Anomaly Detection

CSEC620 - Project 5 - Mehul Sen

Overview

We conducted a study on the BETH Dataset to implement and evaluate various methods of anomaly detection. The dataset comprises 8,004,918 events and each record contains a label called "sus", which indicates suspicious or unusual activity, and outliers in the data distribution. We assessed the performance of several methods, including Robust Covariance, One-Class SVM, Isolation Forest, and Variational Auto Encoder, and compared our results with those presented in the paper. In addition, we implemented a Gradient Boosting Machine method (XGBoost), which outperformed all other methods.

Quick Start

This section will outline the necessary steps to use the Robust Covariance, One-Class SVM, Isolation Forest, Variational Auto Encoder, and XGBoost. This project was completed using Visual Studio Code with Jupyter Notebook. If you require assistance, we recommend referring to the following resource for setting up the notebook: Running Jupyter Notebook on Visual Studio Code | Medium

1. *Setup Dataset Folder*: To set up the dataset folder, start by unzipping and extracting the archive.zip files and placing them in a Dataset folder in the root directory. The dataset should be correctly located using the provided zip file, `project_5_sen.zip`. While we will only be using the `labelled_training_data.csv`, and the `labelled_testing_data.csv`. Other files do not interfere with the working of our algorithms. The desired folder structure should be as follows with the required files:

```
- requirements.txt
- project_5_anomaly_detection.ipynb
- Dataset/
    - archive/
        - labelled_training_data.csv
        - labelled_testing_data.csv
```

2. *Import Libraries*: Next, import the necessary libraries from requirements.txt using the following command: `pip install -r requirements.txt`.
3. *Run Notebook*: Finally, run the Jupyter Notebook (`project_5_anomaly_detection.ipynb`) and execute each cell block sequence from top to bottom.

Dataset

The project utilizes a single dataset known as the BETH dataset, which is detailed in Highnam et al.'s "BETH Dataset: Real Cybersecurity Data for Anomaly Detection Research." The dataset comprises 8,004,918 events gathered over 23 honeypots, operating for approximately five non-contiguous hours on a major cloud provider. It was divided into training, validation, and testing sets, with a rough 60/20/20 split for benchmarking purposes. We will be using the training and testing sets to evaluate our model.

The training set comprises 763,144 events from 8 hosts, while the testing set comprises 188,967 events from 1 host. We followed the same preprocessing steps as outlined in the paper and used in their baselines. The preprocessing involves dividing the dataset into its data and label counterparts. Additionally, the preprocessing involves extracting the following columns from the data: `processId`, `parentProcessId`, `userId`, `mountNamespace`, `eventId`, `argsNum`, and `returnValue`. hese columns are utilized as features for model training and testing. The dataset underwent the following feature transformation, which is identical to the benchmarks presented in the paper:

- `processId` and `parentProcessId` are transformed: values 0, 1, 2 are mapped to 0 (representing OS processes), and all other values are mapped to 1 (non-OS processes).
- `userId` is transformed: values less than 1000 are mapped to 0, and values 1000 and above are mapped to 1. To distinguish between system users (typically having lower user IDs) and regular users.
- `mountNamespace` is transformed: a specific value (4026531840) is mapped to 0, and all other values are mapped to 1. To differentiate between a specific mount namespace (OS's mountspace) and others.
- `returnValue` is transformed into three categories: 0 for a return value of 0 (indicating success), 1 for positive return values (success with a value), and 2 for negative return values (error).

For more information about the dataset, please refer to the previously mentioned paper.

Methods:

Method #1: Robust Covariance

To detect anomalies in a dataset, the first model used is Robust Covariance. To achieve this, we use the EllipticEnvelope model from the sklearn library. This method is helpful in detecting outliers in a Gaussian distributed dataset.

Robust Covariance is useful in situations where the data may contain some amount of contamination like anomalies or outliers. These outliers can skew the results of standard covariance measurements. The method employs an Elliptic Envelope approach that fits an ellipse to the central data points, assuming a Gaussian distribution. Points that lie outside this ellipse are considered outliers or anomalies. To determine whether a data point is inside or outside the ellipse, the method uses the Mahalanobis distance. The Mahalanobis distance is a measure of the distance between a point and a distribution. Higher Mahalanobis distances indicate that a point is likely an outlier.

Method #2: One-Class Support Vector Machine(SVM)

The second model used is the one-class support vector machine. This model discriminates between in-distribution data and out-of-distribution data to perform anomaly detection. It is specifically designed for anomaly detection in unsupervised learning. The model works by finding a decision boundary, which is a hyperplane in a high-dimensional space, that best separates the data points. The key idea behind this model is to map the input data into a high-dimensional feature space and then identify the smallest region within this space that encloses most of the data points. Any data points that fall outside this region are considered anomalies.

The One Class SVM tries to encompass most of the normal data while isolating the outliers. It uses a non-linear kernel to project the data into a higher dimensional space and then seeks to separate the data from the origin with the maximum margin. This creates a boundary around the 'normal' data, and anything that falls outside this boundary is flagged as an anomaly.

Method #3: Isolation Forest

This model is designed to identify anomalous behavior within a dataset by using a small set of discrete features and the conspicuous nature of attacks. It does not rely on profiling normal data points, but instead explicitly isolates anomalies. To achieve this, the model randomly selects a feature and a split value between the maximum and minimum values of the selected feature, producing shorter paths for anomalies that have distinct value ranges. As a result, when a forest of random trees collectively produces shorter path lengths for particular points, these points are more likely to be anomalies. This makes the model highly effective at differentiating suspicious events from benign ones.

Method #4: Variational Auto Encoder(VAE) + Density of States Estimate(DoSE) Support Vector Machine(SVM)

An effective method for detecting anomalies involves using a Variational Auto Encoder (VAE) to model observations as a product of categorical distributions. This model is then used by Density of State Estimates (DoSE) with a linear one-class SVM trained using SGD to identify anomalies.

VAEs are a type of generative model that can learn complex data distributions. DOSE is an approach that balances the dataset. The VAE learns to encode and decode input data, effectively creating a compressed representation of the data. It can then reconstruct what normal data should look like. The DOSE approach generates synthetic samples of the minority class (in this case, anomalies) to balance the dataset. By training on this balanced dataset, the model can better learn to distinguish between normal and anomalous data. When new data is input into the system, the VAE attempts to reconstruct it. If the reconstruction error is high, it implies that the data is not similar to the 'normal' data the VAE was trained on, flagging it as an anomaly.

Method #5: Gradient Boosting Machine (XGBoost)

The eXtreme Gradient Boosting (XGBoost) algorithm is a regression and classification tool that can be adapted for anomaly detection. It was used as a new method to evaluate its performance against other techniques in the paper.

Gradient boosting is an ensemble strategy that builds models sequentially, with each new model attempting to correct the errors made by the previous models. In XGBoost, decision trees are used as base learners. Each tree is added to the model in a way that minimizes the loss function (such as cross-entropy loss for classification problems), using a gradient descent-like algorithm.

XGBoost can be trained on labeled data (normal and anomalous) and can learn to differentiate between these two classes. Once the model is trained, it can predict whether new data points are normal or anomalies based on the learned patterns.

Evaluation

The following were the performance metrics of each of the methods used:

- **Robust Covariance**
    - **AUCROC** : 0.49653769660705876

    - **Accuracy**: 0.09260347044722095
- **One-Class SVM**
    - **AUCROC** : 0.5984245500758777
    - **Accuracy**: 0.8781903718638704
- **Isolation Forests**
    - **AUCROC** : 0.7707815144740071
    - **Accuracy**: 0.8805452803928728
- **VAE + DoSE (SVM)**
    - **AUCROC** : 0.7355800062161345
    - **Accuracy**: 0.9147099758158832
- **XGBoost**
    - **AUCROC** : 0.9702319504954537
    - **Accuracy**: 0.9459799859234681

Results

The paper titled "BETH Dataset: Real Cybersecurity Data for Anomaly Detection Research" by Kate Highnam, Kai Arulkumaran, Zachary Hanif and Nicholas R. Jennings, evaluates the dataset with four anomaly detection methods. The authors report the AUCROC score for each of their methods as benchmarks. In this project, we attempted to reproduce these methods and our findings are as follows:

- **Robust Covariance**: The paper reports an AUCROC value of 0.519, while my implementation achieved a value of 0.496 which is very close to the reported findings.
- **One-Class SVM**: The paper reports an AUCROC value of 0.605, while my implementation achieved a value of 0.598 which is very close to the reported findings.
- **Isolation Forests**: The paper reports an AUCROC value of 0.850, while my implementation achieved a value of 0.770 which is close to the reported findings.
- **VAE + DoSE (SVM)**: The paper reports an AUCROC value of 0.698, while my implementation achieved a value of 0.735 which is close to the reported findings.
- **XGBoost**: Lastly, we implemented a new method, Gradient Boosting Machine, which significantly outperformed all previous methods, achieving an AUCROC value of 0.970.

While the paper did not report the accuracy of the methods implemented, we found that XGBoost was the most accurate in its predictions, while robust covariance performed the poorest.

Conclusion

In project 5, we successfully implemented the four methods described in the paper and trained the provided BETH dataset to achieve similar results. This proves that the results mentioned in the paper can be replicated. Additionally, we implemented a new algorithm called XGBoost, which outperformed the other algorithms significantly in anomaly detection. This suggests that XGBoost is much better suited for detecting anomalies on this dataset compared to the other proposed algorithms.