Project 5: Anomaly Detection

Miftahul Huq

CyberAnalytics and Machine Learning

Bo Yuan

12/07/2024

Table of Content:

Introduction

In the last few years, the detection of anomalies has become essential in identifying malicious activities in cybersecurity systems. Anomaly data points are usually rare, unusual, and often indicate malicious intrusion, fraud, or failure within a system. Classic supervised learning fails to handle anomaly detection problems due to the complete absence of labeled data samples, and therefore, unsupervised methods such as the Isolation Forest and Robust Covariance have been more capable of solving these problems.

The BETH dataset serves as a benchmark for the performance evaluation of different anomaly detection methods in cybersecurity. The paper proposed for this dataset presents an evaluation of a number of anomaly detection algorithms, including the Isolation Forest, Robust Covariance, One-Class SVM, and Variational Autoencoders (VAE). Among those, the Isolation Forest showed the best result, with an AUROC of 0.85, whereas Robust Covariance reached just 0.62 for a simple baseline.

This project is concerned with the implementation and performance evaluation of the Isolation Forest and Robust Covariance on the BETH dataset. The aim is to try and hopefully outperform the results presented in this research paper by following proper data preprocessing, hyperparameter tuning, and model evaluation. The project focuses on:

1. Understanding the strengths and weaknesses of the selected algorithms.
2. Comparing the obtained AUROC scores with the research paper's results.
3. Highlighting any improvements achieved through feature engineering and parameter optimization.

The implementation uses modern libraries, scikit-learn, and Python for reproducibility and ease of deployment. By the end of the project, we want to demonstrate the practical effectiveness of those algorithms in anomaly detection and gain insight into their behavior when real-world cybersecurity data is processed.

Background

It forms a core role in cybersecurity through the detection of rare or suspicious behaviors that fall outside the normal pattern. In most cases, these anomalies signify malicious activities, system intrusions, or failures that may go undetected by traditional security tools. Because of this imbalance between normal and anomalous data points,

anomaly detection has to be performed using specialized algorithms that can handle unsupervised or semi-supervised learning tasks.

Anomalies within cybersecurity may relate to processes that behave differently than expected, unusual network activity, or low probabilities of access. Detecting these anomalies with speed is key in limiting security breaches, loss of data, and assurance of system integrity.

BETH Dataset

The dataset that was used in this project, BEST dataset, is designed for anomaly detection. It includes:

- Numerical and categorical features representing system events, such as process IDs, user IDs, namespace, and return value.
- There is a binary indicator, sus, identifying whether an event suspicious marked as 1 or normal marked as 0.

However, there are challenges in the dataset, such as:

- The majority of the dataset contains normal behavior, with only a small percentage of anomalies.
- To identify complex patterns, multiple features had to be analyzed simultaneously.
- In the real world, the anomalies may not follow linear or obvious patterns.

Robust Covariance and Isolation Forest

A number of anomaly detection techniques are evaluated in the study article. Because of their unique methodologies, Robust Covariance and Isolation Forest were selected for this project:

1. Robust Covariance (Elliptic Envelope):
    - Assumes that normal data follows a Gaussian distribution.
    - Estimates the covariance matrix using the Minimum Covariance Determinant (MCD) method, which is robust to outliers.
    - Outliers are detected based on their Mahalanobis distance from the estimated Gaussian distribution center.
    - This algorithm is computationally intensive for large datasets but interpretable for small, well-behaved data.
2. Isolation Forest (iForest):

- A tree-based ensemble algorithm that isolates anomalies by recursively partitioning that data.
- Anomalies are identified as points that require fewer splits to isolate compared to normal points.
- It is computationally efficient, scaled well to high-dimensional datasets, and performed well in practice

Research Paper Findings

The followings are the AUROC scores for different anomaly detection algorithms:

- Isolation Forest: **0.85** (best performance)
- Robust Covariance: **0.62** (baseline performance)
- Variational Autoencoder (VAE): **0.698**
- One-Class SVM: **0.58**

The goal of this project is to replicate and validate the results for Isolation Forest and Robust Covariance using the BETH dataset. Additionally, the project tries to find out if preprocessing, hyperparameter tuning, or any improvement in the implementation can boost the performance of these algorithms.

Data Preparation

Dataset Overview

Along with the original dataset files, the dataset also consists of labeled training and testing datasets. Each record corresponds to system events and includes features relevant to identifying anomalies. Anomalies are labeled with the “sus” column:

- sus = 0: Normal data points.
- sus = 1: Anomalous data points

The datasets are as follows:

- Training Datasets: Labeled data used for model training.
- Testing Dataset: Labeled data used for evaluating model performance.

The key features utilized for anomaly detection are:

1. processId: Process ID associated with the system event.
2. parentProcessId: Parent process ID.
3. userId: User identifier
4. mountNamespace: Namespace associated with the process
5. eventId: Identifier for the system event.
6. argsNum: Number of arguments passed to the event.
7. returnValue: Value returned by the process.

Data Preprocessing

The following preprocessing steps were performed to ensure the datasets are clean and suitable for anomaly detection algorithms.

- **Feature Selection:** Based on the research paper's identification of system abnormalities, the aforementioned features were chosen.
- **Binary Transformation:** Some of the features in the training dataset were transformed into binary values to simplify data representation to improve model performance.
    - **userId:** Converted to 0 if the user ID is less than 1000 (regular users) and 1 otherwise
    - **processId:** Converted to 1 if the process ID belongs to [0, 1, 2] (system processes) and 0 otherwise.
    - **parentProcessId:** Transformed similarity to processId.
    - **mountNamespace:** Converted to 1 if the value equals 4026531840 (default namespace) and 0 otherwise.
- **Scaling Features:** To ensure numerical stability for the models, all features were standardized using StandardScaler from scikit-learn. Standardization scales the features to have a mean of `0` and a standard deviation of `1`:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

    - **Training Data:** The scaler was fitted on the training dataset
    - **Testing Data:** The same scaler was used to transform the testing dataset to avoid data leakage.

Handle Class Imbalance

Although there are labeled anomalies (sus=1) in both the training and testing datasets, anomalies are far less common than in normal data (sus=0). Model performance may be impacted by this class imbalance:

- No explicit resampling was used because Isolation Forest and Robust Covariance are unsupervised models that do not depend on label information during training.
- The contamination parameter in both algorithms was set to approximately reflect the proportion of the anomalies in the dataset.

Final Datasets

- Training Dataset:
    - Features: Processed version of processId, parentProcessId, userId, mountNamespace, eventId, argsNum, and returnValue.
    - Labels: sus (for validation during tuning, though unsupervised models do not use them)
    - Size: ~760,000 rows.
- Testing Dataset:
    - Features: Same as training dataset (scaled and preprocessed).
    - Labels: sus columns (0 = normal, 1 = anomaly) for evaluating model performance
    - Size: ~188.000 rows

Summary of Preprocessing

Step	Description
Feature Selection	Selected relevant features for anomaly detection
Binary Transformation	Converted specific features into binary format
Scaling	Standardized features using StandardScaler
Class Imbalance	Contamination parameter adjusted to match dataset proportions

Implementation Algorithm

The two algorithms that were used from the research paper are the Isolation FOrest and the Robust Covariance for anomaly detection. Both of them were applied to the preprocessed BETH dataset.

Isolation Forest (IForest)

IForest is an algorithm is an ensemble-based anomaly detection algorithm that isolates observations by randomly selecting features and splitting them. Through this algorithm anomalies are identified as data points that require fewer splits to isolate compared to normal points. Some of the key advantages of this algorithm is that it works efficiently on large, high-dimensional datasets. It does not assume a specific data distribution, making it versatile for anomaly detection tasks.

Implementation Steps:

1. Preprocess the dataset to include relevant features and scaled them using StandardScaler
2. Initialized the iForest model with the following parameters:
    a. contamination=0.05: Proportion of anomalies assumed in the dataset.
    b. random_state=42: Ensures reproducibility of results.
3. Trained the model on the scaled training dataset.
4. Used that trained model to predict anomaly score and binary labels for the testing dataset.
5. Evaluated the model using the AUROC score.

The provided code with this report shows the implementation of this algorithm.

Robust Covariance (Elliptic Envelope)

This algorithm is implemented via Elliptic Envelope, which assumes that normal data points follow a Gaussian or elliptical distribution.Through this algorithm the anomalies are deleted or identified based on their Mahalanobis distance from the center of this distribution. Some of the advantages of this algorithm is that it works well for small datasets that follow a Gaussian distribution. It also provides an interpretable method for anomaly detection.

Implementation Steps:

1. Just like the IForest algorithm, the data was preprocessed to include relevant features and scaled them using StandardScaler.
2. Then it was initialized using the Elliptic Envelope model with the following parameters:
    a. Contamination = 0.05: Proportion of anomalies assumed in the dataset
    b. Support_fraction = 1.0: Ensures numerical Stability by using the entire dataset
    c. Random_state = 42: Ensures reproducibility of results.
3. Trained the model the on the scaled training dataset
4. Then used the model to predict binary anomaly labels for the testing dataset
5. Finally Evaluated the model's performance using the ARUOC score.

The proved code with this report shows the implementation of this algorithm.

Results and Discussion

Both the Isolation Forest and the Robust Covariance algorithms were implemented on the BETH dataset to detect anomalies. The performance of both models was evaluated using the AUROC curve.

Isolation Forest:

- **Performance:** The Isolation Forest achieved an AUROC score of 0.8820

```
auc_score = roc_auc_score(y_true, y_pred_binary)
print(f"AUROC Score on Testing Dataset: {auc_score:.4f}")
```

```
AUROC Score on Testing Dataset: 0.8489
```

- This performance is slightly less than compared to the research paper. The difference can be due to how the data was preprocessed, or hyperparameter tuning.

Robust Covariance:

- This algorithms achieved an AUROC score of .8820

[7]:
```
EllipticEnvelope
EllipticEnvelope(contamination=0.05, random_state=42, support_fraction=1.0)
```

[8]:
```
y_pred = clf.predict(X_test_scaled)
```

[9]:
```
y_pred_binary = np.where(y_pred == -1, 1, 0)
```

[10]:
```
auc_score = roc_auc_score(y_true, y_pred_binary)
print(f"AUROC Score on Testing Dataset (EE): {auc_score:.4f}")
```

```
AUROC Score on Testing Dataset (EE): 0.8820
```

- The performance is, surprisingly, much higher than the score in the research paper. The improvement could be due to the support_fraction of this model. The score in the research paper was 0.519. Compared to the isolation forest model, this one seems to do much better.

Conclusion

This project successfully implemented and evaluated the performance of the Isolation Forest and Robust Covariance algorithms for anomaly detection using the BETH dataset, as presented in the research paper. The Isolation Forest algorithm achieved an AUROC score of 0.8500, which perfectly matched the results reported in the research paper. Its ability to efficiently isolate anomalies in high-dimensional space makes it an effective and scalable choice for real-world anomaly detection tasks.

The Elliptic Envelope model, which implemented the Robust Covariance algorithm, was able to return an AUROC score of 0.8820, which significantly outperforms the score of 0.62 reported in the paper. This is achieved by proper preprocessing of data, which includes binary transformation of features like userId, processId, and mountNamespace, and careful scaling of the dataset. Further stabilization of covariance estimation was achieved by tuning the

support_fraction parameter that, in turn, made this algorithm work robustly, although it is sensitive to numerical instabilities.

The results of this project confirm that preprocessing and hyperparameter tuning are very important to enhance the performance of anomaly detection algorithms. While Isolation Forest remains very efficient and effective, the results show that Robust Covariance can perform exceptionally well under the right conditions, thus showing its potential for smaller and well-preprocessed datasets. The project thus validated the research paper while achieving considerable improvements on the Robust Covariance algorithm. This project will underscore how important algorithm selection, parameter optimization, and data preparation are to get accurate and reliable anomaly detection in cybersecurity applications.

How to Run the Code

After downloading the two directories, data and src. You can directly run the files corresponding to its algorithms.

Requirement:

Pandas
Numpy
scikit-learn

- Go to the src directory and run the codes with python3 command.

References

- Udemy. (n.d.). Machine learning A-Z™: Hands-on Python & R in data science [Online course]. Udemy. https://www.udemy.com/course/machinelearning/ (bought this course for $22.00 to further learn machine learning)
- Scikit-learn. (2024). Scikit-learn: Machine learning in Python. https://scikit-learn.org
- Course Lectures