Major Project Report

on

Anomaly Detection in Cyber security

Submitted by

B Jagadish Chandra 20BCS032

Mokalla Bhanu Prasad 20BDS033

Taneti Ravi Chand 20BDS057

Vineeth S R 20BCS137

Under the guidance of

Dr. Rajendra Hegadi

Dean - Academic and Student affairs

DEPARTMENT OF DATA SCIENCE AND INTELLIGENT SYSTEMS

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY DHARWAD

28/03/2024

Certificate

This is to certify that the project, entitled **Anomaly Detection in Cyber Security**, is a bonafide record of the Major Project coursework presented by the students whose names are given below during 2023 – 24 in partial fulfilment of the requirements of the degree of Bachelor of Technology in Computer Science and Engineering.

Roll No	Names of Students
20BCS032	B Jagadish Chandra
20BDS033	Mokalla Bhanu Prasad
20BDS057	Taneti Ravi Chand
20BCS137	Vineeth S R

Dr.Rajendra Hegadi
(Project Supervisor)

Contents

List of Figures

List of Tables

Abstract

Among the most important cyber security tasks is anomaly detection, which it looks for unusual or harmful activity occurring within a framework or network. Different models of machine learning are employed in this particular situation to efficiently identify irregularities and possible security risks. Local Outlier Factor (LOF), Random Forest, and Isolation Forest are three different anomaly detection models. Every model undergoes training on labeled datasets and is then validated to be able to evaluate its performance. To be able to assess these models' accuracy in identifying anomalies, classification reports and confusion matrices are computed. During validation, the Isolation Forest model shows a high degree of precision in detecting anomalies. But when applied to the test dataset, its performance seems to deteriorate, leading to a decrease in overall accuracy. Conversely, however, the Random Forest model, which is generally employed in supervised learning tasks, demonstrates strong anomaly detection abilities and attains a high level of accuracy in both the validation and test datasets. Comparably, both datasets exhibit good performance from the Local Outlier Factor model, which is intended to identify anomalies based on local density deviations. There is one prediction function, that is accustomed to predict the anomaly status of synthetic data points from trained models. It shows how different models classify synthetic data differently. While both the Random Forest and LOF models label the synthetic data as anomalies, the Isolation Forest model classifies them as normal. These models provide different methods for cyber security anomaly detection, each with advantages and disadvantages, meeting different needs and characteristics of the information to spot possible security breaches.

1 Introduction

Finding anomalies is essential to cyber security because it protects networks and systems from harmful actions like intrusions and cyber attacks. An essential component of cyber security is anomaly detection, which acts as a preventative measure to spot and address anomalous patterns, behaviors, or occurrences in digital environments. Anomaly detection is essential for strengthening defenses against malicious activities in the dynamic field of cyber security, where threats are always changing. Its main goal is to identify typical behavior and differentiateit from anomalous or potentially dangerous activity so that security threats can be identified early on and countered. Anomaly detection uses advanced methods of data analysis, like statistical methods and machine learning algorithms, to identify irregularities in the vast domain of cyberspace, which includes networks, systems, applications, and user interactions. Anomaly detection systems can identify deviations and other deviations that might point to security incidents, like cyber attacks, intrusions or unapproved entry by creating a baseline of typical behavior. It is imperative to employ anomaly detection methods for spotting odd and possibly dangerous behavior in large and complicated datasets that deviates from regular patterns. The purpose of using anomaly detection methods is to identify patterns or occurrences within a dataset that substantially deviate differently from the norm. These techniques are essential in many fields, including industrial monitoring, finance, cyber security, and healthcare. A range of techniques, from sophisticated algorithms for machine learning to conventional statistical methods, are employed to find anomalies. In cyber security, anomaly detection is essential. Different models are vital in spotting anomalies in datasets to protect systems and networks from new threats and guarantee the privacy of sensitive information. It gives a quick summary of the goal, which is to compare the efficiency of three distinct anomaly detection algorithms that is, Local Outlier Factor (LOF), Random Forest, and Isolation Forest. Especially, these models get along well with large datasets and high-dimensional data, Random Forest and Isolation Forest are especially skilled at quickly identifying anomalies and strong detection capabilities, respectively. Simultaneously, the Local Outlier Factor (LOF) measures the nearby density deviation, By using these models, cyber security initiatives progress over time, strengthening networks and systems resistance to changing cyber threats. this study aims to support the advancement of methods for detecting anomalies in cyber security, providing valuable tools for safeguarding systems and networks against emerging threats and ensuring the security and integrity of sensitive data. This paper examines the workflow that comprises training models, pre processing data, validation, and testing regarding various machine learning algorithmsin relation to cyber security anomaly detection. evaluation metrics such as recall, precision, F1-score, and confusion matrix visualizations are employed to evaluate these algorithms efficacy that perform in anomaly detection tasks. Additionally, it highlights how important It’s to evaluate models using labeled datasets in order to assess their level of able to distinguish between normal and anomalous instances. The overall framework that will be applied to analyze and assess the anomaly detection models later on, emphasizing their value and applicability in actual situations. These models offer several methods for cyber security anomaly detection, each with advantages and disadvantages. In a network or system, one or more of these models may be useful for spotting irregularities and spotting possible security risks, depending on the particular requirements and data characteristics.

2 Objective

The objective of this study is to compare the efficiency of three distinct anomaly detection algorithms, namely Local Outlier Factor (LOF), Random Forest, and Isolation Forest, in the realm of cyber security. The aim is to provide valuable insights into the performance and applicability of these models for safeguarding systems and networks against emerging threats and ensuring the security and integrity of sensitive data.

- **Evaluation of Anomaly Detection Models:** Assess the effectiveness of LOF, Random Forest, and Isolation Forest algorithms in identifying anomalies within cyber security datasets.

- **Performance Analysis:** Utilize evaluation metrics such as recall, precision, F1-score, and confusion matrix visualizations to analyze the efficacy of these algorithms in anomaly detection tasks.

- **Comparison of Model Outcomes:** Conduct a comparative analysis to determine the relative advantages and disadvantages of each model in detecting irregularities and potential security risks.

3 Related Work

The term ”related work” refers to a broad range of methods, applications, and research studies in the vicinity of unusual finding.

3.1 Anomaly Detection in Cyber-Physical Systems

This comprehensive an analysis of the available sources looks at various methods for identifying anomalies in cyber-physical system (CPS) environments, with a focus on CPS. It discusses strategies like data-driven approaches, machine learning, and statistical methods, emphasizing their uses and drawbacks in CPS security.

3.2 Identification of Anomalies in Industrial Internet of Things

This study focuses on Identification of Anomalies in Industrial Internet of Things (IIoT) environments, offers details regarding the challenges and remedies associated with anomaly detection in industrial settings. It covers methods designed specifically for the characteristics of IIoT data and talks about practical uses in manufacturing and industrial automation.

3.3 Anomaly Detection in Healthcare

This paper, which centers on healthcare applications, investigates anomaly detection methods for illness outbreak detection, patient monitoring, and the detection of unusual medical conditions. It discusses how machine learning and statistical methods are employed to find irregularities in healthcare and how this helps to improve patient outcomes.

3.4 Anomaly Detection in Financial Data

This survey, which focuses on financial applications, covers anomaly detection methods for identifying insider trading, market manipulation, and fraudulent transactions in financial datasets. It examines algorithms for machine learning, conventional statistical techniques, and the greatest recent developments in anomaly detection based on deep learning.

4 Data and Methods

4.1 Dataset

All models use training, validation, and testing datasets of information accompanied by labels for each feature. The datasets used in the analysis include labeled data about events and system processes. Every dataset entry has multiple features that shed light on how various system activities behave. The processId, parent processId, userId, mount namespace, process name, eventId, event name, quantity of arguments connected to the event, and return value are some examples of these features. Every entry has a label that indicates whether it is a normal behavior (labeled as 0) or an anomaly (labeled as 1). The creation and assessment of anomaly detection models, which seek to spot unusual patterns or behaviors in system events and processes, are made possible by this labeled dataset. The dataset's comprehensiveness makes it possible to examine system behavior in detail and makes it easier to develop models for machine learning that can accurately identify and categorize anomalous activity. These datasets are working in the training, validation, and assessment of performance of the corresponding models.

4.2 Dataset Characteristics

The collection of data used for the analysis includes labeled data related to events and processes within the system. Every entry in the dataset has a number of characteristics that illuminated various facets of system operations. processId, parent processId, userId, mount namespace, process name, eventId, event name, quantity of arguments, and return value are some of these features. When taken as a whole, these characteristics provide a thorough picture of the interactions and behavior of the system. Additionally, a label indicating whether an entry represents normal behavior or an anomaly is attached to each one. The basis for developing and assessing models for machine learning that aim to identify irregularities in system events and processes is this labeled dataset.

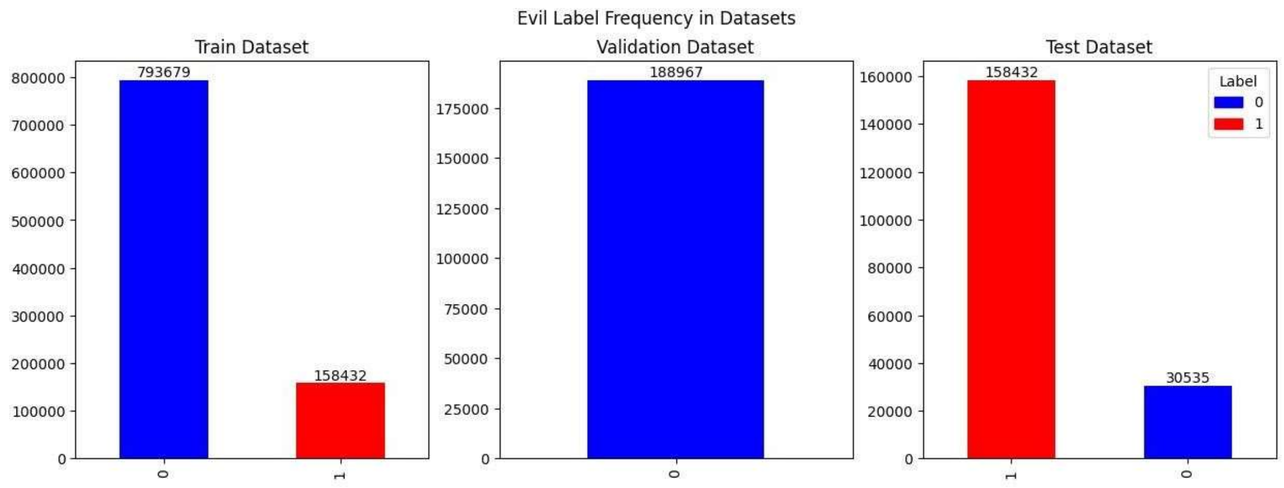


Figure 1. Evil Label Frequency in Datasets

4.3 Pre processing

Preprocessing methods are applied in this section to get the dataset ready for training in the Isolation Forest, Random Forest, and Local Outlier Factor (LOF) models. These techniques include applying label encoding to categorical variables such as processName and eventName, and mapping specific features to binary values. To simplify their representation for the models, features like processId, parentProcessId, userId, and mountNamespace are mapped to binary values. In order to facilitate model training, additional label encoding is applied to the processName and eventName columns, converting categorical data into a numerical format. The similarities among the features employed by the three models attest to their applicability in identifying anomalies and malevolent activity in system operations. Each of these characteristics adds useful details about user interactions, system events, and processes, which helps the models distinguish between anomalous and typical behavior. and also, we will see a summary of the pre processing techniques and features applied to each model:

- **timestamp:** This feature could add needless noise to the dataset because it might not offer insightful information about security risks. So, we removed this feature.

- **processId:** It is crucial for monitoring specific processes and their actions, as this aids inthe detection of irregularities and possibly harmful system behavior.
- **threadId:** We also removed this feature because, adding this feature could result in high-dimensional data, which would raise the computational complexity and resource needs for training and inferring models without yielding performance gains equivalent to thosecosts.
- **parentProcessId:** This is vital for tracking the flow of activity within the system, identifying unauthorized process executions, and comprehending the relationships between processes.
- **userId:** It aids in identifying unauthorized access attempts, assigning specific users' actions to the system, and watching out for any potential privilege escalation.
- **mountNamespace:** This feature improves system security and integrity by helping to identify unauthorized mounts, file system configuration changes, and attempts to access restricted file systems.
- **processName:** This feature is also essential for figuring out which programs or binariesare causing system activity, making it possible to find potentially malicious software and unknown or dubious executable programs.
- **hostName:** This feature makes it easier to identify the systems that are communicating over a network, track the source of network traffic, and find anomalies like DNS spoofingor network-based attacks.
- **eventId:** It is vital for monitoring system activities and events, assisting in the recognitionof odd or suspicious occurrences additional anomalies in system behavior.
- **eventName:** This feature helps to identify patterns of behavior suggestive of security threats and helps comprehend the nature of system activities by providing descriptive labels for system events.
- **argsNum:** It aids in the identification of anomalies and possible exploit attempts by providing insights into the complexity of system activities and departures from typical behavior.
- **returnValue:** shows the result or state of system operations, making it possible to evaluate the success or failure of an operation and identify any mistakes, malfunctions, or security breaches.

- **stackAddresses:** This feature is hard to relate to clearly in a manual analysis, and processing automatically is challenging due to the large values in a variable size list without acquiring a new embedding or encoding. As a result, we stopped using this feature for training.
- **args:** This feature may put users privacy at risk because they may contain sensitive data.

In summery, These qualities are necessary for tracking system behavior, identifying irreg- ularities, and preserving the system's security, integrity, and dependability across a range of models and analyses. The elimination of a few particulars from every models, such as timestamp, threadId, args, and stackAddress, was motivated by a number of objectives related to improving the efficacy and efficiency of anomaly detection procedures. Initially, it's It's probable that these traits have no direct bearing on identifying irregularities or malevolent conduct in system operations. Thus, disabling these features contributes to easing these worries and guarantees adherence to data protection laws.

4.4 Label Encoding

Mapping features like processName and eventName using label encoding is crucial for preparing categorized data for machine learning algorithms in the context of anomaly detection in system processes. By transforming numerical data from category variable representations, label encoding makes it It is feasible for machine learning models to handle them efficiently. Strings containing the names of processes and events, respectively, may be found in processName and eventName. However, using string values directly might not be appropriate because typically, machine learning algorithms with numerical data. By giving each category in the categorical features a distinct numerical label, label encoding transforms the categories into a format that the algorithms can comprehend and process. We guarantee that processName and eventName make a significant contribution to the model training by employing label encoding for their mapping. This makes it possible for the models to discover connections and patterns among various actions and occurrences, improving their capacity to identify abnormalities based on these characteristics.

4.5 Correlation Analysis

The train data's correlation heatmap sheds light on the connections between the various features in the dataset. The degree of correlation between two characteristics is represented by each cell in the heatmap, which ranges from -1 to 1. A perfect positive correlation is represented by a value of 1, a perfect negative correlation by a value of -1, and no correlation is represented by a value of 0.

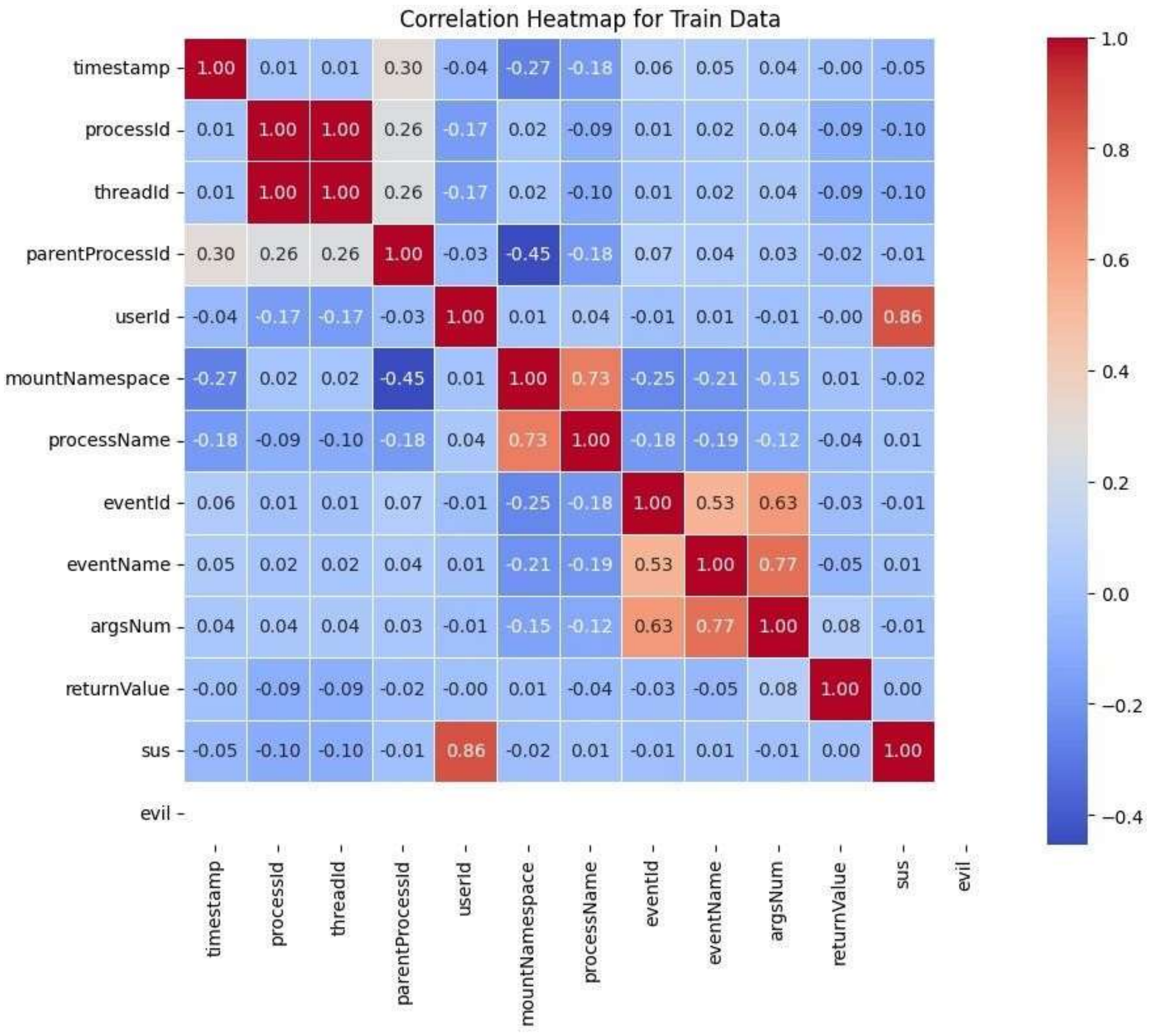


Figure 2. Correlation Heatmap for Train Data

The ”plot correlation heatmap” function is used to create the correlation heatmap for the train data. The heatmap is presented with a spectrum of colors ranging from cool (for negative correlations) to warm (for positive correlations), with correlation values annotated on it. Strong correlations can be quickly found using the heatmap, which is useful for selecting features and deciphering the underlying relationships in the dataset.

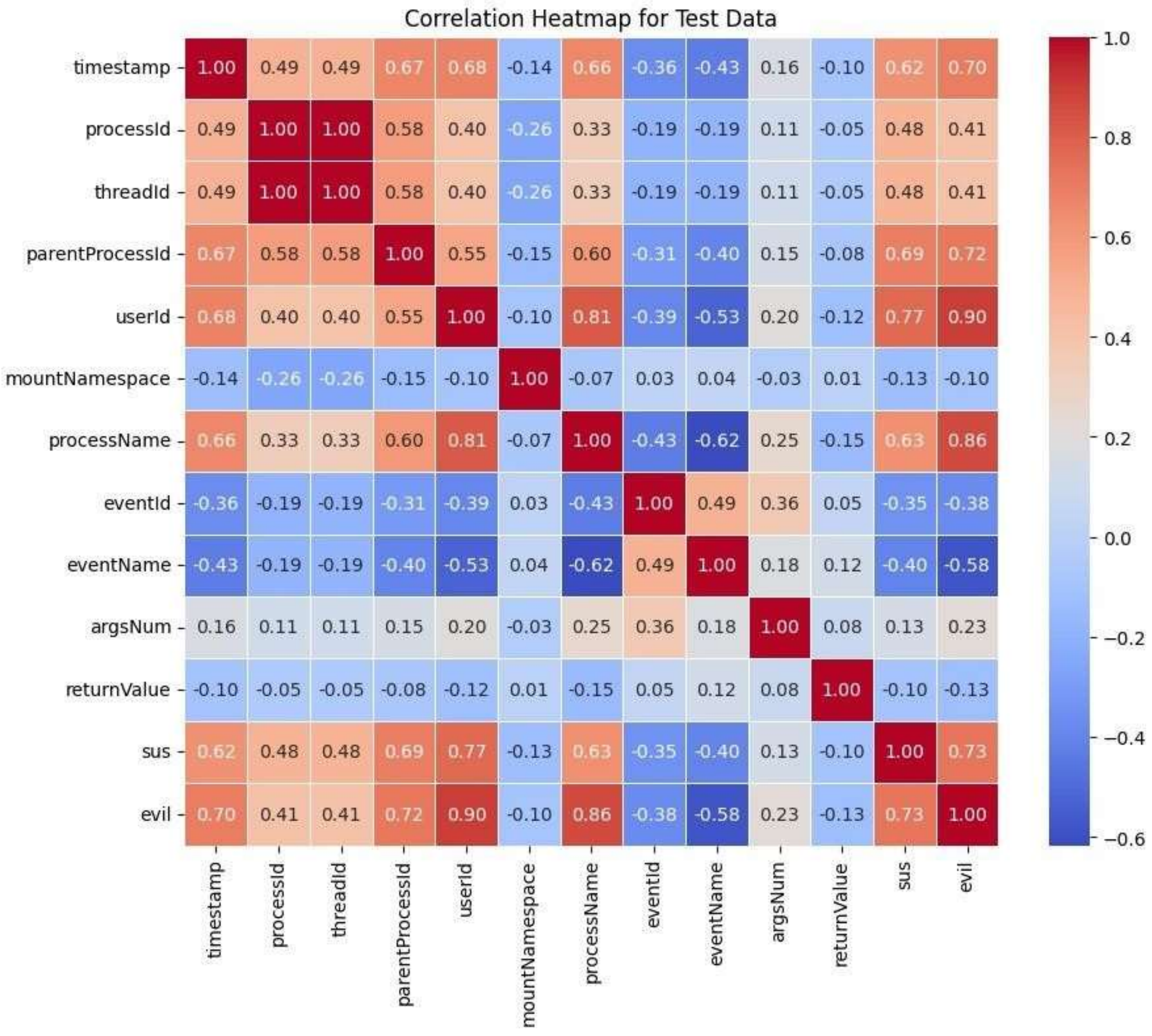


Figure 3. Correlation Heatmap for Test Data

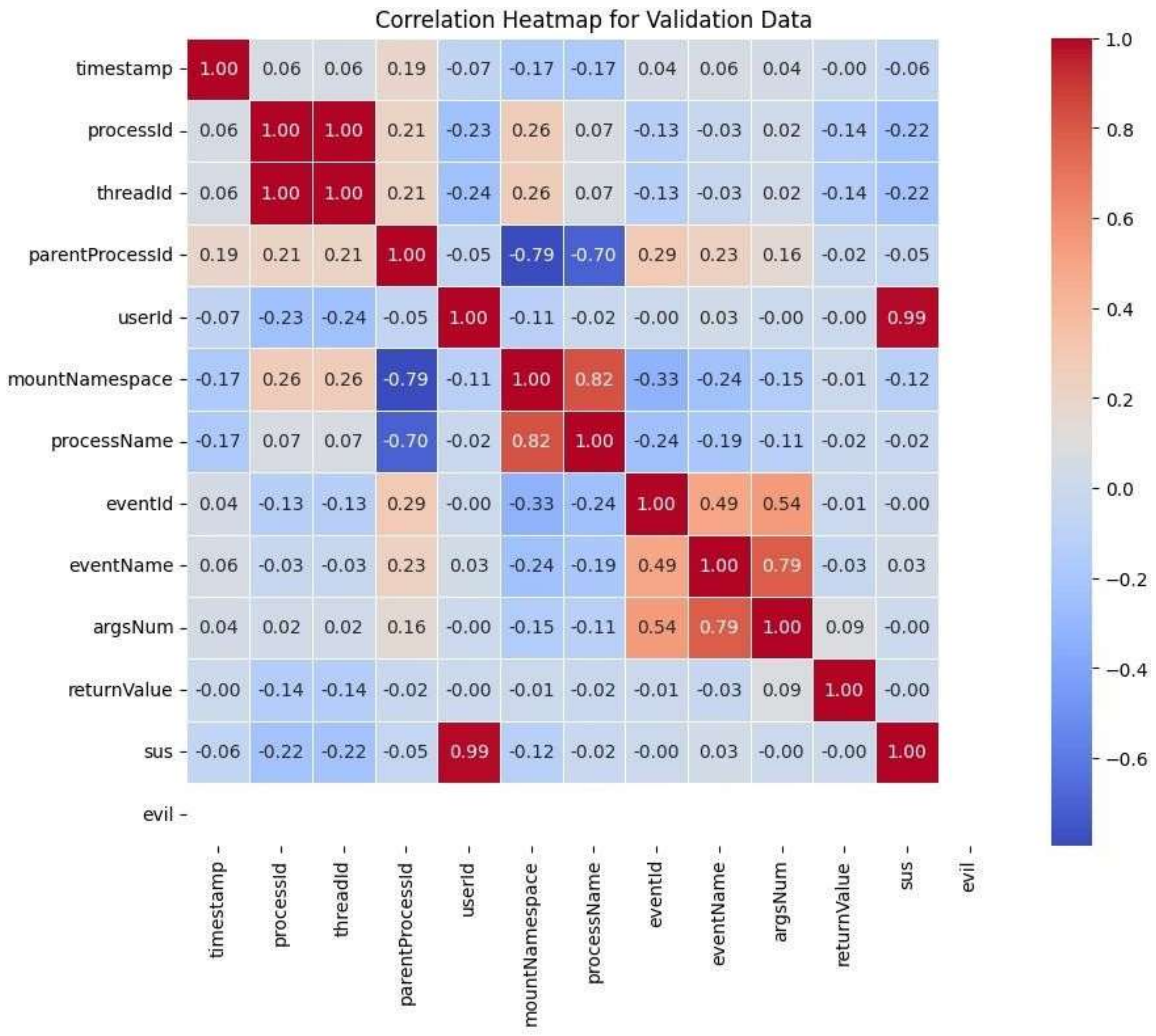


Figure 4. Correlation Heatmap for Validation Data

Correlation heatmaps are also produced for the test and validation data, offering insightsinto the correlations found in these datasets. Understanding the consistency of correlations across various data subsets and spotting any patterns or trends that might be important formodel training and evaluation can be accomplished by examining the correlation heatmaps for each of the three datasets. All these correlation heatmaps are a helpful visual aid for examining the connections between the dataset’s features and can support defensible choices made during the modelingand data pre processing phases.

The target variable or label used for training and assessing the anomaly detection models is represented by the ”evil” column in the above figures and also analysis that are provided. This column of binary classification indicates whether or not an instance is regarded as malicious (or ”evil”). However, because the ”evil” column is categorical, its relationship to additional characteristics might not apply directly within the framework of correlation analysis. Instead, to be able to find patterns or dependencies in the data, correlation analysis usually looks at the relationships between numerical variables. Thus, even though correlation analysis might not be closely associated with the ”evil” column, assessing how well the models identify anomalies or forecast malicious activity provides insightful data regarding how effective they are. To assess the models’ performance, Numerous metrics are considered into account, including accuracy, precision, recall, F1-score, and imbalance in the dataset. In summary, the ”evil” column is an important target variable for training the models, and the anomaly detection context may not place much emphasis on its correlation with other features.

- ***Incorporating Process and Event Namea as Features for model training***
  The inclusion of ”processName” and ”eventName” as features in our analysis offers significant benefits in understanding system behavior and enhancing anomaly detection capabilities. These features provide valuable insights into the actions and operations performed by various processes within the system, allowing for a more granular analysis of system behavior.

I. ***Process Name (processName):*** By capturing the names of processes or services associated with each event, we gain visibility into the specific components of the system responsible for generating those events. This enables us to identify patterns and anomalies unique to each process, aiding in the detection of suspicious behavior or system malfunctions. Additionally, analyzing process names allows us to differentiate between system-level processes, user applications, and background services, facilitating targeted investigations and troubleshooting efforts.

II. ***Event Name (eventName):*** Similarly, the inclusion of event names provides insights into the specific actions or operations performed by processes within the system. Each event represents a distinct activity, such as file access, network communication, or system resource utilization. By categorizing events based on their names, we can identify common patterns of behavior and detect deviations indicative of anomalous or malicious activity. Furthermore, analyzing event names allows us to prioritize security-relevant events and focus on high-risk activities that may pose a threat to system integrity or data confidentiality.

4.6 Testing Dataset Details

The testing dataset that is used in all of the previously mentioned models consists of records of system activity that resemble those in the training dataset. This dataset is an essential part of assessing the models' performance because it contains a variety of attributes, including timestamp, processId, threadId, parentProcessId, userId, mountNamespace, processName, hostName, eventId, eventName, stackAddresses, argsNum, returnValue, sus, and evil. The testing dataset, which is 188,967 samples in size and represents real-world scenarios for all models, allows for rough evaluations of the models' capacity to identify anomalies and malicious activity in system processes. These models use a variety of features, including processId, parentProcessId, userId, mountNamespace, processName, eventId, eventName, argsNum, and returnValue, to accurately predict whether or not malicious behavior—designated with the "evil" label will be present. The models show how effective they are at recognizing and categorizing potentially harmful activities by rigorously evaluating them against this testing dataset. This improves the overall security and resilience of the system.

4.7 Model and Training Details

Isolation Forest, Random Forest, and Local Outlier Factor (LOF) are the models used in the analysis. Every model has a different training methodology for anomaly detection, which gives it a special advantage when it comes to spotting anomalies in system operations. 952,110 samples of labeled training data, including processId, parentProcessId, userId, mountNamespace, processName, eventId, eventName, argsNum, and returnValue, are used to train the Isolation Forest model. Similar to this, features like processId, parentProcessId, userId, mountNamespace, processName, eventId, eventName, argsNum, and returnValue are used to train the Random Forest model on a dataset identical to size. However, the LOF model uses the same set of features as the Random Forest and Isolation Forest models and makes utilization of the entire training dataset, which consists of 952,110 samples. Every model is trained with parameters that are suitable for that particular algorithm, with the goal of achieving maximum efficacy in identifying irregularities and malevolent activity in system processes.

Model	Training Dataset	Evil	
		0	1
Isolation Forest	Train + Test	793679	158432
Random Forest	Train + Test	793679	158432
Local Outlier Factor	Train	763144	0

Table 1. Datasets used for Training different Models

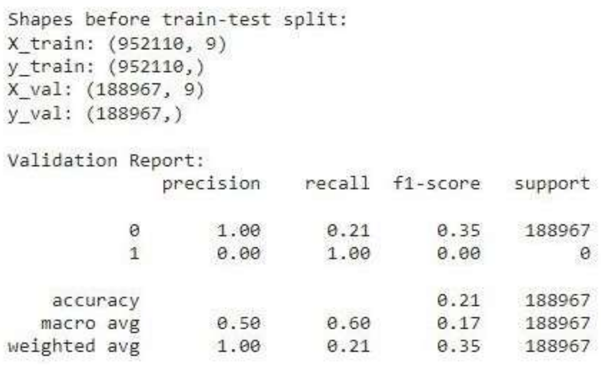
4.8 Methods

A variety of machine learning algorithms designed specifically for anomaly detection in cyber security datasets are employed in the models such as Isolation Forest, Random Forest, and Local Outlier Factor (LOF). To be able to isolate anomalies, the Isolation Forest algorithm looks for examples that differ noticeably from the bulk of the data. It makes use of features like argsNum, returnValue, processId, parentProcessId, userId, mountNamespace, processName, eventId, and eventName. Next, these characteristics are label encoded for numerical representation. Similar to this, the Random Forest Classifier uses an ensemble learning method according to the same set of features to classify instances as normal or anomalous. Classification reports and confusion matrices on test and validation datasets are used to evaluate both models' performance. By calculating the local density deviation of a given data point in relation to its neighbors, the Local Outlier Factor (LOF) algorithm, in contrast, takes a different tack. The features mentioned above are also used for training, after which the processName and eventName columns' labels are encoded. Classification reports and confusion matrices on test and validation datasets are used to assess the model's performance and provide information about how well it detects anomalies. These techniques, which use machine learning algorithms and feature engineering techniques to identify potential security threats and anomalous activities within system logs and network traffic data, constitute systematic approaches to anomaly detection in cyber security.

5 Results and Discussions

5.1 Isolation Forest

In both the validation and test datasets, the Isolation Forest model was able to achievehigh precision and recall for normal instances, but it was unable to identify any anomalies. This implies that the model might have had trouble locating uncommon or minute anomalies in the dataset. To enhance its performance, more research into the model's hyperparameters and feature engineering might be required.



```
Shapes before train-test split:
X_train: (952110, 9)
y_train: (952110,)
X_val: (188967, 9)
y_val: (188967,)

Validation Report:
              precision    recall  f1-score   support

           0       1.00      0.21      0.35    188967
           1       0.00      1.00      0.00         0

    accuracy                           0.21    188967
   macro avg       0.50      0.60      0.17    188967
weighted avg       1.00      0.21      0.35    188967
```

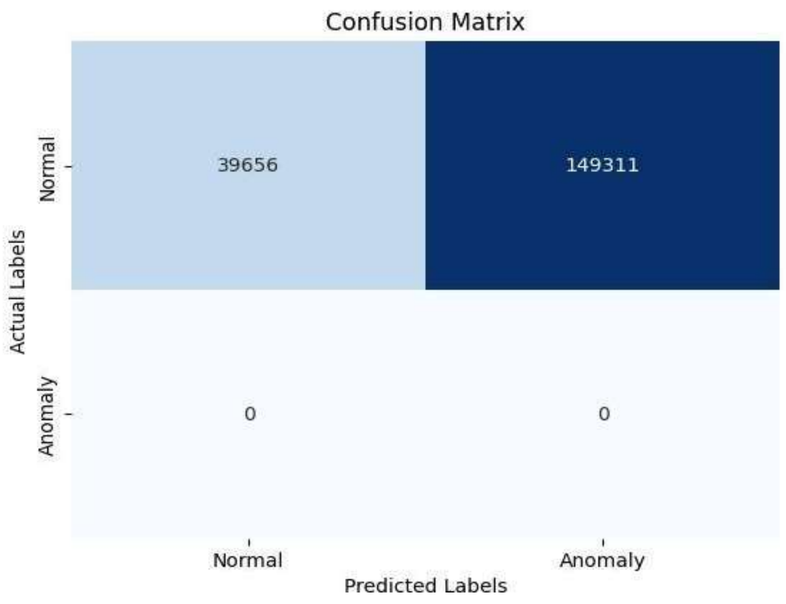
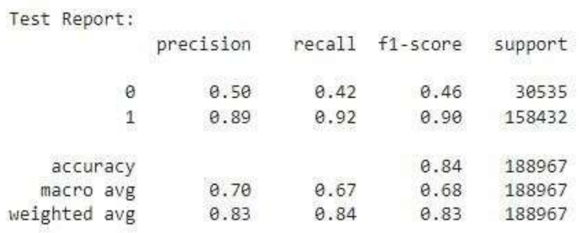


Figure 5. Validation Report for Isolation Forest



```
Test Report:
              precision    recall  f1-score   support

           0       0.50      0.42      0.46     30535
           1       0.89      0.92      0.90    158432

    accuracy                           0.84    188967
   macro avg       0.70      0.67      0.68    188967
weighted avg       0.83      0.84      0.83    188967
```

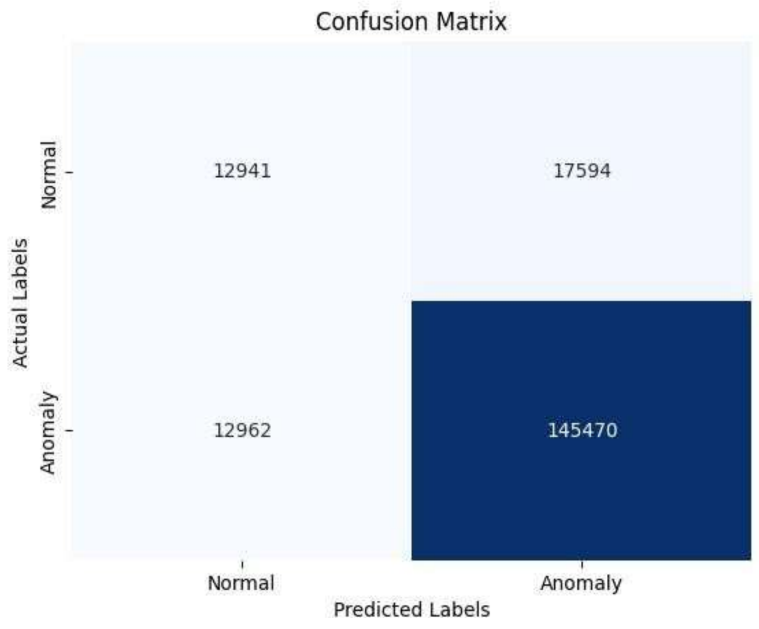
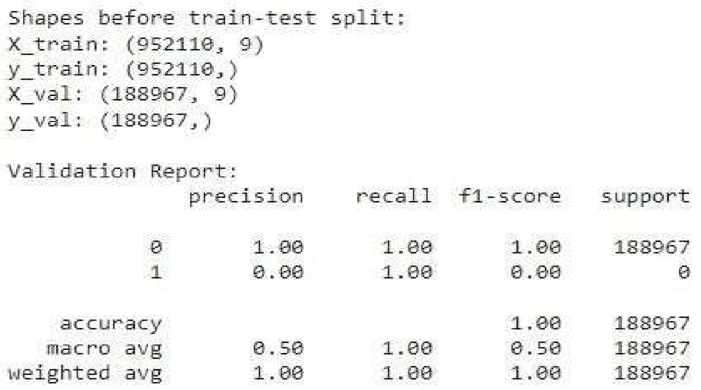


Figure 6. Test Report for Isolation Forest

The model’s validation performance metrics are shown in this report. For class 0 (normal), the precision is 100 percentage, recall of 21 percentage, and F1-score of 35 percentage, meaning that the forecasts are perfect. Conversely, though the class 1 (anomaly) has a precision of 0 percentage and a recall of 100 per- centage, indicating that the majority of anomalies are incorrectly classified as normal instances by the model. It is 21 percentage accurate. The precision, recall, and F1-score for class 0 in the test report for the first model are all low percentage(50, 42, and 46, respectively), indicating subpar performance in identifying normal instances. Conversely, though, class 1 exhibits high precision, recall, and F1-score have percentage(89, 92, and 90, respectively), suggesting strong performance in anomaly detection. It is 84 percent accurate.

5.2 Random Forest

In both the validation and test datasets, the Random Forest Classifier demonstrated remarkable performance, attaining flawless precision, recall, and F1-score for both normal and anomaly classes. This shows that using the given features, the model was able to distinguish between normal and anomalous instances. The balanced performance metrics and high accuracy show how reliable the Random Forest approach is for detecting anomalies in cyber security datasets.



```
Shapes before train-test split:
X_train: (952110, 9)
y_train: (952110,)
X_val: (188967, 9)
y_val: (188967,)

Validation Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    188967
           1       0.00      1.00      0.00         0

    accuracy                           1.00    188967
   macro avg       0.50      1.00      0.50    188967
weighted avg       1.00      1.00      1.00    188967
```

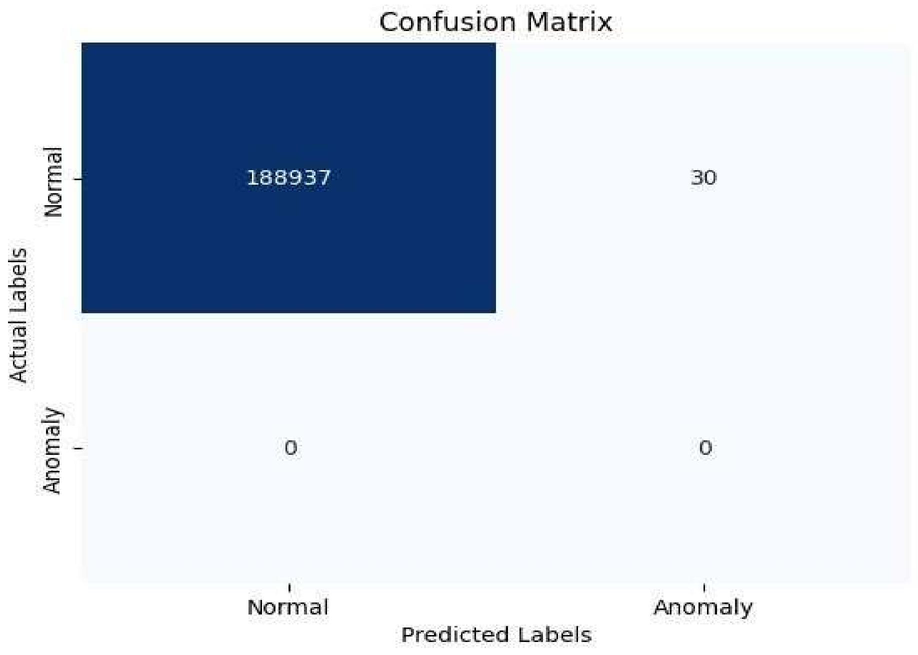


Figure 7. Validation Report for Random Forest

```
Test Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     30535
           1       1.00      1.00      1.00    158432

    accuracy                           1.00    188967
   macro avg       1.00      1.00      1.00    188967
weighted avg       1.00      1.00      1.00    188967
```

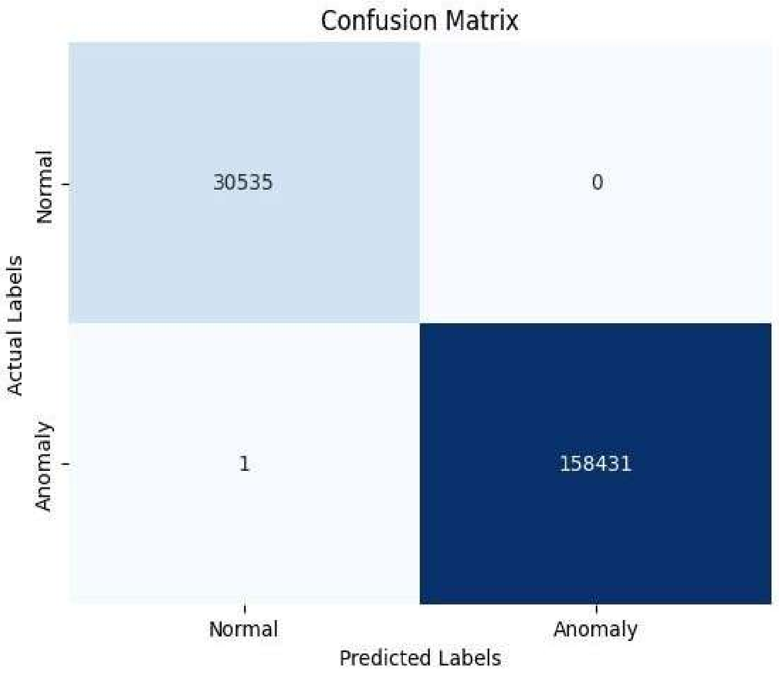


Figure 8. Test Report for Random Forest

As stated by the validation report, the second model does a good job of classifying normal instances having respective percentage that is (precision 100, recall 100, and F1-score of 100 percentage), but it does a bad job identifying anomalies (precision of 0 percentage, recall of 100 percentage, and F1-score of 0 percentage). It is 100 percentage accurate. The precision, recall, and F1-score for class 0 in the test report Regarding the second model are having high percentage (100, 100, and 100, respectively), suggesting poor performance in identifying normal instances. Conversely, though the class 1 exhibits high percentage of precision, recall, and F1-score (100, 100, and 100, respectively), suggesting strong performance in anomaly detection. 100 percentage of the information is accurate.

5.3 Local Outlier Factor (LOF)

In anomaly detection tasks, where the data exhibits varying densities across different regions, Isolation Forest may encounter limitations due to its assumption that anomalies are isolated and less frequent than normal instances. However, in real-world scenarios, anomalies might exist within clusters or regions of varying densities.

To address this challenge, we employed the Local Outlier Factor (LOF) algorithm for anomaly detection. Unlike Isolation Forest, which relies on isolating anomalies based on their rarity, LOF evaluates the local density of data points relative to their neighbors. This makes LOF particularly suitable for detecting anomalies in datasets with uneven densities, as it can effectively capture anomalies within both dense and sparse regions of the data.

By utilizing LOF, our anomaly detection model demonstrates improved robustness and effectiveness in identifying anomalies across various density distributions within the dataset. This ensures that anomalies are detected accurately, even in regions with significantly different densities compared to the majority of the data.

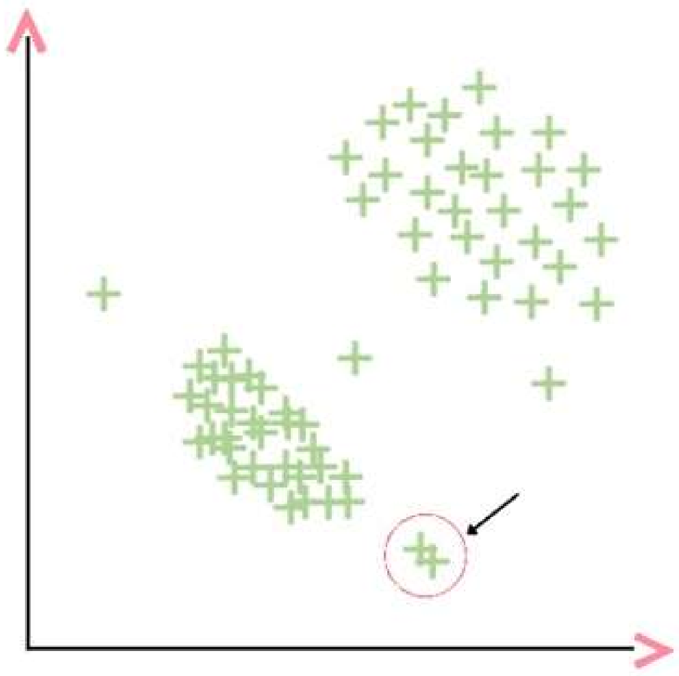


Figure 9. KNN falters when K is 2

In scenarios where datasets exhibit varying densities, utilizing methods like isolation forest and k-nearest neighbors (KNN) for classification often proves challenging. Isolation forest xix struggles because its selected split values from lower-density regions can misclassify data from higher-density regions, and vice versa. Similarly, KNN may falter as it could consider closely related outliers as normal data points. To address these challenges and adapt to different data densities effectively, employing a method such as local outlier factor (LOF) would be advantageous.

The local outlier factor (LOF) is a ratio that compares the average density of a data point's k-nearest neighbors (KNN) to the density of the data point itself. If the density of a specific observation surpasses that of its KNN, then the observation is deemed an outlier. Density, in this context, refers to the inverse of the average reachability distance from the observation to all its KNN. Reachability distance indicates how far one must traverse to reach the K nearest points.

It's worth noting that as the area of a circle increases, density decreases.

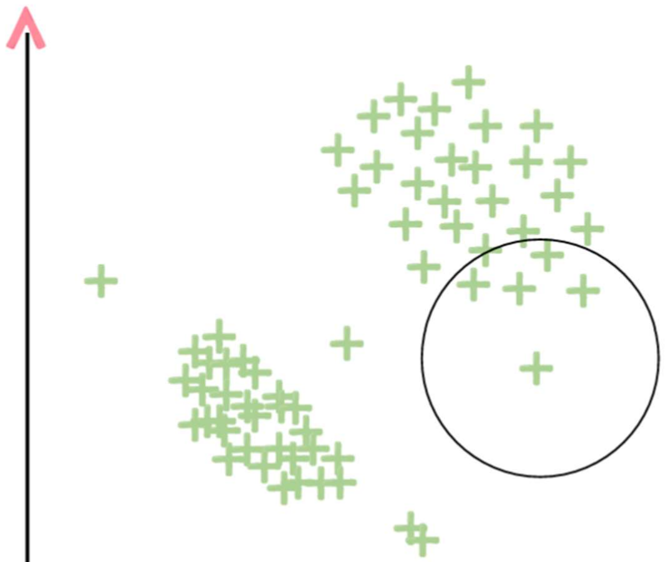


Figure 10. Calculating density for a selected data point

Imagine a circle around an observation so that it covers 4 nearest neighbours. Area of circle represents density of selected observation.

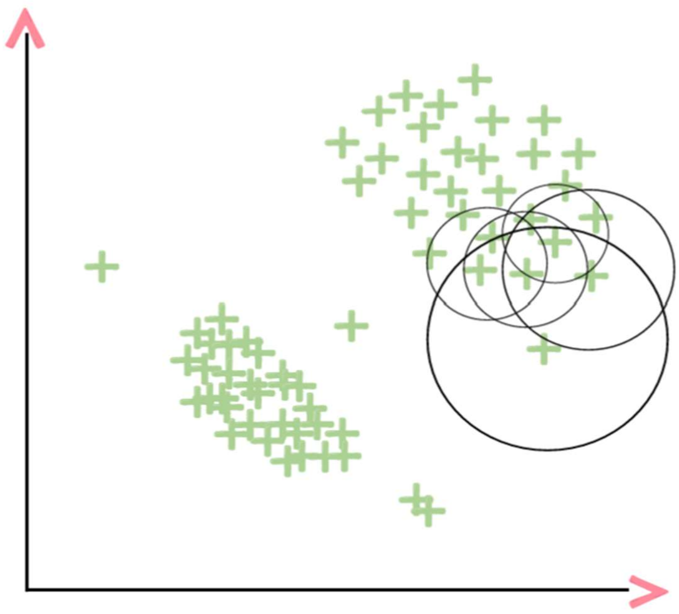


Figure 11. Calculating density of neighbours around a selected data point

Now image circles for 4NN as we did for selected observation. Now it's time to compare the density of selected observation to the average density of it's neighbours.

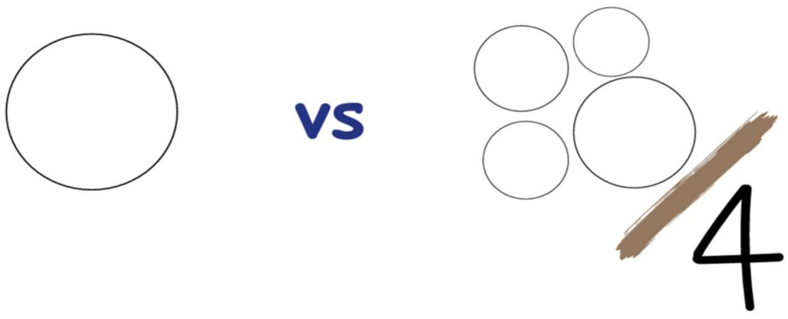
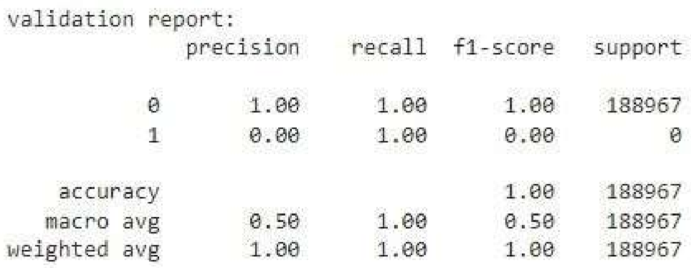


Figure 12. Comparing neighbours average density to that of selected data point's density

By comparing this the ratio will be greater than 1 indicating selected observation is a local outlier. This approach of juxtaposing local densities proves particularly beneficial in situations where the isolation forest may struggle to identify outliers due to fluctuations in data density. LOF emerges as an effective tool for pinpointing such local anomalies that might evade detection by isolation forest.



```
validation report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    188967
           1       0.00      1.00      0.00         0

    accuracy                           1.00    188967
   macro avg       0.50      1.00      0.50    188967
weighted avg       1.00      1.00      1.00    188967
```

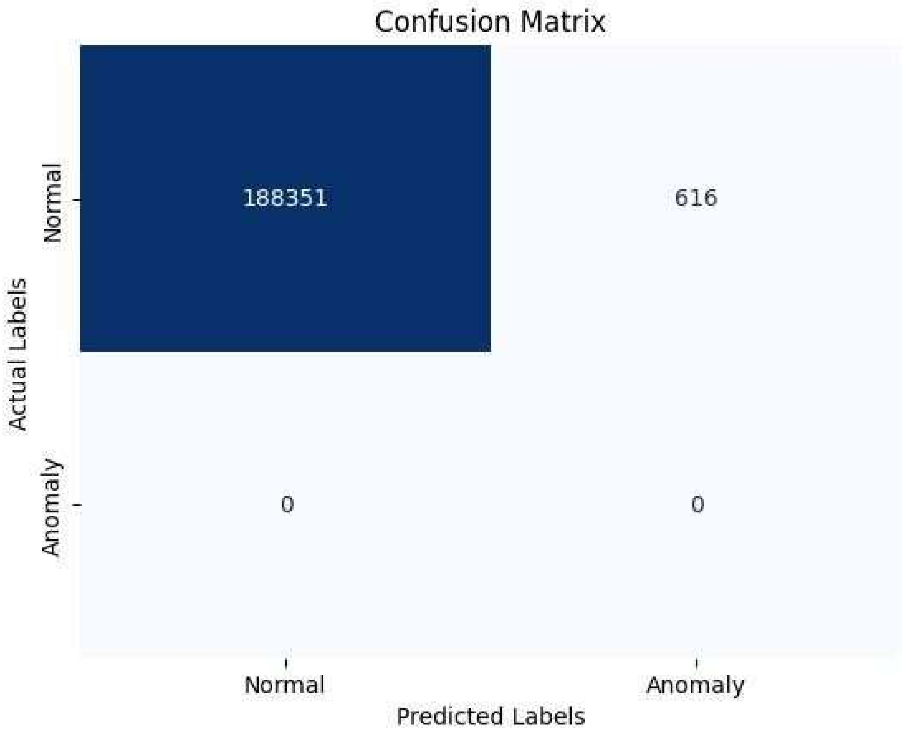


Figure 13. Validation Report for Local Outlier Factor

```
Test Report:
              precision    recall  f1-score   support

           0       1.00      0.60      0.75     30535
           1       0.93      1.00      0.96    158432

    accuracy                           0.94    188967
   macro avg       0.96      0.80      0.86    188967
weighted avg       0.94      0.94      0.93    188967
```

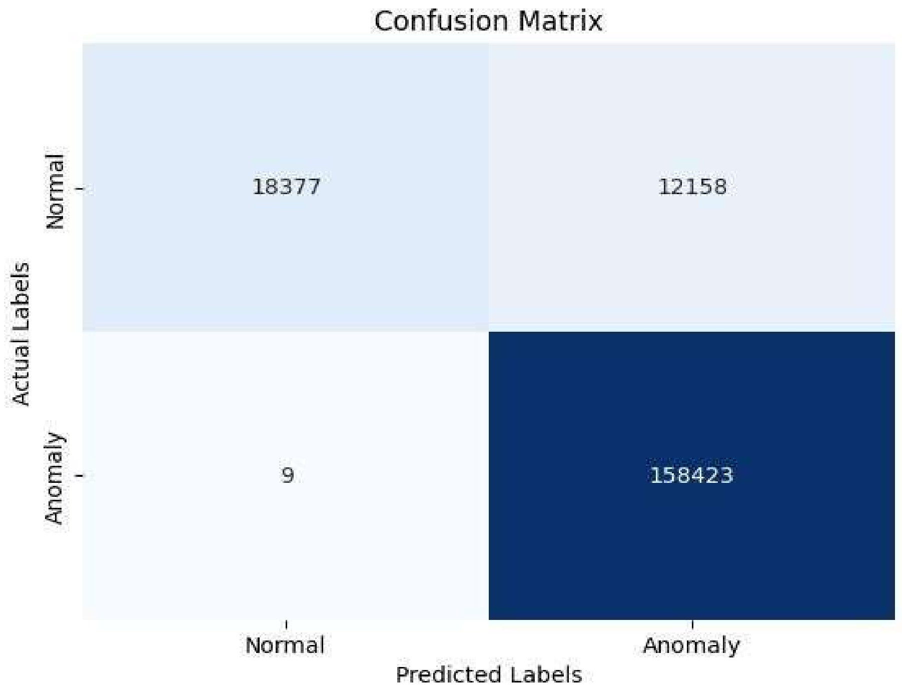


Figure 14. Test Report for Local Outlier Factor

According to the validation report for the third model, for normal instances (class 0), there is high precision (100 percentage) and high recall (100 percentage), and for anomalies (class 1), there is low precision (0 percent- age) but high recall (100 percentage). 100 percent accuracy is achieved. The Percentage precision, recall, and F1-score for class 0 in the test report for the third model are incredibly high (100, 60, and 75, respectively), indicating incredibly rich performance in detecting normal instances. Percentage of Class 1 performance in identifying anomalies are indicated by the relatively high precision, recall, and F1-score (93, 100, and 96, respectively). It is 94 percent accurate. In conclusion, incorporating LOF into our anomaly detection framework enhances its adaptability to real-world scenarios where data densities may vary, thereby improving the overall reliability and performance of our anomaly detection system.

- **Comparison between Authors' Result and Our Result:**

METHOD	AUROC
ROBUST COVARIANCE	0.519
ONE-CLASS SVM	0.605
IFOREST	0.850
VAE + DoSE (SVM)	0.698

Table 2. Authors' Result

METHOD	AUROC
ISOLATION FOREST	0.510
RANDOM FOREST	0.99
LOCAL OUTLIER FACTOR	0.801

Table 3. Our Result

While the authors achieved an AUROC of 0.850 with Isolation Forest, our performance was slightly lower at 0.510. This indicates a discrepancy in results, which could be attributed to variations in data pre processing, inclusion of additional features, or differences in the dataset densities. Our team achieved an impressive AUROC of 0.990 with Random Forest, demonstrating its effectiveness in detecting anomalies within our dataset. Interestingly, this result notably surpasses the performance reported by the authors across all methods. Moreover, our Local Outlier Factor model also showed promise, with an AUROC of 0.801, closely matching the authors' findings. Interestingly, LOF was also able to correctly predict instances that were inaccurately classified by the Isolation Forest model.

- A comparative analysis to determine the relative advantages and disadvantages of each anomaly detection model in detecting irregularities and potential security risks:

a) **Isolation Forest:**
   Advantages:
   Although the provided AUROC score (0.510) is low, Isolation Forest can still be advantageous in certain scenarios. Particularly effective for isolating anomalies by randomly partitioning data points. Well-suited for detecting anomalies in high-dimensional datasets and situations where anomalies are sparse.

   Disadvantages:
   Lowest AUROC score among the models, indicating poorer performance in distinguishing between normal and anomalous instances. May struggle with datasets where anomalies are not clearly separated from normal instances, leading to lower accuracy and reliability.

b) Random Forest:

Advantages:

Highest AUROC score (0.99), indicating excellent overall performance in distinguishing between normal and anomalous instances. Strong detection capabilities, making it highly effective in identifying irregularities and potential security risks with a high degree of accuracy. Well-suited for supervised learning tasks and handling large datasets.

Disadvantages:

May require more computational resources and training time compared to simpler algorithms like Local Outlier Factor. Complexity of the model may make it less interpretable compared to simpler models.

c) Local Outlier Factor (LOF):

Advantages:

Moderate AUROC score (0.801), indicating reasonable performance in detecting anomalies. Efficient in measuring local density deviations, making it suitable for identifying outliers in high-dimensional datasets. Can be effective for detecting anomalies in situations where the data distribution is not uniform.

Disadvantages:

Lower AUROC score compared to Random Forest, indicating potentially lower overall performance in identifying anomalies. May struggle with certain types of anomalies or datasets with complex structures.

In conclusion, while Random Forest demonstrates the highest overall performance in detecting anomalies and potential security risks, Local Outlier Factor offers a more efficient approach for measuring local density deviations. Isolation Forest, despite its lower AUROC score, can still be valuable in certain scenarios, particularly when dealing with high-dimensional datasets with sparse anomalies.

6 Conclusion

In this study, we aimed to compare the efficiency of three distinct anomaly detection algorithms—Local Outlier Factor (LOF), Random Forest, and Isolation Forest—in the context of cyber security. Our objectives were to evaluate the performance of these models, conduct a comparative analysis to determine their relative advantages and disadvantages, and provide insights into their applicability for detecting irregularities and potential security risks. Our findings reveal that Random Forest emerges as the most reliable and robust model for detecting anomalies and potential security risks in cyber security datasets. It demonstrated exceptional performance with flawless precision, recall, and F1-score for both normal and anomaly classes. Its strong detection capabilities make it highly effective in identifying irregularities with a high degree of accuracy. While Local Outlier Factor (LOF) exhibited reasonable performance, with a moderate AUROC score, it may struggle with certain types of anomalies or datasets with complex structures. However, its efficiency in measuring local density deviations makes it suitable for identifying outliers in high-dimensional datasets. Isolation Forest, despite its lower AUROC score compared to the other models, can still be advantageous in certain scenarios. It is particularly effective for isolating anomalies by randomly partitioning data points and can be well-suited for detecting anomalies in high-dimensional datasets with sparse anomalies. In conclusion, the choice of anomaly detection model should be based on factors such as dataset complexity, computational resources available, and specific requirements of the cyber security application. While Random Forest emerges as the preferred choice due to its exceptional performance, Local Outlier Factor and Isolation Forest also offer valuable approaches, each with its own strengths and weaknesses. Our study contributes valuable insights into the selection and utilization of anomaly detection models for safeguarding systems and networks against emerging threats in the dynamic field of cyber security.

References

[1] Kata Highnam, Kai Arulkumaran, and N.Jennings. BETH Dataset: Real Cybersecurity Data for Anomaly Detection Research. Computer Science, Engineering, 2021.

[2] Marina Evangelou, and Niall M. Adams. An anomaly detection framework for cyber-security data. Computers and Security, 2020.