DDOS Detection using ML

Capstone Project final Review

Team Members

- SIDDHU CHELLURU - 19BCY10178
- SAHITHI D - 19BCY10164
- KLB SHANKAR - 19BCY10166
- ROHIT KUMAR SINGH - 19BCY10160

Introduction

DDoS attacks are a major threat to online services, making accurate detection crucial. Machine Learning (ML) algorithms, such as Random Forest, KNN, Logistic Regression, and SV classifiers, can be used to detect these attacks by analyzing network traffic patterns and identifying anomalies. These algorithms learn from historical data to identify normal network behavior and detect any deviations that may indicate a DDoS attack, leading to increased accuracy, fewer false alarms, and improved response time.

Working Process of the Code

1. Import Libraries

The libraries used in code include:

- Numpy and Pandas for data manipulation and analysis
- Matplotlib and Seaborn for data visualisation
- Sklearn for building and evaluating machine learning models

The code imports the following ML algorithms from Sklearn:

- Random Forest Classifier
- K-Nearest Neighbors (KNN) Classifier
- Support Vector Classifier (SVC)
- Gaussian Naive Bayes
- Decision Tree Classifier

The code also imports evaluation metrics from Sklearn to measure the performance of the model such as accuracy, precision, recall, F1-score, and confusion matrix.

2. Import Dataset

NSL-KDD Dataset:

NSL-KDD is a widely used dataset for the evaluation of intrusion detection systems (IDSs) in the field of cybersecurity. It is an improved version of the original KDD Cup 99 dataset, which was created to address some of its limitations. The NSL-KDD dataset contains various types of network attacks, including Distributed Denial of Service (DDOS) attacks, making it suitable for this project on DDOS detection.

In the code, We are importing KDDTest+.txt and KDDTrain+.txt into df and test_df (Training dataset and Testing dataset).

3. Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a crucial step in the data science process as it helps to understand the structure and distribution of the data. In the context of this project, EDA was performed on the NSL-KDD dataset to understand the distribution of different types of attack classes, to detect any missing or inconsistent values, and to identify any outliers in the data. The EDA process involved visualizing the data using various plots and statistical methods to gain insights into the characteristics of the data and inform the feature selection process.

EDA Helps us in:

- Understanding the Data
- Checking for missing Values
- Analysing the Distribution of Data

4.Data Encoding

This code prepares the training and testing datasets for machine learning by converting categorical columns into numerical data. This is done using the ("get_dummies") method in Pandas. The resulting data has multiple columns, one for each unique value in the original categorical column. The target column ("attack") is then removed from both the training and testing datasets. Data encoding is important for machine learning as it ensures that the data is in a format that can be used with the selected algorithms.

5. Feature Selection

The code performs feature selection for a classification problem using a NSL-KDD dataset. The dataset is divided into training and testing sets, and the correlation between the features and the target variable ("attack_class") is calculated. Features with a correlation greater than 0.1 are selected and the remaining features are dropped. This is done to reduce the number of features and improve the performance of the machine learning model. The heatmap of the selected features is plotted to visualize the correlation between the features. The final datasets with only the selected features are used for further analysis and modeling.

6. Classification

The code Performs a classification task on a given dataset. The dataset is divided into two parts, training data (X_train and y_train) and testing data (X_test and y_test). The 'attack_class' column is used as the target variable for both training and testing data.

The function add_predictions takes three inputs - the data set, predicted values, and actual target values. It adds the predicted and actual target values to the data set and creates a confusion matrix to evaluate the performance of the model. The function also captures rows with failed predictions, non-zero values and differentiates between false positives and false negatives. Finally, the function returns a dictionary containing original data, confusion matrix, errors, non-zero values, false positives, and false negatives.

7. Evaluating the Model Performance

In the code, various classification algorithms are applied on the data set, including Naive Bayes, Decision Tree, K-Nearest Neighbors, Random Forest, and Support Vector Machines. For each model, the accuracy of the model is calculated using the metrics.accuracy_score method. The confusion matrix is also plotted using the splt.plot_confusion_matrix method. Finally, the classification report, which provides a summary of the precision, recall, F1-score, and support of the model, is printed. The accuracy, confusion matrix, and classification report are useful in evaluating the performance of the models, allowing us to determine which model is the best fit for our data set.

8. Repeating the Process with Normalisation

The code Performs feature normalization, specifically using MinMaxScaler, on the training and test data. This means that the values of the features in the training and test datasets will be transformed such that all features will be in the range of 0 to 1. Normalization is a common preprocessing step in machine learning, especially in cases where the features have different scales, to avoid bias in the model towards features with higher magnitude.

And then Various classification algorithms are applied on the data set, including Naive Bayes, Decision Tree, K-Nearest Neighbors, Random Forest, and Support Vector Machines.Which gives us the best result i.e, More accuracy for the model.

Total evaluations in the Code

The code performs a total of 10 evaluations using 5 different classification algorithms. These algorithms are Gaussian Naive Bayes, Decision Tree, K-Nearest Neighbors, Random Forest, and Support Vector Machines. Each algorithm is evaluated twice, once with normalization and once without normalization of the dataset. The evaluation metrics used to determine the performance of the models are accuracy, confusion matrix, and a classification report. These metrics provide insight into how well the models are able to accurately predict the class labels in the test data. The results of the evaluations can then be used to determine which algorithm performs the best on the given dataset and can be used for further analysis.

Advantages of this Model

1. Simplicity: The code has a straightforward approach to DDOS detection and can be easily understood and implemented by others.
2. Flexibility: The code allows for the evaluation of 5 different machine learning algorithms. Gaussian Naive Bayes, Decision Tree, K-Nearest Neighbors, Random Forest, and Support Vector Machines., and provides the option to perform these evaluations with and without normalization, providing flexibility in testing various models.
3. Customizability: The code can be easily modified to fit specific requirements or be optimized for a particular use case.
4. Versatility: The code can be used for a variety of DDOS detection tasks, including network-based and host-based DDOS detection, making it a versatile solution.

Conclusion

In this project code, various classification algorithms such as Naive Bayes, Decision Tree, K-Nearest Neighbors, Random Forest and Support Vector Machines have been evaluated on the NSL-KDD dataset. The evaluation of each algorithm is done twice, once with normalization and once without normalization. The evaluation is done by fitting the algorithm to the training data and making predictions on the test data. The accuracy of each algorithm is then determined using the accuracy_score metric from the scikit-learn library. Additionally, a confusion matrix and classification report is also generated for each algorithm to provide a more comprehensive evaluation of the performance of the algorithm. Overall, the evaluation of the algorithms in this project code provides insight into the performance of different classification algorithms on the NSL-KDD dataset.

“Empowering our technological advancements to create a safer digital world, one step at a time.”

Thank You!