Implementing a machine learning based anomaly detection model using a 5g dataset

Rita Emenike

April 2023

1 Introduction

Anomalies are data patterns that have different data characteristics from normal instances [1]. They are also known as outliers and may represent unexpected events, errors or fraudulent activities. Anomaly detection techniques typically involve statistical methods or machine learning algorithms that can learn patterns in data and identify observations that deviate significantly from those patterns. There are several techniques used for anomaly detection, including statistical methods, clustering, neural networks, and time-series analysis. The choice of technique depends on the nature of the data and the problem at hand. Anomaly detection finds extensive use in a wide variety of applications such as fraud detection for credit cards, insurance or health care, intrusion detection for cyber-security, fault detection in safety critical systems, and military surveillance for enemy activities. [2].

Anomaly detection algorithms can be categorized into supervised or unsupervised depending on whether labels are used in the training phase [4]. To train the model, supervised anomaly detection algorithms require labeled data. The labeled data assists the algorithm in learning what the system's normal behavior looks like and identifying data points that deviate from this normal behavior, The algorithm is trained on a set of input-output pairs, where the input is the data point and the output is the label indicating whether it is normal or abnormal. Random forests, Decision trees, support vector machines (SVM), and neural networks are examples of supervised learning algorithms used for anomaly detection. However, unsupervised anomaly detection algorithms do not require labeled data. In this approach, the algorithm learns the system's normal behavior without prior knowledge of anomalies. The algorithm detects anomalies by detecting data points that deviate significantly from the system's normal behavior. Examples of unsupervised learning algorithms used for anomaly detection include clustering-based methods, density-based methods, and principal component analysis (PCA).

Often the dataset to be used contains noise which tends to be similar to the actual anomalies and thus difficult to distinguish and remove [4]. Hence, Random forest algorithm is used to detect anomaly in this paper because it is

an ensemble learning algorithm that is less sensitive to noise and outliers. It combines multiple decision trees to make predictions which allows it to capture complex relationships between features and identify subtle anomalies that might be missed by other algorithms [3].

Random forests assist by averaging multiple deep decision trees trained on the same training set but on different parts, with the goal of reducing variance and preventing training set overfitting. The process flow is as follows: Each tree is traversed by a new input fed into the system. Random forests have the disadvantage of being unable to predict beyond the range of the training data when used for regression [5].

The random forest is not reliant on decision trees. Instead, it receives a prediction from each tree and forecasts the final result based on polls of prevalence estimations. The greater the number of trees in the forest, the greater the accuracy and avoidance of overfitting issues. It has supported the concept of the ensemble technique, which combines multiple classifiers to solve a difficult problem and improve model performance [6].

The following are the main characteristics of the random forest algorithm:

- It outperforms all current data mining algorithms in terms of accuracy.
- It performs well on large data sets with a large number of features.
- It can provide an estimate of which features are important.
- It has no nominal data issue and does not over-fit.
- It can deal with unbalanced data sets. [7]

The dataset used was obtained from [14]. A recent survey summarizes a comprehensive summary of such datasets that are effective for evaluating intrusion detection on networks until 2020. The majority of these datasets are out of date for modern networks because they were created prior to some significant technological advancements. The primary goal of this paper is to bridge that gap by creating a novel dataset from a real-world 5G network implementation and publishing it for future research. The approach for benign traffic generation is a key feature that distinguishes 5G-NIDD from other datasets. Instead of simulated traffic, actual mobile devices in the network generate benign traffic. Adding prior simulated benign traffic has been the common approach in most of the other datasets, which affects the traffic behavior in the dataset, Live traffic from mobile devices in the network, which includes both the attack traffic and the benign traffic was captured. Attack types considered were Denial o service attack: all 3 categories (volume-based, protocol-based and application layer attacks) were considered., port scan attack: the dataset contains attack data from different port scans such as the SYN scan, TCP Connect scan, and UDP scan. The data collection process was carried out on two separate days, and both malicious and benign traffic passing through the network at the two base stations was captured; raw data post-processing was required before feeding into the machine learning models. To evaluate and compare performance, the most

common performance evaluation matrices such as confusion matrix, accuracy, precision, and F1-score are used. 5G-NIDD includes a full capture of network traffic from a functional 5G network in the presence of various attack scenarios such as Port Scans and DoS attacks. In addition to the malicious traffic, the dataset includes nonmalicious traffic from real users, making it more realistic.

2 Related Work

The authors of [8] provided an overview of the most recent applications of machine learning techniques for data anomaly detection in wireless sensor networks (WSNs) domains. They discussed the characteristics of WSNs as well as the challenges of WSN anomaly detection, as well as various ML techniques that have been applied to WSN data anomaly detection. The survey showed the importance and effectiveness of using machine learning to look for anomalies in WSNs. They came to the conclusion that more research is still needed in this area, particularly to develop algorithms that are more efficient and effective for use on large and complex networks.

The authors of [9] discussed the anomalies that could occur in an IoT system as well as the specific challenges that complicate the process of anomaly detection. The potential applications of anomaly detection in a variety of IoT settings are also discussed, as are the approaches that have been historically used as well as those that are currently being developed.

Rashmi H and Prof. N. V. Buradkar proposed a system to improve anomaly detection accuracy over existing systems. They mentioned that most machine learning algorithms lack inherent superiority techniques. The Random Forest machine learning algorithm is used in the paper due to its accuracy and ability to provide proximity functionality. The proposed system tries to reduce the false positive rate of the Anomaly detection by refining the tuning of parameters in Random Forest algorithm. [10].

In [11], the authors proposed SwitchTree, which embeds the Random Forest algorithm inside a programmable switch so that Random Forest is configurable and re-configurable at runtime. SwitchTree was able to accurately identify network attacks at line speed. Additionally, they demonstrated how certain flow level stateful features, like the bitrate and round trip time of each flow, can be estimated.

In [12], the authors reviewed literature on anomaly detection in terms of fundamental approaches, mechanisms, datasets employed, performance metrics, and potential future research using convolution neural networks. When the input has multiple dimensions, such as images, which is difficult for a traditional neural network to handle, deep learning is implemented using convolutional neural networks. A framework called a unified cross framework, which emphasizes the methods that operate in the background of each solution to provide anomaly detection mechanisms, was proposed to support the survey analysis.

The authors of [1] proposed Isolation Forest, or iForest, a method that explicitly isolates anomalies rather than profiling normal points. It exploits sub-

sampling to an extent not possible in existing methods, resulting in an algorithm with a linear time complexity, a low constant, and a low memory requirement. To accomplish this, the authors' proposed method takes advantage of two quantitative properties of anomalies:

- they are a minority with fewer instances
- they have attribute-values that are very different from those of normal instances.

iForest has a linear time complexity, a low constant, and a small memory footprint. The best-performing existing method, to the best of our knowledge, achieves only approximate linear time complexity with high memory usage. The detection of anomalies with iForest is a two-step process. The first (training) stage involves the construction of isolation trees from subsamples of the training set. The second (testing) stage runs the test instances through isolation trees in order to generate an anomaly score for each one. [1]

The iForest was evaluated using four sets of experiments. The first experiment compared iForest to ORCA, LOF, and Random Forests; the second experiment investigated the effect of different sub-sampling sizes; the third experiment extended iForest to handle high-dimensional data; and the fourth experiment investigated iForest's performance when only normal instances are available for training. At the conclusion of the experiments, it was discovered that iForest's quick execution with low memory requirements is a direct result of building partial models and requiring only a significantly smaller amount of memory. This capability is unparallel in the domain of anomaly detection.[1]

3 Implementation

These Machine learning algorithm model implemented in this paper is shown in figure 1. The random forest algorithm works in 7 steps. Application used to build the model is Pycharm 2022 Community Edition because it comes with built-in support for data analysis libraries like NumPy, Pandas, and Matplotlib which can be used to manipulate and visualize data, which is important when preparing data for modeling.

Figure 1: Random forest model flowchart

1. Data Collection: The dataset used is the combined dataset gotten from the paper 5G-NIDD: A Comprehensive Network Intrusion Detection Dataset Generated over 5G Wireless Network. Figure 2 shows how to insert the dataset into the model.

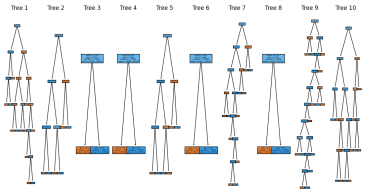

Figure 2: Dataset collection

2. Data Pre-processing: This step involves processing the raw data and making it suitable for the random forest model by handling missing values, encoding categorical variables; and also identifying the most important features that are relevant for detecting anomalies. Feature selection can be done before or after splitting the data. Columns with lots of null values were replaced with their mean values, also columns with categorical values were replaced with integer value using the find and replace method in excel. Figure 3 shows how data is preprocessed in the model.

```
#Step 2: Data Preprocessing

columns_null = []
columns = data.columns
for column in columns:
    c = data[column].isnull().sum()
    if c != 0:
        print(column, 'has {} null values'.format(c))
        columns_null.append(column)

data['sTtl'] = data['sTtl'].fillna(data['sTtl'].mean())
data['dTtl'] = data['dTtl'].fillna(data['dTtl'].mean())
data['dHops'] = data['dHops'].fillna(data['dHops'].mean())
data['SrcWin'] = data['SrcWin'].fillna(data['SrcWin'].mean())
data['DstWin'] = data['DstWin'].fillna(data['DstWin'].mean())
data['sVid'] = data['sVid'].fillna(data['sVid'].mean())
data['dVid'] = data['dVid'].fillna(data['dVid'].mean())
data['SrcTCPBase'] = data['SrcTCPBase'].fillna(data['SrcTCPBase'].mean())
data['DstTCPBase'] = data['DstTCPBase'].fillna(data['DstTCPBase'].mean())

#convert non-numeric data to numeric, this can also be done manually by using the find and replace method in excel
data.Proto[data.Proto == 'icmp'] = 0
data.Proto[data.Proto == 'udp'] = 1
data.Proto[data.Proto == 'tcp'] = 2
```

Figure 3: Data preprocessing

3. Training and testing dataset split and Feature Selection: The training set will be used to build the random forest model, while the testing set will be used to evaluate the performance of the model. In this step, 60% of the dataset was used for the training model while the remaining 40% was used for the testing model. The random state parameter ensures that the same split is generated each time the code is run. The Label column was selected as he primary feature because the paper to predict if it is benign or malicious to be able to detect an anomaly. Figure 4 shows how feature selection was carried out in the model, while figure 5 shows how the dataset was split into the training and testing data.

Figure 4: Feature selection

```
# Step 3: feature selection
#define dependent variable
Y = data['Label'].values
Y = Y.astype('int')
#Step3.5: Training and testing dataset split
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.4, random_state=42)

print(X_train)
print(X_test)
print(Y_train)
print(Y_test)
```

Figure 5: Training and testing dataset split

4. Build the Random Forest Model: The process involves creating a random forest model by building multiple decision trees on random subsets of the training data. The hyper-parameters like the number of trees and the depth of each tree are set based on the dataset and problem.

```
#Step4: Build the random forest model
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
print("====")
model.fit(X_train, Y_train)
```

Figure 6: Random forest model

5. Evaluate the Model: The model was evaluated based on the performance of the random forest model on the testing set using the accuracy and confusion matrix metrics. The accuracy metric measures the proportion of correctly classified observations out of the total number of observations in the dataset. The accuracy can be calculated using the following formula:

   accuracy = (number of correctly classified observations) / (total number of observations)

   Figure 7 shows how the model was evaluated.

```
#step 5: evaluate result
from sklearn.metrics import accuracy_score
ypred = model.predict(X_test)
print(ypred,Y_train)
acc_score = accuracy_score(Y_test, ypred)
print(acc_score)
```

Figure 7: Model Evaluation

6. Detect Anomalies: The trained random forest model was used to detect anomalies in the 5G dataset. Anomalies can be detected based on the

predicted class labels or the predicted values of the target variable. Figure 8 shows how anomaly detection was carried out.

```
#step 5: evaluate result
from sklearn.metrics import accuracy_score
ypred = model.predict(X_test)
print(ypred,Y_train)
acc_score = accuracy_score(Y_test, ypred)
print(acc_score)
```

Figure 8: Anomaly Detection

7. Interpret Results: The result of a random forest model is typically a set of decision trees, each of which represents a possible path through the features that leads to a predicted outcome. These trees can be visualized as a diagram with nodes representing decision points and edges representing possible outcomes. The best possible value of an accuracy score is 1 (if a model got all the predictions right), and the worst is 0 (if a model did not make a single correct prediction. The accuracy score of our random model is 1.0. A scatter plot of the dataset with the predicted anomaly labels as the color of the points to enable us to interpret the result better. Anomalies ae depicted with the pink color.

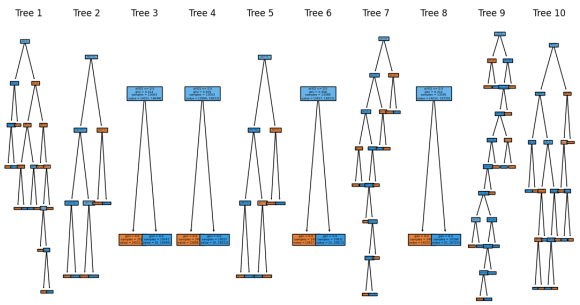

Figure 9: Random Forest Model

```
Accuracy Score
1.0
```

Figure 10: Accuracy Score

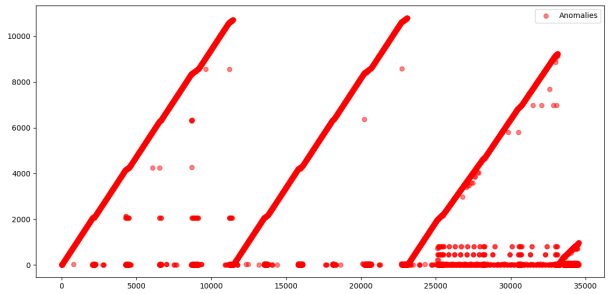

Figure 11: Anomaly scatter p

4 Conclusion

This paper shows how the Random forest model can be used for anomaly detection. It illustrates the steps necessary to create a Random Forest model, reviewed existing literatures and also interprets the result.

5 References

1. Liu, F. T., Ting, K. M., Zhou, Z. H. (2012). Isolation forest. In Proceedings of the 2012 12th IEEE International Conference on Data Mining (pp. 413-422). IEEE.

2. A Survey of Anomaly Detection in Text Data by Chandola et al. (2009). Department of Computer Science and Engineering University of Minnesota.

3. Breiman, Leo. "Random forests." Machine learning 45.1 (2001). Springer.

4. Anomaly Detection in Time Series Sensor Data using Variational Autoencoders by S. Sharma, S. K. Jena, and S. K. Rath.

5. Girik Pachauria and Sandeep Sharma. (2015). Anomaly detection in medical wireless sensor networks using machine learning algorithms.

6. Zhang, Jiong Zulkernine, Mohammad Haque, A.. (2008). Random-Forests-Based Network Intrusion Detection Systems. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on. 38. 649 - 659.10.1109/TSMCC.2008.923876.

7. Prashanth, G. Prashanth, V. Padmanabhan, Jayashree Srinivasan, N.. (2008). Using Random Forests for Network-based Anomaly detection at Active routers. 93 - 96. 10.1109/ICSCN.2008.4447167.

8. Ahsnaul Haque et al. Wireless Sensor Networks anomaly detection using Machine Learning: A Survey.

9. Andrew Cook, Goksel Mısırlı, and Zhong Fan. (2019). Anomaly Detection for IoT Time-Series Data: A Survey.

10. Rashmi H Roplekar1 and Prof. N. V. Buradkar. (2017). Survey of Random Forest Based Network Anomaly Detection Systems.

11. Jong-Hyouk Lee and Kamal Singh. (2020). SwitchTree: In-network Computing and Traffic Analyses with Random Forests.

12. Montdher Alabadi and Yuksel Celik. (2020). Anomaly Detection for Cyber-Security Based onConvolution Neural Network : A survey

13. Sehan Samarakoon et al. (2022). 5G-NIDD: A Comprehensive Network Intrusion Detection Dataset Generated over 5G Wireless Network.