Cybersecurity attack type detection

DSTI - Machine Learning Labs Project Report—Group 17

Authors: Omar EL Osta, Nandini Gantayat, Vivian Kibet, Omprakash, Laura van de Baan and Jasen Steve Zapfack

Github link

Complete_pipeline_link

Application_code_link

Introduction

Description of the dataset and the problem at hand

The objective of this project is to predict the type of cyberattack based on the dataset at hand. This pipeline was constructed in Jupyter notebooks using multiple Python packages, and the target variables can be predicted using an application based on Streamlit. The model is an XGBoost classification model. The dataset used was provided by DSTI, and contains 25 variables, with varying types. In total the dataset contains 40000 entries, with no duplicate entries.

Methodology

Exploratory data analysis

The dataset was analyzed using different Python packages such as pandas to create data frames and matplotlib to create visualizations.
First the dataset is checked for missing values. The variables “Malware Indicators”, “Alerts/Warnings”, “Proxy Information”, “Firewall Logs” and “IDS/IPS Alerts” contain missing values. Furthermore, the “Source IP Address” contains only distinct values, and therefore the “Source IP Address” can be considered as the primary key of the dataset.

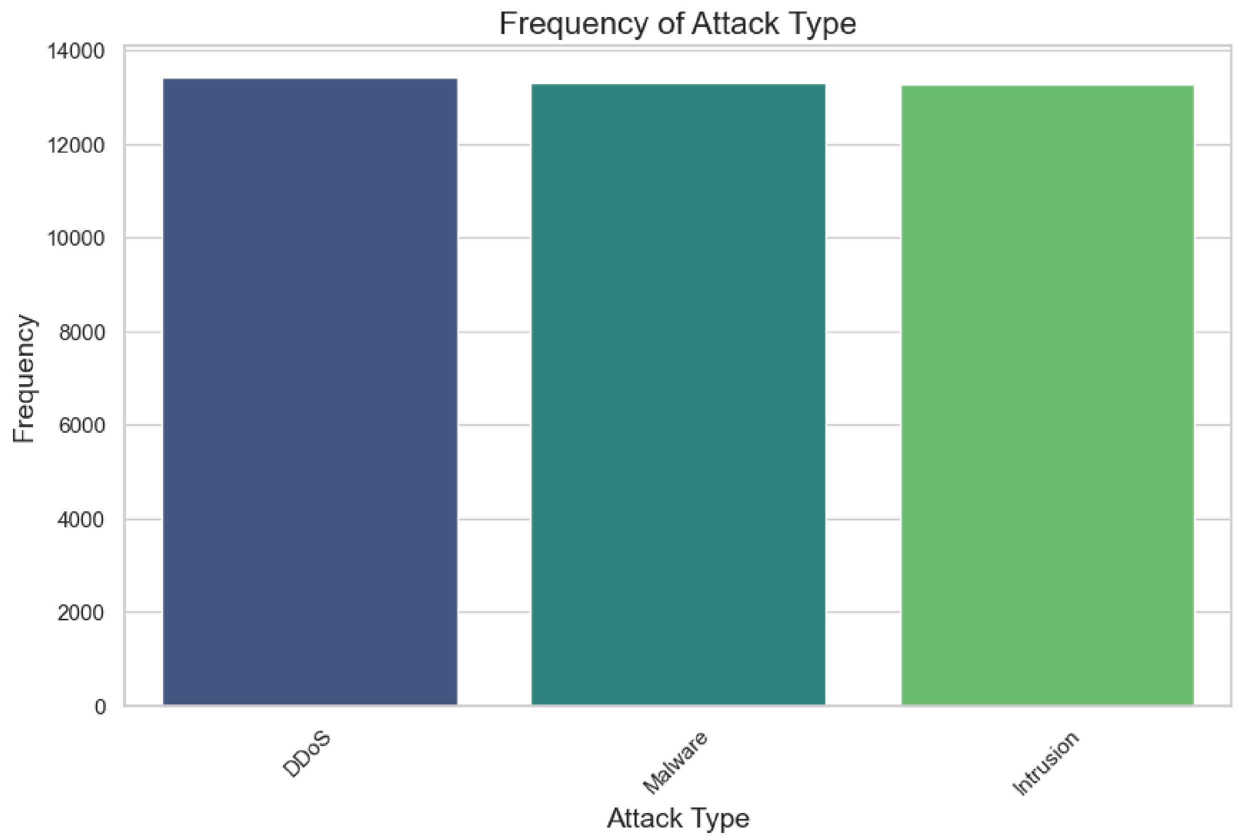
Second, the correlation between the variables is checked. There exists no correlation between any variables as all correlation between the variables is approximately equal to 0. It is also noted that variables that contain missing variables do not have the same amount of missing data, it can therefore be concluded that the alerts are not triggered only by which type of attack it is, this is also demonstrated later in the section.

After this global summary of the dataset, the variables are analyzed more closely, notably the interaction between certain of them and the target variable, “Attack Type”.

Figure 1.1 shows the different amount of times each attack types appears in the dataset. It can be seen that it is evenly distributed, each attack appears approximately one third of the time in the dataset. In appendix A, the figures show that this is the case for most of the variables, such as the protocols used and the severity levels of the attacks.

Figure 1.1

The number of attacks for each attack Type (DDoS, Malware, Intrusion)



It can be shown that for most variables that have a relationship with the attack, such as the technology used during the attack or what occurred during the attack, the distribution is evenly spaced for each attack type. Figures 1.2 and 1.3 display this distribution for the actions taken for each attack type, and the alerts send for each attack type.

Figure 1.2

Actions taken for each attack type (DDoS, Malware, Intrusion)

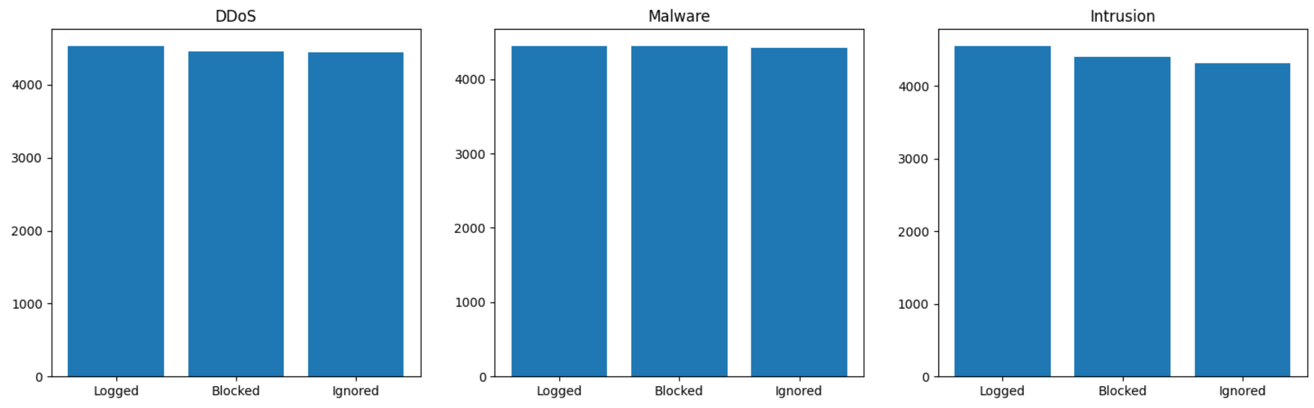
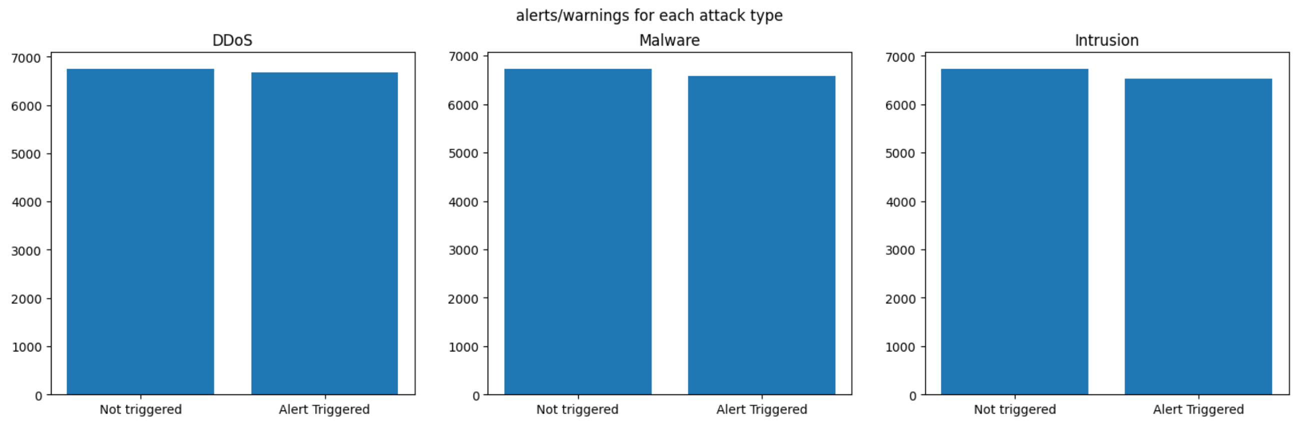


Figure 1.3

Alerts send based on each attack type (DDoS, Malware, Intrusion)



Furthermore, looking more deeply into the ports and the packet types, it can be stated that there are 33 destination ports that appear more than 5 times in the dataset and only 23 source ports. The source ports are more often the same than the destination port. There are two different packet types, data and control. The data packets all have a length above 350, and the control packets have a length above 300. In appendix A, the histogram of the packet lengths in depicted.

The user information variable is not unique, which indicates that there are users that change source IP addresses as this is an unique variable.

Feature engineering and selection

The variables containing missing values, with the exception of “Proxy Information” and “Firewall Logs”, represent whether the system had this variable present during the attack or not. The missing values are thus a representation of when the entry does not have the concerning variable and are therefore encoded using ordinal and vector encoding. The variables 'Protocol', 'Packet Type', 'Traffic Type', 'Malware Indicators', 'Alerts/Warnings', 'Attack Signature', 'Action Taken', 'Network Segment', 'Firewall Logs', 'IDS/IPS Alerts', 'Log Source' are encoded using vector encoding and the variable ‘Severity Level’ is encoded using ordinal encoding. The original variables are dropped and replaced by their encoded versions.

Model selection, comparison and evaluation

The target variable “Attack Type” has three categories and therefore a multiclass classification model is necessary. As there are more than 1000 entries and less than 1000 features, the model chosen to predict the target is an XGBoost Tree classifier. An XGBoost Tree classifier is an extreme gradient boosting based on decision trees. It is a powerful prediction model that deals with outliers, missing values and correlations if necessary (Aydin & Ozturk, 2021). In the processed data there are no such values.

Other models are also tested, such as a simple tree classifier, however, the XGBoost classifier has the best overall accuracy. XGboost has a high computational speed thanks to the parallel processing it uses. It is made up of several decision trees, each of which is a weak learner, and therefore individually they do not have a high prediction accuracy (Wang et al, 2022). But boosting technology promotes the classifier to a strong learner, with a higher prediction accuracy than other classifiers. Thanks to the addition of regularization, XGBoost is more capable of preventing overfitting (Amjad et al, 2022). Since the dataset is limited in its size and has considerable features to consider, it is more prone to overfitting (Ying, 2019). Therefore, the XGBoost classifier makes an excellent choice for predicting the target variable and has a better performance than other models for this problem.

To validate the model, the F1 scores, precision and recall of the classifier for each target class is calculated to make sure the classifier predicts the right value.

Furthermore, feature importance is calculated using the build in feature importance command.

Results

Final Model Performance (XGBoost)

The XGBoost model achieved an accuracy of 33%, with low precision, recall, and F1-scores across all attack types:

- **DDoS**: Precision 0.32, Recall 0.36, F1-score 0.34

- **Intrusion**: Precision 0.34, Recall 0.30, F1-score 0.32
- **Malware**: Precision 0.32, Recall 0.33, F1-score 0.33

These results indicate that the model struggles to predict cyber-attacks, with a high number of false positives. The low performance may be due to issues like insufficient data or lack of key features. Indeed, the features importance show that all features have an impact of less or equal than 0.1. This value is not high and therefore the features do give enough indication as to which type of attack is happening. There should be more variables added to the dataset to be able to have a pattern that helps predict which attack type it is.

Deployment

The application was created using streamlit.

Libraries used:

- Streamlit: For building the application UI
- Joblib: For model serialization
- Streamlit Cloud: For deployment

Explanation of the syntax and workflow:

The user interface of the **Cyber Attack Detection App** is built using **Streamlit**, a Python framework for creating interactive web applications. The app is designed to provide predictions to users by uploading data and to receive predictions on potential cyber threats.

1. Uploading Data

The application begins by allowing users to upload a CSV file containing data. Once the file is uploaded, the app reads and displays a preview of the dataset to ensure that the correct file has been selected.

2. Making Predictions

After the data is uploaded, the user can click a button to initiate the attack prediction process. The system processes the data using a **pre-trained machine learning model** and predicts the type of cyber attack based on the given input.

3. Saving and Downloading Results

Once the predictions are generated, they are automatically stored in a new CSV file. Users have the option to download this file for further analysis or reporting.

4. Sidebar for Attack Insights

To enhance the user experience, the app includes a **sidebar section** that provides additional insights about different attack types. The sidebar serves as an informative panel, guiding users on potential threats and recommended precautions. This feature can be further expanded to dynamically update based on detected attack patterns.

5. Understanding the Streamlit Syntax

Streamlit provides a declarative syntax for building applications. The key component include:

- **st.title()** and **st.subheader()** for headings.
- **st.file_uploader()** for users to upload CSV files.
- **st.dataframe()** to display data tables.
- **st.button()** to create an actionable button for running predictions.
- **st.success()** for displaying success messages after predictions.
- **st.download_button()** to enable users to download the prediction results.
- **st.sidebar.header()** and **st.sidebar.write()** for adding additional insights on the sidebar.

Streamlit apps ensures that users can interact with machine learning models in a user-friendly way without requiring extensive programming knowledge, making it accessible to both technical and non-technical users.

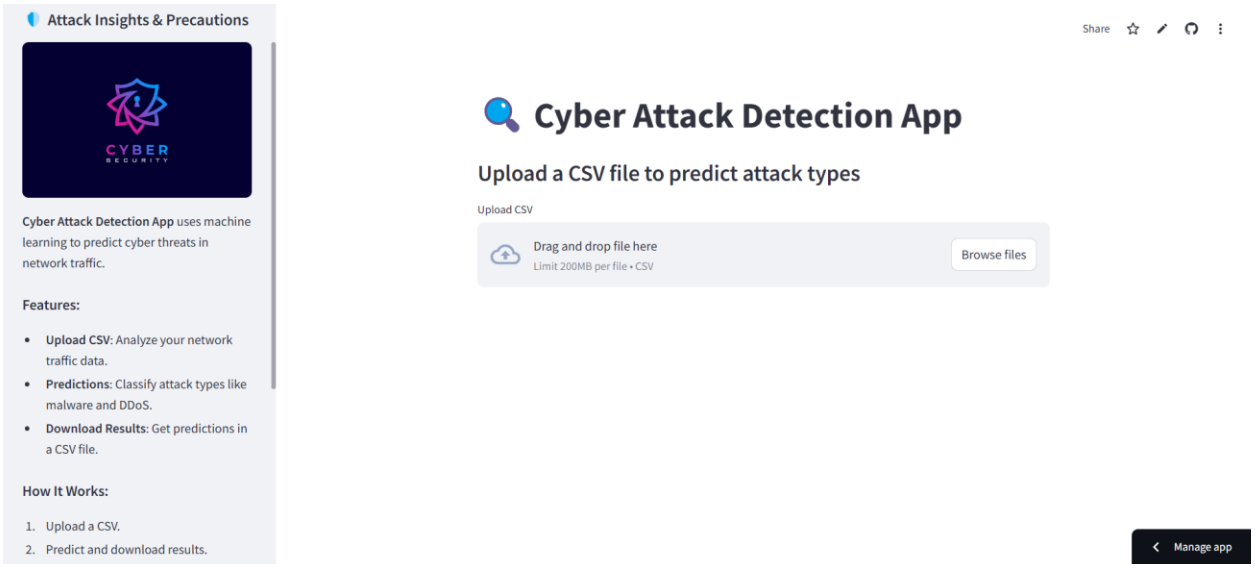
Deployment Process on Streamlit Cloud

1. Commit and push the repository to GitHub.
2. Go to Streamlit cloud and Select the GitHub repo and set the app entry point.
3. Deploy and monitor logs for issues.

This describes the entire lifecycle of the project, from development to deployment.

Snapshot of the app

Application flow link- https://drive.google.com/file/d/173vwTrGB99GHQd1dL7KjQ1JsEeZxAh55/view?usp=sharing



Attack Insights & Precautions

Cyber Attack Detection App uses machine learning to predict cyber threats in network traffic.

Features:

- **Upload CSV**: Analyze your network traffic data.
- **Predictions**: Classify attack types like malware and DDoS.
- **Download Results**: Get predictions in a CSV file.

How It Works:

1. Upload a CSV.
2. Predict and download results.

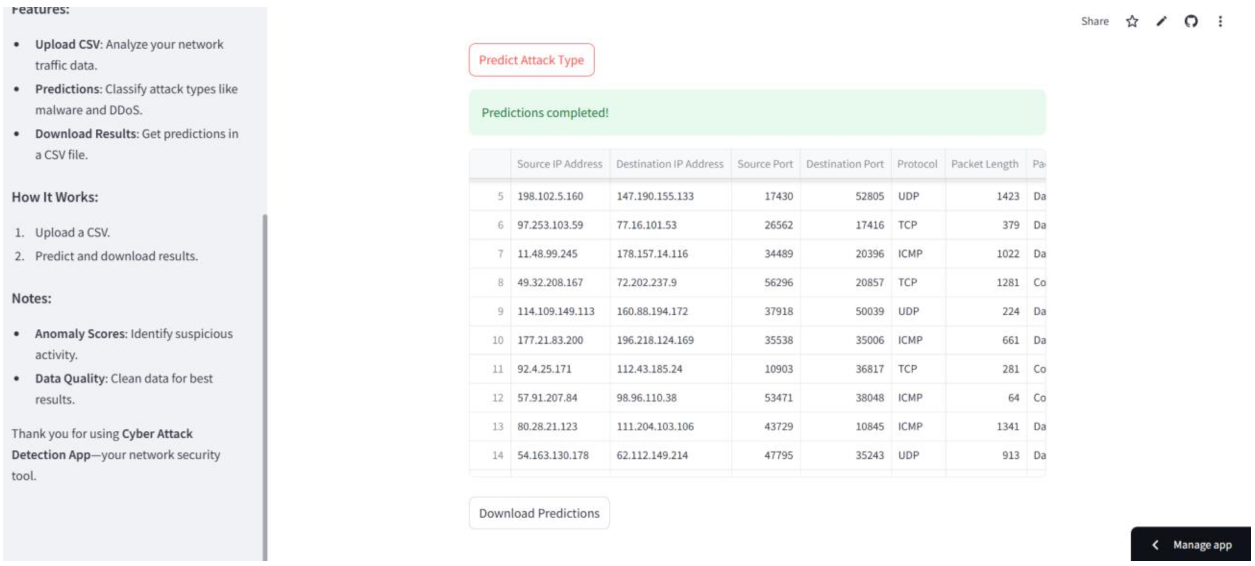
Cyber Attack Detection App

Upload a CSV file to predict attack types

Upload CSV

Drag and drop file here

Limit 200MB per file • CSV



Features:

- **Upload CSV**: Analyze your network traffic data.
- **Predictions**: Classify attack types like malware and DDoS.
- **Download Results**: Get predictions in a CSV file.

How It Works:

1. Upload a CSV.
2. Predict and download results.

Notes:

- **Anomaly Scores**: Identify suspicious activity.
- **Data Quality**: Clean data for best results.

Thank you for using **Cyber Attack Detection App**—your network security tool.

Predictions completed!

	Source IP Address	Destination IP Address	Source Port	Destination Port	Protocol	Packet Length	Pa
5	198.102.5.160	147.190.155.133	17430	52805	UDP	1423	Da
6	97.253.103.59	77.16.101.53	26562	17416	TCP	379	Da
7	11.48.99.245	178.157.14.116	34489	20396	ICMP	1022	Da
8	49.32.208.167	72.202.237.9	56296	20857	TCP	1281	Co
9	114.109.149.113	160.88.194.172	37918	50039	UDP	224	Da
10	177.21.83.200	196.218.124.169	35538	35006	ICMP	661	Da
11	92.4.25.171	112.43.185.24	10903	36817	TCP	281	Co
12	57.91.207.84	98.96.110.38	53471	38048	ICMP	64	Co
13	80.28.21.123	111.204.103.106	43729	10845	ICMP	1341	Da
14	54.163.130.178	62.112.149.214	47795	35243	UDP	913	Da

Conclusion

The most important takeaway from this project is that sometimes data is not robust enough for making predictions. Despite extensive cleaning, feature engineering, and selection, the results of the processed data do not provide enough results to reach a good model accuracy. No correlation and no discernible patterns make it difficult to predict attack types with high accuracy. For a model to learn patterns, it's important to have some recurring information, either the ports, known patterns, or signatures. The highest number of recurrences in such an important variable was found to be 5, which is insufficient for a model to learn a behavior.

In a real-life scenario, this data might need to go through a lot of enrichment, or the volume needs to be increased to discover patterns.

References

Amjad, M., Ahmad, I., Ahmad, M., Wróblewski, P., Kamiński, P., & Amjad, U. (2022). Prediction of pile bearing capacity using XGBoost algorithm: modeling and performance evaluation. *Applied Sciences*, *12*(4), 2126.

Aydin, Z. E., & Ozturk, Z. K. (2021). Performance analysis of XGBoost classifier with missing data. *Manchester Journal of Artificial Intelligence and Applied Sciences (MJAIAS)*, *2*(02), 2021.

Wang, K., Li, M., Cheng, J., Zhou, X., & Li, G. (2022). Research on personal credit risk evaluation based on XGBoost. *Procedia computer science*, *199*, 1128-1135.

Ying, X. (2019, February). An overview of overfitting and its solutions. In *Journal of physics: Conference series* (Vol. 1168, p. 022022). IOP Publishing.

https://docs.streamlit.io/deploy/streamlit-community-cloud/deploy-your-app

Team Contributions Summary

Our team of six collaborated equally:

Jasen: GitHub creation, Modeling
Laura: EDA report, Modeling
Nandini: Streamlit App Development, EDA, Modeling
Osta, Om, Vivian: EDA, Modeling & Fine-Tuning

Appendix A

Figure 1

Frequency of each protocol in the dataset

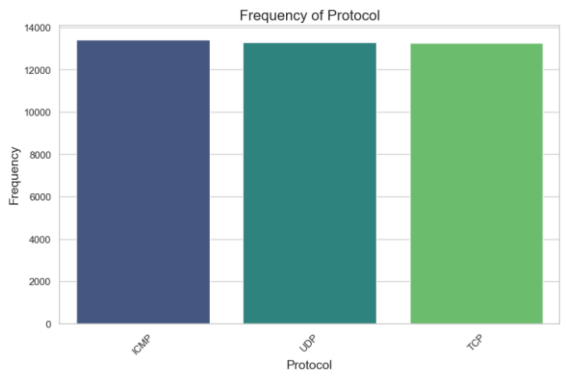


Figure 2

Frequency of the severity levels of the attacks

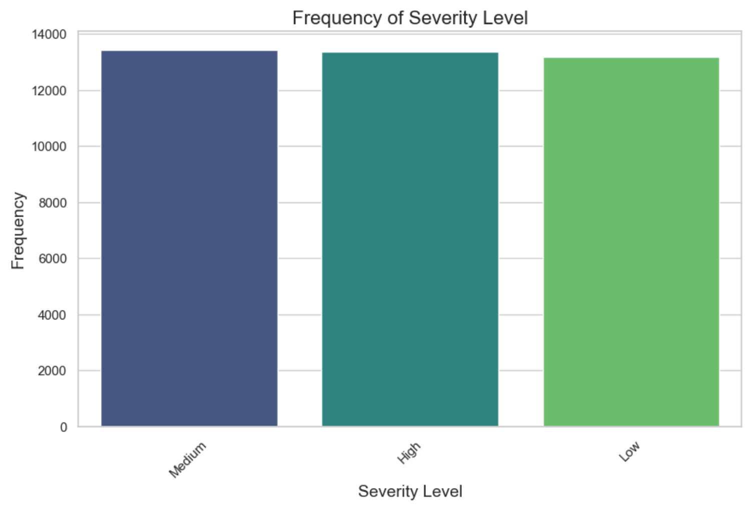
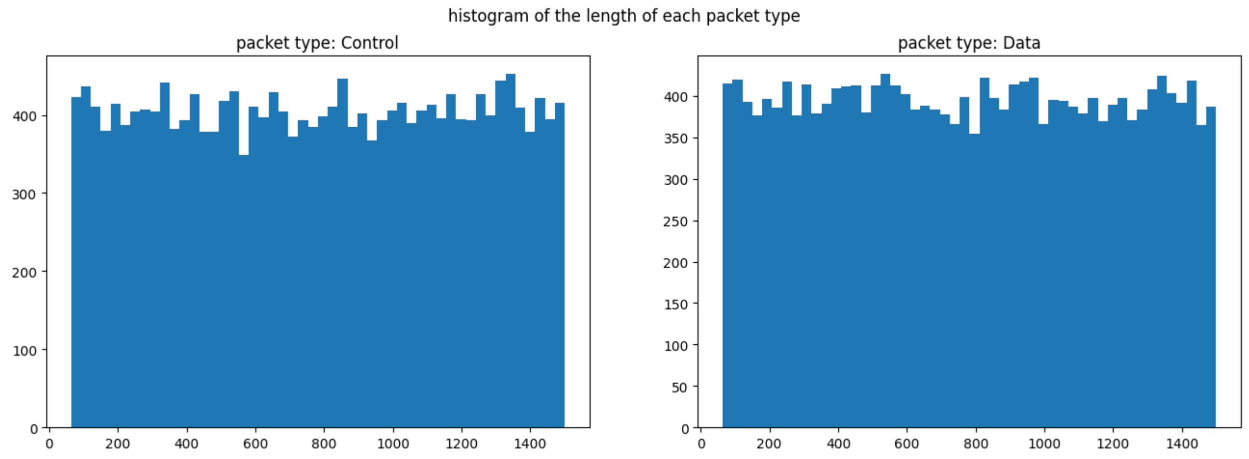


Figure 3

Histogram of the packet length of each packet type


histogram of the length of each packet type
packet type: Control
packet type: Data