Anomaly-Based Network Intrusion Detection Using KDD-99

By: Enrique Rodriguez, Andre Lasola, Ian Gower

Abstract

We plan to use several machine learning models to identify potential threats using the KDD Cup 1999 dataset to train and test our system. We will pre-process the dataset by handling missing values, encoding categorical data, and feature scaling. This dataset will then be run through three different models, Random Forest, Support Vector Machine, and Isolation Forest. Lastly, we will evaluate the model across a variety of metrics to best detect and prevent bad actors from accessing the network.

Introduction

Network Intrusion Detection Systems, or NIDS, are security tools that monitor network traffic for suspicious activity and potential threats. They aim to detect unauthorized access or abuse of network resources. NIDS can mostly be classified into two categories.

Signature based detection relies on known patterns of malicious activity to identify intrusions. It works well for known threats, but struggles with unknown attacks.

Anomaly based detection estables a baseline of normal network behavior and identifies deviations from this baseline as potential intrusions. Anomaly detection is more effective at identifying novel attacks that may not have specific known patterns of malicious behaviour.

Anomaly detection is crucial for a few reasons. Since anomaly-based methods do not rely on pre-defined signatures, they can detect previously unknown attacks that do not match any existing patterns. These types of attacks are typically referred to as novel attacks. Networks are continuously evolving, with new applications and users frequently introduced. Anomaly detection can adapt to these changes by updating the baseline of normal behavior dynamically. By identifying deviations from normal behavior, anomaly detection can reduce the risk of missing attacks that signature-based systems might overlook.

The origins KDD Cup 1999 dataset contains about 4.3 million connection records. Each record represents a network connection and consists of 41 features, which can be categorized as basic features, content features, and traffic features.

Basic features consist of characteristics of the connection, such as duration, protocol type, or service. Whereas, content features contain information about the connection’s payload, including

the number of failed login attempts. Lastly, traffic features are statistical properties of connections over a time window, such as the number of connections from the same host.
The dataset includes labeled instances of both normal and attack traffic. The attacks are categorized into several types, such as Denial of Service (DoS), User to Root (U2R), Remote to Local (R2L), and probe attacks.

This project seeks to create a NIDS to effectively detect and prevent bad actors from accessing the network. Our NIDS will be anomaly based, to better detect novel attacks as well as known threats. We will employ a variety of pre-processing methods, machine learning models, and evaluation metrics, to maximize the capabilities of our system.

Methodology

Data Preprocessing

In order to prepare the dataset for training and evaluation, we will preprocess the data with the following methods:
Handling missing values:

- The first step would be to go through to address missing or inconsistent values, when they are found, they will be replaced with the column's mean or median, or the row will be removed if necessary.
- No missing values were detected, so no changes were made to the dataset.

Categorical Encoding:

- The dataset has categorical columns such as protocol_type, service, and flag, and in order to train the models these will be converted into numerical representations using one hot encoding.
- The categorical columns were adjusted to be encoded using one-hot encoding. This allows for the models to treat each separate value within these columns as its own feature.
- The attack label was encoded to 5 separate attack categories. This allows for the model to group similar types of attacks without introducing unnecessary complexity to its output.

Feature Scaling:

- The numerical values will be scaled using Standardization, specifically the standard scalar.
- The values were scaled to have a mean of ~ 0 and a standard deviation of ~ 1.

Model Development

These are the three models we are going to use:

Random Forests:

- With the data in training and testing sets, a Random Forest classifier will be trained using the training data. The model will learn to distinguish between normal and malicious connections based on the features provided in the training data.
- We can then optimize hyperparameters such as the number of trees, maximum depth of trees, and minimum samples per leaf. This can be accomplished by using techniques like grid search or random search.
- Random Forest provides a measure of feature importance, determining which features are most important in making predictions. This can help in understanding the most important characteristics of malicious network traffic.
- Random Forest is able to handle datasets with high dimensionality, which will serve this dataset well. It is also less prone to overfitting as other models, which makes it suitable for this complex dataset, while maintaining accuracy for novel attacks.

Support Vector Machines:

- With SVM, we would be able to effectively detect anomalies as SVM is able to distinguish the boundary between normal and anomalous classes with a hyperplane that separates them, SVM works especially well with high dimensional data. It can be enhanced using kernel functions for nonlinear relationships.
- SVM will use the scikit-learn library, and the hyperparameter tuning will be done by using grid search to find the best values.

Isolation Forest:

- Isolation Forest is an algorithm specifically designed for Outlier Detection and operates by isolating outliers rather than profiling normal data points. Any inconsistent or imbalanced data will be handled appropriately.
- Isolation Forest's computational complexity is fairly low, making it great for large datasets. It can also handle high-dimensional data very well without requiring too much hyperparameter tuning, making it especially useful for the KDD Cup 1999 dataset, which has a mix of categorical and numerical features. It would be helpful to use Grid Search or Randomized Search to find the best parameter set and use traditional supervised metrics listed below to evaluate different hyperparameter configurations.
- Since each tree is built using a random subset of features and split points, the algorithm's randomness can help to decorrelate the trees and make Isolation Forest robust to irrelevant and redundant features. Because it isolates points through random partitioning and aggregates results across many trees, it is much less likely to overreact to minor noise in the area.

Performance Metrics

To evaluate our model performance, we will use the following metrics:

Accuracy

- This will measure the overall performance of the model.

Precision

- This will measure the proportion of all identified positive outcomes out of the predicted positive outcomes. TP/(TP + FP)

Recall

- This will measure the ability of the model to identify the anomalies. TP/(TP + FN)

F1-Score

- This will calculate the harmonic mean of Precision and Recall. $2 * \frac{Precision * Recall}{Precision + Recall}$

Runtime Analysis

- This will determine the efficiency of the models, especially due to the large dataset.

Results

Random Forest

The Random Forest classifier reached its peak performance with a configuration of 100 estimators, balanced class weights, and tuned parameters including min_samples_leaf=2 and min_samples_split=5. Under this setup, the model achieved a near flawless overall accuracy of ~100% on the KDD-99 dataset. The weighted averages for Precision, Recall, and F1-score were all at a rounded 1.00, indicating excellent overall classification. Even under the macro average (which reflects performance across imbalanced classes), the F1-score remained strong at 0.93, confirming the model's robustness in detecting both prevalent and rare classes effectively.
The Random Forest model was optimized through a combination of manual parameter tuning and domain-aware adjustments aimed at improving classification performance on the imbalanced KDD-99 dataset. Key hyperparameters were set to enhance generalization and handle skewed class distributions, including n_estimators=100 to ensure robust ensemble learning, min_samples_leaf=2 and min_samples_split=5 to control overfitting and promote deeper, more meaningful splits, and class_weight='balanced' to compensate for the disproportionate frequency of attack versus normal traffic. A full grid search was used to inform the final model to strike an effective balance between accuracy and computational efficiency. This streamlined tuning process ultimately yielded a high-performing classifier with a strong macro F1-score, validating the model's ability to accurately capture both common and rare intrusion patterns.

Isolation Forest

The Isolation Forest achieved its best configuration with `n_estimators=500, max_samples=4096, contamination=0.01,` and `max features=1.0`. With this setup, the model produced an overall accuracy of 99.705%, with the weighted averages of Precision, Recall, and F1-score being at a consistent 0.99 for normal data and at an astounding 1.00 for attack data, demonstrating near-perfect separation of inliers and outliers under the KDD-99 distribution.

Support Vector Machine (SVM)

The Support Vector Machine model was hypertuned for performance and speed, being done so with RandomizedSearchCV for its hyperparameter tuning, specifically using uniform(0.1, 5),

scale gamma, rbf kernel, while also balancing class weight and using 2 fold cross validation for efficiency. After its hypertuning and training, the model was able to reach an overall accuracy of 99.86% and a runtime of 10.27 seconds. It excelled on high support classes like Normal, DoS, and Probe, with an almost perfect precision and recall. The macro averaged F1 score was 88%, while the weighted average F1-score was 100%, showcasing its ability to correctly classify the dominant classes. Although it had strong performance overall, it struggled with the small classes like R2L and U2R, which U2R had only 8 instances in the test set and resulted in a low F1 score of 59%.

Random Forest Confusion matrix:

[[58716 3 0 0 0]
[1 14585 3 1 2]
[0 0 616 0 0]
[1 7 0 159 2]
[0 1 0 0 7]]

SVM Confusion matrix:

[[58713 6 0 0 0]
[4 14509 17 58 4]
[0 0 616 0 0]
[1 9 0 159 0]
[0 3 0 0 5]]

Isolation Forest Confusion matrix:

[[288965 2869]
[1467 1176229]]

Discussion

Isolation Forest is ideally suited for unsupervised anomaly detection. By isolating outliers in a random partitioning process, it does not require labeled attack data during training. On the KDD-99 set, however, most records are attacks and only about 20% are normal traffic, so "normal" becomes the minority class in testing. Although an accuracy rate of 99.705% sounds exceptional, the remaining 0.295% error still corresponds to thousands of false positives and false negatives when scaled to millions of records. To reduce these mistakes, one might adjust the contamination parameter, select only the most informative features, or pair Isolation Forest with other methods, such as a supervised model, to cut down on these errors.

The Random Forest and Isolation Forest models serve complementary roles in intrusion detection on the KDD-99 dataset, each excelling in different scenarios. Random Forest, a supervised learning algorithm, is designed for situations where labeled data is available and accurate classification of known attack types is critical. With its configuration of 100 estimators and balanced class weights, it achieved an accuracy very close to 100%, along with weighted precision, recall, and F1-scores all at 1.00. Even its macro-average F1-score, which accounts for performance across imbalanced classes, remained high at 0.93, showcasing its robustness and reliability. This makes Random Forest particularly valuable for detailed, interpretable predictions and for operational environments where known threats must be identified with high fidelity.

Despite its good metrics, there is still room for improvement in the Random Forest model. Given the relatively imbalanced dataset, the model can struggle to identify each and every attack, however by accounting for this by specifying the model to use balanced classes, this is corrected to some degree.

The Support Vector Machine (SVM) also was highly effective on the KDD-99 dataset, with an overall accuracy of 99.86% and a runtime of approximately 10 seconds. After performing hyperparameter optimization with RandomizedSearchCV, the SVM model showed high precision and recall on the most common attack categories, while still being able to classify the rarer classes. Although the macro averaged F1 score was lower than Random Forest due to the extreme class imbalance, the weighted F1 score stayed at 100% showing strong reliability. The class weighting and rbf kernel allowed SVM to do well on non linear decision boundaries that are normal in this data. SVM is typically more computationally intensive, but despite that the

model maintained its efficiency and strong generalization when tuned properly, making it a valuable supervised option along with Random Forest.

Conclusion:

All three models, Random Forest, SVM, and Isolation Forest, showed excellent performance on the KDD-99 dataset, each reaching accuracy rates within a few thousandths of perfect and weighted F1-scores above 0.98. However, even a 0.1-0.3% error rate can lead to thousands of false positives or false negatives when monitoring millions of network events, potentially missing important attacks or flooding analysts with alerts. To improve real-world performance, future work could mix supervised and unsupervised models, adjust how strictly attacks are flagged, and keep the feature set up to date as traffic patterns change. These steps should maintain high detection rates while cutting down the few but important errors that can have big impacts.

Task Division

Ian Gower - Random Forest
Enrique Rodriguez - Support Vector Machine (SVM)
Andre Lasola - Isolation Forest

Each member serves as the primary developer for their assigned model and will produce detailed documentation covering its hyperparameters and training process. General project documentation is divided equally among all members, and everyone will participate in proofreading and finalizing the report.