Fault Prediction in the Crowd?

By

NILS P. PEDERSEN

A Dissertation submitted in partial fulfilment
of the requirements for the degree of
MSc in Big Data and Digital Futures

Supervised by TESSIO NOVACK

University of Warwick
Centre for Interdisciplinary Methodologies
September 2020

“Det er en vanskelig sak å spå om fremtiden”

(Paasche, 1918)

Abstract

An investigation was conducted into a 40 GB, 326 million record event dataset. This dataset contained anonymised event information representing performance, availability and security issues of 172,000 network devices from approximately 150 Cisco Systems customers. It was hypothesised that network device event data gathered from one customer environment could be used to predict events in another customer environment. After analysis of the dataset, a binary model was developed to predict when a process might request too much compute resources on a device. The model was developed on one set of customer data and tested on another unseen set of customer data. The Matthews correlation coefficient for the model on the unseen test data was 0.66, the F1 score was 0.72, and the False Negative rate was 27%. This was a substantial improvement over a model with no skill.

List of Contents

List of Tables

List of Figures

Acknowledgements

I would like to express my sincere gratitude for all the help and support I have had while completing this dissertation.

- Tessio Novack for being my dissertation supervisor
- Michael Castelle for suggesting the original research direction
- Emma Uprichard and Andreas Murr for advice with the practicalities of conducting the research
- James Tripp and Iain Emsley for advice dealing with the large datasets and help with R
- Don Allen and Dmitry Goloubev for providing the data
- Shishir Srivastava for help with SQL
- Christopher Adare for help with Python
- Rebecca Floyd and Matthew Leeming for proofreading the dissertation

And finally, my parents - without their support, this would not have been possible.

One of the fundamental goals of network management is predicting when faults will occur, to avoid potential network failures, service and performance degradation (Boutaba et al., 2018). This dissertation will describe an investigation into network device data gathered from one customer environment and how to apply that data to support device management tasks in another customer environment. For example, if it is possible to predict whether a network switch will experience an operational issue in one customer's network by analysing that issue in another customer's network?

With the growing pervasiveness of computing systems, it is essential that we place trust in the services they deliver (Eusgeld et al., 2008, p. v). Computer networks have become an essential component in facilitating business processes in a global economy (Allen & Goloubew, 2020, p. 1). Proactive fault management is a method of enhancing the availability of these systems, and short-term predictions are especially effective in preventing or limiting the damage caused by these failures (Salfner et al., 2010, p. 10.1). The motivation for this investigation is to increase the availability of the services provided by these networks by predicting network device faults. By providing a warning of an impending fault on a device, network administrators will be able to take remedial action prior to the fault occurring and potentially impacting services that the business is reliant on.

The structure of this document is as follows: there will be a description of data and how it maps to the motivation of this dissertation. Next, a discussion of the literature will be presented, both from within the field of information technology (IT) operations analytics[1] and other domains with similar data properties and structure. Then, the model and feature development and analysis will be described.

Dataset

Cisco Systems, Inc., is involved in designing and selling a range of products and services across networking, security, communication, applications and the cloud. It also offers technical support and advanced services. A part of Cisco's product and services portfolio includes infrastructure platforms; which constitute its core networking technologies of switching, routing, data centre and wireless products (Financial Times, 2020).

As part of its goal of providing customers with improved business continuity and risk management, Cisco developed a service to identify device issues in these core networking technologies proactively. The service leveraged the collective diagnostic and remediation knowledge and experience from Cisco Technical Assistance Centre (TAC) support engineers. The primary goal of the service is to proactively identify device issues before they become problems that could significantly impact network performance, availability and security (Cisco Systems, 2017). The service is known as Connected TAC and has been marketed as a limited-time trial service to allow Cisco customers to run diagnostics routines on one or more devices at a time, either through the command line or polled automatically through an application installed on an on-premises Microsoft Windows server (Cisco Systems, 2020).

The event data used in this investigation was obtained from the Cisco Connected TAC development team. The events represent network device status information collected by the Cisco Connected TAC service. The data was generated by customers taking part in the Cisco Connected TAC trial. Due to the proprietary nature of the data, some of them have been anonymised to protect commercial and intellectual property. Table 1 shows a description of the main attributes of the dataset.

Table 1

Attributes and Values

Attribute	Description
hit_date	Date/time when an event occurs
device_id	Unique identifier for each device (anonymised)
gateway	Software/hardware architecture, for example, Cisco IOS or NX-OS
hit_issue_id	Unique identifier for each issue type detected on a device (anonymised)
hit_labels	Tags or keywords describing the issue. A string with labels separated by colons (partially anonymised)
hit_module	Issue type - logic that triggers a specific issue (partially anonymised)
hit_severity	The severity of the issue - Danger, Warning, Info, OK
cu_id	Unique identifier for each issue customer (anonymised)

The dataset includes approximately 326 million events (or hits), from 130 customers, for 15 million issue life cycles on approximately 172,000 devices over a period of thirty months, from January 2017 to November 2019. The hit_issue_id is unique for that issue on that specific device. Thus, if an issue was resolved but then reoccurred, the hit_issue_id would remain the same. Appendix A – First 100 shows the first 100 rows of the dataset, while
Table 2 shows an example row.

Table 2

Typical Dataset Row

hit_date	device_id	gateway	hit_issue_id	hit_labels	hit_module	hit_severity	cu_id
2017-01-01 00:43:18	d2	g2	h13	:Automation:Prod6052_FW_ Appliance:Prod6044_ Prod6027_Prod6009_ Series_Adaptive_Security_ Appliances:Diagnostic _Signature	ip_audit_ Prod	ok	c2

Note: Individual hit_labels or tags are separated by colons. Thus, in this example, there are four separate labels, Automation, Prod6052…, Prod6044…, and Diagnostic_Signature.

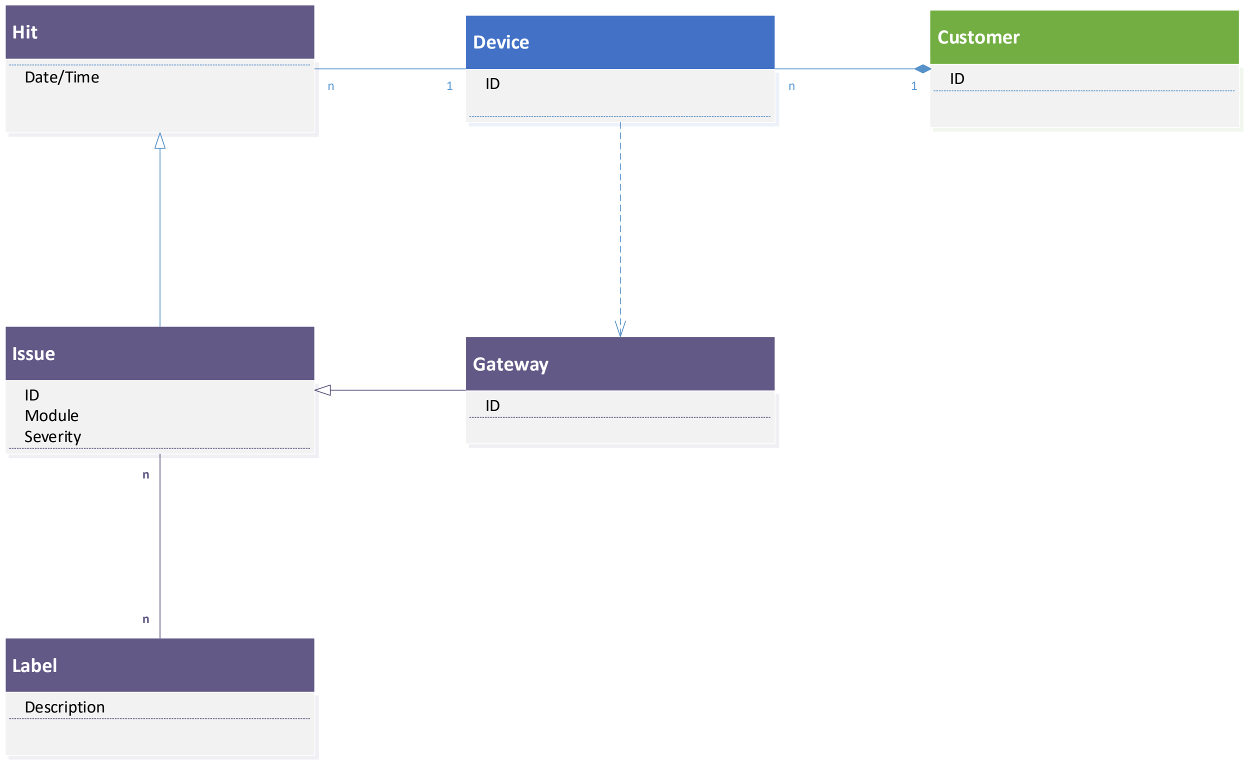
Casual observations of the dataset seemed to imply that anonymization seemed to remove specific customer and product information while retaining as much of the semantic information as possible.

Figure 1 shows the class diagram for the information described in the data. There are three main objects, hit, device and customer. A hit is an event which captures the severity of an issue (triggered by the hit_module logic) at a point in time. The severity may be Danger, Warning, Info or OK. The code defining the module logic can only trigger events on a specific architecture or gateway. For example, one might have conceptually similar issues (for example, out of memory) for different Cisco operating systems like IOS or NX-OS on switches, but they would be implemented as different modules, under different gateways, and have different names. Thus, the same type of conceptual issue may be implemented in multiple separate modules yet, other than what may be inferred from the module name and labels the data does not provide that association.

No metadata is provided in the dataset - only the event data itself. For example, no information is provided about what industries customers represent. In addition, other than deducing from labels, it is not known what the product version or type the device belongs to. Perhaps more importantly, it is not known from the dataset if a device can experience an issue unless that device has already previously experienced that issue. In an analogy from the medical world, it would not be possible to predict a prostate cancer diagnosis, unless that patient had already received a previous prostate cancer diagnosis. Also, it would not be known if the patient was male or female – and thus would be unable to develop prostate cancer. An issue taken from the dataset might be Prod357_Enable_Password. As the device product type is unknown, unless that device has already experienced that issue, it is not possible to determine which devices can experience that issue or not.

Figure 1

Class Diagram

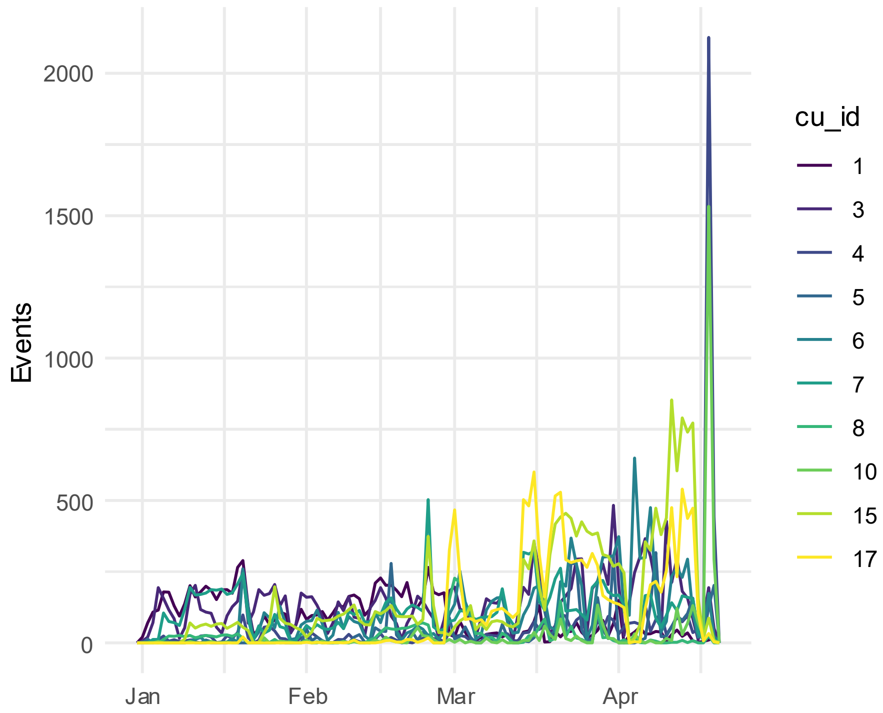


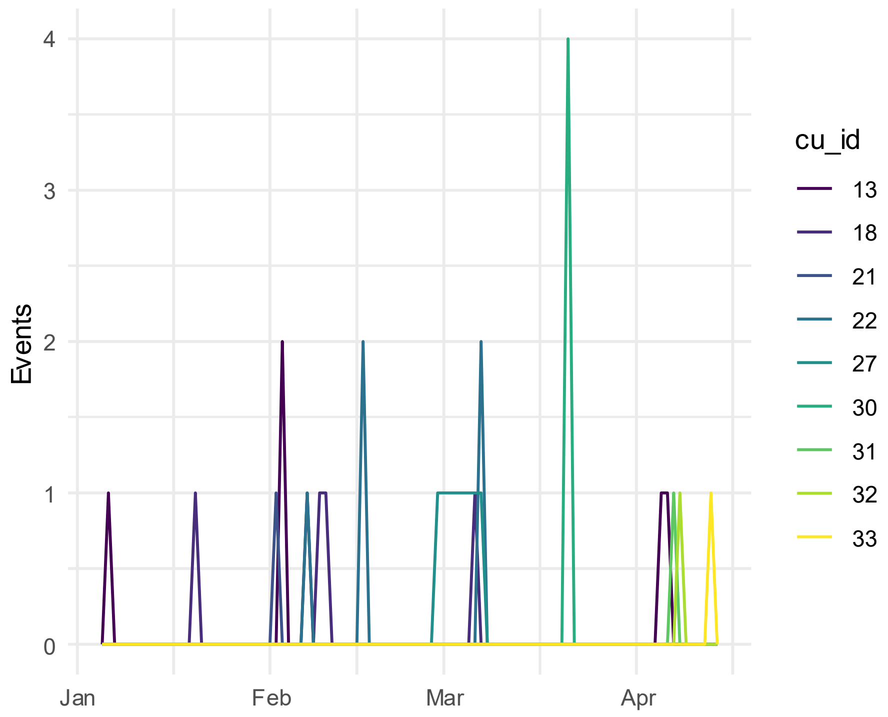
An initial investigation of the data was conducted. The analysis was initially limited to the first one million hits of the dataset because of software and hardware constraints.

Figure 2 shows the number of event counts for the most and least active customers from January 2017 to mid-April 2017. Based on the event count by customer graphs shown in Figure 2, one might hypothesise that high event volume customers used the automated event gathering feature of Connected TAC, while those that generated one or so events per month used the manual method.

Figure 2

Events for Top and Bottom 10 Customers

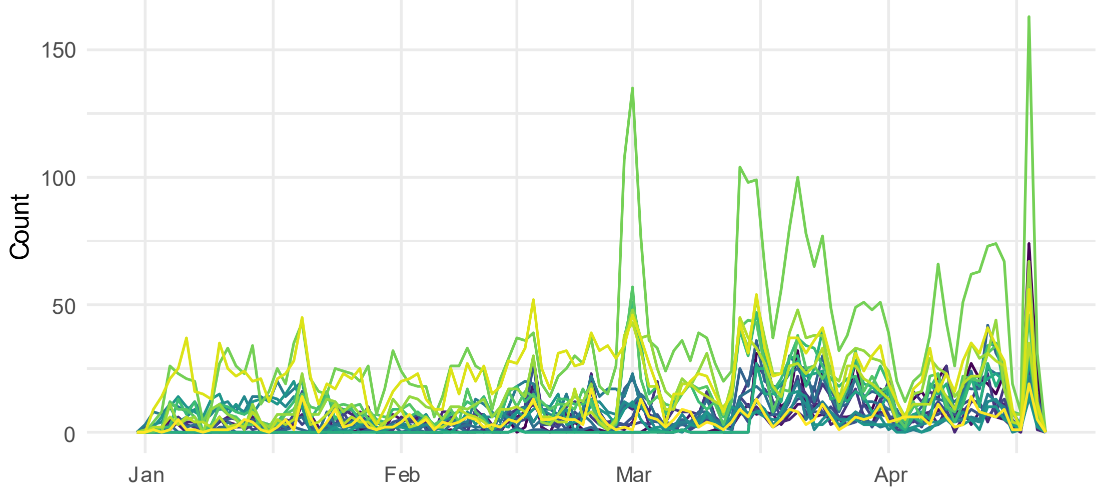




cu_id = Customer ID

Figure 3

Top 20 Issues by Date and Ordered from Most to Least



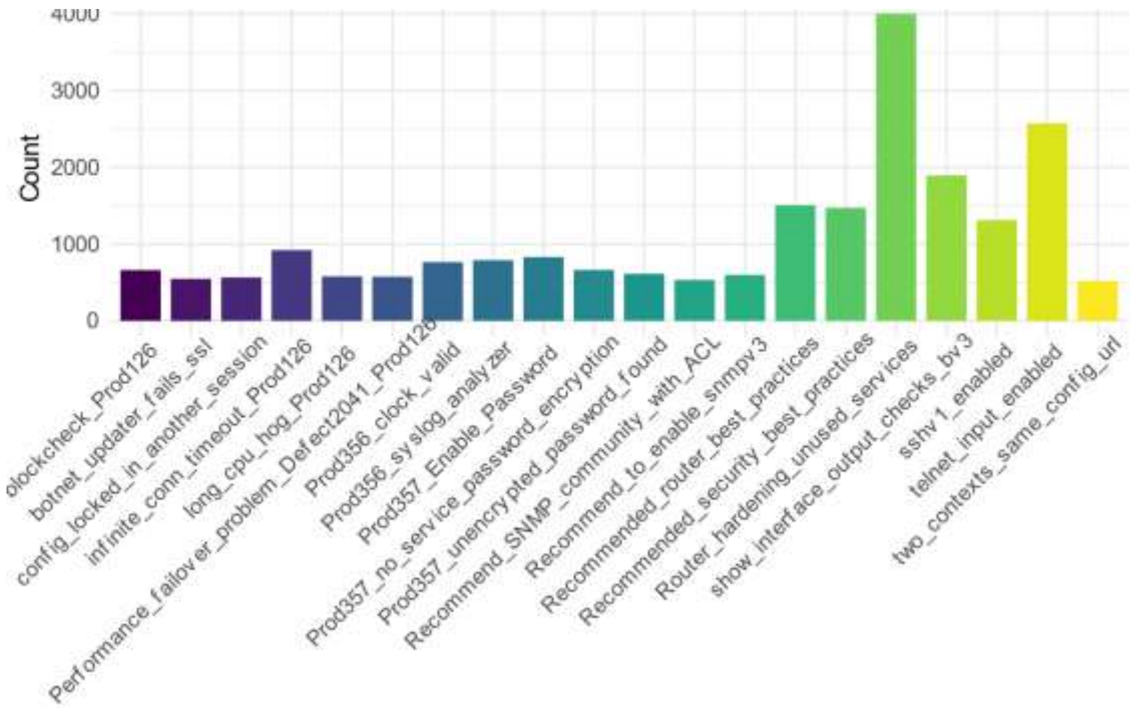


Figure 3 shows the event count for the issues with the top 20 number of events from the first one million hits. Most of the issues seem to be configurational in nature, for example, telnet_input_enabled might refer to the fact that a device has its telnet[2] service enabled (and consequently may be more vulnerable to a security breach). Only recently, Cisco published details of a Telnet vulnerability. In that case, the interim remedy was to disable the Telnet process on the impacted devices (*Cisco Telnet Vulnerability*, 2020). The issue long_cpu_hog_Prod126 may be performance-related. CPU hogging refers to the case when a process is deemed to request compute resources over a specific threshold. It may be normal behaviour during a reboot of the device, or it may be indicative of a security issue, such as a worm or virus operating in the network (Cisco Systems, 2016).

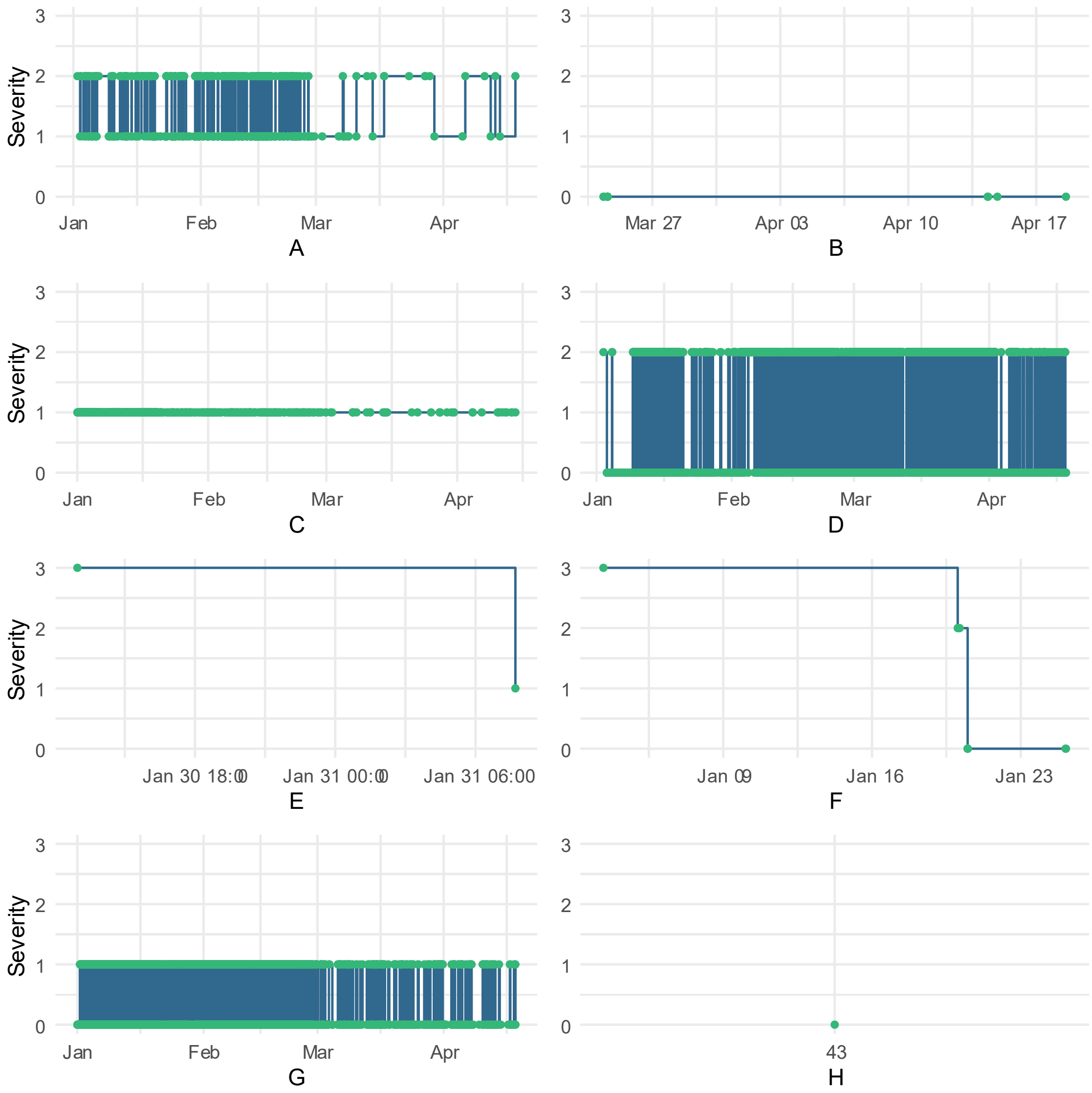
Figure 4 shows events associated with eight example issues over time. The ‘dot’ indicates when that event occurred. The issues were chosen to exemplify how severity could change (or not) over time. In an ideal world one might expect the issue to occur with a high severity; then at some point come to be resolved and return to a low severity – similar to

Figure 4E.

However, the event issue updates are under the control of the customer – it is dependent on how they have configured event updates. As mentioned previously, devices can be polled for events automatically or manually. Devices polled automatically can be scheduled daily or weekly, and at a specific time of day (Cisco Systems, 2020, p. 18). The event date/time is dictated by when the polling occurred. Thus, event data spikes shown in
Figure 2 in April may be due to customer increased polling activity and not directly due to network device activity. In addition, when comparing events from different customers, care must be taken as different customers may have adopted different polling schedules, and the date/times associated with different customer's events may not be synchronised with one another.

Figure 4

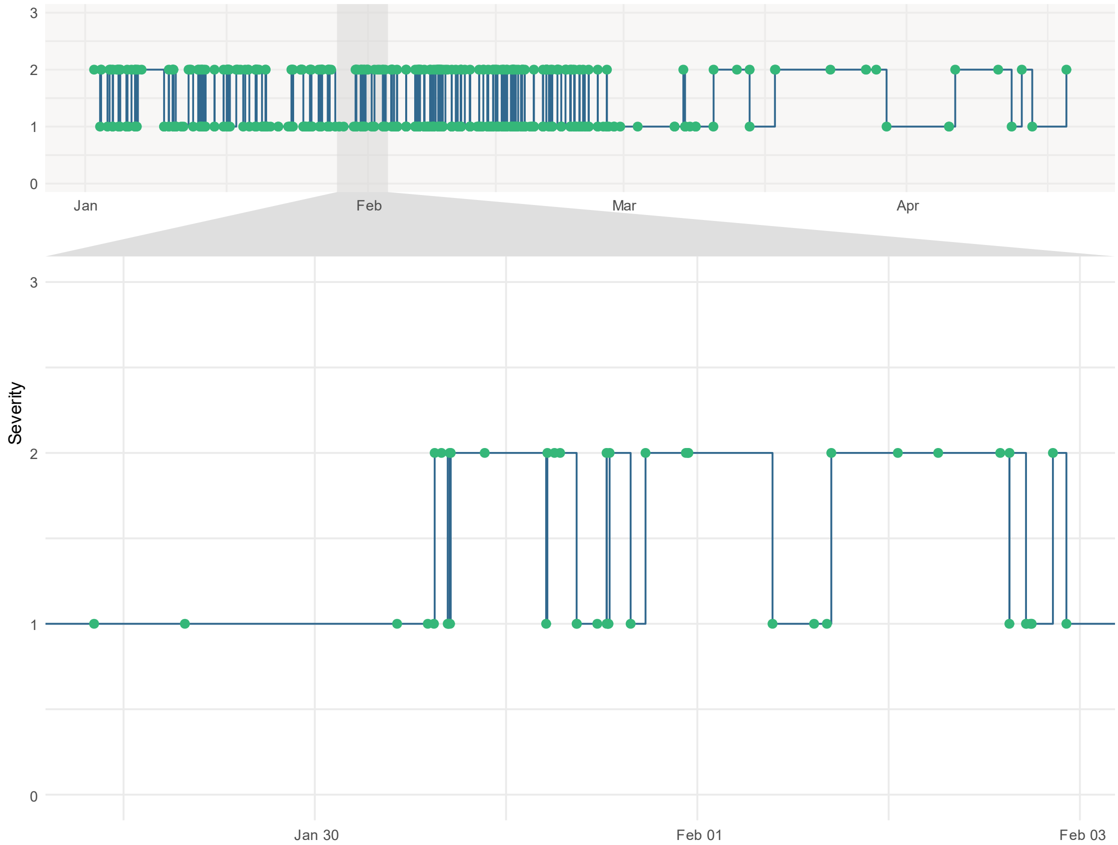
Events Over Time for the Same Issue



Severity: 0= OK, 1=Info, 2= Warning, 3-Danger

Figure 5

Detailed Timeline of Late Jan to Early Feb



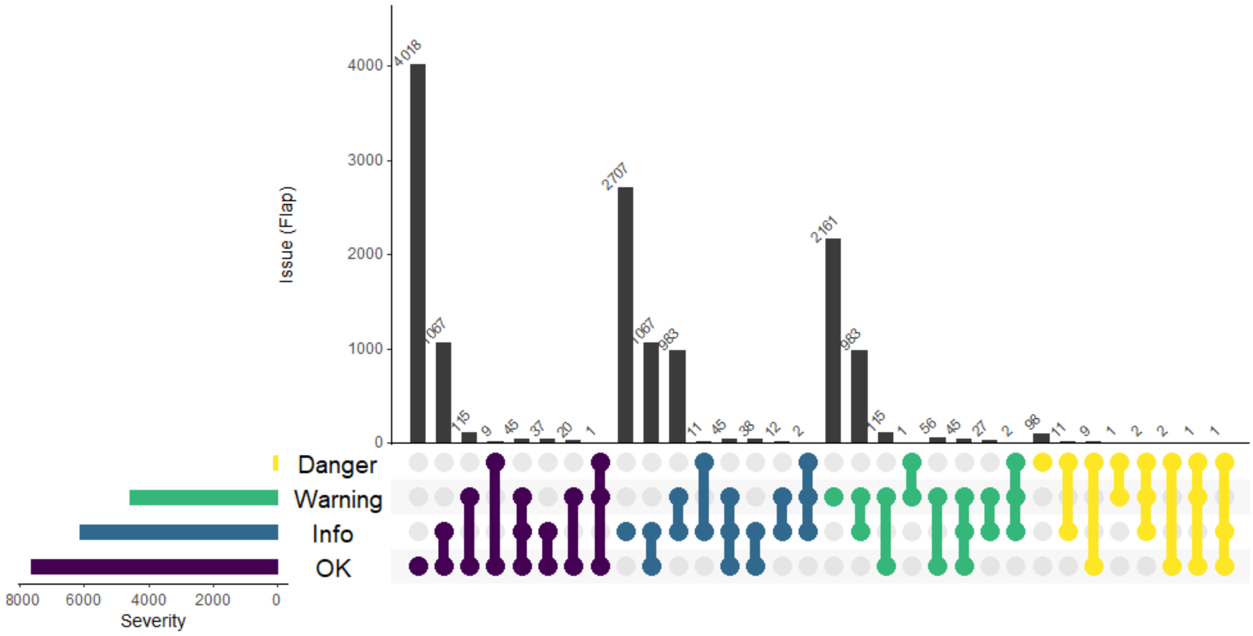
Severity: 0= OK, 1=Info, 2= Warning, 3-Danger

An expanded version of four days of the event data points in Figure 4A highlights that there are multiple events in the same time series with the same severity. For example, starting on Jan 29th, there are four events before the severity of that issue changes to 'Warning' late in the evening on Jan 30th. Those four events are redundant information. As no information on the polling schedule is provided, the implication is that only events where severity changes are significant. In order to reduce this redundancy, an algorithm was developed that would only select the **next** event if the severity had changed from the previous event referencing that issue. It has been labelled a flap, in deference to the concept of event flapping (IBM, 2014).

Figure 6 shows an Upset diagram depicting those flap events and their severities. Upset diagrams are a useful replacement for Venn diagrams when there are more than three sets (Conway & Gehlenborg, 2019). The diagram shows the count for each event and corresponding severity for that flap. In other words, the diagram shows the intersection of severities for each issue.

Figure 6

Flap Counts for Severity Sets



This diagram shows the intersection of all severities for each issue. For example, there are 45 issues that have OK, Info and Warning severities, and there are 96 issues that only have a danger severity.

One interesting observation is that most issues have one severity. For example, the 98 Danger severities (shown in yellow) have no prior lower severity notification before that event occurs – the event just happens!

Fault Model and Taxonomy

In order to help determine the methods available to analyse the dataset, it is advantageous to align the concept of issues in the dataset within the context of the fault models used in the field of IT operations analytics. This will enable the prediction methods that have been earlier developed by others to be leveraged in this investigation.

The International Electrotechnical Commission (IEC) defines:

- Failure as the loss of ability [of a service] to perform as required.
- Fault as the inability to perform as required, due to an internal state, and
- Error as the discrepancy between a computed, observed, or measured value or condition, and the true, specified or theoretically correct value or condition.

(IEC, 2015)

The relationship between faults, errors and failures are often complex and dynamic (Kochs, 2018, p. 10). For example, the result of an error by a programmer leading to a system with a memory leak in its software. However, if this part of the software is never run, the fault remains inactive. But, once the piece of code is run, the software enters an error state - memory is consumed but is not released when it is not needed anymore. This may be repeated multiple times, and at some point, there might not be enough memory for some memory allocation to occur, and the error is detected by the system. Nevertheless, if it is a fault-tolerant system, the failed memory allocation still might not necessarily lead to a service failure – there may be a backup system. Only if the system, as observed externally, cannot provide its service acceptably, does failure occur (Salfner et al., 2010, p. 10:6). The relationships between faults, errors and failures are not explicitly defined in the dataset. Thus, analysis of the dataset may never result in the ability to predict system failure without additional meta-information such as device attributes and other configuration and service information. However, the Cisco dataset notion of issue does seem to encompass the IEC concept of error and in some cases an issue may also map to the notion of a fault, for example Prod356_crash_Defected. The dataset may have the richness to predict errors and faults, but it is not adequate to predict service failures.

Avizienis et al. (2004) provided a taxonomy of fault classifications. In terms of lifecycle, they define that faults can be caused either during the development stage of the system lifecycle or the operational stage of the system lifecycle. Additionally, in the context of the system boundary (for example, a network device like a router or switch), they define that faults can either originate internally to the system or external to the system. (Avizienis et al., 2004, p. 16). For example, an internal development fault might be the memory leak bug described previously, an internal operational fault might be a configuration error performed by an administrator, and an external operational fault might be reduced data throughput performance due to the switch being overworked. These three classes of faults are all apparent in the dataset. However, it is hypothesised that external faults will be more straightforward to predict than internal faults as the errors that cause external faults may already be captured in the dataset. It is unlikely that internal errors caused during the product development process, say a bug, would be detectable in the dataset.

> “The key notion of failure prediction based on monitoring data is that errors like memory leaks can be grasped by their side effects on the system such as exceptional memory usage, CPU load, disk I/O, or unusual function calls in the system. These side effects are called symptoms”. (Salfner et al., 2010, p. 10:14)

A model has been presented which describes the progression of errors, to faults and then to failures. In addition, a fault classification taxonomy (internal versus external and development versus operational) was described. It was also highlighted how the model and taxonomy mapped to the dataset. Finally, it was hypothesised that it might be easier to predict external faults as the error conditions that cause them would be more visible in the dataset.

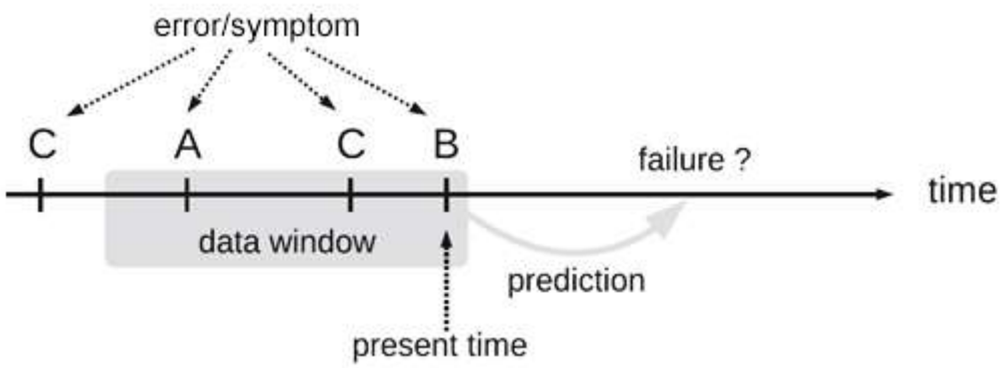
Problem Definition

This next section will discuss the methods and techniques that have been adopted by others with similar datasets. It will then describe the specific problem to be addressed and the goals of that investigation.

In "A Survey of Online Failure Prediction Methods", Salfner et al. (2010) describe several different techniques that may be used to predict failures. The two closest methods that mapped to the event-based dataset were failure prediction models based on error reporting data and on symptom monitoring data (p. 10:16).

Figure 7

Time-Series Failure Prediction



A failure prediction model based on the prior occurrence of errors or symptoms A, B, and C. Adapted from "A Survey of Online Failure Prediction Methods" by Salfner et al., 2010, ACM Computing Surveys, Volume 42:3, page 10:16.

Figure 7 illustrates a failure prediction model whose goal is to determine the probability of failure at some point in the future. The prediction is performed by using some set of data (known as a data window) that has occurred before the present time. These predictions could either be generated from error logs or continuous symptom monitoring (like memory usage and CPU load). Due to the similarities of both sets of time-series data, there is a considerable overlap in analysis techniques performed on both types data – the main difference seemed to be how and when those data were extracted from the system.

In “A Survey of Predictive Maintenance: Systems, Purposes and Approaches”, Ran et al. (2019) proposed a classification of three methods for fault diagnosis and prognosis namely, knowledge-based, traditional machine learning, and deep learning (p. 4). They define the knowledge-based method as employing a priori expert knowledge and deductive reasoning to generate a prediction. In fact, Cisco Connected TAC is one such system, leveraging previously acquired technical support knowledge and experience to attempt to proactively identify security, configuration, software and hardware issues (Allen & Goloubew, 2020, p. 4). Traditional machine learning examples include Logistic Regression, Decision Trees and Support Vector Machines. Deep learning methods described include Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) and General Adversarial Networks (GANs) (Ran et al., 2019, p. 4).

Combining two traditional machine learning techniques, Ganguly et al. (2016) used SVMs and Logistic Regression to build a model to predict hard disk failures in a cloud environment. Similarly, the Ensemble methods AdaBoost and XGBoost have also been used to predict hard disk failures (Huang, 2017). At the same time, deep learning methods like Long Short Term Memory (LSTM) networks have been increasingly employed in time-series failure detection. For example, in an experiment comparing fault detection with four different sets of time-series data LSTM networks were found to give better results than comparable RNNs (Malhotra et al., 2015, p. 94). Also, a deep learning approach was used to predict failure in a computer system using LSTM networks using a sliding data window to fetch 50 data points (such as memory usage, CPU load and disk information) in order to determine the failure state at the 51st data point (Dutta, 2019).

The core hypothesis is that there is a dependency between some specific error or symptom events that will eventually lead to fault events. As no meta information was provided about the relationship between devices, it seemed logical to initially investigate a specific issue experienced on a set of devices. In other words, only the events on the devices that experienced the issue would be included in the dataset.

The basic process was to:

- Choose an externally generated fault.
- Select all events on all the devices that experience that fault.
- Perform a feature extraction process on those events.
- Train and evaluate the models on the extracted features.

Figure 3 shows the top 20 issues experienced. It seemed reasonable to pick one of those issues as there would hopefully be enough data to conduct an analysis. Of those top 20 issues, only long_cpu_hog_Prod126 seemed to be one fault that might be caused by an external factor. Thus, long_cpu_hog_Prod126 was chosen for the initial investigation – the issue of interest.

Therefore, the initial experiment will focus on developing models to predict if a Warning or Danger severity event will occur on a specific device. Based on a data window of previous events, the model will attempt to predict if the next event is the issue of interest. After choosing the best model, it will be further trained in a second experiment to see if it can predict that issue occurring on a device in another customer’s environment. In order words, can network device issue data sourced from one set of customers be used to predict that same issue at another customer?

However, before starting the model development, it needs to be determined how those models will be evaluated. The next section is a description and evaluation of some binary classification model evaluation metrics.

Evaluation Metrics

Deciding on an appropriate metric is an important but difficult part of a machine learning project. Even for something as seemingly simple as a binary classification metric, there are many different ones, and each has different characteristics and are suitable for different purposes (Czakon, 2020). The following metrics are discussed in the context of this investigation:

Confusion Matrix

The performance of a classification model can be shown in a table, called a confusion matrix, describing observed (i.e. what is true) and predicted classes for the data (Kuhn & Johnson, 2013, p. 254), as shown below.

Table 3

Binary Confusion Matrix

Observed	Predicted	
	Non-event	Event
Non-event	TN	FP
Event	FN	TP

TN – True Negative, FP – False Positive,
FN – False Negative, TP – True Positive

Compared to accuracy, confusion matrices are a much better way to evaluate the performance of a classification model (Géron, 2019, p. 90). For example, the model shown in Figure 11 on page 33 has an accuracy of nearly 95%; however, the model did not successfully predict any of the failure events correctly.

Receiver Operating Characteristics (ROC) curve and AUC

The True Positive rate (or sensitivity) is defined as the fraction of correctly predicted events over all the observed events.

$$tp - rate = \frac{TP}{TP + FN}$$

The False Positive rate is defined as the fraction of incorrectly predicted events over all the observed non-events.

$$fp - rate = \frac{FP}{FP + TN}$$

A ROC graph plot tp-rate on the Y-axis and fp-rate is plotted on the X-axis (Jin Huang & Ling, 2005, p. 300). A ROC curve portrays trade-offs between benefits (TPs) and costs (FPs). Although the ROC curve is a two-dimensional representation of classifier performance, a method to express classifier performance as a single value is to calculate the area under the ROC curve - AUC (Fawcett, 2006). For class-balanced problems, where both classes are distributed evenly, accuracy and AUC are suitable metrics. For class-imbalanced problems, where the total number of one class is much greater than the other, precision and recall are better choices (Chollet & Allaire, 2018, p. 103).

Precision-Recall curve and F1-score

In many ways, credit card fraud detection is a similar problem to network device fault prediction. For example, credit card purchases classification models may prioritise the detection of fraudulent transactions. However, it is also important not to cry wolf, to reduce the number of times the customer is contacted about transactions that are flagged as unusual but turn out not to be fraudulent.

Precision is defined as the fraction of correctly predicted events over all the predicted events.

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

Recall (also known as sensitivity) is defined as the fraction of the correctly predicted events over all the observed events.

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

In the credit card fraud detection example, optimising for recall helps with minimising the chance of not detecting fraud. But this comes at the cost of predicting fraud in normal transactions - increasing FPs. On the other hand, optimising for precision prioritises correctly detecting fraud. But this comes at the cost of missing fraudulent transactions more frequently - increasing FNs (Raschka & Mirjalili, 2019, p. 320).

The F1-score is a measure of overall accuracy. It attempts to balance the effects of optimising for precision and recall.

$$\mathrm{F1} = 2 * \frac{precision * recall}{precision + recall}$$

The F1 score favours classification models that have similar precision and recall (Géron, 2019, p. 93). However, as discussed in the credit card example, in the case of network device fault prediction, it might be more prudent to optimise for recall. Again using the results shown in Figure 11 on page 33 as an example, the model's precision was zero (and the recall value blew up my calculator).

Matthews correlation coefficient

F1 score is one of the more popular metrics in binary classification tasks. However, these statistical measures are not optimal, especially on class-imbalanced datasets. The Matthews correlation coefficient (MCC), is a more representative metric which produces a high score only if the prediction obtained good results in all four categories of a binary confusion matrix (Chicco & Jurman, 2020). The MCC can be calculated using the confusion matrix and is defined as:

$$\mathrm{MCC} = \frac{TP * TN - FP * FN}{\sqrt{(TP + FN)(TP + FP)(TN + FP)(TN + FN)}}$$

MCC can take values between −1 and +1. 1 a perfect positive correlation, −1 a perfect negative correlation, and 0 no correlation (Shmueli, 2020).

False Negative Rate

False Negative rate (or type II error) is defined as the fraction of the incorrectly predicted non-events over all the observed events.

$$fn - rate = \frac{FN}{TP + FN}$$

In the credit card analogy, it is the fraction of missed fraudulent transactions that the classification model fails to predict (Czakon, 2020).

These metrics will be captured during the experimental phase.

Experiment 1 – Labels as Features

Data Preparation

Due to the 40GB dataset size, it was decided to use a MySQL server to manage the data. This would enable SQL queries in the initial investigation of the data and gathering a subset of the data of interests for further analysis while reducing the exposure to in-memory limitations (and workarounds) of programming languages like R.

Feature Selection

After doing a literature search, a popular focus on this type of problem – using time-series event data to predict future events was in the medical field. One well-cited paper "Using recurrent neural network models for early detection of heart failure onset" (Choi et al., 2017) seemed a great fit to the device event log data. There is potentially an interesting contrast between event health record (EHR) data and medical event prediction and network issue dataset and device failure prediction.

So, the initial goal of the project will be to leverage the methodology presented in the Choi et al. paper and apply it to the domain of network device failure prediction. The methodology captured each unique EHR in an n-dimension binary vector. The equivalent for this investigation would be to create a vector-based on each unique hit_module (issue type). The count for unique hit_modules for all the issues for devices that had experienced long_cpu_hog_Prod126 was 3618. Thus, the N dimension for the vector would be 3618.

$$\text{N unique hit\_modules} \left\{ \begin{array}{c} long\_cpu\_hog\_Prod126 \\ telnet\_input\_enabled \\ \vdots \\ hit\_module_N \end{array} \right. \qquad \overbrace{\begin{array}{cccc} [1, & 0, & \dots & 0] \\ [0 & 1, & \dots & 0] \\ & & \vdots & \\ [0, & 0, & \dots & 1] \end{array}}^{N-dimension\ binary\ vector}$$

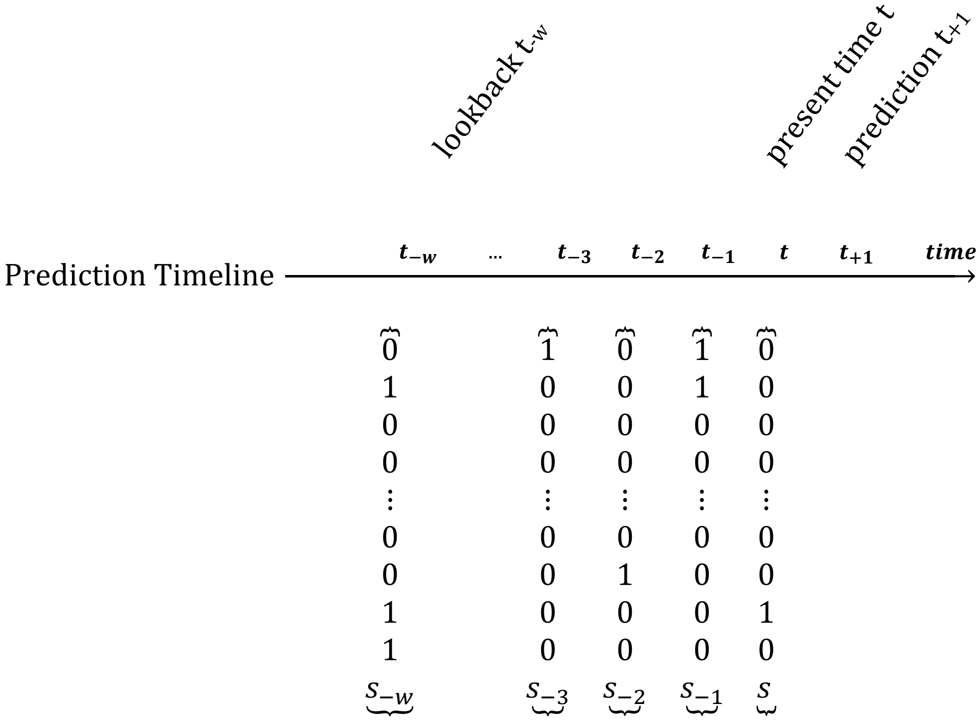
There are advantages to removing predictors or feature dimensions prior to modelling. Fewer dimensions mean reduced computational resource requirements. Second, if two predictors are highly correlated, removing one might mean a simpler, more transparent model (Kuhn & Johnson, 2013, p. 43). However, unlike the EHR data, the issue dataset had an additional attribute comprising labels, or tags, describing the issue. It would be possible to describe every hit_module with the shared tags instead of the unique hit_module ID. The count of the unique labels (as shown in Appendix B – Issue Labels) for issues for devices that had experienced long_cpu_hog_Prod126 was only 363. Thus, a combination of 363 labels now describes 3618 hit_modules and a 3618 long vector could be represented by a 363 long vector – as shown below:

$$\text{363 unique labels}\begin{cases} CPU\_1 \\ Performance \\ \vdots \\ label_{363}\end{cases} \qquad \overbrace{\begin{matrix}[1, & 0, & \dots & 0] \\ [0 & 1, & \dots & 0] \\ & & \vdots & \\ [0, & 0, & \dots & 1]\end{matrix}}^{363-dimension\ binary\ vector}$$

The data window matrix for predicting the next Warning or Danger long_cpu_hog_Prod126 was constructed by adding the individual label vectors together for each issue at time t. In addition, the severity of the issue would be appended to the end of the vector. For example, using the above definition as a reference, for an issue with CPU_1 and Performance labels at time t, the vector would be [1, 1, … 0, s_t], where s_t is the severity at time t.

Figure 8

Features for Issue/Severity Prediction



s_t = issue severities at time t.

For this first prediction model, it was decided not to use the specific time/date just the event sequence information. This would enable the application of the model to different customer datasets without worrying about different polling schedules between customers. Thus, the prediction at the present time t will be for the next issue/severity in the sequence t_{+1}, not for an issue/severity and time. The lookback defines the data window described in Figure 7– the number (w) of previous sequences to use as input variables to predict the next issue event in the sequence. Thus, this prediction model has been defined as a many to one machine learning problem – a sequence of data is delivered as input and a single result provided as output.

Several methods have been described to predict sequences within the domain of Fault Prediction. There are many binary classification algorithms (Fernández-Delgado et al., 2014). However, it was initially decided to compare a deep learning method with a traditional machine learning method. A simple Recurrent Neural Network (RNN) model was chosen to investigate the label features. Initially, all 363 label features with a lookback of 15 were used to train the RNN model. However, this resulted in a model that underfitted to the training data. By excluding labels with less than 200 hits, the list of features was reduced from 363 to approximately 70, whilst increasing the lookback. Even then, underfitting still occurred.

Table 4

Comparison of Feature Selection Methods

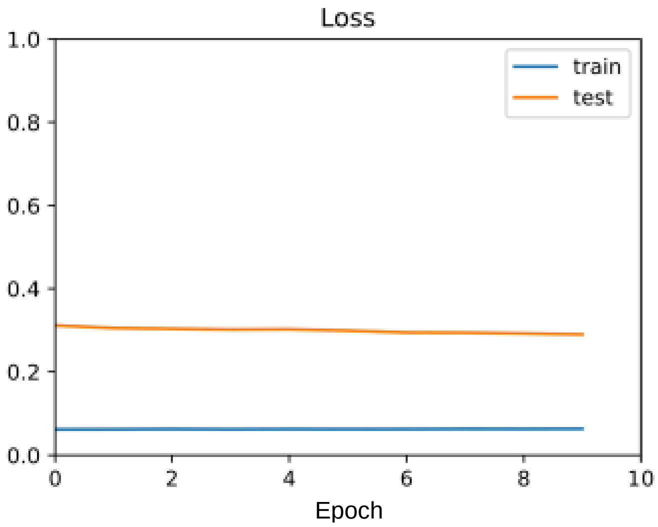
Original labels/tags long_cpu_hog_Prod126	CHI^2	Mutual Information	Ad-Hoc
Automation	CPU_1	CPU_1	CPU_1
CPU_1	Management_1	Management_1	Memory
Diagnostic_Signature	Prod6028	Prod6028	Performance
Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Appliances	Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Appliances	Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Appliances	
Software_Failure	Prod6052_FW_Appliance	Optimization_Opportunity	

Table 4 shows the original labels associated with the long_cpu_hog_Prod126 issue under investigate, the top 5 features chosen by the chi-squared and mutual information methods, and the three ad-hoc chosen features chosen by the author.

Another method had to be found to select the features used to train the model. After exploring the literature, two methods were found that were recommended for feature selection of categorical data are the chi-squared test and analysing the mutual information between the features and dependent variables (Brownlee, 2019). The loss learning curve for the original RNN model for the mutual information selected features is shown below. The curve for the chi-squared test was similar. The shape of the curve is typical of an underfit model that appears too simplistic (Brownlee, 2017).

Figure 9

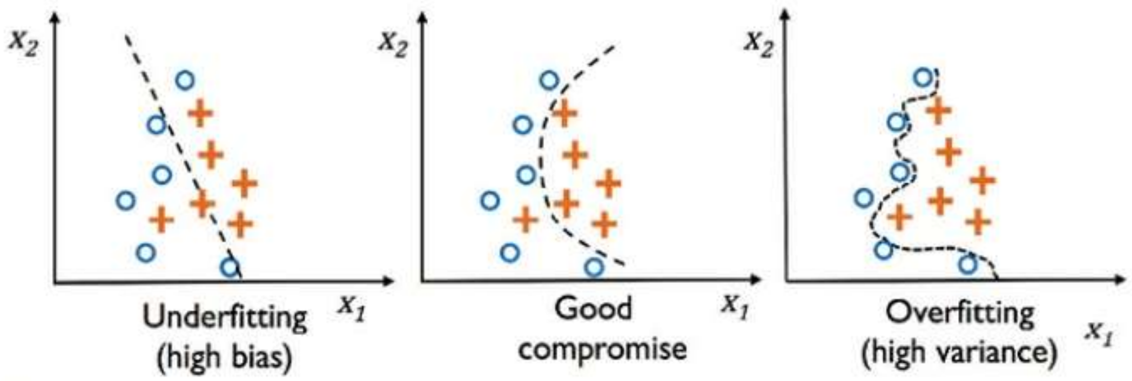
Loss Learning Curve – Mutual Information Selected Features



Conceptually, underfitting is linked with the inability of a machine-learning algorithm to capture the underlying structure of the training data. Contrary to that, overfitting is associated with a model that corresponds too closely or exactly to a particular set of data to be generalisable. Simply put, “underfitting models are sort of dumb while overfitting models tend to hallucinate” - in other words, predict things that don’t exist (Rodriguez, 2017). The problems of overfitting and underfitting can be best shown by comparing a simple model to more complex ones with the same data (Raschka & Mirjalili, 2019, p. 137), for example, a linear decision boundary model to more complex, nonlinear models, as shown below.

Figure 10

Underfitting versus Overfitting

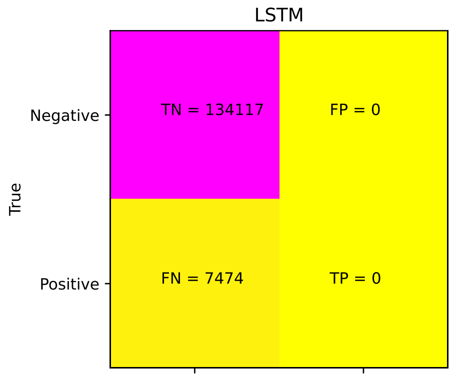


Reprinted from: Python Machine Learning: Machine Learning and Deep Learning with Python, Scikit-learn, and TensorFlow 2, 3rd Edition (p. 137), by S. Raschka, 2019, Packt Publishing.

This suggests that an underfitted model displayed in a confusion matrix would result in many false negatives and not very many true positives. The next figure shows the confusion matrix results for the model based on the mutual information selected features.

Figure 11

Confusion Matrix – Mutual Information Selected Features



TN – True Negative, FP – False Positive, FN – False Negative, TP – True Positive

As shown in Figure 10, underfitting occurs when the model is too simple to learn the underlying structure of the data. The main options for fixing the issue are:

- Define a more powerful model, with more parameters[3].
- Choose better features for the machine learning algorithm (feature engineering).
- Reduce the constraints on the model - e.g., decrease the regularisation and dropout (Géron, 2019, p. 29).

As Géron (2019) mentioned, one option to rectify underfitting is to choose better features for the learning algorithm (p. 29). In “An Introduction to Variable and Feature Selection”, the first item in the checklist for feature selection was, “do you have domain knowledge? If yes, construct a better set of ‘ad hoc’ features” (Guyon & Elisseeff, 2003, p. 1159). Therefore, it was decided to attempt to choose features that might be associated with directly device performance as those features might also be predictors of future CPU hogging and increase the lookback window to provide more data points. The issue labels CPU_1, Memory and

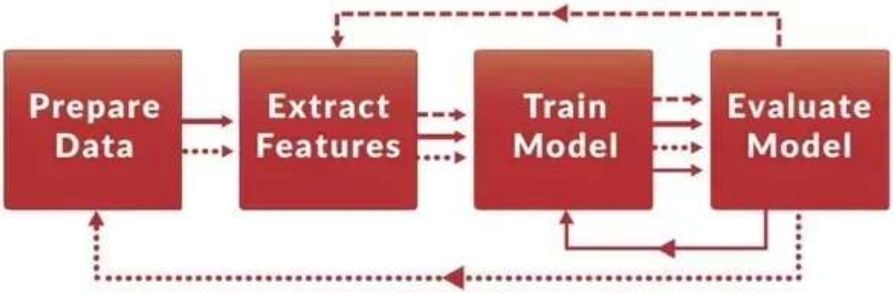
Performance (in addition to the feature Severity) seemed to map the closest to the selection criteria and were selected to train the model that was to predict when the issue long_cpu_hog_Prod126 was at Warning or Danger severities. Appendix B – Issue Labels, shows the complete list of labels. The RNN model did not seem to exhibit as much underfitting as it did with the previous attributes, so it was decided to proceed with those three features to train a deep learning model and a more traditional logistic regression model.

Technologies and Pipeline

A machine learning workflow was adopted as the one described in Figure 8. Once constructed and debugged, the data pipeline helped experimentation by supporting iterative workflows.

Figure 12

Machine Learning Workflow



Downloaded from: https://www.datasciencecentral.com/profiles/blogs/deep-learning-pictures

The data was prepared and analysed on an MSI GL65 SC[4] laptop, with an Intel Core i7, an NVIDIA GeForce GTX 1650 and 64 GB RAM. The data were prepared and initially analysed using MySQL and R, and the machine learning algorithms were investigated using Keras (Chollet & others, 2015) and Scikit-learn (Pedregosa et al., 2011) in Python.

A description of the high-level tasks and software used is shown below.

1. Prepare Data: MySQL
    a. Select an issue (hit_module) of interest.
    b. Generate a table with all issues of devices that have experienced the issue of interest.
    c. Export table to a comma-separated value text file (CSV).
2. Extract Features: RStudio
    a. Import table from CSV.
    b. Filter after first 10 million entries (memory limited larger data manipulation).
    c. Extract predictor features as separate entities.
    d. Ordinal encode categorical features.
    e. Generate feature to be predicted (y), recombine with predictors, and export to CSV.
3. Train Model: Python
    a. Normalise severities.
    b. Create the datasets based on lookback and delay (how many steps to predict into the future, which was one).
    c. Configure and train models.
    d. Save model (for potential future training/evaluation with additional data).
    e. Generate graphs and attributes to facilitate evaluating the models.

The data and the code used to perform the analysis described in this dissertation are available on GitHub at https://github.com/nilspeder/IM906.

Model Training and Analysis

The simple RNN used in the initial analysis can be good at forecasting sequences, but they do not always perform as well on longer data sequences. On the other hand, LSTM networks can be used very much like a basic RNN, and they will perform much better, training will converge faster, and they will detect long-term dependencies in the sequences (Géron, 2019 p. 511-5). After some informal experimentation with different types of RNN, multiple layers and dropout, a single layer LSTM was chosen. LSTM seemed to offer better performance than vanilla RNN or GRU approaches, and there did not seem to be any benefit to adding multiple layers to the model. There were five variables (CPU_1, Memory and

Performance, Severity and the predicted fault state) and the number of lookback steps, defining the data window (initially described in Figure 7), was 250. The delay, or how many steps in the sequence to predict into the future, was 1 step. A graphical representation of this network is shown in Figure 13.

Figure 13

Labels LSTM Network

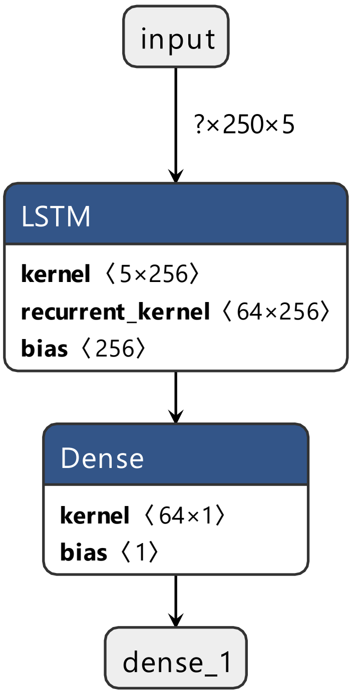
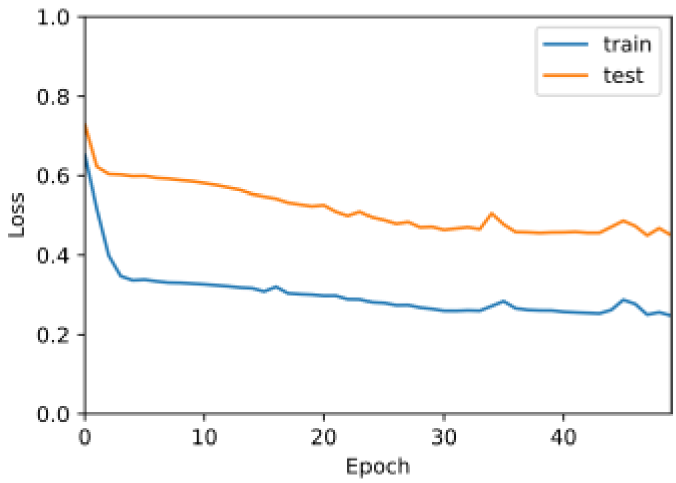


Diagram generated using: Netron (Roeder, 2010/2020).

Figure 14

Loss Learning Curve – LSTM Labels



The loss graph shown above still exhibited non-convergence between the training and testing (also known as validation) curves that is typical of an underfitting model. The model was tested again out to 500 epochs, and the pattern of loss graph remained very similar. However, some of the classification metrics seemed more promising. The Area Under the Curve (AUC) value was 0.86. The confusion matrix showed a ratio of nearly 4:1 for the proportion of true positives to false negatives, which was a lot better than the initial RNN. So, it was decided to proceed further with the investigation.

The Scikit-learn logistic regression model was used with its default settings. The AUC value as 0.81; however, the confusion matrix did not seem to be as good as the LSTM. Since the pipeline was already constructed, it would now be relatively easy to explore other methodologies. Perhaps Ensemble methods might provide a better result?

Ensemble Methods

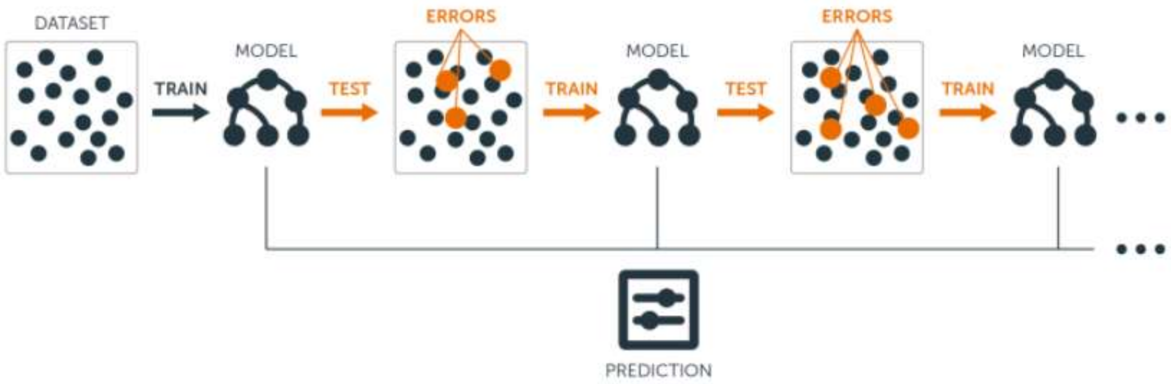
Recently many of the prize winners of Kaggle competitions[5] are using ensemble methods. Why are algorithms like AdaBoost and XGBoost the go-to models in these competitions (Albert, 2018)?

> “Suppose you pose a complex question to thousands of random people, then aggregate their answers. In many cases, you will find that this aggregated answer is better than an expert’s answer. This is called the wisdom of the crowd. Similarly, if you aggregate the predictions of a group of predictors (such as classifiers or regressors), you will often get better predictions than with the best individual predictor. A group of predictors is called an ensemble; thus, this technique is called Ensemble Learning, and an Ensemble Learning algorithm is called an Ensemble method.”(Géron, 2019, p. 189)

Therefore, the goal of an Ensemble algorithm is to combine several weak learners into a stronger one. Boosting algorithms, as opposed to other Ensemble methods, attempt to evaluate predictors sequentially, where each subsequent iteration attempts to fix the errors of its predecessor (Singh, 2018).

Figure 15

Boosting Algorithms Methodology



Downloaded from: https://blog.bigml.com/2017/03/14/introduction-to-boosted-trees/

AdaBoost (Freund & Schapire, 1997) and XGBoost (Chen & Guestrin, 2016) both work by sequentially adding predictors to an ensemble, each one improving its predecessor. However, they use different statistical methods to achieve this goal. AdaBoost alters the weights of the predictor variables while XGBoost, similar to regression analysis, tries to fit the new predictor to the residual errors made by the prior predictor (Géron, 2019, p. 203).

Therefore, it was decided to employ the Scikit-learn AdaBoost and XGBoost in the experiment as well. The starting point for initial model configurations was obtained from such sources as towardsdatascience.com (Maklin, 2019) and machinelearningmastery.com (Brownlee, 2016).

Results

Viewing the ROC Curve and PR Curve (Figure 16 and Figure 17) it seemed that the LSTM network and XGBoost were most effective. However, the Confusion Matrix (Figure 18) shows that the False Negative rate for the LSTM model was better than XGBoost model. This would imply that the LSTM model would be less likely to predict no fault when a fault was about to occur.

Figure 16

Experiment 1: ROC Curve

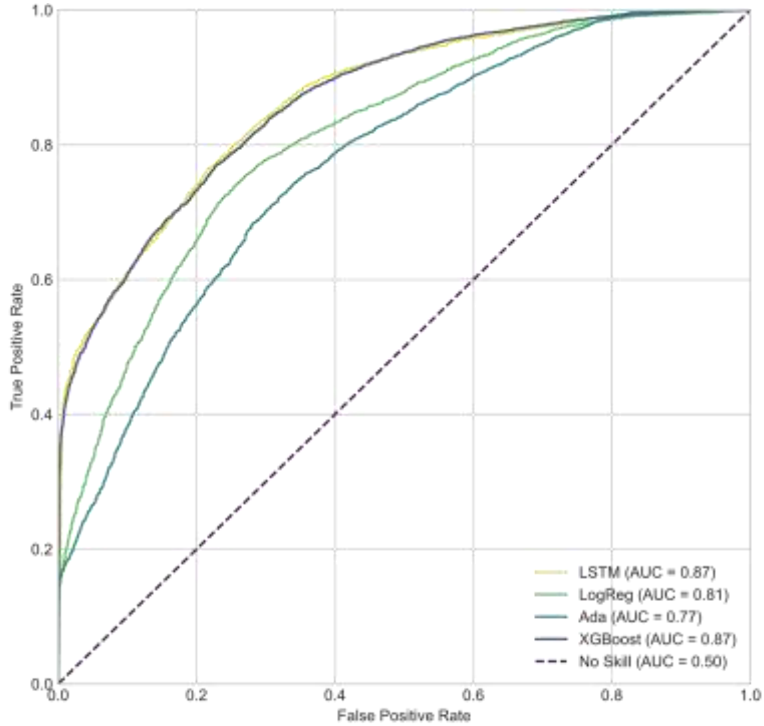


Figure 17

Experiment 1: PR Curve

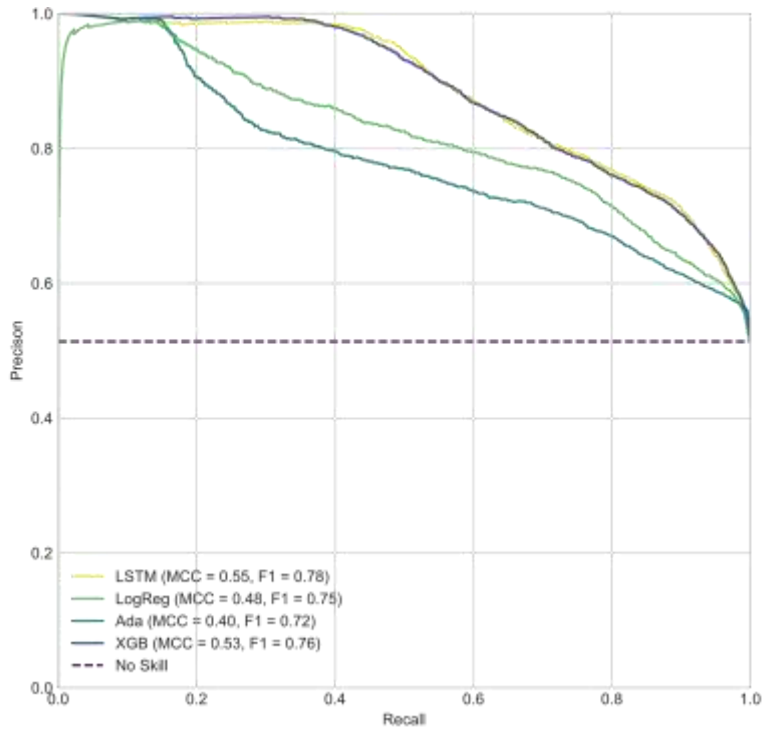
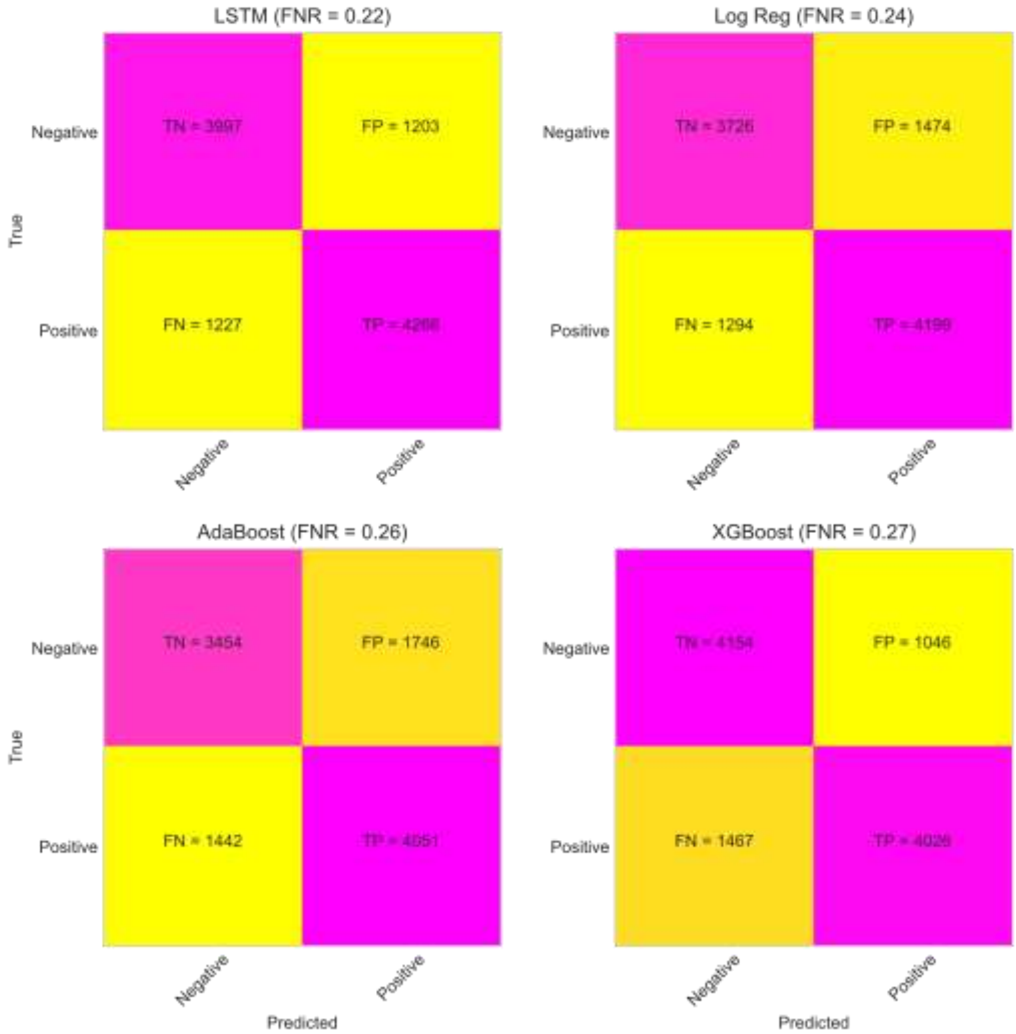


Figure 18

Experiment 1: Confusion Matrix



The next stage of the experiment was to test the model on unseen data. Instead of, ignoring customer_id and collecting the data as one continuous sequence and then sorting on device_id (thus obtaining all sequence of all events for each device that had experienced the issue), the data were first divided into groups of customers. Then the sort on device_id was performed. The LSTM was trained on one set of customer data and then was tested on the other set of customer data. The results were equivalent to the Confusion Matrix shown in Figure 11 on page 33. The model was unable to predict faults in the other customer's data. The model could only predict True Negatives and False Negatives. Combinations of different groups of customers were chosen, those that had many events versus not so many, but the result was always the same. It seemed that the model was too simple to predict what was being asked of it. That result was consistent with the underfitting uncovered in the Loss Learning curve in Figure 14 on page 36.

Most of the Machine Learning literature seems to focus on overfitting, not underfitting. How could the model's complexity be increased? Labels were chosen explicitly to reduce the complexity of having to manage every issue type. However, the number of labels seemed to have been reduced (CPU_1, Memory and Performance) to produce an overly simplified model.

Experiment 2 – Issues as Features

It was hypothesised that one way to increase the number of predictors would be to use the label terms (cpu and memory) as search strings to gather corresponding issue types (hit_module name). When a search for *cpu* and *memory* was performed on the long_cpu_hog_Prod126 dataset, 94 issue types were returned that contained those strings in their hit_module ID (shown in Appendix C – Issue Module IDs). The table below shows the top 10.

Table 5

Top 10 CPU/Memory Issues

Issue Type	Devices	Events
long_cpu_hog_Prod126	9259	220964
asa_high_cpu_Prod126	7474	153901
Low_Free_Memory_Available	6861	148278
memory_used_greater_than_100percent_Prod334	6837	148259
snmp_cpu_hog_Prod126	6676	150181
extremely_long_cpu_hog	6282	144878
CPU_hogs_due_to_SNMP_polling	4724	33040
Defect2938_Defection_in_memory	3509	26063
low_memory_Prod126	479	4314
Prod128_compliancy_checks_CPU_memory_failure_Prod126	187	3172

The data preparation algorithm selected all issues with Memory and CPU in their module_id name and then filtered out those issues with less than 500 events. The resultant issue types were used as predictors for training with the same data as the first experiment. The dependent variable was created in the same manner as in the first experiment. After a few iterations attempting to minimise underfitting, the LSTM network shown in Figure 19 was developed. It had 11 predictor variables, and the lookback was 100. As before, there was no dropout or regularisation.

Figure 19

Issues LSTM Network

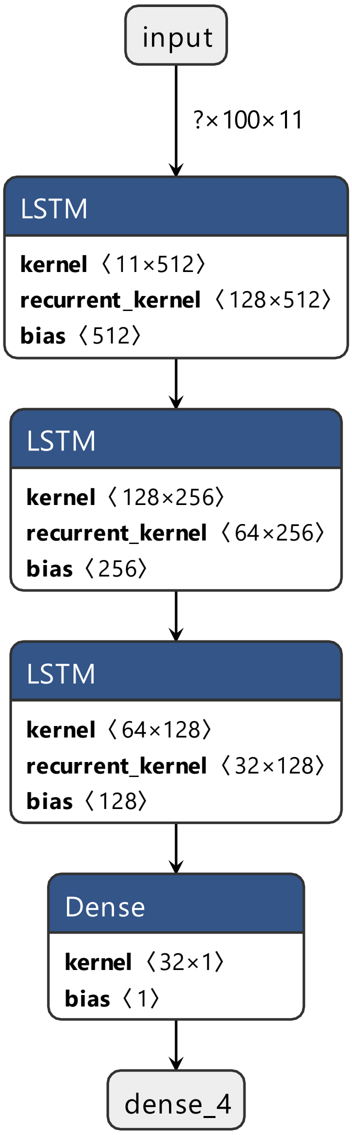


Diagram generated using: Netron (Roeder, 2010/2020).

The Loss Learning Curve is shown in Figure 20. It shows the test model reducing its losses over the training run. This improvement levelled off around epoch 50. There seemed to be no indication of overfitting.

Figure 20

Loss Learning Curve – LSTM Issues

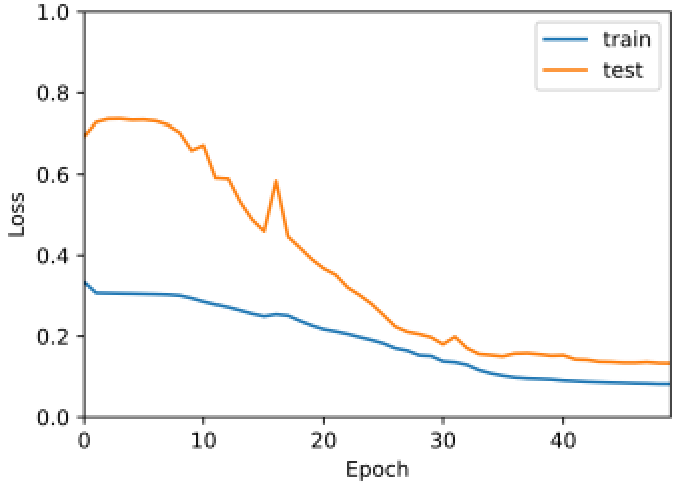


Figure 21

Experiment 2: ROC Curve

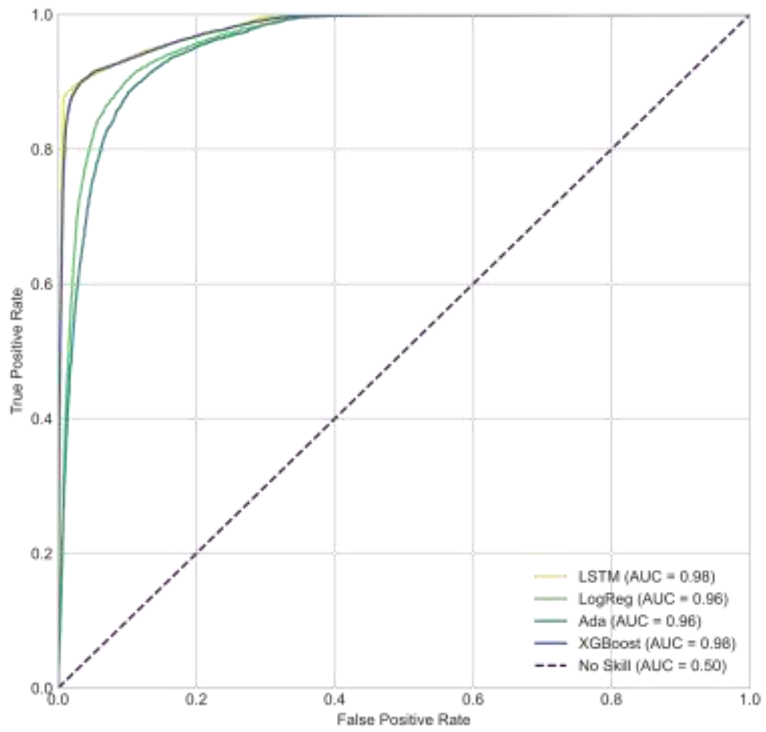
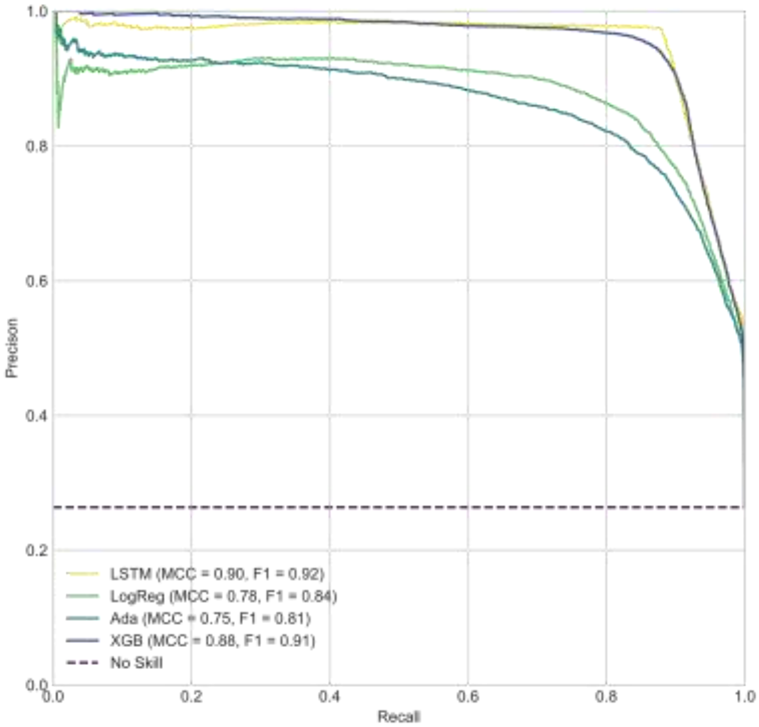


Figure 22

Experiment 2: PR Curve



The LSTM RNN and XGBoost seemed to be the most effective models in Experiment 2, even though minimal optimisation of the XGBoost hyperparameters occurred.

Table 6

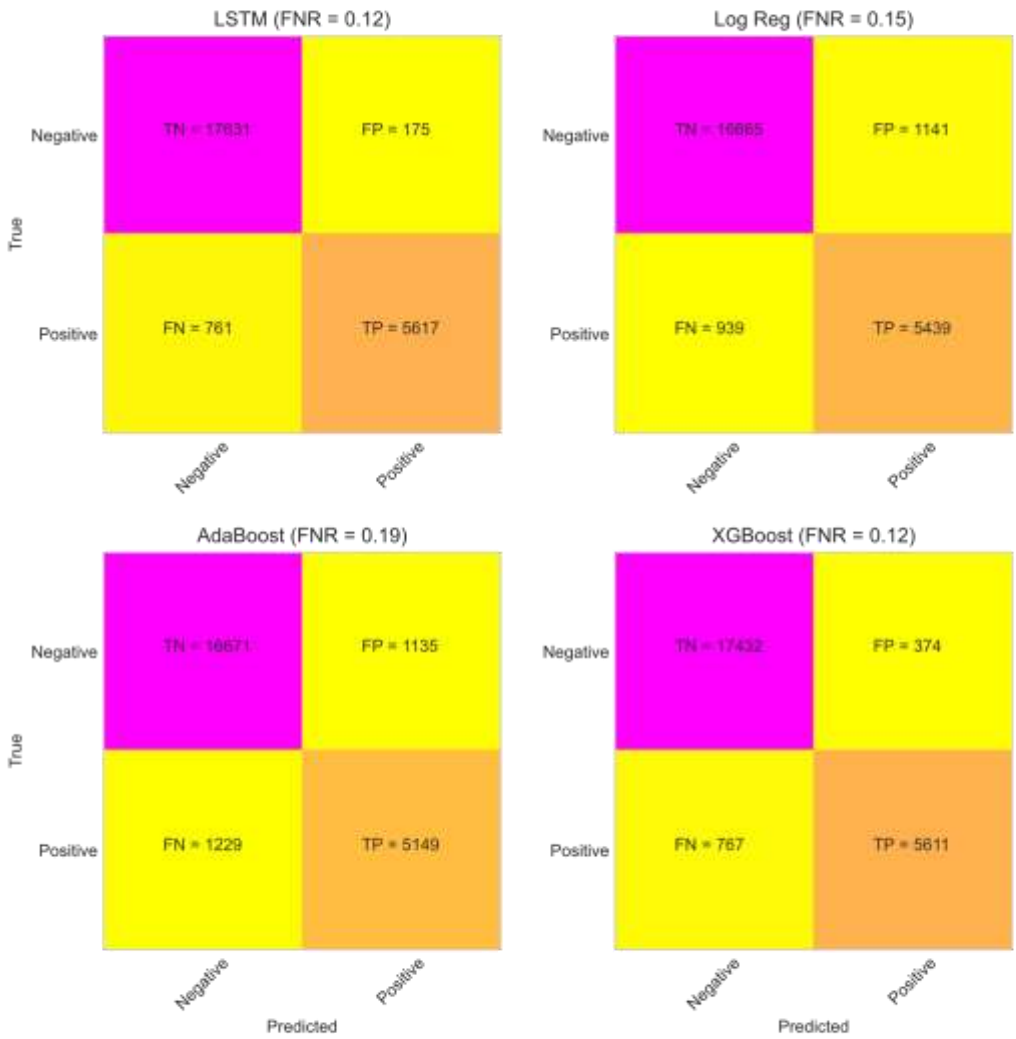
Metrics Comparison Between Experiments

	Experiment 1 – Labels		Experiment 2 – Modules	
	LSTM	XGBoost	LSTM	XGBoost
AUC	0.87	0.87	**0.98**	**0.98**
MCC	0.55	0.53	**0.90**	0.88
F1	0.78	0.76	**0.92**	0.91
FNR	0.22	0.27	**0.12**	**0.12**

The metrics for the prediction models using the module based features showed an improvement over the label based features used in Experiment 1. The False Negative rate for LSTM and XGBoost improved from 22% and 27% to 12%, respectively. Encouraged by the improvement in the prediction model based features derived from issues, it was decided attempt to retrain a model on customer-specific data, so that it could be tested on unseen data from another customer.

Figure 23

Experiment 2: Confusion Matrix



Experiment 3 – Unseen Data

The long_cpu_hog_Prod126 dataset was split in to two - between the top 10 customers by event count, as shown in the table below.

Table 7

Train/Test and Unseen Data Split

Customer ID	Event Count	Dataset
11	11071405	Train
17	7607379	Test
1	3763166	Test
4	1645975	Train
7	1604525	Test
21	1477668	Test
15	600788	Train
10	434417	Train
14	376397	Train
12	308809	Test

The same function was used to generate the training and testing data for both datasets . But, when the data frames were returned from the ordinal encoding process, there was a mismatch between the two data frames. In other words, one dataset had dissimilar issue types. However, both datasets must have the same features for the machine learning algorithm to function correctly. Even though this implied possibly altering the unseen data, it was decided to modify each data frame to be an intersection of the set of features. Perhaps, more correctly, the data frames should have been a union of features with zeros padding the features in the relative complement[6]. However, for expediency's sake, it was decided to proceed with the intersection method. There were ten features, including severity and the dependent variable, which was one less than the previous experiment. The XGBoost algorithm was chosen; the lookback and other hyperparameters remained the same as for the second experiment. The model was trained on the first set of customer data; then the unseen set was used to test the model.

Figure 24

Unseen Data: PR Curve

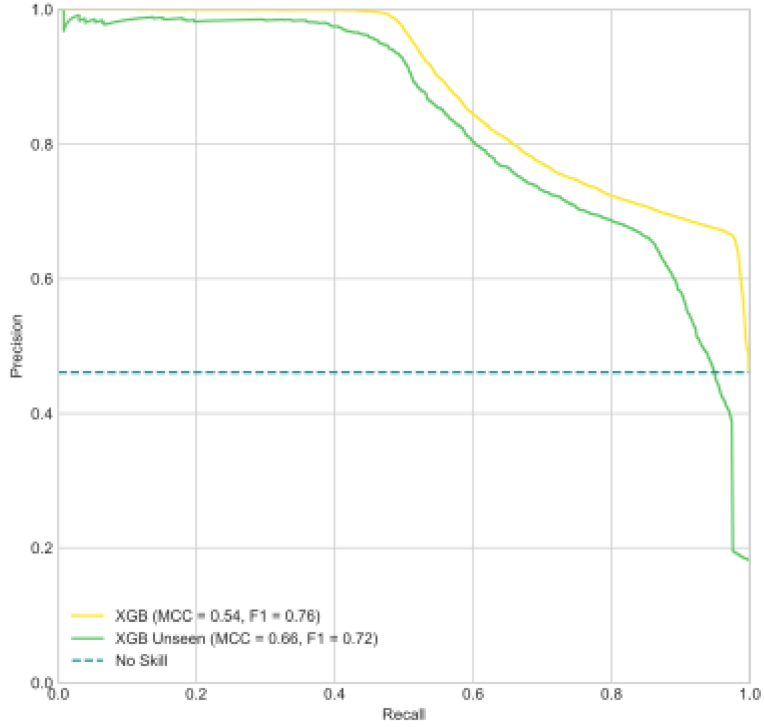
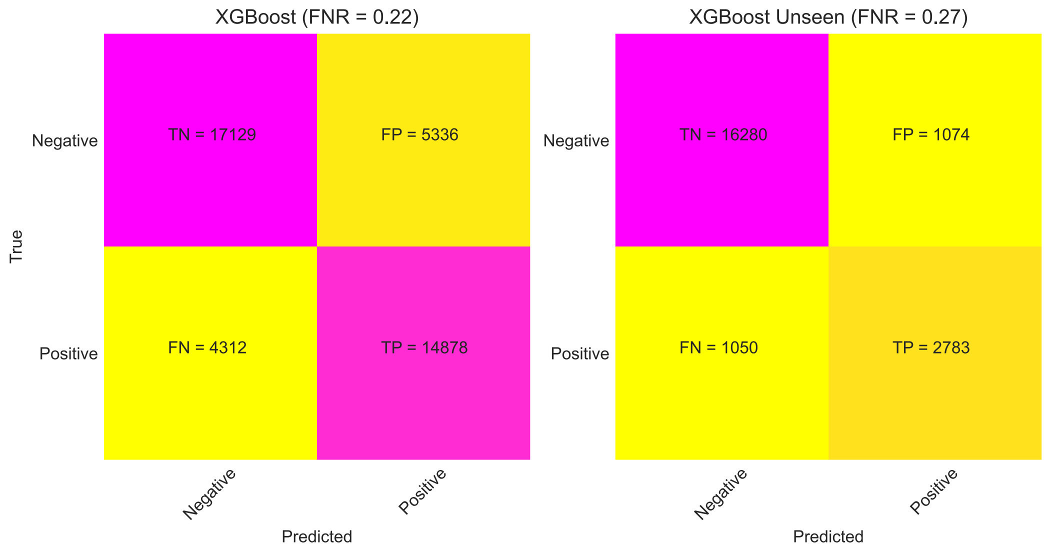


Figure 25

Unseen Data: Confusion Matrix



As the number of events and features were reduced, not surprisingly, the performance of the model was not as effective as in the second experiment. The PR curve in Figure 24 shows that unseen data did not perform as well as the seen test data. The confusion matrix showed a False Negative rate of 27%. That was a substantial improvement over a model with no skill; the model did succeed in predicting over 70% of the long_cpu_hog_Prod126 events correctly, even on the unseen test data. Thus, it was shown, with the data provided, a model developed from network events in one set of customers environments could potentially be used to predict events in another distinctly separate set of customer environments.

Discussion

This next section includes a discussion of various topics that were revealed during this investigation.

Healthcare Analogy

While conducting this investigation, it has been useful to think of analogous problem domains – not just to leverage analysis techniques but to aid in understanding the overall problem space. When initially investigating the fault prediction domain space, it was beneficial to think about the medical diagnostic field. There seemed to be many similarities between the problem of clinical event prediction and network device event prediction. However, in terms of this investigation, there is one major difference between the two, and that is in the structure and purpose of the data. Electronic Health Records (EHRs) can contain diagnoses, medications, treatment plans, immunisations, allergies, medical imaging, and laboratory and test results (*What Is an EHR?*, 2019). One of the goals of EHRs is to facilitate clinical diagnoses. The dataset used in this investigation comprised network device status events collected by the Cisco Connected TAC service. This dataset was not originally intended to facilitate network device fault diagnosis. It was co-opted for a purpose that it was not originally designed to support.

The paper by Choi et al. (2017) focused on predicting one clinical event, namely heart failure. However, the Cisco Connected TAC dataset described many different types of events. An emergency room (ER) triage admission process might perhaps be a more accurate medical analogy. For example, a patient may be admitted due to an internal clinical issue or because an external trauma had occurred. The events leading up to a clinical event, like heart failure, versus a trauma event, such as breaking a leg, would be very different. Therefore, it was understood that to keep the scope of the investigation practicable that it would be prudent to focus on one class of issue, not every possible issue. It was also hypothesised that the externally caused events would have an increased likelihood of prediction rather than events internal to the device – hence the initial focus on CPU hogging.

Improvements to the Model

The models were developed to illustrate the potential for predicting future network events if a network-device-issue data set sourced from one set of customers could be used to predict that same issue within another customer's environment. It was not necessarily a primary goal to develop the most efficient prediction model. This next section discusses ways in which the performance of the prediction models could be enhanced.

Perhaps to the detriment of developing a better model, the training process was simplified to reduce the complexity of the data manipulation. For example, the data was fed into the model as one long sequence, ordered by device and date. It might have been better to have fed the data in separate event sequences by device so that one device's event sequence would not contaminate another's. In the clinical example, it would not make sense for EHR data to be entered in one sequence and instead divided by patient. Date/time was also removed as a feature from the dataset. This reduced the utility of the model in terms of predicting when a CPU hogging event might occur – limiting it to just the prediction of the next event. An argument was presented that it was difficult to ensure that the polling schedules would be consistent between different customers. That may be true; however, there are prediction models that address irregular sampling rates (Futoma et al., 2017) and predicting when an event may occur would increase the usefulness of the predictions.

Another approach to developing a better model would be to increase the complexity by adding more predictor features to the dataset. For example, when Allen & Goloubew (2020) describe a different version of the dataset, they discuss the attribute detection_type, which described the type of issue being detected (p.

8). For example, when selecting features, it might be beneficial to know which issues were classified as availability, operational, or performance-related. In addition, being able to classify like devices together as well as knowing which issues could occur on which devices might also lead to the development of an improved model.

In order to minimise the level of imbalance between the predicted classes and simplify the analysis, the models predicted both a Warning and Danger severity events. An alternative would have been to restrict the classifier to just predict a Danger severity event. This might result in a model that was more useful to the customer as it would be predicting an event whose symptoms might be more likely to result in failure of the device. However, it would probably result in the need to employ additional analysis methods, such as to accommodate extreme class imbalance (Kuhn & Johnson, 2013, p. 419).

Finally, no rigorous effort was made to tune the boosting algorithms. Other better classification algorithms, such as random forest (Fernández-Delgado et al., 2014, p. 1) could also have been studied.

Redundant Data

When conducting the initial data investigation, there seemed to be a lot of redundant events in the dataset, as highlighted in Figure 4 on page 16. As a consequence, an algorithm was developed to remove that redundant information before generating the Upset diagram shown in Figure 6. (Allen & Goloubew, 2020) describe a similar issue of user interface noise (p. 7) in their paper, so hopefully, this issue has been addressed in the front-end of the customer-facing product.

Other Analysis Techniques

Other techniques like process and sequence mining provide tools for the analysis of event logs, resulting in the visualisation of the dependencies in the process (Reinkemeyer, 2020, pp. 1–2). These methods may shed light on the causality relationships that may exist in the event dataset and perhaps be used to generate new features for the prediction models.

Taking a completely different tack, some type of time-based customer cohort or churn analysis may also provide insight on different customers behaviour, how long they participated in the trial and in what ways they made use of the Connected TAC service.

Analysis Environment

Considerable time was dedicated to developing a technology environment that could analyse the 40GB dataset. The major limiting factor was system memory and not compute resources. Although the system was configured with 64GB RAM, workarounds had to be developed to minimise out-of-memory errors.

Conclusions

To summarise, a machine learning classifier was developed for predicting a CPU hogging issue using a network event dataset. This data was generated by the Connected TAC service provided by Cisco Systems. The classifier was trained on one set of customer data and tested on an unseen set of data from other customer's environments. Even though that dataset was not developed specifically for event prediction, the classifier was found to have some efficacy in predicting CPU hogging events.

The current classifier would need to be refined and developed further prior to production. However, if implemented in real-time, a crowdsourced prediction classifier could potentially be used to complement the existing knowledge-based Connected TAC service.

In addition, it is hypothesised that the methodology could be extended to other devices and other external performance-related issues, such as memory. However, it is unknown if it could be applied to internal issues like configuration errors. Perhaps approaches like process mining, which attempts to discover dependencies between events, might be more successful in exposing those dependencies with configuration errors.

[1] IT Operations Analytics is the process of collecting, identifying, and analysing patterns to detect problems and improve IT system performance and availability (*IT Operations Analytics - BMC Software*, 2020).

[2] Telnet is both a protocol and application which facilitates remote text-based communication between a client and server over a network (Cisco Systems, n.d.).

[3] Parameters are variables used to configure the model's algorithm. Features are attributes describing the characteristics that define the scope of the problem.

[4] MSI GL65 specifications: https://www.msi.com/Laptop/GL65-9SX-GTX/Specification

[5] Kaggle is an online community known for its machine learning competitions: https://www.kaggle.com/competitions

[6] The relative complement of set A with respect to a set B, is the set of elements in B but not in A.

This page intentionally left blank.

Appendix A – First 100

hit_date	device _id	gate way	hit _id	hit_labels	hit_module	hit _severity	cu_id
1/1/2017:12:13:08:AM	1	1	1	:Device_Hardening:Automation:Diagnostic_Signature:Prod6060	Recommended_router_best_practic	1	1
1/1/2017:12:13:08:AM	1	1	2	:Device_Hardening:Automation:Diagnostic_Signature:Prod6060	Router_hardening_unused_services	0	1
1/1/2017:12:13:08:AM	1	1	3	:Device_Hardening:Automation:Diagnostic_Signature:Prod6060	Recommended_security_best_practi	1	1
1/1/2017:12:13:08:AM	1	1	4	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap evice_Hardening:Diagnostic_Signature:Prod6060:Parser:Management_1	telnet_input_enabled	0	1
1/1/2017:12:13:08:AM	1	1	2	:Device_Hardening:Automation:Diagnostic_Signature:Prod6060	Router_hardening_unused_services	0	1
1/1/2017:12:13:08:AM	1	1	2	:Device_Hardening:Automation:Diagnostic_Signature:Prod6060	Router_hardening_unused_services	0	1
1/1/2017:12:13:08:AM	1	1	2	:Device_Hardening:Automation:Diagnostic_Signature:Prod6060	Router_hardening_unused_services	0	1
1/1/2017:12:13:08:AM	1	1	5	:VPN:Automation	Prod357_Weak_Encryption_Algorith	1	1
1/1/2017:12:30:54:AM	1	1	6	:Device_Hardening:Automation	Prod357_unencrypted_password_fo	1	1
1/1/2017:12:30:54:AM	1	1	7	:Device_Hardening:Automation:Diagnostic_Signature:Prod6060	Prod356_type_4_password_used	1	1
1/1/2017:12:30:54:AM	1	1	8	:Device_Hardening:Automation	Prod357_Enable_Password	1	1
1/1/2017:12:30:54:AM	1	1	4	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap evice_Hardening:Diagnostic_Signature:Prod6060:Parser:Management_1	telnet_input_enabled	0	1
1/1/2017:12:43:18:AM	2	2	9	:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Appliances:Dia gnature:Management_1	configuration_locked_in_another_se	0	2
1/1/2017:12:43:18:AM	2	2	10	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap iagnostic_Signature:Traffic	significant_TCP_embryonic_connect ›ing	0	2
1/1/2017:12:43:18:AM	2	2	11	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap iagnostic_Signature:Management_1:Parser:Software	asa_non_release_signed_image	0	2
1/1/2017:12:43:18:AM	2	2	12	:ACL:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security ›s:Diagnostic_Signature:Optimization_Opportunity	wide_open_acl	0	2
1/1/2017:12:43:18:AM	2	2	13	:Automation:Prod6052_FW_Appliance:Prod6044_Prod6027_Prod6009_Seri /e_Security_Appliances:Diagnostic_Signature	ip_audit_Prod126	0	2
1/1/2017:12:43:18:AM	2	2	14	:Application_Inspection:Automation:Prod6044_Prod6027_Prod6009_Series_ Security_Appliances:Diagnostic_Signature:Parser:Performance	Prod122_CX_IPS_performance_fail em_Defect2041_Prod126	0	2

hit_date	device _id	gate way	hit _id	hit_labels	hit_module	hit _severity	cu_id
1/1/2017:12:43:18:AM	2	2	15	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap evice_Hardening:Diagnostic_Signature:Optimization_Opportunity	recommended_asa_security_best_p	1	2
1/1/2017:12:43:18:AM	2	2	16	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap iagnostic_Signature:Prod6059_1:Failover:Parser:troubleshooting	failover_int_checks_Prod126	0	2
1/1/2017:12:43:18:AM	2	2	17	:ACL:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security es:Diagnostic_Signature:Prod6059_1:Optimization_Opportunity	acl_element_count_Prod126	1	2
1/1/2017:12:43:18:AM	2	2	18	:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Appliances:Dia gnature:Automation:Optimization_Opportunity	unused_config_module	1	2
1/1/2017:12:43:18:AM	2	2	19	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap onfiguration:Diagnostic_Signature:Prod6059_1:MPF:Optimization_Opportuni	infinite_conn_timeout_Prod126	0	2
1/1/2017:12:43:18:AM	2	2	20	:Automation:CPU_1:Prod6044_Prod6027_Prod6009_Series_Adaptive_Secu nces:Diagnostic_Signature:Software_Failure	snmp_cpu_hog_Prod126	0	2
1/1/2017:12:43:18:AM	2	2	21	:Management_1:Prod6023:Software_Failure:Automation	Defect4907_Prod819_not_displayin Prod80_clients	1	2
1/1/2017:12:43:18:AM	2	2	22	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap onfiguration:Diagnostic_Signature:Optimization_Opportunity:Routing	route_check_Prod126	0	2
1/1/2017:12:43:18:AM	2	2	23	:Security:Optimization_Opportunity:Automation	console_timeout_of_0	1	2
1/1/2017:12:43:18:AM	2	2	24	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap onfiguration:Diagnostic_Signature:Hardware_Limitation:Prod6060:Interface: on_Opportunity	show_interface_output_checks_bv3	0	2
1/1/2017:12:43:18:AM	2	2	25	:Management_1:Optimization_Opportunity:Automation	Prod121_NTP_authentication_not_e	1	2
1/1/2017:12:43:18:AM	2	2	26	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap iagnostic_Signature:Memory	low_memory_Prod126	0	2
1/1/2017:12:43:18:AM	2	2	27	:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Appliances:Bot _Filter:Automation:Diagnostic_Signature	botnet_updater_fails_ssl	0	2
1/1/2017:12:43:18:AM	2	2	28	:Automation:CPU_1:Prod6044_Prod6027_Prod6009_Series_Adaptive_Secu nces:Diagnostic_Signature:Optimization_Opportunity:Sham_Link	asa_high_cpu_Prod126	0	2
1/1/2017:12:43:18:AM	2	2	29	:Automation:Prod6052_FW_Appliance:Prod6044_Prod6027_Prod6009_Seri ve_Security_Appliances:Configuration:Diagnosis:Diagnostic_Signature:Parse	two_contexts_same_config_url	0	2
1/1/2017:12:43:18:AM	2	2	30	:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Appliances:Dia gnature:Interface:Optimization_Opportunity:Traffic	throughput_calc_Prod126	1	2
1/1/2017:12:43:18:AM	2	2	31	:Automation:Bugs:Prod6044_Prod6027_Prod6009_Series_Adaptive_Securit ces:Diagnostic_Signature:Interface:Parser:troubleshooting	Defect1	0	2
1/1/2017:12:43:18:AM	2	2	32	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap evice_Hardening:Diagnostic_Signature:Prod6060:Parser:Management_1	telnet_input_enabled	0	2
1/1/2017:12:43:18:AM	2	2	33	:VPN:Optimization_Opportunity:Automation	Prod121_weak_encryption_hash_al n_use	1	2

hit_date	device _id	gate way	hit _id	hit_labels	hit_module	hit _severity	cu_id
1/1/2017:12:43:18:AM	2	2	34	:Management_1:Optimization_Opportunity:Automation	Timestamp_logging_disabled_in_co	1	2
1/1/2017:12:43:18:AM	2	2	35	:Automation:CPU_1:Prod6044_Prod6027_Prod6009_Series_Adaptive_Secu nces:Diagnostic_Signature:Software_Failure	long_cpu_hog_Prod126	0	2
1/1/2017:12:45:11:AM	2	2	14	:Application_Inspection:Automation:Prod6044_Prod6027_Prod6009_Series_ Security_Appliances:Diagnostic_Signature:Parser:Performance	Prod122_CX_IPS_performance_fail em_Defect2041_Prod126	0	2
1/1/2017:12:45:11:AM	2	2	35	:Automation:CPU_1:Prod6044_Prod6027_Prod6009_Series_Adaptive_Secu nces:Diagnostic_Signature:Software_Failure	long_cpu_hog_Prod126	0	2
1/1/2017:12:45:11:AM	2	2	31	:Automation:Bugs:Prod6044_Prod6027_Prod6009_Series_Adaptive_Securit ces:Diagnostic_Signature:Interface:Parser:troubleshooting	Defect1	0	2
1/1/2017:12:45:11:AM	2	2	18	:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Appliances:Dia gnature:Automation:Optimization_Opportunity	unused_config_module	1	2
1/1/2017:12:45:11:AM	2	2	15	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap evice_Hardening:Diagnostic_Signature:Optimization_Opportunity	recommended_asa_security_best_p	1	2
1/1/2017:12:45:11:AM	2	2	21	:Management_1:Prod6023:Software_Failure:Automation	Defect4907_Prod819_not_displayin Prod80_clients	1	2
1/1/2017:12:45:11:AM	2	2	10	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap iagnostic_Signature:Traffic	significant_TCP_embryonic_connect oing	0	2
1/1/2017:12:45:11:AM	2	2	16	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap iagnostic_Signature:Prod6059_1:Failover:Parser:troubleshooting	failover_int_checks_Prod126	0	2
1/1/2017:12:45:11:AM	2	2	20	:Automation:CPU_1:Prod6044_Prod6027_Prod6009_Series_Adaptive_Secu nces:Diagnostic_Signature:Software_Failure	snmp_cpu_hog_Prod126	0	2
1/1/2017:12:45:11:AM	2	2	29	:Automation:Prod6052_FW_Appliance:Prod6044_Prod6027_Prod6009_Seri ve_Security_Appliances:Configuration:Diagnosis:Diagnostic_Signature:Parse	two_contexts_same_config_url	0	2
1/1/2017:12:45:11:AM	2	2	9	:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Appliances:Dia gnature:Management_1	configuration_locked_in_another_se	0	2
1/1/2017:12:45:11:AM	2	2	22	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap onfiguration:Diagnostic_Signature:Optimization_Opportunity:Routing	route_check_Prod126	0	2
1/1/2017:12:45:11:AM	2	2	26	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap iagnostic_Signature:Memory	low_memory_Prod126	0	2
1/1/2017:12:45:11:AM	2	2	17	:ACL:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security es:Diagnostic_Signature:Prod6059_1:Optimization_Opportunity	acl_element_count_Prod126	1	2
1/1/2017:12:45:11:AM	2	2	12	:ACL:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security es:Diagnostic_Signature:Optimization_Opportunity	wide_open_acl	0	2
1/1/2017:12:45:11:AM	2	2	23	:Security:Optimization_Opportunity:Automation	console_timeout_of_0	1	2
1/1/2017:12:45:11:AM	2	2	27	:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Appliances:Bot _Filter:Automation:Diagnostic_Signature	botnet_updater_fails_ssl	0	2
1/1/2017:12:45:11:AM	2	2	11	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap iagnostic_Signature:Management_1:Parser:Software	asa_non_release_signed_image	0	2

hit_date	device _id	gate way	hit _id	hit_labels	hit_module	hit _severity	cu_id
1/1/2017:12:45:11:AM	2	2	28	:Automation:CPU_1:Prod6044_Prod6027_Prod6009_Series_Adaptive_Secu nces:Diagnostic_Signature:Optimization_Opportunity:Sham_Link	asa_high_cpu_Prod126	0	2
1/1/2017:12:45:11:AM	2	2	25	:Management_1:Optimization_Opportunity:Automation	Prod121_NTP_authentication_not_e	1	2
1/1/2017:12:45:11:AM	2	2	13	:Automation:Prod6052_FW_Appliance:Prod6044_Prod6027_Prod6009_Seri /e_Security_Appliances:Diagnostic_Signature	ip_audit_Prod126	0	2
1/1/2017:12:45:11:AM	2	2	19	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap onfiguration:Diagnostic_Signature:Prod6059_1:MPF:Optimization_Opportuni	infinite_conn_timeout_Prod126	0	2
1/1/2017:12:45:11:AM	2	2	34	:Management_1:Optimization_Opportunity:Automation	Timestamp_logging_disabled_in_co	1	2
1/1/2017:12:45:11:AM	2	2	30	:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Appliances:Dia gnature:Interface:Optimization_Opportunity:Traffic	throughput_calc_Prod126	1	2
1/1/2017:12:45:11:AM	2	2	33	:VPN:Optimization_Opportunity:Automation	Prod121_weak_encryption_hash_al n_use	0	2
1/1/2017:12:45:11:AM	2	2	32	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap evice_Hardening:Diagnostic_Signature:Prod6060:Parser:Management_1	telnet_input_enabled	0	2
1/1/2017:12:45:11:AM	2	2	24	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap onfiguration:Diagnostic_Signature:Hardware_Limitation:Prod6060:Interface: ɔn_Opportunity	show_interface_output_checks_bv3	0	2
1/1/2017:12:45:58:AM	1	1	8	:Device_Hardening:Automation	Prod357_Enable_Password	1	1
1/1/2017:12:45:58:AM	1	1	7	:Device_Hardening:Automation:Diagnostic_Signature:Prod6060	Prod356_type_4_password_used	1	1
1/1/2017:12:45:58:AM	1	1	6	:Device_Hardening:Automation	Prod357_unencrypted_password_fo	1	1
1/1/2017:12:45:58:AM	1	1	4	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap evice_Hardening:Diagnostic_Signature:Prod6060:Parser:Management_1	telnet_input_enabled	0	1
1/1/2017:12:55:10:AM	3	2	36	:Automation:Prod6052_FW_Appliance:Prod6044_Prod6027_Prod6009_Seri /e_Security_Appliances:Diagnostic_Signature	ip_audit_Prod126	0	3
1/1/2017:12:55:10:AM	3	2	37	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap evice_Hardening:Diagnostic_Signature:Optimization_Opportunity	recommended_asa_security_best_p	1	3
1/1/2017:12:55:10:AM	3	2	38	:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Appliances:Bot _Filter:Automation:Diagnostic_Signature	botnet_updater_fails_ssl	0	3
1/1/2017:12:55:10:AM	3	2	39	:ACL:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security ɜs:Diagnostic_Signature:Prod6059_1:Optimization_Opportunity	acl_element_count_Prod126	1	3
1/1/2017:12:55:10:AM	3	2	40	:ACL:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security ɜs:Diagnostic_Signature:Optimization_Opportunity	wide_open_acl	0	3
1/1/2017:12:55:10:AM	3	2	41	:Automation:CPU_1:Prod6044_Prod6027_Prod6009_Series_Adaptive_Secu nces:Diagnostic_Signature:Software_Failure	long_cpu_hog_Prod126	0	3

hit_date	device _id	gate way	hit _id	hit_labels	hit_module	hit _severity	cu_id
1/1/2017:12:55:10:AM	3	2	42	:Security:Optimization_Opportunity:Automation	console_timeout_of_0	1	3
1/1/2017:12:55:10:AM	3	2	43	:Application_Inspection:Automation:Prod6044_Prod6027_Prod6009_Series_ Security_Appliances:Diagnostic_Signature:Parser:Performance	Prod122_CX_IPS_performance_fail em_Defect2041_Prod126	0	3
1/1/2017:12:55:10:AM	3	2	44	:Automation:Bugs:Prod6044_Prod6027_Prod6009_Series_Adaptive_Securit :es:Diagnostic_Signature:Interface:Parser:troubleshooting	Defect1	0	3
1/1/2017:12:55:10:AM	3	2	45	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap iagnostic_Signature:Management_1:Parser:Software	asa_non_release_signed_image	0	3
1/1/2017:12:55:10:AM	3	2	46	:VPN:Optimization_Opportunity:Automation	Prod121_weak_encryption_hash_al n_use	1	3
1/1/2017:12:55:10:AM	3	2	47	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap onfiguration:Diagnostic_Signature:Optimization_Opportunity:Routing	route_check_Prod126	0	3
1/1/2017:12:55:10:AM	3	2	48	:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Appliances:Dia gnature:Interface:Optimization_Opportunity:Traffic	throughput_calc_Prod126	1	3
1/1/2017:12:55:10:AM	3	2	49	:Management_1:Optimization_Opportunity:Automation	Timestamp_logging_disabled_in_co	1	3
1/1/2017:12:55:10:AM	3	2	50	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap onfiguration:Diagnostic_Signature:Prod6059_1:MPF:Optimization_Opportuni	infinite_conn_timeout_Prod126	0	3
1/1/2017:12:55:10:AM	3	2	51	:Automation:CPU_1:Prod6044_Prod6027_Prod6009_Series_Adaptive_Secu nces:Diagnostic_Signature:Software_Failure	snmp_cpu_hog_Prod126	0	3
1/1/2017:12:55:10:AM	3	2	52	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap evice_Hardening:Diagnostic_Signature:Prod6060:Parser:Management_1	telnet_input_enabled	1	3
1/1/2017:12:55:10:AM	3	2	53	:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Appliances:Dia gnature:Management_1	configuration_locked_in_another_se	0	3
1/1/2017:12:55:10:AM	3	2	54	:Automation:Prod6052_FW_Appliance:Prod6044_Prod6027_Prod6009_Seri ve_Security_Appliances:Configuration:Diagnosis:Diagnostic_Signature:Parse	two_contexts_same_config_url	0	3
1/1/2017:12:55:10:AM	3	2	55	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap iagnostic_Signature:License:Traffic:licensing	Defect3_conns_dropped_with_licen rod126	0	3
1/1/2017:12:55:10:AM	3	2	56	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap iagnostic_Signature:Hardware_Limitation:Memory	256mb_5505_Prod126	0	3
1/1/2017:12:55:10:AM	3	2	57	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap iagnostic_Signature:Memory	low_memory_Prod126	0	3
1/1/2017:12:55:10:AM	3	2	58	:Automation:CPU_1:Prod6044_Prod6027_Prod6009_Series_Adaptive_Secu nces:Diagnostic_Signature:Optimization_Opportunity:Sham_Link	asa_high_cpu_Prod126	0	3
1/1/2017:12:55:10:AM	3	2	59	:Automation:Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Ap iagnostic_Signature:Traffic	significant_TCP_embryonic_connect ping	0	3

Appendix B – Issue Labels

#	Tag / Label	Events
1	Automation	9249928
2	Diagnostic_Signature	6728559
3	Prod6060	3003747
4	Prod6044_Prod6027_Prod6009_Series_Adaptive_Security_Appliances	2984271
5	Configuration	2363640
6	Bugs	1982050
7	Optimization_Opportunity	1918388
8	NX_OS	1686555
9	Software_Failure	1490081
10	Crash	1228464
11	Parser	852486
12	Routing_Protocols	754951
13	Interface	744118
14	Failover	538493
15	Hardware_Limitation	510041
16	NetworkAddressTranslation	506936
17	Prod6059_1	491489
18	Management_1	434831
19	troubleshooting	393194
20	Prod6052_FW_Appliance	387020
21	OSPF	343531
22	Memory	327564
23	Diagnosis	292519
24	Routing	286577
25	Healthcheck	280593
26	ACL	247119
27	Security	246064
28	Education	223184
29	Logging	222658
30	Device_Hardening	216477
31	CPU_1	206909
32	BGP	202848
33	Software_Limitation	181266
34	VPN	175832
35	Hardware_Failure	157428
36	Platform	154042
37	Voice	139697
38	Prod6066	138541
39	MPLS	137642
40	Memory_Depletion	129366
41	Hardware	119254
42	Application_Inspection	114200
43	Prod6023	114149
44	EIGRP	103970
45	Voice_Protocol	103957
46	Performance	102733
47	Traffic	95235
48	MPF	94935
49	Traceback	86730
50	licensing	76690
51	Best_Practice	75671
52	Operations_Guide	75671

#	Tag / Label	Events
53	Validation	75671
54	Field_Notice	74283
55	Voice_Gateway	70832
56	Voice_Apps	69504
57	SIP	69225
58	Prod6054	69091
59	SRST	68952
60	id_Prod6013_Switch	68919
61	PBR	68615
62	System	68420
63	AAA	59494
64	Software	52105
65	Botnet_Traffic_Filter	51150
66	Sham_Link	49501
67	Prod6093	48103
68	Prod6034	46015
69	Prod6028	44521
70	Content_Filtering	43065
71	Transparent_Firewall	43059
72	Prod6043_Switch	42501
73	Prod6060_XE	38787
74	Access_List	38612
75	System_Resource	37250
76	Prod6026	37201
77	Call_Routing	35806
78	Prod6098	34637
79	Fax_Modem	34635
80	NTP	34600
81	Dial_peer	34576
82	Prod6053	34573
83	Prod6053_Enterprise	34573
84	Multicast	34398
85	IGMP	34333
86	IP_Multicast	34333
87	SCCP	34328
88	RIP	34293
89	RIPv2	34293
90	MPLS_TE	34224
91	id_Prod6000_Switch	33715
92	id_Prod6004_Switch	33423
93	id_Prod6002_Switch	33099
94	Dynamic_Fabric_Automation	32178
95	Prod6067	32178
96	Speed	32178
97	Clustering	18577
98	Prod6082_Prod6014	16455
99	Incident1	15044
100	Prod6080	14977
101	Prod6082_Prod6012	13478
102	Sourcefire_on_Prod6027	12120
103	Prod6070	11802
104	License	10181
105	Prod6057	9985
106	Prod6082_Prod6008	9928

#	Tag / Label	Events
107	Prod6082_2000	9824
108	Spanning_Tree	8923
109	MDS	8432
110	MTS	8432
111	Reset	8027
112	id_Prod6005_Switch	6600
113	System_Management	6019
114	Prod6082_Prod6010	3927
115	Diagnostic signature	3715
116	Parity_Error	3546
117	GOLD	3198
118	Tool	3104
119	PKI	2270
120	Prod6082_Prod6020	1939
121	IPSEC	1783
122	Prod6082_Prod6009	1727
123	Forwarding_Engine	1201
124	vPC	1155
125	Other	1094
126	Switching	1018
127	Power	1003
128	Voice_Security	785
129	Diagnostic-signature	719
130	Diagnostic_Signature_	660
131	id_Prod6007_Switch	632
132	Prod6082_Prod6021	543
133	Etherchannel	519
134	Transceiver	509
135	Route	467
136	IPv6	374
137	PFR	343
138	Diagnostic_Signature_Prod6093_Automation_Diagnostic signature	339
139	AppNav_Controllers	337
140	Debugging	318
141	Diagnostic_Signature_Prod6093_Prod6082_Prod6014_Automation_Bugs_Software_Failure	307
142	Diagnostic_Signature_Automation_Prod6080_Hardware_Failure_Bugs	237
143	id_Prod6022_Router	208
144	Prod6082_Prod6015	183
145	Prod6082_Prod6016	177
146	Temperature	176
147	Python	170
148	Diagnostic_Signature_Prod6093_Prod6080_Prod6070_Software_Failure_Automation_Bugs	169
149	Diagnostic_Signature_Prod6080_Automation_Prod6093_Software_Failure_Bugs	142
150	Diagnostic_Signature_Prod6082_Prod6014_Automation_Bugs_Software_Failure	142
151	Diagnostic_Signature_Prod6093_Prod6082_Prod6014_Bugs_Software_Failure_Automation	142
152	Diagnostic_Signature_Prod6080_Automation_Bugs_Software_Failure_Prod6093_Prod6070	141
153	Diagnostic_Signature_Prod6080_Bugs_Software_Failure_Automation_Prod6093	141
154	Diagnostic_Signature_Prod6080_Software_Failure_Bugs_Prod6093_Prod6070_Automation	141
155	Diagnostic_Signature_Prod6080_Prod6070_Bugs_Software_Failure_Automation_Prod6093	140
156	Prod6082_Prod6019	139
157	EEM	137
158	ASR_1	130
159	Cognitive	109
160	PSU	109

#	Tag / Label	Events
161	Prod6077	90
162	Diagnostic_Signature_Prod6082_Prod6012_Software_Failure_Bugs_Automation_Prod6093	82
163	Diagnostic_Signature_Prod6080_Software_Failure_Bugs_Prod6093_Automation	80
164	Flash	77
165	Diagnastic signature	75
166	Diagnostic_Signature_Prod6070_Software_Failure_Bugs_Automation_Prod6093	73
167	MTU	72
168	GRE	65
169	Diagnostic_Signature__	50
170	SSO	50
171	Fabric	49
172	Diagnostic_Signature_Software_Failure_Automation_Bugs_Prod6080	40
173	Module	40
174	Errors	38
175	Redundancy	31
176	Voice_Quality	31
177	xbar	29
178	Storage	28
179	Bootup	27
180	Diagnostic_Signature_Automation_Software_Failure_Bugs_Prod6082_Prod6010_Prod6082_Prod6012	27
181	RPR	27
182	RPR_1	27
183	L2L	26
184	Call_Control	24
185	Duplex	24
186	Stacking	23
187	DHCP	21
188	Diagnostic_Signature_Prod6070_Bugs_Software_Failure_Automation_Prod6093	21
189	RMA	21
190	FCOE	19
191	Prod6079	19
192	Diagnostic_Signature_Prod6080_Software_Failure_Bugs_Automation_Prod6093	18
193	Enhancement	18
194	Bug	16
195	CoPP	16
196	Diagnostic_Signature_Prod6070_Automation_Bugs_Software_Failure_Prod6093	16
197	Lead_Generation	16
198	Netflow	16
199	Switchover	16
200	Diagnostic_Signature_Prod6080_Prod6093_Software_Failure_Bugs_Automation	15
201	Diagnostic_Signature_Prod6080_Software_Failure_Automation_Bugs	15
202	Detect6350	14
203	Prod6084	14
204	Diagnostic_Signature_Prod6080_Software_Failure_Automation_Bugs_Prod6093	13
205	Diagnostic_Signature_Prod6080_Software_Failure_Automation_Prod6093_Bugs	13
206	Silent_Monitoring	13
207	Diagnostic_Signature_Automation_Prod6093_Prod6080_Software_Failure_Bugs	12
208	Diagnostic_Signature_Prod6080_Automation_Prod6093_Bugs_Software_Failure	12
209	Diagnostic_Signature_Prod6080_Automation_Software_Failure_Bugs_Prod6093	12
210	Diagnostic_Signature_Prod6080_Prod6093_Automation_Bugs_Software_Failure	12
211	MAC_Address	12
212	Diagnostic_Signature_Prod6080_Automation_Bugs_Software_Failure_Prod6093	11
213	Prod6101	11
214	Diagnostic_Signature_Prod6080_Software_Failure_Bugs_Prod6093_Software_Failure_Automation	10

#	Tag / Label	Events
215	LACP	10
216	PoE	10
217	Diagnostic_Signature_Prod6070_Automation_Prod6093_Software_Failure_Bugs	9
218	Diagnostic_Signature_Prod6070_Automation_Software_Failure_Bugs_Prod6093	9
219	Diagnostic_Signature_Prod6070_Prod6093_Software_Failure_Automation_Bugs	9
220	Diagnostic_Signature_Prod6080_Prod6093_Software_Failure_Automation_Bugs	9
221	Automaiton	8
222	Diagnostic-sognature	8
223	Diagnostic_Signature_Prod6079_Prod6070_Automation_Software_Failure_Bugs_Prod6093	8
224	Diagnostic_Signature_Prod6080_Software_Limitation_Automation_Prod6093_Bugs	8
225	Diagnostic_Signature_Prod6082_Prod6014_Software_Failure_Bugs_Prod6093_Automation	8
226	DMVPN	8
227	NHRP	8
228	Diagnostic_Signature_Automation_Software_Failure_Bugs_Prod6080	7
229	Diagnostic_Signature_Prod6070_Automation_Prod6093_Bugs_Software_Failure	7
230	Diagnostic_Signature_Prod6082_Prod6014_Automation_Software_Failure_Bugs_Prod6093	7
231	Software_Failuref	7
232	V1	6
233	Diagnostic_Signature_Hardware_Failure_Prod6093_Prod6082_Prod6014_Automation_Bugs	6
234	Diagnostic_Signature_Prod6070_Prod6093_Software_Failure_Bugs_Automation	6
235	Diagnostic_Signature_Prod6070_Software_Failure_Bugs_Prod6093_Automation	6
236	Diagnostic_Signature_Prod6093_Automation_Bugs_Software_Failure	6
237	Diagnostic_Signature_Prod6093_Prod6080_Automation_Software_Failure_Bugs	6
238	Prod6082_Prod6014_Bugs	6
239	Prod6083	6
240	Troubleshooting_Guide	6
241	Controller	5
242	Diagnostic_Signature_Automation_Prod6093_Bugs_Prod6070_Software_Failure	5
243	Diagnostic_Signature_Automation_Prod6093_Prod6070_Software_Failure_Bugs	5
244	Diagnostic_Signature_Prod6070_Automation_Software_Failure_Incident1_Prod6093	5
245	Diagnostic_Signature_Prod6070_Prod6080_Software_Failure_Bugs_Automation_Prod6093	5
246	Diagnostic_Signature_Prod6070_Software_Failure_Incident1_Automation_Bugs_Prod6093	5
247	Diagnostic_Signature_Prod6080_Incident1_Software_Failure_Prod6093_Prod6070	5
248	Diagnostic_Signature_Prod6080_Prod6070_Automation_Bugs_Software_Failure	5
249	Diagnostic_Signature_Prod6082_Prod6014_Prod6093_Automation_Bugs_Software_Failure	5
250	Diagnostic_Signature_Prod6093_Automation_Bugs_Software_Failure_Prod6082_Prod6014_Prod6070_Prod6080	5
251	Diagnostic_Signature_Prod6093_Automation_Bugs_Software_Failure_Prod6082_Prod6014_Prod6080	5
252	Diagnostic_Signature_Prod6093_Prod6080_Software_Failure_Bugs_Automation	5
253	ISDN	5
254	Diagnostic Signature	4
255	Diagnostic_Signature_Automation_Prod6080_Software_Failure_Bugs_Prod6093	4
256	Diagnostic_Signature_Automation_Prod6082_Prod6014_Prod6080_Software_Failure_Bugs_Prod6093	4
257	Diagnostic_Signature_Automation_Prod6093_Prod6082_Prod6014_Bugs_Software_Failure	4
258	Diagnostic_Signature_Automation_Software_Failure_Bugs_Prod6082_Prod6014_Prod6093	4
259	Diagnostic_Signature_Automation_Software_Limitation_Prod6080_Bugs	4
260	Diagnostic_Signature_Prod6070_Automation_Software_Failure_Incident1	4
261	Diagnostic_Signature_Prod6070_Bugs_Automation_Software_Failure	4
262	Diagnostic_Signature_Prod6070_Prod6093_Bugs_Software_Failure_Automation	4
263	Diagnostic_Signature_Prod6070_Software_Failure_Prod6093_Automation_Bugs	4
264	Diagnostic_Signature_Prod6079_Software_Failure_Automation_Prod6093_Bugs	4
265	Diagnostic_Signature_Prod6080_Prod6070_Prod6082_Prod6014_Prod6082_Prod6010_Prod6082_Prod6012_Automation_Bugs_Software_Failure_Prod6093	4

#	Tag / Label	Events
266	Diagnostic_Signature_Prod6082_Prod6014_Automation_Prod6093_Bugs_Software_Failure_Prod6080_Prod6070	4
267	Diagnostic_Signature_Prod6082_Prod6014_Prod6093_Automation_Bugs_Software_Failure_Incident1	4
268	Diagnostic_Signature_Prod6082_Prod6014_Software_Failure_Automation_Bugs_Software_Failure	4
269	Diagnostic_Signature_Prod6093_Automation_Prod6082_Prod6014_Bugs_Software_Failure	4
270	Diagnostic_Signature_Prod6093_Prod6070_Automation_Bugs_Software_Failure	4
271	Diagnostic_Signature_Prod6093_Prod6070_Automation_Software_Failure_Bugs	4
272	Diagnostic_Signature_Prod6093_Prod6070_Software_Failure_Automation_Bugs	4
273	Diagnostic_Signature_Prod6093_Prod6077_Prod6082_Prod6009_Prod6082_Prod6010_Prod6082_Prod6012_Automation_Bugs_Software_Failure	4
274	Diagnostic_Signature_Prod6093_Prod6080_Automation_Bugs_Software_Failure	4
275	Diagnostic_Signature_Prod6093_Prod6082_Prod6014_Software_Failure_Bugs_Automation	4
276	Diagnostic_Signature_Prod6093_Software_Failure_Bugs_Automation_Prod6077	4
277	HA	4
278	IP_Phone	4
279	Prod6045	4
280	Prod6089	4
281	QoS	4
282	AA	3
283	Diagnostic_Signature_Automation_Prod6070_Bugs_Software_Failure_Automation	3
284	Diagnostic_Signature_Automation_Prod6082_Prod6014_Software_Failure_Prod6093_Bugs	3
285	Diagnostic_Signature_Automation_Prod6093_Bugs_Prod6080_Software_Failure	3
286	Diagnostic_Signature_Automation_Prod6093_Bugs_Software_Failure_Prod6070	3
287	Diagnostic_Signature_Automation_Prod6093_Diagnostic-signature	3
288	Diagnostic_Signature_Automation_Prod6093_Prod6070_Bugs_Software_Failure	3
289	Diagnostic_Signature_Automation_Software_Failure_Bugs_Prod6070_Prod6080_Prod6082_Prod6014_Prod6093	3
290	Diagnostic_Signature_Prod6070_Automation_Bugs_Prod6093_Software_Failure	3
291	Diagnostic_Signature_Prod6070_Automation_Bugs_Software_Failure_Prod6093_Prod6080	3
292	Diagnostic_Signature_Prod6070_Prod6093_Automation_Software_Failure_Bugs	3
293	Diagnostic_Signature_Prod6070_Prod6093_Bugs_Automation_Software_Failure_Prod6080	3
294	Diagnostic_Signature_Prod6080_Automation_Bugs_Software_Failure	3
295	Diagnostic_Signature_Prod6080_Automation_Prod6082_2000_Software_Failure_Bugs_Prod6093	3
296	Diagnostic_Signature_Prod6080_Automation_Software_Failure_Prod6093_Bugs	3
297	Diagnostic_Signature_Prod6080_Bugs_Software_Failure_Prod6093_Automation	3
298	Diagnostic_Signature_Prod6080_Enhancement_Bugs_Automation	3
299	Diagnostic_Signature_Prod6080_Prod6070_Automation_Software_Failure_Bugs_Prod6093	3
300	Diagnostic_Signature_Prod6082_Prod6010_Prod6082_Prod6012_Prod6093_Software_Failure_Bugs_Automation	3
301	Diagnostic_Signature_Prod6082_Prod6014_Prod6093_Software_Failure_Automation	3
302	Diagnostic_Signature_Prod6082_Prod6014_Software_Failure_Bugs_Automation_Prod6093	3
303	Diagnostic_Signature_Prod6093_Automation_Software_Failure_Bugs_Prod6082_Prod6014	3
304	Diagnostic_Signature_Prod6093_Prod6077_Software_Failure_Automation	3
305	Diagnostic_Signature_Software_Failure_Automation_Prod6070_Prod6093_Bugs	3
306	Diagnostic_Signature_vpc_Prod6058_inactive_Prod6083_Prod6076_AA	3
307	inactive	3
308	netflow	3
309	Prod6058	3
310	Prod6071	3
311	Prod6074	3
312	Prod6076	3
313	Prod6090	3
314	Specialized_IC	3
315	vpc	3

#	Tag / Label	Events
316	Diagnostic_Signature_Automation_Prod6093_Diagnostic signature	2
317	Diagnostic_Signature_Automation_Prod6093_Software_Failure_Prod6080_Prod6093	2
318	Diagnostic_Signature_Prod6070_Automation_Incident1_Prod6093_Software_Failure	2
319	Diagnostic_Signature_Prod6070_Automation_Software_Failuref_Bugs_Prod6093	2
320	Diagnostic_Signature_Prod6077_Automation_Prod6093_Bugs_Software_Failure	2
321	Diagnostic_Signature_Prod6080_Prod6070_Software_Failure_Automation_Prod6093_Bugs	2
322	Diagnostic_Signature_Prod6080_Prod6070_Software_Failure_Bugs_Automation_Prod6093	2
323	Diagnostic_Signature_Prod6080_Prod6082_2000_Bugs_Software_Failure_Automation_Prod6093	2
324	Diagnostic_Signature_Prod6082_Prod6010_Prod6093_Prod6082_Prod6012_Software_Failure_Bugs_Automati on	2
325	Diagnostic_Signature_Prod6082_Prod6014_Software_Failure_Automation_Prod6093_Bugs	2
326	Diagnostic_Signature_Prod6093_Automation_Bugs_Software_Failure_Prod6070	2
327	Diagnostic_Signature_Prod6093_Automation_Prod6080_Software_Failure_Bugs	2
328	Diagnostic_Signature_Prod6093_Automation_Software_Failure_Prod6082_Prod6014_Bugs	2
329	Diagnostic_Signature_Prod6093_Bugs_Software_Failure_Prod6082_Prod6014_Prod6080_Automation_Bugs	2
330	Diagnostic_Signature_Prod6093_Software_Failure_Bugs_Automation_Prod6082_Prod6014	2
331	Diagnostic_Signature_Software_Failure_Automation_Prod6093_Prod6070_Bugs	2
332	Diagnostic_Signature_Software_Failure_Bugs_Prod6093_Automation_Prod6082_Prod6014	2
333	DSP	2
334	FD Error	2
335	Netstack	2
336	Prod6091	2
337	PVDM	2
338	Clientless	1
339	Clocking	1
340	Diagnostic_Signature_Automation_Diagnostic signature	1
341	Diagnostic_Signature_Automation_Prod6093_Software_Failure_Bugs_Prod6080	1
342	Diagnostic_Signature_Detect6350	1
343	Diagnostic_Signature_Prod6070_Prod6080_Bugs_Software_Failure_Automation	1
344	Diagnostic_Signature_Prod6070_Prod6093_Automation_Bugs_Software_Failure	1
345	Diagnostic_Signature_Prod6070_Software_Failure_Bugs_Automation	1
346	Diagnostic_Signature_Prod6080_Automation_Software_Failure_Bugs	1
347	Diagnostic_Signature_Prod6080_Bugs_Software_Failure_Automation	1
348	Diagnostic_Signature_Prod6080_Prod6093_Automation_Software_Failure_Bugs	1
349	Diagnostic_Signature_Prod6080_Software_Failure_Bugs_Automation	1
350	Diagnostic_Signature_Prod6082_2000_Prod6082_Prod6010_Prod6082_Prod6012_Prod6093_Software_Failure _Field_Notice_Bugs_Automation	1
351	Diagnostic_Signature_Prod6082_Prod6010_Prod6082_Prod6012_Prod6093_Bugs_Automation_Software_Failu re	1
352	Diagnostic_Signature_Prod6082_Prod6014_Bugs_Software_Failure_Automation_Prod6093	1
353	Diagnostic_Signature_Prod6082_Prod6014_Prod6093_Software_Failure_Automation_Bugs	1
354	Diagnostic_Signature_Prod6082_Prod6014_Prod6093_Software_Failure_Bugs	1
355	Diagnostic_Signature_Prod6082_Prod6014_Software_Failure_Automation_Bugs_Prod6093	1
356	Diagnostic_Signature_Prod6082_Prod6014_Software_Failure_Prod6093_Automation_Bugs	1
357	Diagnostic_Signature_Prod6093_Diagnostic Signature_Automation	1
358	Diagnostic_Signature_Prod6093_Prod6070_Bugs_Automation_Software_Failure	1
359	Diagnostic_Signature_Prod6093_Prod6082_Prod6014_Software_Failure_Automation_Bugs	1
360	Diagnostic_Signature_Prod6093_Prod6084	1
361	Diagnostics	1
362	Failure	1

Appendix C – Issue Module IDs

	Issue	Devices	Events
1	long_cpu_hog_Prod126	9259	220964
2	asa_high_cpu_Prod126	7474	153901
3	Low_Free_Memory_Available	6861	148278
4	memory_used_greater_than_100percent_Prod334	6837	148259
5	snmp_cpu_hog_Prod126	6676	150181
6	extremely_long_cpu_hog	6282	144878
7	CPU_hogs_due_to_SNMP_polling	4724	33040
8	Defect2938_Defection_in_memory	3509	26063
9	low_memory_Prod126	479	4314
10	Prod128_compliancy_checks_CPU_memory_failure_Prod126	187	3172
11	recent_datapath_CPU_hogs	53	115
12	4500_CPU_Tshoot	2	8
13	bdblib_show_proc_cpu_Prod356_v2	2	496
14	Defect1477_memory_leak_qos_mon_periodic	2	482
15	Defect5215_memory_leak_auth_manager	2	277
16	EEM_Tool_HighCPU	2	344
17	memory_validation_Prod357_fork	2	337
18	Prod262_Defect1460_CPU_queue_gets_stuck	2	6
19	Prod262_Defect4512_Defect3013_Port_sec_DHCPv6_clogging_CPU	2	5
20	Prod262_Defect4785_memory_inconsistency_Defected	2	9
21	Prod269_Defect4070_High_cpu	2	9
22	Prod269_Prod797_high_memory_util	2	15
23	Prod270_Defect1161_CPU_HOG_UDLD	2	18
24	Prod270_Defect2395_high_cpu_6704	2	6
25	Prod270_Defect2835_cpu_hog	2	11
26	Prod357_Defect1895_High_memory_utilization	2	5
27	Prod357_Defect2440_Prod267_High_CPU	2	103
28	Prod357_Defect2876_Memory_leak_observed	2	151
29	Prod357_Defect3527_High_CPU_utilization	2	46
30	Prod357_Defect4104_Memory_leak_on	2	154
31	Prod357_Defect4596_IO_pool_memory	2	86
32	Prod357_Defect4740_Prod140_Memory_Leak	2	48
33	Prod357_Defect5358_High_CPU_due	2	13
34	Prod357_Defect5996_Low_memory_issues	2	120
35	Prod357_Defect6239_Prod266_High_CPU	2	138
36	Prod357_Defect6339_High_memory_utilization	2	27
37	Prod357_Defect6452_Memory_leak_with	2	161
38	Prod357_Defect684_CPUHOG_seen_during	2	16
39	Prod357_Defect6968_Prod198_High_CPU	2	132
40	Prod357_Defect7159_Temporary_high_CPU	2	45
41	Prod359_Defect309_Share_line_memory	2	5947
42	Prod359_Defect919_Memory_leak_observed	2	4355
43	Prod575_Defect2627_Prod507_VccP_Memory_Component_Issue	2	19535
44	Prod575_Defect2855_Enh_Need_CPU	2	8
45	Prod575_Defect3349_PTP_memory_leak_leading_to_a_crash	2	46
46	Prod575_Defect3981_Prod526_Kernel_Panic_watchdog_timeout_issue_on_CPU2	2	18028
47	Prod575_Defect5472_Prod489_eem_policy_dir_memory	2	420
48	Prod575_Defect6027_SNMPd_Memory_Leak_in_libport_mgr_common	2	19535
49	Prod575_Defect7241_DHCP_paks_punted_to_CPU_when_feature_is_disabled_on_trans	2	19533
50	Defect1477_memory_leak_qos_mon_periodic [!](DUPLICATE 1)	1	2
51	Prod121_Defect5742_Memory_leak_in_DP_udp_host_logging	1	4
52	Prod247_Prod254_Defect5263_memory_leak_Event198	1	6

	Issue	Devices	Events
53	Prod262_Defect6805_Memory_leak_under_Event199	1	1
54	Prod357_Defect1075_Confusing_CPU_Over	1	103
55	Prod357_Defect1466_cman-fpcman-cc_slow_memory	1	77
56	Prod357_Defect1497_Buffer_memory_leak	1	44
57	Prod357_Defect1610_Event401_Spurious_memory	1	127
58	Prod357_Defect1784_Prod263_memory_leak	1	6
59	Prod357_Defect2106_X25_memory_leak	1	13
60	Prod357_Defect2174_IO_memory_leak	1	157
61	Prod357_Defect2291_High_CPU_and	1	3
62	Prod357_Defect2382_Memory_leak_under	1	47
63	Prod357_Defect263_Memory_exhaustion_by	1	60
64	Prod357_Defect2805_CPU_hog_TB	1	5
65	Prod357_Defect338_memory_leak_in	1	1
66	Prod357_Defect3398_ISDN_memory_leak	1	310
67	Prod357_Defect345_SSLVPN_PROCESS_memory_leak	1	123
68	Prod357_Defect351_Memory_leak_in	1	3
69	Prod357_Defect3573_Memory_Fragmentation_in	1	27
70	Prod357_Defect3769_Event364_CPUHOG_1_fed	1	94
71	Prod357_Defect3902_Prod254_-_Memory	1	32
72	Prod357_Defect4035_Memory_Leak_in	1	73
73	Prod357_Defect4114_High_memory_utilization	1	115
74	Prod357_Defect4172_Prod150_memory_leak	1	16
75	Prod357_Defect4329_Memory_leak_when	1	226
76	Prod357_Defect4501_Memory_leak_under	1	108
77	Prod357_Defect4581_NHRP_CPUHOGs_seen	1	2
78	Prod357_Defect47_AP_IO_memory	1	2
79	Prod357_Defect5279_ESP_Committed_memory	1	131
80	Prod357_Defect5309_Memory_leak_in	1	3
81	Prod357_Defect5957_Memory_buildup_with	1	143
82	Prod357_Defect6067_CSM_memory_leak	1	5
83	Prod357_Defect6096_Memory_Leak_due	1	133
84	Prod357_Defect6660_Memory_leak_at	1	10
85	Prod357_Defect6936_Prod312_-_Memory	1	68
86	Prod357_Defect711_High_memory_utilization	1	102
87	Prod357_Defect7145_Prod16_High_CPU	1	36
88	Prod357_Defect7260_High_CPU_in	1	13
89	Prod359_Defect1716_ISDN_Memory_Leak	1	2587
90	Prod359_Defect6820_corrupted_memory_crash	1	3852
91	Prod575_Defect1170_With_VXLAN_VPC_IPV6_RA_leaving_Prod553_CPU	1	29
92	Prod575_Defect6010_High_CPU_caused	1	434
93	Prod575_Defect6333_Prod553_Prod60_memory	1	3
94	show_proc_cpu_Prod356_v2	1	3

References

Albert, S. (2018, November 13). *Boosting with AdaBoost and Gradient Boosting*. Medium. https://medium.com/diogo-menezes-borges/boosting-with-adaboost-and-gradient-boosting-9cbab2a1af81

Allen, D. M., & Goloubew, D. (2020). Customer Self-remediation of Proactive Network Issue Detection and Notification. In H. Degen & L. Reinerman-Jones (Eds.), *Artificial Intelligence in HCI* (pp. 197–210). Springer International Publishing. https://doi.org/10.1007/978-3-030-50334-5_13

Avizienis, A., Laprie, J.-C., Randell, B., & Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, *1*(1), 11–33. https://doi.org/10.1109/TDSC.2004.2

Boutaba, R., Salahuddin, M. A., Limam, N., Ayoubi, S., Shahriar, N., Estrada-Solano, F., & Caicedo, O. M. (2018). A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities. *Journal of Internet Services and Applications*, *9*(1), 16. https://doi.org/10.1186/s13174-018-0087-2

Brownlee, J. (2016, August 16). A Gentle Introduction to XGBoost for Applied Machine Learning. *Machine Learning Mastery*. https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

Brownlee, J. (2017, August 31). How to Diagnose Overfitting and Underfitting of LSTM Models. *Machine Learning Mastery*. https://machinelearningmastery.com/diagnose-overfitting-underfitting-lstm-models/

Brownlee, J. (2019, November 26). How to Choose a Feature Selection Method For Machine Learning. *Machine Learning Mastery*. https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, *21*(1), 6. https://doi.org/10.1186/s12864-019-6413-7

Choi, E., Schuetz, A., Stewart, W. F., & Sun, J. (2017). Using recurrent neural network models for early detection of heart failure onset. *Journal of the American Medical Informatics Association : JAMIA*, *24*(2), 361–370. https://doi.org/10.1093/jamia/ocw112

Chollet, F., & Allaire, J. J. (2018). *Deep learning with R*. Manning Publications Co.

Chollet, F., & others. (2015). *Keras*. https://keras.io

Cisco Systems. (n.d.). *Cisco Nexus 1000V - Configuring Telnet*. Cisco. Retrieved August 18, 2020, from https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus1000/sw/4_0/security/configuration/guide/n1000v_security/security_7telnet.html

Cisco Systems. (2016). *Troubleshooting High CPU Utilization*. Cisco. https://www.cisco.com/c/en/us/support/docs/routers/10000-series-routers/15095-highcpu.html

Cisco Systems. (2017). *Connected_TAC_At-A-Glance.pdf*. https://www.cisco.com/c/dam/en/us/support/docs/services/connected-tac/Connected_TAC_At-A-Glance.pdf

Cisco Systems. (2020). *Cisco Diagnostic Bridge Installation and User Guide*. https://www.cisco.com/c/dam/en/us/support/docs/services/connected-tac/Cisco_Diagnostic_Bridge_Getting_Started_Guide.pdf

Cisco Telnet Vulnerability. (2020, June). https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-telnetd-EFJrEzPx

Conway, & Gehlenborg. (2019). *UpSetR Basic Usage*. https://cran.r-project.org/web/packages/UpSetR/vignettes/basic.usage.html

Czakon, J. (2020, January 16). *The ultimate guide to binary classification metrics*. Medium. https://towardsdatascience.com/the-ultimate-guide-to-binary-classification-metrics-c25c3627dd0a

Dutta, A. (2019, December 26). *System Failure Prediction using log analysis*. Medium. https://towardsdatascience.com/system-failure-prediction-using-log-analysis-8eab84d56d1

Eusgeld, I., Freiling, F., & Reussner, R. (2008). *Dependability Metrics*. 304.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, *27*(8), 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

Fernández-Delgado, M., Cernadas, E., Barro, S., & Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems? *The Journal of Machine Learning Research*, *15*(1), 3133–3181.

Financial Times. (2020). *CSCO*. https://markets.ft.com/data/equities/tearsheet/profile?s=CSCO:NSQ

Freund, Y., & Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139. https://doi.org/10.1006/jcss.1997.1504

Futoma, J., Hariharan, S., & Heller, K. (2017). Learning to Detect Sepsis with a Multitask Gaussian Process RNN Classifier. *ArXiv:1706.04152 [Stat]*. http://arxiv.org/abs/1706.04152

Ganguly, S., Consul, A., Khan, A., Bussone, B., Richards, J., & Miguel, A. (2016). A Practical Approach to Hard Disk Failure Prediction in Cloud Platforms: Big Data Model for Failure Management in Datacenters. *2016 IEEE Second International Conference on Big Data Computing Service and Applications (BigDataService)*, 105–116. https://doi.org/10.1109/BigDataService.2016.10

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (2 edition). O’Reilly Media.

Guyon, I., & Elisseeff, A. (2003). An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, *3*, 26.

Huang, X. (2017). *Hard Drive Failure Prediction for Large Scale Storage System* [UCLA]. https://escholarship.org/uc/item/11x380ng

IBM. (2014, October 24). *Detecting flapping events*. www.ibm.com/support/knowledgecenter/ssurrn/com.ibm.cem.doc/em_flapping.html

IEC. (2015). *IEC 60050-192: Dependability*. https://webstore.iec.ch/publication/21886

IT Operations Analytics—BMC Software. (2020). https://www.bmc.com/it-solutions/it-analytics.html

Jin Huang, & Ling, C. X. (2005). Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, *17*(3), 299–310. https://doi.org/10.1109/TKDE.2005.50

Kochs, H.-D. (2018). *System Dependability Evaluation Including S-dependency and Uncertainty*. Springer International Publishing. https://doi.org/10.1007/978-3-319-64991-7

Kuhn, M., & Johnson, K. (2013). *Applied Predictive Modeling*. Springer New York. https://doi.org/10.1007/978-1-4614-6849-3

Maklin, C. (2019). *AdaBoost Classifier Example In Python*. Towards Data Science. https://towardsdatascience.com/machine-learning-part-17-boosting-algorithms-adaboost-in-python-d00faac6c464

Malhotra, P., Vig, L., Shroff, G., & Agarwal, P. (2015). Long Short Term Memory Networks for Anomaly Detection in Time Series. *Computational Intelligence*, 7.

Paasche, F. (1918). Bemerkninger. *Samtiden*, *29*(8).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Ran, Y., Zhou, X., Lin, P., Wen, Y., & Deng, R. (2019). A Survey of Predictive Maintenance: Systems, Purposes and Approaches. *ArXiv:1912.07383 [Cs, Eess]*. http://arxiv.org/abs/1912.07383

Raschka, S., & Mirjalili, V. (2019). *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2, 3rd Edition*.

Reinkemeyer, L. (Ed.). (2020). *Process Mining in Action: Principles, Use Cases and Outlook*. Springer International Publishing. https://doi.org/10.1007/978-3-030-40172-6

Rodriguez, J. (2017, October 2). *A Different Way to Think About Overfitting and Underfitting in Machine Learning Part I: Capacity*. Medium. https://medium.com/@jrodthoughts/a-different-way-to-think-about-overfitting-and-underfitting-in-machine-learning-part-i-capacity-738aa1bd5498

Roeder, L. (2020). *Lutzroeder/netron* [JavaScript]. https://github.com/lutzroeder/netron (Original work published 2010)

Salfner, F., Lenk, M., & Malek, M. (2010). A survey of online failure prediction methods. *ACM Computing Surveys*, *42*(3), 1–42. https://doi.org/10.1145/1670679.1670680

Shmueli, B. (2020, May 20). *Matthews Correlation Coefficient is The Best Classification Metric You've Never Heard Of*. Medium. https://towardsdatascience.com/the-best-classification-metric-youve-never-heard-of-the-matthews-correlation-coefficient-3bf50a2f3e9a

Singh, A. (2018, June 18). Ensemble Learning. *Analytics Vidhya*. https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/

What is an EHR? (2019). https://www.healthit.gov/faq/what-electronic-health-record-ehr