Avoiding "Hot Potato" Problems in Internet Service Providers

Khanh Huu The Dam
ICTEAM/INGI
UCLouvain, Belgium

Gorby Kabasele Ndonda
ICTEAM/INGI
UCLouvain, Belgium

Axel Legay
ICTEAM/INGI
UCLouvain, Belgium

Ramin Sadre
ICTEAM/INGI
UCLouvain, Belgium

***Abstract*—Internet service providers (ISPs) strive to provide the best possible services to their customers. Service outages, or incidents, due to technical failures are inevitable, so the aim of ISPs must be to respond as quickly as possible to error notifications. However, services may rely on thousands of devices and components that are interconnected and managed by different teams (network administrators, technicians, etc.). Identifying the team to which an incident ticket should be assigned becomes a tedious task that slows down recovery time.**

In this paper we focus on the problem of finding the right team when an incident occurs. We group teams into logical team groups and use machine learning models that we train on previous resolved incidents to predict the most appropriate team group from a failure description. Using a large dataset from a national ISP and telecommunication company, we demonstrate that, even with a small amount of information available at the beginning of the incident, machine learning models can achieve an accuracy of 88.52% and an F1 score of 90.17%. With more complete information about the incident, the accuracy and F1 score increase to 90.52% and 91.7%.

***Index Terms*—Network management, failure handling, Machine Learning.**

I. Introduction

Given the highly interconnected world that we live in, computer networks and the Internet have become essential services. This evolution has been made possible by the development of powerful network equipment, reliable and secure network protocols, and numerous tools and methods for the efficient and effective management of large-scale networks.

However, in any kind of network, failures of equipment and service outages, or, more generally, *incidents*, are inevitable. In the case of ISPs and other types of telecommunication companies, such incidents lead to the loss of customer trust or even breaches of Service Level Agreement (SLAs) and thus can have a significant negative financial impact [1]. To avoid this, network operators must respond to incidents as quickly as possible and perform appropriate repair actions in order to restore the offered services. Service time recovery is therefore an important aspect of the service quality.

Unfortunately, several obstacles slow down the response. When a service fails, the underlying cause is not always obvious and it requires special knowledge and experience to identify the cause and fix it. Therefore, an essential step in the incident handling process is to find the most appropriate team or team type (e.g., core network administrators, access network technicians, etc.) for the incident. This is not an easy task, as there might be many teams, each responsible for different services or components. For example, the company the experimental study in this paper is based on has 73 teams. Furthermore, it can happen that a malfunctioning service cannot be fixed by the team responsible for that service because the actual root cause is located in a different component the service depends on or because the team has not the right experience. Identifying the right team can take time and is a dynamic process during which the incident ticket moves from one team to another like a hot potato that no one wants to hold until more information about the incident is available.

In general, identifying the right team depends on two sources of information. The first source is the available dynamic information on the incident. An incident ticket starts with very little information, such as a customer complaint, and is enriched with more details during its lifetime. The second source is the knowledge of the quasi-static structure of the network and the dependencies between its services and components. The extent to which this knowledge is centralized and available for an automated root cause analysis varies from company to company and service to service.

In this paper, we tackle the problem of identifying the most appropriate team to fix incidents in a large-scale network by leveraging machine learning techniques. Our system learns the relationship between incidents and the incident-resolution teams from previously resolved incident tickets. We evaluate our system on a two-year long dataset from a national ISP and telecommunication company containing more than 46,000 resolved tickets. Due to various legacy components still in operation, the company does not maintain a complete and up-to-date description of its network. Our system can therefore only rely on the information contained in the tickets. Furthermore, since some teams only resolve one or two tickets per year, we group teams into larger logical team groups.

The experimental evaluation shows that our system can identify the most appropriate team group to fix a problem with an accuracy of 88.5% and an F1-score of 90.17% when applied very early in the ticket lifetime. The accuracy and F1-score increase to 90.25% and, respectively, 91.7% when we do the identification later in the lifetime of the ticket, i.e., after

This work was funded by the Win2Wal PrEDICT project #2110102 of the Région Wallonie, Belgium.

the human operator has already started a manual investigation of the problem and added more information to the ticket.

The rest of the paper is organized as follows. We start with a discussion of related work in Section II. We give a detailed description of the operating scenario as it presents itself for the company under study in Section III. We then describe our methodology in Section IV and present the evaluation results in SectionV. The paper is concluded in Section VI.

II. Related Work

Failure handling in IT infrastructures has been intensively studied in existing works [2]. Their focus is on detecting failures [3], understanding their nature and root cause, and assessing their impact on the network. Other works address the problem of post-incident report handling [4]. The used techniques vary greatly depending on the available information. Takeshita *et al.* detect network failures by monitoring social media looking for discontent posts from customers [5]. They create a machine learning model to distinguish posts about network failure from others. Turner *et al.* use several data sources to analyze the duration of failures, their impact and the cause [6]. The data sources include router configurations to obtain the network topology, *syslog* events to get the dynamic aspects of the networks, the tickets, and mailing lists of operators. They create a set of rules based on time correlation to group failures and examine the tickets to find the cause. Gill *et al.* analyze failures in datacenters and leverages statistical techniques to find the most unreliable devices and network links as well as the most likely cause [7]. Kandula *et al.* diagnose network failures and application failures [8]. They extract information from logs and use it to create a dependency graph of fine-grained components (processes, network devices, firewall,...) where the edge weights, representing the likelihood of dependency between components, are computed from the joint historical behavior of the components.

Our work differs from the above ones in two ways: first, because of the limitations imposed by the studied use case (see Section I), only little information about the network and its dynamic state (outside the incident ticket itself), are available to our system. Nevertheless, we show that our solution can identify the suitable team group to resolve an incident with high accuracy. Second, our end goal is not just to identify the cause of a failure, but rather the team that has successfully resolved similar incidents in the past.

III. Scenario

A. System Overview

Network monitoring is an important task of network management as it allows to detect when failures occur. Figure 1 shows a high-level view on the components of a typical monitoring system that can be found in one form or the other at telecommunication and ISP companies. The network infrastructure is composed of several devices, such as routers, switches, and servers. Those devices are monitored by specific services that are part of a monitoring infrastructure and that collect alarms signalling an undesired state of any one of the devices. In Figure 1, we consider those services as a single logical entity and we refer it as the Alarms Collector[1]. Depending on the devices and how they are configured, the sending of alarms can be done periodically or sporadically. In the first case, devices send alarms about their state regardless of whether or not they are malfunctioning while in the second case, only an error will trigger an alarm. The Alarms Collector processes the alarms (pre-filtering, formatting, etc.), before writing them to the Alarm Logs. Those logs are analyzed by the Network Operating Center (NOC) who determines the severity of the situation. Depending on the NOC's assessment of the situation, incident tickets are created. Incidents occur when one or several services provided by the company are not running correctly. An incident ticket is a collection of any useful information for fixing the incident. Analysts at the NOC populate the tickets with all relevant information they know about the system, for example the services that are impacted by the incident.

Incident tickets can be also created for two other reasons. The first one is when the NOC is contacted by other departments to report an issue. For example, Customer Care can contact the NOC because clients are complaining that something is not correctly working. Finally, the Alarms Collector automatically classifies alarms by level of severity based on rules defined by the network operators. Alarms classified as critical lead to the creation of a ticket.

Once the tickets are created, there are assigned to the most likely relevant team at the company who is then tasked to investigate the incident and respond to it.

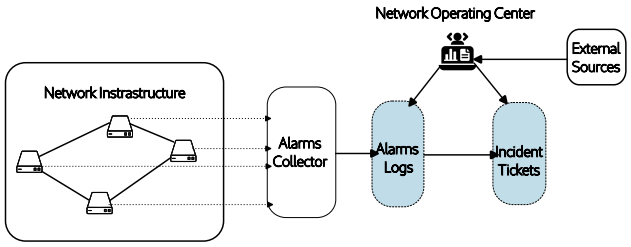

Fig. 1. Architecture of a network monitoring system

B. Incident tickets

Incident tickets consist of a large number of information fields. Some of them are filled at the creation of the ticket, others are filled or updated later when more information about the incident becomes available.

The concrete format of a ticket varies between operators. In the specific use case discussed and evaluated in this paper, a national telecommunication and ISP company, the ticket consists of 59 fields. After removing those that are not always used or contain redundant information or information not relevant for the task of team identification, 20 fields are left, which we describe in the following.

[1]In practice, the Alarms Collector can consist of several physical or virtual machines that are responsible for different parts of the network.

Each ticket is automatically assigned a unique identifier *ticket_id*. A textual *description* of the incident is linked to the ticket. This description is manually written by the analyst. *time_slot* specify whether the incident was reported during the day or during the night.

As mentioned in Section III-A, tickets are not only created from the analysis of alarms but also from reports by other departments than the NOC, like the Customer Care service. The field *source* gives the source of the incident report.

The incident management workflow uses a double ranking system to quantify how critical an incident is. The corresponding fields are *priority* and *urgency*. The *priority* field gives information about how many services are impacted by the incident. The *urgency* field describes how urgent it is to resolve the incident. An incident with high urgency may have a low priority if it does not impact many services, while a high priority incident can only have a medium urgency if it does not require immediate attention. The order in which the incidents will be dealt with depends on those two values.

The remaining fields contain information about the incident itself. *impacted_services* stores the list of all the services impacted by the incident. This field is divided into six sections to group the impacted services based on the purpose of the services. There are six groups: access network, mobile, IT, telephony, environment and TV. These services can be related to either internal services, services used by the company for its operation, or the services offered to the customers.

The *service_level_impact_percentage* field gives a quantitative estimation of the negative impact of the incident on the services. The field *impact_qualification* gives information about the behavior of the failure. *network_topology_element* lists the network topology elements related to the incident. This fields is separated in two sections. The first gives the parent station, a zone in the network to which many nodes are connected. The second section is the list of elements.

The fields *number_of_impacted_customer_estimate*, *customer_impacted_services* *impacted_customer_type* give respectively an estimation of the number of customer that are impacted by the incident, the list of customer-oriented services impacted, and the list of the types of impacted customers, typically business customers or regular customers.

Finally, the last information in the ticket is the field *most_relevant_team_in_charge* which specifies the team that ended up resolving the incident.

C. Ticket handling workflow

During an incident, not all information about the incident is directly available and some investigation is required to understand its impact and nature. We divide the life cycle of a ticket into five states, as shown Figure 2. The information fields of a ticket will be available at specific states, as indicated in the column "Available in" of Table I. When the ticket is created (state OPEN), only few information is available. The analyst will then do a first analysis of the incident, thus adding more information (state BASIC ANALYSIS), and assigns the ticket to a team. The field *impacted_services* is a dynamic field that is updated as investigation progresses to the DEEP ANALYSIS state or when the cause has been identified and there is no more alarms regarding the incident (state CAUSE IDENTIFIED). It may happen that there is a back and forth between the DEEP ANALYSIS and the CAUSE IDENTIFIED state if it is discovered during the attempt to resolve the incident that the cause was not correctly identified. The incident is considered as FINISHED when there is no more impact on any services. Only then it is definitely known which team resolved the incident.

Fig. 2. Ticket handling workflow

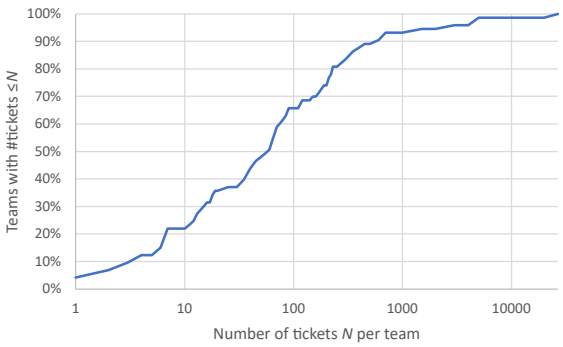

Fig. 3. Empirical CDF of the number of tickets per team

IV. Methodology

Our goal in this work is to identify the most appropriate team to resolve an incident. We leverage machine learning models to predict the team group for a ticket from the fields that are available during the incident treatment (see Table I).

A. Data collection and preparation

We collected a dataset of 46K incident tickets that were created and resolved in 2021 and 2022 at a national ISP and telecommunication company. The incidents were dispatched among 73 teams by the ticket handling workflow described in Section III-C. We observe that the number of tickets per team is highly imbalanced and varies between 1 and 26K (mean=640, median=59). As shown in Figure 3, 30% of the teams saw less than 15 tickets each, and only 7% of the teams resolved more than 1000 tickets each.

It is hard to implement an efficient machine learning model for such a dataset. We explored different approaches to overcome the imbalance, such as resampling. However, as the dataset contains several teams with just one or two incident tickets, upsampling is difficult. In order to obtain a solution that is practicable, we group logically related teams into *team groups* and perform the ticket assignment task on those groups instead. For example, we group the teams IT-WEB, IT-EIM, IT-GIS, etc. into the IT group since these

TABLE I
DESCRIPTION OF THE TICKET FIELDS

Field Name	Description	Available in	Feature-attribute
ticket_id	Identifier of the incident.	OPEN	NO
description	High level description of the incident.	OPEN	NO
time_slot	Information on whether or not the incident was reported during the night or the day	BASIC ANALYSIS	YES
source	Source from where the incident was reported like the monitoring workflow or the Customer Care service	OPEN	YES
priority	Priority of the incident in term of service impacted	BASIC ANALYSIS	YES
urgency	Urgency of the incident	BASIC ANALYSIS	YES
impacted_services (divided in 6 sections)	List of the services impacted by the incident, such as access network, mobile, IT, telephony, environment and TV.	DEEP ANALYSIS	YES
service_level_impact_percentage	Aggregate value of how much the services are running correctly	BASIC ANALYSIS	YES
impact_qualification	Information on the behavior of the failure, intermittently or permanently	BASIC ANALYSIS	YES
network_topology_elements (2 sections)	List of the network topology element related to the incident, such as the impacted elements and their parent stations.	BASIC ANALYSIS	YES
number_of_impacted_customer_estimate	An estimation of the number of customers impacted by the incident	BASIC ANALYSIS	YES
impacted_customer_type	List of the type of customer impacted by the incident	BASIC ANALYSIS	YES
customer_impacted_services	List of customer-oriented services impacted by incident	BASIC ANALYSIS	YES
most_relevant_team_in_charge	Team who resolved the incident	FINISHED	/

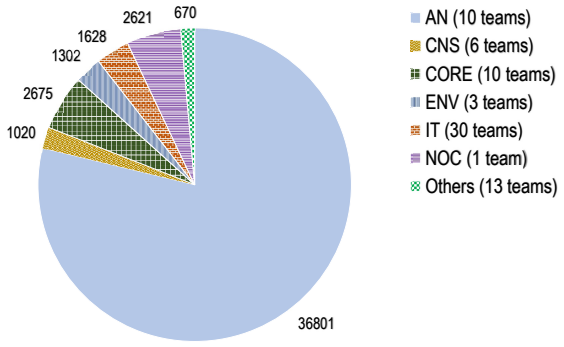

Fig. 4. Distribution of tickets and teams per team group

teams handle incidents concerning the IT services. We obtain 7 groups: AN, CNS, CORE, ENV, IT, NOC, and "Others". The latter is a collection of teams which have only few incident tickets and cannot be logically assigned to one of the major six groups. Figure 4 reports the details of the resulting dataset. Although the data is still imbalanced, the number of incident tickets in each group is large enough to train and evaluate a machine learning model.

B. Feature selection

The monitoring system presented in Section III records 20 main fields for an incident, as shown in Table I. As explained in Section III-C, the number of available fields depends on the state of the ticket. Consequently, the earlier we want to predict the team the less fields are available as features for a prediction model. After the BASIC ANALYSIS, the feature set consists of 11 fields. Six more fields become available if we wait until the DEEP ANALYSIS.

C. Classification model

We employ a classification model to identify the team group to which a ticket should be assigned. The model should take the field values of the ticket as input, and produces a group for the incident as output. The input values are a mix of numeric and categorical data on which machine learning based classifiers do not work well in general. Thus, we plug an one-hot encoding system into the model to convert the input values into feature vectors. The encoding system converts categorical data to one-hot vectors. Then, these vectors are concatenated with numeric data to obtain the final feature vector that is fed to the classifier for classification.

In this work, we implement four common machine learning techniques: Logistic Regression (LR), Random Forest (RF), Gradient Boosted Trees (GBT), and Support Vector Machines (SVM). These techniques are representative models of the statistical model, the decision tree based model and support vector machines. They have been proved working well for classification. LR is based on a statistical model that computes the probability of a label [9]. RF is a combination of multiple decision trees [10]. Since the outcome of the classifier is derived from the results of its decision trees, overfitting is mitigated. While RF produces the output by a majority voting of its decision trees, the output of GBT is the outcome of a sequence of decision trees where each decision tree tries to minimize the error from the previous tree [11]. Finally, SVM computes support vectors separating data points [12]. This classifier works well for binary classification. For classification by team group, we implement a One-vs-Rest SVM which consists of multiple SVM-based classifiers. The decision is made by the classifier producing the highest weight output.

These classification models are trained on a balanced training set, which is sampled 500 tickets for each team group from the data collection in Section IV-A. Then, the rest samples are

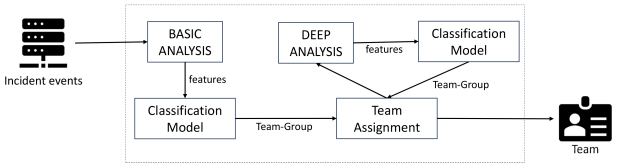

Fig. 5. New model-assisted ticket handling workflow

TABLE II
PERFORMANCE OF THE CLASSIFICATION MODEL

Classifier	11 features		17 features	
	Accuracy (%)	F1 (%)	Accuracy (%)	F1 (%)
LR	66.65 ±2.7	74.39 ±2.2	77.21 ±3.9	82.23 ±2.7
RF	71.32 ±2.1	77.64 ±1.7	84.38 ±1.3	87.53 ±0.9
GBT	**88.52** ±1.0	**90.17** ±0.7	**90.25** ±0.5	**91.7** ±0.4
SVM	72.5 ±0.9	78.76 ±0.8	85.68 ±1.3	88.38 ±0.9

used for the test set (See Section V).

D. Team assignment workflow

By integrating our classification models into the ticket handling workflow shown in Figure 2, we obtain the model-assisted workflow shown in Figure 5. When an incident event arrives, the BASIC ANALYSIS performed by the analyst makes 11 features available for the first classification model, which identifies a first team group to which the incident ticket is assigned to. Then, the process enters an iterative phase when the ticket is in the DEEP ANALYSIS state. In that state, new information comes in about the incident, which either adds new features (resulting in 17 features) or updates the existing ones. Each time this happens, the second classification model is triggered to identify a better matching team group. The process finishes when the ticket reaches the FINISHED state.

V. EVALUATION

We evaluate the performance of our classification model with the 4 machine learning classifiers LR, RF, GBT, and One-vs-Rest SVM. First, the dataset is randomly split into two partitions: the training set and the test set. The training set consists of 500 incident tickets for each group. The test set consists of 43217 incident tickets. The splitting is repeated 10 times. Each pair of training set and test set is used to train and evaluate the classification model. The performance of the model is measured by the average of the accuracy rate and the F1-score over 10 splittings.

The experiment runs on a Spark cluster of 4 nodes. Each node is a virtual machine equipped with at least 4 CPUs and 16GB RAM. The Hadoop system is deployed on the same cluster for our dataset. We take advantage of the MLlib for Spark to implement our classifiers. In this experiment, the RF classifier is configured with *maxDepth* = 22, *maxBins* = 32, *numTrees*=42. The other classifiers (LR, GBT, and SVM) take the default configuration of the MLlib.

We first evaluate all classifiers with two feature sets, i.e., the 11 features available after BASIC ANALYSIS and the 17 features available after DEEP ANALYSIS. Table II shows the accuracy rates and F1 scores of each classifiers and their

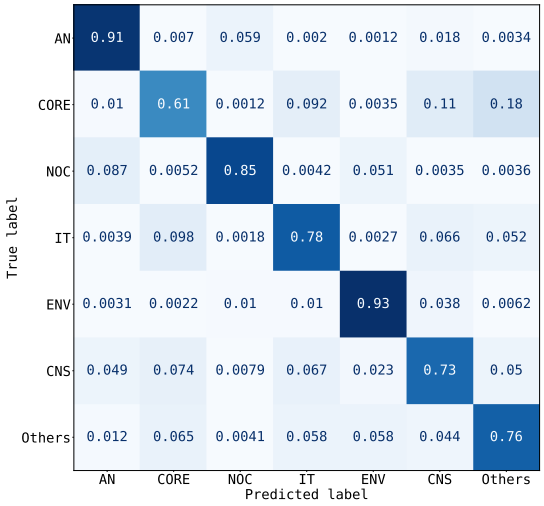

Fig. 6. Confusion Matrix of the GBT Classifier with 11 features

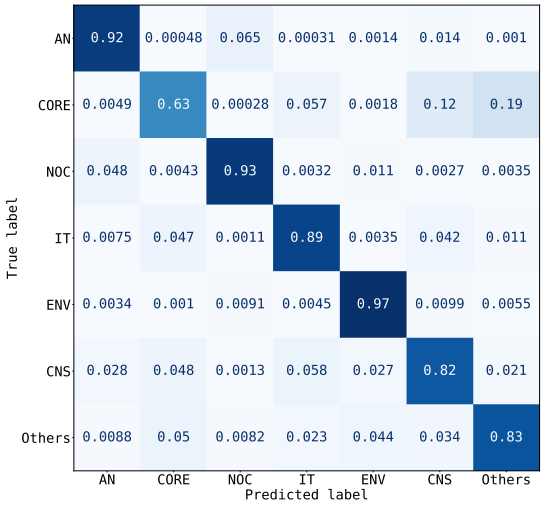

Fig. 7. Confusion Matrix of the GBT Classifier with 17 features

standard deviation over 10 runs. The results show that the model improves when the number of features increases from 11 to 17 features. A significant improvement of around 10% is found for the classifiers LR, RF and SVM, while the performance of the GBT classifier increases less than 2%. However, the GBT classifier achieves a F1-score of nearly 92%. It outperforms the other classifiers on both feature sets. Therefore, we further investigate the results of the GBT classifier.

Figures 6 and 7 show the confusion matrix of the GBT classifier with 11 features and 17 features, respectively. According to the matrix, the classifier is challenged by the CORE group, since its accuracy rate is lower than 65% with both feature sets while the other groups get over 72% and over 82% with 11 features and 17 features, respectively. Significantly, with both

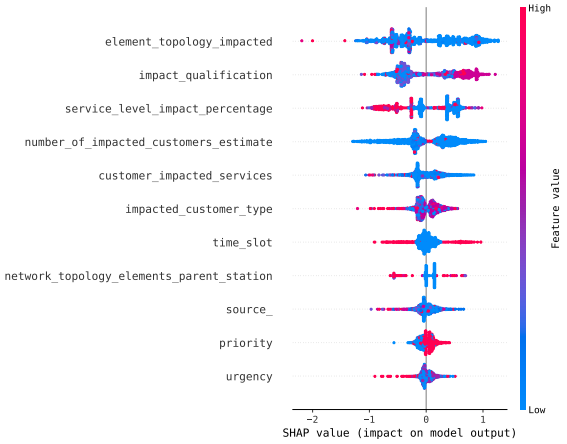

Fig. 8. Mean SHAP values of the GBT classifier with 11 features

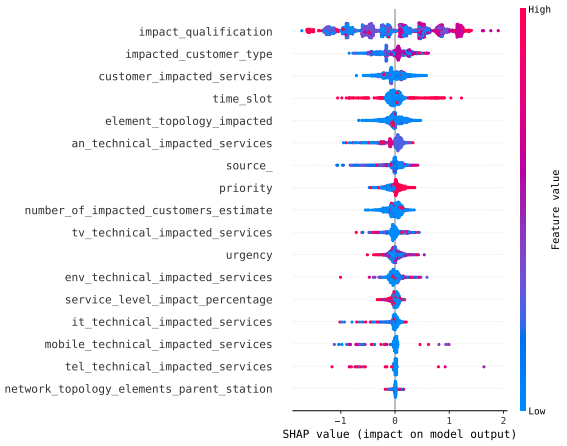

Fig. 9. Mean SHAP values of the GBT classifier with 17 features

feature sets, the dominant group (AN) achieves above 90%. Moreover, the highest accuracy rate of 97% is achieved for the ENV group. The accuracy for the IT group increases about 11% after the deep analysis. Despite of the improvement, the increased number of features can also increase the false positive rates between groups, such as for AN, NOC and IT.

We also examine the features which are the most impactful to the classifier using SHAP [13]. Figures 8 and 9 show that the top 5 features stem from the BASIC ANALYSIS, followed by the *an_technical_impacted_services* , (one of the sections of the *impacted_services*) field which is only available after the DEEP ANALYSIS. This shows that an initial analysis of the incident by the analyst is an important basis for identifying the right team group. As shown in Table II, the accuracy is 88.5% after the BASIC ANALYSIS has been done. Nevertheless, the prediction outcome improves with the number of features.

Although we cannot prove this with the historical dataset available to us, in practice it can be expected that identifying the right team group already in BASIC ANALYSIS will make the entire incident response more efficient and effective, as the right people will be working on the incident early in the life cycle of the ticket. Also, team assignment depends on the analyst experience, so depending on the analyst, the decision may change. We were told that analysts sometimes dig into old tickets to identify teams, so automating this process would remove this time consuming process.

VI. Conclusion and Future Work

In this paper, we address the problem of predicting the team to which an incident should be assigned at a national ISP. We collect incident tickets over a period of two years where we group teams into logical team groups, and evaluate different machine learning algorithms for prediction. We show that even with only basic information about the incident, we achieve an overall accuracy of 88.52% and an F1 score of 90.17%. When more information is added to tickets, the accuracy and F1 score increase to 90.25% and 91.7%, respectively.

Currently, we do not use all the information contained in a ticket. The *description* field can contain valuable information to classify the type of failure. However, exploring this field is challenging because its content is manually written by the human analysts and it does not follow a specific structure. Furthermore, the language of the description may vary from ticket to ticket (French, English, or a mix) and make use of non-standard abbreviations, which further complicates the extraction of relevant information. We plan to use natural language processing (NLP) techniques similar to Dogga *et al.* [4] in the future to make efficient use of this field.

References

[1] T. N. Quach et al, "Internet service providers' service quality and its effect on customer loyalty of different usage patterns," *Journal of Retailing and Consumer Services*, vol. 29, pp. 104–113, 2016.

[2] M. łgorzata Steinder and A. S. Sethi, "A survey of fault localization techniques in computer networks," *Science of Computer Programming*, vol. 53, no. 2, pp. 165–194, 2004, topics in System Administration.

[3] E. C. Molero et al., "Fast in-network gray failure detection for isps," in *Proceedings of the ACM SIGCOMM 2022 Conference*, 2022, pp. 677–692.

[4] P. Dogga et al., "{AutoARTS}: Taxonomy, insights and tools for root cause labelling of incidents in microsoft azure," in *2023 USENIX Annual Technical Conference (USENIX ATC 23)*, 2023, pp. 359–372.

[5] T. Kei et al., "Early network failure detection system by analyzing twitter data," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015, pp. 279–286.

[6] D. Turner et al., "California fault lines: understanding the causes and impact of network failures," in *Proceedings of the ACM SIGCOMM 2010 Conference*, 2010, pp. 315–326.

[7] P. Gill et al., "Understanding network failures in data centers: measurement, analysis, and implications," in *Proceedings of the ACM SIGCOMM 2011 Conference*, 2011, pp. 350–361.

[8] S. Kandula et al., "Detailed diagnosis in enterprise networks," in *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, ser. SIGCOMM '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 243–254.

[9] S. Sperandei, "Understanding logistic regression analysis," *Biochemia medica*, vol. 24, no. 1, pp. 12–18, 2014.

[10] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[11] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Frontiers in neurorobotics*, vol. 7, p. 21, 2013.

[12] W. S. Noble, "What is a support vector machine?" *Nature biotechnology*, vol. 24, no. 12, pp. 1565–1567, 2006.

[13] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 2017, pp. 4765–4774.