Ensemble Methods For Detecting Internet Outages

Abhinav Hampiholi
hampiholi@gatech.edu

Siddharth Singh Solanki
siddharth.solanki@gatech.edu

1 ABSTRACT

Internet Outages happen when users experience complete disconnection from the internet. Some reasons for such outages could be power outages, natural disasters or government-mandated outages. Today we have a lot of our essential infrastructure dependent upon internet and these outages can be devastating economically or socially. Monitoring and keeping track of these outages thus becomes an important task. The IODA[2] project of the Internet Intelligence Lab at Georgia Tech monitors internet activity across the world and has an API to return the data of monitoring signals. Our aim is to develop an 'outage detector' which uses these signals and raises alerts when there is an internet outage. Furthermore, we explore how ensembling various different (possibly weak) 'outage detectors' can produce better results than using the output of a single detector.

2 PRELIMINARIES

The IODA data can be accessed at varying different granularities: AS (Autonomous System), Region, and Country. The AS is the finest granularity and we have conducted our study on AS-level data. The API allows us to provide as arguments an Autonomous System Number (ASN), a start time and an end time. When called with these parameters, we obtain the time-series values of all three signals BGP, AP and Darknet (explained below) at 5 minute intervals within the provided window. For example, if the end time is 50 minutes after the start time, the API will return 10 values (or 11 if endpoints are included) each of BGP, AP and Darknet readings. These signals can be effectively used to do different kinds of outage analyses [1] [3] For the sake of concreteness, we shall define some terms.

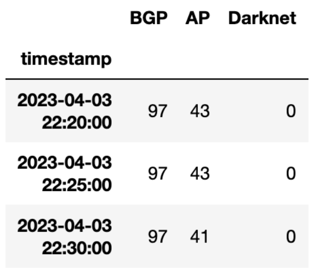
2.1 Definitions

BGP #visible /24s - Using the BGP protocol, network routers advertise the blocks of IP addresses (i.e prefixes) they are responsible for. These announcements get forwarded and spread around the global Internet. IODA collects all of these prefix announcements from BGPstream, which collects announcements using peers.If more than 50% of the peers of all the route collectors can see the same route advertisements, then IODA counts the number of /24 blocks of IP addresses associated with the route announcement.

Active Probing (AP) # /24s up - IODA's backend has a set of IP addresses that it probes to check for connectivity. it sends ping packets to these IPs. If an IP address responds, then the entire /24 is considered to be active. As an example, if an IP address like 129.222.58.192 responds to a ping, then 129.222.58.0/24 (this is a block of the 256 IPs in the same "group") is considered to be active. The raw values of the AP signal are the number of /24 blocks that are considered active.

Darknet - Keeps track of how many requests from this 'region' are 'wrongly' sent to the telescope. This is a measure of how much traffic is coming out of this region but fluctuates noticeably more than the first two signals.

Point - A point refers to the readings of BGP, AP and Darknet values at a particular timestamp. For example, the below figure shows three points from an AS.



timestamp	BGP	AP	Darknet
2023-04-03 22:20:00	97	43	0
2023-04-03 22:25:00	97	43	0
2023-04-03 22:30:00	97	41	0

Figure 1: Three points

Window - A window is a collection of consecutive points. A window of size 10 means 10 points that are consecutive and separated by 5 minutes each. The figure above is a window of size three. And the figure below is a plot of AP values for a window of size 107.

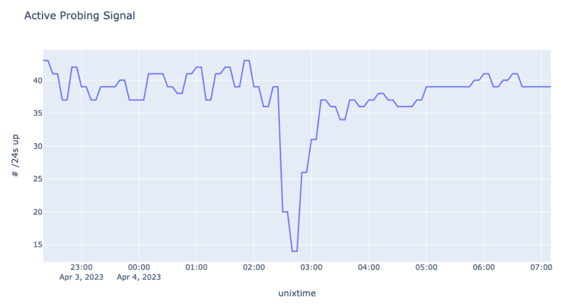


Figure 2: AP plot over a large window

Outlier - This word can be used to describe either a point or a window. A window is an 'outlier' if it contains readings (of BGP/AP/Darknet data) that is unusual compared to neighboring windows. A point is an 'outlier' if the readings of the data at that timestamp are unusual compared to neighboring points. We may refer to outlier windows as 'bad windows' and outlier points as 'bad points'. The figure below shows a bad window and bad points.

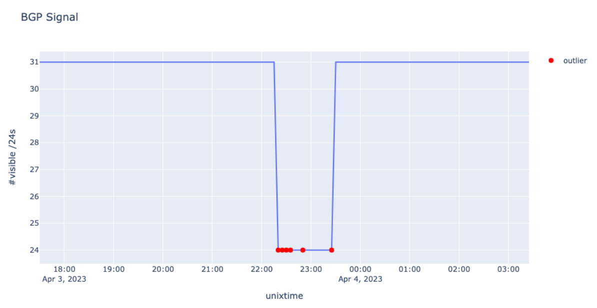
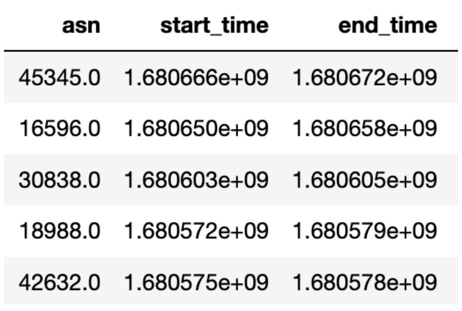


Figure 3: Outliers

3 DATASETS

We obtained readings of IODA data from the publicly available API. We also obtained a list of outages that were confirmed by the IODA team. This list includes ASNs and start and end times of outages and whether or not the IODA auto detection algorithm detected the outage. We used this to make three datasets where a 'dataset' is a collection of (asn, start_time, end_time) tuples which we call 'timeperiods'. Below is an example of a dataset.



	asn	start_time	end_time
	45345.0	1.680666e+09	1.680672e+09
	16596.0	1.680650e+09	1.680658e+09
	30838.0	1.680603e+09	1.680605e+09
	18988.0	1.680572e+09	1.680579e+09
	42632.0	1.680575e+09	1.680578e+09

Figure 4: Example of a dataset

Our three datasets are

(1) **dataset_detected_outages**: A set of 639 timeperiods each including a window (within the corresponding start and end times) that raised the IODA auto alert. In words, this dataset contains examples of instances where there was an outage and IODA's autodetection system detected it and raised an alarm.
(2) **dataset_undetected_outages**: A set of 94 timeperiods each including a window that contained an outage (confirmed manually) but did not raise the IODA auto alert. In words, these are examples of when the IODA detection system failed to detect outages.
(3) **dataset_non_outages**: A set of 1998 timeperiods which do not include an outage. These are examples of windows that should *not* raise an alarm.

4 OUTLIER DETECTORS

An 'outlier detector' is defined as a function that given a timeperiod (asn, start_time, end_time), determines whether or not there is a bad window within that time period (and raises an alarm if there is).
In this paper we use a point-based time-series anomaly detector from the pycaret library to construct an outlier detector. Then we build more 'weaker' outlier detectors and combine the signals produced by the pycaret based detector with those produced by the weaker detectors to obtain better performance than the pycaret detector alone. This section is organized as follows.

(1) We will go over the pycaret library and describe how the pycaret anomaly detector works.
(2) We will describe how we used pycaret to create an outlier detector.
(3) We will evaluate this detector on our three datasets and gain some insights on how its performance could be improved.
(4) We add more 'weak' signals and use this to create a new outlier detector and evaluate it's performance.

4.1 PyCaret

PyCaret is an open-source, low-code machine learning library in Python that automates machine learning workflows. PyCaret is essentially a Python wrapper around several machine learning libraries and frameworks such as scikit-learn, XGBoost, LightGBM, CatBoost, Optuna, Hyperopt, Ray, and few more. We used the library and its algorithms for anomaly detection algorithms. It's Anomaly Detection Module is an unsupervised machine learning module that is used for identifying rare items, events, or observations. It provides over 15 algorithms and several plots to analyze the results of trained models.

4.2 Using Pycaret

As discussed in the above section, pycaret contains a suite of point-based anomaly detectors, after experimentation on the data with different point-based anomaly detection algorithms available in the suite. We found that the Histogram-based Outlier Detection algorithm works best for our use case.
Histogram-based Outlier Detection is a statistical anomaly detection algorithm which calculates an outlier score by creating a univariate histogram for each single feature of the dataset. It assumes that features are independent. The height of each single bin of the histogram represents the density estimation. To ensure an equal weight of each feature, the histograms are normalized in such a way that the maximum height of the bin would be equal to one. Then, calculated values are inverted so that anomalies have a high score and normal instances have a low score.

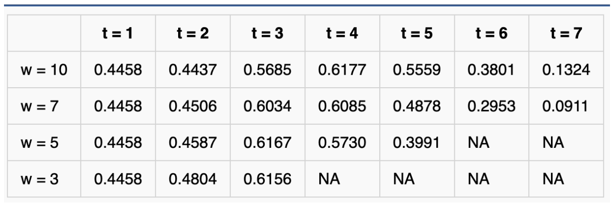
$$OutlierScore(v) = \sum_{i=0}^{d} log(\frac{1}{hist_i(v)})$$

where d is the number of features, v is the vector of features, and $hist_i(v)$ is the density estimation of each feature instance.

Given a window of points (along with their corresponding signal values), the algorithm flags 'bad' points. However, an 'outlier detector' must flag bad windows, not necessarily bad points. Thus, to extend the pycaret algorithm to detect bad windows, we use a simple rule. A window of size y is bad if at least x points in that window are bad. Here we call y the window size and x the 'threshold'.

4.3 Hyperparameter tuning

Now that there is a way to convert a pycaret point outlier detector into an outlier detector as defined in section 4, there are two hyperparameters to be tuned. The window size y and the threshold x. We ran a grid search on a set of reasonable values of x and y. In order to evaluate the performance of the algorithms, we calculated the $F1$ score of the algorithms when run on our three datasets. A 'perfect' algorithm would not raise any alerts for any timeperiod in the non_outages dataset and would raise alerts for *every* timeperiod in the other two datasets. Below is a table of our results.



	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7
w = 10	0.4458	0.4437	0.5685	0.6177	0.5559	0.3801	0.1324
w = 7	0.4458	0.4506	0.6034	0.6085	0.4878	0.2953	0.0911
w = 5	0.4458	0.4587	0.6167	0.5730	0.3991	NA	NA
w = 3	0.4458	0.4804	0.6156	NA	NA	NA	NA

Figure 5: F1 scores

We see that the best results are roughly when the threshold is half the size of the window. However, even with the best combination of values from the table above, the results are as follows:
Non Outages: 294 / 1998
Detected Outages: 398.0 / 639
Undetected Outages: 61.0 / 94
With an F1 score of $\boxed{0.6177}$. This means that almost half the outages are not raising alarms.

4.4 Weaknesses of the algorithm

Let us examine some examples of timeperiods in the outage dataset that didn't raise alarms. Consider figure 6 where we have only plotted the BGP values along with the point outliers according to the pycaret algorithm. Clearly, there is a significant drop in BGP values. However, the algorithm simply doesn't flag the points with a high enough density to trigger the alarm.

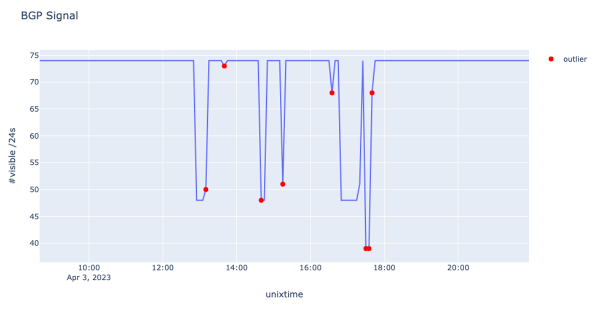


Figure 6: No alarm raised

Consider another example. In figure 7 we have plotted the AP data for an example timeperiod along with the points that the algorithm flags. We see that despite the AP signal falling to zero, the algorithm does not flag enough points to trigger the alarm.

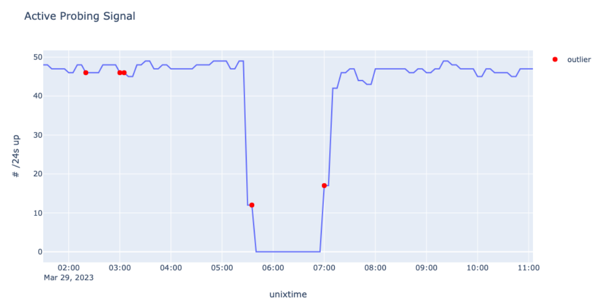


Figure 7: No alarm raised despite AP going to zero

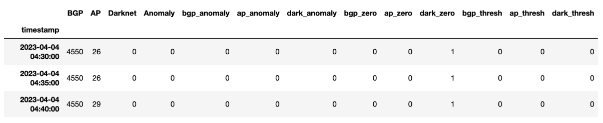
This indicates strongly that there are certain blind spots of the algorithm. We claim that in general any algorithm will have such 'blind spots' and suggest that using an ensemble of multiple algorithms can help mitigate this problem. Intuitively, one algorithm will cover for the blind spot of another.

4.5 Ensemble

We can treat the output of the pycaret algorithm as a binary value for each point. By extension, there can be other binary values generated for each point. For example 'isBGPzero?' can be a value that is 1 if the value of the BGP signal at that point is zero and 0 otherwise.
We constructed a total of 9 such 'weak' signals to be used along with the original pycaret algorithm's signal. Such weak signals were of interest to us as prior work has been done to show that even weak signals could be effective in classification task if they satisfy certain assumptions [4] These signals are:

(1) **isBGPzero**: As discussed earlier, this signal is 1 if the value of the BGP signal at that point is zero and 0 otherwise.
(2) **isAPzero**: Similar to isBGPzero.
(3) **isDarkZero**: Similar to isBGPzero.
(4) **isBGPunderThresh**: this column is 1 if the BGP value at this point is less than 20% of the mean BGP value in the last 10 timesteps . The intuition for this signal is clear. It accounts for significant drops in signals (but not drops to zero).
(5) **isAPunderThresh**: similar to isBGPunderThresh.
(6) **isDarkunderThresh**: similar to isBGPunderThresh.
(7) **bgpAnomaly**: This is a column that is generated by passing only the BGP signal to the pycaret algorithm (as opposed to all three signals in the previous section).
(8) **APanomaly**: similar to bgpAnomaly.
(9) **DarkAnomaly**: similar to bgpAnomaly.

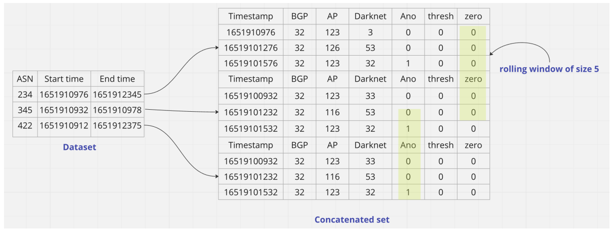
Below (figure 8) is an example of the points with all the columns associated with them filled. To understand the distributions of the signals and how they vary with different datasets, we create 'concatenated sets' for each of the three datasets. For each timeperiod



	BGP	AP	Darknet	Anomaly	bgp_anomaly	ap_anomaly	dark_anomaly	bgp_zero	ap_zero	dark_zero	bgp_thresh	ap_thresh	dark_thresh
timestamp													
2023-04-04 04:30:00	4550	26	0	0	0	0	0	0	0	1	0	0	0
2023-04-04 04:35:00	4550	26	0	0	0	0	0	0	0	1	0	0	0
2023-04-04 04:40:00	4550	29	0	0	0	0	0	0	0	1	0	0	0

Figure 8: Ensemble

in the dataset, we retrieve all of the associated points and then add columns to them to get a filled window. We then vertically stack all of these windows associated with each timeperiod in the dataset. The below figure illustrates how we go from a dataset to a ‘concatenated set’. Next, we sum up values in a rolling window



ASN	Start time	End time
234	1651910976	1651912345
345	1651910932	1651910978
422	1651910912	1651912375

Timestamp	BGP	AP	Darknet	Ano	thresh	zero
1651910976	32	123	3	0	0	0
16519101276	32	126	53	0	0	0
16519101576	32	123	32	1	0	0
Timestamp	BGP	AP	Darknet	Ano	thresh	zero
16519100932	32	123	33	0	0	0
16519101232	32	116	53	0	0	0
16519101532	32	123	32	1	0	0
Timestamp	BGP	AP	Darknet	Ano	thresh	zero
16519100932	32	123	33	0	0	0
16519101232	32	116	53	0	0	0
16519101532	32	123	32	1	0	0

Figure 9: Creating the concatenated set

of size 5 (for instance, the highlighted values in the figure above are added together) and we then plot the values of this rolling sum in a histogram for each signal for each dataset to see if there is a noticeable difference in the distributions.

In fig 10 we see the histogram for the ‘Anomaly signal’ (the one

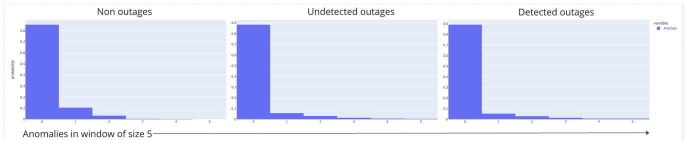


Figure 10: Anomaly set

generated by the pycaret algorithm). While the non-outage dataset in fact has a greater chance of having 2/5 anomalies in a window, the probabilities for 3 anomalies or more in a window of size 5 are negligible for the non-outages set but noticeably higher for the outage sets.

Another observation is summarized in Fig 11. We see that there is a significantly higher probability that a signal dropped to zero in the ‘undetected by IODA’ dataset than in the detected by IODA dataset. This suggests that one of the IODA autoalert system’s blind spots is that it does not respond strongly enough to situations when the signals drop to zero.

4.6 Ensemble Implementations

Now that we have multiple signals for each point, we can build outlier detectors that use all (or a subset of) these signals to decide whether or not they raise an alarm. We designed a simple algorithm to demonstrate the potential of such ensemble methods. Consider the following simple algorithm:

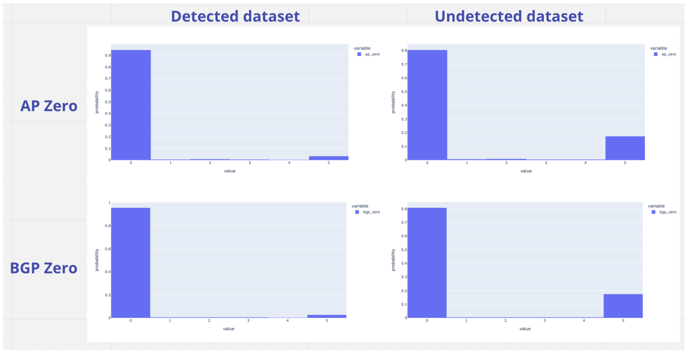


Figure 11: Zero signals

(1) Calculate the rolling sum of the values for a window of size 5. And for each window:
(2) If bgpZero or apZero are ever set to 1, raise an alert
(3) If isBGPunderThresh is ever set to 1, raise an alarm
(4) If isAPunderThresh is set to 1 at least twice in the window (of size 5), raise an alarm.
(5) If the pycaret histogram algorithm detects at least 4 point outliers in the window of size 5, raise the alarm.

The ensemble can be thought of as two parts: one part consisting of the pycaret histogram algorithm and the other consisting of several ‘weak’ signals that simply detect falls in signals. In addition to evaluating the ensemble algorithm, we can evaluate these two parts independently. Concretely, consider the following two algorithms.

Pycaret 4-5 Algorithm: This algorithm raises an alert on a timeperiod iff there is a window of size 5 within which the pycaret histogram algorithm flags at least 4 points.

Weak-signals-only algorithm: This algorithm raises an alert on a timeperiod iff there is a window of size 5 where either:

(1) bgpZero or apZero are set to one at least once.
(2) isBGPunderThresh is set to one at least once.
(3) isAPunderThresh is set to one at least twice.

Consider also the following ‘augmented ioda’ algorithm which is defined as follows:
Augmented-ioda: This algorithm raises an alert on a timeperiod iff either

(1) IODA autoalert is raised. (i.e the existing IODA autodetect algorithm raises the algorithm).
(2) bgpZero or apZero are set to one at least once
(3) isBGPunderThresh is set to one at least once.
(4) isAPunderThresh is set to one at least twice in a window of size 5.

Below is a table summarizing the performance of the algorithms we have seen so far.

	Ensemble	P4-5	Weak-only	ioda	ioda-aug
Non	122	121	1	x	x+1
Detect	468	298	306	639	639
Undetect	67	45	52	0	52
F1 score	0.77	0.57	0.65	y	z

We think this is an interesting result because the combination of p4-5 and weak-only does significantly better than either of them individually. Furthermore, although we don't know how many timeperiods of the non-outage set set off the IODA alarm, we can be sure that the ioda-aug algorithm (which is an ensemble of ioda and the weak-signals algorithm) will have a better F1 score than the ioda autodetect algorithm. We think this is evidence that ensemble methods can be very effective for detecting outliers in the IODA dataset.

5 CONCLUSION

We started our investigation with the aim of developing a method for detecting internet outages for the internet outage data. The data contained three time-varying signals namely BGP, AP and Darknet.We processed the input dataset by parsing through a list of confirmed outages and creating three parts - *dataset_detected_outages*, *dataset_undetected_outages* and *dataset_non_outages*. Upon experimenting on this data with point-based anomaly algorithms, we found that Histogram-based Outlier Detection worked best for the data. The algorithm, however, has some weaknesses, which led us to explore an ensemble method. In the ensemble method, we defined some rules which were naive but still satisfied a better-than-random assumption. The combination of simple rules was reasonably effective in detecting the outages, with an F1 score of 0.77 when used in conjunction with the chosen point-based algorithm.

REFERENCES

[1] K Benson, A Dainotti, k claffy, and E Aben. 2013. Gaining Insight into AS-level Outages through Analysis of Internet Background Radiation. In *Traffic Monitoring and Analysis Workshop (TMA)*.
[2] Internet Outage Detection and Analysis (IODA). [n.d.]. https://ioda.inetintel.cc.gatech.edu/
[3] R Padmanabhan, A Schulman, A Dainotti, D Levin, and N Spring. 2019. How to Find Correlated Internet Failures. In *Passive and Active Measurement Conference (PAM)*.
[4] Renzhi Wu, Shen-En Chen, Jieyu Zhang, and Xu Chu. 2023. Learning Hyper Label Model for Programmatic Weak Supervision. arXiv:2207.13545 [cs.LG]