Machine learning on real cybersecurity data

Jamel Belgacem - Papa Moryba Kouate

2024-06-12

Table of contents:

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 datset represents more than 8 milions events collected over 23 honeypots, only nearly 1 milion 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 has those features: - timestamp: time in seconds since system boot (float)

- processId: id of the process spawning this log (integer)

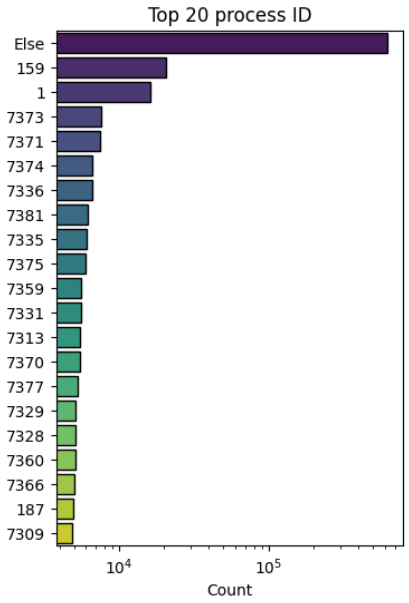


Figure 1: Process ID

- threadId: id of the thread (integer)
  There is a total of 545 thread ids. ThreadId and processId seems to have the same distribution.

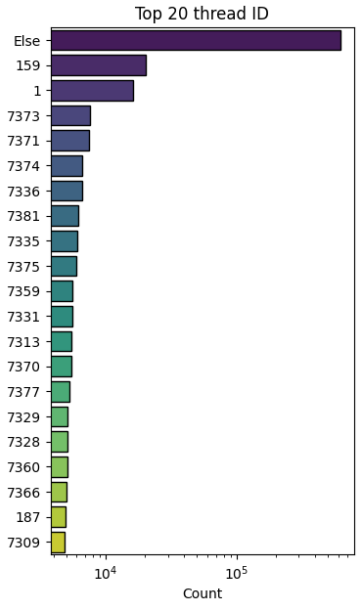


Figure 2: Thread ID

- parentProcessId: parent process id (integer)

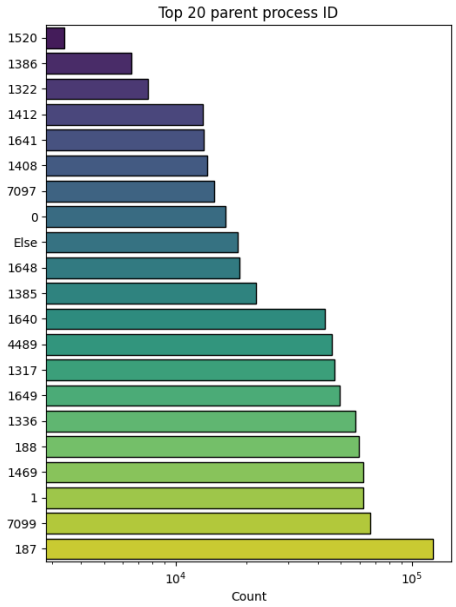


Figure 3: Parent process ID

- 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.

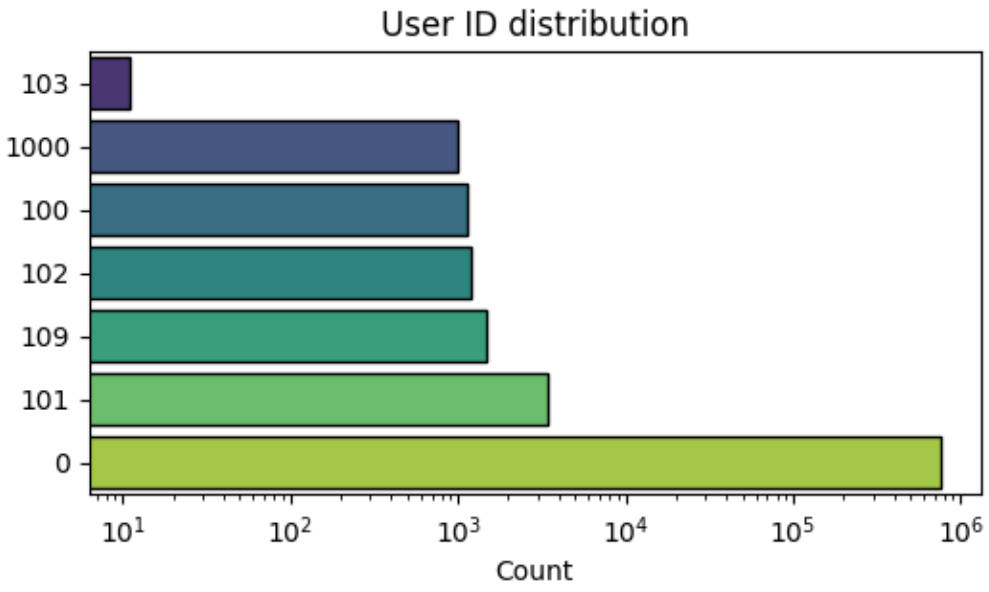


Figure 4: User ID

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

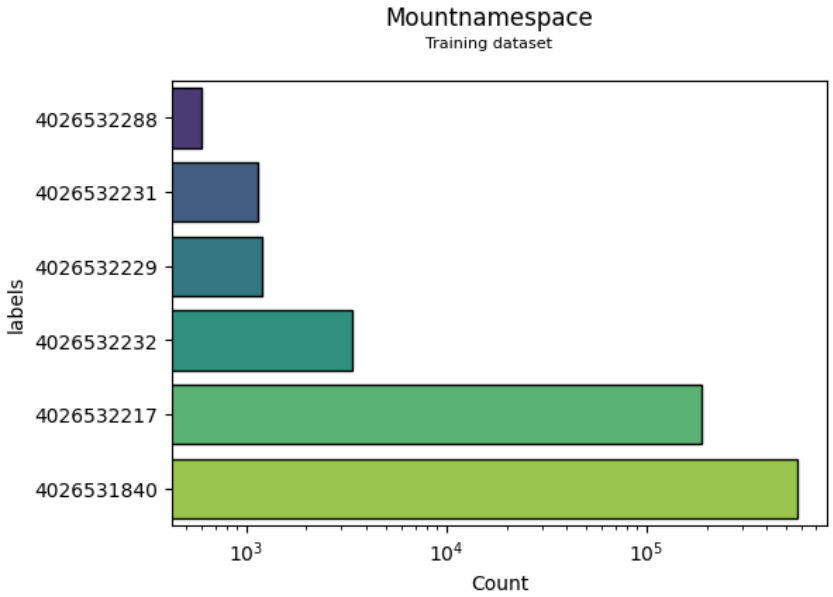


Figure 5: Mount name space

- processName: command executed (string)

  – The dataset is dominated by a few process names, with "ps" being the most frequent.
  – A large number of process names have very low counts, contributing to a highly skewed distribution.

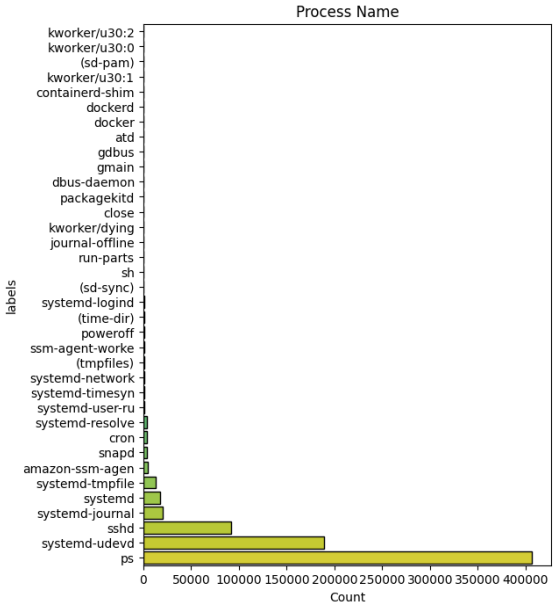


Figure 6: Process name

- hostName: host server (string)

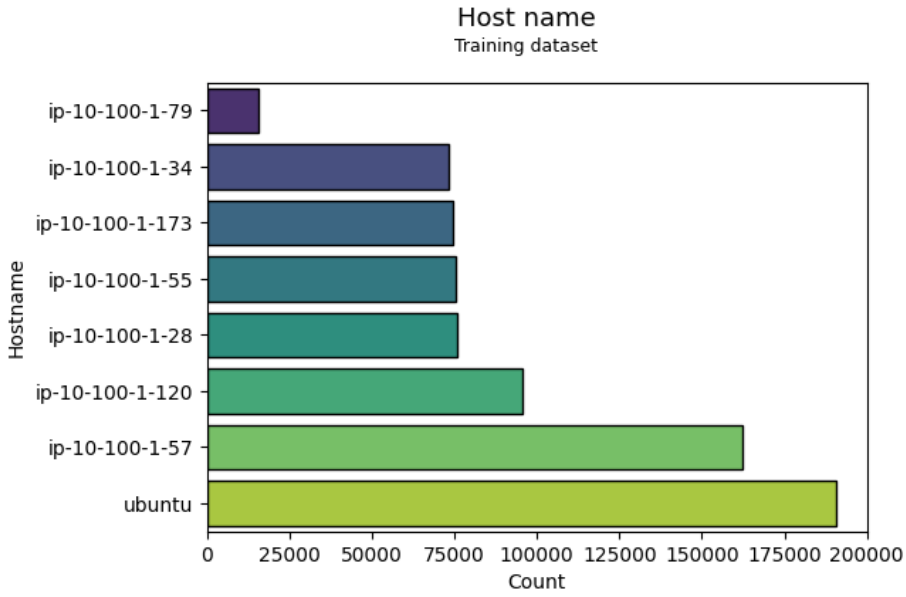


Figure 7: Host name

- eventId: id of the event generating this log (integer)

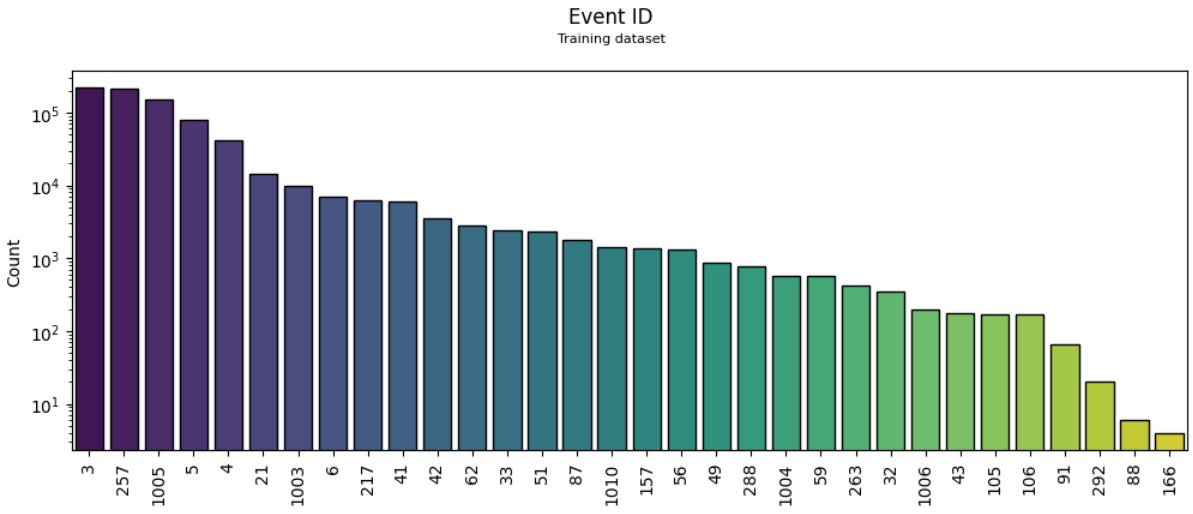


Figure 8: Event ID

- eventName: name of the event (string)

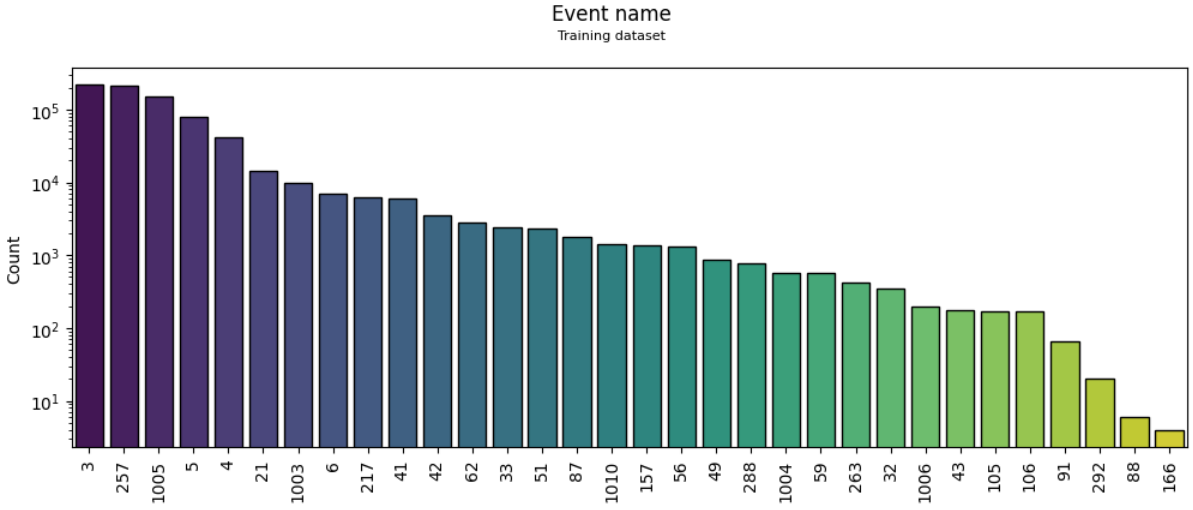


Figure 9: Event name

- 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).

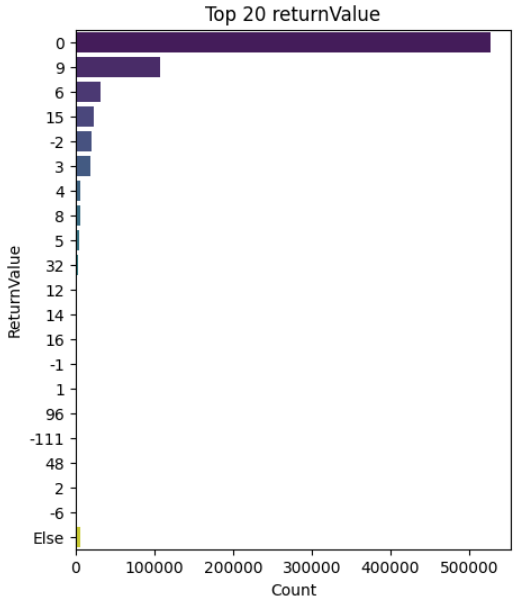


Figure 10: Return value

- 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.

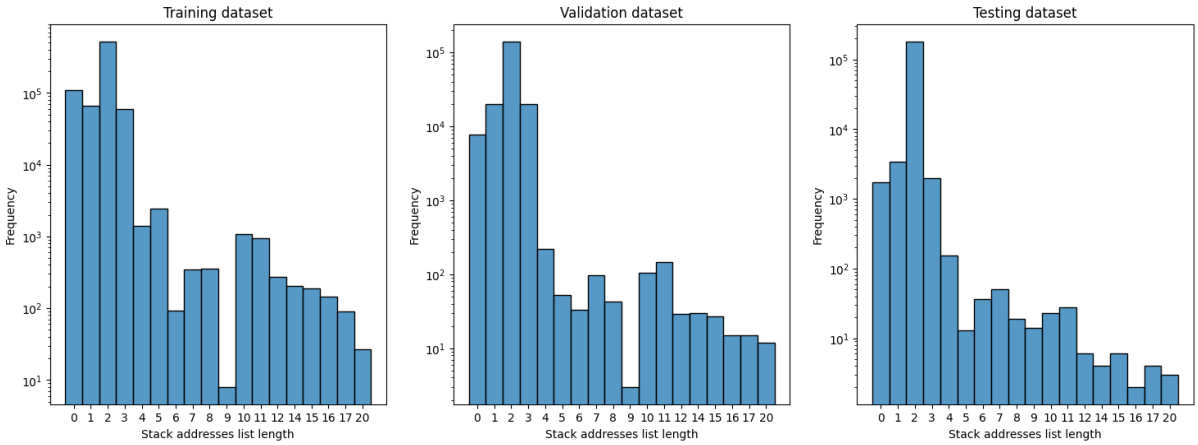


Figure 11: Stack length name

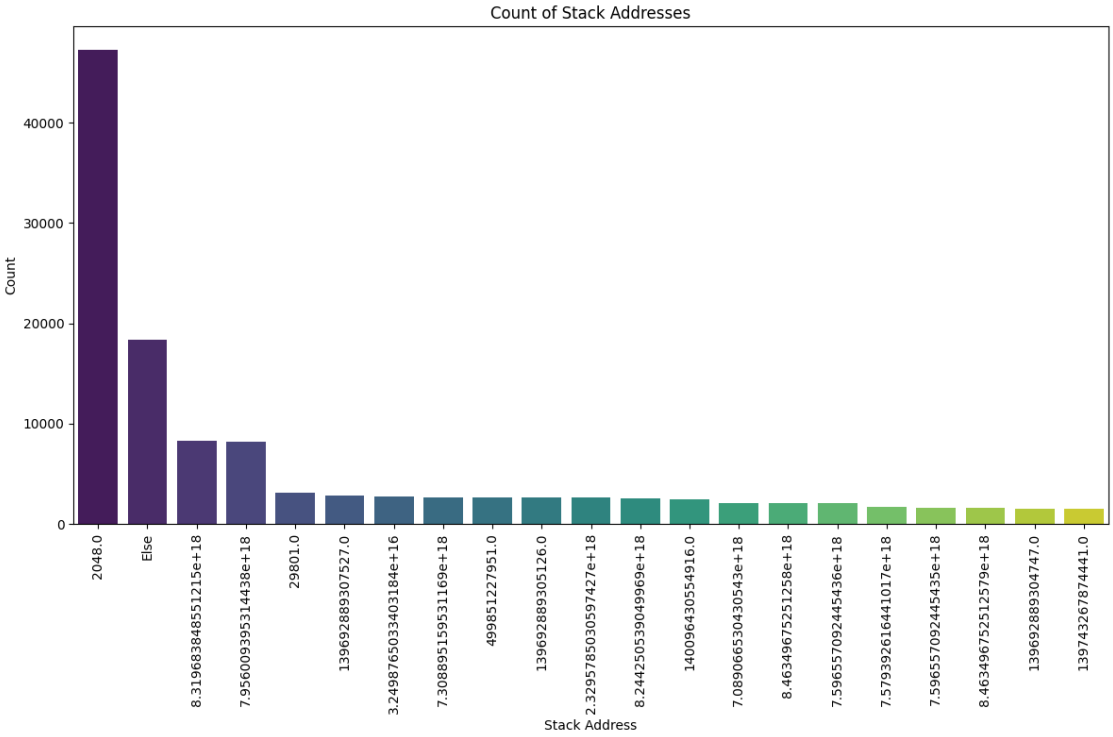


Figure 12: Stack addresses

- argsNum: number of arguments (integer)

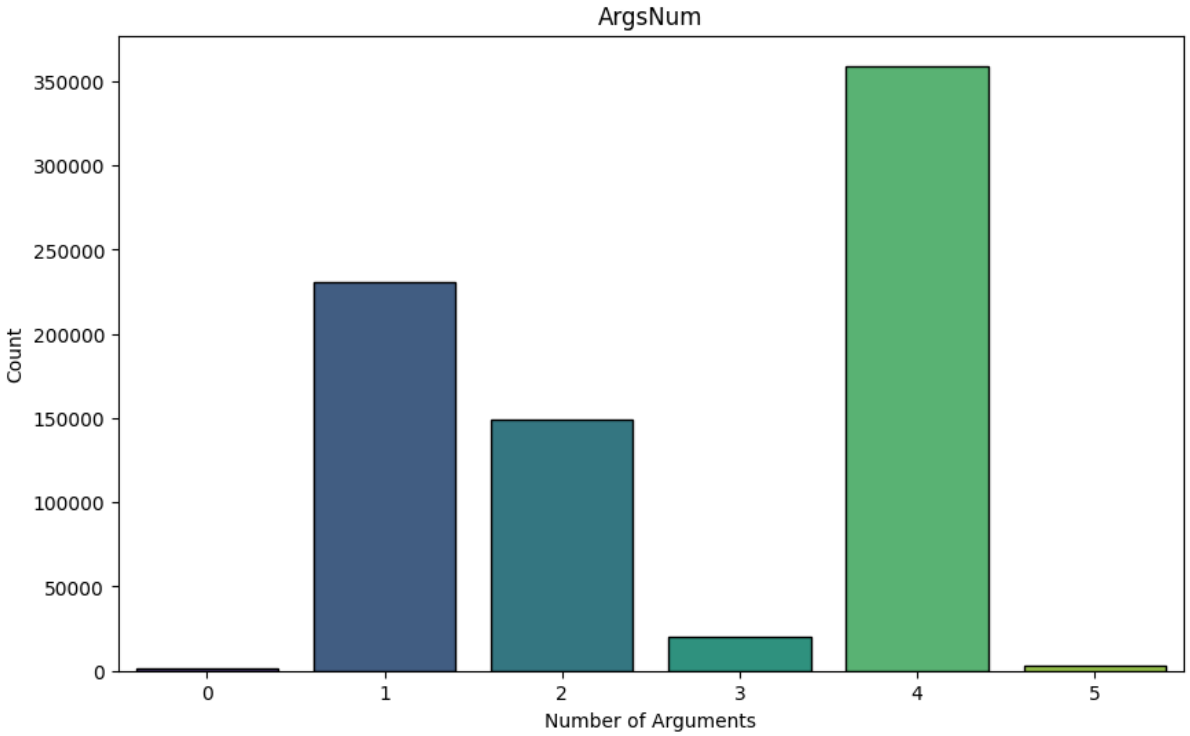


Figure 13: Args

- 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.

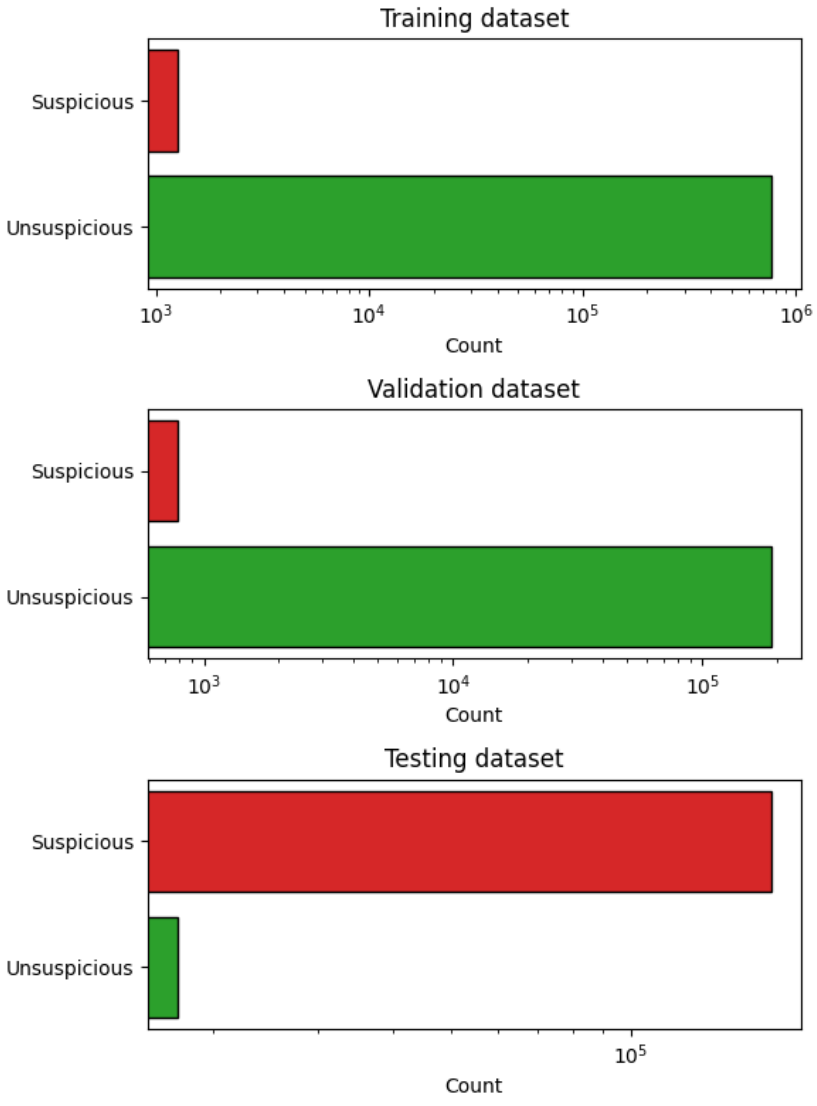


Figure 14: Sus label

- 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.

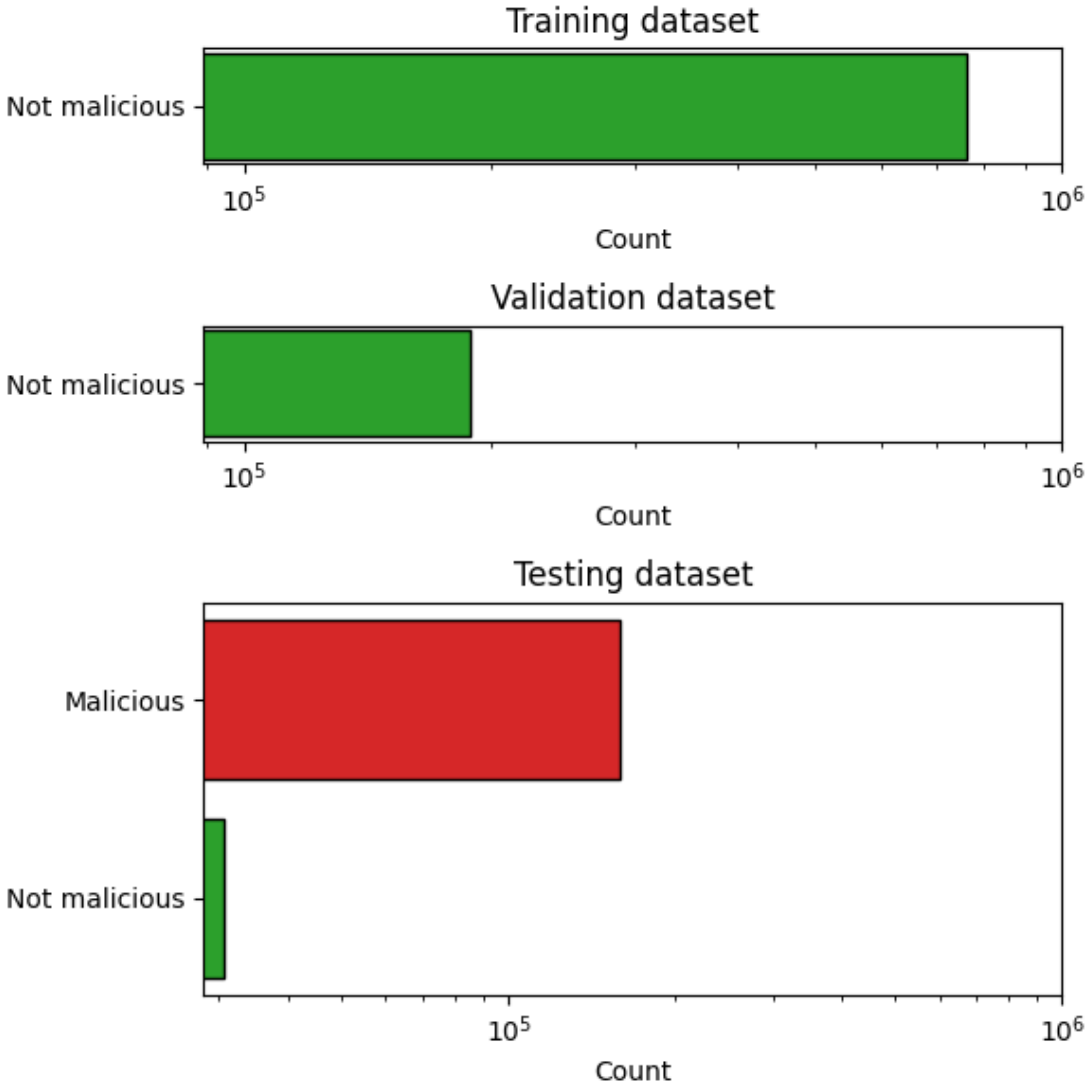


Figure 15: Evil label

2.2 Correlation matrix

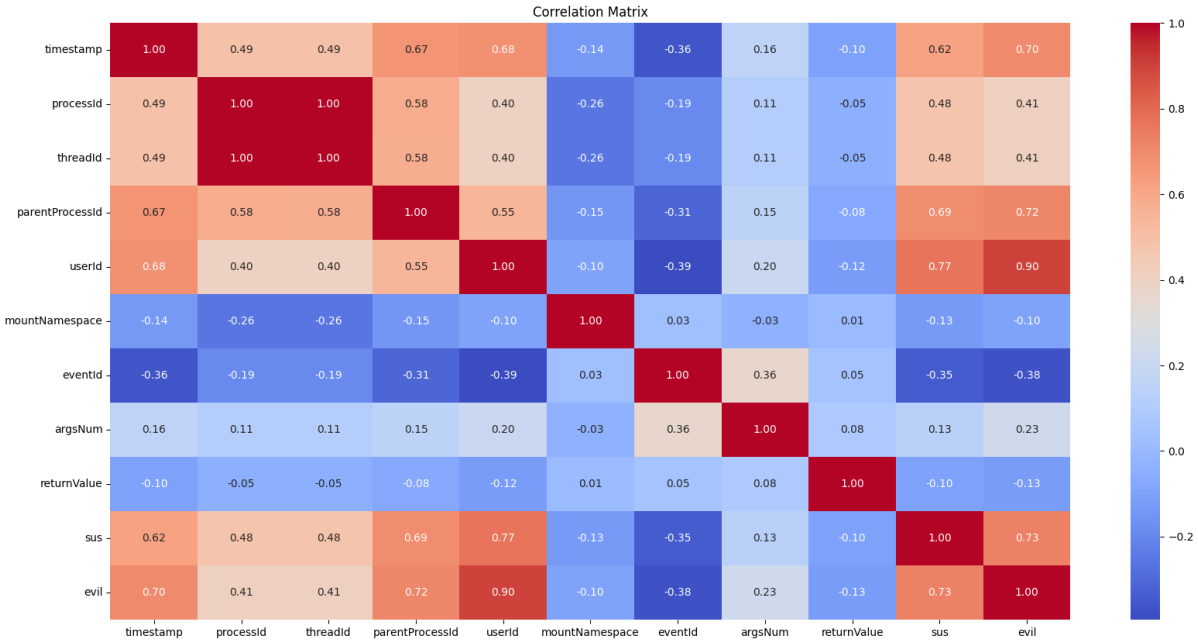


Figure 16: Correlation matrix

- 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 suspicius and not suspicius event:

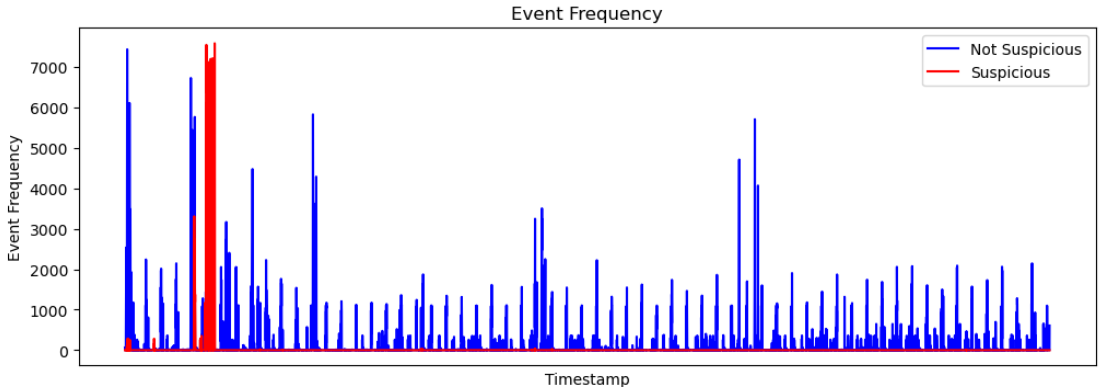


Figure 17: Suspicius and not suspicius event

- 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 adviced 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 approch 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)

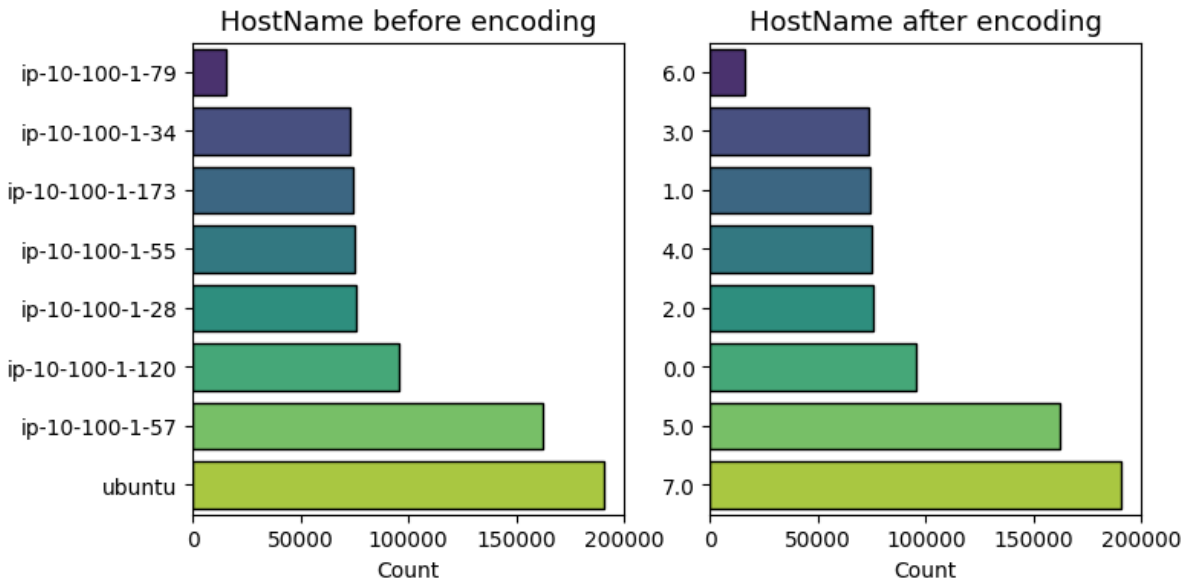


Figure 18: Ordinal encoder

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 althought, as it is our case, minority class is more important. To balance that, we tried to use Smote library combined as it is adviced with randoom undersampling for the majority class. SMOTE (Synthetic Minority Oversampling TEchnique) works by interpolating new instances along line segments joining existing minority class instances.

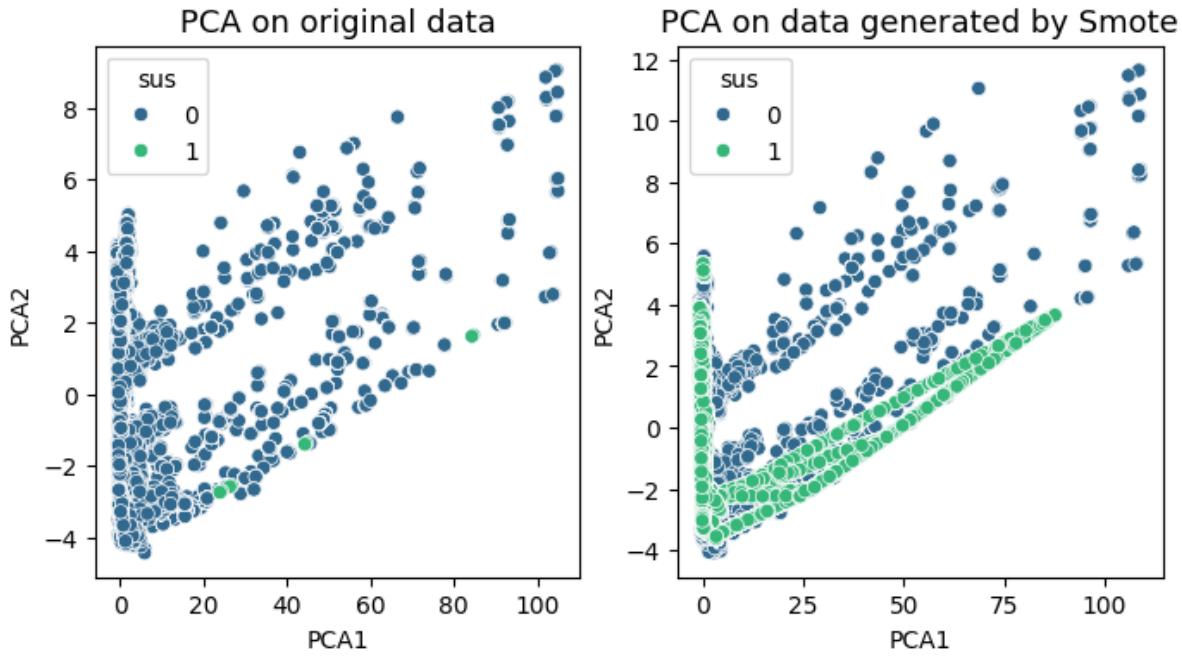


Figure 19: Smote

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.

The following chart describe how the comparison with the suspicious activity differs to the comparison with the not suspicious activity.

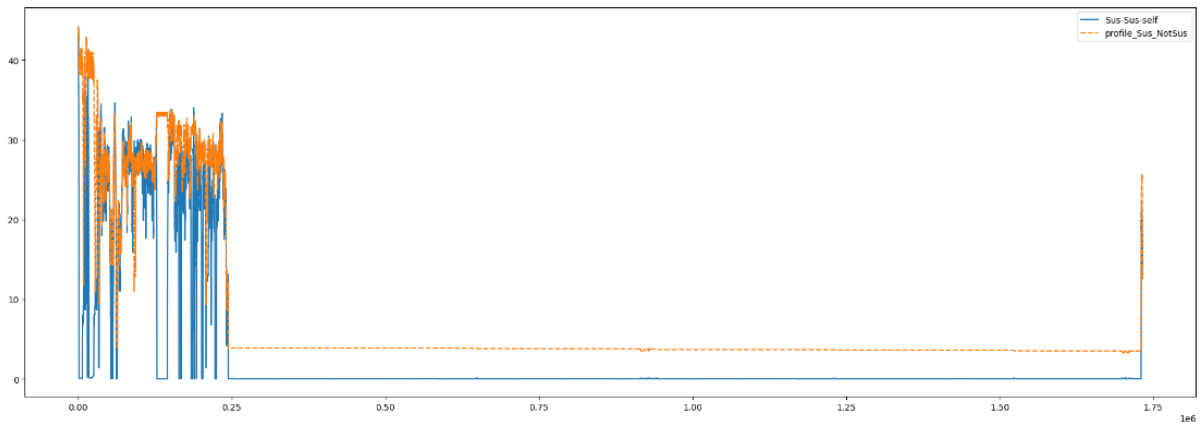


Figure 20: Shapelet discovery

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.

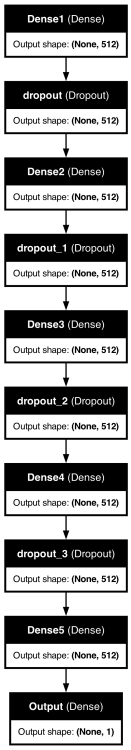


Figure 21: Dense neural network - Model 1

- **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.

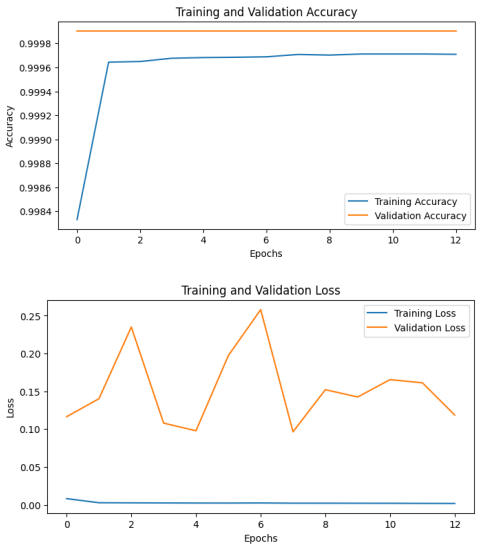
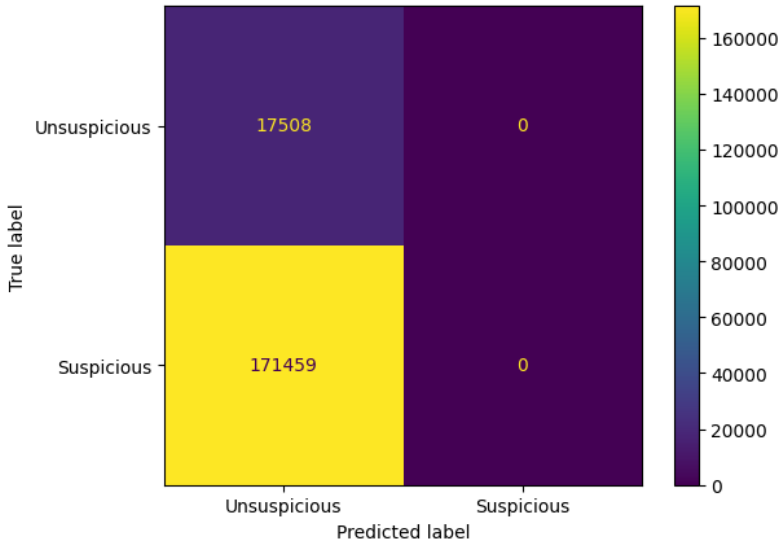


Figure 22: Model 1 accuracy and loss

- **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.



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.

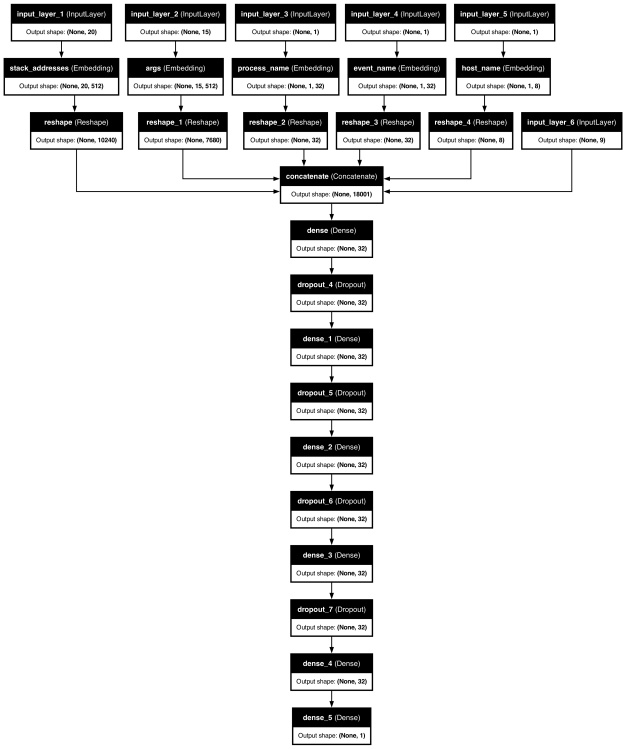


Figure 23: Dense neural network - Model 2

- **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.

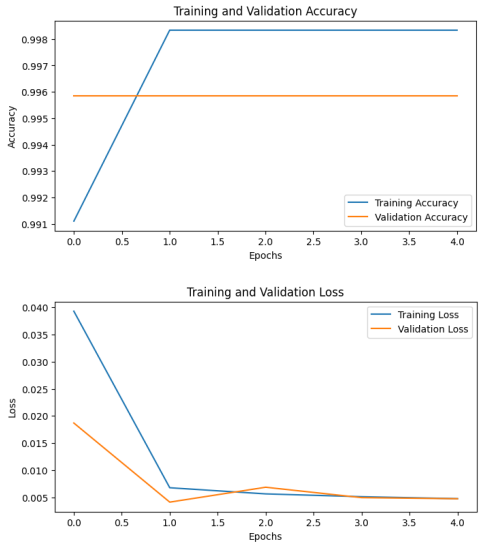
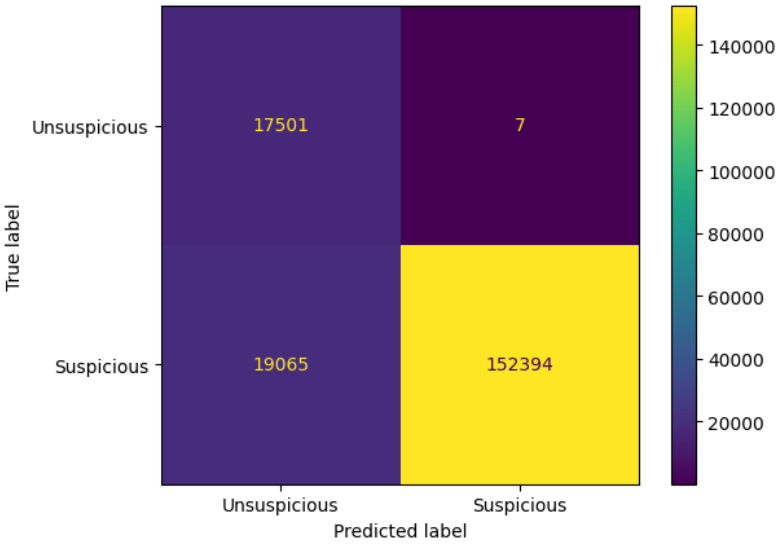


Figure 24: Model 2 accuracy and loss

- **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.



4.1.3 Model 3

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

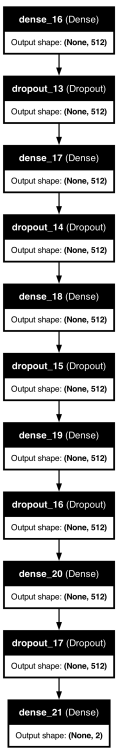


Figure 25: Dense neural network - Model 3

- **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.

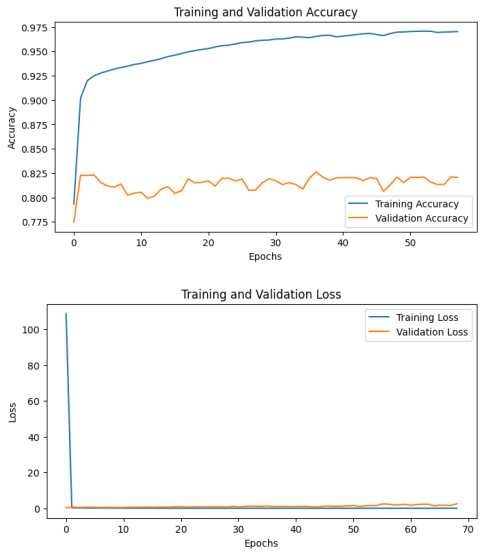


Figure 26: Model 3 accuracy and loss

- **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.

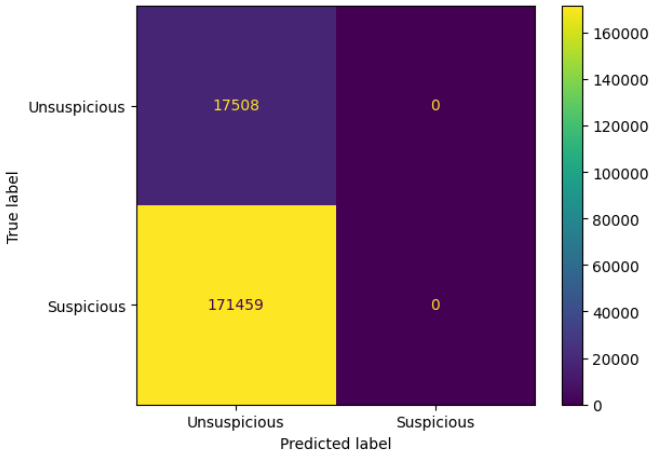


Figure 27: Model 3 Confusion matrix

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.

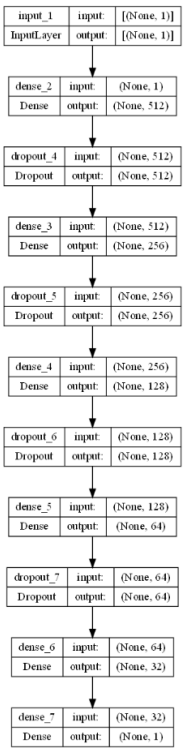


Figure 28: Dense neural network - Model 4

- **Training**

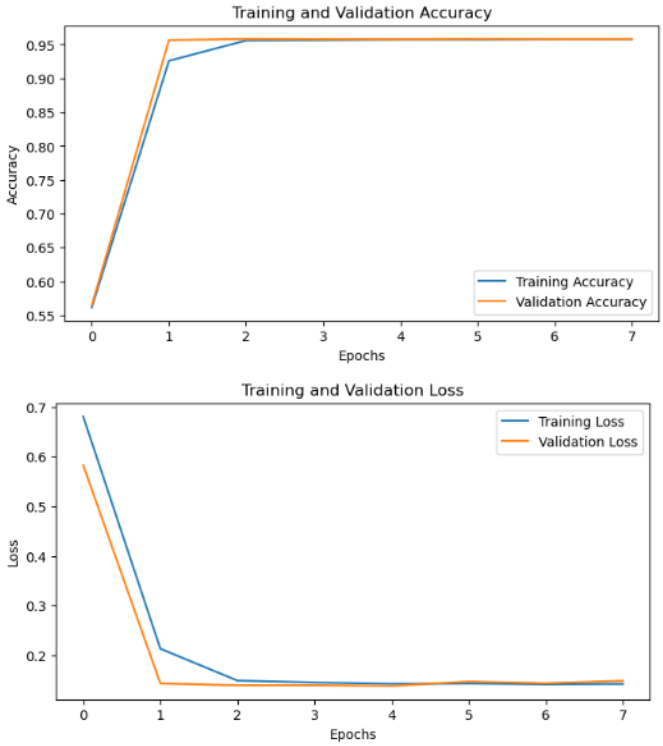


Figure 29: Model 4 accuracy and loss

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

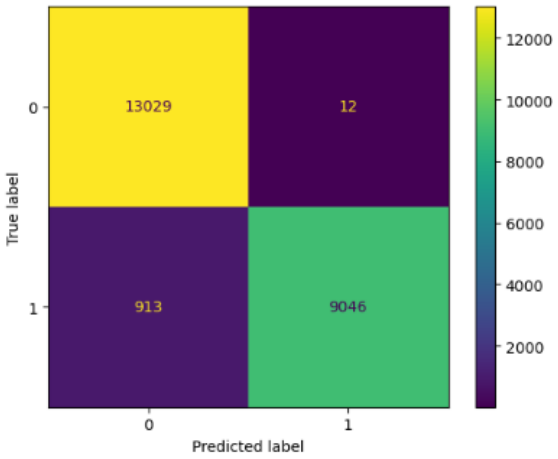


Figure 30: Model 4 confusion matrix

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.

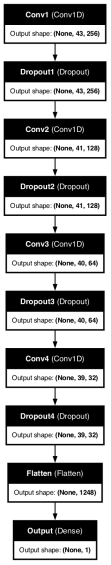


Figure 31: Conv. neural network - Model 5

- **Training**

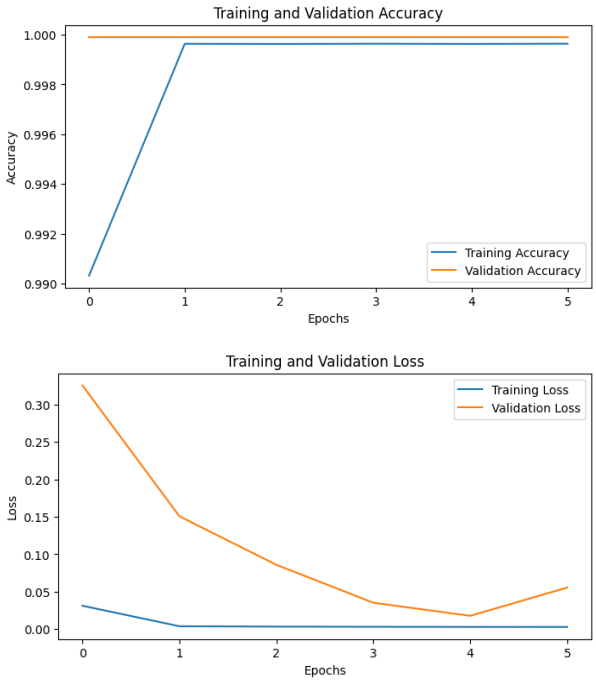


Figure 32: Model 5 accuracy and loss

- **Prediction**

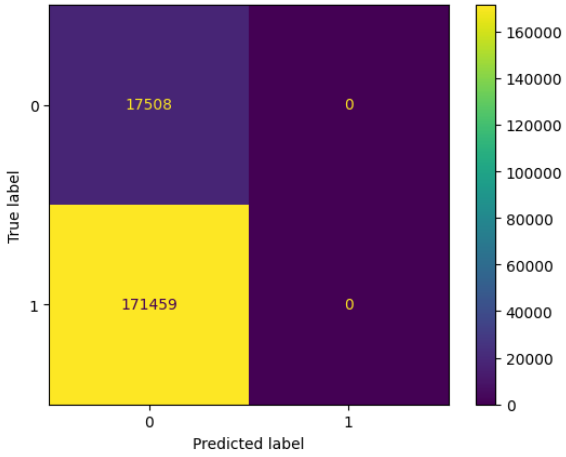


Figure 33: Model 5 Confusion matrix

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.

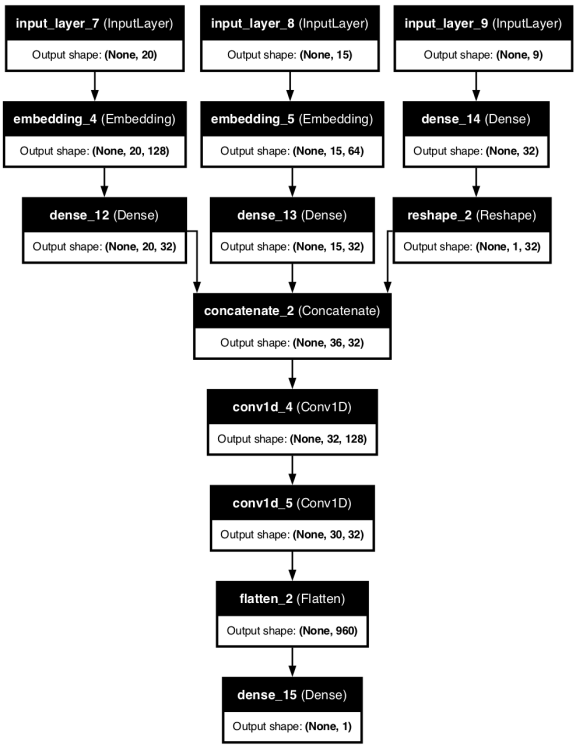


Figure 34: Conv. neural network - Model 6

- **Training**

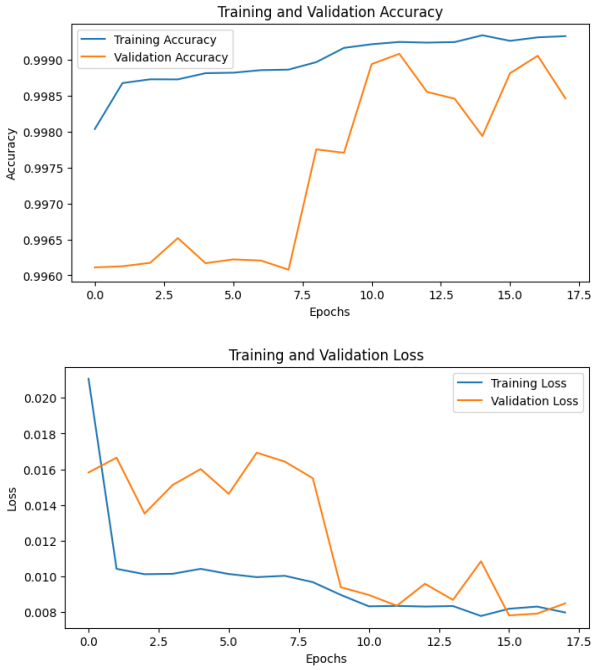


Figure 35: Model 6 accuracy and loss

- **Prediction**

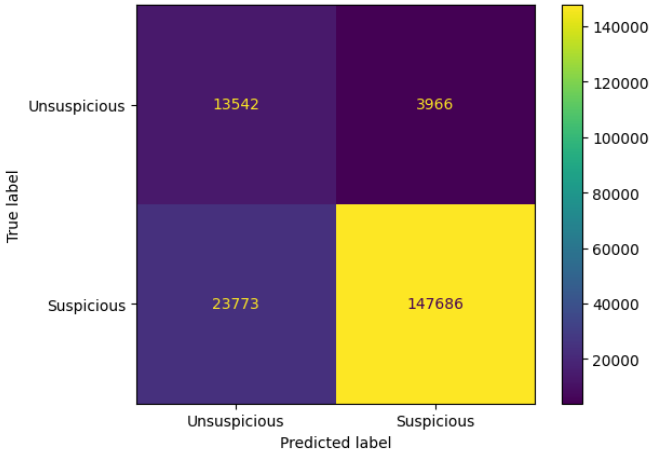


Figure 36: Model 6 Confusion matrix

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.

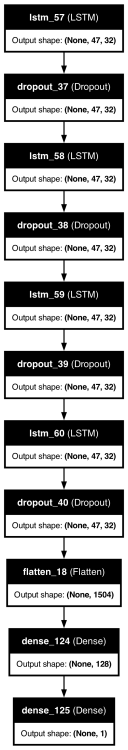


Figure 37: LSTM - Model 7

- **Training**

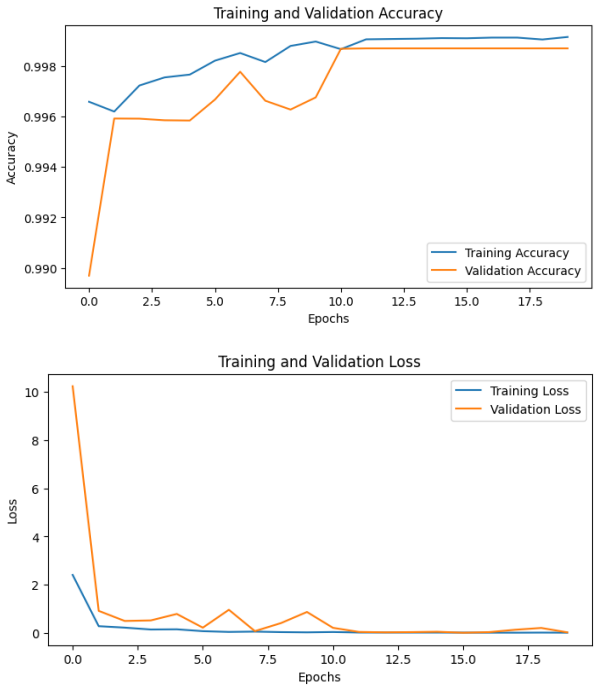


Figure 38: Model 7 accuracy and loss

- **Prediction**

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

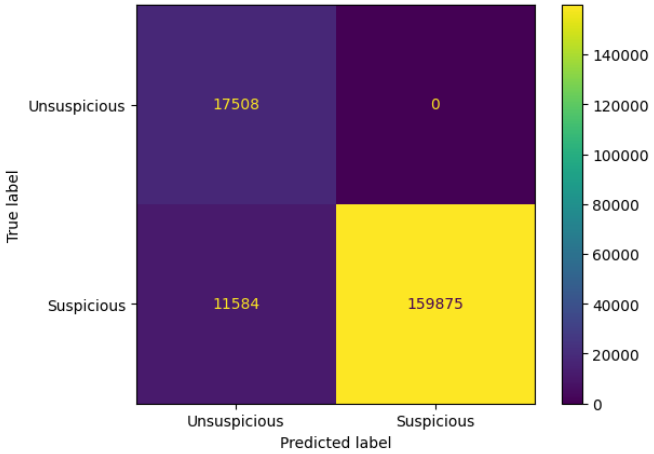


Figure 39: Model 7 Confusion matrix

negative rate with 17,508 correct unsuspicious classifications. However, it still misses 11,584 suspicious instances

4.3.2 Model 8

- **Description**

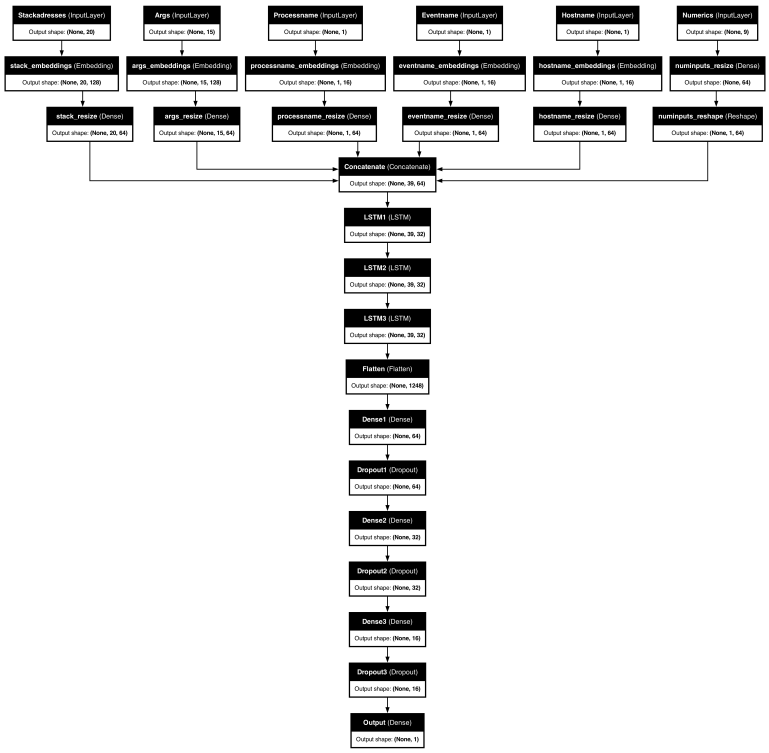


Figure 40: LSTM - Model 8

- **Training**

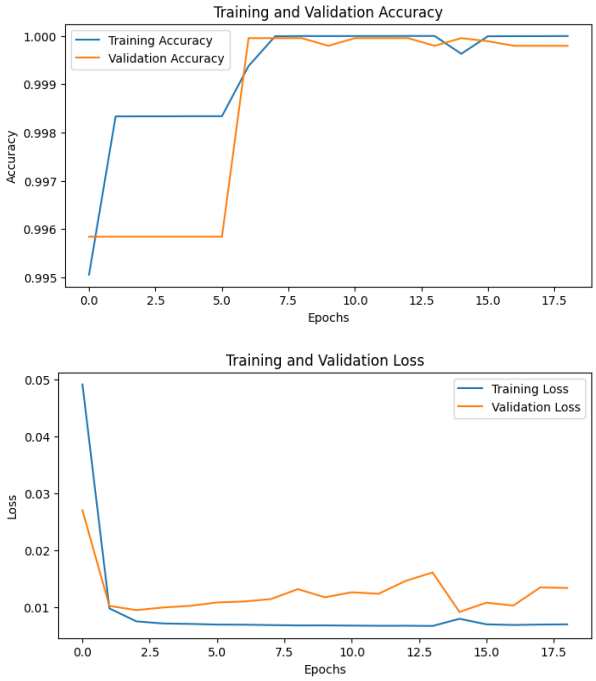


Figure 41: Model 8 accuracy and loss

- **Prediction**

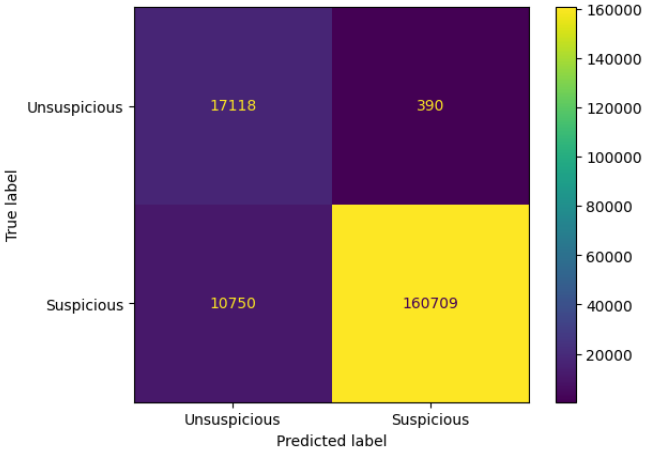


Figure 42: Model 8 Confusion matrix

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.

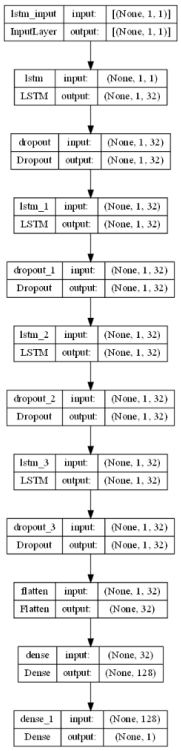


Figure 43: LSTM - Model 9

- **Training**

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

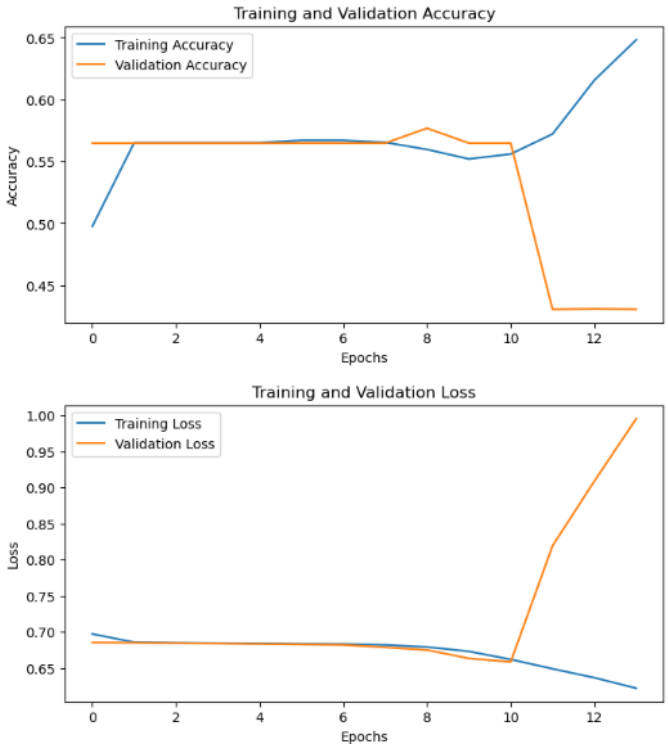


Figure 44: Model 9 accuracy and loss

- **Prediction**

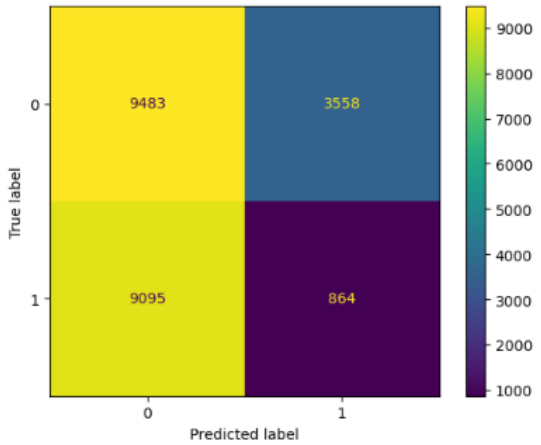


Figure 45: Model 9 Confusion matrix

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.

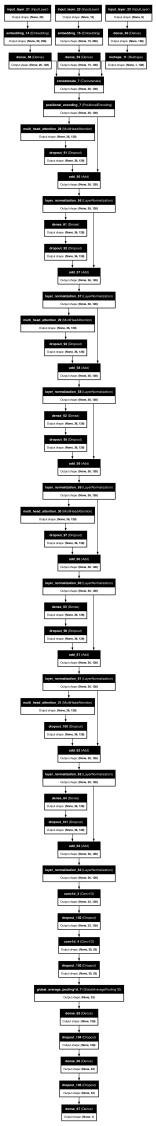


Figure 46: Transformer structure

```
- Positional encoding adds information about the position of each element in the sequence by creat
 This matrix is added to the input embeddings, allowing the model to incorporate the order of the

 - The transformer_encoder_block applies multi-head self-attention to the inputs, enabling the mod
 This is followed by a dropout layer for regularization and layer normalization to stabilize and s
 Finally, a feed-forward neural network with a dense layer, dropout, and another layer normalizati

 - This model uses Adam optimizer with a <b>WarmUpCosineDecay</b> learning rate.\
```

```
During the warmup phase, the learning rate increases linearly, and once the warmup steps are comp
```

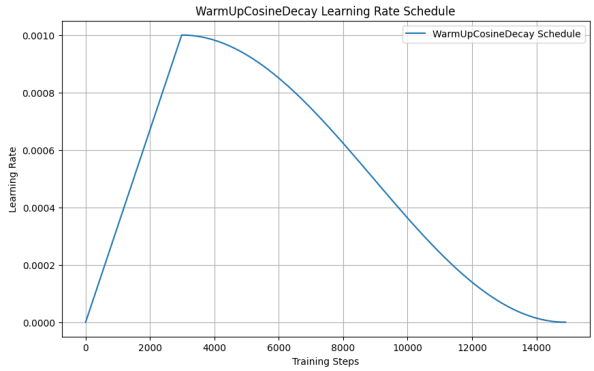


Figure 47: Learning rate decay

- **Training**:

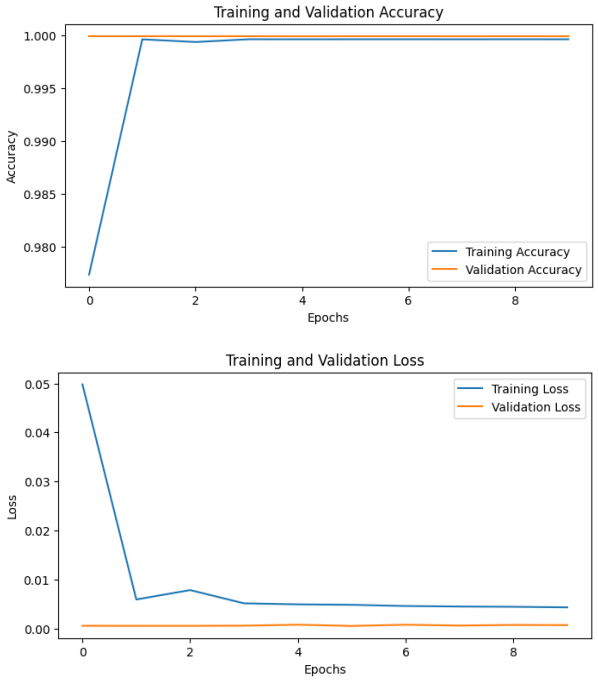


Figure 48: Transformer accuracy and loss

- **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.

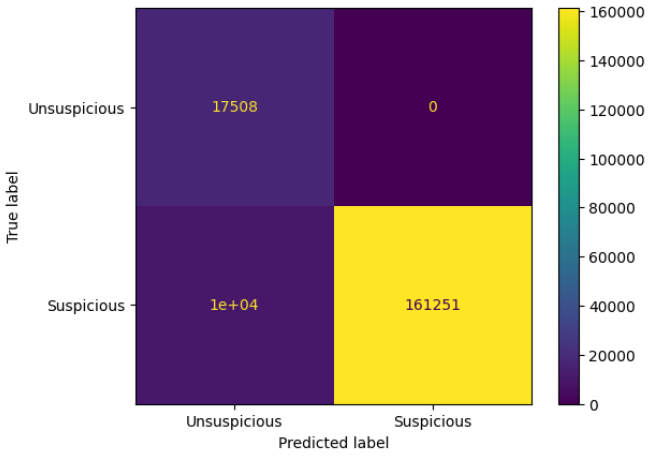


Figure 49: Transformer Confusion matrix

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.

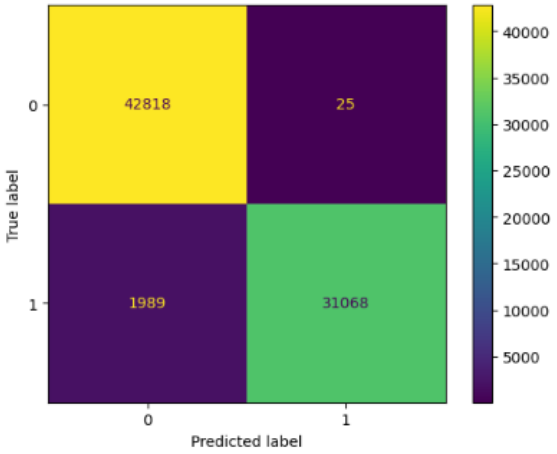


Figure 50: Transformer Confusion matrix

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 51: Results

6 Discussion

- Most of the models present good validation accuracy and seem to converge rapidly, which could be due to the similarity between the training and validation datasets compared to the testing dataset or because the task is relatively simple.
- Most of the models achieve a high recall score for the unsuspicious class (100% for LSTM model1), indicating that unsuspicious instances are well predicted. However, the precision score for the same class does not exceed 63%, which may be due to the larger number of unsuspicious instances in the training dataset.
- The Shapelet Discovery method, when applied to models such as the Decision Tree Classifier or the Dense models on a small sample of 230.000 data points, it gives good accuracy levels. Therefore, this methodology, because of the structure of the BETH dataset (imbalances between the classes) and the high computational level that it requires, it doesn't seem the most effective method.
- Embeddings: The use of embeddings significantly enhances the performance of models across the board, particularly for CNN and RNN models.
- Model Performance: CNN and RNN models with embeddings, as well as the transformer model, exhibit high accuracy, precision, recall, and ROC scores.
- Model Suitability: Dense models without embeddings and LSTM models in the shapelet discovery method underperform, indicating that they may require additional tuning or may not be suitable

for this specific task.

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