Capstone Project

Lewis Parnell

Student ID: 011651581

2025-10-21

Table of Contents

[Part A: Letter of Transmittal 4](#_Toc211933066)

[Part B: Project Proposal Plan 6](#_Toc211933067)

[Project Summary 6](#_Toc211933068)

[Problem Description 6](#_Toc211933069)

[Client and their Requirements 6](#_Toc211933070)

[Deliverables 6](#_Toc211933071)

[Benefits to Client 6](#_Toc211933072)

[Data Summary 6](#_Toc211933073)

[Data Source 6](#_Toc211933074)

[Data Processing and Management 6](#_Toc211933075)

[Relevance of Data 7](#_Toc211933076)

[Legal and Ethical Concerns 7](#_Toc211933077)

[Implementation 7](#_Toc211933078)

[Timeline 8](#_Toc211933079)

[Evaluation Plan 8](#_Toc211933080)

[Costs 9](#_Toc211933081)

[Part C: Application 11](#_Toc211933082)

[Part D: Post-implementation Report 12](#_Toc211933083)

[Solution Summary 12](#_Toc211933084)

[Data Summary 12](#_Toc211933085)

[Machine Learning 13](#_Toc211933086)

[Method 13](#_Toc211933087)

[Development 13](#_Toc211933088)

[Justification 13](#_Toc211933089)

[Validation 14](#_Toc211933090)

[Visualizations 15](#_Toc211933091)

[Class Distribution 15](#_Toc211933092)

[Correlation Heatmap 16](#_Toc211933093)

[User Guide 18](#_Toc211933094)

[uv Installation and Repository Cloning 18](#_Toc211933095)

[Download Dataset and Data Preprocessing 18](#_Toc211933096)

[Model Training 18](#_Toc211933097)

[Sample Generation 19](#_Toc211933098)

[Inference 19](#_Toc211933099)

[Model Evaluation 19](#_Toc211933100)

[References 20](#_Toc211933101)

# Part A: Letter of Transmittal

October 20, 2025

Senior Leading Review Committee, XYZ Hospitals, 1234 Address St., Townsville, WA

Lewis Parnell, Machine Learning Engineer, [lewis.parnell@email.com](mailto:lewis.parnell@email.com), (123) 456-7890

Subject: Proposal for an Enhanced Intrusion Detection System (IDS) based on Advanced Machine Learning

Dear Senior Leadership Review Committee,

This letter accompanies the formal proposal for developing and implementing a new intrusion detection system (IDS) to enhance your organization’s cybersecurity defenses significantly. This critical security initiative aims to secure organizational approval and resource allocation.

The core problem that your organization faces is a faulty intrusion detection system (IDS). The current IDS fires many false alarms, which fatigue cybersecurity staff and delay their responses to real, genuine threats. This misbehavior is caused by the current IDS’s primitive implementation, using basic anomaly-based detection methods that produce a high rate of false positives, wherein they alert on benign but unusual traffic. This inefficiency burdens security operations and drives operational costs due to time wasted on non-threats. Critically, it leaves the organization vulnerable to high-impact breaches and operational downtime. The existing system is not sufficiently accurate for a modern threat environment, necessitating an upgrade.

The proposed solution involves developing a novel IDS built around the XGBoost algorithm, the premier algorithm for this machine learning task. The XGBoost algorithm is an optimized distributed gradient boosting library designed to handle classification tasks quickly and accurately. It is well-known for its exceptional performance, in both speed and accuracy, on tabular datasets. The data involved in networking is naturally stored in a table, like an Excel spreadsheet, making this algorithm especially well-suited for determining hostile traffic. In speed and accuracy, XGBoost performs well against deep learning methods on this task domain. Finally, XGBoost will outperform large language models by classifying hostile traffic faster and more accurately than these other model architectures.

The CICIDS2017 dataset will serve as the basis for this project. This is a dataset created by security researchers to develop their field. It is a well-used dataset that corrects many flaws found in previous IDS datasets. The dataset contains a breadth of attack types, making any model trained from it much more capable of accurately recognizing a wide range of attack traffic patterns.

The costs associated are the salary of a machine learning engineer and the necessary computational infrastructure.

A four-week implementation plan is proposed to ensure a stable and effective deployment. The first week will be spent on data preparation and exploration. The second week will cover the core of model development and training. The third week will be spent on performance tuning of the developed model. The final week will cover the testing and evaluation of the model with a suite of metrics.

My professional background provides the necessary experience to oversee this project successfully. Through my employment as a machine learning engineer working on many similar projects, I do not foresee significant challenges or concerns regarding the development of this IDS solution.

This project offers a necessary upgrade to your intrusion detection capabilities. I look forward to discussing the enclosed full proposal and the potential of this application to secure your organization’s IT infrastructure.

Sincerely,

Lewis Parnell, Machine Learning Engineer

# Part B: Project Proposal Plan

## Project Summary

### Problem Description

The problem is that the current IDS is overly sensitive and raises too many false alarms, which results in alarm fatigue. This alarm fatigue results in slower response and unnecessary work as cybersecurity team members must determine if an incident represents a genuine threat to the organization.

### Client and their Requirements

The client is XYZ Hospitals, a large hospital in Townsville, WA. Due to the sensitive nature of information stored on their network, XYZ Hospital requires an on-site solution for IDS.

### Deliverables

The items created by this project are a repository that can train a configurable IDS model, analyze the data used to train the model, evaluate the model's performance with a set of metrics, and run inference over traffic flows to perform intrusion detection.

### Benefits to Client

The benefits to the client are that the client acquires a private and secure IDS that performs efficiently and effectively, detecting various attack types.

## Data Summary

### Data Source

The data will be sourced from the CICIDS2017 dataset (Sharafaldin et al., 2018), available at <https://www.unb.ca/cic/datasets/ids-2017.html>. This dataset will provide the project with all its data; no data will be collected or simulated.

### Data Processing and Management

The data will be processed to remove values that prevent it from being effectively used by the XGBoost model. This includes infinite values, a known problem with this dataset, and any other rows with missing values. Columns will have whitespace stripped.

Since the dataset does not contain personally identifying or sensitive information, there is no need to encrypt or perform any extra security measures on the data.

### Relevance of Data

The dataset contains traffic patterns that are difficult to collect from a real-world environment. By using the CICIDS2017 dataset, the resulting model will be able to classify many more attacks.

Due to the domain context of an IDS dataset, outliers represent essential information that the model must learn from. Therefore, outliers are not explicitly removed from the dataset. Additionally, the XGBoost algorithm, due to its design, is robust to outliers. Consequently, eliminating outliers would be less necessary even if outliers were not crucial to the model’s performance in this context.

### Legal and Ethical Concerns

The only legal concern regarding this dataset is that the authors do not specify commercial use. The authors describe the dataset as being made available for researchers. When downloading the dataset, researchers must list their name, contact information, organization/company, job title, and country. It could be inferred that the authors are OK with commercial use, considering that they have not indicated otherwise and have made no attempt to dissuade such usage.

There are no clear and immediate ethical concerns with this dataset. The authors constructed a simulated network and performed controlled attacks against hardware they owned and operated. No unconsenting individuals were involved in creating this dataset, and the dataset does not contain any personally identifying or sensitive information.

## Implementation

The industry standard methodology to be used in this project is CRISP-DM. During business understanding, project objectives will be clarified, and requirements will be translated into the problem definition. Machine learning effort will be aligned with business goals. During the data understanding phase, initial data will be collected (the dataset will be downloaded), and the dataset will be explored to identify quality issues and form initial hypotheses. It will be cleaned and integrated into the project’s codebase to prepare the data. During the modeling phase, the XGBoost algorithm will be applied to the data, and a model will be learned from the dataset. After this phase, the model will undergo an evaluation phase, wherein it will be determined if model settings must be tuned (hyperparameter tuning, for instance), and the performance will be evaluated using a suite of metrics. Finally, in the deployment phase, a command-line interface will be constructed that allows end-users to make inferences on traffic flows using the model.

## Timeline

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Milestone** | **Dependencies** | **Resources** | **Start and End Date** | **Duration** |
| **Environment and Data Acquisition** | NA | Computer, Python environment, Internet access | 2026-01-05 to 2026-01-05 | 1 day |
| **EDA & Cleaning** | Milestone 1 completed: successful dataset downloaded | VS Code, Pandas, NumPy, Matplotlib, Seaborn | 2026-01-06 to 2026-01-06 | 1 day |
| **Preprocessing and Feature Engineering** | Milestone 2 completed: the identification of missing values, irrelevant features, and data types | Scikit-learn Standard Scalar & LabelEncoder, SMOTE | 2026-01-07 to 2026-01-09 | 3 days |
| **XGBoost Model Training** | Milestone 3 completed: preprocessing completed | XGBoost, CPU/GPU compute resources | 2026-01-12 to 2026-01-16 | 5 days |
| **Final Model Training and Evaluation** | Milestone 4 completed: XGBoost model trained | XGBoost model, evaluation scripts | 2026-01-19 to 2026-01-23 | 5 days |
| **Documentation and Conclusion** | Milestone 5 completed: final model trained and evaluated | Microsoft Word | 2026-01-26 to 2026-01-30 | 5 days |

## Evaluation Plan

The verification methods for each stage of CRISP-DM applied to this project follow:

In business understanding, the key deliverables are a project charter and success criteria. This means that project stakeholders have reviewed and signed off on a document that aligns the business problem with the proposed ML task (in this case, multi-class classification). In data understanding, the key deliverable will be a data quality report generated through exploratory data analysis scripts and visualizations. These artifacts will help guide the machine learning engineer and confirm data structure, type, distribution, and correlation. Verification for this stage passes if data quality is deemed sufficient.

The key deliverable will be a generated and processed dataset during data preparation. This dataset will be created through a transformation pipeline that processes the data to remove unusable values and scales, encodes the data to work better with the XGBoost algorithm, and splits the dataset into train, test, and validation subsets. Verification for this phase passes if the dataset is partitioned and ready for model input.

During the modeling phase, the key deliverable will be a trained model. This model will be created from the training set, and its performance will be evaluated with the validation to tune the model’s hyperparameters. The performance on the testing set will be used to determine if verification for this phase passes.

The key deliverable will be a final performance report during the evaluation phase. This report will contain the results of the metrics used for this machine learning project. Verification for this stage passes if the model performs to the desired standards.

During the deployment phase, the key deliverable is a command-line interface. This CLI allows end-users to use the model in inference mode to detect hostile traffic from traffic flows. Verification for this stage will pass if such a CLI is provided.

The validation for the project, to be used upon completion, involves evaluating the project with the test set and acquiring the technical performance metrics. These metrics are accuracy, precision, recall, and F1-score.

## Costs

Hardware costs are limited to the computer on which the machine learning engineers work. A suitable MacBook Air costs approximately $1,500. A desktop computer with a GPU is roughly $2,000. Both of these hardware devices have variable costs depending on configuration.

The software costs for this project are $0; all software used is free of charge. This includes the IDE (Visual Studio Code) and the libraries used in the project.

The estimated environmental costs of the application are: $2000 for the server that hosts the model for inference, $200 for the networking hardware that connects the server to the hospital network, and around $500 for the one-time labor cost of paying a professional consultant to install the specially configured server and its hardware. That one-time cost could be ignored if the hospital has its own IT staff.

# Part C: Application

<https://github.com/lewisparnellofficial/capstone-simple>

# Part D: Post-implementation Report

## Solution Summary

The problem was that XYZ Hospitals’ existing IDS generated too many false alarms. This caused alarm fatigue and delayed response to genuine threats. The solution was a new IDS created using the XGBoost algorithm. This algorithm was used to make a classifier on the CICIDS2017 dataset, achieving high accuracy. The final application solved the problem by providing a command-line tool that allowed the client to analyze their network traffic flows. The application used the trained model to classify the traffic and would output a summary report. This summary report identified specific detected threats, which reduced false positives and improved security.

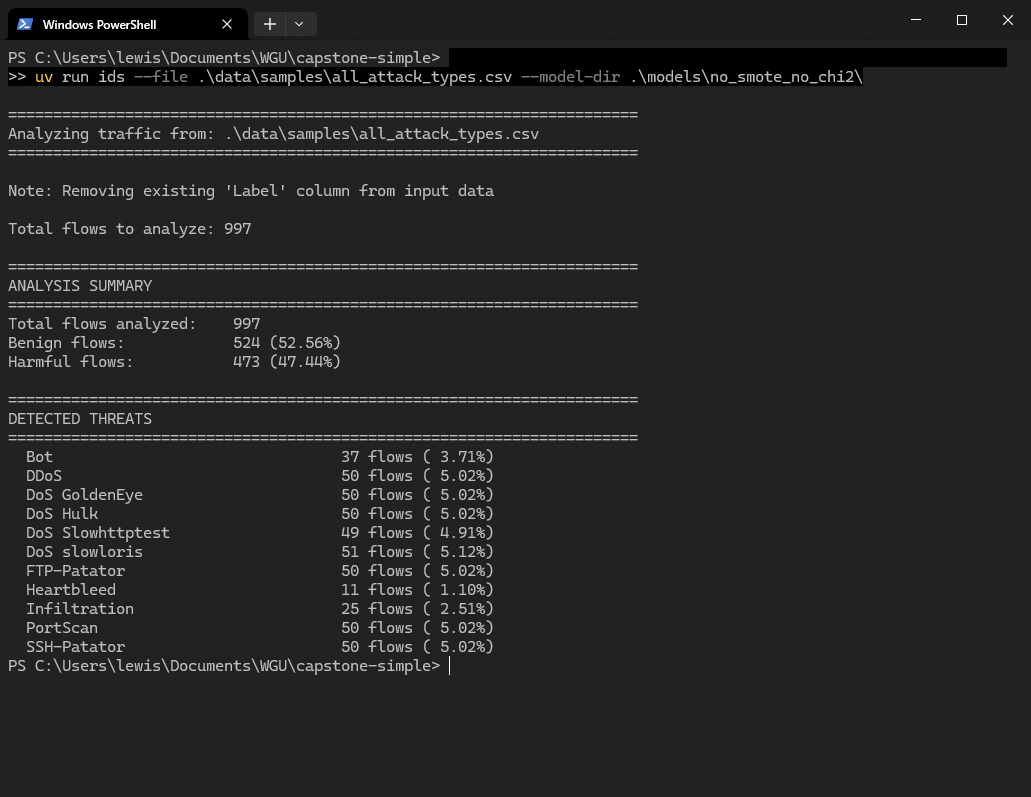
## Data Summary

The source of the data was the CICIDS2017 dataset. (Sharafaldin et al., 2018). The data was processed to remove values that prevented its use with the XGBoost model, such as infinite values and rows with missing data. Column names also had whitespace stripped. Throughout the development cycle, the data was managed without special encryption or extra security measures, as it was publicly available, did not contain personally identifiable information, and involved no unconsenting individuals.

## Machine Learning

### Method

The machine learning method was the XGBoost algorithm, applied to an intrusion detection classification problem.



The screenshot above illustrates how the model created by the project can be used. With the model created by the project, users could input formatted traffic flows and receive a report of the hostile traffic detected.

### Development

The model was developed by analyzing the data, determining how XGBoost required the data to be formatted, and then formatting the data to match the “shape” requirements. For instance, the data could not have infinite values, so those values were removed.

### Justification

These development decisions were based on the requirements of an effective IDS. These requirements mandate high accuracy and a strong capability to detect rare attack types. The algorithm chosen performs well on multiclass classification, which the IDS task typically involves. The algorithm chosen was also known for its efficiency and robustness in scaling and distribution of features.

Significantly, the rejection of two standard preprocessing techniques, SMOTE and Chi-Squared feature selection, was based on comparative, empirical results. The model trained without either method performed better than models that used one technique, and the model that used both methods experienced the most significant performance degradation. Due to the speed boost of GPU acceleration, the decision was made not to use Chi-Squared in the final model, since the purpose of feature selection is to enable the training and use of models that would otherwise struggle with hardware limitations. Although the dataset exhibited severe class imbalance, the model trained with SMOTE demonstrated degraded performance despite balancing the training set to 6.67% per class.

## Validation

The model is a supervised machine learning model, which uses the labeled dataset to learn a multi-class classifier. The metrics chosen for this model were accuracy, precision, recall, and F1-score.

Accuracy is the proportion of total predictions that were correct across all classes. Accuracy measures the overall effectiveness of the classifier. Precision is the proportion of positive predictions that were actually correct. Precision measures the quality of a positive prediction. Recall is the proportion of actual positive instances that were correctly identified. Recall measures the model’s ability to find all positive samples. Finally, the F1-score is the harmonic mean of precision and recall. F1-score balances both metrics, which is helpful in imbalanced datasets.

Table 1 - Evaluation Metric Comparison

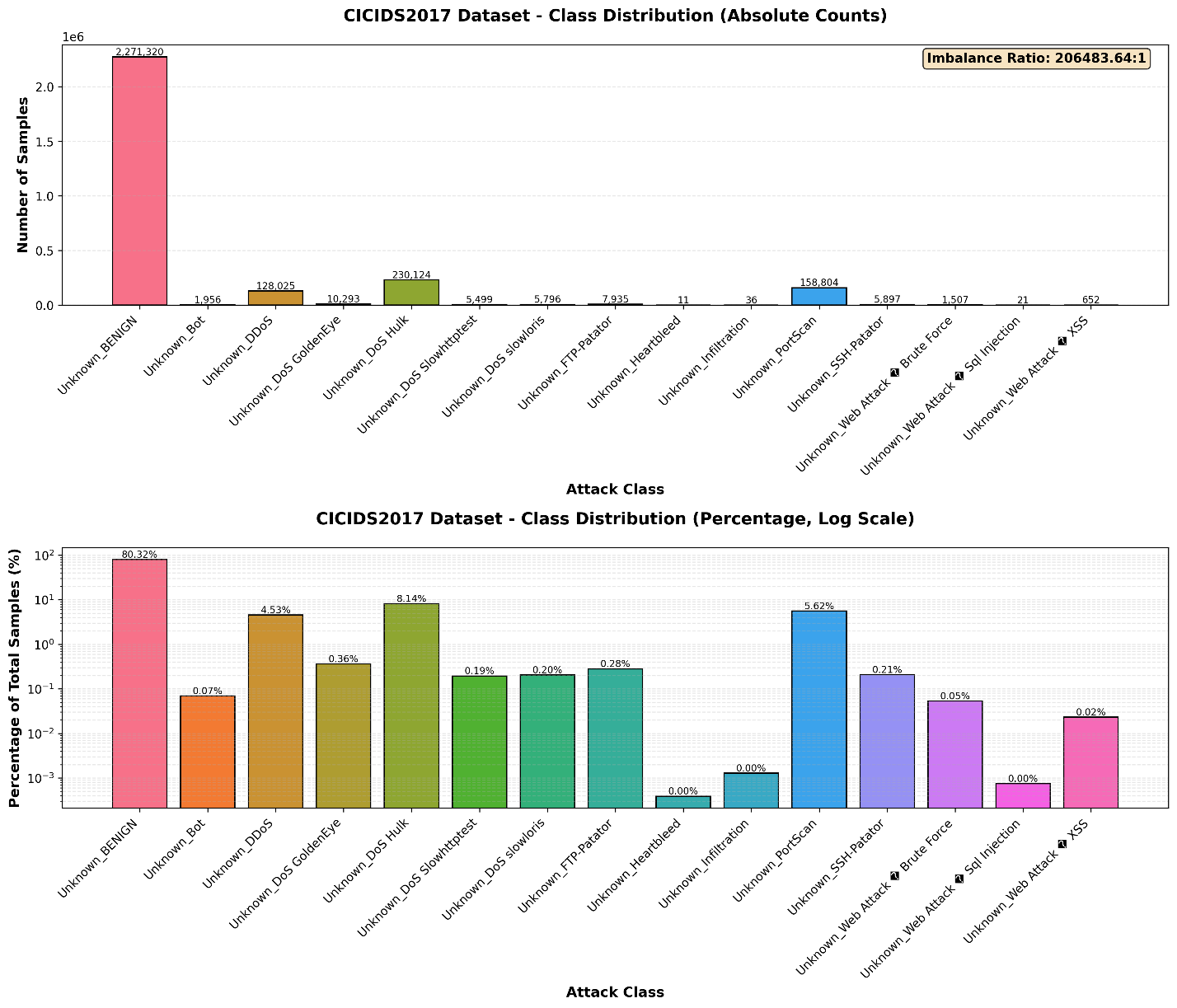
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Configuration** | | **Accuracy** | **Precision** | **Recall** | **F1-Score** |
| SMOTE | Chi2 | 90.56% | 97.85% | 90.56% | 93.73% |
| SMOTE | No Chi2 | 99.51% | 99.75% | 99.51% | 99.61% |
| No SMOTE | Chi2 | 98.43% | 98.49% | 98.43% | 98.40% |
| No SMOTE | No Chi2 | 99.88% | 99.87% | 99.88% | 99.87% |

Precision, recall, and F1-score are the weighted averages, accounting for the class imbalance that the dataset exhibited. These results show that the optimization techniques chosen for this project did not have their intended effect. While Chi-Squared alone did not degrade performance significantly, the combination of SMOTE and Chi-Squared did.

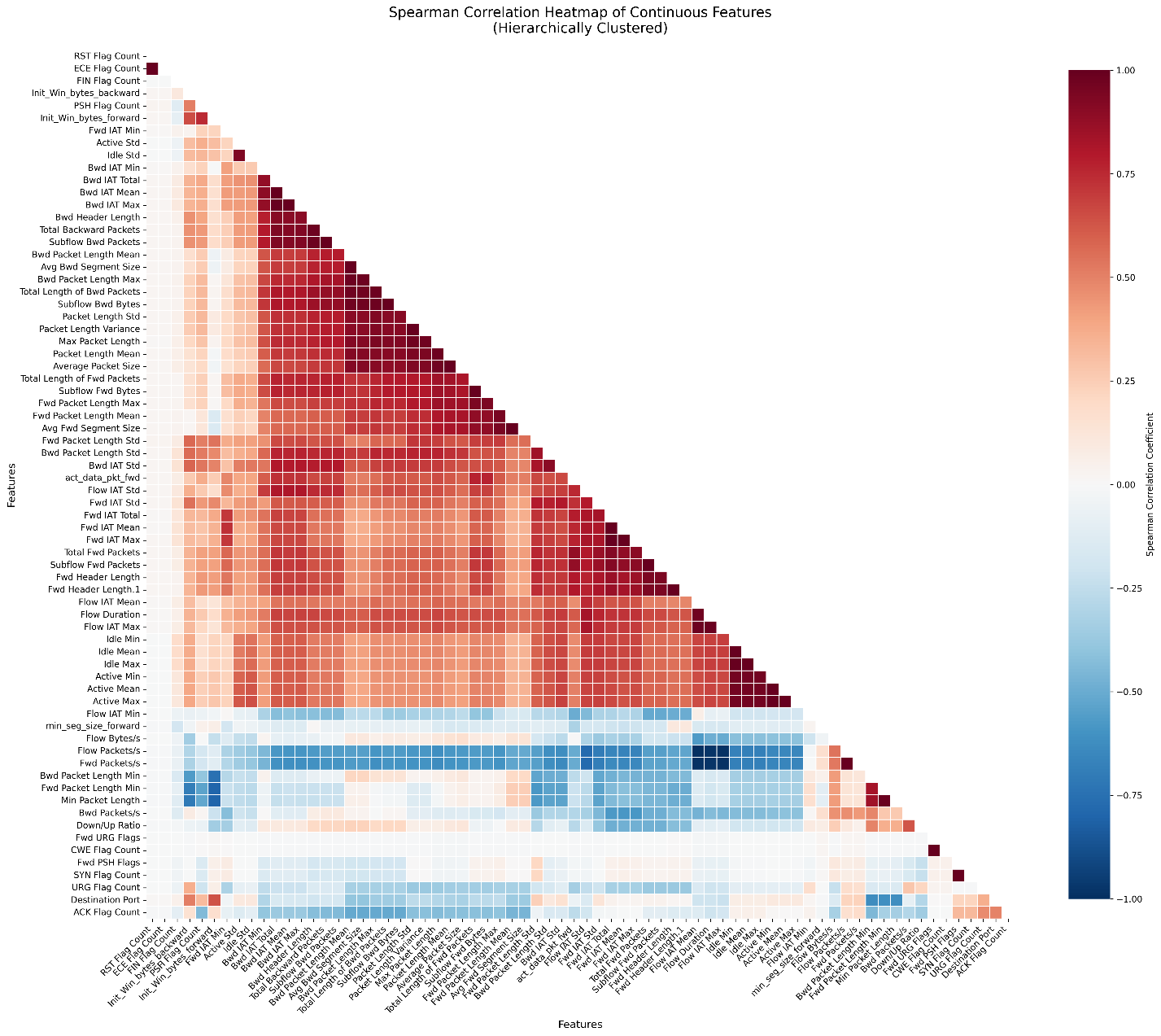
## Visualizations

Visualizations can be found in outputs/plots, saved as class\_distribution.png, correlation\_heatmap.png, and violin\_plot\_top\_features.png. They are included here for convenience.

### Class Distribution



### Correlation Heatmap



## Violin Plot of Top FeaturesA chart of different shapes AI-generated content may be incorrect.

## User Guide

This project requires git and uv. uv is an all-in-one Python development tool that combines many different tools into one package.

### uv Installation and Repository Cloning

To install uv, type:

powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/install.ps1 | iex"

into PowerShell. You will need to restart PowerShell after installing uv, or input the command that uv provides, to add uv to your path. Clone the repository:

git clone <https://github.com/lewisparnellofficial/capstone-simple>

cd capstone-simple

uv sync

uv will download all of the requirements.

### Download Dataset and Data Preprocessing

Once completed, download the dataset by running the following:

uv run ids-download

After the dataset has been downloaded, preprocess it with this command:

uv run ids-preprocess

Now you can train the model if desired. Alternatively, you can simply test inference. To see how, skip to the Inference section.

### Model Training

The training command has several optional flags of interest. Note, there are already four different trained models in the models folder. Regardless, if you wish to train a model, this is the command:

uv run ids-train --no-smote --no-chi2 --no-gpu --output-dir .\models\no\_smote\_no\_chi2\

The option “--no-smote” turns off the SMOTE training feature.

The option “--no-chi2” turns off using chi-squared for feature selection.

The option “--no-gpu” turns off GPU acceleration and should be used if you do not have a GPU or CUDA installed. Enabling GPU acceleration will drastically reduce training time.

The option “--output-dir” determines where the model will be saved.

### Sample Generation

To generate a sample traffic pattern from the dataset, use this command:

uv run ids-sample-gen --scenario under\_attack --output data/samples/under\_attack.csv

To see a list of scenarios available, use the “--list-scenarios” flag.

### Inference

To test inference, you can run this command, using a bundled sample traffic flow:

uv run ids --file data/samples/test\_small.csv

To test a specific model:

uv run ids --file data/samples/test\_small.csv --model-dir models/no\_smote\_no\_chi2

### Model Evaluation

You can also evaluate the model to see its performance metrics by running:

uv run ids-evaluate

To evaluate a specific model:

uv run ids-evaluate --model-dir .\models\no\_smote\_no\_chi2\

# References

Sharafaldin, I., Habibi Lashkari, A., & Ghorbani, A. A. (2018). Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization: *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, 108–116. https://doi.org/10.5220/0006639801080116