Swan: A Framework to Bootstrap Trust in Network Data Science

by

ABDUARRAHEEM ELFANDI

A thesis accepted and approved in partial fulfillment of the

requirements for the degree of

Master of Science

in Computer Science

Thesis Committee:

Ram Durairajan, Chair

Michal Young, Member

University of Oregon

Winter 2024

THESIS ABSTRACT

Abduarraheem Elfandi

Master of Science in Computer Science

Title: SWAN: A Framework to Bootstrap Trust in Network Data Science

Two significant challenges must be overcome before machine learning models can be deployed in an operational setting: the ability to achieve trust within and across enclaves which includes addressing data privacy concerns. In this thesis, we propose **SWAN**, a framework to tackle these challenges by allowing data to be labeled at scale, achieving trust within an enclave by providing insight into black-box machine learning models through a hybrid explainability technique which is done by utilizing the combination of global and local interpretability techniques. Furthermore, the framework allows for collaboration across enclaves while maintaining data privacy requirements.

This thesis includes unpublished co-authored material by Ramakrishnan Durairajan and Walter Willinger.

ACKNOWLEDGEMENTS

I would like to thank the co-authors Ramakrishnan Durairajan and Walter Willinger for their collaboration in this thesis. I also would like to thank Ramakrishnan Durairajan for patiently guiding me and supporting me throughout the past year-and-a-half. Additionally, I would like to thank Professor Michal Young for providing very insightful feedback. Finally, I would like to thank my family for their continuous support throughout the years.

TABLE OF CONTENTS

Chapter Page

LIST OF FIGURES

Figure Page

LIST OF TABLES

CHAPTER I

INTRODUCTION

Ensuring the success of ML4Nets—defined as the application of artificial intelligence (AI) and machine learning (ML) techniques to address real-world network security and performance problems—in practice hinges on convincing network operators to deploy ML4Nets solutions in their production networks. This, in turn, requires that network operators can trust these solutions and have means to assess their safety. Here, following [15], we say "a network operator has trust in a ML model" iff "the operator is comfortable with relinquishing control to the model." Moreover, referring to [6], by "assessing the safety of ML solutions", we mean "studying the problem of accidents, defined as unintended and harmful behavior that may emerge from poor design of real-world ML solutions." Unfortunately, due to the black-box nature of many of the currently considered ML models, today's network operators lack the means to reason about the decisions and predictions made by these models. These models' inability to provide explanations for their decision-making engenders distrust, prevents network operators from understanding the models' safety, and explains the operators' overall reluctance to using ML4Nets solutions in practice [12].

To address these issues, Explainable AI (XAI) has emerged as a field of study aimed at enhancing the comprehensibility of learning models and their decision-making processes (e.g., see the surveys [8, 5, 7]). At a high level, XAI encompasses two categories of techniques. The first category consists of *global explainability* techniques that leverage approximations in the form of explainable models to provide an overall explanation of a given black-box model and typically entail a tradeoff between the complexity (e.g., size) of the explainable model, its

accuracy (e.g., number of input instances it explains), and the computational effort its generation requires, In theory, using such explainable approximation models, operators can reason about a given black-box model's decision-making (i.e., how and why the model arrives at a specific decision and not at some other decision) and gain confidence in the overall reliability of its decisions and predictions (i.e., when the model works or when and why the model does not work). The second category is composed of *local explainability* techniques that are typically designed to provide explanations for individual instances that a trained model is given as input. Local techniques employ concepts such as feature importance scores, attention mechanisms, and rule-based explanations, and applying them at scale (number of instances) requires being aware of their per-instance computational complexity. These local techniques are useful vehicles for operators to reason about a given model's specific decisions or predictions and to assess the safety of a given ML4Nets solution in corner case scenarios (i.e., understanding the potential consequences of certain incorrect decisions for a given input instance).

While these techniques have been successfully applied in a number of different application domains (e.g., computer vision [9] and autonomous vehicles [1, 3]), their suitability and effectiveness in the networking domain to address network performance and security problems of practical interest have attracted little to no attention to date, mainly because of data-related issues that are specific to networking [13, 10, 12]. Most critical among these issues are a general paucity of (labeled) data, the high volume and velocity of network data collected from real-world production networks, the one-off nature of existing data collection efforts, and important privacy and security concerns associated with collecting network data from operational networks. Furthermore, faced with a

growing number available explainability techniques, network researchers and operators alike are largely left in the dark about how to apply the latest techniques so as to simultaneously satisfy the dual requirements of network operators—gaining trust in ML4Nets solutions (by means of having a broad understanding of the solution's global behavior) and being able to assess the solutions' safety (by means of providing specific, case-by-case, local explanations that can be scrutinized with respect to the impact of the associated decisions and predictions). Finally, meeting both of these requirements concurrently also necessitates systems innovations that are aimed at striking a balance between the indiscriminate use of resource-intensive local explainability techniques and the selective application of efficient but inaccurate approximation models supplied by global explainability techniques.

The second challenge stems from data-privacy issues considering the nature of networking data containing sensitive information. The nature of networking data causes data owners to impose privacy requirements making it difficult to collaborate across different enclaves. Recent efforts have made progress on each one of the these capabilities individually. For example, Yin et al. [20] developed a Generative Adversarial Network (GAN)-based to share synthetic datasets across different enclaves. Knofczynski et al. [12] proposed a local interpretability technique for bootstrapping trust into trained ML models based on network data. Although these individual efforts are commendable, a solution to simultaneously address the aforementioned factors remains an open problem.

In this thesis, we propose to tackle two challenges, ensuring that the ML models can trusted within enclaves (i.e., operational network) and across different enclaves by creating **SWAN**, a framework that operators and researchers can use a pipeline in which different enclaves can share metadata, and benefit from datasets

that exist outside of their own enclaves while ensuring data privacy. Additionally, it enables reasoning with decisions made by the trained black-box model using hybrid explainability techniques that combine global (i.e., insights about overall working on the model) and local (i.e., insights into specific predictions) explainability methods.

The code source code of **SWAN** is available here: `https://gitlab.com/onrg/swan`.

CHAPTER II

MOTIVATION AND PRIOR EFFORTS

In this section, we describe our motivation and limitations of prior efforts.

2.1 Motivation

The motivation behind this work comes from two important factors (1) maintaining privacy across different operators in different enclaves, and (2) explaining the black-box nature of ML models that plays a significant role in the application of machine learning for network performance and security problems.

2.1.1 Availability of labeled data. The lack of labeled data in the Network Data Science domain arises from the lack of defined and generally accepted features for describing various events in the data. This problem is made more complicated due to how the data is collected from different locations and during other conditions along with different information. To label this data, only an operator who is familiar with this data can label it.

Recent efforts have proposed solutions to the lack of labeled data by using weak-supervision-based learning. This is can done by having an operator provide ground truth labels. The amount of labels an operator can label is limited, hence weak supervision uses a finite amount of ground truth labels and generates noisy labels for the rest of the dataset. This can be done by using data programming methods such as the usage of labeling functions to label data. Operators can create labeling functions that divide the data into different categories.

Prior frameworks such as NoMoNoise [17], and EMERGE [13] use weak supervision learning to label networking data. NoMoNoise denoises internet delay measurements with the usage of weak supervision learning along with what Snorkel [18] offers. To swiftly and seamlessly eliminate and potentially correct noise data,

NoMoNoise is able to generate measurement noise labels. Furthermore, EMERGE is another weak supervised learning framework that extends on the ideas from NoMoNoise. EMERGE uses a generative model to produce weak labels on the data. These labels are then used to train a classifier discriminative model such as LSTM where to see the quality of the data.

2.1.2 Privacy. Due to the nature of networking data containing sensitive information, it is quite difficult for researchers to collaborate with different groups. One solution is to share machine learning models. With the data they have available, two groups can train their own ML models and then share the final model. Beyond sharing datasets and ML models there is not a framework that allows researchers and operators who don't own data to benefit from existing datasets while maintaining the owner of the data demands on privacy requirements. Because different groups are unable to give away the dataset information used for training ML models, or model predictions, as a result, these privacy concerns are a hindrance to research groups working together to use ML in the networking domain.

While sharing ML models would seem like an ideal way for different groups to collaborate with one another, ML models have no way of safeguarding the privacy of the data. This is due to multiple different types of attacks that ML models are vulnerable to. Typically, adversaries take advantage of these flaws to gain information about the data that was used to train the model. Depending on the information gathered from these attacks adversaries may be able to create datasets that reflect a realistic representation of the classes in the dataset.

2.1.3 Interpretability. Network operators in the networking domain face trust issues due to the black-box nature of ML models. Network

operators frequently require interpretability in the ML models they want to use since the decision made matters, allowing them to understand and support the decisions made by the models. Because of the explainability provided, network operators may make correct decisions based on the outputs of the ML model. This interpretability can come in two forms (1) local interpretability and (2) global interpretability. The former aims to provide insight into why the model made a particular prediction for a specific input, focusing on individual data points or instances. Local interpretability offers precise and detailed explanations for individual predictions but it does not provide the overall behavior of the model. On the other hand, the latter, global interpretability aims to provide insight into the model's behavior across the datasets or a significant portion of it along with analyzing the dataset's patterns, trends, and feature importance. As global interpretability deals with complicated models and attempts to summarize them into an easy-to-understand format, this process can be error-prone and may not accurately convey the model's behavior. Using the combination of both local and global interpretability we create a hybrid interpretability that provides the global context while also using local interpretability to provide error-free results.

2.2 Limitations of Prior Efforts

Prior efforts have two main shortcomings: (1) provide collaboration in a privacy-preserving fashion, and (2) the interpretability of ML models.

Although there have been prior efforts [13], [12] that attempt to democratize the use of ML to label networking data by providing low-cost and high-quality methods with the usage of data programming and multi-task learning techniques.

Prior efforts such as EMERGE [13] assume that networking data is present and focus on allowing collaboration by providing at a large scale high-quality and

low-cost, labeled data by building on the concept of weak supervision-based data labeling methods. EMERGE specifically leverages traceroute data from the CAIDA Ark project that spans one day to label data in a programmable manner. This process is done in multiple steps, where the data is analyzed to create thresholds to differentiate between noisy and non-noisy data. Afterward, the data is split and a small portion is labeled. To evaluate the label quality generated by EMERGE, discriminative models are trained using probabilistic training labels. Instead of sharing raw data and machine learning models to allow for collaboration, EMERGE allows metadata to be shared (e.g. labeling functions) to maintain privacy. The EMERGE framework addresses one of the two shortcomings but it does not provide any way to begin reasoning with the decisions that the model makes.

Similarly, ARISE [12] provides a multi-task weak supervision framework. ARISE applies various learning techniques including multitask learning and meta-learning to improve information exchange between tasks and shorten overall training time. It labels network data at scale using weak supervision-based data programming. ARISE provides local interpretability to understand the decisions that the model is making based on the data. ARISE is built upon multiple different components such as an interface that network operators can use to translate their understanding of the data to "a programmatic representation" by using labeling functions. Another component of ARISE allows unlabeled data to be inputted and used with labeling functions to generate weak labels. ARISE then uses the weakly labeled data to train a classification model which is done in different sub-tasks. One of the main issues of this framework is that it does not address anything in regard to collaboration and maintaining privacy among collaborators.

CHAPTER III

DESIGN AND IMPLEMENTATION

The hybrid explainability technique described in this chapter (3.2.3) was written by the co-authors Ramakrishnan Durairajan and Walter Willinger while I implemented the hybrid explainability technique.

In this section, we describe the design and implementation of **SWAN**.

3.1 Overview of SWAN

Designing and implementing **SWAN** is the primary goal of this work. **SWAN** is a novel framework that provides a technique to label network data in a scalable and cost-effective fashion by utilizing prior efforts [13], a pipeline in which network operators and researchers can share metadata and not share any sensitive data or ML model, and lastly provides a method to show the decisions made by trained ML models and how they compare to a domain expert along with how the trained ML model operates.

3.1.1 Design Details. The architecture of **SWAN** can be seen in Figure 1. Below, we list and provide a description for each of the modules.

3.1.2 User Interface. One of the key contributions of **SWAN**, is to provide a way for researchers and network operators to translate their domain knowledge and allow collaboration between different groups. In order to accomplish this, we have created an interface that is exposed to researchers and operators that may be in different groups to use by providing programs in this case a labeling function using Python. This interface exposes the EMERGE pipeline which can use the labeling functions provided to generate labels for that dataset. This interface also allows users (researchers and operators) to fine-tune the hyperparameters for the generative and discriminative models along with the ability to specify

what dataset (e.g CAIDA & RIPE) they would like to train their model on. Furthermore, the interface allows users to view other users' submissions along with how their labeling functions perform. Using this information different groups can decide if they would like to train a model by using a combination of labeling functions provided by them and by others and how they perform in comparison.

3.1.3 Labeling function. In order to provide weak labels for unlabeled networking datasets that are used in ML model training, users (researchers & network operators) can create labeling functions as in Figure 2. This labeling function in Figure 2 implements a rule, that classifies data as good (1) or bad (0) if the RTT value is less than or equal to the mean. While labeling functions can be customized for a particular enclave, they can also include statistical or general thresholds that allow them to be used for different networks. Labeling functions can also be built in such a way that allows them to be more general across different datasets by taking into account different factors. Using this approach allows for more scalability by using the same labeling functions throughout different datasets.

3.1.4 EMERGE Pipeline. EMERGE [13] is a weak supervised learning framework that builds on NoMoNoise. The EMERGE pipeline provides a method to produce weak labels on the data, this is based on the NoMoNoise [17] module. This is done by using the given labeling functions that have been acquired from the user interface module. Using the weak labels produced and the Snorkel [18] library a generative model is produced. The generative model provides the probabilities values of the weak labels generated. To be more specific it assigns probability values to each RTT value which indicates how confident the model is

assigning a particular label to a given RTT value. The labels that are generated are then used to train the discriminative model (e.g LSTM).

3.1.5 Interpretability. Another key contribution that is provided in this framework is the ability to reason with decisions regarding the model by ensuring that the ML model is interpretable globally and locally. This done by providing a hybrid interpretability framework where we can view explanations for each data point and reason about why it was classified correctly or incorrectly.

3.2 Implementation

In this section, we describe the implementation of **SWAN**, which allows different entities to collaborate with each other while maintaining privacy. The framework shown in Figure 1 is built using three Docker containers: (1) a container that contains the Flask application; (2) a container for the DEEP as a Service (DEEPaaS) REST API to expose the model; (3) a container that contains the reverse proxy that is used, in this case, Nginx.

3.2.1 Web Service. Using the Flask micro web framework we present an endpoint that different entities can access to collaborate with each other. The Flask application presents users with a dashboard where new users can create an account to access the service. When a user creates their account their username and password are stored in a database (e.g. MongoDB). Upon signing in the user is presented with the ability to share their labeling functions, and select between two datasets (e.g. CAIDA and RIPE) to use for the generative to produce the labels which are used on the discriminative model. Along with that, a user can specify the hyperparameters for both generative and discriminative models.

A user can submit their labeling functions by providing a Python file that contains the functions. Furthermore, users can specify what labeling functions

they would like to use from the submitted file. We provide a template file that contains different labeling functions for users to understand how they should write their own labeling functions that they can share. Upon submission, we check if the provided file type is a Python file and if the specified labeling functions exist. If any of the conditions are not met or any error occurs that will be reported to the user. Otherwise, if all the inputs are valid we store all the submission information in our database and we make a call to the DEEPaaS REST API to train the model. Users can then view all their submissions with a drop-down menu to check if the training is currently running or completed and view the results. Users can view the results as a plot of F1 scores of all labeling functions that they have submitted which makes it easier to compare labeling functions.

To allow for collaboration between different users, a user can see if other registered users have shared their labeling functions. They can combine them with their own labeling functions to see how well they perform by checking the F1 score returned. Along with that, a user can compare how all of the labeling functions they have submitted perform, in comparison to what other users submitted, and how both sets of labeling functions perform when they are combined.

3.2.2 DEEP as a service REST API. Using DEEPaaS [16] a REST API is run in front of the model which can then be accessed using an HTTP request. DeepaaS provides a Swagger UI for this API. This allows you to visualize and interact with the API and the underlying model. In this work, the model that we expose is a portion of the EMERGE pipeline which is the module that extends ideas from the NoMoNoise [17] framework.

The two main endpoints that we expose to the user are the POST training request and the GET status of training. The former is used when a user submits

their labeling function or combines their labeling function with another user. When that POST request is sent a response is given back that contains the Universal Unique Identifier (UUID). The UUID is used to uniquely identify each training request. The latter leverages the UUID to provide users with the training status of the model using labeling functions that they have submitted. There are four possible statuses which are: (1) running, (2) error, (3) completed, (4) canceled. Users at any time can get the status to see if the training has been completed and view the results.

3.2.3 Hybrid Explainability. In this work, we present a novel hybrid explainability technique that is specifically designed to simultaneously satisfy network operators' dual requirements for deploying ML4Nets solutions—being able to trust these solutions and to assess their safety. At the core of this novel technique is the following three-step approach:

Step 1: Enhancing Explainable Model Accuracy. This step focuses on addressing accuracy issues that stem from using post-hoc global explainability techniques. These techniques include recently developed methods such as Trustee [11] or ARISE [12] and typically generate explainable models in the form of decision trees that approximate a given black-box models. Because of their approximate nature, the generation of such decision trees entails a complexity-accuracy tradeoff whereby high-accuracy decision trees necessitate large-sized (i.e., high-complexity) tree structures. To navigate this tradeoff, we incorporate computationally intensive yet highly accurate local explainability techniques in an opportunistic manner. Specifically, we enhance the explainability of certain branches of the approximate decision-tree model only if it results in improved accuracy. We determine this improvement through a straightforward majority

voting mechanism that is aimed at resolving uncertainties or inconsistencies stemming from the fusion of global and local explainability techniques. This step essentially acts as a "model distillation" process, simplifying complex interpretations generated by various techniques and consolidating them into a more concise explanation. Moreover, the employed voting mechanism prevents unnecessary computational overhead when multiple techniques are in agreement.

Step 2: Handling Exceptional Cases. In this step, we address situations where global techniques fail to provide an explanations or don't produce a meaningful explanation. Such situations are common for training data collected from operational networks [13, 12], but network operators nevertheless want to be able to reason about a black-box models' decision-making process when faced with such "corner cases." For each data point in the test data that cannot be explained by any of the branches of the decision tree generated in Step 1, we apply the same majority voting mechanism used earlier, but this time exclusively with findings derived from applying local explainability techniques. This approach saves computational resources by only applying local techniques to only a subset of the test data (i.e., corner cases). It also avoids unnecessary resource overhead when there is consensus among the employed local techniques. At the end of this step, we compile a comprehensive list of corner cases, complete with rule-based explanations, potentially adding new branches to the decision tree generated in Step 1.

Step 3: Expanding the Global Decision Tree. The final step involves integrating the new branches that emerge from the corner cases considered in Step 2 into the global decision tree constructed in Step 1. The goal is to expand this tree while ensuring that both the existing and new branches are integrated cohesively. In efffect, this step serves as an "explanation summarization" process, where

interpretations are harmonized to facilitate engendering trust, assessing safety, and guaranteeing computational efficiency. To determine the proper placement of each of the corner cases into the already existing decision tree, we start by examining the nodes of the decision tree where the conditions of the new branches (i.e., corner cases) align with the those in the existing decision tree. We then check if any existing nodes can accommodate the new branches with some modifications. If so, we update the nodes of the decision tree to incorporate the rules from the corner cases. Otherwise, we create new nodes in the decision tree. These new nodes are added as child nodes to existing parent nodes in the decision tree, ensuring that the conditions of the new nodes are compatible with the rules of their parent nodes and that they lead to the intended outcomes. After applying this process to each considered corner case, we review the entire new set of branches to maintain the order of traversal. In operational networks, these trees have to be updated on an ongoing basis by means of real-world feedback and new training data to enhance their accuracy.

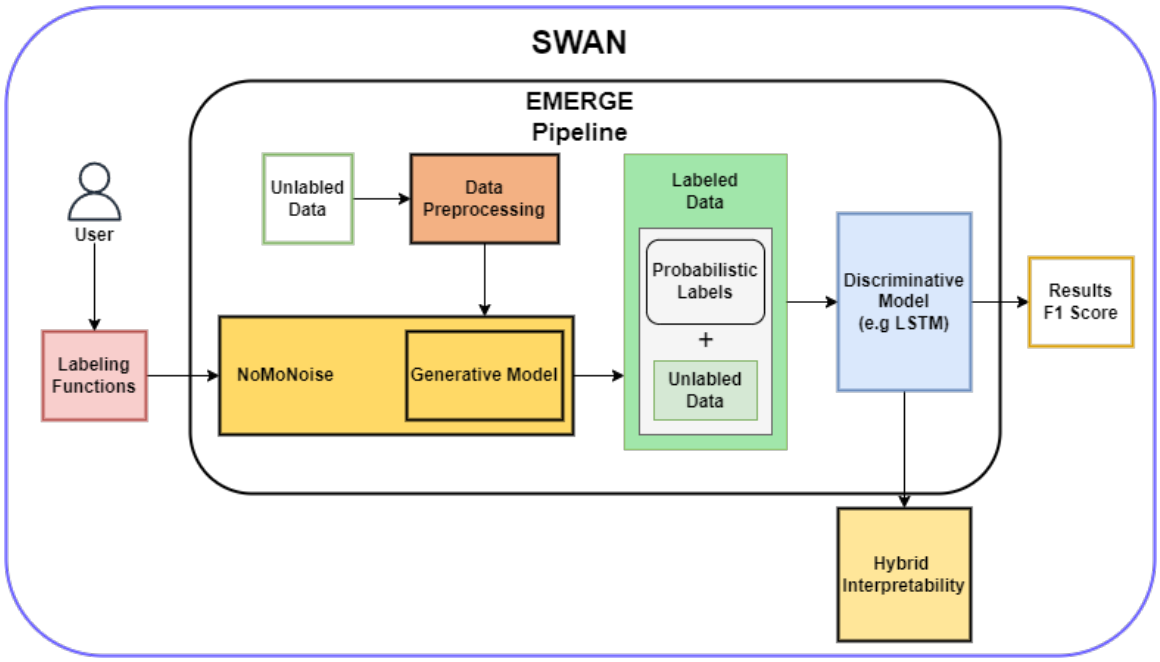

Figure 1. Pipeline of SWAN that includes the EMERGE [13] pipeline.

```
def LF_Mean(c):
    val = c.number1.get_attrib_tokens()
    if float(val[0])<=mean_threshold:
        return 1
    else:
        return 0
```

Figure 2. Example Mean labeling function.

CHAPTER IV

EVALUATION

The illustrative use case 1 and results for use case 1 that are provided in 4.2 were written by the co-authors Ramakrishnan Durairajan and Walter Willinger. I conducted all the experiments in addition to writing the results that were found when applying Steps 2-3 of the hybrid explainability technique.

4.1 Evaluation of collaboration

The following describes the steps for conducting the experiments along with the findings of how collaboration across different enclaves with the usage of sharing and combining labeling functions can result in benefits.

4.1.1 Datasets Used. The datasets used in our experiments we use CAIDA Ark [2] dataset which contains traceroute and RIPE Atlas project [4] dataset which contains ping measurements. The CAIDA Ark project dataset contains over 1 million traceroutes, and extracts around 75,359 round-trip time (RTT) measurements among 28 source and destination (SD) pairs [2] that are collected over a one-day period. Each source and destination pair contains 2,000 to 4,000 measurements, where the average number of measurements is 2692 for each source and destination pair. Along with that, we use a portion of RIPE Atlas [4] project ping dataset and use 2.5 million latency measurements that are split into 100 time series with numerous sources to numerous destinations. Every measurement contains IP addresses for the source and destination, the average RTT, along with timestamps.

4.1.2 Experiments and hyperparameter tuning. We analyze each SD dataset in order to determine a threshold value that is used to label that RTT value as normal or noise. Due to our data being imbalanced where there

are few noisy data we create random values that are greater than the threshold value, and add them into the original data. By doing this, we make sure that our classifier has sufficient normal and noisy samples to gain information. Since noisy RTT measurements in our dataset can be regarded as outliers we consider outlier detection methods such as Local Outlier Factor (LOF), Elliptic Envelope (EE), and Overly-Robust Covariance Estimation (ORCE). We split our data into 80% training set, 10% validation set, and 10% test set. Our training set will serve as our unlabeled data while the validation and test sets will serve as our labeled data. We use our generative model to create probabilistic training labels that are then used in our discriminative model (LSTM) which is our end-classifier. The LSTM model is trained using the training set and evaluated it on the validation and test set. The LSTM model hyperparameters are fine-tuned by trying different values of epochs, learning rate, batch size, and LSTM units.

4.1.3 Case 1 Results: Combination of labeling functions that are based on one feature. We present the F1 score of the labeling functions along with the classifier's F1 in Table 1 and 3. The feature that we consider is the Round Trip Time (RTT) value. The LF F1 evaluates the labels generated directly from the labeling functions compared to the ground truth labels, while the classifier F1 scores evaluate the classifier trained using the training set along with the probabilistic labels corresponding with the ground truth labels. In Table 1 we can see that some of the end-classifier model achieved a higher F1 score than the labeling functions F1 scores. Labeling functions may introduce noise or errors in their predictions due to being one-sided. By training a classifier on the noisy labeled data the model can learn to distinguish between noisy and good data, which can generalize and reduce the impact of noisy labeling functions. On the

other hand, taking a look at Table 3 none of the Classifier F1 scores out perform the LF F1 scores. This could be due to the Classifier overfitting the training data or failing to generalize well to the test data.

4.1.4 Case 2 Results: Combination of labeling functions that are based on two features. In this experiment, we analyze the effects of labeling function combinations based on two features. The two features that we consider are the Round Trip Time (RTT) value and the jitter. The RTT and jitter have a correlation coefficient of 0.526 and 0.629 for both the CAIDA and RIPE data set respectively and is viewed as moderate. In Tables 2 and 4 we provide the individual and combined F1 scores of the labeling functions based on the two features. We consider different outlier detection methods when we build our labeling functions. In both Tables 2 and 4 we see that both the LF and the Classifier F1 scores are quite low. This could be due to the low number of true positives, which affects the precision and recall which in the process results in a low F1 score. Due to some of the labeling functions being skewed that can make one of the labeling functions irrelevant. Additionally, the low LF and Classifier F1 scores could be due to the labeling functions not being suitable with the jitter feature. This could come from the design of the labeling functions which may not adequately account for the variability of the jitter feature.

4.2 Evaluation of Hybrid Explainability

4.2.1 Illustrative use case 1. To illustrate the practicality of our proposed hybrid explainability technique, we evaluate in the following the effectiveness of the above-described steps (3.2.3) in the context of ARISE, a previously-published weak-supervision-based framework for labeling different network datasets in an automated fashion and at scale [12]. In short, ARISE

leverages network operators' domain knowledge in the form of *labeling functions* to programmatically label network datasets and uses multi-task learning to enable concurrent learning of network classification tasks (e.g., congestion vs. non-congestion). Its workflow (see Appendix C in [12] for details) requires to first create a noisy generative model and then train a predictive LSTM model. We choose ARISE because of its ability to produce a decision tree that enables operators to reason about the labeling decisions made by ARISE. Here we show how to embellish this decision tree by applying Steps 1-3 of our proposed hybrid explainability technique.

4.2.2 Illustrative use case 2. Additionally, we illustrate the practicality of the proposed hybrid explainability technique using Trustee [11], a framework that provides insight into black-box learning models. To summarize, Trustee is a framework that extracts decision tree explanations from black-box ML models and verifies the model's trustworthiness. The workflow of Trustee requires a black-box model and the dataset used to train the model as input which then outputs a decision tree explanation. Specifically, we focus on one Trustee application for an intrusion detection system that is trained using CIC-IDS-2017 [19] dataset. We show how the decision tree is modified using our hybrid explainability technique.

4.2.3 Results for use case 1. For our evaluation, we use CAIDA's Ark dataset, which comprises over 1.2 million round-trip time (RTT) measurements between 28 source-destination pairs collected over the course of a day [2]. Using this dataset, we trained a predictive LSTM model by utilizing the labeling function that classifies a data point as "experiencing congestion" if the RTT value falls within the range of [1.2 times β, 1.5 times α], where α and β represent the RTT

values corresponding to the 75th and 25th percentiles, respectively. Our data partitioning scheme allocates 80% of the data for training, 10% for validation, and 10% for testing for each link. Additionally, we randomly selected 1000 measurements from a single source-destination pair and manually labeled them with many false negatives to create a dataset for evaluating the decision tree generated by ARISE.

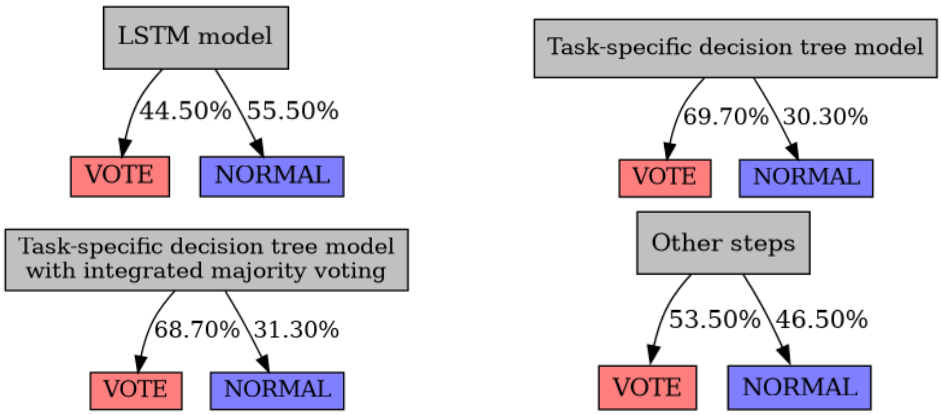

Figure 3. Initial LSTM model for congestion detection on a link (top left), decision-tree-based model explainable model derived using ARISE [12] (top right), the explainable model corrected using majority voting mechanism (bottom left), and lastly the explainable model with step 2-3 applied (bottom right).

Figure 3 shows four key outcomes, along with the percentage of times the data points were labeled as congested ("VOTE") or not ("NORMAL"). On the top left, we show the LSTM model created by ARISE. At the top right, we show the decision tree generated through ARISE's task-specific explainability capability. On the bottom left, we show the explainable model with our integrated majority voting mechanism. At the bottom right we show the application of Steps 2-3 after applying majority voting. Table 5 complements Figure 3 and lists the model evaluation metrics. In particular, the LSTM model achieves a good balance between precision (0.816) and recall (0.964), with an F1 score of 0.884

and an accuracy of 0.881. This indicates that the LSTM model performs well in labeling data points as congested or not can while minimizing false positives. The task-specific explainable model achieves perfect precision (1.000) but has a lower recall (0.645), resulting in an F1 score of 0.784 and an accuracy of 0.833. However, combining this explainable model with our majority voting mechanisms yields predictions with improved recall (0.998) and adequate precision (0.826), resulting in an impressive F1 score of 0.904 and a high accuracy of 0.900. These preliminary findings indicate that including a simple majority voting mechanism can produce a more balanced and accurate classification, addressing the accuracy issues of post-hoc global explainability techniques. Furthermore, given that the task-specific decision tree model's accuracy of 83% and our sample size of 1000 for model evaluation, the voting scheme effectively diminishes the number of "corner cases" from 17% (i.e., 170 corner cases) to 10% (i.e., 100 corner cases). Applying Steps 2-3 results in a perfect precision (1.000) and a decent recall (0.989), with in an improved F1 score of 0.994 and a high accuracy of 0.995. Considering the 10% corner cases after the application of majority voting, applying the rest of the steps 2-3 further diminishes the number of corner cases from 100 to 4.

4.2.4 Results for use case 2. Furthermore, we use the CIC-IDS-2017 [19] dataset, which includes 13 different attacks, including DDoS, Heartbleed, SQL injection, and it also includes benign traffic. The dataset also contains 78 network traffic features such as flow duration, total forward packets. Using this dataset, we train a multi-class Random Forest Classifier to label the data points. The data is partitioned in such a way where 75% of the data is allocated for training and 25% is for testing, where the training data is balanced using Random Over Sampler [14]. A decision tree explanation is extracted using Trustee with the Top-k Pruning

method with $k = 10$. Additionally, we select 1000 random data points from this dataset to evaluate the decision tree generated by Trustee.

Figures 4-6 shows the decision trees generated along with how they are modified using Steps 1-3. Figure 4 shows the decision tree explanation of a Random Forest classifier, extracted using Trustee with top-k pruning method of $k = 10$. In Figure 5, we see the rearranged tree where the features are present in the nodes, and the edges coming out of a node are the rules. In Figure 6, we show the rearranged tree with our integrated rules which are highlighted. Table 6 provides the model evaluation metrics that reflect Figures 4, 5, and 6 for first, second, and third row respectively. The Trustee model achieves a precision of 0.850, a recall of 0.833, an F1 score of 0.821, and an accuracy of 0.829. This suggests that the Trustee model performs well in labeling data points. The rearranged tree model achieves a precision of 0.508 a recall of 0.552, an F1 score of 0.546, and an accuracy of 0.519990. Similarly, the rearranged tree with rules integrated achieves a precision of 0.508 a recall of 0.552, an F1 score of 0.546, and an accuracy of 0.519.

Table 1. Case 1: The LF and classifier F1 scores of the labeling functions that are based on one feature from the CAIDA Ark dataset

LF 1	LF 2	LF 3	LF 4	LF F1	Classifier F1
LF_Mean	-	-	-	0.646117	0.478532
LF_Elliptic	-	-	-	0.727858	0.637661
LF_LOF	-	-	-	0.904137	0.609691
LF_Mean_2SD	-	-	-	0.790268	0.638784
LF_Mean	LF_Elliptic	-	-	0.727858	0.696371
LF_Mean	LF_LOF	-	-	0.904137	0.703294
LF_Elliptic	LF_LOF	-	-	0.904137	0.803738
LF_Mean_2SD	LF_Elliptic	-	-	0.790434	0.691973
LF_Mean_2SD	LF_LOF	-	-	0.904137	0.709458
LF_Mean	LF_Elliptic	LF_LOF	-	0.727858	0.776400
LF_Mean_2SD	LF_Elliptic	LF_LOF	-	0.790434	0.789703
LF_Mean	LF_Mean_2SD	LF_Elliptic	LF_LOF	0.790434	0.702802
LF_Mean_SD	-	-	-	0.766566	0.585658
LF_ORCE	-	-	-	0.701705	0.609254
LF_LOF	LF_Mean_SD	-	-	0.904137	0.805841
LF_LOF	LF_ORCE	-	-	0.904137	0.775798
LF_Mean_SD	LF_ORCE	-	-	0.767457	0.623670
LF_LOF	LF_Mean_SD	LF_ORCE	-	0.767457	0.763401

Table 2. Case 2: The LF and classifier F1 scores of the labeling functions that are based on two features from the CAIDA Ark dataset

LF 1	LF 2	LF 3	LF 4	LF F1	Classifier F1
LF_Mean_SD	LF_Jitter_Mean_SD	-	-	0.020693	0.391774
LF_Mean_SD	LF_Jitter_ORCE	-	-	0.021053	0.464457
LF_Mean_SD	LF_Jitter_LOF	-	-	0.028086	0.369720
LF_ORCE	LF_Jitter_ORCE	-	-	0.017447	0.402051
LF_ORCE	LF_Jitter_Mean_SD	-	-	0.017084	0.300487
LF_ORCE	LF_Jitter_LOF	-	-	0.025453	0.325536
LF_LOF	LF_Jitter_LOF	-	-	0.028062	0.553264
LF_LOF	LF_Jitter_Mean_SD	-	-	0.023611	0.451379
LF_LOF	LF_Jitter_ORCE	-	-	0.023972	0.503357
LF_Mean	LF_Jitter_Mean	-	-	0.015031	0.471121
LF_Mean	LF_Jitter_Elliptic	-	-	0.016058	0.379940
LF_Mean	LF_Jitter_LOF	-	-	0.024451	0.316791
LF_Elliptic	LF_Jitter_Elliptic	-	-	0.018107	0.418650
LF_Elliptic	LF_Jitter_Mean	-	-	0.017258	0.407629
LF_Elliptic	LF_Jitter_LOF	-	-	0.025354	0.382712
LF_LOF	LF_Jitter_Mean	-	-	0.023438	0.472840
LF_LOF	LF_Jitter_Elliptic	-	-	0.024288	0.467367
LF_Jitter_Mean	-	-	-	0.024390	0.110858
LF_Jitter_Elliptic	-	-	-	0.026197	0.212314
LF_Jitter_LOF	-	-	-	0.163881	0.208578
LF_Jitter_Mean_SD	-	-	-	0.021542	0.179522
LF_Jitter_ORCE	-	-	-	0.025658	0.151104
LF_Jitter_Mean	LF_Jitter_Elliptic	-	-	0.026197	0.142008
LF_Jitter_Mean	LF_Jitter_LOF	-	-	0.163881	0.174444
LF_Jitter_Elliptic	LF_Jitter_LOF	-	-	0.163881	0.192129
LF_Jitter_Mean_2SD	LF_Jitter_Elliptic	-	-	0.022796	0.182702
LF_Jitter_Mean_2SD	LF_Jitter_LOF	-	-	0.163881	0.204539
LF_Jitter_Mean	LF_Jitter_Elliptic	LF_Jitter_LOF	-	0.026197	0.203947
LF_Jitter_Mean_2SD	LF_Jitter_Elliptic	LF_Jitter_LOF	-	0.022796	0.195509
LF_Jitter_Mean	LF_Jitter_Mean_2SD	LF_Jitter_Elliptic	LF_Jitter_LOF	0.022796	0.282779

Table 3. Case 1: The LF and classifier F1 scores of the labeling functions that are based on one feature from the RIPE Atlas project dataset

LF 1	LF 2	LF 3	LF 4	LF F1	Classifier F1
LF_Elliptic	LF_LOF	-	-	0.750853	0.623192
LF_Elliptic	-	-	-	0.745049	0.527499
LF_LOF	-	-	-	0.750853	0.494363
LF_Mean_2SD	LF_Elliptic	LF_LOF	-	0.745106	0.598503
LF_Mean_2SD	LF_Elliptic	-	-	0.745112	0.658453
LF_Mean_2SD	LF_LOF	-	-	0.750859	0.637811
LF_Mean_2SD	-	-	-	0.729346	0.580166
LF_Mean	LF_Elliptic	LF_LOF	-	0.745049	0.677733
LF_Mean	LF_Elliptic	-	-	0.745049	0.530618
LF_Mean	LF_LOF	-	-	0.750853	0.528895
LF_Mean	LF_Mean_2SD	LF_Elliptic	LF_LOF	0.745106	0.654784
LF_Mean	-	-	-	0.522303	0.335037
LF_Mean_SD	-	-	-	0.687104	0.533501
LF_ORCE	-	-	-	0.709759	0.642847
LF_LOF	LF_Mean_SD	-	-	0.750853	0.607644
LF_LOF	LF_ORCE	-	-	0.750853	0.621174
LF_Mean_SD	LF_ORCE	-	-	0.710082	0.704314
LF_LOF	LF_Mean_SD	LF_ORCE	-	0.710082	0.704617

Table 4. Case 2: The LF and classifier F1 scores of the labeling functions that are based on two features from the RIPE Atlas project dataset

LF 1	LF 2	LF 3	LF 4	LF F1	Classifier F1
LF_Jitter_Mean	-	-	-	0.005245	0.118757
LF_Jitter_Elliptic	-	-	-	0.050694	0.138830
LF_Jitter_LOF	-	-	-	0.055245	0.101906
LF_Jitter_Mean_SD	-	-	-	0.024518	0.167459
LF_Jitter_ORCE	-	-	-	0.032772	0.135628
LF_Jitter_Mean	LF_Jitter_Elliptic	-	-	0.005245	0.119109
LF_Jitter_Mean	LF_Jitter_LOF	-	-	0.005245	0.068415
LF_Jitter_Elliptic	LF_Jitter_LOF	-	-	0.050591	0.083415
LF_Mean	LF_Jitter_Mean	-	-	0.000000	0.138355
LF_Elliptic	LF_Jitter_LOF	-	-	0.000000	0.235859
LF_LOF	LF_Jitter_LOF	-	-	0.000000	0.216548
LF_LOF	LF_Jitter_Mean	-	-	0.000000	0.236249
LF_ORCE	LF_Jitter_Mean_SD	-	-	0.000000	0.241576
LF_Mean	LF_Jitter_Elliptic	-	-	0.000000	0.203514
LF_Mean	LF_Jitter_LOF	-	-	0.000000	0.133225
LF_Elliptic	LF_Jitter_Elliptic	-	-	0.000000	0.250673
LF_Elliptic	LF_Jitter_Mean	-	-	0.000000	0.256557
LF_LOF	LF_Jitter_Elliptic	-	-	0.000000	0.228274
LF_Mean_SD	LF_Jitter_Mean_SD	-	-	0.000000	0.190205
LF_Mean_SD	LF_Jitter_ORCE	-	-	0.000000	0.211360
LF_Mean_SD	LF_Jitter_LOF	-	-	0.000000	0.198985
LF_ORCE	LF_Jitter_ORCE	-	-	0.000000	0.192774
LF_ORCE	LF_Jitter_LOF	-	-	0.000000	0.170300
LF_LOF	LF_Jitter_Mean_SD	-	-	0.000000	0.218073
LF_LOF	LF_Jitter_ORCE	-	-	0.000000	0.241187
LF_Jitter_Mean_2SD	LF_Jitter_Elliptic	-	-	0.044124	0.116875
LF_Jitter_Mean_2SD	LF_Jitter_LOF	-	-	0.044099	0.113031
LF_Jitter_Mean	LF_Jitter_Elliptic	LF_Jitter_LOF	-	0.005245	0.122860
LF_Jitter_Mean_2SD	LF_Jitter_Elliptic	LF_Jitter_LOF	-	0.044099	0.096842
LF_Jitter_Mean	LF_Jitter_Mean_2SD	LF_Jitter_Elliptic	LF_Jitter_LOF	0.005245	0.117525

	Precision	Recall	Accuracy	F1 score
LSTM model	0.816216	0.963830	0.881000	0.883902
Task-specific explainable model	1.000000	0.644681	0.833000	0.783959
After majority voting	0.825704	0.997872	0.900000	0.903661
Other steps	1.000000	0.989362	0.995000	0.994652

Table 5. Evaluation metrics for the four models depicted in Figure 3.

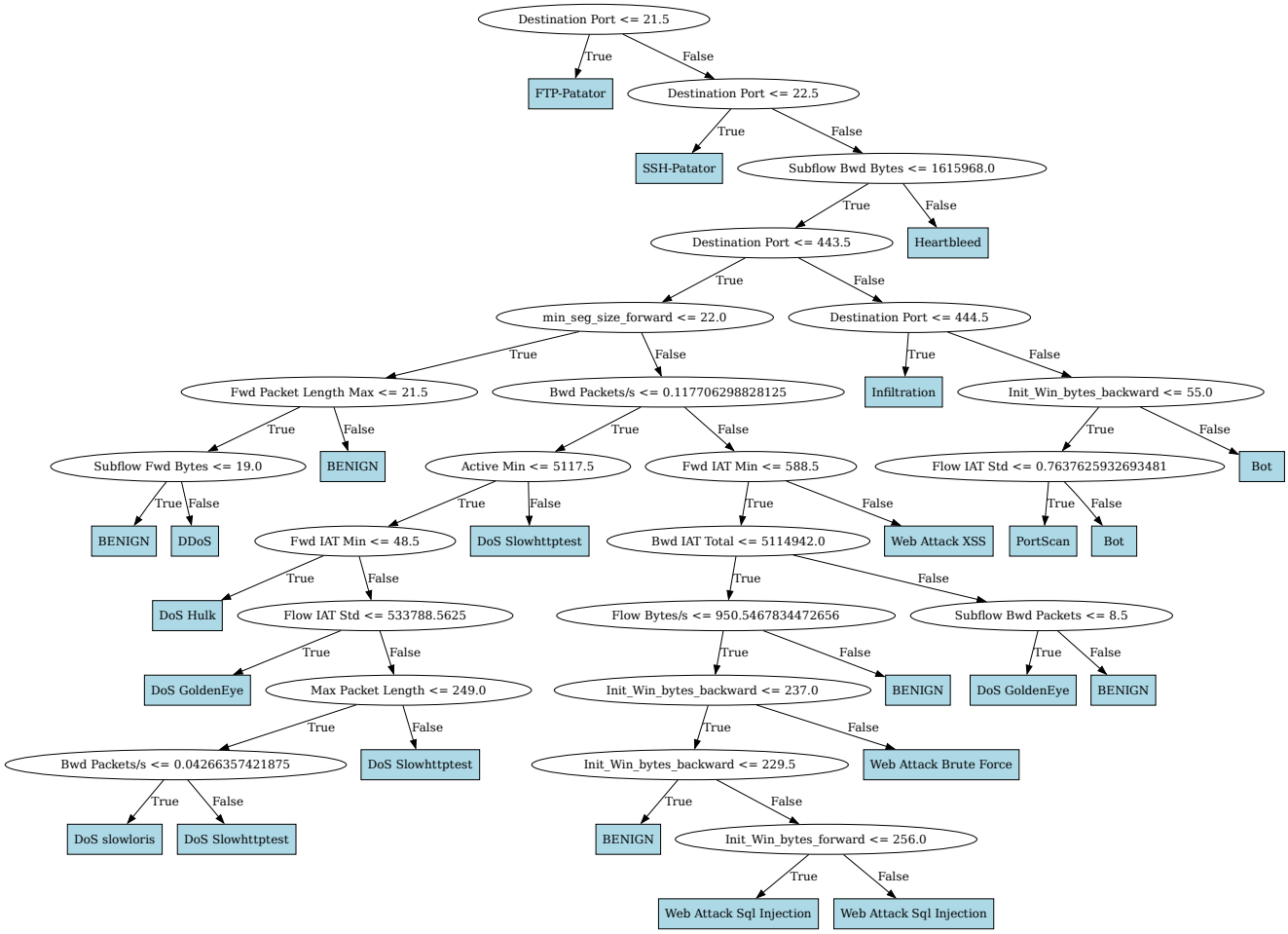

Figure 4. Decision tree generated from Trustee.

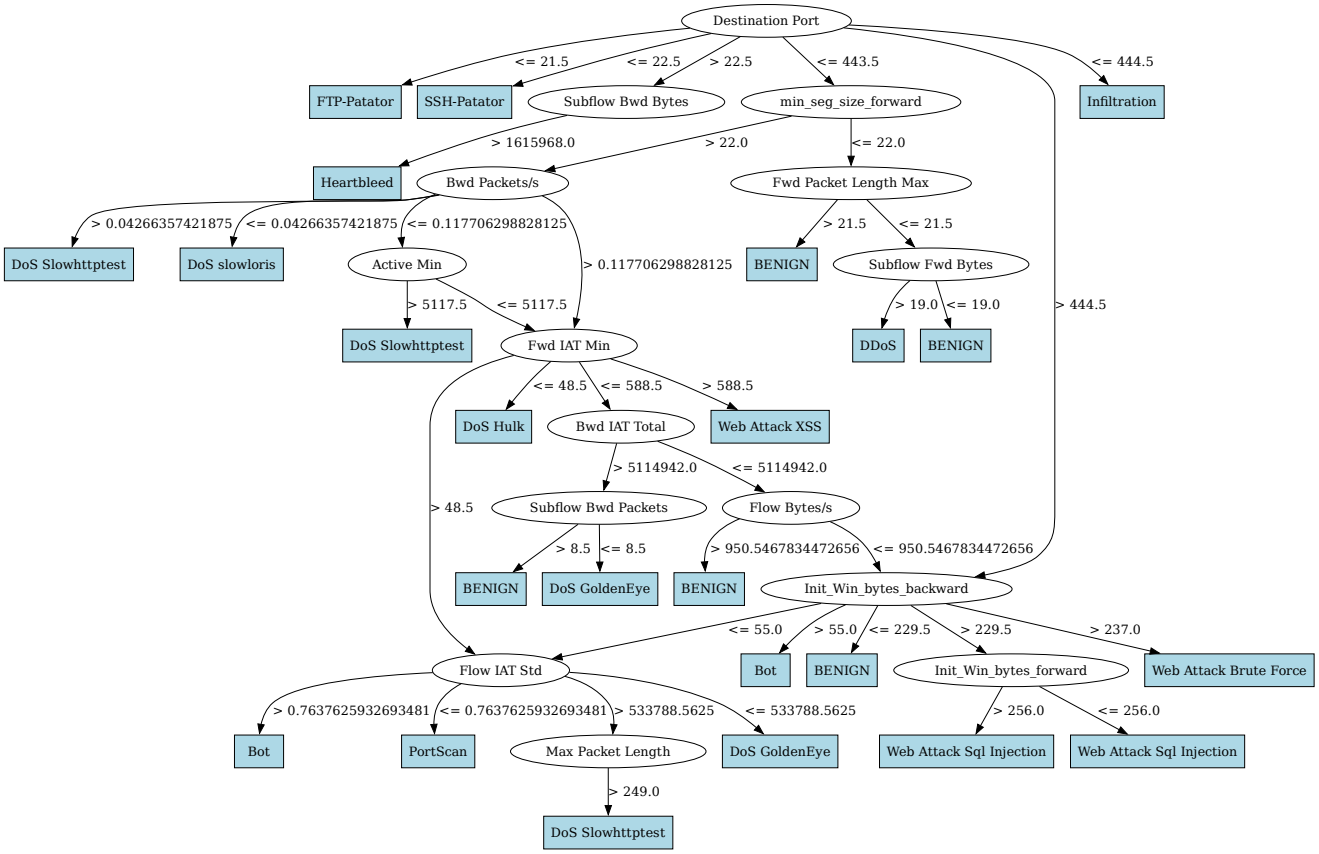

Figure 5. Rearranged tree where the nodes are the features and the edges are the rules.

	Precision	**Recall**	**Accuracy**	**F1 score**
Trustee Model	0.850444	0.833162	0.829000	0.821656
Rearranged Tree	0.508045	0.552451	0.546000	0.519990
After integrating rules	0.508045	0.552451	0.546000	0.519990

Table 6. Evaluation metrics for the three models depicted in Figure 4, Figure 5, and Figure 6.

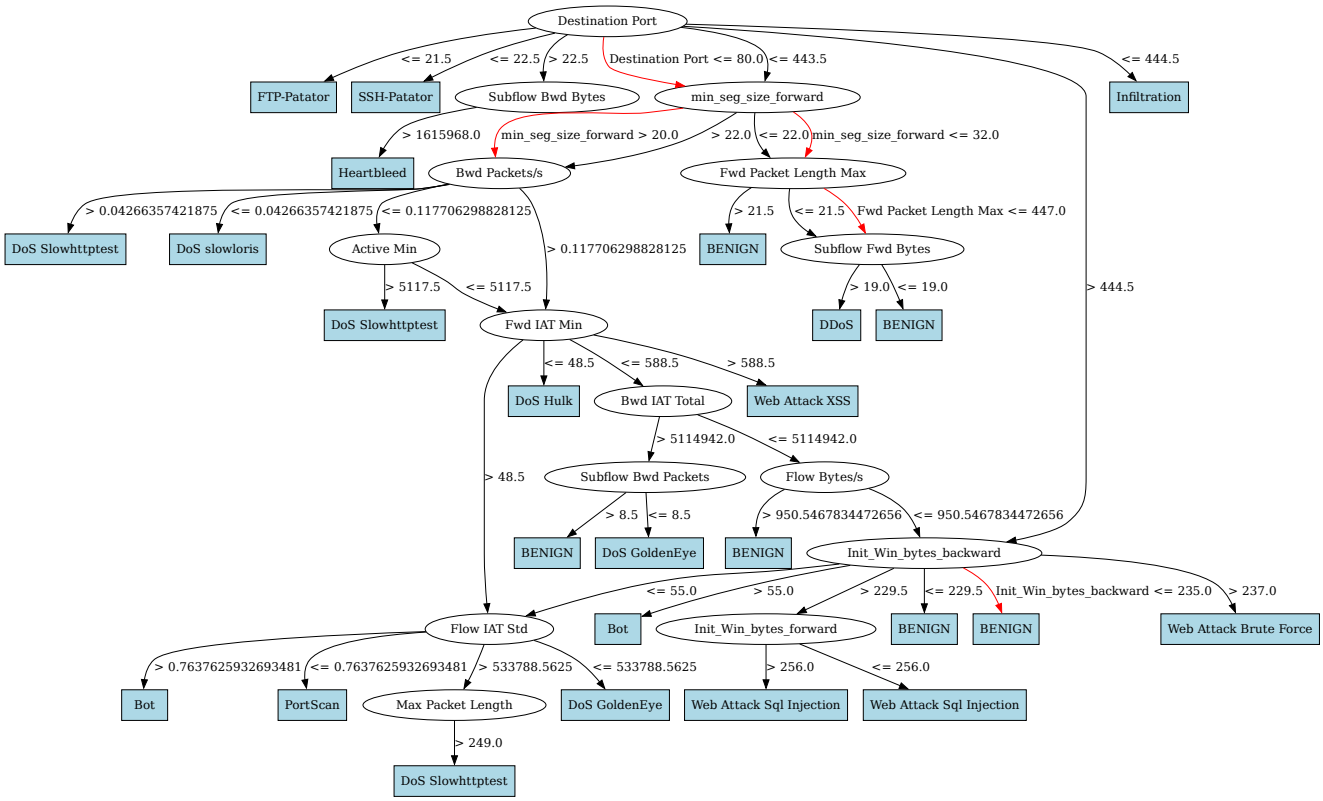

Figure 6. Rearranged tree after integrating nodes.

CHAPTER V

DISCUSSION

The decrease in accuracy in the rearranged tree models in comparison to the original decision tree model by Trustee suggests that there are limitations due to the tree rearrangement. The decrease in accuracy is due to factors such as (1) data property consistency and (2) the impact of the tree rearrangement. A potential reason for the decrease in accuracy could be because of the inconsistency of data properties across trees. The Trustee generated decision tree is able to catch the properties of the data and lead to variations in decision boundaries. On the other hand, the challenge of rearranging a tree is that it may not be able to accurately capture the patterns that were present in the data. The impact of rearranging the decision tree by reordering features and integrating rules can change the decision boundaries and decision-making process that was present in the original tree.

CHAPTER VI

CONCLUSION

In this thesis, we address the issues of achieving trust within and across enclaves by building **SWAN** a framework that provides hybrid explainability of the decisions made by the black-box machine learning model, and the ability to collaborate between different groups by using a pipeline that allows for network operators and researchers to share metadata. We evaluate our framework across different datasets with the combination of different labeling functions which are based on one and or two features. We then evaluate our hybrid explainability technique by considering two use cases, where we can see the effectiveness and limitations of the hybrid explainability technique.

6.1 Future Work

In future work, we plan to deploy this framework in an operational setting. Additionally, we plan to explore methods that ensure consistency in data properties across various decision trees. Furthermore, another approach that may be taken is to explore methods for modifying the data in regard to the tree rearrangements made.

REFERENCES CITED

[1] ARGOVERSE. `https://www.argoverse.org/`.

[2] CAIDA Ark Datasets. `www.caida.org/projects/ark/topo_datasets.xml`.

[3] nuScenes. `https://www.nuscenes.org/`.

[4] RIPE Atlas. `https://atlas.ripe.net`, 2018.

[5] Amina Adadi and Mohammed Berrada. Peeking Inside the Black-box: A Survey on eXplainable Artificial Intelligence (XAI). *IEEE access*, 6:52138–52160, 2018.

[6] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete Problems in AI Safety. *arXiv preprint arXiv:1606.06565*, 2016.

[7] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI. *Information fusion*, 58:82–115, 2020.

[8] Arun Das and Paul Rad. Opportunities and Challenges in eXplainable Artificial Intelligence (XAI): A Survey. *arXiv preprint arXiv:2006.11371*, 2020.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A Large-scale Hierarchical Image Database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[10] Arpit Gupta, Chris Mac-Stoker, and Walter Willinger. An Effort to Democratize Networking Research in the Era of AI/ML. In *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, pages 93–100, 2019.

[11] Arthur S Jacobs, Roman Beltiukov, Walter Willinger, Ronaldo A Ferreira, Arpit Gupta, and Lisandro Z Granville. AI/ML for Network Security: The Emperor has No Clothes. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 1537–1551, 2022.

[12] Jared Knofczynski, Ramakrishnan Durairajan, and Walter Willinger. ARISE: A Multitask Weak Supervision Framework for Network Measurements. *IEEE Journal on Selected Areas in Communications*, 40(8):2456–2473, 2022.

[13] Yukhe Lavinia, Ramakrishnan Durairajan, Reza Rejaie, and Walter Willinger. Challenges in Using ML for Networking Research: How to Label If You Must. In *Proceedings of ACM SIGCOMM Workshop on Network Meets AI ML*, 2020.

[14] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017.

[15] Zachary C Lipton. The Mythos of Model Interpretability: In Machine Learning, the Concept of Interpretability is both Important and Slippery. *Queue*, 16(3):31–57, 2018.

[16] Álvaro López García, Jesús Marco De Lucas, Marica Antonacci, Wolfgang Zu Castell, Mario David, Marcus Hardt, Lara Lloret Iglesias, Germán Moltó, Marcin Plociennik, Viet Tran, Andy S. Alic, Miguel Caballer, Isabel Campos Plasencia, Alessandro Costantini, Stefan Dlugolinsky, Doina Cristina Duma, Giacinto Donvito, Jorge Gomes, Ignacio Heredia Cacha, Keiichi Ito, Valentin Y. Kozlov, Giang Nguyen, Pablo Orviz Fernández, Zděnek Šustr, and Pawel Wolniewicz. A cloud-based framework for machine learning workloads and applications. *IEEE Access*, 8:18681–18692, 2020.

[17] Anirudh Muthukumar and Ramakrishnan Durairajan. Denoising internet delay measurements using weak supervision. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 479–484, 2019.

[18] Alexander Ratner, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré . Snorkel. *Proceedings of the VLDB Endowment*, 11(3):269–282, nov 2017.

[19] Iman Sharafaldin., Arash Habibi Lashkari., and Ali A. Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy - ICISSP*, pages 108–116. INSTICC, SciTePress, 2018.

[20] Yucheng Yin, Zinan Lin, Minhao Jin, Giulia Fanti, and Vyas Sekar. Practical gan-based synthetic ip header trace generation using netshare. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 458–472, 2022.