2023 Asia Communications and Photonics Conference/2023 International Photonics and Optoelectronics Meetings (ACP/POEM) | 979-8-3503-1261-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ACP/POEM59049.2023.10369987

Fault Prediction for Optical Access Network Equipment using Decision Tree Methods

K. Murphy*, A. Lavignotte†, and C. Lepers‡
SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris, 91120 Palaiseau, France
Email: *killian.murphy@telecom-sudparis.eu,

***Abstract*— Predicting optical equipment failures reliably before they happen, offers the promise to significantly improve network Quality of Service (QoS) and to reduce maintenance costs. State-of-the-art prediction methods for Network Fault Prediction are not easily comparable. In this paper, a Decision-Tree based Machine Learning benchmark for optical Network Fault Prediction alarms is presented based on a real-world dataset. The Machine Learning architectures are compared with a fixed lead time between the prediction and the window for which alarms are considered. Precision, recall, F1-score metrics, a cost function representing theoretical monetary gain are used for the comparison. Additionally, a Quality of Service gain metric is proposed and used for the comparison. Perspectives for future research are proposed.**

***Index Terms*—Optical communication networks, network reliability, network fault management, network fault prediction, network failure prediction, network maintenance, photonics, machine learning.**

This work was financed by SPIE ICS and by ANRT CIFRE research grant n°2020/1281. We thank SPIE ICS for sharing their knowledge on network maintenance.

I. Introduction

NETWORKS grow ever more complex, as they are constantly integrating new services, and thereby increasing the cost of network service degradations. A loss of profit of around $40,000 per year per hundred users has been estimated for companies due to network downtime [1], [2]. Due to the critical services running on network infrastructure, there is a growing interest in keeping the network running with minimal interruptions.

In the field of Network Fault Management, Network Fault Prediction (NFP) aims to detect early signs of equipment and network failure, in order to improve maintenance response. However, NFP is not yet widely available commercially. This partly due to a lack of reliable measurements of performance models in nowadays networks.

Benchmarks of ML methods were made for homogeneous cellular base stations data [3], and for heterogeneous network data [4]. To our knowledge, although studies exist for failure detection for optical networks [5], no benchmark has been done failure prediction for optical equipment. In this study, a benchmark of Decision Tree (DT) Machine Learning (ML) approaches for alarm prediction on a real-world data set of optical equipment is provided. DT-based methods were chosen as they allow for performance while both preserving explainability and having good generalization properties [6].

First the data collection process is described. A metric quantifying improvement in Quality of Service is proposed. Then the implementation and results of the different models are provided. Finally, future perspectives are discussed.

II. Data acquisition and pretreatment

A. Data acquisition

The data are acquired from a Network and Systems Integrator (NSI) with more than fifty clients in network maintenance. The data consist of Simple Network Management Protocol (SNMP) monitoring of networks and the associated alarms. Data are concentrated in the Network Operation Center (NOC) with a sampling period of 5 minutes (300 seconds), on selected monitored equipment. The label class is generated at the NOC depending on certain pre-established thresholds. In normal network operation, if these alerts persist for more than 20 minutes, a fault ticket is created, and resources are deployed to resolve the issue.

These data are collected on a heterogeneous set of equipment, over real optical IP networks. We segregate the data by the equipment model and brand, as different data is usually collected for each in the monitoring process, to avoid dealing with empty values. This means not working with sparse data, which would heavily restrict the possible types of architectures. Instead, a new ML model and potential changes in the architecture are needed for each subset of the dataset.

For the subset mentioned in this study, data are collected over three weeks. Around twenty Cisco ASR optical routers are used, representing around 60,000 data points. There are around 800 data points in a state of alarm over this dataset, which is around 1.3% of the data.

B. Characteristics of the data

The data are time series, containing variables of the Management Information Base (MIB) of SNMP. These variables can be either proprietary or not, and are updated regularly by each piece of network equipment in their own MIB. The variables can contain information regarding network connectivity (e.g. ping test: latency and packet loss; transferred data) or the state of the machine itself (e.g. transceiver power bias; temperature sensors). At the level of the network equipment, the sampling period is 1 minute. A mean-aggregated value of these data is then polled every 5 minutes, the sampling period of the monitoring system.

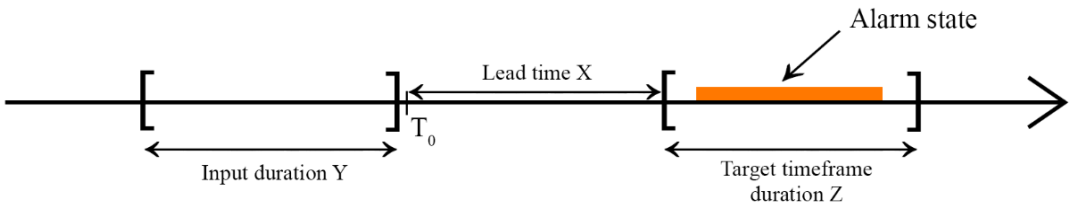

Fig. 1: Classification problem modeling for NFP. Varying X, Y and Z has an effect on performance. Alarm state is shown in orange.

C. Modeling the problem

The problem is modeled as a classification problem in most NFP studies [3], [7]–[9]. In Fig 1, for each point in time, T_0, present in the data, we predict whether the equipment will be in alarm at any point during the time interval [T_0 + X; T_0 + X + Z]. X is the lead time to the prediction. A large lead time is desirable when predicting network faults, as this gives time to intervene and prevent equipment damage. Z is the width or duration of the prediction window. The larger Z is, the higher the probability is that the window contains an alarm. In the example in Fig 1 the label is set to 1 for the prediction at T0. Y is the input duration, this value is used when past information is given to the models for each prediction. Here, we consider Y = 0 as, on their own, DT-based models have no way to use the contextual differences between the different timestamps. It was shown in a previous study that simply providing additional time points was detrimental to the prediction performance.

D. Data pretreatment

In this section, the different transformations that were applied to the data are described.

The columns of the dataset are checked for Not a Number (NaN) values. Fully NaN columns are dropped, and lines with NaN values present are also dropped afterwards. This poses the potential issue of having nonconsecutive data with wide time deltas. The verification was made, and it was determined not to be a problem for the data considered here (the average time between two samples is around 5 minutes). In practice, this eliminates equipment, the information of which is not measured for all the columns from the dataset.

Yeo-Johnson normalization is applied to the data. Normalization helps to deal with outlier values that would otherwise skew the scale of the data. This method was selected due to its ability to handle negative values that are present in the different subsets.

Then, the data are scaled to adjust for the various scales for the different variables, using a min-max scaler.

The two following operations are realized on the data of each equipment independently from others.

A trend feature is created for every column, and the original feature is "de-trended" with the operation $x_{detrended} = x - x_{trend}$. The trend feature is obtained by using a rolling window average with the following parameters: window=12 samples, center=False (can only use past values) and $minperiods = 6samples$ [10].

The pretreated dataset is divided into training and test datasets along a 70%-30% split, respectively. The division is made along the equipment axis.

III. Quality of Service metric

In this section we describe a Quality of Service (QoS) metric proposed to link precision and recall of the ML model to expected QoS improvement by following the predictions of a model.

A. Usual classification metrics

Papers in NFP commonly present their results using the precision and recall metrics. We briefly introduce them here using table I, a more complete description can be found in [10].

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

TABLE I: Classification of prediction results.

Precision is the proportion of cases where the model is right when it predicts a positive sample.

$$Precision = \frac{TP}{TP + FP}$$

where TP=Number of True Positives and FP=Number of False Positives.

Recall is the proportion of positives that were predicted.

$$Recall = \frac{TP}{TP + FN}$$

where FN=Number of False Negatives.

B. Initial setting

QoS impact of a failure is heavily related to the importance of the failing equipment in the network, and to the volume and importance of applications and services unavailable due to the failure. The Network and Systems Integrator (NSI) assign a priority to each piece of equipment, alarm, and failure. In order to quantify QoS lost due to failure in a more accurate manner, the priorities assigned to equipment and failures are

used to weight failures in the following formulae. The vector of adjustment coefficients for each priority C. The Precision and Recall vectors are calculated per priority class. This will allow to adjust the QoS to different client contexts.

$$C = \begin{pmatrix} C_1 \\ C_2 \\ C_3 \end{pmatrix}, C_i \in \mathbb{R}^{+3}, \quad Recall = \begin{pmatrix} R_1 \\ R_2 \\ R_3 \end{pmatrix} \in [0,1]^3$$

$$\text{and } Precision = \begin{pmatrix} P_1 \\ P_2 \\ P_3 \end{pmatrix} \in [0,1]^3$$

C. Metric definition and simplification

The improvement in QoS is calculated on a scale of]$-\infty$;1], with 1 being perfect (meaning no impact on QoS from failures), 0 meaning no improvement and anything negative meaning worse QoS than not predicting anything. Improvement in QoS brought by a prediction model is the weighted average of the improvement for each priority in equation 1.

$$\begin{aligned} QoS_G &= \frac{1}{\Sigma C_i}\Sigma C_i(1 - \frac{QoSwithPred_i}{QoSwithoutPred_i}) \\ &= \frac{1}{\Sigma C_i}\Sigma C_i(1 - \frac{QoS_{Wi}}{QoS_{WOi}}) \end{aligned} \tag{1}$$

QoS_{Wi} is expressed in the following way:

$$QoS_{Wi} = InterventionCost_i + CostofNonPredicted_i \tag{2}$$

The QoS intervention cost represents the time lost in degraded service due to a maintenance that is done as a consequence of predicting a failure. It is defined in the following formula, as the total number of predicted as failures (Predicted Positives PP) samples multiplied by λ. λ is used to adjust the QoS cost of intervening when a failure is predicted, and the impact of the corrective measures on the QoS for the client. This value can be intuited as a loss of time in hours per intervention. This value should be quite low to balance with the cost of an actual failure that would result in a way longer service interruption.

$$InterventionCost_i = \lambda * PP_i \tag{3}$$

The cost of the non predicted failures is defined as the number of failures that we failed to predict, multiplied by the sum of their average duration $E[T_F]$ (T_F for time duration of failure) in hours and μ. μ is a parameter to account for the relative importance of the existence of a failure compared to its duration. A failure and the lack of service it involves, along with the communication made by the IT service will have an impact on the perceived QoS by the client, which is Quality of Experience (QoE), even if the duration of the failure is short. It can be set to zero to account for pure QoS.

$$CostofNonPredicted = FN_i * (E[T_F] + \mu) \tag{4}$$

In the case without prediction, we can reuse the previous formulae using PP=0 and replacing FN by Positives (since we can consider nothing is predicted as positive TP=0 and Positives=TP+FN).

$$\begin{aligned} QoS_{WOi} &= InterventionCost_i + CostofNonPredicted_i \\ &= 0 + CostofNonPredicted_i \\ &= Positives_i * (E[T_F] + \mu) \end{aligned} \tag{5}$$

As a consequence, the gain for each priority is:

$$\begin{aligned} QoS_{Gi} &= C_i(1 - \frac{QoS_{Wi}}{QoS_{WOi}}) \\ &= C_i(1 - \frac{\lambda * PP_i + FN_i * (E[T_F] + \mu)}{Positives_i * (E[T_F] + \mu)}) \end{aligned} \tag{6}$$

Using the formulae for precision and recall, we obtain the following relations equation 7 and 8 :

$$FN = (1 - Recall) * Positives \tag{7}$$

$$PP = \frac{Recall}{Precision} * Positives \tag{8}$$

This simplifies equation 6 as :

$$QoS_Gi = C_i(1 - \frac{\lambda * \frac{R_i}{P_i} + (1 - R_i) * (E[T_F] + \mu)}{E[T_F] + \mu}) \tag{9}$$

The final result of the QoS gain metric can be found in equation 10:

$$QoS_G = \frac{1}{\Sigma C_i}\Sigma C_i(R_i - \frac{\lambda}{(E[T_F] + \mu)}\frac{R_i}{P_i}) \tag{10}$$

This metric was plotted in Fig 2 using $\lambda = 0.3$, $\mu = 1$, and constant precision and recall for each failure priority.

Depending on the adjustment of λ and μ, having a good precision can be necessary or not for good performance. However, when using reasonable adjustment parameters, good recall is always necessary to attain performance.

We will use this metric, with the same setting as in figure 2, along with a cost performance gain metric introduced in previous works and the common classification metrics to measure performance of NFP methods for our dataset.

IV. Comparison of different ML methods

Common ML methods : Decision Tree (DT), Random Forest (RF), AdaBoost and XGBoost, are chosen to be compared. Each model had hyper-parameters tuned until no improvement is attained for several attempts. Table I provides the results of the best performing models, for each type of model.

- DT maximum depth=None, minimum samples leaf=5, class weight='balanced'
- RF maximum depth=20, minimum samples leaf=5, class weight='balanced', number of estimators=40
- AdaBoost maximum depth=20, minimum samples leaf=1, class weight='balanced', number of estimators=30, learning rate=1.5
- XGBoost maximum depth=10, number of estimators=150, learning rate=1.02

The results using these hyper-parameters are compiled in table II. The cost metric introduced in [10] - available at [11] -

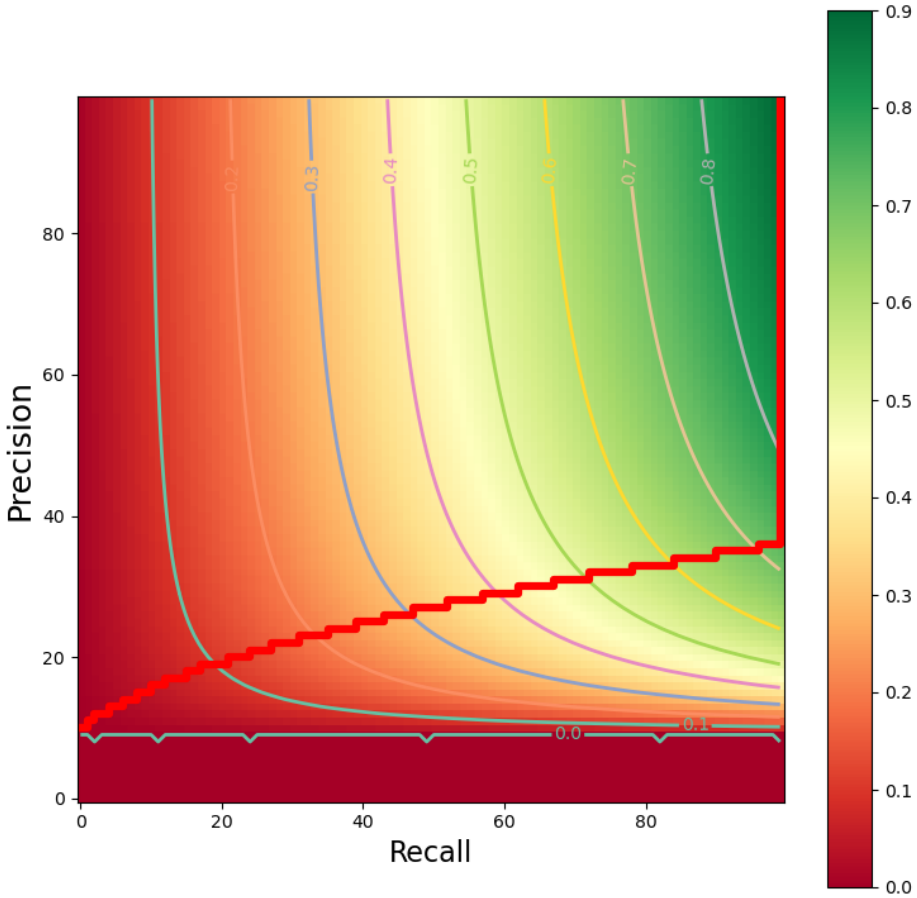

Fig. 2: Theoretical QoS gain of a model according to precision and recall, compared to not having predictions. Parameters: λ=0.3, μ = 1 and $E[T_F]$ = 2 hours. λ and μ can be considered to be in time units for the equation as λ is comparable to network downtime due to intervention, and μ is comparable to perceived lost time by the client. Precision = 80%, Recall=60% and X=1 hour correspond for example to 50% reduction in lost QoS due to failures. The red line shows the optimal path for increasing QoS gain in function of Precision and Recall.

ML Method	Scores			
	Precision	Recall	Cost	QoS
Decision Tree	0.774	0.836	0.422	0.728
Random Forest	0.882	0.904	0.482	0.802
AdaBoost	**1.000**	0.856	0.477	0.770
XGBoost	0.991	**0.924**	**0.513**	**0.831**

TABLE II: Test results on specific optical equipment. **NB:** The Cost and QoS impact metrics were designed for failure prediction and not alarms.

and the newly introduced QoS metric were used to measure performance. Both of these metrics are used with the parameters given when they were described. Let us note that both metrics were created for fault prediction and not alarm prediction.

According to these scores, XGBoost outperforms other DT-based models significantly on this data. This can be explained by the large number of columns where it is expected for XGBoost, having a pruning mechanism, to be able to eliminate useless portions of the data. Having Adaboost with 100% precision would also be very useful if this can be conserved with more data, as this would allow to trust when the model predicts a positive, and run other models when AdaBoost predicts a negative.

V. Conclusion and Perspectives

In this paper, a QoS metric for measuring network performance improvement when using a fault prediction method was introduced. It was used, along with the previous metric, precision and recall to compare DT-based alarm prediction methods. In the case of this study on optical equipment, XGBoost shows convincing performance on our dataset for predicting alarms.

We plan to study the effects of modifying the X, Y and Z parameters of figure 1 in depth to establish the influencing parameters of the problem.

In near future research, more data will be considered (more rows), and failures will be also added to the dataset (other label to predict), to provide a better benchmark reference for NFP methods applied to optical equipment. As failures are a subset of alarms and are potentially a more complex

target of prediction, due to a larger class imbalance and the possible lack of discernability of alarms from failures. Having both labels will allow to potentially combine information to improve prediction performance for both tasks. For example, first predict the presence of an alarm, and then whether or not it becomes a failure (would reduce class imbalance for the second problem).

Sequential methods such as Recurrent Neural Networks, Convolutional Neural Networks, Auto Encoders will also be tested to further complete the benchmark.

REFERENCES

[1] "Calculating the cost of downtime in your business," Axcient, White Paper, 2018. [Online]. Available: https://axcient.com/blog/calculating-the-cost-of-downtime-in-your-business/

[2] "Business value of cisco sd-wan solutions: Studying the results of deployed organizations," IDC, White Paper #US44952119, April 2019. [Online]. Available: https://transform.cisco.com/seller/idcreport_en?xs=99167

[3] M. Boldt, S. Ickin, A. Borg, V. Kulyk, and J. Gustafsson, "Alarm prediction in cellular base stations using data-driven methods," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1925–1933, 2021.

[4] K. Murphy, A. Lavignotte, and C. Lepers, "A machine learning method benchmark for heterogeneous ip network fault prediction," submitted to Globecom2023.

[5] J. Babbar, A. Triki, R. Ayassi, and M. Laye, "Machine learning models for alarm classification and failure localization in optical transport networks," *Journal of Optical Communications and Networking*, vol. 14, no. 8, pp. 621–628, 2022.

[6] V. G. Costa and C. E. Pedreira, "Recent advances in decision trees: An updated survey," *Artificial Intelligence Review*, vol. 56, no. 5, pp. 4765–4800, 2023.

[7] S. Zhang, Y. Liu, W. Meng, Z. Luo, J. Bu, S. Yang, P. Liang, D. Pei, J. Xu, Y. Zhang *et al.*, "Prefix: Switch failure prediction in datacenter networks," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 2, no. 1, pp. 1–29, 2018.

[8] Y. Zhao, K. Tian, Y. Wang, B. Zhang, Z. Li, Z. Zhao, and J. Zhang, "Failure location and prediction with cross-layer ai in self-optimized optical networks (soon)," in *2021 Asia Communications and Photonics Conference (ACP)*. IEEE, 2021, pp. 1–3.

[9] Z. Tan and P. Pan, "Network fault prediction based on cnn-lstm hybrid neural network," in *2019 International Conference on Communications, Information System and Computer Engineering (CISCE)*. IEEE, 2019, pp. 486–490.

[10] K. Murphy, A. Lavignotte, and C. Lepers, "Fault prediction for heterogeneous telecommunication networks using machine learning : a survey," submitted to Transactions on Networks and Service Management.

[11] "Code for cost function," https://github.com/Serpoignet/NFP_gain_cost_formula, accessed: 2023-07-12.