**Anomaly Detection in Network Traffic**

By Sawyer Blankenship, Anka Engin, Stephen Parker, Elijah Uy

**Abstract**

The objective of this project is anomaly detection in network traffic. The aim is to compare multiple models and their performance in anomalous network activity, to identify potential threats and ensure system security. We will compare these models on their evaluation of the KDD Cup 1999 dataset. Using different preprocessing and ML models, we will evaluate the effectiveness of the methods based on computational efficiency and the ability to handle skewed or uneven data.

**Introduction**

To identify potential threats and ensure system security, our team is developing four different machine learning models to catch anomalies and prevent potential threats and security breaches. As such, the branch of Network Intrusion Detection we are focusing on is Anomaly Detection. To catch anomalies means in our context means to identify data points, events, or observations that deviate significantly from the norm. These unexpected phenomena can indicate a potential threat to the system, making them significant to identify and investigate.

For this project, we will be utilizing the KDD Cup 1999 dataset for comparison and evaluation of our models. This is the data set used for The Third International Knowledge Discovery and Data Mining Tools Competition, which was held in conjunction with KDD-99 The Fifth International Conference on Knowledge Discovery and Data Mining. The competition’s goal was to create a network intrusion detector that could differentiate between normal and abnormal connections. In the same vein, our team will create four models to tackle the same task of identifying potential security threats. We will develop decision trees, random forests, neural networks, and support vector machines.

**Methodology**

**Preprocessing**

To enhance our model’s ability to understand our data, we will preprocess with pandas. Since we are building a neural network, we need to normalize numerical features to prevent exploding gradients. The duration parameters range from 0 to 58329, and are heavily skewed to the right. 58329 is an outlier. We will perform a log transformation to reduce the impact of outliers (otherwise 0 and 100 would be treated very similarly). Then, we will min-max scale the distribution to [0, 1] to prevent exploding gradients. For skewed-features with maximums ranging from 30-1000 including hot, num\_compromised, etc., we will perform the same. We will do the same with the src\_bytes and dest\_bytes features, as they are also heavily skewed. Src\_bytes’s maximum is 693,375,640, while every other point is less than 10,000,000.

For numerical features with maximums less than 10, including urgent, num\_failed\_logins, num\_shells, etc., we will simply perform min-max scaling to [0, 1]. For count and srv\_count, which have a heavily bimodal distribution we’ll min-max scale to [-1, 1] so the model can easily distinguish between peaks, and place more emphasis on more polar data. The rate features are already fit to the range [0, 1] and follow a heavily bimodal distribution (either 0-10% or 90-100%). We’ll scale this to [-1, 1] to make polarity slightly increase the network’s emphasis on these features.

In addition to these steps, we will apply one-hot encoding to categorical features. This preprocessing strategy ensures that each category is converted into its own feature dimension. This will help to prevent any unintended ordinal relationships. One-hot encoding will also allow our decision trees to better learn from these categorical inputs.

**Model Selection**

A lot of these features are very bimodal, so decision trees will easily separate the data. We will use a random forest to improve the accuracy of decision trees. Random Forests don’t require much tuning and are good at making accurate predictions, especially for classification. Decision trees don’t need their input data to be normalized. We still need to normalize data for the neural networks though. Decision trees also are not thrown off by outliers. Unlike neural networks, which are black boxes, decision trees give us transparency into how the tree uses features to make a decision. We’ll train a forest both as a binary classifier predicting whether or not a connection is malicious, and as a classifier predicting which type of attack is most likely.

We will also incorporate a standalone Decision Tree model in addition to the random forest. While the random forest gives us an ensemble of trees, a single Decision Tree is straightforward to interpret, quick to train, and can serve as a baseline for comparing the performance of more complex models.

In addition to random forest and decision trees, we will implement a Neural Network, specifically, a Multi-Layer Perceptron. Because the KDD dataset has a lot of dimensions, a NN allows for a thorough, hierarchical evaluation of the dataset. We will use ReLu as our activation function, and Adam as our optimizer. To prevent overfitting, we will use dropout and L2 regularization. Incorporating these into our NN will produce a very flexible, robust model to evaluate the KDD dataset.

Finally, we will implement support vector machines (SVM).used for both classification and regression tasks, with a strong emphasis on classification that works by identifying the optimal hyperplane in an N-dimensional space that best separates data points into distinct classes. This model can be effective in high-dimensional data which will allow it to better analyze the KDD dataset and can handle non-linear relationships. However, it can be difficult to tune the SVM model as it is prone to a high amount of false positives and additionally can be computationally intensive which may lead to the model being inefficient overall.

**Evaluation Metrics**

We will evaluate the random forest as a binary classifier by testing the trained forest on the KDD test dataset, and measuring the accuracy, recall, and precision. We will also evaluate the random forest as a classifier, measuring the percentage of attack types correctly identified. We will prioritize recall over precision since the consequences of missing an attack are more severe than falsely flagging a legitimate connection.

We will also evaluate the performances of the decision tree model versus the random forest model by analyzing how well each model handles bimodal features and outlines and how interpretable the results are for security analysis. This will enable us to discuss their pros and cons for network anomaly detection more effectively and give us a better idea of which model is more suitable for network security.

To evaluate the NN model, we can monitor the accuracy, recall, and F1 score. We can also monitor the validation loss to see if any overfitting occurs. We can visualize the model’s performance using a confusion matrix. We will prioritize accuracy over recall, because precision in detecting network anomalies is the NN’s main goal.

We can evaluate the SVM model by creating a confusion matrix that will illustrate the true positives and negatives and false positives and negatives that the model has made. From this, we can find its precision, recall, F1-score, and accuracy. A high recall with this model is to be expected and is important for our task, but we will need to evaluate its precision to evaluate the model’s effectiveness.

**Task Division**

Stephen: Contribute to the Decision Tree model, implement one-hot encoding for data preprocessing, and assist with overall coding.

Sawyer: Normalization preprocessing, Random Forest Model

Elijah: Support Vector Machines, Introduction

Anka: Neural Network/Multilayer Perception, Abstract