Faculty of Computer Science

Software Proposal Document for Machine Learning & Deep Learning Intrusion Detection Workbench System

Mohab Sameh Ibrahim
Supervised by: Dr. Ayman Ezzat & Eng. Ahmed Neil

March 28, 2021

Abstract

The main idea of this project is to provide an all-in-one graphical-based workbench system for machine learning and deep learning for anomaly-based intrusion detection systems development with detailed benchmarking and live packet capture classification capabilities.

1 Introduction

1.1 Background

Within the past decade, people, organizations, and businesses have been exponentially becoming more dependant on network-attached devices. Consequently, with such dependence, the need to secure network attached devices has been more important than ever before; hence, the growth of the field of network security and the increase in organizational resources allocation to network security.

Among other interests, network security strives to prevent malicious attacks to occur on organizational or personal networks, which usually leads to disastrous outcomes to such organizations, industrial entities, governments, or individual privacy, hence the development of Intrusion Detection Systems (IDSs). Intrusion Detection Systems are hardware or software systems that are deployed within an environment to detect the occurrence of malicious attacks. Consequently, IDSs holds critical importance within a network's security infrastructure and are subject to increasing development and improvement demand. Intrusion Detection Systems are mainly categorized into two main categories, Host-based Intrusion detection Systems (HIDS)

and Network-based Intrusion Detection Systems (NIDS). Host-based Intrusion Detection Systems rely on being deployed within the network end-points, such as the users' devices, usually as a software solution; while Network-based Intrusion Detection Systems rely on being directly attached to the network's infrastructure, such as critical network gateways or routing and switching devices, usually as a hardware solution. Both categories of IDSs detect malicious attacks using two different detection mechanisms: Signature-Based Detection and Anomaly-Based Detection [8] [5]. Signature-Based detection relies on detecting a predefined malicious signature which represents a specific previously detected action set. On the other hand, Anomaly-Based detection relies on utilizing intelligent approaches to define what shall be considered a normal (benign) packet and what shall be considered a malicious packet [8] [4]. Consequently, Signature-based detection is considered extremely effective and efficient in stable environments with minimal change in potential network attack types, while Anomaly-Based detection is much more preferable in environments with ever-changing types of zero-day network attacks [8]. Therefore, Anomaly-based detection has recently gained traction due to the increasing development and uncertainty of novel network attacks rising during the past decade, and proves to be a much more viable and promising approach of detecting such attacks.

Therefore our system will provide a workbench that provides all the needed tasks within the pipeline of benchmarking, testing, and development of machine learning and deep learning Anomaly-Based Network Intrusion Detection Systems.

1.2 Motivation

The development and dependence of anomaly-based IDS is currently spiking, and a benchmarking workbench specifically tailored such systems and workflows is yet to be developed. Therefore, the motivation behind the development of this project is filling the need of a workbench which eases defensive/testing security professionals, data science researchers, and governmental agencies in the development of anomaly-based IDS.

1.3 Problem Definitions

Lack of a workbench system for machine learning and deep learning anomaly-based intrusion detection systems with detailed evaluation and live packet capture classification capabilities.

2 Project Description

2.1 Scope

The developed system is a workbench system which aims to provide a platform that supports users to perform machine learning and deep learning tasks on cyberattack-related data. The system shall enable users to execute their needed pipelines with ease; those of which depend on providing flexible use of different datasets, preprocessing methods, visualization techniques, classification models, performance metrics, and analytical comparisons.

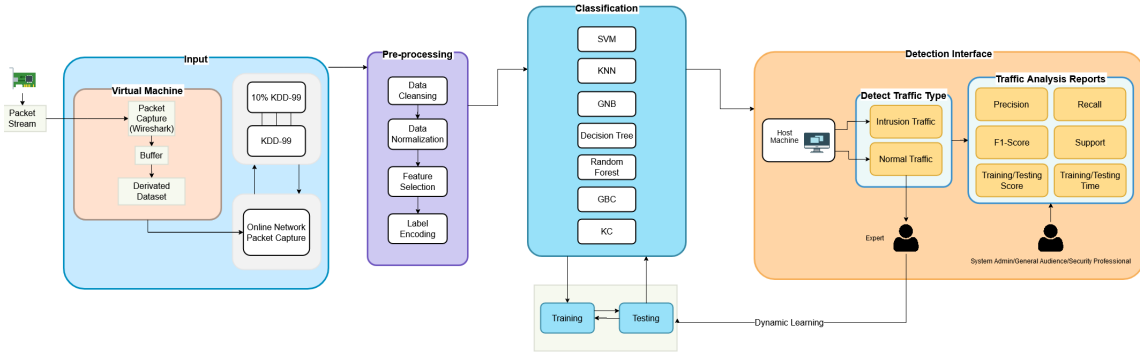

Figure 1: Architectural Design

The objective of the system is to ease the development of such pipelines needed by the system's audience & users and provide a hassle-free, all-in-one package for machine learning and deep learning intrusion detection systems benchmarking, testing, & development.

2.2 Project Overview

The system's main components are: the input component, preprocessing component, classification component, and metrics components. All of which communicate in a consecutive manner to provide the user with the needed workflow platform for machine learning and deep learning intrusion detection systems development.

The data input layer enables either importing datasets which have previously been developed by 3rd parties, or developing data through the live capture of network packets and deriving a set of features based on predefined schemas. To add, the preprocessing components supports execution of various preprocessing methods on the input data in order to aid with tasks such as data cleansing, transformation, and reduction. Moreover, the classification component supports executing of a wide variety of predefined classifiers or importing a user-defined classification model; such classifiers can be used to classify the network data and perform predictions on the existence of attack packets and their types within the processed network data. Finally, the metrics component provides an elaborate report of performance results of the utilized classification models and a comparative analysis of the results to aid with decision processes.

3 Similar System Information

Syarif et al. [11] proposed an intrusion detection mechanism based on particle swarm optimization and k-nearest-neighbor algorithm. The motivation of the work is improving the accuracy of anomaly-based intrusion detection systems. The main problem of the work is the unsatisfying accuracy found by the researchers within the reviewed previous work. The researchers contributed by proposing and developing a new hybrid-based methodology which combines both particle-swarm-optimization, and k-nearest-neighbors. The authors deployed 5-closest neighbors and achieved average accuracy of 97.92%, 99.43%, 93.44%, 99.63%, and 99.19% of detecting the normal, dos, probe, u2r, and r2l classes of the KDD-99 dataset respectively. This

paper is important for our work since it sheds the light on a promising new hybrid technique of anomaly-based intrusion detection.

Divekar et al. [1] have researched the viability of the UNSW-NB15 dataset as a substitute for the popular KDD-99 dataset. The motivation of the work is an effort to shift the research community from applying their machine learning and deep learning models on the KDD-99 dataset to a more modern dataset representable of modern-day network attacks. The main problem of the work is the fact that the KDD-99 dataset has various inherent issues [1] and is the most widely used intrusion detection dataset to this date as far as we know, and a review of alternatives to the KDD-99 dataset is highly needed. The researchers found that the UNSW-NB15 can be considered an improvement over the KDD-99 dataset and is a viable alternative for future research, to add the researchers found that the UNSW-NB15 addresses skewness issues and minority class neglection found within the KDD-99 dataset. We believe that this paper showed great importance to us as it discusses and illustrated the usefulness of the UNSW-NB15 dataset and opens the possibility for us to utilize different machine learning and deep learning models on the UNSW-NB15 dataset.

Hamid et al. [2] have presented a comparative study of different classifiers within the WEKA benchmarking system in the context of intrusion detection systems. The main problem was the need of a recent review of different classifiers and providing a comparative analysis of those classifiers' performance in intrusion detection systems. The researchers contributed to solve the problem by utilizing a diverse set of classifiers within the WEKA (Waikato Environment for Knowledge Analysis) machine learning and deep learning workbench on the KDD-99 dataset and provide a comparative analysis of the utilized classifiers and their achieved accuracies. This paper was of critical importance to us as it shed the light on the possibility of providing a worbench system, as an alternative to WEKA which was used by the authors, specifically tailored to benchmarking machine learning and deep learning intrusion detection systems which provides the specific toolset needed in such workflows and provides network data gathering techniques.

Tang et al. [12] has presented a novel architecture for network intrusion detection systems. The motivation of the work is trying to achieve a real-time network intrusion detection system that utilizes deep neural networks in the detection of network anomalies. The main problem statement of the work is the difficulty of achieving high accuracy results in detection of such network attacks while maintaining the real-world usability and providing an architecture representative of real world intrusion detection scenarios. The researchers contributed to solve the problem by presenting the novel Software Defined Networking (SDN) architecture for network intrusion detection. The main results was achieving an accuracy of 75.75% on a simulated live flow of the packets within the NSL-KDD dataset. We believe this paper was important to our work as the researchers successfully scratched the surface of developing a system based on live detection of network packets, however they left a large potential of improving upon the architecture where a more modern dataset can be used, or perhaps a live capture of real network packets.

Vinayakumar et al. [13] proposed a hybrid-based framework for network intru-

sion detection. The main problem of the work is the difficulty of analyzing live packet flow and the difficulty of maintaining highly-accurate and real-world architecture suitable for real world packets based on an architecture trained on existing datasets that suffer from some inherent issues. The researchers contributed to solving the problem by presenting a hybrid-based framework that is able to detect malicious network packets using a distributed deep learning model which is able to potentially handle large scale network packets in real time. The main result was that the proposed model was able to effectively handle large flows on network packets while maintaining higher accuracies than the utilized classical classifiers. This paper is important for our work as the authors provided a detailed description of a live packet intrusion detection system that has the room for improvement with using such architecture on actual real-time captured network packets while preserving appropriate detection time.

Pham et al. [10] have presented a methodology based on an ensemble method for detecting anomaly network traffic. The motivation of the work is improving the accuracy of anomaly-based intrusion detection systems. The main problem of the work is that a single classifier may be insufficient for detection of malicious network traffic in intrusion detection systems, therefore a review of ensemble methods is a promising area of research. The reserachers contributed to solve the problem by presenting an ensemble model which utilizes bagging and boosting with tree-based classifiers. The main results of the work is that the authors were able to achieve an average accuracy of 84.25% on the NSL-KDD dataset. This work was important to us as it discussed the promising potential of reviewing different ensemble methods with different datasets to further explore the potential of ensemble classifiers in intrusion detection.

Mostafa et al. [9] presented an intrusion detection dataset. The main problem of the work was the lack of a modern dataset representable of modern-day network attacks that can be used for research purposes. The researchers contributed to solve the problem by presenting a new dataset called UNSW-NB15 which aims to solve the problems within the previously available datasets and provide a new benchmark dataset for research utilization. The main result was that the researchers were able to effectively provide such dataset and has since started to be used extensively within the research community. The paper is of importance to us as it presents a dataset that can be used within our work which complements the classification models to be trained and tested on data that enables the models to be deployed in a real intrusion detection system.

Jing et al. [7] has presented a methodology for network intrusion detection using Support Vector Machines(SVM) on the UNSW-NB15 dataset. The motivation of the work is further exploring classification models on the modern UNSW-NB15 dataset as a more viable modern network intrusion detection dataset. The main problem was the lack of sufficient research done on the UNSW-NB15 dataset and analysis of the potential accuracy results that can be achieved on such data. The researchers contributed to solve the problem by utilizing an SVM model for classification of 10 classes within the UNSW-NB15 dataset. The main result was that the researchers were able to achieve a prediction accuracy of 75.77%. This paper was

important to our work as it explored the potential room for improvement within the researchers method through improvement of the overall accuracy and discussing the training and testing times of using classifiers such as SVM on the UNSW-NB15 dataset.

Zhiqiang et al. [14] have explored the utilization of feed forward neural networks in network intrusion detection. The motivation of the work is working exploring the possibility of using modern datasets such as the UNSW-NB15 to classify network malicious attacks using feed forward neural networks which open the gate to online self taught intrusion detection systems. The main problem of the work is the lack of using feed forward neural networks on modern network attack datasets such the UNSW-NB15 dataset. The researchers contributed to solving the problem by proposing a feed forward deep neural network on the dataset. The main result was that the researchers were able to achieve an accuracy of 99.5% on binary classification of attack packets within the dataset. This paper was important to us as it explored the possibility of development of self taught intrusion detection systems and further discussion of utilizing such feed forward networks in multi-class classification rather than binary classification.

Janarthanan et al. [6] have explored the potential of applying feature selection methods to the UNSW-NB15 dataset. The motivation of the wok is further improving the utilization of UNSW-NB15 network intrusion detection dataset for machine learning and deep learning classifier by presenting feature subsets which improve learning times while relatively preserving malicious attack detection. The main problem is the high training and testing times caused by the large number of features within the UNSW-NB15 dataset which holds back some classification models. The researchers contributed to solving the problem by presenting 3 feature subsets and analyzing the potential accuracy trade-off of those subsets. The main result was that the researchers were able to elicit one of the three developed subsets that is comprised of 5 features with a kappa statistic of 0.7567, therefore highly preserving classification accuracy while dramatically reducing training and testing time. This work is of great importance to us as it presented a subset which we can use to classify network attacks within the UNSW-NB15 dataset with minimal accuracy impact and immensely decreased training and testing times; which opens the door for utilizing much more computationally intensive classification models.

3.1 Similar System Description

WEKA, a system proposed by Holmes at al. [3] has been developed to present a workbench for machine learning tasks in an easy graphical-based interface. WEKA aims to provide an environment which facilitates the development of machine learning pipelines by providing easy access to various preprocessing, classification, visualization, and analytical tools which can be used on demand with a simple graphical interface. WEKA was developed with machine learning tasks related to agricultural and horticultural domains in mind, however since its development, WEKA has gained major acclaim in the research community due to its ease of use and the wide arsenal of preprocessing and classification tools available. The utilization WEKA has been discussed in [[6],[2]].

3.2 Comparison with Proposed Project

Due to the specific needs required in the development of intrusion detection systems that rely on machine learning and deep learning, WEKA leaves a potential for a highly needed workbench specifically tailored for the development of intrusion detection system. Therefore our work aims to fill this critical market need, by presenting a workbench which provides data input, preprocessing, classification, visualization, metrics, and comparative analysis of intrusion detection systems development through innovating the needed tools that can facilitate the development of such systems.

3.3 Screen Shots from previous systems (if needed)

Three screenshots illustrated in fig.2, fig.3, and fig.4 below.

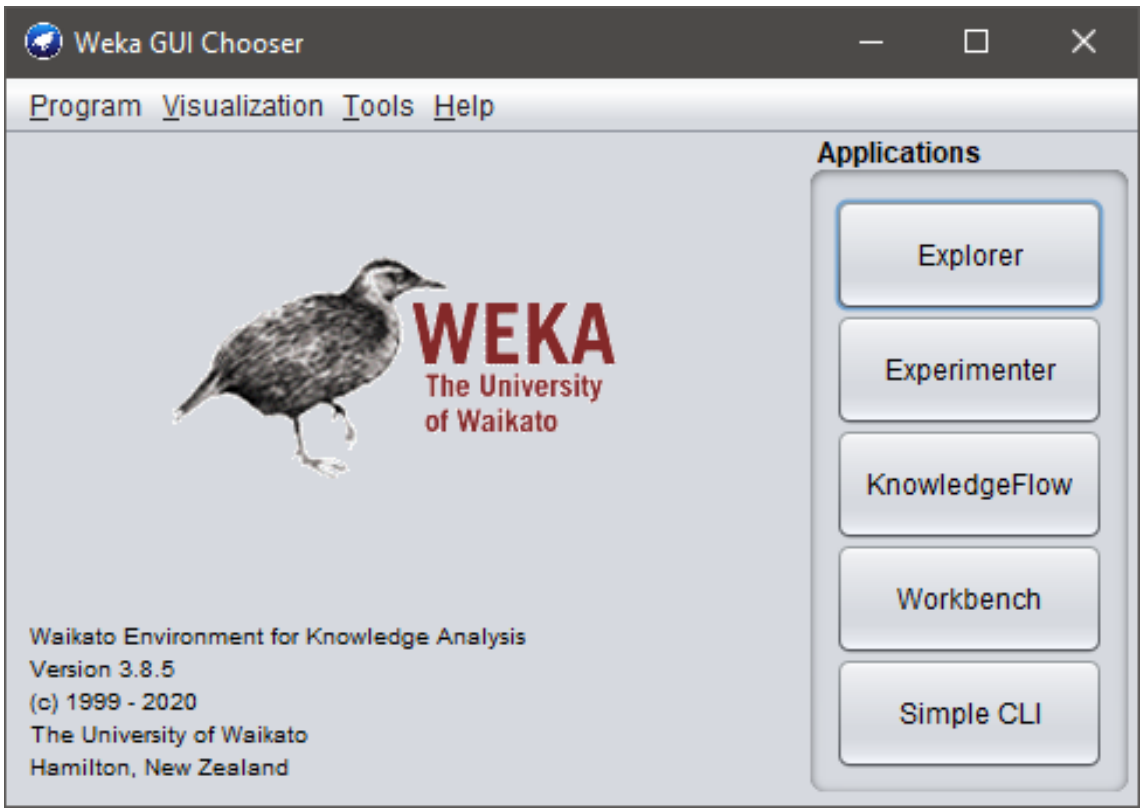

Figure 2: Screenshot of previous system (1)

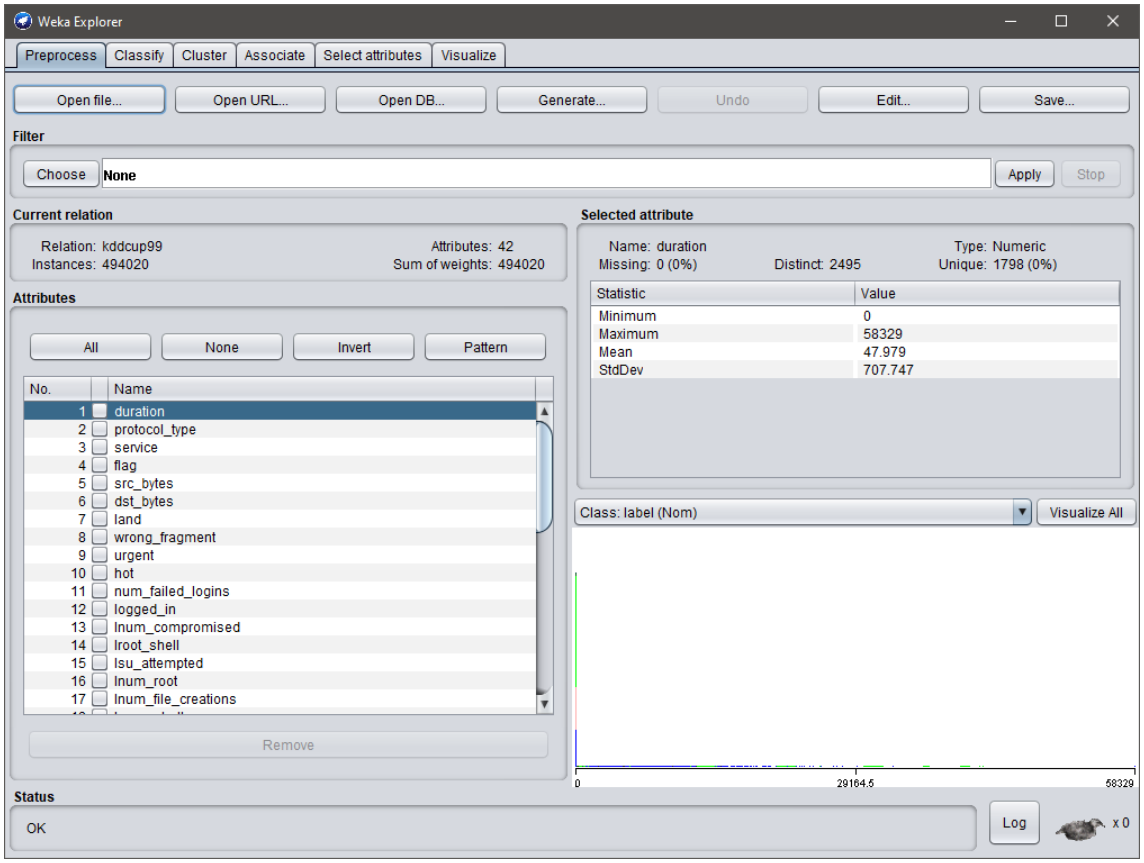

Figure 3: Screenshot of previous system (2)

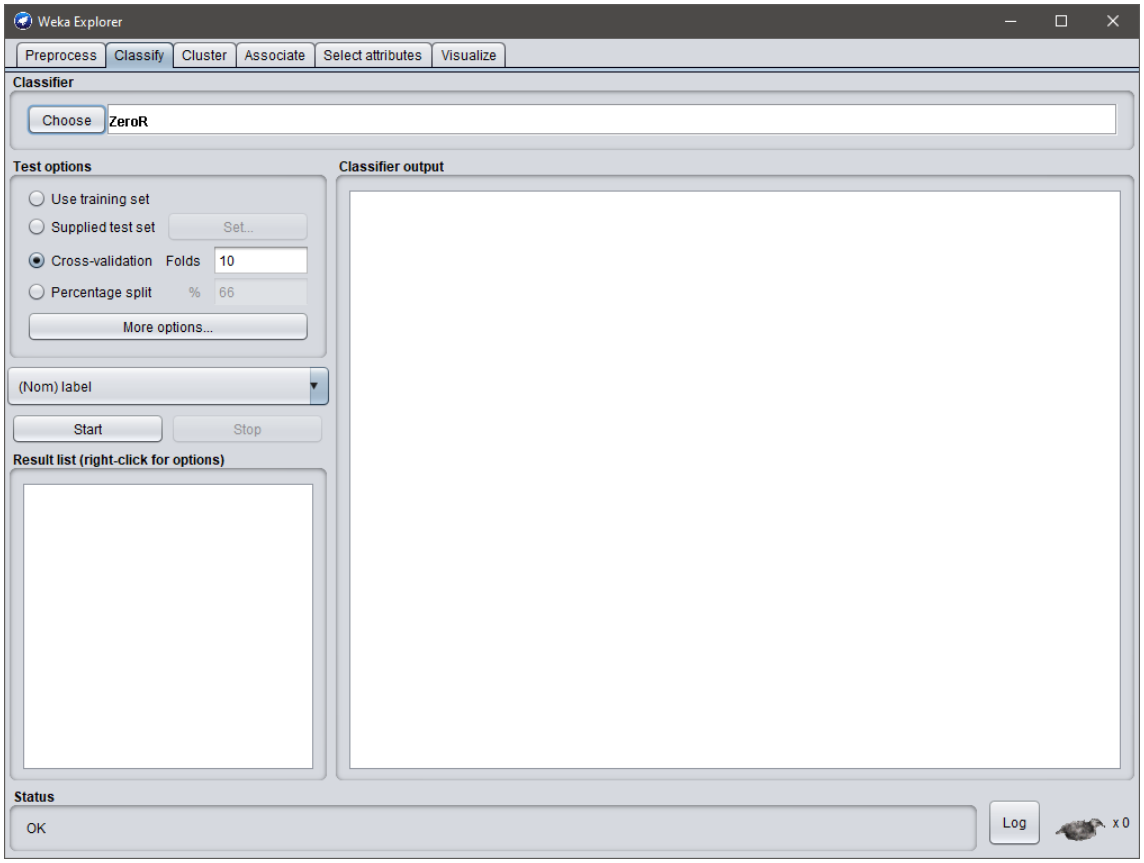

Figure 4: Screenshot of previous system (2)

4 Project Management and Deliverable

4.1 Tasks and Time Plan

Primitive plan illustrated in gantt chart fig.5

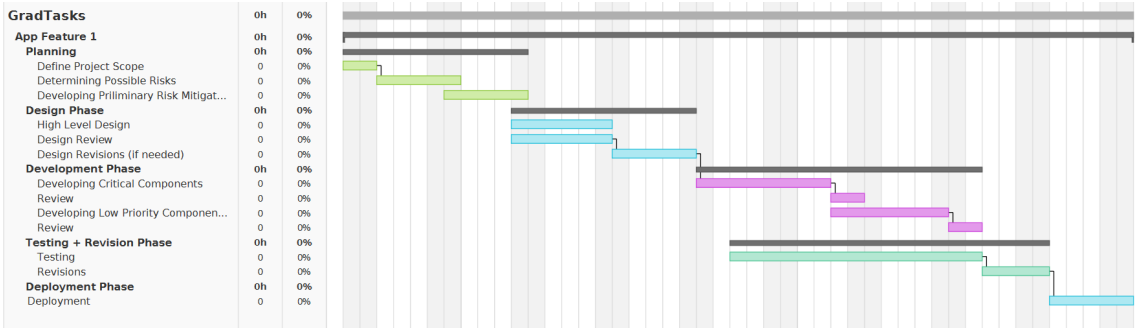

Figure 5: Primitive Plan

4.2 Budget and Resource Costs

1. Development computers: to be excluded from budget
2. Research Publications: 2,500 EGP
3. AWS EC2: 500 EGP
4. AWS SageMaker: 300EGP
5. AWS S3: 150 EGP
6. AWS RDS: 150 EGP

4.3 Supportive Documents

You can put here documents you collect for your project from customers or survey results.

5 References

References

[1] A. Divekar, M. Parekh, V. Savla, R. Mishra, and M. Shirole, "Benchmarking datasets for anomaly-based network intrusion detection: Kdd cup 99 alternatives," in *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*, 2018, pp. 1–8.

[2] Y. Hamid, M. Sugumaran, and L. Journaux, "Machine learning techniques for intrusion detection: A comparative analysis," in *Proceedings of the International Conference on Informatics and Analytics*, ser. ICIA-16. New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2980258.2980378

[3] G. Holmes, A. Donkin, and I. H. Witten, "Weka: a machine learning workbench," in *Proceedings of ANZIIS '94 - Australian New Zealnd Intelligent Information Systems Conference*, 1994, pp. 357–361.

[4] W. Hu and W. Hu, "Network-based intrusion detection using adaboost algorithm," in *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, ser. WI '05. USA: IEEE Computer Society, 2005, p. 712–717. [Online]. Available: https://doi.org/10.1109/WI.2005.107

[5] B. Ingre and A. Yadav, "Performance analysis of nsl-kdd dataset using ann," in *2015 International Conference on Signal Processing and Communication Engineering Systems*, 2015, pp. 92–96.

[6] T. Janarthanan and S. Zargari, "Feature selection in unsw-nb15 and kddcup'99 datasets," in *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*, 2017, pp. 1881–1886.

[7] D. Jing and H. Chen, "Svm based network intrusion detection for the unsw-nb15 dataset," in *2019 IEEE 13th International Conference on ASIC (ASICON)*, 2019, pp. 1–4.

[8] W. Lee and S. J. Stolfo, "A framework for constructing features and models for intrusion detection systems," *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 4, p. 227–261, Nov. 2000. [Online]. Available: https://doi.org/10.1145/382912.382914

[9] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6.

[10] N. T. Pham, E. Foo, S. Suriadi, H. Jeffrey, and H. F. M. Lahza, "Improving performance of intrusion detection system using ensemble methods and feature selection," in *Proceedings of the Australasian Computer Science Week Multiconference*, ser. ACSW '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3167918.3167951

[11] A. R. Syarif and W. Gata, "Intrusion detection system using hybrid binary pso and k-nearest neighborhood algorithm," in *2017 11th International Conference on Information Communication Technology and System (ICTS)*, 2017, pp. 181–186.

[12] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, 2016, pp. 258–263.

[13] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41 525–41 550, 2019.

[14] L. Zhiqiang, G. Mohi-Ud-Din, L. Bing, L. Jianchao, Z. Ye, and L. Zhijun, "Modeling network intrusion detection system using feed-forward neural network using unsw-nb15 dataset," in *2019 IEEE 7th International Conference on Smart Energy Grid Engineering (SEGE)*, 2019, pp. 299–303.

Software requirement specification document for Machine Learning & Deep Learning Intrusion Detection Workbench System

Mohab Sameh Ibrahim
Supervised by: Dr. Ayman Ezzat & Eng. Ahmed Neil

March 28, 2021

1 Introduction

1.1 Purpose of this document

The goal of the system is to provide a workbench system for machine learning and deep learning anomaly-based intrusion detection systems. The intended audience are system administrators, defensive/testing security professionals, governmental agencies, and data science researchers.

1.2 Scope of this document

The system provides an all-in-one workbench for developing machine learning and deep learning methods that can be used in the domain of anomaly-based intrusion detection systems (IDS). The system handles the pipeline required in developing such IDSs in through the following modules: data input, preprocessing, classification, and metric generation; where the pipeline is targeted towards IDS-specific requirements and metric evaluation is IDS-relevant.

The system's user is able to import or create input network data, preprocess the data as required, utilize various classification methods, and view performance evaluation metrics that aid the user in decision-making processes and the development of anomaly-based IDSs.

1.3 Overview

The developed system is a workbench system which aims to provide a platform that supports users to perform machine learning and deep learning tasks on cyberattack-

related data. The system shall enable users to execute their needed pipelines with ease; those of which depend on providing flexible use of different datasets, preprocessing methods, visualization techniques, classification models, performance metrics, and analytical comparisons.

The objective of the system is to ease the development of such pipelines needed by the system's audience & users and provide a hassle-free, all-in-one package for machine learning and deep learning intrusion detection systems benchmarking, testing, & development.

1.4 Business Context

The development of this product is sponsored by an ongoing support from IBM. The current development of the product does not specify a targeted profitable business goal. The organizational objective is the development of the workbench as deployable product within the work environment.

2 General Description

2.1 Product Functions

The general functionality of the system is to provide an environment that works as a workbench for benchmarking and evaluating machine learning and deep learning anomaly-based intrusion detection system.

2.2 Similar System Information

This system is an all-in-one workbench platform that utilizes and encompasses multiple components that aid in the development of the system's pipeline. The external components to be utilized are tcpdump, wireshark, sklearn, and keras.

2.3 User Characteristics

The system's users are system administrators, defensive/testing security professionals, governmental agencies, and data science researchers. The system's users are expected to have basic computer skills, basic background in network security, and a basic to intermediate background in machine learning and deep learning.

2.4 User Problem Statement

Lack of a benchmarking system for machine learning and deep learning anomaly-based intrusion detection systems with detailed evaluation and live packet capture capabilities.

2.5 User Objectives

The system's users aim to have an all-in-one easy workbench for testing, developing, and benchmarking machine learning and deep learning anomaly-based intrusion detection systems' pipelines. Users wish to have such a workbench that enables them

to work on previously gathered data or a live capture of network data within their lab environment so that they can build tailored datasets. To add, users wish to have the flexibility of utilizing a wide set of preprocessing and classification methods, as well as being able to define their own classification models.

2.6 General Constraints

General constraints include the limitations dictated by the users' environment and its hardware capabilities, the time latency introduced by data importing, and time latency introduced between backend (preprocessing, classification, & evaluation computation server) and front-end communication.

3 Functional Requirements

Table 1: Functional Requirement: Import Dataset

Function Name	Import Dataset
Description	The system shall enable the user to import a specified dataset within the Input Data module. (importDataset(String datasetName, String datasetDescription, String directory):void)
Input	String datasetName, String datasetDescription, String directory
Output	void
Critically	High-priority
Technical issues	Handling corrupted data, import exceptions, and supporting various file formats
Cost and schedule	Follows "Developing Critical Components" schedule within gantt chart fig.8 illustrated in section 10.
Risks	Import exceptions might cause difficulties with satisfying this requirement, therefore the usage of try and catch methods for handling exceptions shall be used.
Dependencies with other requirements	none
Pre-Condition	The system shall be executed and graphical interface shall display successfully.
Post-Condition	Dataset is loaded into memory. The system is ready to view & manipulate the loaded dataset.

Table 2: Functional Requirement: Select Packet Capture Feature Set

Function Name	Select Packet Capture Feature Set.
Description	The system shall allow the user to select the required feature set from the dropdown menu "Select Feature Schema" in "Live Packet Capture" menu. This will allow the system to capture packets based on the feature set described within the specified feature set.
Input	String featureSetId
Output	void
Critically	High-Priority
Technical issues	Supporting various feature schemas that can be considered useful to the application's target audience.
Cost and schedule	Follows "Developing Critical Components" schedule within gantt chart fig.8 illustrated in section 10.
Risks	Failure to derive all the features specified within the dataset's original feature set.
Dependencies with other requirements	none
Pre-Condition	Network device must support and be set into monitor mode.
Post-Condition	The system is ready to capture packets with the specified feature set on demand.

Table 3: Functional Requirement: Capture Packets

Function Name	Capture packets.
Description	The system shall enable the user to network packets on demand using TCPDump. Captured packets shall be viewed in raw format within the "Captured Packets Output" section in "Live Packet Capture" menu.
Input	int featureSetId
Output	Bool captureState
Critically	High-Priority
Technical issues	Developing an interface that shall utilize TCPDump as a subcomponent within the system's environment.
Cost and schedule	Follows "Developing Critical Components" schedule within gantt chart fig.8 illustrated in section 10.
Risks	Failure due to unsupported hardware or misconfigured packet capture settings. Hardware must conform to requirements specified in Section 4.3.
Dependencies with other requirements	Dependant on "Select Packet Capture Feature Set" requirement.
Pre-Condition	Required packet capture feature set must be selected before executing packet capture.
Post-Condition	The raw packet capture data is available within the "Captured Packets Output" section in "Live Packet Capture" menu and ready to be exported on demand.

Table 4: Functional Requirement: Export Captured Packets

Function Name	Export Captured Packets
Description	The system shall enable the user to export the captured packets using the "Export" button in "Live Packet Capture" menu to CSV format file. Moreover, the system shall prompt the user to a directory browser window, where the user shall specify the CSV file name and the required directory to save the file. The user can then press on "Save" to successfully save the CSV-formatted packet capture.
Input	void
Output	Dataframe capturedPackets
Critically	High-Priority
Technical issues	Handling CSV file size exceptions through try and catch exception handling and skipping possible special characters within the raw packet capture that might interfere with the CSV format.
Cost and schedule	Follows "Developing Critical Components" schedule within gantt chart fig.8 illustrated in section 10.
Risks	Failure to successfully handle CSV file size exceptions and special characters within the raw packet capture that might interfere with the CSV format.
Dependencies with other requirements	Dependant on "Select Packet Capture Feature Set" & "Capture Packets".
Pre-Condition	Packet capture feature set must be first selected, then packet capture must be started to gather instances of packets.
Post-Condition	The system is successfully exports instances of packets with a specified feature format in a CSV file ready for utilization on demand.

Table 5: Functional Requirement: Load Pipeline

Function Name	Load Pipeline
Description	The system shall enable the user to load a previously saved pipeline; this includes the pipeline's imported dataset, preprocessing techniques used, classifiers applied, and metric reports extracted.
Input	int pipelineId
Output	Pipeline loadedPipeline
Critically	Low-Priority
Technical issues	Fetching the loaded pipeline's dataset from an expired directory.
Cost and schedule	Follows "Developing Low Priority Components" schedule within gantt chart fig.8 illustrated in section 10.
Risks	Fetching the dataset's dataset through a saved directory string might lead to an expired directory, thus corrupting the pipeline. The system must enforce a directory saving standard to avoid fetching for a dataset within an empty directory.
Dependencies with other requirements	"Save Pipeline" requirement must be satisfied.
Pre-Condition	The system is either loaded with a previous pipeline or is empty and awaiting a new pipeline.
Post-Condition	The system is loaded with the pipeline's dataset, preprocessors, classifiers, and metric reports.

Table 6: Functional Requirement: Save Pipeline

Function Name	Save Pipeline
Description	The system shall enable the user to save a developed pipeline; this includes the pipeline's imported dataset, preprocessing techniques used, classifiers applied, and metric reports extracted.
Input	Pipeline savedPipeline
Output	Bool saveState
Critically	Low-Priority
Technical issues	Saving the pipeline's dataset would cause redundant storage build up. Therefore the system must store the directory string which points to the utilized dataset within the pipeline.
Cost and schedule	Follows "Developing Low Priority Components" schedule within gantt chart fig.8 illustrated in section 10.
Risks	
Dependencies with other requirements	None
Pre-Condition	Current pipeline is not saved and progress would be lost upon exiting the application.
Post-Condition	The current pipeline is saved on the storage media and progress is preserved and can be continued in upcoming development sessions.

Table 7: Functional Requirement: Show Dataset Overview

Function Name	Show Dataset Overview
Description	The system shall show an overview about the currently loaded dataset. The overview is composed of the dataset name, number of features within the dataset, and the number of rows within the dataset.
Input	int datasetId
Output	String datasetName, int featureNo, int rowNo
Critically	High-Priority
Technical issues	None
Cost and schedule	Follows "Developing High Priority Components" schedule within gantt chart fig.8 illustrated in section 10.
Risks	None
Dependencies with other requirements	"Import Dataset" requirement must be satisfied.
Pre-Condition	The system does not show the overview information of the currently imported dataset.
Post-Condition	The system shows the overview information of the currently imported dataset.

Table 8: Functional Requirement: Show Dataset Features

Function Name	Show Dataset Features
Description	The system shall show distinct features within the dataset in "Data Input and Preprocessing" component within the "Features" section. Each feature must be included with its feature number and feature name.
Input	int datasetId
Output	String datasetFeatures
Critically	High-Priority
Technical issues	None.
Cost and schedule	Follows "Developing Low Priority Components" schedule within gantt chart fig.8 illustrated in section 10.
Risks	None.
Dependencies with other requirements	"Import Dataset" requirement must be satisfied.
Pre-Condition	Dataset's features are not displayed in "Features" section within the "Data Input and Preprocessing" tab.
Post-Condition	Dataset's features are displayed in "Features" section within the "Data Input and Preprocessing" tab.

Table 9: Functional Requirement: Show Selected Feature Overview

Function Name	Show Selected Feature Overview
Description	Upon selecting a feature in "Features" section in "Data Input and Preprocessing" tab, the system shall show an overview of the selected feature. The system overview is composed of (feature_name, distinct_values, data_type, missing_value_percentage, unique_value_percentage). In the case of the feature being of a numeric type, the overview shall also include a table which displays the feature's (minimum_value, maximum_value, mean_value, standard_deviation). Otherwise, in the case of the feature being of a nominal type, the overview shall show the feature's each distinct label and distinct_value_count of that label.
Input	string featureName
Output	String feature_name, int distinct_values, String data_type, int missing_value_percentage, int unique_value_percentage, int minimum_value, int maximum_value, int mean_value, int standard_deviation, int distinct_value_count.
Critically	High-Priority
Technical issues	Handling the different feature overview presented based on the feature type; where numerical features present a different overview than nominal features.
Cost and schedule	Follows "Developing High Priority Components" schedule within gantt chart fig.8 illustrated in section 10.
Risks	Possible technical issues within handling the different feature overview presented based on the feature type; where numerical features present a different overview than nominal features. A specific flag shall be set on the server side which informs the facade layer which overview form shall be sent within the message carrying the feature overview.
Dependencies with other requirements	"Import Dataset" requirement must be satisfied.
Pre-Condition	User shall import a dataset and select a feature within the displayed features in the "Features" section.
Post-Condition	The system displays an overview with the specified characteristics as described.

Table 10: Functional Requirement: Show Selected Feature Visualization

Function Name	Show Selected Feature Visualization
Description	Upon selecting a feature in the "Features" section in "Data Input and Preprocessing", the system shall show a visual graph of the selected feature. The visual graph shall be a bar graph of the feature's distinct_values on the x-coordinate and the number_of_occurrences on the y-coordinate.
Input	String featureName
Output	Image visualizationImg
Critically	Low-Priority
Technical issues	Retrieving an image of the visualization from the backend server.
Cost and schedule	Follows "Developing Low Priority Components" schedule within gantt chart fig.8 illustrated in section 10.
Risks	Retrieving an image of the visualization from the backend server might cause specific difficulties, where the image shall not be transferred within the API transaction to avoid overflowing the message; therefore the backend server must create and export the visualization of the required feature to a predefined directory, which the facade fetches upon user request.
Dependencies with other requirements	"Import Dataset" requirement must be satisfied.
Pre-Condition	User shall import a dataset and select a feature within the displayed features in the "Features" section.
Post-Condition	The system displays an overview with the specified characteristics as described.

Table 11: Functional Requirement: Choose Preprocessor

Function Name	Choose Preprocessor
Description	The system shall enable the user to select a preprocessing technique from the "Choose Preprocessor" dropdown list in the "Data input and Preprocessing" tab.
Input	String preprocessorName
Output	void
Critically	High-Priority
Technical issues	None
Cost and schedule	Follows "Developing High Priority Components" schedule within gantt chart fig.8 illustrated in section 10.
Risks	None
Dependencies with other requirements	"Import Dataset" requirement must be satisfied.
Pre-Condition	No preprocessing technique is specified nor ready to be executed upon user request.
Post-Condition	A preprocessing technique is specified and ready to be applied to the input data upon user's request.

Table 12: Functional Requirement: Change Preprocessor Parameters

Function Name	Change Preprocessor Parameters
Description	Within the preprocessing module, the system shall enable the user to change the specified preprocessor parameters which changes the parameters passed to the preprocessing model.
Input	String preprocessorName, String preprocessorParams
Output	void
Critically	Low-Priority
Technical issues	Each preprocessing technique may have a different set of parameters that can be changed, therefore the system must dynamically generate and display the set of possible parameters to be changed of the preprocessing technique.
Cost and schedule	Follows "Developing Low Priority Components" schedule within gantt chart fig.8 illustrated in section 10.
Risks	A risk is present where each preprocessing technique may have a different set of parameters that can be changed, therefore the system must dynamically generate and display the set of possible parameters to be changed of the preprocessing technique. Therefore the facade must first send the specified preprocessing technique specified, then a set of parameters shall be returned to the facade layer.
Dependencies with other requirements	"Choose Preprocessor" requirement must be satisfied.
Pre-Condition	The specified preprocessor is using the default parameters specified by its library
Post-Condition	The preprocessor displays the set of parameters that can be changed by the user on demand.

Table 13: Functional Requirement: Apply Preprocessing Technique

Function Name	Apply Preprocessing Technique
Description	within the preprocessing module, the system shall enable the user to execute the specified preprocessing technique specified.
Input	Dataframe dataset, String preprocessorName, String preprocessorParams
Output	void
Critically	High-Priority
Technical issues	Possible improper specified parameters of the utilized preprocessor might be passed.
Cost and schedule	Follows "Developing High Priority Components" schedule within gantt chart fig.8 illustrated in section 10.
Risks	Risk might occur where improper specified parameters of the utilized preprocessor might be passed; therefore exception handling must be done and improper preprocessing output shall be returned to facade.
Dependencies with other requirements	Mandatory dependency on "Choose Preprocessor" requirement must be satisfied and optional dependency on "Change Preprocessor Parameters".
Pre-Condition	No preprocessing technique is applied on the input data.
Post-Condition	The specified preprocessing technique is applied on the input data.

Table 14: Functional Requirement: Choose Classifier

Function Name	Choose Classifier
Description	The system shall enable the user to select a classification model from the "Choose Classifier" dropdown list in the "Classification" tab.
Input	String classifierName
Output	void
Critically	High-Priority
Technical issues	None
Cost and schedule	Follows "Developing High Priority Components" schedule within gantt chart fig.8 illustrated in section 10.
Risks	None
Dependencies with other requirements	Mandatory requirement "Import Dataset" must be satisfied and optional dependency on "Apply Preprocessing Technique".
Pre-Condition	No classification model is specified nor ready to be executed upon user request.
Post-Condition	A classification model is specified and ready to be executed upon the input data upon user's request.

Table 15: Functional Requirement: Change Classifier Parameters

Function Name	Change Classifier Parameters
Description	The system shall enable the user to change the specified classifier's parameters before execution.
Input	String classifierName, String classifierParams
Output	void
Critically	Low-Priority
Technical issues	Each classification model may have a different set of parameters that can be changed, therefore the system must dynamically generate and display the set of possible parameters to be changed of the classification model.
Cost and schedule	Follows "Developing Low Priority Components" schedule within gantt chart fig.8 illustrated in section 10.
Risks	A risk is present where each classification model may have a different set of parameters that can be changed, therefore the system must dynamically generate and display the set of possible parameters to be changed of the classification model. Therefore the facade must first send the specified classifier specified, then a set of changeable parameters shall be returned to the facade layer.
Dependencies with other requirements	"Choose Classifier" requirement must be satisfied.
Pre-Condition	The specified classifier is using the default parameters specified by its library.
Post-Condition	The classifier displays the set of parameters that can be changed by the user on demand.

Table 16: Functional Requirement: Change Classifier Test Options

Function Name	Change Classifier Test Options
Description	The system shall enable the user to change the classifier's test options. Where the test options shall include applying cross-validation with a specified number of folds, train-to-test specified percentage, and specifying the predicted feature.
Input	String classifierName, int foldsNo, float trainToTestRatio, String predictedFeature
Output	void
Critically	Low-Priority
Technical issues	Applying string sanitization, handling logical, and syntax typos/errors typed into the text fields available in the test options.
Cost and schedule	Follows "Developing Low Priority Components" schedule within gantt chart fig.8 illustrated in section 10.
Risks	User might enter a flawed or an injection string into the test options fields therefore, validation and sanitization of the entered parameters within the test options must be performed.
Dependencies with other requirements	"Choose Classifier" requirement must be satisfied.
Pre-Condition	Test options are set to default values, where a train-to-test ratio of 70% is set.
Post-Condition	User-defined test options are set and classification is ready to be performed using the specified test options.

Table 17: Functional Requirement: Apply Classification Model

Function Name	Apply Classification Model
Description	The system shall enable the user to execute the specified classification model specified.
Input	Dataframe dataset, String classifierName, String classifierParams
Output	void
Critically	High-Priority
Technical issues	Possible improper specified parameters of the utilized classifier might be passed.
Cost and schedule	Follows "Developing High Priority Components" schedule within gantt chart fig.8 illustrated in section 10.
Risks	Risk might occur where the classification task might cause a system failure; therefore the system shall be able to gracefully terminate the classification task upon such event. Another risk might occur if improper specified parameters of the utilized classifier might be passed; therefore exception handling must be done and improper classification output shall be returned to facade.
Dependencies with other requirements	Mandatory dependency on "Choose Classifier" requirement must be satisfied and optional dependency on "Change Classifier Parameters".
Pre-Condition	No classification model is applied on the input data.
Post-Condition	The specified classification model is applied on the input data.

Table 18: Functional Requirement: View Classification Task Output

Function Name	View Classification Task Output
Description	Upon selecting a previously executed classification task from "Results List" in the "Classification" tab, the system shall preview the classification task output in the "Classifier Output" section. The classifier output shall include the classification task's execution result as described from the utilized classifier's library (sklearn or keras).
Input	String classifierName
Output	String taskOutput
Critically	High-Priority
Technical issues	The classification result string might be too large to be transferred within the api transaction.
Cost and schedule	Follows "Developing High Priority Components" schedule within gantt chart fig.8 illustrated in section 10.
Risks	A risk exists where the classification result string might be too large to be transferred within the api transaction; therefore the system shall be able to divide the output string on multiple api messages if necessary.
Dependencies with other requirements	"Apply Classifier" requirement must be satisfied.
Pre-Condition	No output of the performed classifier is available within the "Classifier Output" section.
Post-Condition	The performed classifier output is available within the "Classifier Output" section.

Table 19: Functional Requirement: View Previous Classification Tasks

Function Name	View Previous Classification Tasks
Description	The system shall show the previously performed classification tasks in the "Results List" section within the "Classification" tab. Where a list including each distinct classifier performed and its name is available.
Input	void
Output	String previousClassificationTasks[n]
Critically	High-Priority
Technical issues	None
Cost and schedule	Follows "Developing High Priority Components" schedule within gantt chart fig.8 illustrated in section 10.
Risks	None
Dependencies with other requirements	"Apply Classifier" requirement must be satisfied.
Pre-Condition	The performed classifier does not exist within the previous classifiers list.
Post-Condition	The performed classifier is listed within the previous classifiers list.

Table 20: Functional Requirement: View Classifiers Metrics Report

Function Name	View Classifiers Metrics Report
Description	The system shall enable the user to view a report of the performance and accuracy metrics; the report shall be printed within the "Metrics Output" section within the Metrics tab. The printed metrics shall be based on the user-specified classifiers and the user-specified metrics selected by the user from "Select Classifiers" list(the previously performed classification tasks) and "Select Metrics" list(precision, recall, F1-Score, Support, Training Time, Testing Time).
Input	Dataframe dataset, String classifiersSelected[n], String metricsSelected[n]
Output	Metric metricsReturned[n]
Critically	High-Priority
Technical issues	Exceptions might occur if the user tries to preview a metrics report without selecting classifiers or metrics.
Cost and schedule	Follows "Developing High Priority Components" schedule within gantt chart fig.8 illustrated in section 10.
Risks	A risk is present where handling must be performed to prevent the user from previewing a metrics report without selecting classifiers or metrics, which might cause an exception occurrence.
Dependencies with other requirements	Dependant on "Import Dataset" and "Apply Classifier" requirements.
Pre-Condition	No metrics report is printed on the "Metrics Output" section in the "Metrics" tab.
Post-Condition	The user-selected metrics of the user-selected classifiers are printed on the "Metrics Output" section.

4 Interface Requirements

4.1 User Interfaces

This system shall interface with the user though a graphical user interface that enables the user to develop the required pipeline tasks through an easy click based interface with minimal code knowledge.

4.1.1 GUI

Screen images illustrated in fig.1, fig.2, fig.3, fig.4

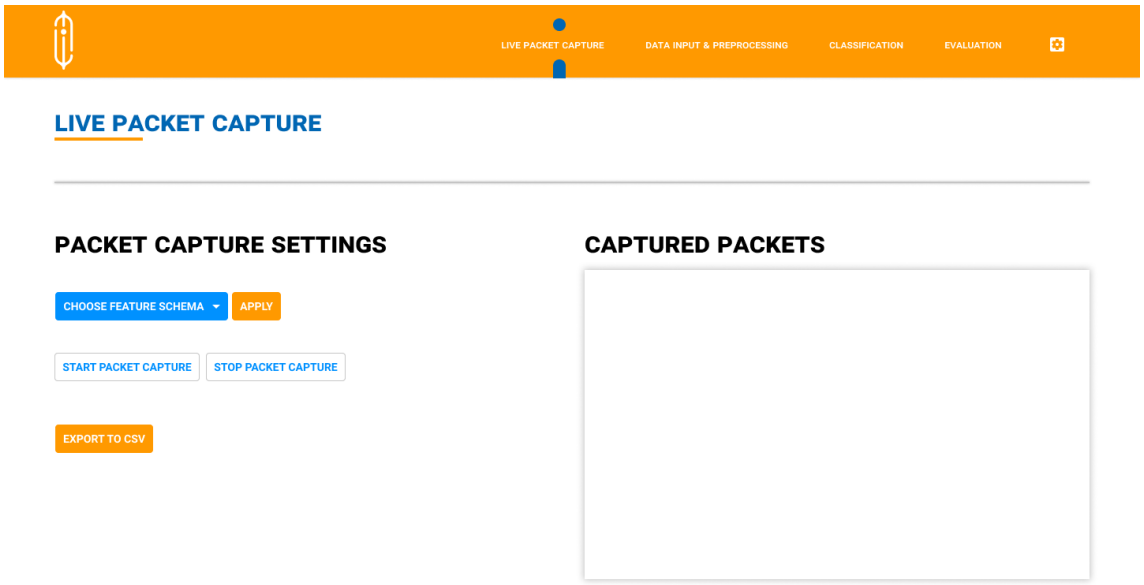

Figure 1: Live Packet Capture screenshot

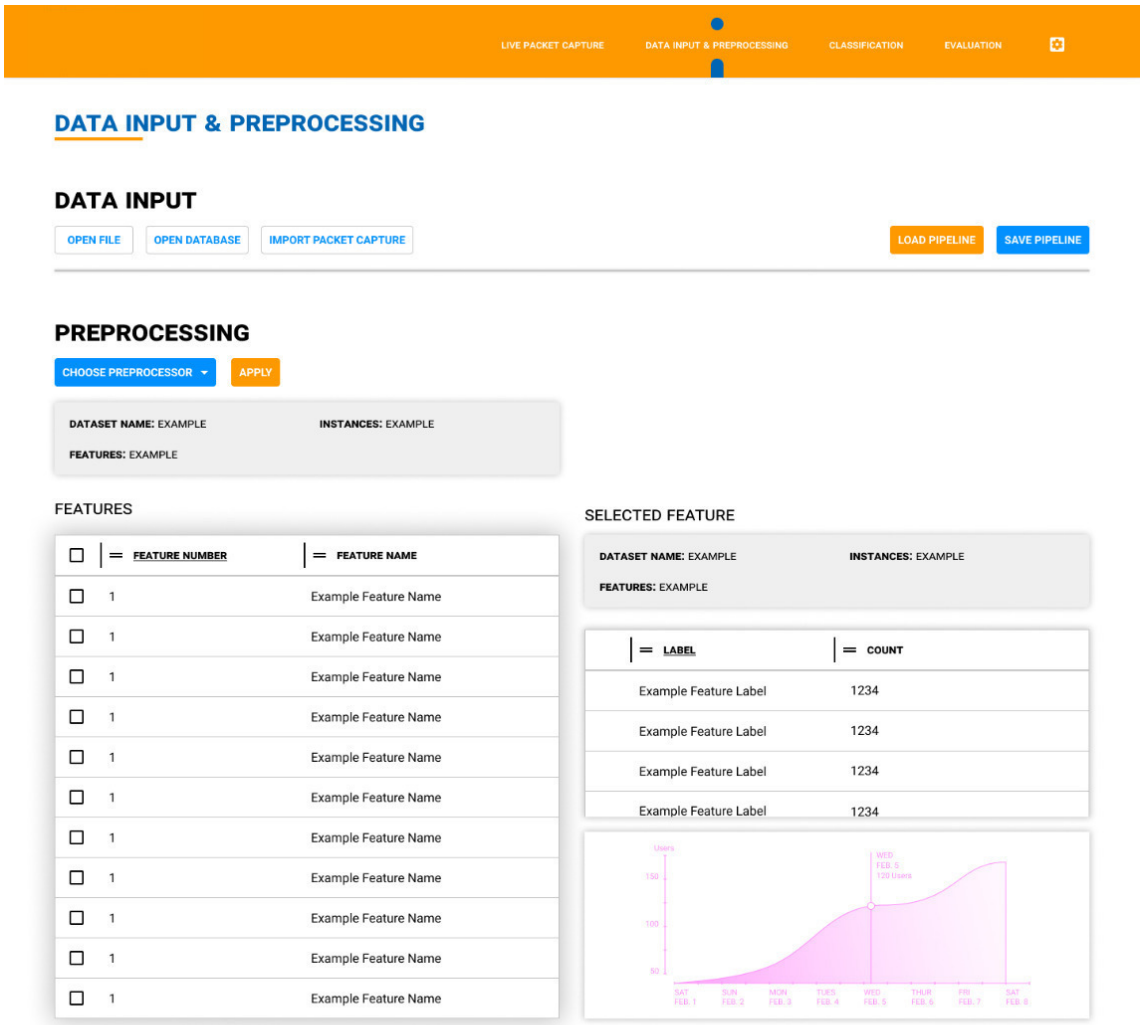

Figure 2: Data input & Preprocessing screenshot

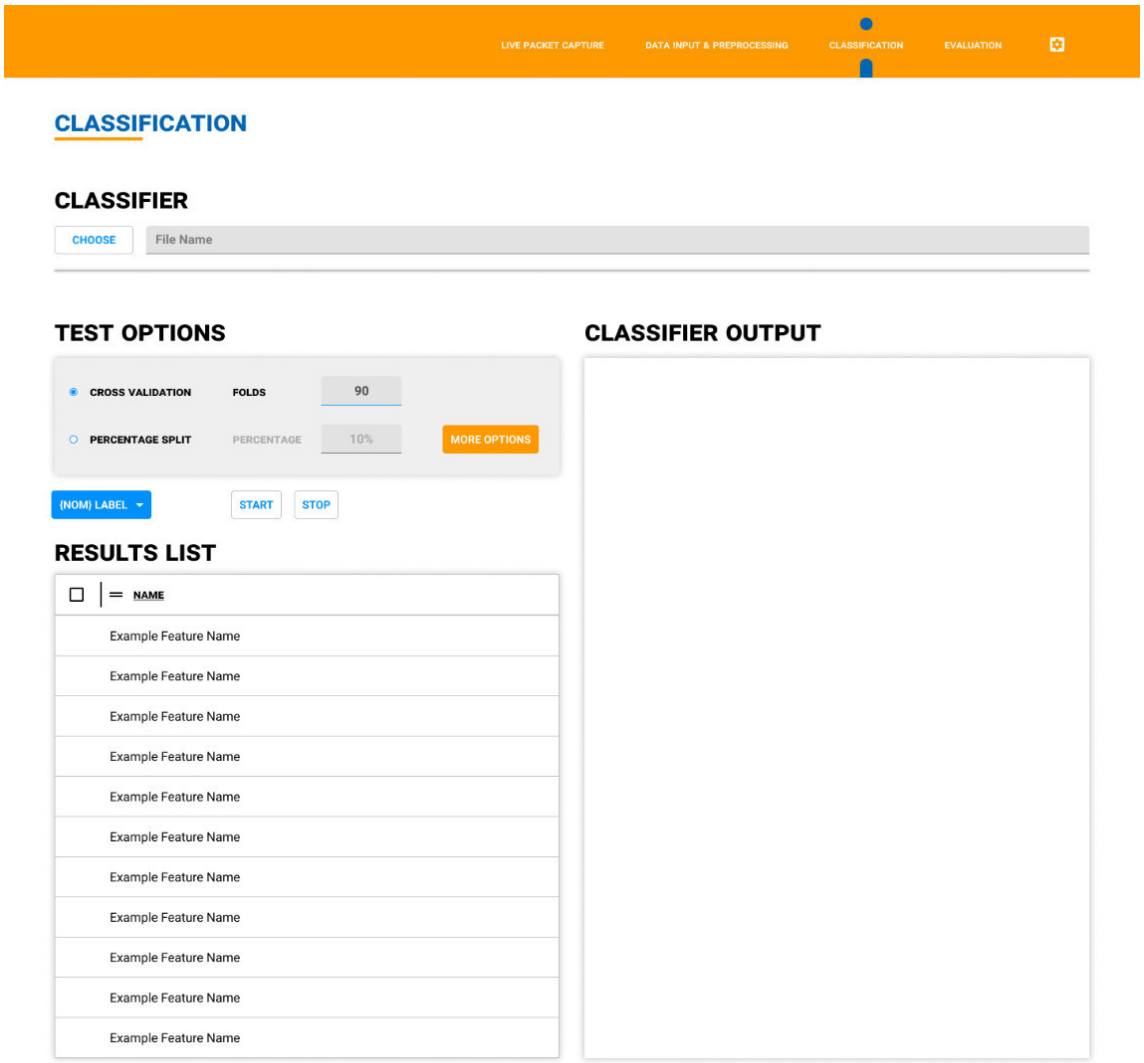

Figure 3: Classification screenshot

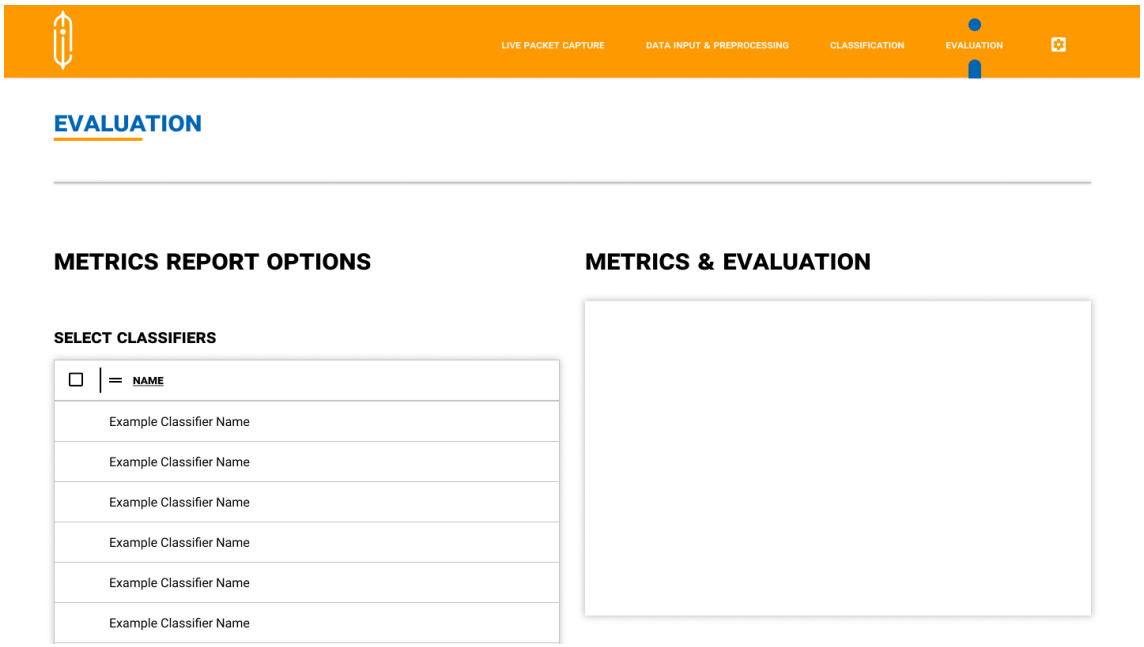

Figure 4: Evaluation screenshot

4.1.2 CLI

The system avoids a command-line-interface for user interaction for the sake of intuitive design and interaction.

4.1.3 API

For each public interface function, the name, arguments, return values, examples of invocation, and interactions with other functions should be provided.

The system shall use REST API to pass the user-determined data, visualizations, preprocessors, classifiers, parameters, and metrics from the web-based interface to the computation server within the backend.

4.1.4 Diagnostics or ROM

Diagnostics shall be obtained within the run output log within the python backend server. Such log can be accessed either through a python IDE or through an exported error log within the root server directory.

4.2 Hardware Interfaces

The system relies on network hardware interfaces, which is discussed in detail within 4.3 (Communications Interfaces).

4.3 Communications Interfaces

The system relies on connection with network interfaces. During a session of packet capture, the system shall set the network interface module into monitor mode through airmon-ng, then the packets are captured using TCPdump as a raw pcap file.

4.4 Software Interfaces

The system shall interface with a backend server running python 3.7 for computation tasks and a mySQL database for logging tasks. Required software packages include: pandas, numpy, sci-kit learn, keras, and matplotlib.

5 Performance Requirements

Thorough testing dictated sufficient hardware requirements being: AMD Ryzen 5 3600 6 core 12 thread CPU (or Intel equivalent), 16GB 3200Mhz RAM, 256GB SATA or NVME SSD drive, Monitor-mode capable Network Interface. Accelerated classification tasks would require any compatible NVIDIA GPU with CUDA-enabled cores.

6 Design Constraints

6.1 Hardware Limitations

Hardware limitations are present in going below the specified hardware requirements discussed in (5), which can result in extremely lengthy data import, classification, and evaluation times. To add, the lack of a monitor-mode capable network interface would cause the inability of performing successful live packet capture as the data input method.

7 Other non-functional attributes

7.1 Security

The system shall follow common security code convention through the development of the product such as: following data protection, access modifiers, hashing sensitive data, and obfuscating gathered packet data on demand.
Sensitive database information shall be encrypted with AES-256 encryption,
String sanitization shall be performed within any input field within the user interface to avoid possible code injection.

7.2 Binary Compatibility

The system shall be able to run without recompiling on different environments, thus successfully satisfying binary compatibility.

7.3 Reliability

Using Mean Time Between Failures:

$$MTBF = \frac{TotalElapsedTime – SumofDowntime}{NumberofFailures}$$

The system shall provide a minimum MTBF of 120 hours. To add, the source code shall follow exception handling code convention; where try and catch methods should be used extensively.

7.4 Maintainability

The system shall provide an output interface through the backend server log which describes any occuring issue. To add, resolution of server errors shall not exceed a downtime of 2 hours per event.

7.5 Portability

The system shall a web based interface that can be deployed or accessed through independent portable devices. However, the computation server shall only provide an interface gateway without requiring server portability.

7.6 Re-usability

The system shall use component based programming with object orientation convention, thus achieving reusable components that can be independently tested or deployed with the presence of stubs and drivers.

7.7 Resource Utilization

The system shall not saturate the recommended 16GBs of memory during a session. Classification tasks shall saturate CPU resources if needed.

8 Preliminary Object-Oriented Domain Analysis

Class diagram is illustrated in fig.5

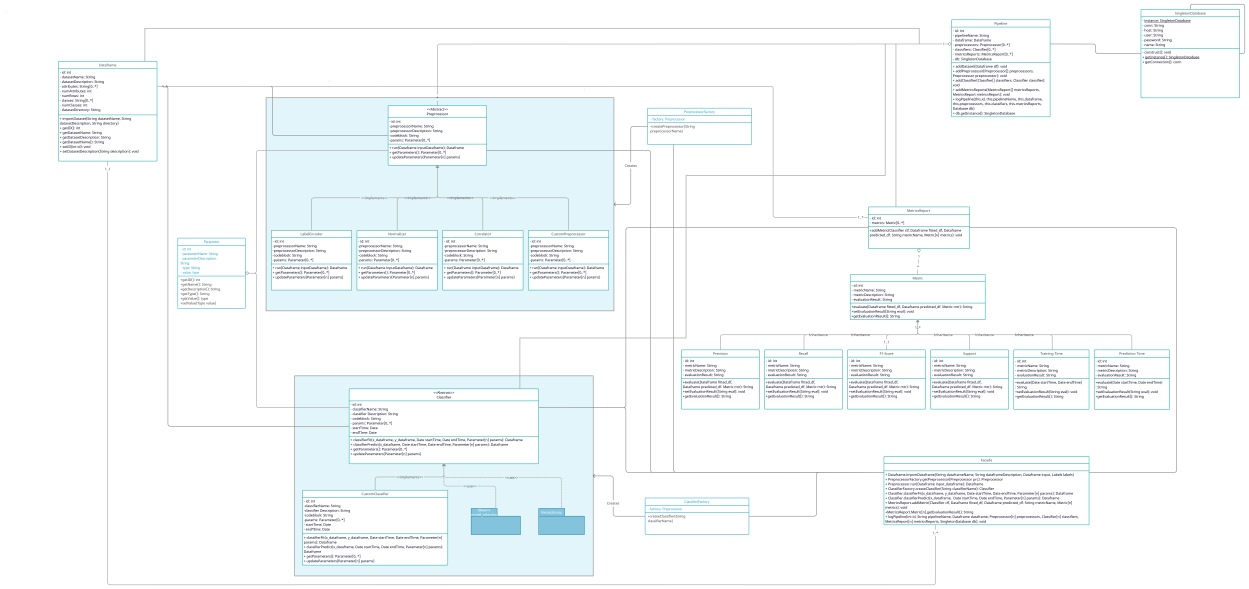

Figure 5: Class Diagram

8.1 Inheritance Relationships

Illustrated in fig.6

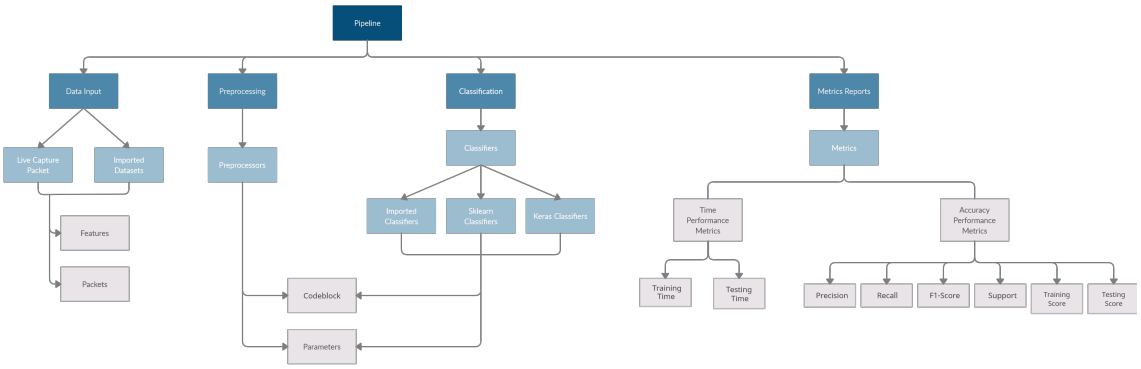

Figure 6: Inheritance Relations

8.2 Class descriptions

8.2.1 Object

Classifier <<Abstract>>

8.2.2 List of Superclasses:

Pipeline

8.2.3 List of Subclasses:

CustomClassifier

8.2.4 Purpose:

Acts as an abstract class for the Classifier Factory. Dictates the classifiers, their attributes, and their methods within the classification module.

8.2.5 Collaborations:

Collaborates with Dataframe class to classify the input data passed. MetricsReport class to create a metrics report from the classified dataframe. ClassifierFactory to create a classifier within the runtime. Parameter to instantiate different parameters for a classifier. Pipeline to add a classifier's details to a pipeline. Facade to start fitting or predicting a classification result.

8.2.6 Attributes:

id: int classifierName: String classifier Description: String codeblock: String params: Parameter[0..*] startTime: Date endTIme: Date

8.2.7 Operations

classifierFit(x_dataframe, y_dataframe, Date startTime, Date endTime, Parameter[n] params): Dataframe classifierPredict(x_dataframe, Date startTime, Date endTime, Parameter[n] params): Dataframe getParameters(): Parameter[0..*] updateParameters(Parameter[n] params)

8.2.8 Object

ClassifierFactory (Concrete)

8.2.9 List of Superclasses:

none

8.2.10 List of Subclasses:

none

8.2.11 Purpose:

ClassifierFactory acts as the Factory class within the abstract factory design pattern, where it creates new classifiers using the abstract Classifier class at runtime.

8.2.12 Collaborations:

Collaborates with Classifier abstract class to create new classifiers at runtime.

8.2.13 Attributes:

factory: Classifier

8.2.14 Operations

createClassifier(String classifierName)

8.2.15 Object

Dataframe (Concrete)

8.2.16 List of Superclasses:

Pipeline

8.2.17 List of Subclasses:

none

8.2.18 Purpose:

Serves as a class which encapsulates the dataset's attributes to be used as an input dataframe in the preprocessing, classification, and evaluation modules.

8.2.19 Collaborations:

Collaborates with: Facade, to import a new dataset. Preprocessor, to run preprocessing techniques on the imported dataset. Classifier, to classify the dataset.

8.2.20 Attributes:

id: int datasetName: String datasetDescription: String attributes: String[0..*] numAttributes: int numRows: int classes: String[0..*] numClasses: int datasetDirectory: String

8.2.21 Operations

importDataset(String datasetName, String datasetDescription, String directory) getID(): int getDatasetName: String getDatasetDescription: String getDatasetName(): String setID(int id): void setDatasetDescription(String description): void

8.2.22 Object

Facade

8.2.23 Operations

Dataframe.importDataframe(String dataframeName, String dataframeDescription, Dataframe input, Labels labels) PreprocessorFactory.getPreprocessor(Preprocessor prc): Preprocessor Preprocessor.run(Dataframe input_dataframe): Dataframe ClassifierFactory.createClassifier(String classifierName): Classifier Classifier.classifierFit(x_dataframe, y_dataframe, Date startTime, Date endTime, Parameter[n] params): Dataframe Classifier.classifierPredict(x_dataframe, Date startTime, Date endTime, Parameter[n] params): Dataframe MetricsReport.addMetric(Classifier clf, Dataframe fitted_df, Dataframe predicted_df. String metricName, Metric[n] metrics): void MetricsReport.Metric[].getEvaluationResult(): String logPipeline(int id, String pipelineName, Dataframe dataframe, Preprocessor[n] preprocessors, Classifier[n] classifiers, MetricsReport[n] metricsReports, SingletonDatabase db): void

8.2.24 Object

Metric (Concrete)

8.2.25 List of Superclasses:

MetricsReport

8.2.26 List of Subclasses:

Precision, Recall, F1-Score, Support, TrainingTime, PredictionTime

8.2.27 Purpose:

Defines the attributes of a metric, engages evaluation of the classification task, and returns evaluation results.

8.2.28 Collaborations:

Collaborates with MetricsReport to add the evaluated classification task result to a metrics report that can be later returned to the facade layer.

8.2.29 Attributes:

id: int metricName: String metricDescription: String evaluationResult: String

8.2.30 Operations

evaluate(Dataframe fitted_df, Dataframe predicted_df, Metric mtr): String setEvaluationResult(String eval): void getEvaluationResult(): String

8.2.31 Object

MetricsReport (Concrete)

8.2.32 List of Superclasses:

Pipeline

8.2.33 List of Subclasses:

Metric

8.2.34 Purpose:

This class gathers a set of instantiated and evaluated Metrics within a single report containing the desired metrics for the classification results that shall be viewed at the facade layer.

8.2.35 Collaborations:

Collaborates with: Pipeline and Dataframe, to add different metrics reports to a single pipeline relative to the utilized dataset. Metric, to instantiate different metrics. Classifier: to evaluate the classification tasks. Facade: to view the metrics report on the facade layer.

8.2.36 Attributes:

id: int metrics: Metric[0..*]

8.2.37 Operations

addMetric(Classifier clf, Dataframe fitted_df, Dataframe predicted_df. String metricName, Metric[] metrics): void

8.2.38 Object

Parameter (Concrete)

8.2.39 List of Superclasses:

1. Preprocessor
2. Classifier

8.2.40 List of Subclasses:

None

8.2.41 Purpose:

Encapsulates the parameters that can be within the instantiated preprocessor or classifier.

8.2.42 Collaborations:

Collaborates with Preprocessor and Classifier to encapsulate either one's parameter within a discrete class to conform to Object-Orientation.

8.2.43 Attributes:

id: int parameterName: String parameterDescription: String type: String value: type

8.2.44 Operations

getID(): int getName(): String getDescription(): String getType(): String getValue(): type setValue(Type value)

8.2.45 Object

Pipeline (Concrete)

8.2.46 List of Superclasses:

None

8.2.47 List of Subclasses:

1. Dataframe
2. Preprocessor
3. Classifier
4. MetricsReport

8.2.48 Purpose:

This class encapsulates the utilized dataset, preprocessors, classifiers, and generated metrics reports within a session in order to enable saving the developed pipeline within a single object.

8.2.49 Collaborations:

Collaborates with dataframe, preprocessor, classifier, and metrics report to save their values within a single development pipeline that can be saved and logged for future access and resuming development progress. Collaborates with Facade to enable saving or loading a pipeline from the facade layer.

8.2.50 Attributes:

id: int pipelineName: String dataframe: Dataframe preprocessors: Preprocessor[0..*] classifiers: Classifier[0..*] metricsReports: MetricsReport[0..*] db: SingletonDatabase

8.2.51 Operations

addDataset(Dataframe df): void addPreprocessor(Preprocessor[] preprocessors, Preprocessor preprocessor): void addClassifier(Classifier[] classifiers, Classifier classifier): void addMetricsReports(MetricsReport[] metricsReports, MetricsReport metricsReport): void logPipeline(int id, String pipelineName, Dataframe dataframe, Preprocessor[n] preprocessors, Classifier[n] classifiers, MetricsReport[n] metricsReports, SingletonDatabase db): void db.getInstance(): SingletonDatabase

8.2.52 Object

Preprocessor <<Abstract>>

8.2.53 List of Superclasses:

None

8.2.54 List of Subclasses:

LabelEncoder, Normalizer, Correlator, CustomPreprocessor

8.2.55 Purpose:

Acts as an abstract class for the Preprocesor Factory. Dictates the preprocessors, their attributes, and their methods within the preprocessing module.

8.2.56 Collaborations:

Collaborates with Dataframe class to preprocess the input data passed. PreprocessorFactory to create a Preprocessor within the runtime. Parameter to instantiate different parameters for a preprocessor. Pipeline to add a preprocessor's details to a pipeline. Facade to start preprocessing input data.

8.2.57 Attributes:

id: int preprocessorName: String preprocessorDescription: String codeblock: String params: Parameter[0..*]

8.2.58 Operations

run(Dataframe inputDataframe): Dataframe getParameters(): Parameter[0..*] updateParameters(Parameter[n] params)

8.2.59 Object

PreprocessorFactory (Concrete)

8.2.60 List of Superclasses:

none

8.2.61 List of Subclasses:

none

8.2.62 Purpose:

PreprocessorFactory acts as the Factory class within the abstract factory design pattern, where it creates new preprocessors using the abstract Preprocessor class at runtime.

8.2.63 Collaborations:

Collaborates with Preprocessor abstract class to create new preprocessors at runtime.

8.2.64 Attributes:

factory: Preprocessor

8.2.65 Operations

createPreprocessor(String preprocessorName)

8.2.66 Object

SingletonDatabase (Concrete)

8.2.67 List of Superclasses:

None.

8.2.68 List of Subclasses:

None.

8.2.69 Purpose:

Enables the Singleton design pattern to restrict the instantiation of multiple database connections at runtime.

8.2.70 Collaborations:

Collaborates with Pipeline to save or load a developed pipeline into the database logs.

8.2.71 Attributes:

instance: SingletonDatabase conn: String host: String user: String password: String name: String

8.2.72 Operations

construct(): void getInstance(): SingletonDatabase getConnection(): conn

8.2.73 Constraints:

Restricts the database connection to a single connection.

9 Operational Scenarios

Use case diagram illustrated in fig7

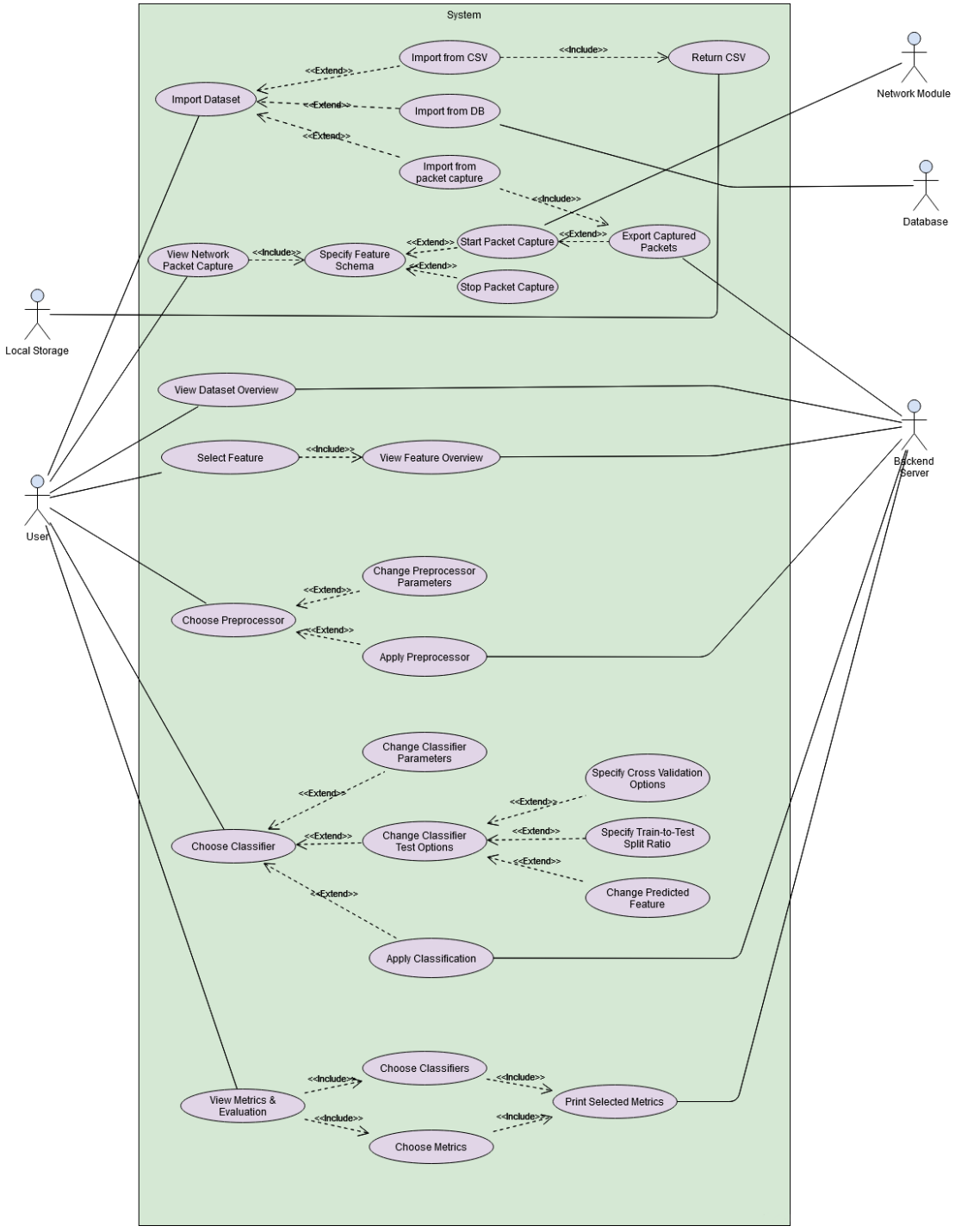

Figure 7: Use Case Diagram

10 Preliminary Schedule Adjusted

Primitive plan illustrated in gantt chart fig.8

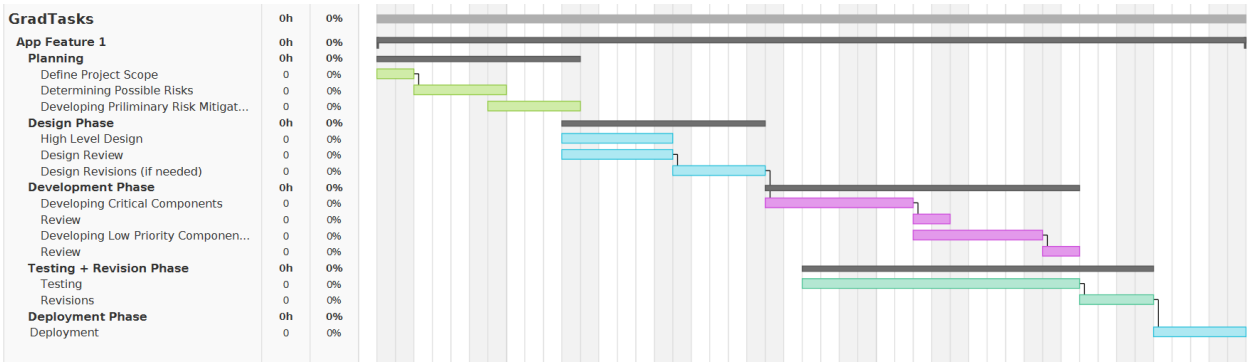

Figure 8: Primitive Plan

11 Preliminary Budget Adjusted

1. Development computers: to be excluded from budget
2. Research Publications: 2,500 EGP
3. AWS EC2: 500 EGP
4. AWS SageMaker: 300EGP
5. AWS S3: 150 EGP
6. AWS RDS: 150 EGP

12 Appendices

12.1 Definitions, Acronyms, Abbreviations

12.2 Collected material

13 References

References

Software Design Document for Machine Learning & Deep Learning Intrusion Detection Workbench System

Mohab Sameh Ibrahim
Supervised by: Dr. Ayman Ezzat & Eng. Ahmed Neil

March 28, 2021

1 Introduction

1.1 Purpose

This software design document describes the architecture and system design of Machine Learning & Deep Learning Intrusion Detection Workbench System.

1.2 Scope

The developed system is a workbench system which aims to provide a platform that supports users to perform machine learning and deep learning tasks on cyberattack-related data. The system shall enable users to execute their needed pipelines with ease; those of which depend on providing flexible use of different datasets, preprocessing methods, visualization techniques, classification models, performance metrics, and analytical comparisons.

The objective of the system is to ease the development of such pipelines needed by the system's audience & users and provide a hassle-free, all-in-one package for machine learning and deep learning intrusion detection systems benchmarking, testing, & development.

1.3 Overview

The system's main components are: the input component, preprocessing component, classification component, and metrics components. All of which communicate in a consecutive manner to provide the user with the needed workflow platform for machine learning and deep learning intrusion detection systems development.

The data input layer enables either importing datasets which have previously been developed by 3rd parties, or developing data through the live capture of network packets and deriving a set of features based on predefined schemas. To add, the preprocessing components supports execution of various preprocessing methods on the input data in order to aid with tasks such as data cleansing, transformation, and reduction. Moreover, the classification component supports executing of a wide variety of predefined classifiers or importing a user-defined classification model; such classifiers can be used to classify the network data and perform predictions on the existence of attack packets and their types within the processed network data. Finally, the metrics component provides an elaborate report of performance results of the utilized classification models and a comparative analysis of the results to aid with decision processes.

1.4 Reference Material

1.5 Definitions and Acronyms

Term	Definition
Software Design Document (SDD)	Used as the primary medium for communicating software design information.
Design Entity	An element of a design that is structurally and functionally distinct from other elements.
ML	Machine Learning
DL	Deep Learning
IDS	Intrusion Detection Systems

2 System Overview

The system's main components are: the input component, preprocessing component, classification component, and metrics components. All of which communicate in a consecutive manner to provide the user with the needed workflow platform for machine learning and deep learning intrusion detection systems development.

3 System Architecture

3.1 Architectural Design

The system is composed of four different components, which are data input, preprocessing, classification, & metrics, that directly support the pipelines carried out by the users. The data input layer enables either importing datasets which have previously been developed by 3rd parties, or developing data through the live capture of network packets and deriving a set of features based on predefined schemas. To add, the preprocessing components supports execution of various preprocessing methods on the input data in order to aid with tasks such as data cleansing, transformation, and reduction. Moreover, the classification component supports executing of a wide variety of predefined classifiers or importing a user-defined classification model; such

classifiers can be used to classify the network data and perform predictions on the existence of attack packets and their types within the processed network data. Finally, the metrics component provides an elaborate report of performance results of the utilized classification models and a comparative analysis of the results to aid with decision processes.

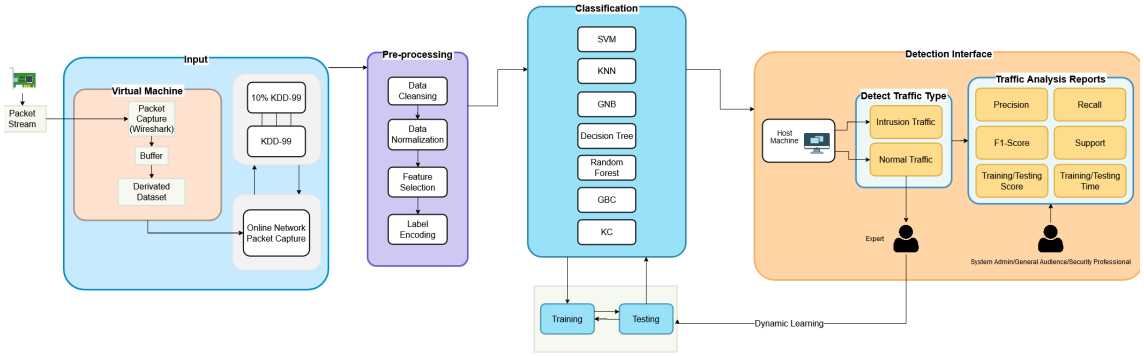

Figure 1: Architectural Design

3.2 Decomposition Description

Top level data flow diagram (DFD) illustrated in fig.2, Subsystem Model in fig.3, Sequence Diagram in fig.5

3.3 Design Rationale

The architectural structure illustrated in 3.1 has been selected after detailed review of the needed operations and tasks within the pipelines of machine learning and deep learning intrusion detection systems. Such operations and tasks have been identified through a review of the academic research done to-date in the field of anomaly-based IDS.

Moreover, the selection of ML and DL algorithms utilized within the preliminary system architecture, as illustrated in 1 within the classification phase, are selected based on the algorithm's ability to provide accurate performance based on precision, recall, f1-score, and support as well as the ability to provide timely performance based on the training time and testing time, and finally operating without overfitting and underfitting the given input training and testing data.

4 Data Design

4.1 Data Description

The input data directly reflects the structure of intrusion detection datasets, which are composed of features and packet instances (rows). The input data is first received as a directory string with resides either on a local storage or a cloud storage media. Then the input data module's controller invokes the importDataset() method in the controller, which will fetch the passed directory and create a dataframe that instantiates the dataset's features and rows into the dataframe within the application's

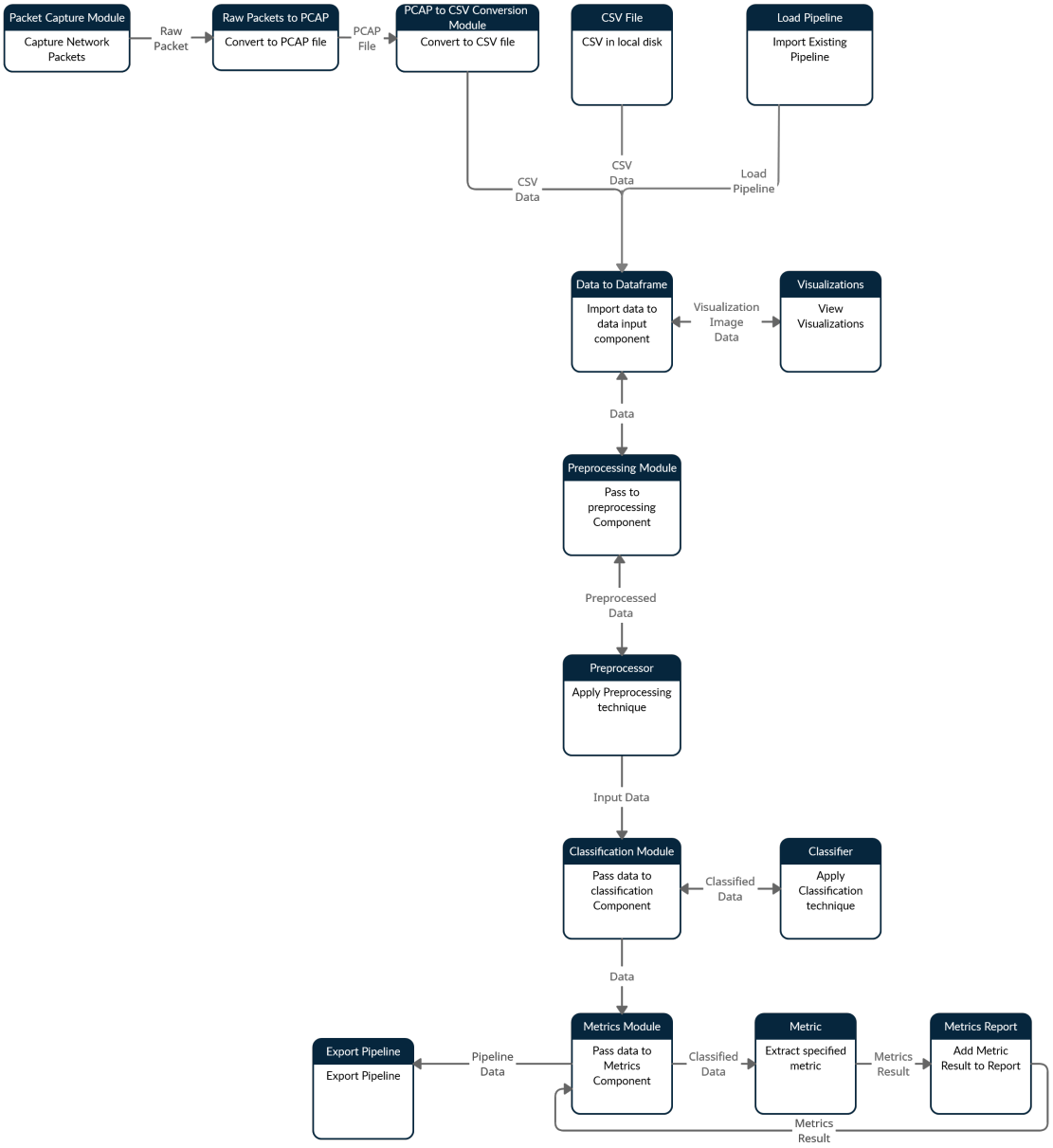

Figure 2: Top Level Data Flow Diagram

memory. Therefore, enabling the preprocessing and classification modules to read & manipulate the dataframe as necessary.

4.2 Data Dictionary

DB illustrated in fig.6

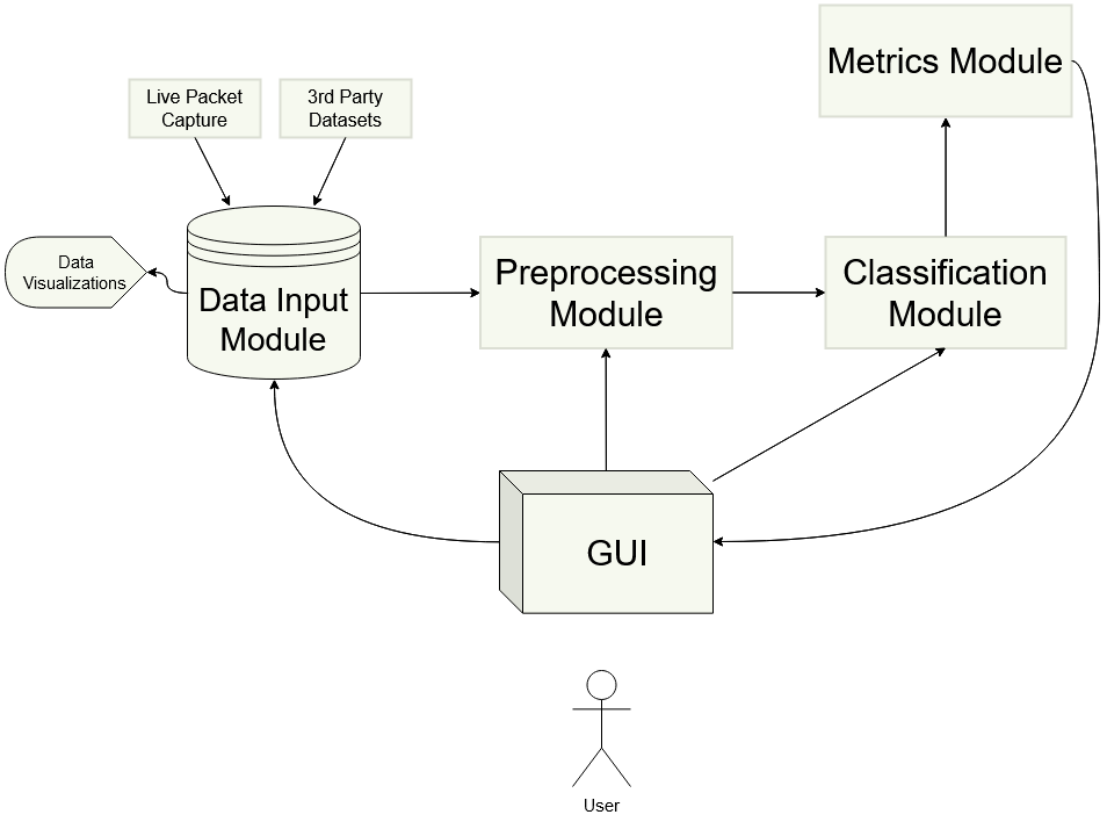

Figure 3: Subsystem Decomposition

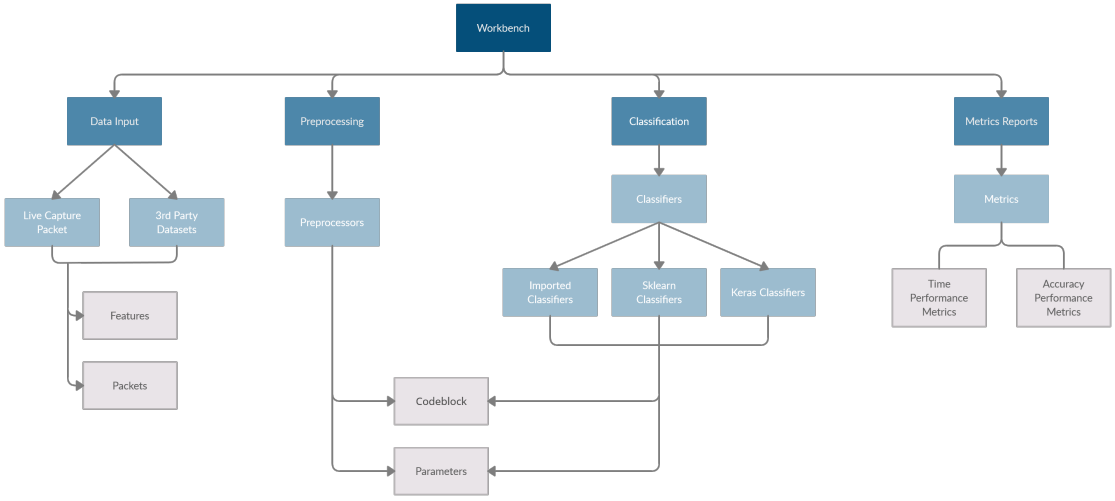

Figure 4: Generalization Hierarchy

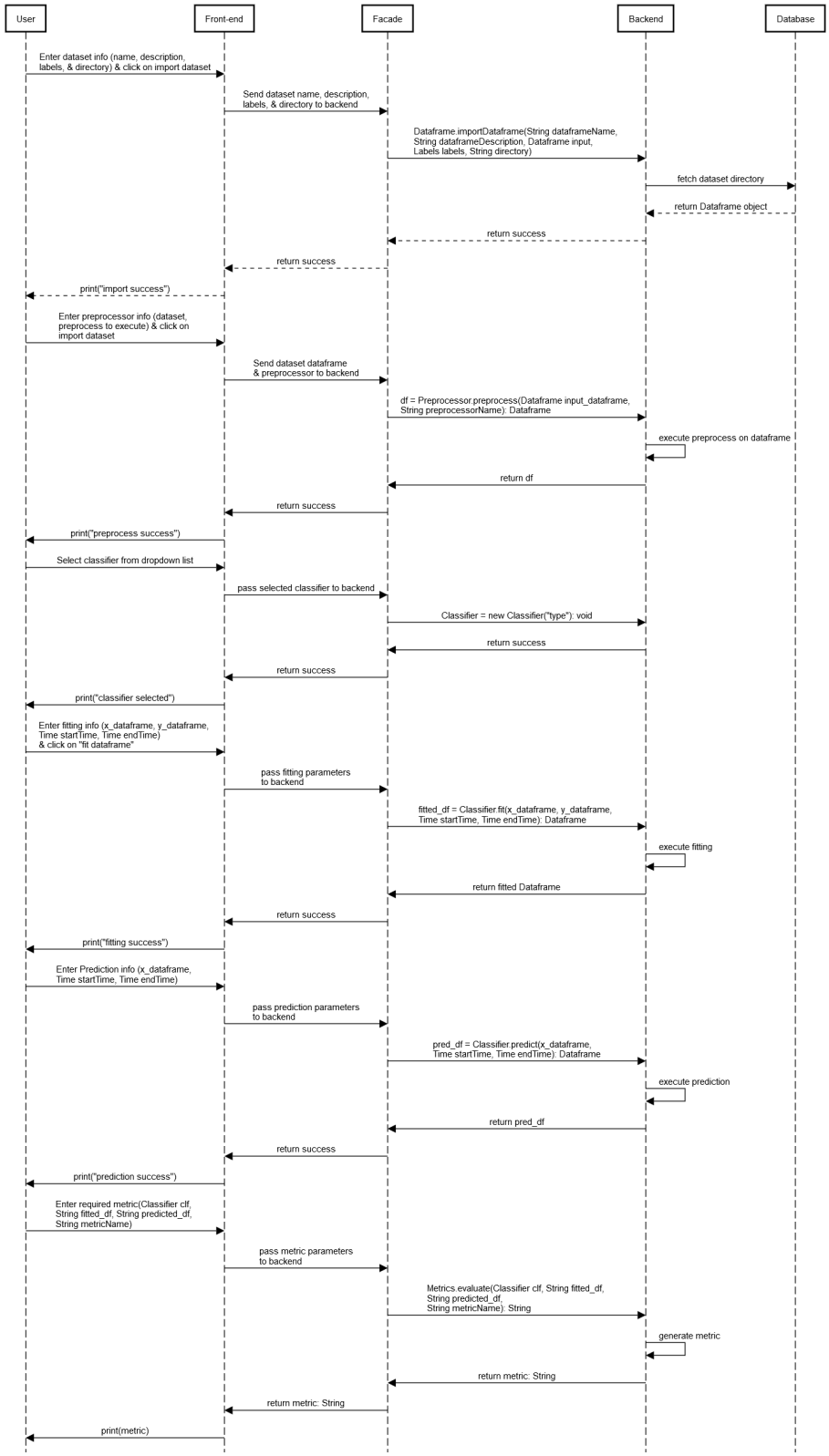

Figure 5: Sequence Diagram

Figure 6: Database Design

4.2.1 Object

Classifier

4.2.2 Attributes

id: int
classifierName: String
classifier Description: String
codeblock: String
params: Parameter[0..*]
startTime: Date
endTIme: Date

4.2.3 Methods & Parameters

classifierFit(x_dataframe, y_dataframe, Date startTime, Date endTime, Parameter[n] params): Dataframe
classifierPredict(x_dataframe, Date startTime, Date endTime, Parameter[n] params): Dataframe
getParameters(): Parameter[0..*]
updateParameters(Parameter[n] params)

4.2.4 Object

ClassifierFactory

4.2.5 Attributes

factory: Preprocessor

4.2.6 Methods & Parameters

createClassifier(String classifierName)

4.2.7 Object

Dataframe

4.2.8 Attributes

id: int
datasetName: String
datasetDescription: String
attributes: String[0..*]
numAttributes: int
numRows: int
classes: String[0..*]

numClasses: int
datasetDirectory: String

4.2.9 Methods & Parameters

importDataset(String datasetName, String datasetDescription, String directory)
getID(): int
getDatasetName: String
getDatasetDescription: String
getDatasetName(): String
setID(int id): void
setDatasetDescription(String description): void

4.2.10 Object

Facade

4.2.11 Methods & Parameters

Dataframe.importDataframe(String dataframeName, String dataframeDescription, Dataframe input, Labels labels)
PreprocessorFactory.getPreprocessor(Preprocessor prc): Preprocessor
Preprocessor.run(Dataframe input_dataframe): Dataframe
ClassifierFactory.createClassifier(String classifierName): Classifier
Classifier.classifierFit(x_dataframe, y_dataframe, Date startTime, Date endTime, Parameter[n] params): Dataframe
Classifier.classifierPredict(x_dataframe, Date startTime, Date endTime, Parameter[n] params): Dataframe
MetricsReport.addMetric(Classifier clf, Dataframe fitted_df, Dataframe predicted_df. String metricName, Metric[n] metrics): void
MetricsReport.Metric[].getEvaluationResult(): String
logPipeline(int id, String pipelineName, Dataframe dataframe, Preprocessor[n] preprocessors, Classifier[n] classifiers, MetricsReport[n] metricsReports, SingletonDatabase db): void

4.2.12 Object

Metric

4.2.13 Attributes

id: int
metricName: String
metricDescription: String
evaluationResult: String

4.2.14 Methods & Parameters

evaluate(Dataframe fitted_df, Dataframe predicted_df, Metric mtr): String
setEvaluationResult(String eval): void
getEvaluationResult(): String

4.2.15 Object

MetricsReport

4.2.16 Attributes

id: int
metrics: Metric[0..*]

4.2.17 Methods & Parameters

addMetric(Classifier clf, Dataframe fitted_df, Dataframe predicted_df. String metricName, Metric[] metrics): void

4.2.18 Object

Parameter

4.2.19 Attributes

id: int
parameterName: String
parameterDescription: String
type: String
value: type

4.2.20 Methods & Parameters

getID(): int
getName(): String
getDescription(): String
getType(): String
getValue(): type
setValue(Type value)

4.2.21 Object

Pipeline

4.2.22 Attributes

id: int
pipelineName: String
dataframe: Dataframe
preprocessors: Preprocessor[0..*]
classifiers: Classifier[0..*]
metricsReports: MetricsReport[0..*]
db: SingletonDatabase

4.2.23 Methods & Parameters

addDataset(Dataframe df): void
addPreprocessor(Preprocessor[] preprocessors, Preprocessor preprocessor): void
addClassifier(Classifier[] classifiers, Classifier classifier): void
addMetricsReports(MetricsReport[] metricsReports, MetricsReport metricsReport): void
logPipeline(int id, String pipelineName, Dataframe dataframe, Preprocessor[n] preprocessors, Classifier[n]
classifiers, MetricsReport[n] metricsReports, SingletonDatabase db): void
db.getInstance(): SingletonDatabase

4.2.24 Object

Preprocessor

4.2.25 Attributes

id: int
preprocessorName: String
preprocessorDescription: String
codeblock: String
params: Parameter[0..*]

4.2.26 Methods & Parameters

run(Dataframe inputDataframe): Dataframe
getParameters(): Parameter[0..*]
updateParameters(Parameter[n] params)

4.2.27 Object

PreprocessorFactory

4.2.28 Attributes

factory: Preprocessor

4.2.29 Methods & Parameters

createPreprocessor(String preprocessorName)

4.2.30 Object

SingletonDatabase

4.2.31 Attributes

instance: SingletonDatabase
conn: String
host: String
user: String
password: String
name: String

4.2.32 Methods & Parameters

construct(): void
getInstance(): SingletonDatabase
getConnection(): conn

5 Component Design

Subsystem Decomposition (fig.3), Generalization Hierarchy (fig.4), and Sequence Diagram (fig.5) are illustrated below.

The subsystem decomposition diagram illustrates the functional responsibility of each subsystem; where the facade layer logically encompasses all the subsystems to the user, the data input subsystem handles importing packet data either through a ready-made dataset or through a live capture of network packet flow, then the input data is passed to the preprocessing module where necessary preprocessing methods are able to manipulate the input data, then the preprocessed data is passed on to the classification module where a wide variety of classification models are available on demand, finally the metrics module is able to extract various evaluation results than is passed to the facade layer.

6 Human Interface Design

6.1 Overview of User Interface

Upon opening the application, the user directed by default to the Data Input and Preprocessing tab, where the user is able to import a data input source either through "Open File" to import a ready CSV file, "Open Database" to import a table data from a database, "Import Packet Capture" to import the packets captured within the "Live Packet Capture" module, or by loading a previously saved

pipeline which automatically loads that pipeline's dataset as well. Upon importing a dataset, the application will preview an overview about the dataset's information.

Then the user is able to view the features within the dataset, and click on any desired feature to be presented with an overview about the selected feature, which includes the feature name, distinct values, data type, missing value percentage, and unique value percentage. In the case of the feature being of a numeric type, the overview shall also include a table which displays the feature's minimum value, maximum value, mean value, and standard deviation. Otherwise, in the case of the feature being of a nominal type, the overview shall show the feature's each distinct label and the distinct value count of that label.

Then the user is able to choose a preprocessing method from the "Choose Preprocessor" dropdown list, then apply a preprocessing technique, where the preprocessing technique will change the previewed feature set's characteristics.

Moreover, in the Live Packet Capture module, the user is able to select a desired packet capture feature schema, start/stop capturing packets, view the captured packets' details, and export the captured packets to a CSV file.

In the classification tab, the user is able to select a classifier from the available classifiers within the "Choose Classifier" dropdown list, then the user is able to click on the parameter text field to specify the classifier parameters as required, where each classifier would preview its own cusotmizable parameters. Furthermore, the user is change the test options, where the user can determine the cross validation folds, a train-to-test split ratio, and the predicted label. Finally, the user is able to press on start/stop to start or stop the classification task; in case of an execution of a classification task, the output of the classification task is previewed in the "Classifier Output" and the classifier will be listed within "Results List".

Finally, the user is able to preview the Metrics tab, which allows the user to select a set of the previously performed classification tasks and a set of metrics, then the user will be presented with a comparative output result of the specified items within the "Metrics Output" section.

6.2 Screen Images

Screen images illustrated in fig.7, fig.8, fig.9, fig.10

6.3 Screen Objects and Actions

A discussion of screen objects and actions associated with those objects.

Fig. 7 shows the packet capture module, where the user can first select from the set of defined feature sets based on a dataset's feature set. Then the user can click the start button to start capturing network packets, where the captured packets will appear on the right output viewer. Then the user can click on the stop button when they are satisfied with the number of captured packets. Finally the user is able to click export and specify a directory within the directory browser and save the exported CSV file.

Fig.8 shows the Data input and preprocessing module. The user is first able to input data through either one of the available options (open file, open database, import packet capture, load pipeline). Open file enables the user to open a dataset

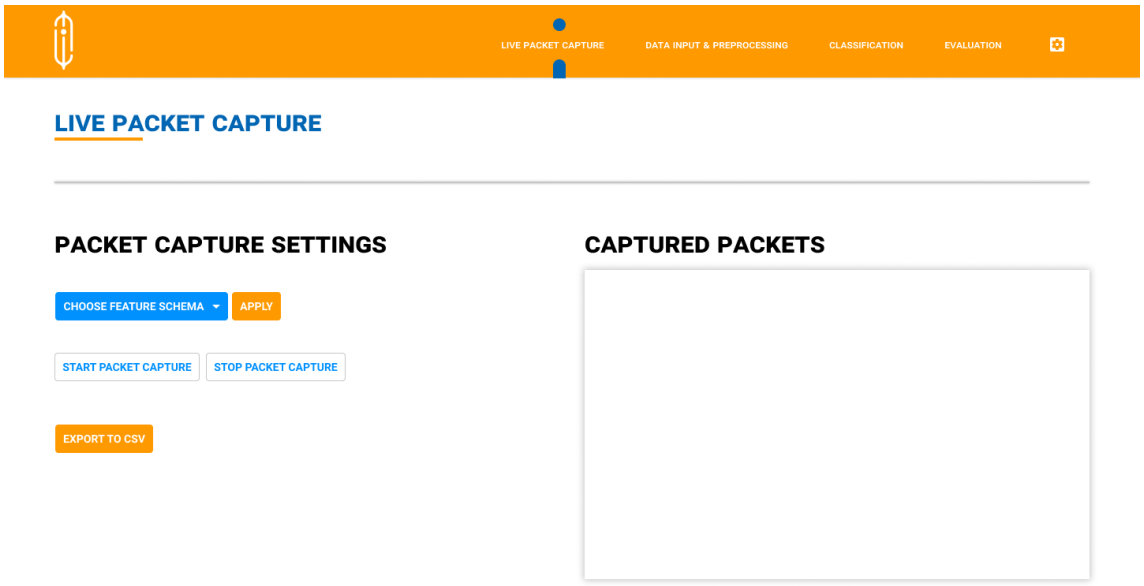

Figure 7: Live Packet Capture screenshot

CSV file, open database enables the user to load attribute and row data through a specified database table, import packet capture enables the user to directly import the packets captured in the packet capture module, finally load pipeline option enables the user to import a complete pipeline that has been developed and saved before, including its used dataset at the time.

Within the same tab, the preprocessing module is available, where the user is able to browse the dataset's overview (name, number of features, instances within the dataset). To add, the user is able to view the dataset's features within the "Features" section, and upon clicking on any feature, the "Selected Feature" section shall show overall statistics about the instances within this feature. To add, a visualization chart about the feature's statistics shall show within the bottom right visualization window; this visualization window changes upon clicking on another feature within the "Features" section. Finally the user is able to click on the "Choose Preprocessor" dropdown list then press on "Apply" to apply a specific preprocessing technique to the feature set. Pressing on the parameter text bar after choosing a preprocessor will allow the user to directly change the preprocessor's parameters if required.

Fig9 shows the classification tab, where the user is able to press "Choose" to choose a specific classification model or define a custom classifier. Pressing on the parameter text bar after choosing a classifier will allow the user to directly change the classifier's parameters if required. The "Test Options" section enables the user to change the testing options such as the training-to-test split ratio or applying cross validation with the number of required folds. To add, "Nom Label" allows the user to specify the predicted label (it's set as the last label by default) if the user wishes to predict a different label. Pressing "Start" or "Stop" will activate/deactivate the classification task. The classification output will be previewed within the "Classifier

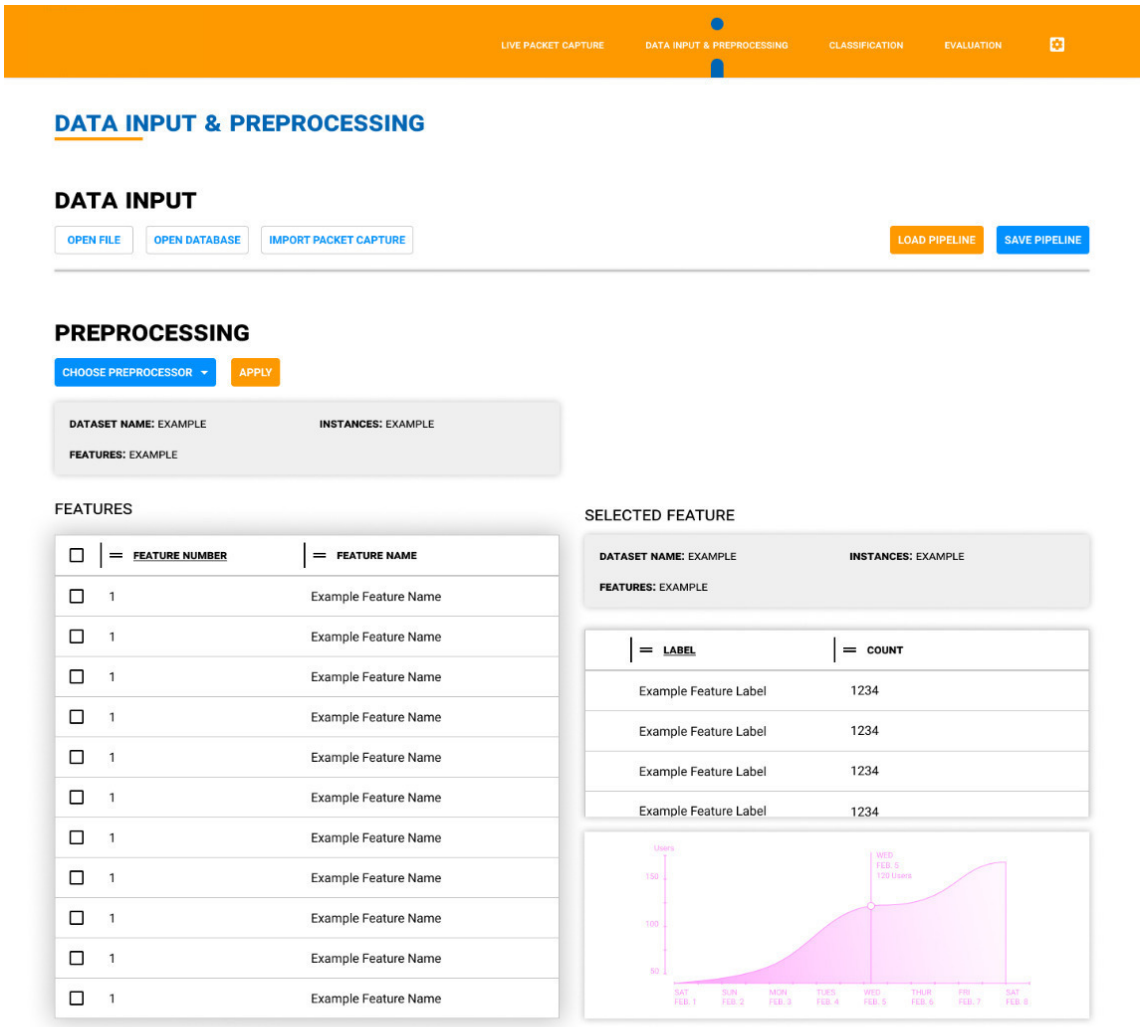

Figure 8: Data input & Preprocessing screenshot

Output" window. Finally, previously executed classification tasks will show within the results list, which enables the user to inspect different classification tasks and their results.

Fig10 shows the evaluation tab, where the user is able to choose a specific metric and any set of performed classification tasks and compare their results and extract analytical report. The "Choose" button enables the user to specify the required metric/analysis technique, while the "Results List" enables the user to select the required classifiers. The Output window will show all the specified metrics and analytical results of the specified items.

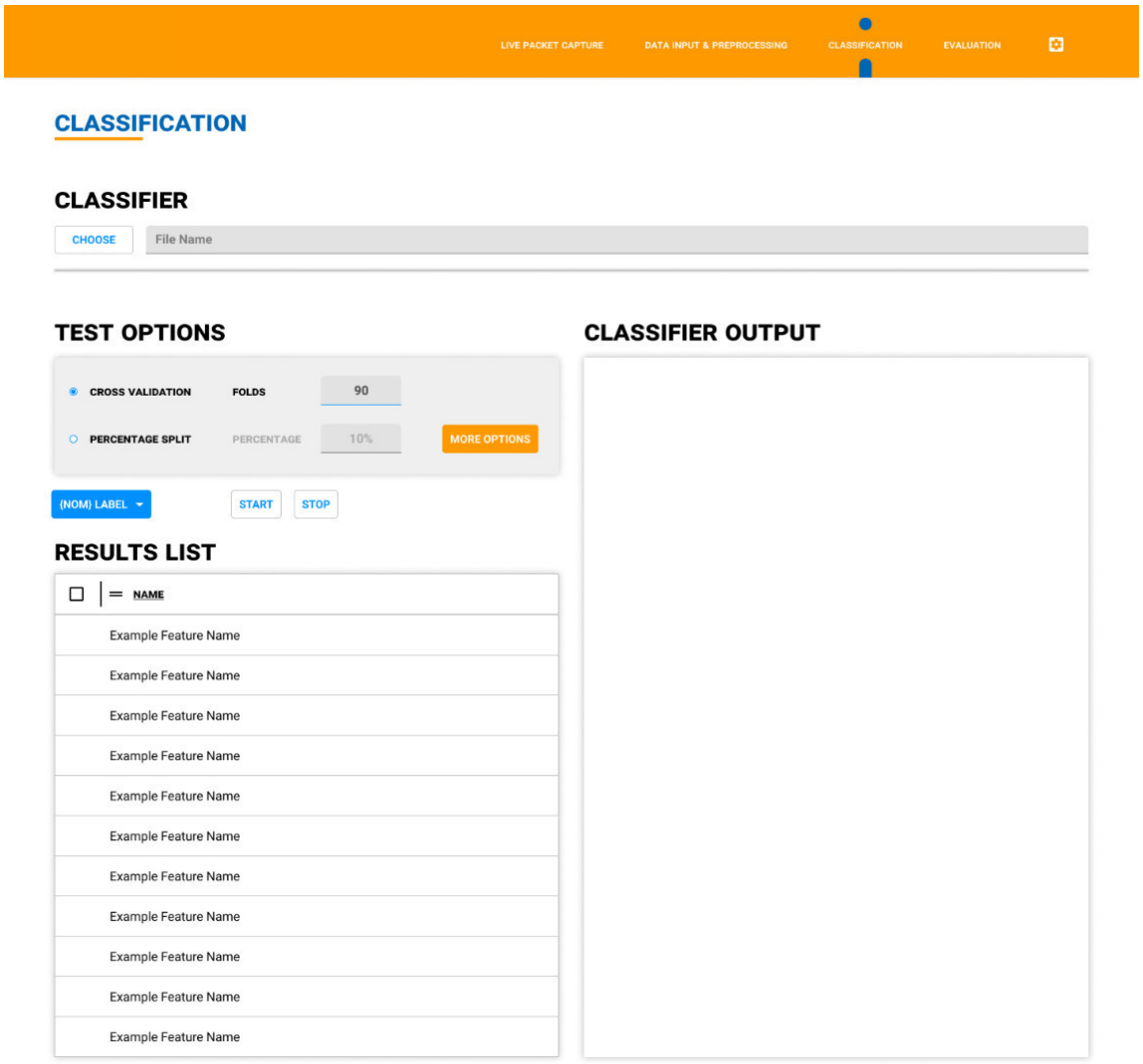

Figure 9: Classification screenshot

7 Requirements Matrix

Illustrated in Table.1

8 APPENDICES

9 References

References

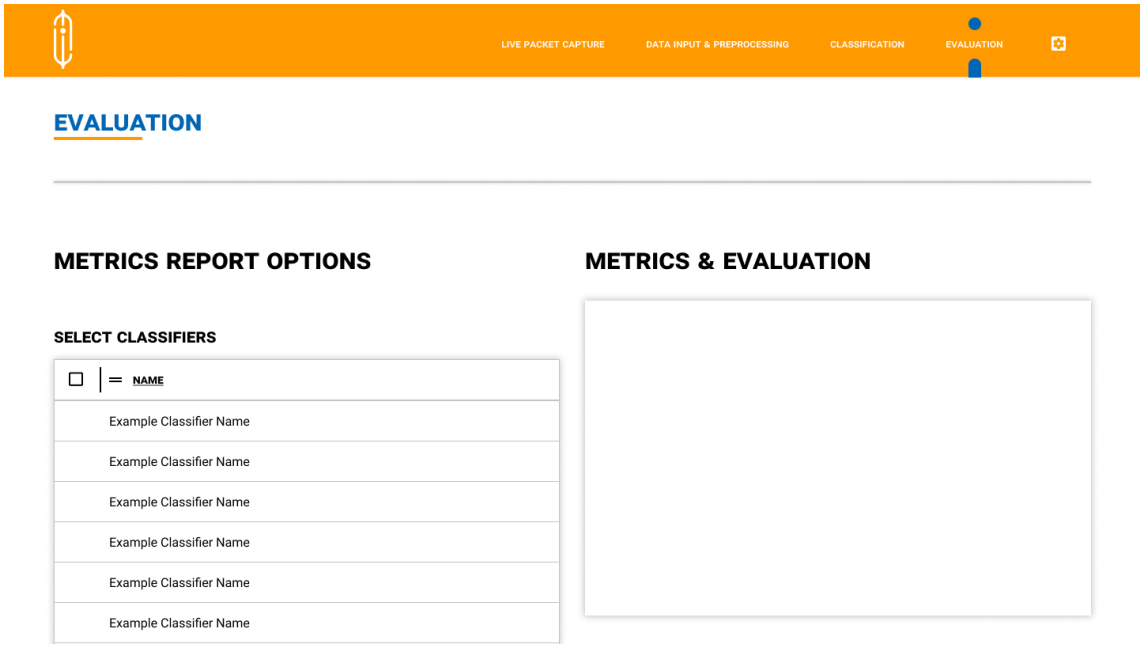

Figure 10: Evaluation screenshot

Requiremnt Number	Requirement Name	Testcase ID	Status
1	Import Dataset	TC01	TC01 - No Run
2	Select Packet Capture Feature Set	TC02	TC02 - No Run
3	Capture Packets	TC03	TC03 - No Run
4	Export Captured Packets	TC04	TC04- No Run
5	Load Pipeline	TC05	TC05- No Run
6	Save Pipeline	TC06	TC06- No Run
7	Show Dataset Overview	TC07	TC07- No Run
8	Show Dataset Features	TC08	TC08- No Run
9	Show Selected Feature Overview	TC09	TC09- No Run
10	Show Selected Feature Visualization	TC10	TC10- No Run
11	Choose Preprocessor	TC11	TC11- No Run
12	Change Preprocessor Parameters	TC12	TC12- No Run
13	Apply Preprocessing Technique	TC13	TC13- No Run
14	Choose Classifier	TC14	TC14- No Run
15	Change Classifier Parameters	TC15	TC15- No Run
16	Change Classifier Test Options	TC16	TC16- No Run
17	Apply Classification Model	TC17	TC17- No Run
18	View Classification Task Output	TC18	TC18- No Run
19	View Previous Classification Tasks	TC19	TC19- No Run
20	View Classifiers Metrics Report	TC20	TC20- No Run

Table 1: Requirement Traceability Matrix