July 2023

Systems and methods to differentiate access ports and uplinks in telecommunication networks using machine learning

Anonymous

Systems and methods to differentiate access ports and uplinks in telecommunication networks using machine learning

ABSTRACT

This invention is adapted to identify uplinks vs access ports automatically using supervised Machine learning (ML) techniques. Once uplinks vs access ports are identified, the present methods automatically suppress/reduce the noise/non-actionable/non-important alarms from the access ports. Network Operations Centers (NOCs) will be able to change the uplinks vs access ports predicated by ML, and based on feedback from NOCs, required action will be taken against the access port alarms. This makes noise reduction and network operations faster and easier for the operators of enterprise networks.

DETAILED DESCRIPTION

Problem statement:

There are tens of thousands of interfaces in a typically enterprise network. Each interface produces a variety of alarms at high rates, including link down, loss of communication, power module failure, etc. To determine whether an alarm is important or not, network engineers must take actions like clearing or acknowledging it, raising a ticket, etc. This takes time and requires a lot of knowledge from many people. Out of all interfaces (Access/Downlink + Uplink) alarms, it has been determined that 80% of them originate from access interfaces and have a severity of "Critical". Access alarms with this severity are essentially false alarms and system noise. Determining the access interfaces and suppressing the associated alarms with critical severity are the objectives of this solution. To aid in this, rule-based systems can be created, but they are difficult to maintain and incapable of handling complex situations. Hence, multi-vendor network products make every effort to automate the management of alarms. Given the very large number of interfaces, requiring network operators to manually maintain the list of access ports and uplinks is not practical.

Solution:

As the objective is to separate the network's interfaces/port types into access and uplink, followed by the suppression of noisy critical alarms generated by access interfaces, it is suggested using ML algorithms to do so. As a result, this problem is now a "Classification" problem, where the goal of machine learning is to categorise ports and interfaces into access and uplink ports. Once an interface is identified as an access port, an end-to-end system or pipeline will suppress the critical network alarms against it. Both, Supervised and Unsupervised ML approaches used to try to solve this problem statement and produce a reasonable result. Thus, by putting this project into action, system noise is reduced before NOC engineers are overloaded. Details of some encouraging PoC experiments can be found in the sections below.

The End-to-end MLOps System:

Several novel moving components of the end-to-end system are shown in the flow diagram (figure 1) and are also listed below.

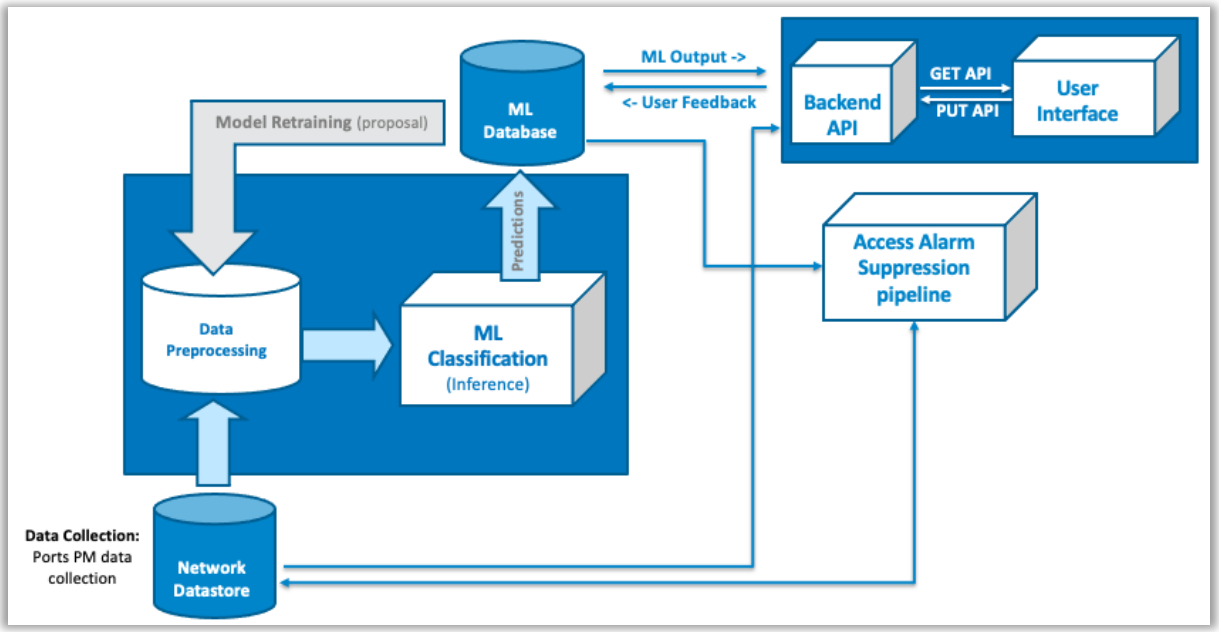

Figure 1

• Network Datastore:
Data is the foundation of data science and is present everywhere. Data is information that can be used as fuel to solve business problems requiring data science. In a manner similar to that, the present systems collect the data from the network datastore; in this case, the datastore is VSure, but it could be any data source.

• Data Pre-processing:
Before using data, pre-processing is necessary. The idea of transforming unclean data into clean data is known as data pre-processing. Before running the algorithm, the dataset is pre-processed to look for missing values, noisy data, and other irregularities. A few features are also engineered to improve model accuracy while also speeding up data transformation; engineered features include computing the ratio of Rx vs Tx BW utilisation, Mean of metrics such as Rx BW, Tx BW, over the period of time, Extraction of key characteristics from textual fields such VLANs existence and Operation State of the interface.

• ML Classification (Inference):
This section is the project's core. It is principally in charge of categorizing interfaces into Access and Uplink types. To carry out the classification, present systems employ the XGBoost ML algorithm. This section might perform the regular task of classifying as online

or a batch process. As soon as the interface has been recognized, the results are stored in the ML database.

• ML Database:
ML, User interface (UI), and alarm suppression pipeline frequently use ML database. It holds the user's ML prediction feedback from the user interface in addition to the output from the ML classification task.

• Role of User interface (UI):
The user interface is the main area where users can view machine learning predictions and offer feedback on them. The same user information will be used for model re-training if the user determines that the ML prediction is incorrect (reinforcement learning).

• Alarm Suppression Pipeline:
In order to achieve the project's main objective, critical alarms against interfaces that are identified as Access types must be suppressed. Every time a new network alarm arises in the system, it runs in real time, pulling ML-prediction references from the ML Database and suppressing any incoming alarms that are determined to be from Access Interface/port.

EXPERIMENTAL POC AND DATA:

With the help of actual measurements taken from production networks, tests implemented a PoC to verify the methodology. Devices are multi-vendor, and come from different criteria (Speed Bandwidths (BW), Rx/Tx BW utilisation, Operational states, VLAN configurations, etc) to test different scenarios.

Methodology:

- Both the supervised and unsupervised ML approaches used to try to solve this problem statement produce a considerable & reasonable result. However, when compared to both, Supervised ML produces much better results, so for the purposes of this disclosure, it will focus mostly on Supervised ML; yet also included is some information from unsupervised machine learning (ML) to be comprehensive.
- For this PoC, examples have considered various devices. The dataset that has been taken into account consists of device interfaces; at random, 500 Access (downlink) and 500 Uplink interfaces/ports were selected.

Speed/Bandwidth allocation: According to the usual data patterns, it was anticipated and also seen that Uplink interfaces have higher values for Speed/Bandwidth allocation than Access. It makes sense because the Uplink interfaces handle much more traffic than the Access interfaces do, as shown by figure 2, which contains the same data.

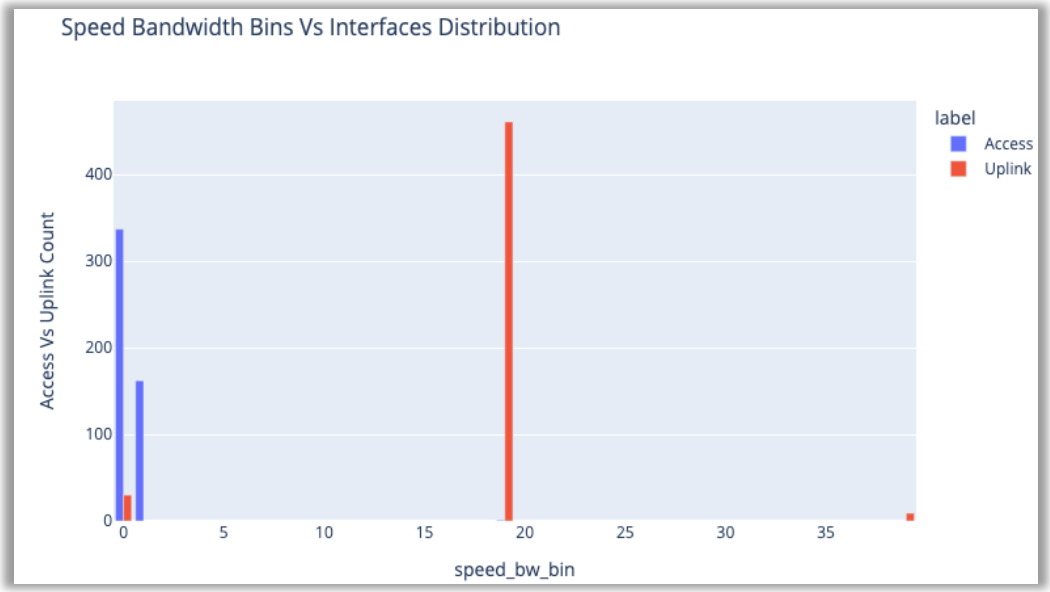

Figure 2

Rx/Tx Utilization:

The typical trend for Rx/Tx bandwidth usage is shown below in figure 3, with access interfaces typically using lower bandwidth values and Uplink interfaces typically using higher bandwidth values. As a proof of concept, tests collected 1 Hour bin PM data against Rx/Tx BW Utilization. As a result, the Rx/Tx BW usage for Access was skewed at lower values whereas the values for uplink ones have wider spread.

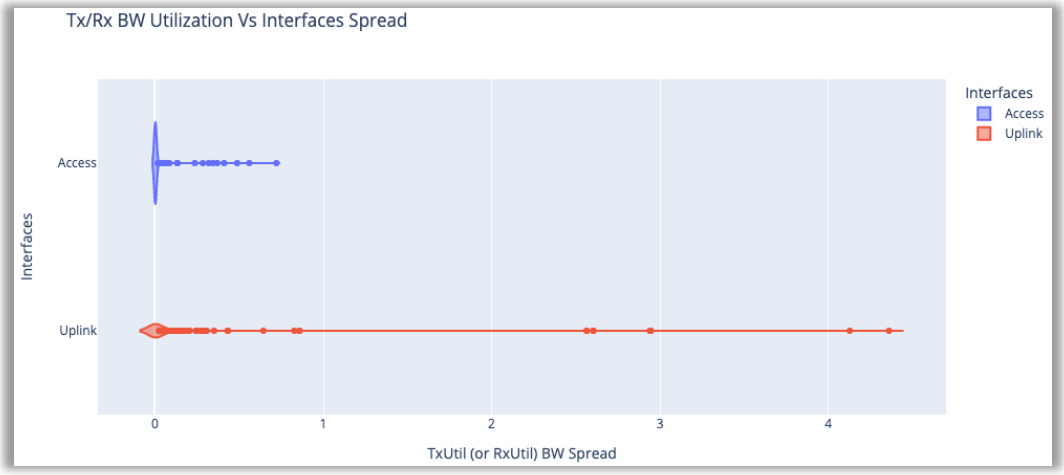

Figure 3

VLAN Configuration count:

According to the usual data trends, it was anticipated and also seen that VLAN configurations in the network are primarily against Uplink interfaces as opposed to Access interfaces. The same information is shown in figure 4 below, the count is high for Uplinks.

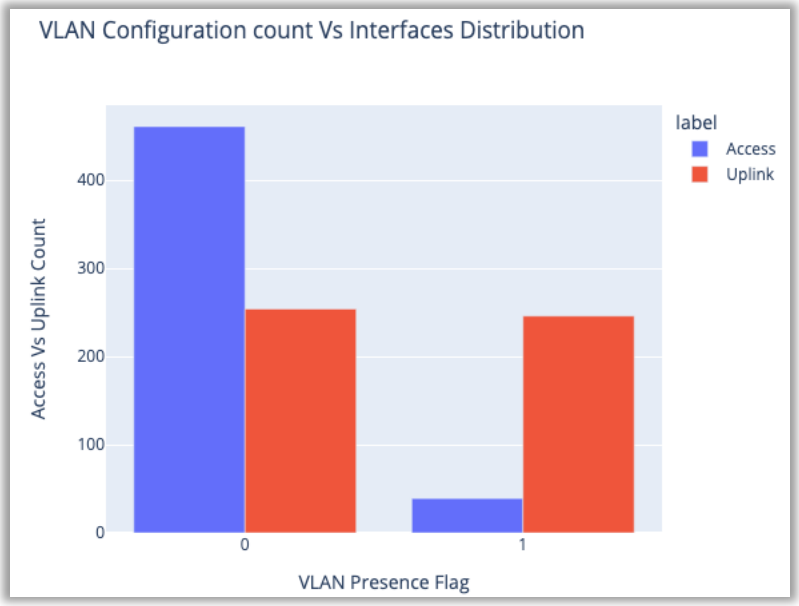

Figure 4

Operation State:

Because the Access interfaces aren't used as frequently as Uplink ones, it was expected and also observed that their operation state is off for the majority of the time. The same information is shown in figure 5 below.

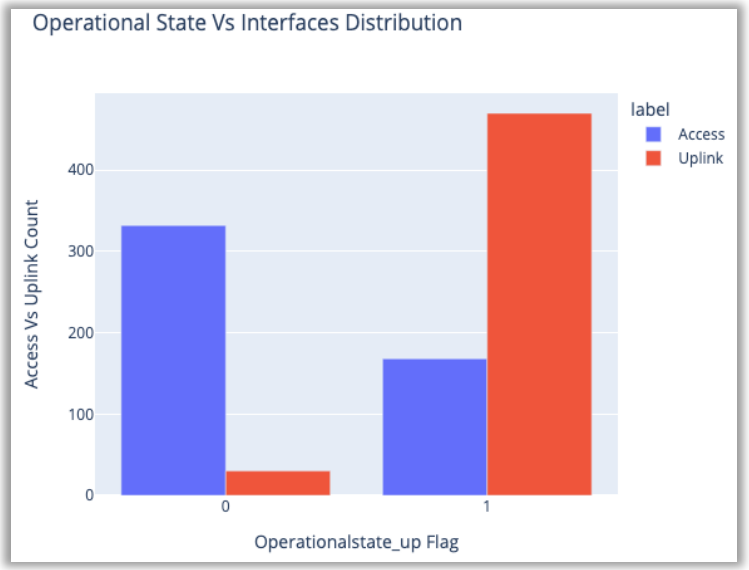

Figure 5

Supervised ML details:
With the aid of supervised learning, the ML can predict the outcome based on prior experiences. In supervised learning, systems can be sure of the object classes.

Datasets Consumed:
For this PoC, tests have considered various devices. The dataset that has been taken into account consists of device interfaces; at random, selected were 500 Access (downlink) and 500 Uplink interfaces/ports.

Parameter	Value	Remarks
Devices used	XX	Production Data
No. of Interfaces Considered	1000 interfaces	500 -> For **Uplink** 500 -> For **Access**
Data Period Considered	Last 10 days (1-Sept to 10-Sept, 2022)	Here, tests could also use data from a single day, but the more data, the more trends that can be captured.

ML Modeling:

- **ML Algorithm:** A Supervised ML classification technique is used for the ML modelling. The XGBoost Classifier was used to implement a prototype.
- **Input/Features:** A list of features including 'vlans_flg', 'speed_bw_bin', 'lastknownuptime', 'service_id_flg', 'operationalstate_up_flg', 'rxutil_mean', 'txutil_mean', 'rxutilmax_mean', 'rxutil_mean_txutil_mean_ratio', 'administrativestate_up_flg'.
- **Labels:** Access vs. Uplink Port gathered from IT.
- **Output:** Classifier model that determines whether the provided interfaces are access or uplink; if access interface is found, the live alarms with critical severity are suppressed via end-2-end pipeline.
- **Important evaluation metrics**
  - High Recall on Uplink interfaces and the suppression/identification of Access Port/Alarms are the main metrics for this model evaluation.
  - Recall >= 99% on Uplinks is required, resulting in fewer false negatives (maximum 1%). This is due to the fact that classification shouldn't disregard the significance of Uplink interfaces/alarm.
  - 10X cross validation: In order to accommodate all of the test scenarios in the dataset, tests also performed 10X cross validation, and achieved a great accuracy score of 99.3%.
  - The graphs that demonstrate ML performance metrics are shown below:

Precision-Recall Curve:
Recall value is on the x-axis of the PR curve, and precision is on the y-axis. Precision highlights the relevance of the results, which is more crucial when evaluating an ML system. As a result of the nearly perfect score, tests can correctly classify the interfaces.

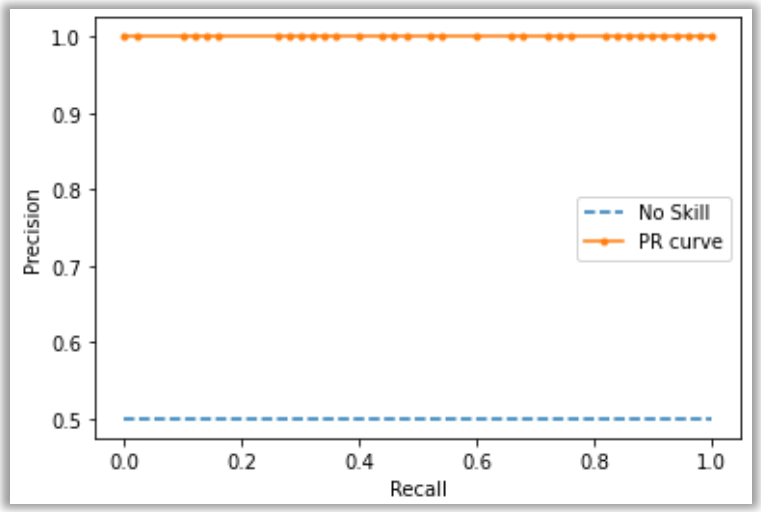

Figure 6

Confusion Matrix:

The performance of the classification models for a specific set of test data is evaluated using a matrix called the confusion matrix. Only when the true values of the test data are known can it be determined. It was clear from the algorithm's diagonal orientation that it had correctly classified the two interface classes.

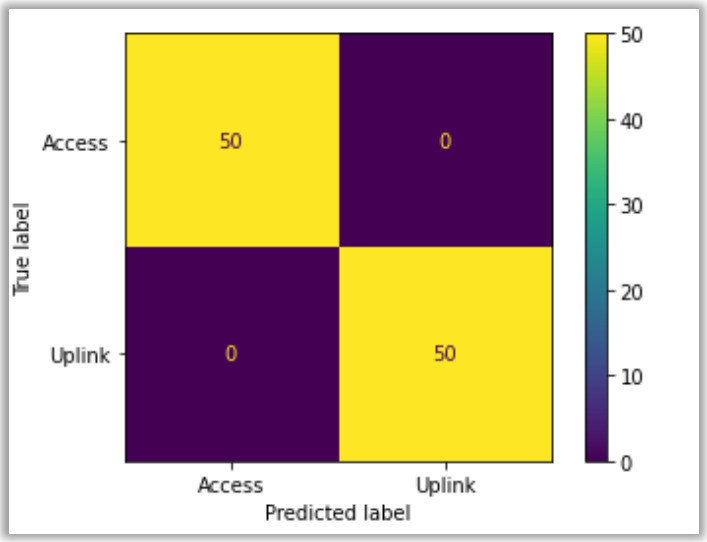

Figure 7

Classification Report:

In machine learning, a classification report is a performance evaluation metric. The precision, recall, F1 Score, and support of the trained classification model are displayed

using this method. Here, tests nearly achieved a score of 100%, indicating that the systems correctly classified the interfaces.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	50
1	1.00	1.00	1.00	50
accuracy			1.00	100
macro avg	1.00	1.00	1.00	100
weighted avg	1.00	1.00	1.00	100

Figure 8

Uplink Recall Vs Access Alarm Suppression rate Trade-off:
Figure 9 demonstrates how effectively the solution can function even when systems assume a recall of 99% on Uplink interfaces, which is a substantial number. This also means that systems will be able to suppress all noisy/unimportant access interface critical alarms, without sacrificing crucial Uplink interface alarms. Additionally, with the above said Uplink recall criteria met, the objective is to have better precision (fewer false positives, greater Access Interface Alarm Suppression).

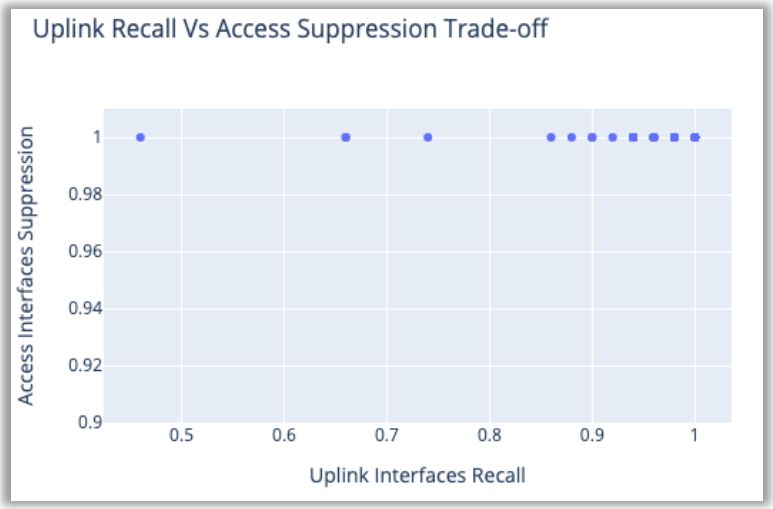

Figure 9

Feature selection:
With the help of only relevant information and the elimination of irrelevant data, feature selection is a technique for lowering the input variable for the model. This involves automatically selecting features for the ML model that are relevant to the problem to solve.

	Feature	Chi-2	RFE	Random Forest	XGB	LightGBM	Total
1	speed_bw_bin	True	True	True	True	True	5
2	txutil_mean	True	True	False	True	True	4
3	rxutilmax_mean	True	True	True	False	True	4
4	rxutil_mean	True	True	True	False	True	4
5	txutilmax_mean	True	True	False	False	True	3
6	rxutil_mean_txutil_mean_ratio	True	True	False	False	True	3
7	lastknownuptime	True	True	True	False	False	3
8	vlans_flg	True	True	False	False	False	2
9	txutil_mean_rxutil_mean_ratio	True	True	False	False	False	2
10	service_id_flg	True	True	False	False	False	2
11	parent_id_flg	True	True	False	False	False	2
12	operationalstate_up_flg	True	True	False	False	False	2
13	administrativestate_up_flg	True	True	False	False	False	2

Figure 10

Unsupervised ML details:

Finding valuable insights from the data is made easier with the aid of unsupervised learning. Unsupervised learning is considerably more like how humans learn to think via their own experiences. Unsupervised learning is more significant because it operates on unlabelled and uncategorized data.

- Unsupervised machine learning was the primary choice as the main strategy for addressing this problem. The typical clusters seen are displayed below as a result of the implementation of a few items in this ML area as well.
- Uplink interfaces PM values over the period in the second plot(cluster-1)(figure 11) have somewhat higher Rx/Tx BW utilization values when compared to the access ports, which are the first plot (cluster-0) (figure 11), where the Rx/Tx BW utilization values over the period are typically close to zero.

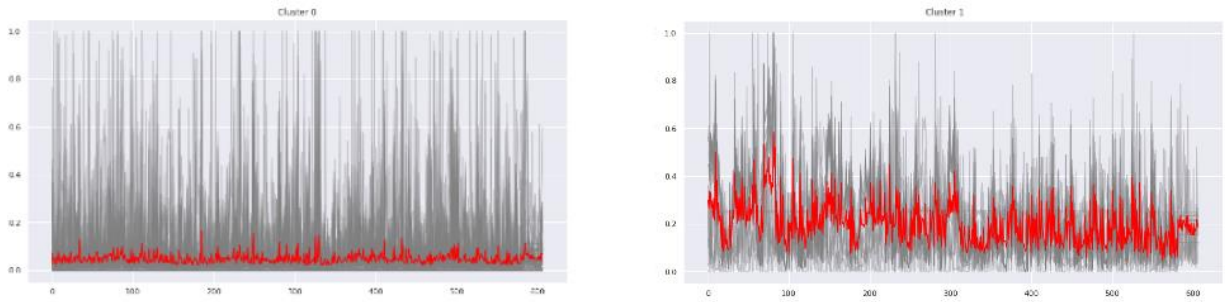

Figure 11

- Additionally, one can observe that the data distribution between the two has some overlap, which implies one could miss-cluster the access interface as an Uplink interface or vice-versa.
- So, since Supervised ML seemed more promising, systems proceed with it. However, in the future, systems can use semi-supervised machine learning (ML),

which combines both the supervise and unsupervised techniques, to further develop the algorithm for additional cases.

Benefits of the present solution include that it is helpful to automatically differentiate between Uplink and Access Ports with great accuracy and precision. It is also helpful to suppress noise/non-actionable/non-important alarms from all type of network infrastructures. ML predication output can be changed by the NOC engineer, and this is considered as feedback from the NOC engineer. Automated learning improvement by ML model based on NOC engineer feedback. This also opens a new avenue in the direction of Reinforcement Learning (RL), a type of machine learning technique that allows an ML agent to learn in an interactive environment through trial and error while taking into account user feedback. Present systems will increase efficiency of NOC engineers to target, focus on only important alarms and will reduce Mean Time To Recovery (MTTR) by prioritising important network impacts. This effectively achieves all Service-Level Agreements (SLAs) and saves penalties for agreed SLAs, and this will be a catalyst as a revenue generation solution. Making decisive actions and offering insights from UI as a reporting tool/dashboard is supported by the present solution. These insights may be useful to shape customer/NOC engineer's decisions for new network site implementations, for designing new network services and any new network devices procurements and inventory managements. Various embodiments help to identify what type of exact specific problems/Root Cause Analysis RCA are more contributing network impacts, and same can help for strategic planning of NOC operations and resources.

The following novel aspects are provided by the present solution.

1) Novel multi-step end-to-end solution (including data collection, ML-based classifier and policy engine) to automatically differentiate between access ports and uplinks, and automatically trigger actions based on interface type.

2) ML model to automatically predict and differentiate Uplinks vs. Access ports. The ML model can be trained using the following features from historical data:
   - Interface level data: Device family, Interfaces identity, transmit and receive metrics utilisation, bandwidth allocation to interfaces, last known uptime, physical and operational status of the interface, availability of the interface
   - Logical interface data: VLAN configuration, access/trunk port configuration on the interface, service configuration on the interfaces
   - Topology: access ports are usually connected to laptop, computers, printers etc. while uplink ports are connected to adjacent switches, routers. Topology information may be obtained automatically using for instance the LLDP protocol or from other sources.
   - Unstructured textual features that are useful for model training can be extracted using natural language processing from interface descriptions or other text fields.
   - Use Supervised ML to implement classifier. ML algorithms, such as Decision Trees, Random Forest, XGBoost, and Deep Learning Frameworks like CNN and LSTM, were tested during the prototype phase.

3) System where the above ML model is integrated with active learning and AutoML capabilities to automatically accommodate changes of topology and traffic patterns
    - Feedback from the NOC team is taken into account as an input feature to the ML model during routine/automated model re-training
    - No traffic on interfaces for more than 30 days, considered as access port and fed it to ML algo to process alarms on port.
    - Auto-ML machinery to retrain ML models automatically when necessary
    - Automatically retrain the model when accuracy degrades and promote model to production
    - A generic ML model may be provided out of the box, and fine-tuned with AutoML to accommodate customer-specific patterns.

4) System where the ML model above is integrated with a policy engine for closed-loop automation
    - Actions may be automatically triggered using the inferences/predictions from of the above ML model
    - Triggered actions may optionally be validated by an external safeguard system (human SME or computer system)
    - Automated actions triggered by the above system include in particular:
        - Rank/Prioritize alarms generated from uplink ports.
        - Change the severity of the alarms from uplinks and access ports. Typically, uplink interface alarms will have a higher severity.
        - Automatically suppress alarms. For instance, an operator may automatically suppress link-down and loss-of-communication alarms for access ports.

The present solution includes the following operating principls:

1. Data Collection:
For the mentioned innovation, the data can be directly collected from customer premise devices or any network platform. The data includes details such as device identity, network topology details, as well as information about transmission + receiver data traffic and its utilisation.

The collected input interface data is implicitly/explicitly and feedback provided by network operators leveraged to label data into access and uplink ports. As the feedback is implicit, the innovation can seamlessly integrate with existing system and reduce the manual intervention required to detect the type of device port and automatically reduce false alarms generated for access ports.

The Feedback/Input includes:
1. Bandwidth allocation information about Interface.
2. Bandwidth utilization by the interface.
3. Physical and operational status of interface.
4. Network topology (good to have)
5. Labels provided by domain/NOC experts.

2. Model Training and Evaluation
Model retraining will be done for the new devices enrolled in existing system to segregate the device ports properly. With Auto retraining approach using MLOps System the solution can automatically retrain itself and choose the best model based on performance.

3. Port Identification on real-time system:
The real-time/batch module in the solution classifies the devices interface ports into access and uplink with corresponding probability. It streams the device data from Network data store and Machine learning model fetch this data, do processing and publishes the prediction on user interface. The computational requirement of solution is minimal which permits the processing of information of several devices in real time to classify them in access and uplink ports.

This Solution will suppress the fault alarms for identified access ports in real time automatically and consider NOCs feedback about ports to retrain itself to improve the performance. The abovementioned end-to-end solution process flow is as shown below in figure 12.

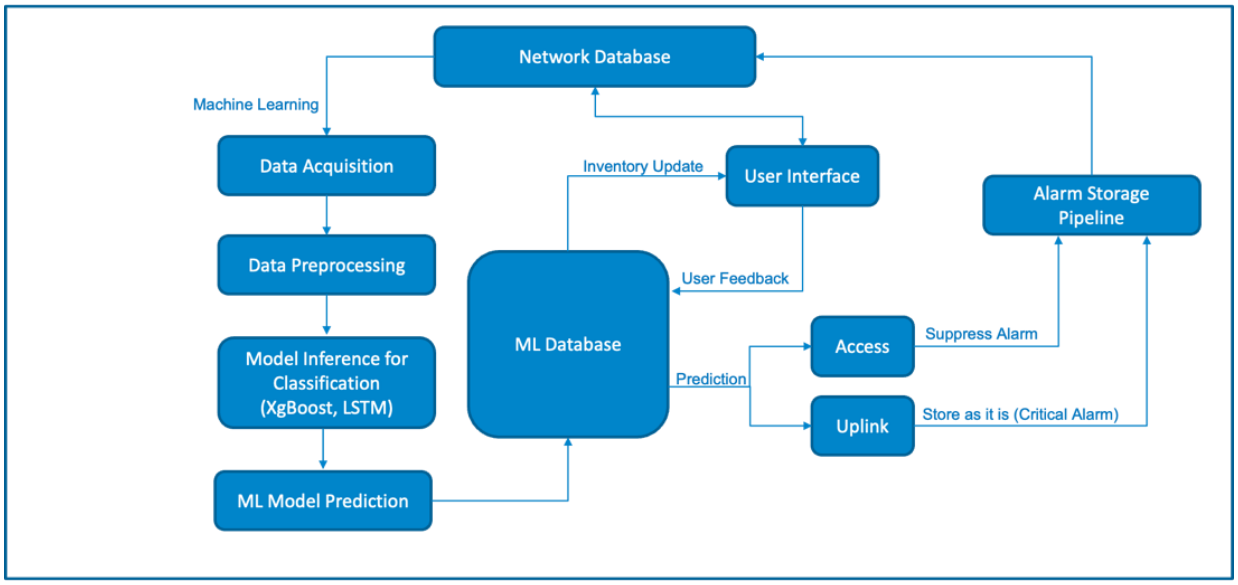

Figure 12

Abbreviations and remarks:
NOC: network operations centre
ML: machine learning
ML Ops: ML Operations
SL: supervised ML
BW: Bandwidth Allocation
Rx/Tx: Receiver/Transmitter End
UI: User Interface
Port:Interface

Access: Downlink

Conclusion:

It will be appreciated that some embodiments described herein may include one or more generic or specialized processors (“one or more processors”) such as microprocessors, digital signal processors, customized processors, and Field-Programmable Gate Arrays (FPGAs) and unique stored program instructions (including both software and firmware) that control the one or more processors to implement, in conjunction with certain non-processor circuits, some, most, or all of the functions of the methods and/or systems described herein. Alternatively, some or all functions may be implemented by a state machine that has no stored program instructions, or in one or more Application-Specific Integrated Circuits (ASICs), in which each function or some combinations of certain of the functions are implemented as custom logic. Of course, a combination of the aforementioned approaches may be used. Moreover, some embodiments may be implemented as a non-transitory computer-readable storage medium having computer-readable code stored thereon for programming a computer, server, appliance, device, etc. each of which may include a processor to perform methods as described and claimed herein. Examples of such computer-readable storage mediums include, but are not limited to, a hard disk, an optical storage device, a magnetic storage device, a ROM (Read Only Memory), a PROM (Programmable Read-Only Memory), an EPROM (Erasable Programmable Read-Only Memory), an EEPROM (Electrically Erasable Programmable Read-Only Memory), Flash memory, and the like. When stored in the non-transitory computer-readable medium, the software can include instructions executable by a processor that, in response to such execution, cause a processor or any other circuitry to perform a set of operations, steps, methods, processes, algorithms, etc.

Although the present disclosure has been illustrated and described herein with reference to preferred embodiments and specific examples thereof, it will be readily apparent to those of ordinary skill in the art that other embodiments and examples may perform similar functions and/or achieve like results. All such equivalent embodiments and examples are within the spirit and scope of the present disclosure.