Capstone Project

Lewis Parnell

Table of Contents

[Part A: Letter of Transmittal 4](#_Toc211893582)

[Part B: Project Proposal Plan 6](#_Toc211893583)

[Project Summary 6](#_Toc211893584)

[Problem Description 6](#_Toc211893585)

[Client and their Requirements 6](#_Toc211893586)

[Deliverables 6](#_Toc211893587)

[Benefits to Client 6](#_Toc211893588)

[Data Summary 6](#_Toc211893589)

[Data Source 6](#_Toc211893590)

[Data Processing and Management 6](#_Toc211893591)

[Relevance of Data 7](#_Toc211893592)

[Legal and Ethical Concerns 7](#_Toc211893593)

[Implementation 7](#_Toc211893594)

[Timeline 8](#_Toc211893595)

[Evaluation Plan 8](#_Toc211893596)

[Costs 9](#_Toc211893597)

[Part C: Application 10](#_Toc211893598)

[Part D: Post-implementation Report 11](#_Toc211893599)

[Solution Summary 11](#_Toc211893600)

[Data Summary 11](#_Toc211893601)

[Machine Learning 11](#_Toc211893602)

[Validation 12](#_Toc211893603)

[Visualizations 13](#_Toc211893604)

[Class Distribution 13](#_Toc211893605)

[Correlation Heatmap 14](#_Toc211893606)

[User Guide 16](#_Toc211893607)

[References 17](#_Toc211893608)

# 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 Enhanced Intrusion Detection System (IDS) based on Advanced Machine Learning

Dear Senior Leadership Review Committee,

This letter accompanies the formal proposal for the development and implementation of a new intrusion detection system (IDS) aimed at significantly enhancing your organization’s current cyber security defenses. The objective is to secure organizational approval and necessary resource allocation to proceed with this critical security initiative.

The core problem that your organization faces is a faulty intrusion detection system (IDS). The current IDS fires many false alarms, that fatigue cybersecurity staff and delay their responses to real, genuine threats. This misbehavior is caused by the current IDS’ 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 up 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 solution that I propose involves developing a novel IDS built around the XGBoost algorithm, the premier algorithm for this kind of machine learning task. The XGBoost algorithm is an optimized distributed gradient boosting library designed to be fast and accurate on classification tasks. 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 the task of determining hostile traffic. XGBoost additionally performs well against deep learning methods on this task domain, again, in both speed and accuracy. Finally, XGBoost will also outperform large language models, by classifying hostile traffic faster and more accurately then these other model architectures.

The CICIDS2017 dataset will serve as the basis for this project. This is a dataset created by security researchers for the development of their field, and is a well-used dataset that corrects many of the 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, which could simply be a MacBook Air.

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 represent 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 repo that can train a configurable IDS model, analyze the data used to train the model, evaluate the performance of the model 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 detects a variety of 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 of its data, and 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 the data, or perform any kind of extra security 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, has a degree of robustness to outliers. Consequently it would be, even if in this context, outliers were not crucial to the model’s performance, less necessary to remove them.

### Legal and Ethical Concerns

The only legal concern regarding this dataset is that commercial use is not specified by the authors. The authors describe the dataset as being made available for researchers. When downloading the dataset, researchers must list their name, contact information, their 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. In the context of this assignment’s roleplay, it would be a potential legal concern.

There are no clear and immediate ethical concerns with this dataset. The author’s constructed a simulated network and performed controlled attacks against hardware that they owned and operated. No unconsenting individuals were involved in the process of 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. To prepare the data, it will be cleaned and integrated into the project’s codebase. 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 perform inference 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 follows:

In business understanding, the key deliverable is 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.

During data preparation, the key deliverable will be a generated and processed dataset. This dataset will be created through a transformation pipeline that processes the data to remove unusable values, scale and encode the data to work better with the XGBoost algorithm, and split 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.

During the evaluation phase, the key deliverable will be a final performance report. 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 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 is evaluating the project with the test set and acquiring the technical performance metrics. These metrics are accuracy, precision, recall, and F1-score. The threshold of these metrics will be determined by

## Costs

Hardware costs are limited to the computer that the machine learning engineers are working on. 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 in this project is free of charge. This includes the IDE used (Visual Studio Code), along with the libraries used in the project.

The time I’m going to have to return to after I finish figuring out what the timeline for this project is.

# Part C: Application

# Part D: Post-implementation Report

## Solution Summary

The problem was a multi-class classification task in the domain of intrusion detection. The solution to this problem was an XGBoost classifier

## Data Summary

The source of the data is the CICIDS2017 dataset (Sharafaldin et al., 2018).

## Machine Learning

The model uses the XGBoost algorithm to perform multi-class classification on traffic flows. The output of the model is a probability distribution of attack types on the traffic flow(s). XGBoost provides gradient boosting of decision trees.

The dataset was processed to remove infinite values, a known problem with the dataset. It was also processed to remove empty values. One-hot encoding was used to convert categorical features into binary vectors, and normalization was applied. The dataset was split into three subsets: a training set (80% of the dataset), a testing set (10%), and a validation set (10%). The training set had feature selection applied using Spearman’s correlation coefficient.

The XGBoost algorithm was chosen because it provides high performance and accuracy on tabular data classification tasks. The algorithm was selected from over a deep neural network because of both its lower training time and reduced computational resource demands, which would allow for more efficient deployment in a production environment. Infinite value clamping was applied to mitigate the effect of infinite values in the dataset. Empty values were removed since the dataset contains enough values that it is meaningless to impute the empty values. Hyperparameter optimization was explored to determine if a worthwhile accuracy and other metric gains could be achieved, but the base hyperparameters proved to be competitive. In fact, while tuning the hyperparameter optimization search function (provided by the Optuna framework), the metrics experienced a regression. This was because to reduce the hyperparameter optimization search time (“study” time) to a functional duration for model development (lower than several hours), the n\_estimators hyperparameter, which controls the number of boosting rounds, was decreased to reduce overall training and study time. While the study time was cut down to approximately 2 hours, the resulting model only had an accuracy of 92%, representing a regression from

1. XGBoost algorithm chosen because of tabular data classification performance
   1. Part of this performance is the fact that XGBoost does not require GPU for training or inference
   2. Deep neural networks do not perform substantially better than XGBoost on these kinds of datasets
2. Data preprocessing:
   1. Infinite value clamping
   2. Missing value dropping rather than imputation, since there were enough values just to drop them and ignore.
   3. Chi-squared test to help reduce the overall features by selecting those that were not excessively collinear/correlated. The point of this processing step was to help reduce the overall training time that the model would have to undergo.
   4. SMOTE to help deal with the severe class imbalance of IDS data.
   5. Hyperparameter optimization search was applied during model refinement but dropped after results showed an unworthwhile performance improvement.

## 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. These metrics were derived from the confusion matrix, which tabulates the outcomes of predictions against the actual class labels.

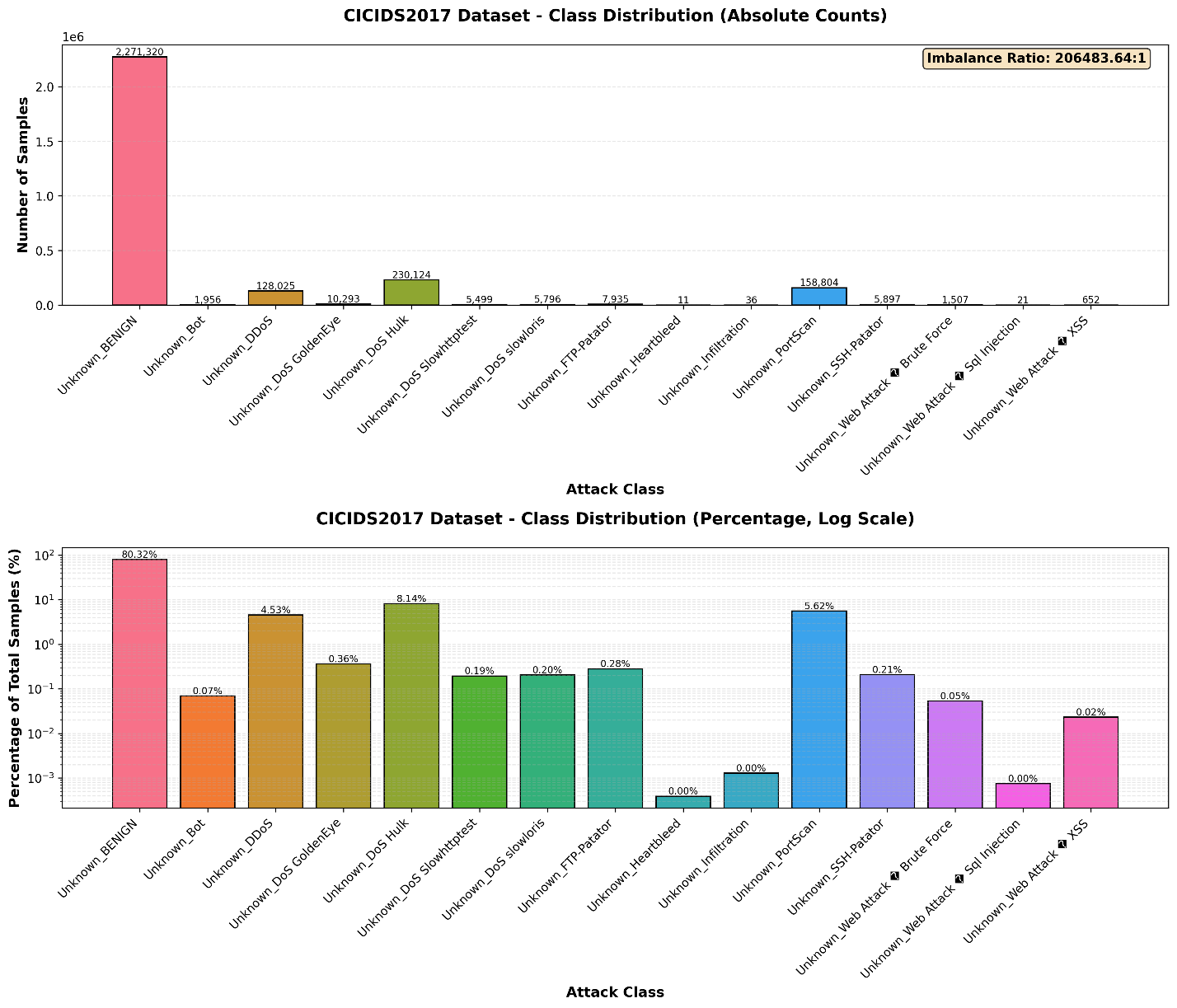
Accuracy is the proportion of total predictions that were correct across all classes. Accuracy measures the overall effectiveness of the classifier. Precision is the proportion of positive predictions that were actually correct. Precision measures the quality of a positive prediction. Recall is the proportion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

[Here should go a table that details the results. I’m considering making a table that shows the results of no SMOTE and no Chi2, SMOTE and no Chi2, no SMOTE and Chi2, and SMOTE and Chi2]

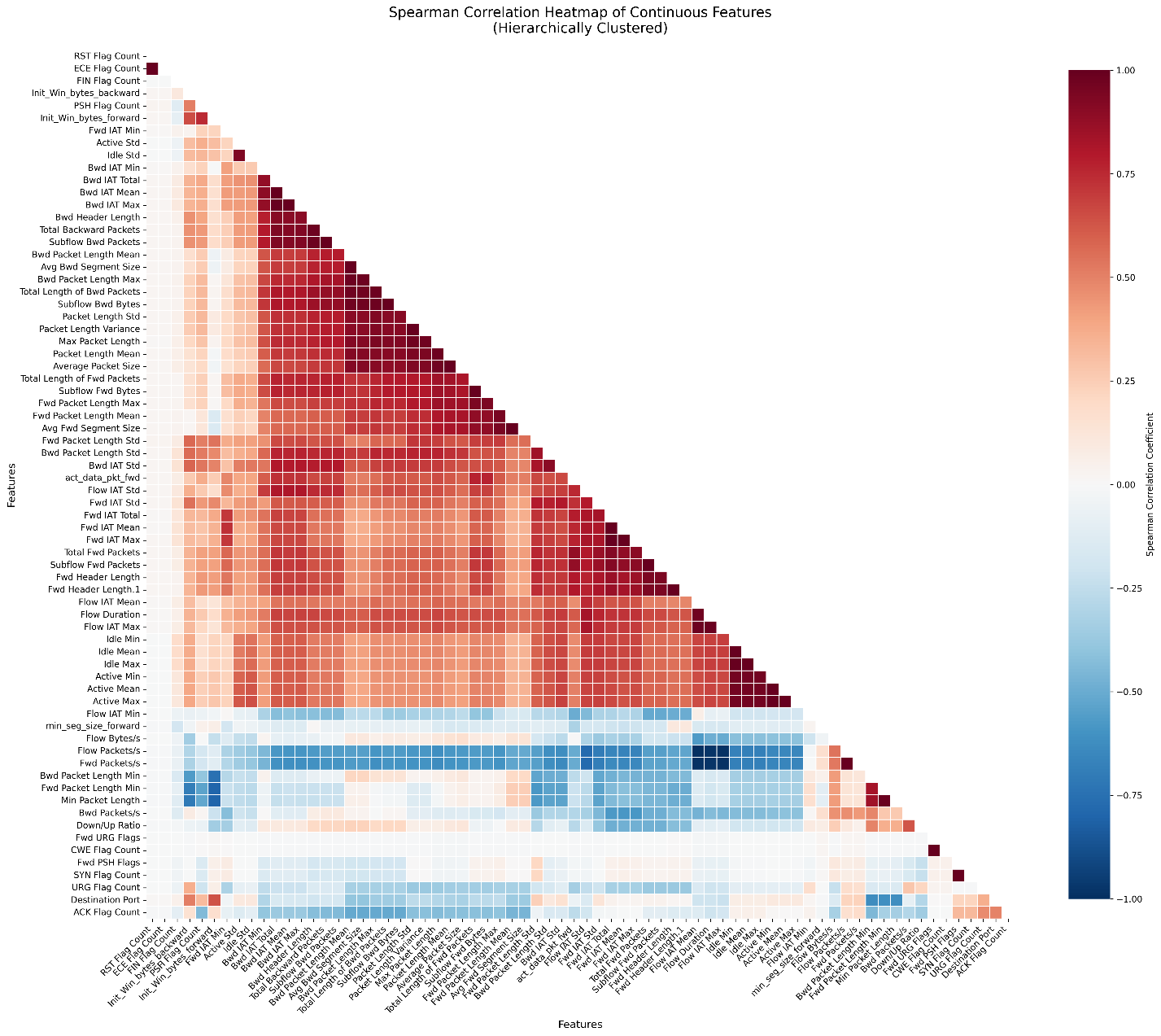
## Visualizations

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 a lot of different tools into one package.

To install uv, type:

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

into PowerShell or git bash. Clone the repository:

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

cd capstone-simple

uv sync

uv will download all of the requirements. 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. The training command has several optional flags of interest.

uv run ids-train --no-smote --no-chi2 --no-gpu

The option “--no-smote” disables the SMOTE training feature. Using SMOTE has a noticeable effect on training time, and disabling it will increase training speed.

The option “--no-chi2” disables the use of chi-squared for feature selection. Using chi-squared seems to have an insignificant effect on training time.

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

Once you have finished training the model, you can generate some sample traffic and then test inference:

uv run ids-sample-gen --scenario small\_test --output data/samples/test\_small.csv

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

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

uv run ids-evaluate

# 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