time series data. Shapelets are subsequences that capture characteristic patterns or behaviors in the data. The process of shapelet discovery involves searching through the time series data to find subsequences that are representative of different classes or categories like in our case for **suspicious activities and not suspicious activities**. The similarity or distance between each subsequence and the rest of the data is computed to determine its discriminative power. The shapelets with the highest discriminative power are selected as representative patterns.

So the shapelet discovery can use the matrix profile as a tool for efficiently computing the distances or similarities between subsequences. By utilizing the matrix profile, shapelet discovery algorithms can reduce the computational complexity and speed up the process of identifying shapelets.

With the concept of matrix profile, we tried to find conserved behaviours in the data. In fact, a comparison between sequences can be done by looking at the euclidean distance between all the points in two subsequences and represent the distances in a matrix profile.

For the high computational requirements, in order to experiment the shapelet discovery method, we decided to adopt a sample of 230.000 datapoints and see how this method performs for the Decision tree classifier, LSTM and the Dense model.

4 Models

4.1 Dense neural network

4.1.1 Model 1

- **Description**

This model is composed with five hidden dense layers each with 512 units and ReLU activation, interspersed with dropout layers for regularization, and an output layer with a single unit and sigmoid activation for binary classification. Each dense layer uses the 'lecun_normal' initializer for the kernel and a RandomNormal initializer for the bias.

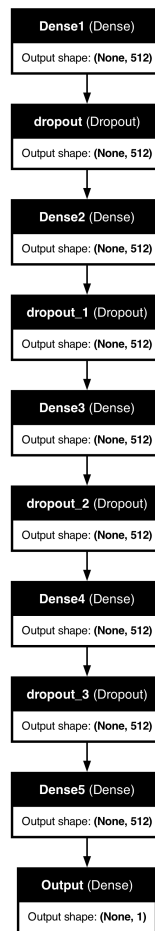


Figure 21: Dense neural network - Model 1

- **Training**

The plot shows that the training accuracy rapidly increases and stabilizes around 0.9996 within a few epochs. However, the validation accuracy remains constant at 0.9998 throughout all epochs, suggesting that the model might be overfitting to the training data or that the validation set might not be sufficiently challenging.

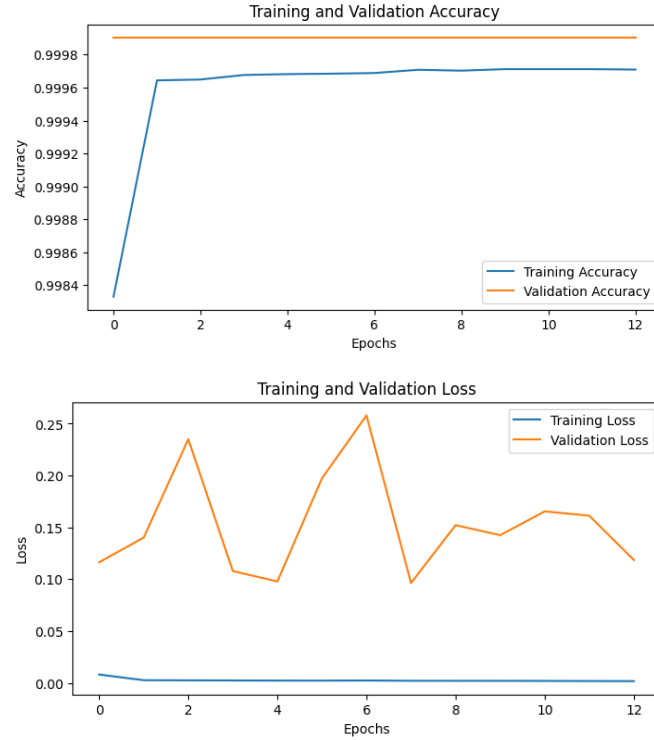


Figure 22: Model 1 accuracy and loss

- **Prediction**

Model 1 seems to predict only the “unsuspicious” class and fails to detect any “suspicious” activities, resulting in poor performance for identifying suspicious activities.

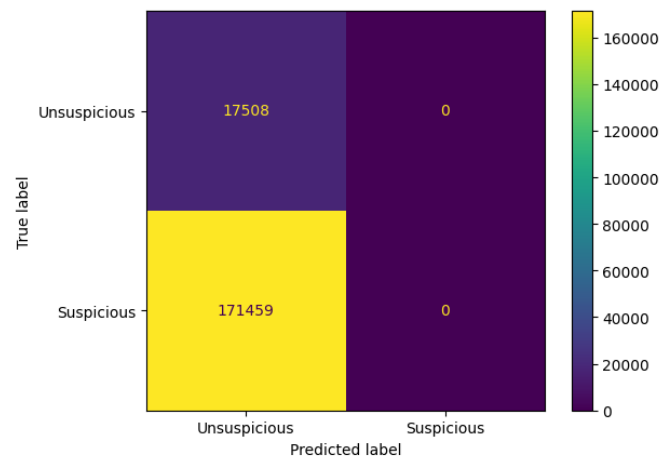


Figure 23: Model 1 Confusion matrix

4.1.2 Model 2

- **Description** This model is a neural network that handle differently categorical and numerical features. It incorporates embeddings for the categorical inputs, which are then reshaped and concatenated with numerical inputs, followed by multiple dense layers with ReLU activations and dropout for regularization. The final output layer uses a sigmoid activation function to produce a binary classification result.

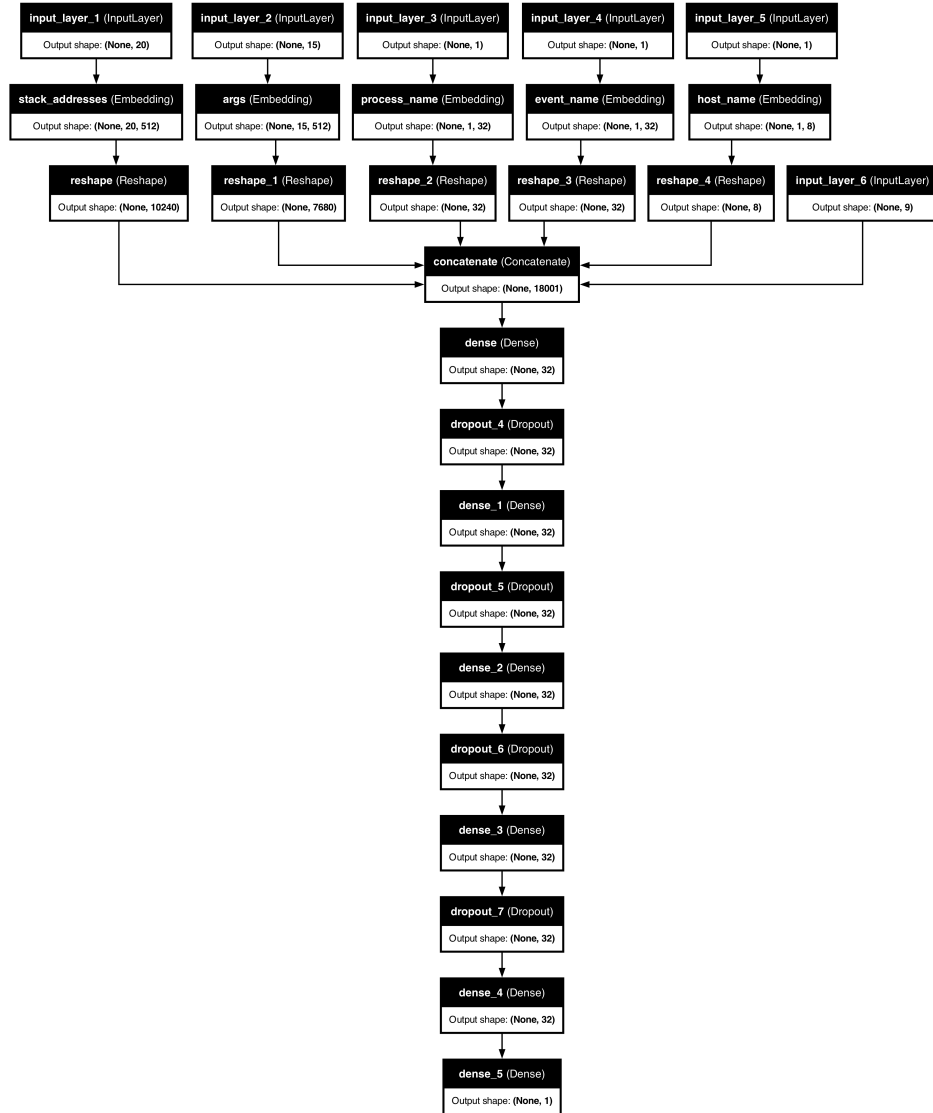


Figure 24: Dense neural network - Model 2

- **Training**

The training accuracy rapidly increases and stabilizes at approximately 0.998 after the first epoch. In contrast, the validation accuracy remains constant at 0.996 throughout all epochs. Both the training and validation losses decrease sharply during the first epoch and after that remains almost constant with minor fluctuations.

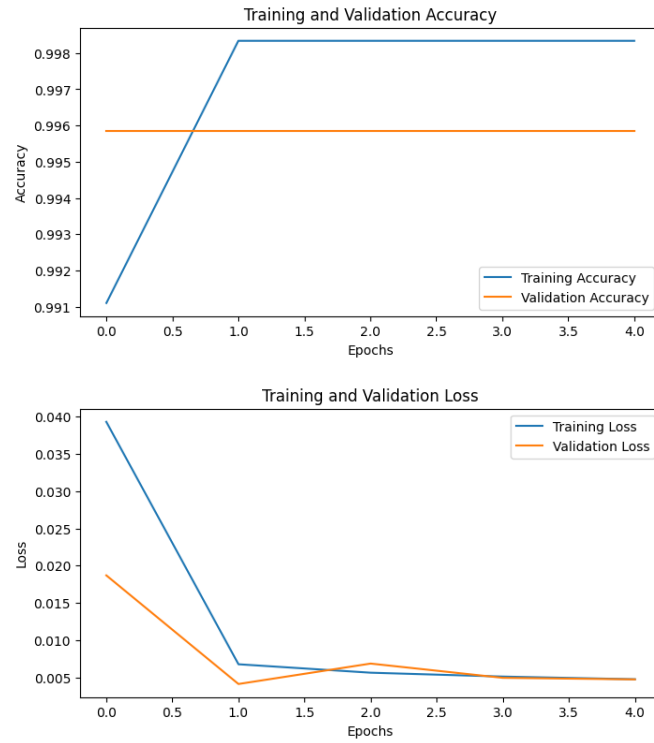


Figure 25: Model 2 accuracy and loss

- **Prediction**

Model 2 shows a strong ability to correctly identify suspicious activities while maintaining a low false positive rate. However, there is still room for improvement in reducing the number of false negatives, which could enhance the model's sensitivity to suspicious activities. We have also very low false positive rate for unsuspicious class.

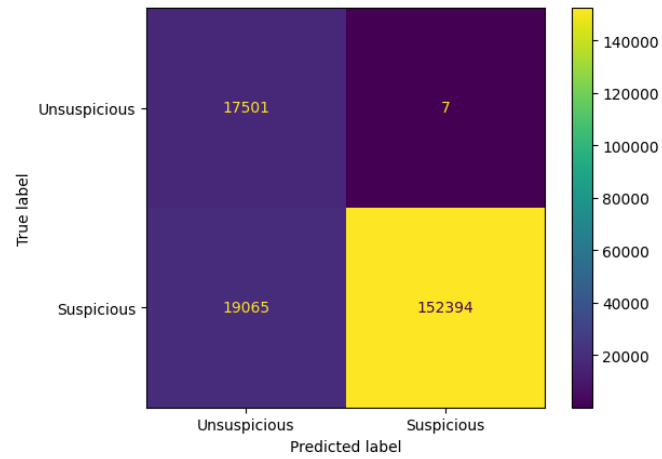


Figure 26: Model 2 Confusion matrix

4.1.3 Model 3

- **Description** This model is similar to **Model 1**, this model is trained on data after applying Smote data augmentation technic.

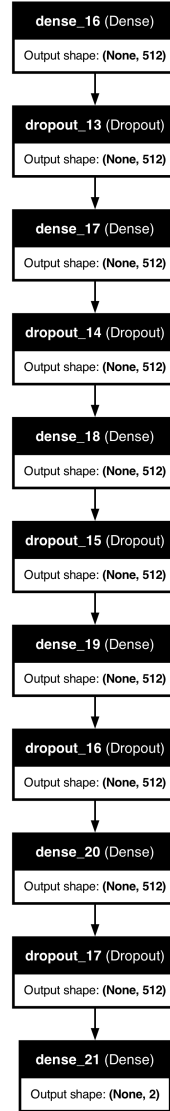


Figure 27: Dense neural network - Model 3

- **Training**

The validation accuracy increases and stabilizes at nearly 83%, though it remains lower than the training accuracy. This indicates a discrepancy between the training and validation datasets after applying Smote.

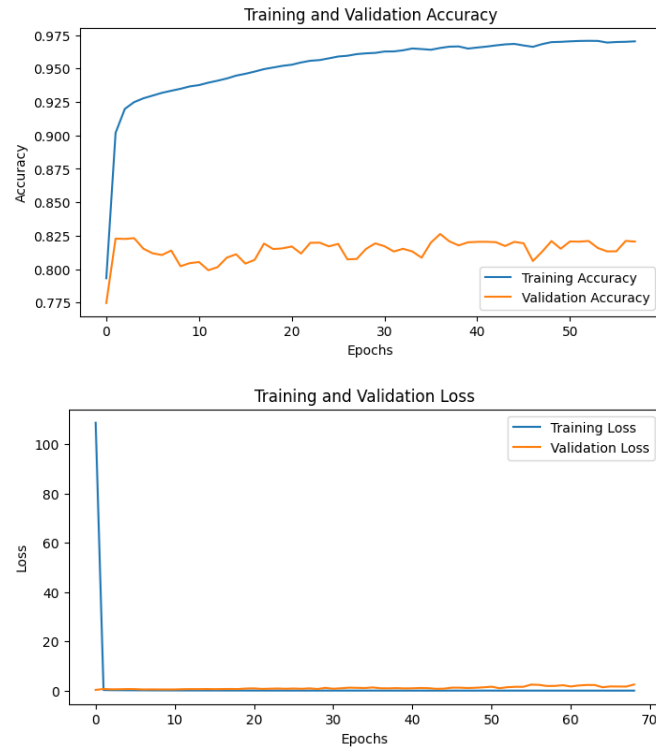


Figure 28: Model 3 accuracy and loss

- **Prediction**

With SMOTE data, the model exclusively predicts the “unsuspicious” class and fails to identify any “suspicious” activities. This indicates that the model is not effectively learning from the augmented data, even with the improved balance in our dataset.

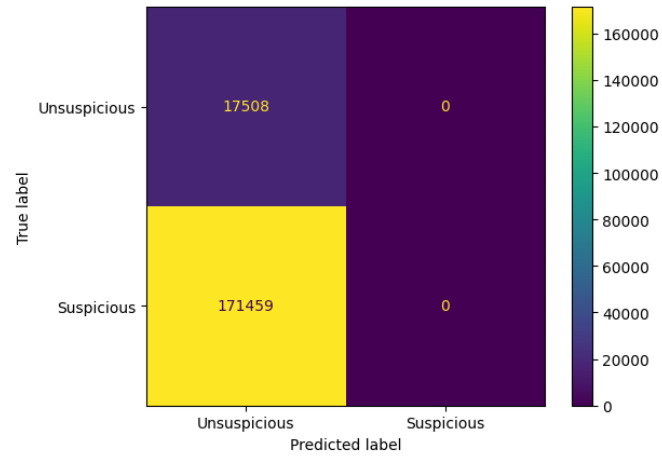


Figure 29: Model 3 Confusion matrix

4.1.4 Model 4

- **Description**

The model starts with an input layer that receives inputs of shape (1). It then passes through a series of dense layers, each followed by dropout layers for regularization. The output of the model is a single value.

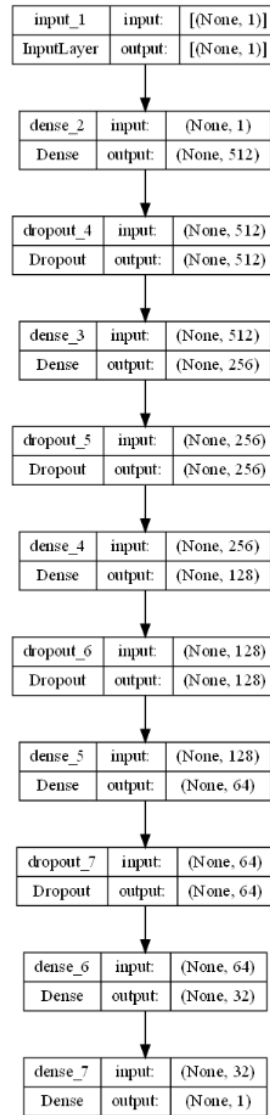


Figure 30: Dense neural network - Model 4

- **Training**

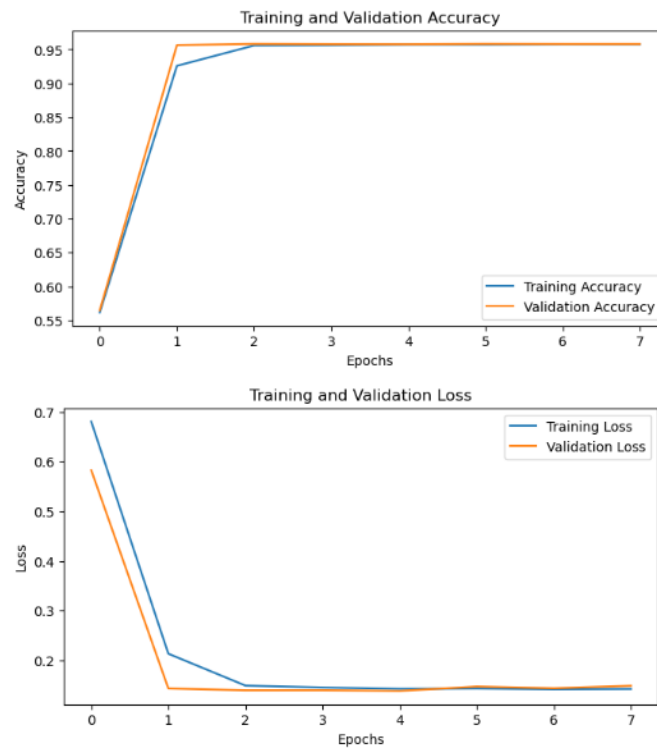


Figure 31: Model 4 accuracy and loss

- **Prediction**

The confusion matrix shows that we have few false negatives (913) and few false positives (12).

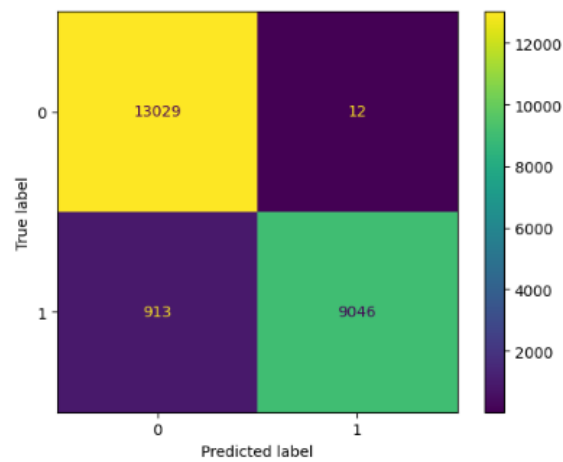


Figure 32: Model 4 confusion matrix

4.2 Convolutional neural network

4.2.1 Model 5

- **Description**

This model is a Convolutional Neural Network (CNN) with an input shape of (47, 1).

It comprises four Conv1D layers with decreasing filter sizes (256, 128, 64, and 32) and ReLU activations, each followed by a dropout layer to prevent overfitting. We applied a Lecun normal initializer for the kernels and a custom random normal initializer for the biases.

The output layer is a dense layer with a sigmoid activation function for binary classification.

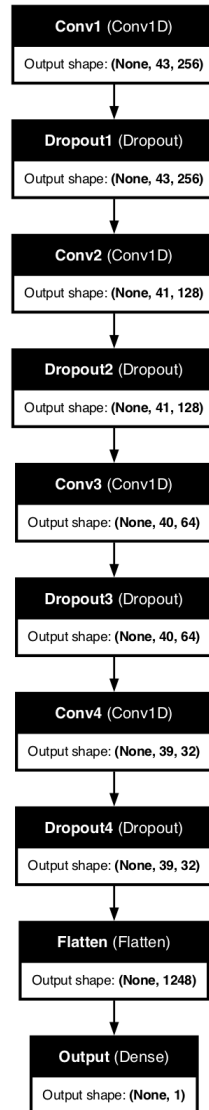


Figure 33: Conv. neural network - Model 5

- Training

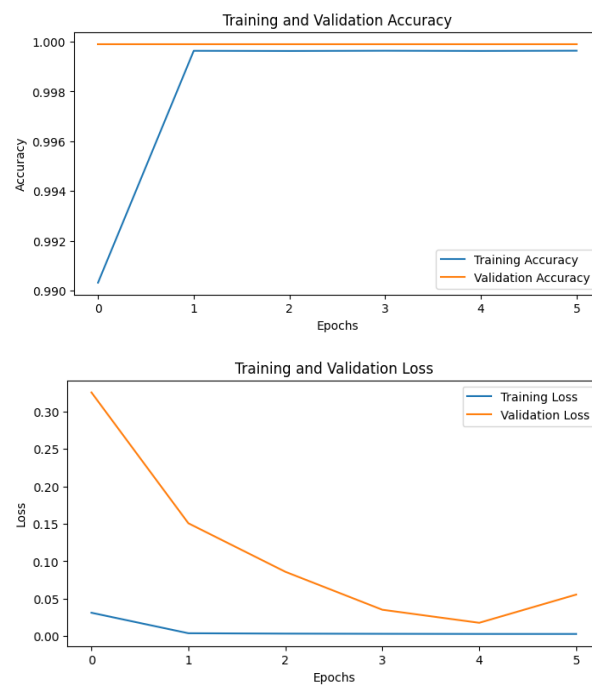


Figure 34: Model 5 accuracy and loss

- Prediction

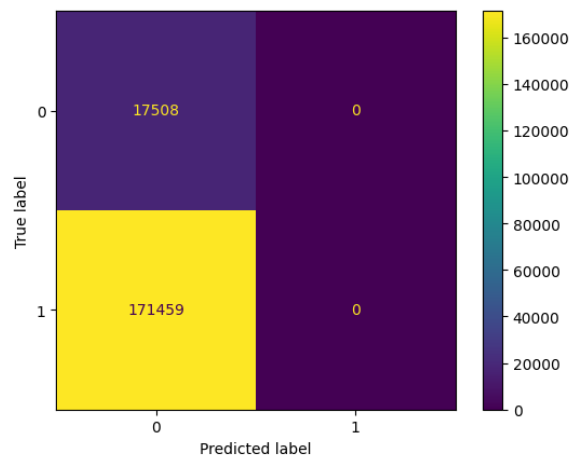


Figure 35: Model 5 Confusion matrix

4.2.2 Model 6

- **Description**

This model handles categorical and numerical inputs separately, using embeddings and dense layers for preprocessing.

It creates embeddings for two categorical features (args and stackaddresses), followed by linear transformations, and processes numerical features through a dense layer and reshaping.

The processed embeddings and numerical features are concatenated and passed through two Conv1D layers with ReLU activations for feature extraction.

Finally, the output layer is a dense layer with a sigmoid activation function for binary classification.

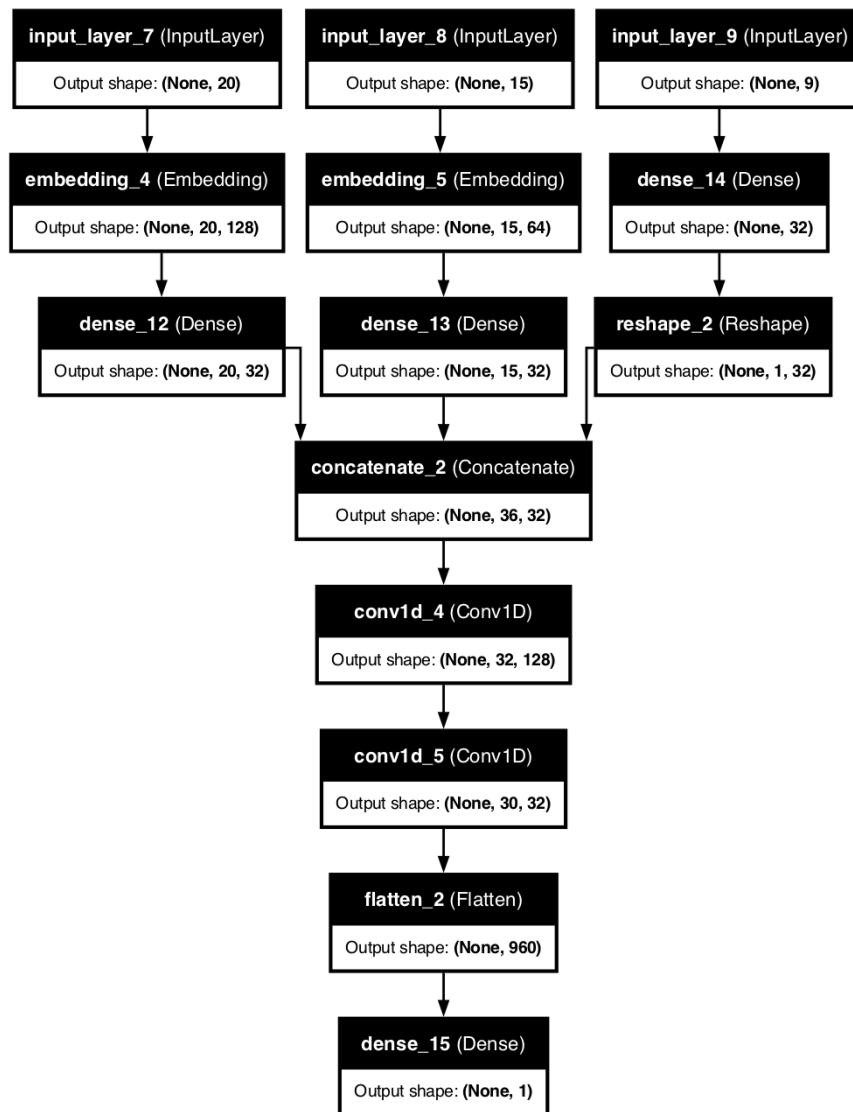


Figure 36: Conv. neural network - Model 6

- Training

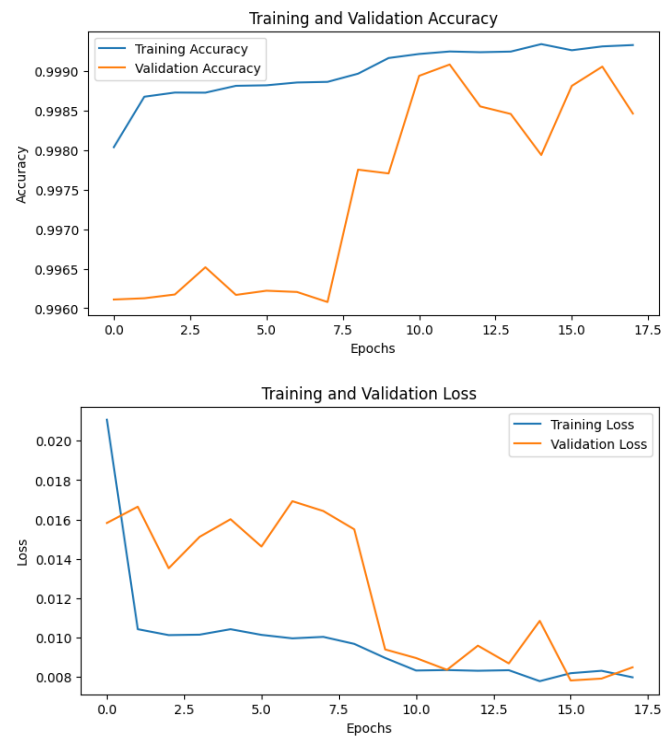


Figure 37: Model 6 accuracy and loss

- Prediction

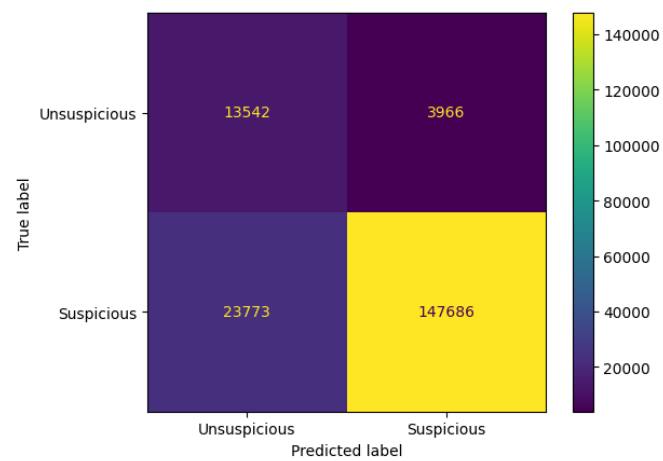


Figure 38: Model 6 Confusion matrix

4.3 LSTM neural network

4.3.1 Model 7

- **Description**

This model is a Sequential Long Short-Term Memory (LSTM) network designed for sequence data with an input shape of (47, 1). It consists of four LSTM layers, each with 32 units and ReLU activations, using Lecun normal initialization for the kernels and a custom random normal initializer for the biases, followed by dropout layers to prevent overfitting. The output from the LSTM layers is flattened and passed through a dense layer with 128 units and a final dense layer with a sigmoid activation function for binary classification.

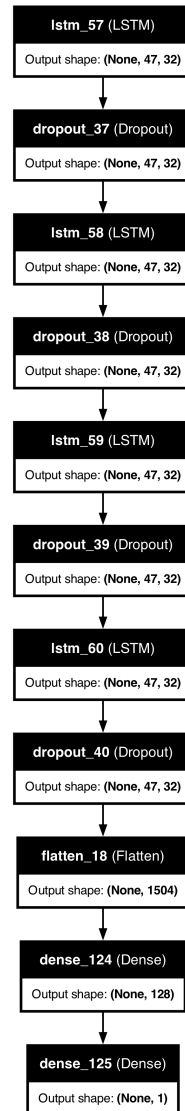


Figure 39: LSTM - Model 7

- Training

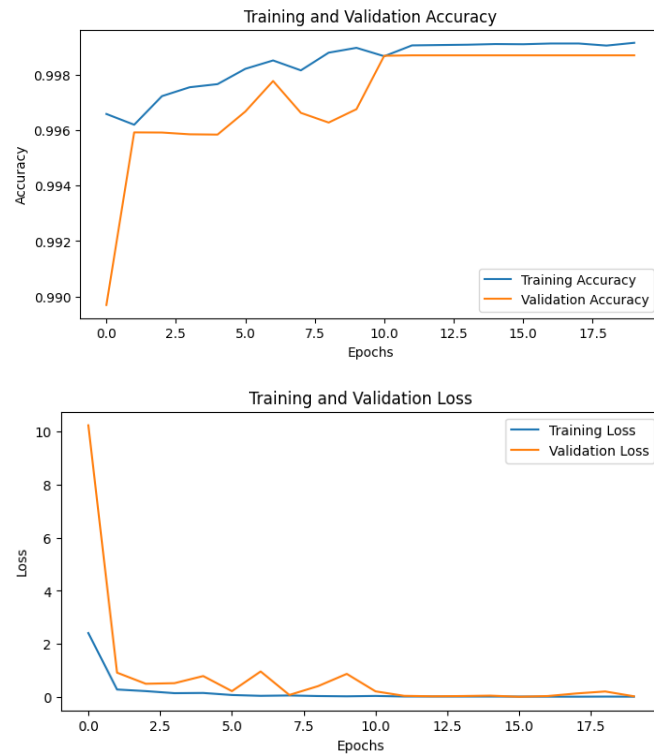


Figure 40: Model 7 accuracy and loss

- Prediction

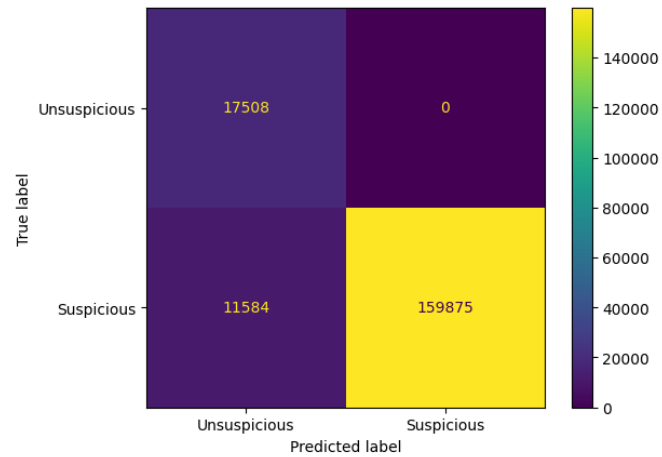


Figure 41: Model 7 Confusion matrix

The confusion matrix indicates that the model is highly effective in identifying suspicious activities, correctly classifying 159,875 out of 171,459 suspicious instances while maintaining a perfect true negative rate with 17,508 correct unsuspicious classifications. However, it still misses 11,584 suspicious instances

4.3.2 Model 8

- **Description**

This model integrates various types of inputs through embedding layers to handle categorical data, and dense layers for numerical data.

The use of LSTM layers suggests that the model is designed to capture temporal dependencies or sequences in the data.

Multiple dense and dropout layers are used to refine the features and prevent overfitting before producing the final classification output.

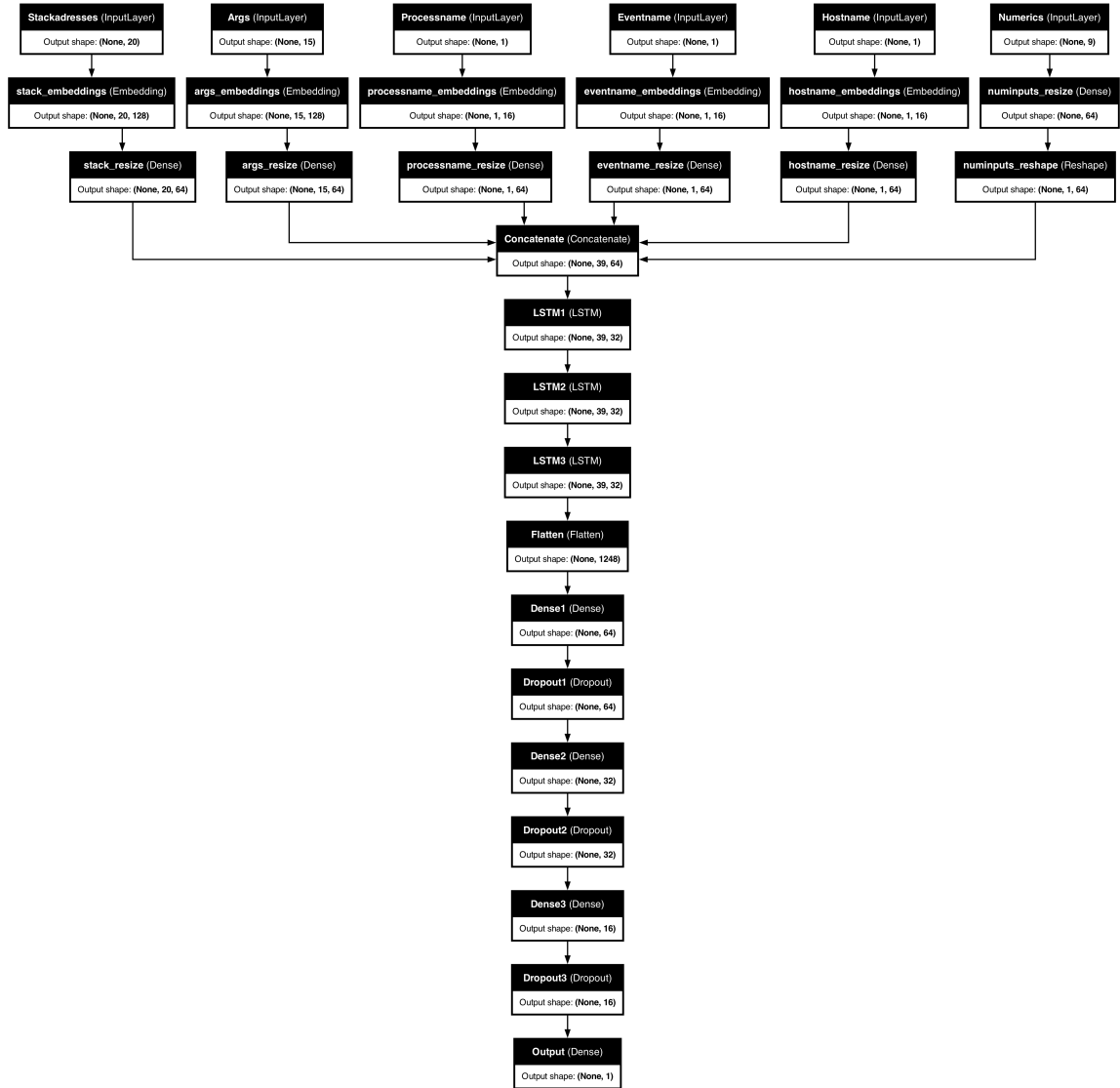


Figure 42: LSTM - Model 8

- Training

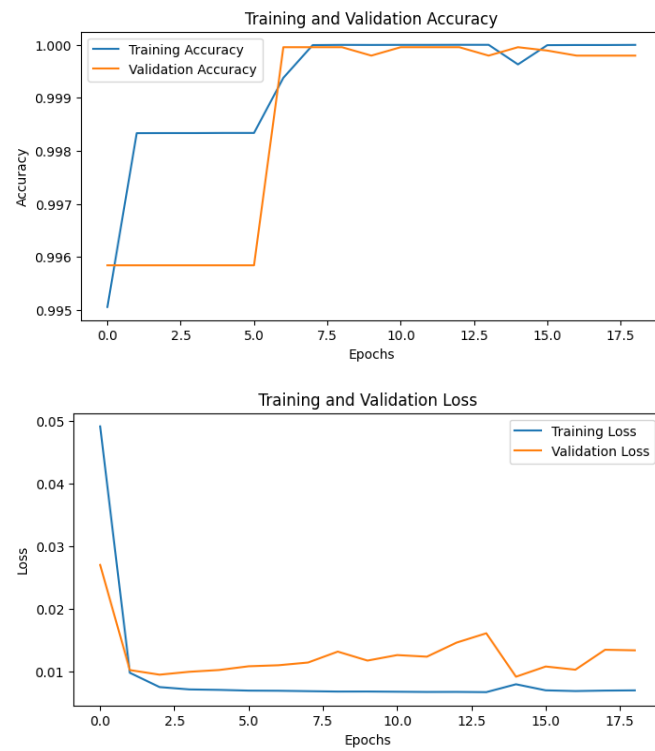


Figure 43: Model 8 accuracy and loss

- Prediction

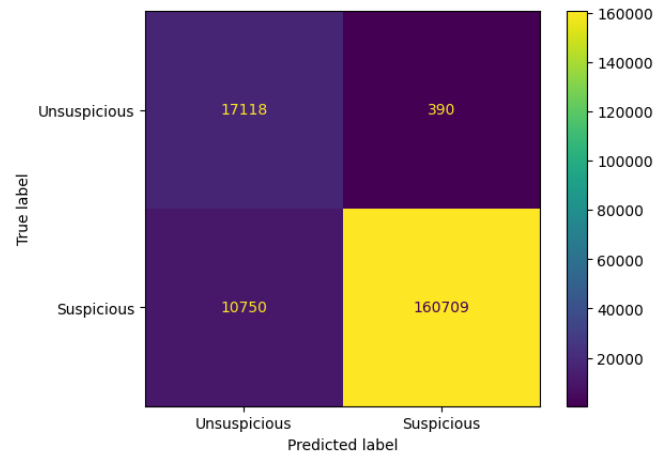


Figure 44: Model 8 Confusion matrix

4.3.3 Model 9

- **Description**

This model is a Sequential Long Short-Term Memory (LSTM) network designed for sequence data with an input shape of (1, 1).

It consists of four LSTM layers, each with 32 units and ReLU activations, using Lecun normal initialization for the kernels and a custom random normal initializer for the biases, followed by dropout layers to prevent overfitting.

The output from the LSTM layers is flattened and passed through a dense layer with 128 units and a final dense layer with a sigmoid activation function for binary classification.

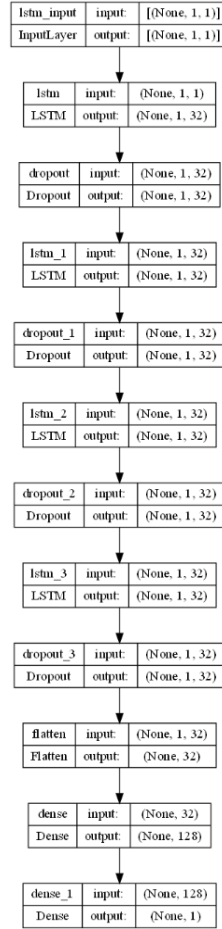


Figure 45: LSTM - Model 9

- **Training**

As the graphes show, the LSTM doesn't give good resultst when you apply the shapeet discovery method. The performance of the model is really poor and ineffective.

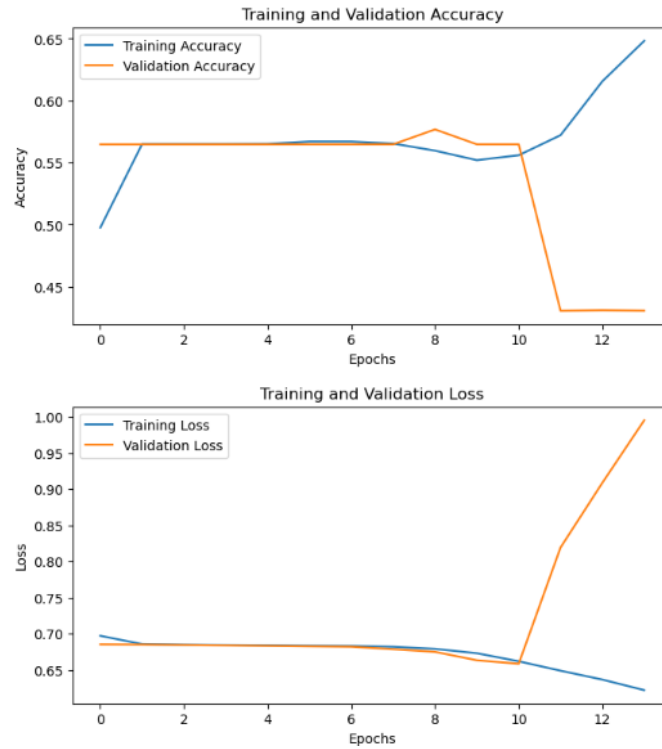


Figure 46: Model 9 accuracy and loss

- **Prediction**

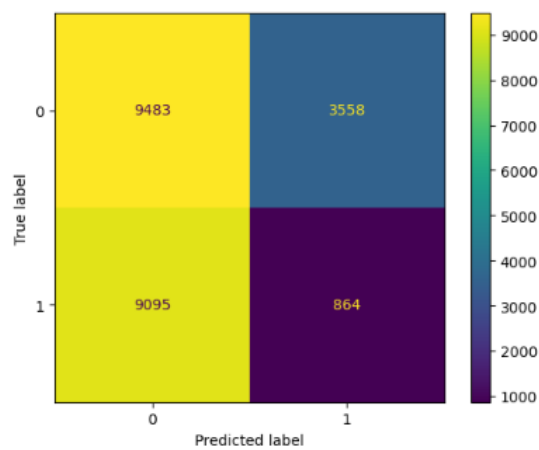


Figure 47: Model 9 Confusion matrix

4.4 Transformer

- **Description**

This model integrates categorical and numerical inputs using a transformer-based architecture. Categorical inputs are embedded and transformed via dense layers, while numerical inputs are processed through a dense layer and reshaped.

The combined embeddings and numerical features are enhanced with positional encoding and passed through several transformer encoder blocks, followed by convolutional layers, global average pooling, and fully connected layers, ultimately producing a single sigmoid-activated output.



Figure 48: Transformer structure

- Positional encoding adds information about the position of each element in the sequence by creating a positional encoding matrix, which applies sinusoidal functions to encode positional information. This matrix is added to the input embeddings, allowing the model to incorporate the order of the sequence elements, which is crucial for the transformer to understand the sequential nature of the data.
- The `transformer_encoder_block` applies multi-head self-attention to the inputs, enabling the model to focus on different parts of the sequence simultaneously. This is followed by a dropout layer for regularization and layer normalization to stabilize and speed up training. Finally, a feed-forward neural network with a dense layer, dropout, and another layer normalization is used to further process the attention output, enhancing the model's capacity to capture complex patterns in the data.
- This model uses Adam optimizer with a **WarmUpCosineDecay** learning rate. During the warmup phase, the learning rate increases linearly, and once the warmup steps are completed, it follows a cosine decay pattern to gradually reduce the learning rate.

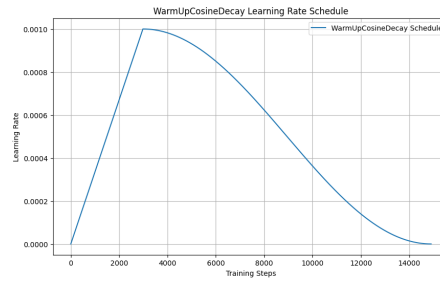


Figure 49: Learning rate decay

- **Training:**

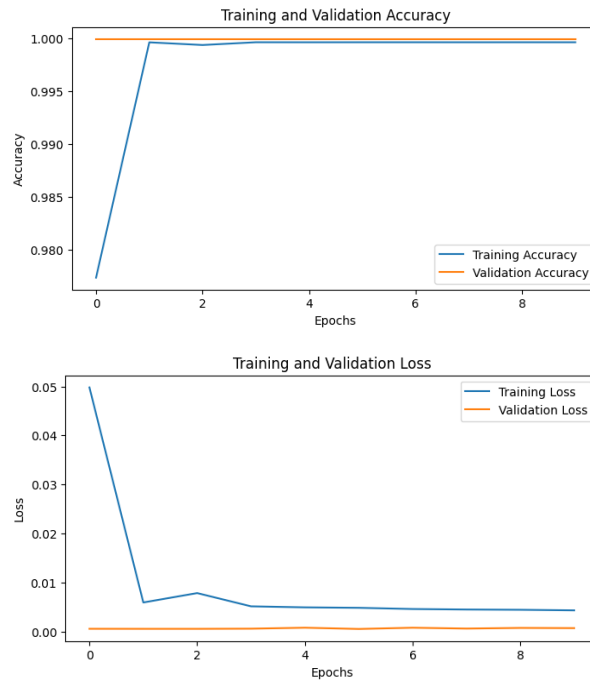


Figure 50: Transformer accuracy and loss

- **Prediction:**

- Strengths:

- * The model has high accuracy (94.6%).
 - * Perfect precision and specificity, meaning there are no false positives.
 - * High recall (94.1%) and a strong F1 score (97%).

- Weaknesses:

- * The model still misses some positive instances (10,208 false negatives), which may be critical depending on the context of the application.
 - * The imbalance in predictions (zero false positives but some false negatives) could indicate a bias towards negative predictions.

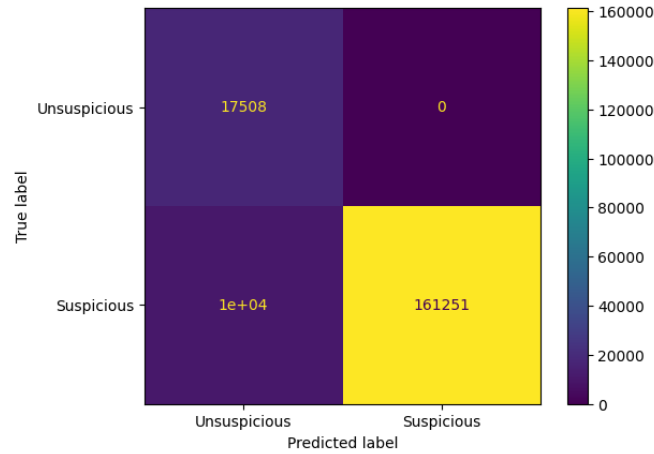


Figure 51: Transformer Confusion matrix

4.5 Decision Tree Classifier with Shapelet Discovery method

- **Prediction:**

The confusuion matrix shows a small number of False Positives (25), which is good as it shows the model rarely predicts class 1 when the true class is 0. A relatively low number of False Negatives (1989), indicating the model occasionally misses class 1 predictions. Overall the model seems to learn.

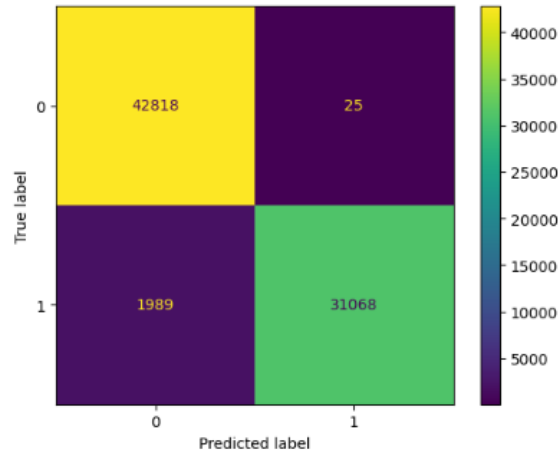


Figure 52: Transformer Confusion matrix

5 Results

Model	Accuracy	Precision avg	Recall avg	ROC score
Dense model	0.09	0.05	0.50	0.50
Dense model + embeddings	0.91	0.75	0.95	0.95
CNN model	0.11	0.53	0.51	0.51
CNN model + embeddings	0.95	0.82	0.97	0.97
RNN model	0.09	0.05	0.50	0.50
RNN model + embeddings.	0.95	0.82	0.97	0.97
Transformer	0.95	0.82	0.97	0.97

Shapelet Discovery method with 230.000 data points

Model	Accuracy	Precision avg	Recall avg	ROC score
Decision tree classifier	0.97	0.98	0.97	0.97
LSTM model	0.45	0.35	0.41	0.41
Dense Model	0.96	0.97	0.95	0.95

Figure 53: Results

6 Discussion

The results of our experiments on the BETH dataset reveal a range of insights into the effectiveness of various machine learning models for anomaly detection in the field of cybersecurity. This discussion aims to analyze the performance of the models, identify challenges in data processing and model development, and discuss potential improvement strategies.

Model Effectiveness

Most models demonstrated high validation accuracy and converged quickly, indicating a high similarity between the training and validation datasets. This could also suggest that the task is relatively simple, which may not fully reflect real-world challenges. Particularly notable are the CNN and RNN models, which showed a significant performance boost through the use of embeddings. The transformer approach also exhibited high accuracy and F1 scores, underscoring its suitability for complex data patterns.

Data Processing Challenges

A major challenge was handling the highly imbalanced dataset. The BETH dataset is characterized by a dominance of non-suspicious events, which affected model performance. Despite using the SMOTE technique to generate synthetic data for the minority class, some models struggled to correctly classify suspicious behavior. This highlights the need for advanced data balancing techniques and the integration

of additional features to improve classification.

Form of Data Preparation

Our experiments with the Shapelet Discovery method showed that while it is theoretically promising, it did not deliver the expected results in practice due to the structure of the BETH dataset and the high computational power required. The method performed well on smaller samples but was not scalable to the entire dataset. Future work could focus on developing more efficient algorithms for shapelet discovery or exploring alternative feature extraction methods.

Model-Specific Results

- **Dense Neural Networks (DNNs):** The DNN models showed varying results, with some models performing well and others not. The use of embeddings in DNNs (Model 2) significantly improved performance compared to models without embeddings. However, Model 3, after applying SMOTE, showed insufficient performance, indicating a lack of learning ability from the augmented data. The method of Shapelet Discovery applied on the Dense model shows a good performance on a small sample of data that indicate a potentiality of this method applied to the Dense algorithm.
- **Convolutional Neural Networks (CNNs):** CNNs, particularly Models 5 and 6, achieved high accuracies and low error rates. This suggests that CNNs are well-suited to recognize structural features in the data, especially when using embeddings.
- **Recurrent Neural Networks (RNNs):** The LSTM models (Models 7 to 9) showed mixed results. While Model 7 exhibited a high capability for detecting suspicious activities, Model 9, which used the Shapelet Discovery method, was ineffective.
- **Transformer:** The transformer showed the best overall performance, particularly through the use of positional encoding and multi-layer self-attention, enhancing its ability to detect complex patterns in the data.
- **Decision Tree Classifier with Shapelet Discovery:** this approach presents a low number of misclassifications and a good accuracy level.

Improvement Suggestions

For future work, we suggest the following approaches:

- **Advanced Data Augmentation:** In addition to SMOTE, other techniques such as GANs (Generative Adversarial Networks) could be used to generate synthetic data.
- **Feature Engineering:** Developing and integrating additional relevant features could further improve model performance.
- **Model Hybridization:** Combining different model approaches could leverage the strengths of individual models and improve overall performance.
- **Training Strategy Adjustment:** Using advanced training strategies such as transfer learning and ensemble learning could enhance the models' generalization ability.
- **More Efficient Algorithms:** Developing more efficient algorithms for methods like Shapelet Discovery could improve the scalability and applicability of these approaches.

In summary, this work highlights the potentials and challenges of using machine learning on real cybersecurity data. Despite the high accuracy of most models, improving sensitivity to suspicious behavior remains a central challenge that needs to be addressed through advanced techniques and approaches

7 References

1. BETH Dataset: Real Cybersecurity Data for Anomaly Detection Research
Kate Highnam, Kai Arulkumaran, Zachary Hanif, Nicholas R. Jennings
<https://www.gatsby.ucl.ac.uk/~balaji/udl2021/accepted-papers/UDL2021-paper-033.pdf>
2. Smote: Synthetic Minority Over-sampling Technique
Authors: Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, W. Philip Kegelmeyer
<https://arxiv.org/pdf/1106.1813>
3. Time Series Shapelets: A New Primitive for Data Mining. Lexiang Ye, Eamonn Keogh
<https://www.cs.ucr.edu/~eamonn/shaplet.pdf>