Georgia Tech ISYE 6748 | Commnet Project 1 | Team 14

Change Detection Algorithms For Noisy Data

Detecting performance deterioration at Commnet cell tower sites

Sean Crow, Ryan Keeney, Haven Penn

1 Abstract

Commnet operates cell phone networks in the rural southwestern United States. The objective is to identify performance degradation amongst unlabeled data with heavy tails, many outliers, and positively-skewed noise. The team simulated the unlabeled data to create labeled data in order to streamline testing. After investigating multiple change detection algorithms on the simulated data, a simple computationally-inexpensive change detection algorithm based on windowed statistics and trained on the simulated data significantly outperformed existing change detection models such as CUSUM, ARIMA, PELT, DBSCAN, Bayes, and SVM at detecting shifts in performance [Table 5]. Scan the QR code to view the demo application or go to url: `https://bit.ly/3uVhDX8`

Figure 1: Application QR Code

2 Introduction

Commnet operates cell networks in the rural southwestern United States and provides voice and data services to customers. The goal of this project is to develop algorithms to accurately detect poor performance at Commnet's cell sites. Specifically, each of Commnet's cell sites produces 31 data points, or key performance indicators, on an hourly basis. The key performance indicators include throughput, data volume, channel quality, connection success rates, and drop rates among others. When detecting changes to the key performance indicators, support staff will be notified to investigate the issue. It is thus important to detect change when there is a problem, while limiting the amount of false positives that take the attention of staff.

3 Background

Commnet's current approach requires daily manual and visual inspection of the data using a Ploty dashboard that pulls data from a SQLite server. This results in significant time usage for operators. The data is noisy, large, and contains a great deal of variability. Each cell site provides daily and hourly readings. The data consists of 31 key performance indicators, a timestamp, and tower indicators. The data includes 594 unique towers over 180 days. Among the key performance indicators, the company sponsor indicated that two indicators of importance include the connection drop percentage due to radio link problems and the phone drop percentage due to core connection problems.

Prior algorithms have had limited success, with a high rates of false positive flags. The approach automates the detection of cell site performance outliers with an algorithm, establishes a robust and flexible reporting tool, and provides a method to analyze proposed algorithms with simulated data.

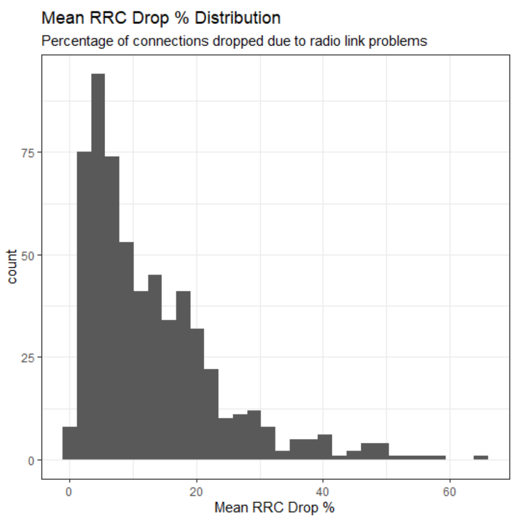
3.1 Exploratory Data Analysis

Data are provided for daily and hourly readings from each cell site. The data consist of 31 key performance indicators, a timestamp, and a tower indicator. It includes 594 tower IDs and 180 days of data. Per the project sponsor, the most important key performance indicators are RRC_Drop_Pct and FDD_ERAB_Drop_Pct. A lower RRC Drop percentage is better, and more than 2 percent is bad. A significant portion of the data is above 2 percent.

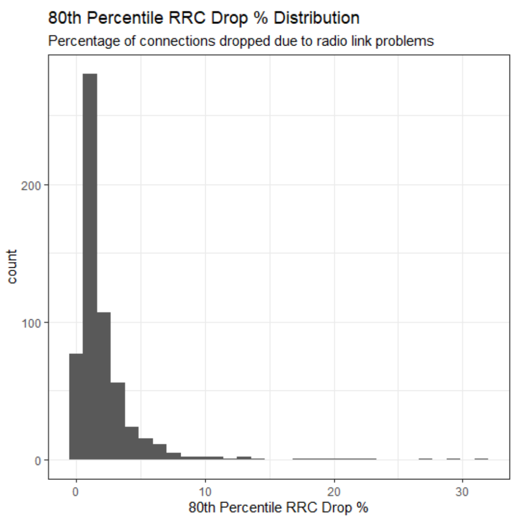
Performance can vary a great deal from tower to tower. KPIs with high variance included RRC_DROP_PCT (percentage of dropped calls due to radio link), MEAN_RRC_CONN_SET UP_TIME_MS (how long to set up connection), and AVG_PRACH_USAGE_PER_SECOND (average number of devices trying to connect per second).

The data are sensitive to outliers and contain considerable noise. The team explored noise reduction and standardization techniques because the noise nor tower baseline performance is normally distributed or centered about a common mode. The team found that a Box-Cox transformation with a $\lambda = 0.05$ will normalize the data, a shift of $+1$ is suggested to account for zeros in the data.

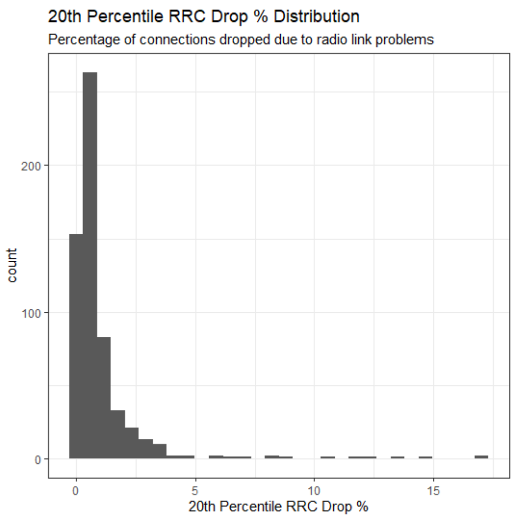
Due to the noise in the data, an effective smoothing technique is beneficial. Multiple smoothing techniques were examined, including exponential smoothing, moving average, moving median, transformed moving average, transformed moving median. The team focused on backward-looking algorithms as they reduce computational requirements.



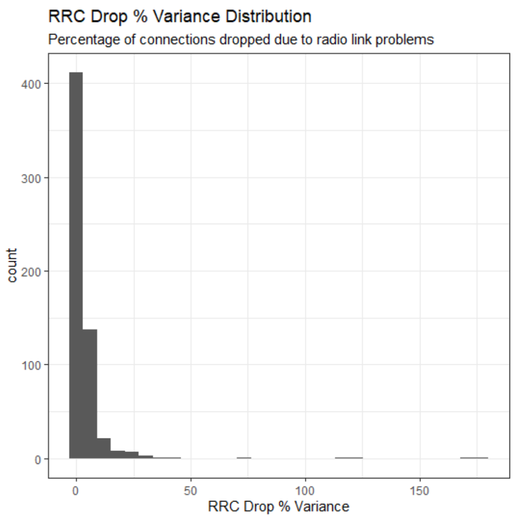
(a) The data is dissimilar, with 25 percent of the data exceeding the sponsor threshold (tower mean).



(b) The data are sensitive to outliers and contain considerable noise.



(c) A third of the data has an 80th percentile above the sponsor suggested threshold of 2 percent.



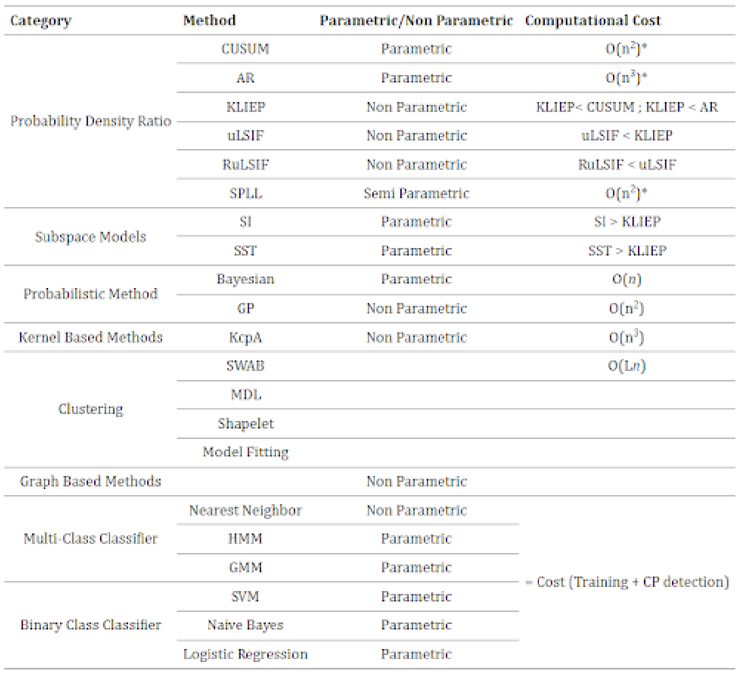
(d) 1-in-10 of the towers have a 20th percentile that is above the sponsor suggested threshold of 2 percent.

Figure 2: Exploratory data analysis observations

3.2 Literature Review

Blending change point detection with concept drift detection is a challenging problem, because many real-world data are non-stationary and multi-dimensional. The team focused on multiple flexible flag rules to identify multiple scenarios.

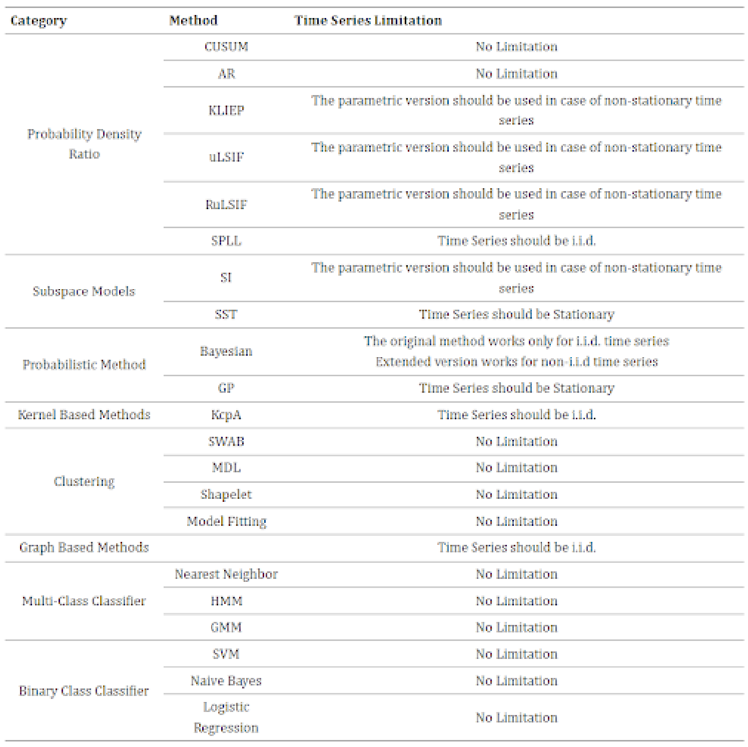
Literature review provided an overview of commonplace change point algorithms, serving as a starting point for closer examination and implementation on the Commnet data set by the team. [AC16] The literature review defines model scalability, computational costs, advantages, disadvantages, and limitations. Summaries of this information are shown in the figures below.



Category	Method	Parametric/Non Parametric	Computational Cost
Probability Density Ratio	CUSUM	Parametric	$O(n^2)$*
	AR	Parametric	$O(n^3)$*
	KLIEP	Non Parametric	KLIEP< CUSUM ; KLIEP < AR
	uLSIF	Non Parametric	uLSIF < KLIEP
	RuLSIF	Non Parametric	RuLSIF < uLSIF
	SPLL	Semi Parametric	$O(n^2)$*
Subspace Models	SI	Parametric	SI > KLIEP
	SST	Parametric	SST > KLIEP
Probabilistic Method	Bayesian	Parametric	$O(n)$
	GP	Non Parametric	$O(n^2)$
Kernel Based Methods	KcpA	Non Parametric	$O(n^3)$
Clustering	SWAB		$O(Ln)$
	MDL		
	Shapelet		
	Model Fitting		
Graph Based Methods		Non Parametric	
Multi-Class Classifier	Nearest Neighbor	Non Parametric	= Cost (Training + CP detection)
	HMM	Parametric	
	GMM	Parametric	
Binary Class Classifier	SVM	Parametric	
	Naive Bayes	Parametric	
	Logistic Regression	Parametric	

Figure 3: Comparison of algorithm scalability based on sliding window size, n.

Change detection techniques are commonplace across a variety of problems today, including medicine, climate change, and speech recognition, among others. After a thorough review of literature and current implementations, the change detection techniques of a median window approach, cumulative sum (CUSUM), auto-regressive integrated moving average (ARIMA), support vector machine (SVM), Bayesian change detection, Pruned Exact Linear Time (PELT) and density-based spatial clustering of applications with noise (DBSCAN) were implemented on the Commnet data. The team designed a workflow that automates the daily collection of data and generates flags for the operators. While most models were taught in the OMSA curriculum, PELT and DBSCAN were not part of the curriculum. These models were researched, with additional background information contained in section four. An additional model, based on simulated data and control plot analysis was examined due to limitations of existing change detection algorithms in detecting level-shifts [Tsa88]. Models



Category	Method	Time Series Limitation
Probability Density Ratio	CUSUM	No Limitation
	AR	No Limitation
	KLIEP	The parametric version should be used in case of non-stationary time series
	uLSIF	The parametric version should be used in case of non-stationary time series
	RuLSIF	The parametric version should be used in case of non-stationary time series
	SPLL	Time Series should be i.i.d.
Subspace Models	SI	The parametric version should be used in case of non-stationary time series
	SST	Time Series should be Stationary
Probabilistic Method	Bayesian	The original method works only for i.i.d. time series Extended version works for non-i.i.d time series
	GP	Time Series should be Stationary
Kernel Based Methods	KcpA	Time Series should be i.i.d.
Clustering	SWAB	No Limitation
	MDL	No Limitation
	Shapelet	No Limitation
	Model Fitting	No Limitation
Graph Based Methods		Time Series should be i.i.d.
Multi-Class Classifier	Nearest Neighbor	No Limitation
	HMM	No Limitation
	GMM	No Limitation
Binary Class Classifier	SVM	No Limitation
	Naive Bayes	No Limitation
	Logistic Regression	No Limitation

Figure 4: Comparison of algorithm limitations

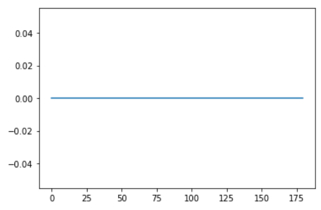
were selected based on computational costs, existing body of work and applicability to the data set. The data, flags, flag rules, and algorithms can be inspected and adjusted through a web application.

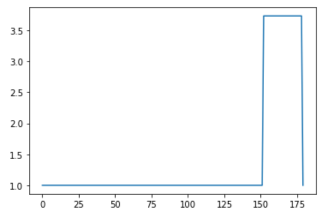
4 Methodology

4.1 Data Generation

Because the original provided data set was unlabeled, rather than manually label individual towers, the team decided to simulate the data to generate a labeled data set. For simplicity, the team focused on the two drop percentage performance indicators. These two performance indicators have a correlation of 0.7301, so the team wanted to preserve that correlation in the generated data. This, however, did not end up being relevant, as the team never used both metrics simultaneously in any methods.

In the simulated data, the project goal was to create two data sets. The first (**Regular**) emulated the noise and baselines of the given data. The baseline was defined as the approximate true drop percentage (e.g. with the absence of noise). The second (**Noisy**) has significantly higher baselines and an artificially large amount of noise with the goal of testing the limits of


(a) A flat example


(b) A sharp example

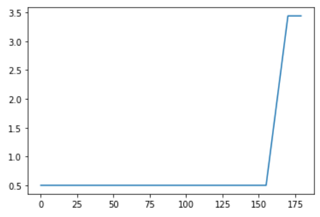
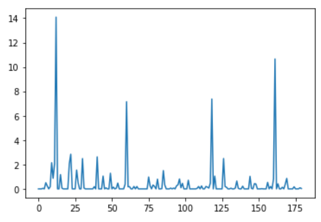
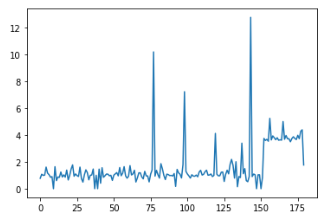

(c) A slope example

Figure 5: Examples of the three scenarios in the generated baselines


(a) A flat example


(b) A sharp example

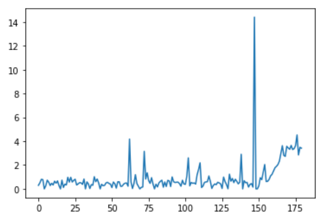

(c) A slope example

Figure 6: Examples of the three scenarios in the generated data with noise

algorithms. In other words, the noisy data set tests to see if an algorithm can perform well on the most difficult towers.

In addition, for each generated data set, three different scenarios were accounted for: Flat, sharp, and slope. This is visually demonstrated without noise in Figure 5 and with noise in Figure 6.

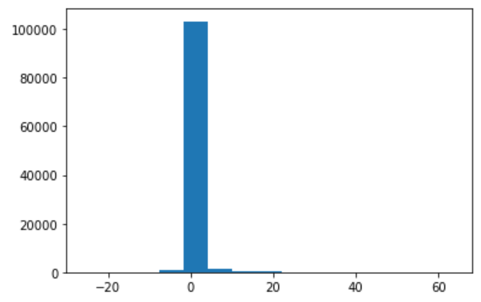
- **Flat:** These are generated towers with no anomalies, and the baseline remains flat throughout the entire time series

- **Sharp:** These are generated towers with a sudden increase in the drop percentage in the span of one day, which results in an anomaly.

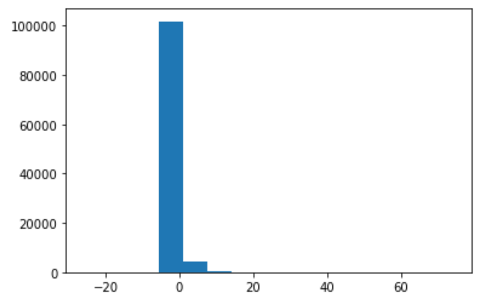
- **Slope:** These are generated towers with a gradual increase in the drop percentage over the span of several days, which results in an anomaly. This determines how quickly the algorithm can detect a tower that is gradually losing performance.

4.1.1 Noise and baseline estimation

For every tower, the data were split in half and the median was taken as the baseline. This was due to the observation that towers with an anomaly have a temporary increase in the baseline, which would skew the noise positively if not taken into account. A split in the data helps to isolate noise from increases or decreases in the baseline. After the baselines were determined, the differences between the estimated baseline and the drop percentages were calculated. This is the **noise** in the data. The distribution of the noise is shown in Figure 7. The noise is clustered around zero, but it is skewed to the right with sparse outliers.



(a) The distribution of noise for RRC_Drop_Pct



(b) The distribution of noise for FDD_ERAB_Drop_Pct

Figure 7: Noise distributions for the provided data

4.1.2 Baseline generation

Randomly generated tower baselines were created. Some were given increases in the drop percentage indicating a problem. For simplicity, the baselines for both types of drop percentage are equal.

For the regular generated data, these baselines were created by approximating the baseline distribution in the real data, which is seen in Table 1.

Drop Percentage	0	0.5	1	1.5	2	2.5	3	4
Proportion	0.1	0.3	0.2	0.1	0.1	0.1	0.05	0.05

Table 1: Baseline proportion in Regular data

For the noisy generated data, these baselines are inflated to larger values. While the sponsor guideline is that a 2% drop percentage is a bad sign, some towers in the real data have approximated baselines well-above this value. The noisy data, therefore, has inflated baselines to ensure that the algorithms do not automatically flag a tower based on a high percentage alone.

Drop Percentage	5	6	8	10	12	15	20	25
Proportion	0.1	0.3	0.2	0.1	0.1	0.1	0.05	0.05

Table 2: Baseline proportion in Noisy data

4.1.3 Noise addition

For the regular data, for every point in the time series, a random index was chosen. Then, the noise corresponding to the chosen index for both was added to the base.

For the noisy data, the approach with the least effect on the data mean was a scaling approach. To avoid unrealistically large spikes, noise indexes in the RRC_Drop_Pct with a value over 3% were filtered out. Then, all remaining noise was multiplied by a factor of 5, and the remaining indexes and corresponding noise were randomly chosen and added.

4.1.4 Correlation

The correlation between FDD_ERAB_Drop_Pct and RRC_Drop_Pct remained <0.10 off in the generated data compared to the provided data. While this could be improved, because the selected methods did rely on both KPIs simultaneously, this was not seen as an issue.

Data set	Correlation
Provided Data	0.730
Regular Data	0.799
Noisy Data	0.685

Table 3: Correlation between FDD_ERAB_Drop_Pct and RRC_Drop_Pct

4.2 Weather Data

As part of noise reduction, the team decided to investigate the impact of weather on the drop percentages for the towers. The goal of this investigation was to create a model that could map weather conditions to noise in the data. This could then be used to subtract the noise from the data to time series closer to the baseline values. Ultimately, the correlation coefficient of the results was too weak to have an impact, but the methodology and results are detailed here for possible future exploration.

Provided with the latitude and longitude of each tower, the team used the Meteostat API [Lam] for weather data, which is an open-source API for weather data. It takes a given latitude and longitude pair and returns weather data for the nearest weather station. This gives both hourly and daily data. One limitation, however, is that there is missing data for some of the stations. This data includes measurements such as:

- Precipitation (daily and hourly)
- Humidity (hourly)
- Minimum, maximum, and average daily temperature
- Wind direction and speed (daily and hourly)

The team then built three linear regression models using variations of the RRC_Drop_Pct as the response variable. While the team observed correlation for both daily and hourly data, models were only built for daily weather data, as the selected change-detection models are all based on daily data. The predictors were chosen via the Forward-Backward-Stepwise method, and Box-Cox transformations were used as necessary. The different response variables are as follows:

- **Daily RRC_Drop_Pct**: A model that uses the original daily Drop Percentage data as the response variable
- **Log of Daily RRC_Drop_Pct**: A model that uses the log of the daily Drop Percentage data as the response variable at the recommendation of the Box-Cox transformation
- **Daily Noise**: A model that uses the noise of the RRC Drop Percentage data as a response variable. The noise is derived from the same method as the Data Generation method.
- **Daily Normalized Standard Deviation Change**: A model that uses the standard deviation change of the RRC Drop Percentage data as a response variable. The noise is derived an early iteration of the median-shift method.
- **Log of Daily Normalized Standard Deviation Change**: A model that uses the log of the standard deviation change of the RRC Drop Percentage data as a response variable at the recommendation of the Box-Cox transformation. The noise is derived an early iteration of the median-shift method.
- **Daily Smoothed RRC_Drop_Pct**: A model that uses drop percentage data after exponential smoothing at the response variable.
- **Log of Daily Smoothed RRC_Drop_Pct**: A model that uses drop percentage data after exponential smoothing at the response variable at the recommendation of the Box-Cox transformation.

4.3 Change Detection Algorithms

4.3.1 Median-Shift

A simulated data set was created using the distributions of noise that appears in the real data and simulated anomalies. This data was used to train a change detection algorithm (random forest) using basic statistics calculated over 5-day and 30-day windows as features. This method is loosely inspired by control plots and limitations in existing change detection algorithms [Tsa88]. It is similar, but not based on the mean-shift methods [Arc] as the tool utilizes training data, is online, and does not assume the noise is normally distributed. The simulated data was split into test and train subsets with a equal proportion of anomalies [R C13; Wic+19]. Model features and a relatively high flag threshold (suggested 0.90) were chosen to reduce false-positives and balance the true-positive rate and true-negative rate. After model training, evaluation, and selection were performed, the model was applied to the data provided by Commnet. Alternative model types (e.g., logistic regression) and features (e.g., alternative statistics, smoothing algorithms, and window periods) were explored and are discussed in the analysis section.

Selected model features (5-day vs. 30-day)

- Median difference (absolute)
- Median difference (standardized)
- Standard deviation ratio

Method advantages

- Model performance: The model significantly outperforms alternative change detection models in accuracy, recall, and precision.
- Computational effort: By using a windowed periods, the computational requirement is low. New computation is limited and does not require independent models for each tower.

Method disadvantages

- Responsiveness: The model intentionally has a 2-4 day lag before it will flag anomalies. This is done to reduce false-positives as requested by Commnet. *The 3rd feature, standard deviation ratio, may be removed to increase responsiveness at the expense of a slight increase false-positives in towers with highly cyclical pattern such as 128-1.*
- Break-in time: The model requires data to be collected for a minimum of 5-days, with a suggested break-in time of 30-days, before it is able to predict flags.

4.3.2 CUSUM

The cumulative sum (CUSUM) algorithm is a widely used change detection model that was implemented on the Commnet cell tower performance data. This implementation used the first 30 days of data, calculating the tower average of the metric. This was used as the basis of comparison when calculating CUSUM scores for day 31 onward. Based on testing results and inspection of the data, a CUSUM score treshold of 4 was used for the RRC_Drop_Pct and FDD_ERAB_Drop_Pct metrics.

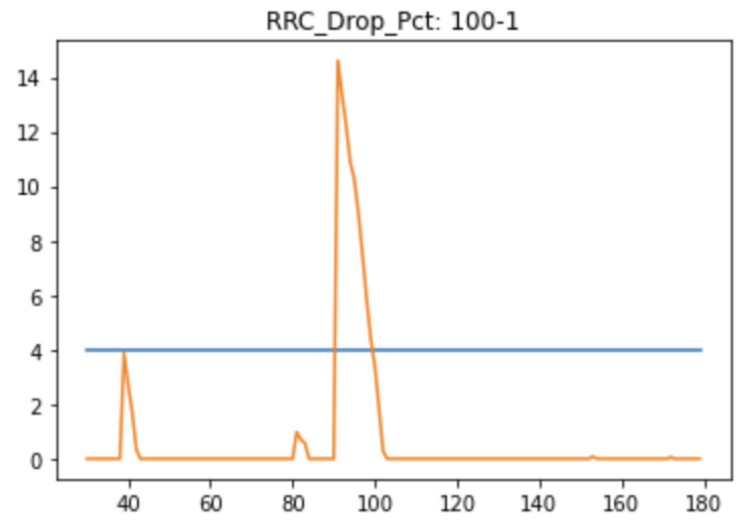


Figure 8: An example CUSUM run for tower 100-1

Selected model features

- Change Score Threshold: set to 4 for RRC_Drop_Pct and FDD_ERAB_Drop_Pct
- 30 day average for mean calculation

Method advantages

- Ease of interpretation: CUSUM is an easy-to-understand option for change detection.
- Presentation of results: CUSUM charts allow for a user-friendly display of change detection results.
- Computational effort and flexibility: The computational requirement is low and the algorithm itself is very flexible, allowing the user to modify parameters with ease for different metrics.

Method disadvantages

- Model parameters need to be configured for each metric.
- Break-in time: The model requires data to be collected for a minimum of 30 days to use as the baseline mean, before it is able to predict flags.
- Model sensitivity: The model is sensitive and has a tendency to flag false positives in comparison to the median shift method.

4.3.3 PELT

Pruned Exact Linear Time (PELT) is a model that was not taught in the Georgia Tech OMSA curriculum, but was researched and implemented as a change detection technique. PELT is designed for larger data sets that could have multiple changepoints. The algorithm minimizes cost functions and offers efficiency and speed advantages [al12].

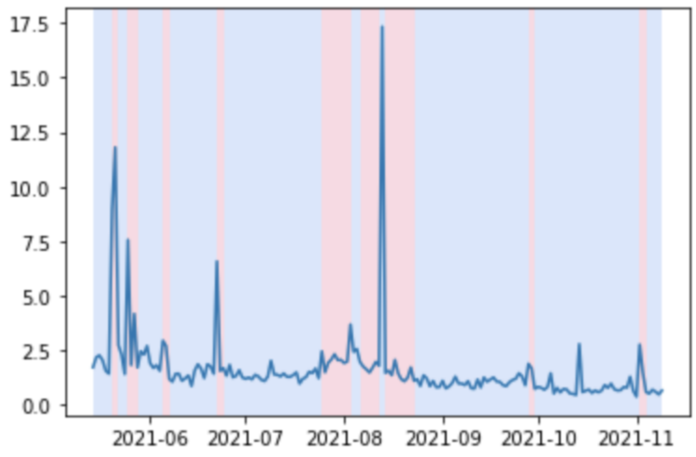


Figure 9: An example PELT run for tower 100-1, with changepoints identified in pink

Selected model features

- Cost class instance: L2
- Minimum length of segments between change points: 1
- Jump between data points: 1

Method advantages

- Computational efficiency: PELT is lauded for its efficiency and speed in comparison to similar methods.
- Performs well on large data sets with multiple changepoints
- Model assumptions and conditions are not considered restrictive

Method disadvantages

- If using smaller amounts of data, such as hourly in one day, may not perform as well
- Model sensitivity: The model is sensitive and has a tendency to flag false positives in comparison to the median shift method.

4.3.4 DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is another model that was not taught in the Georgia Tech OMSA curriculum, but was researched and implemented as a change detection technique. The algorithm identifies clusters of data points that are close together and outliers are identified as points in low-density areas. These outliers are flagged as changes for staff to look into. DBSCAN finds core samples of high density and expands clusters from them. The algorithm is considered effective for data which contains clusters of similar density. [Est+96]

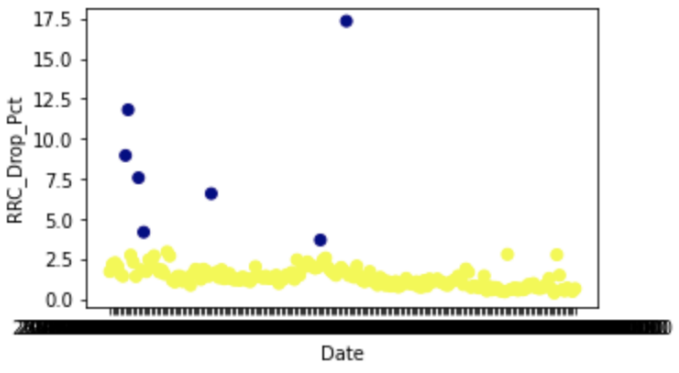


Figure 10: An example DBSCAN run for tower 100-1, with clusters identified in blue (change detected) and yellow (no change)

Selected model features

- Epsilon: 0.2 (identified from testing for chosen metrics)
- Number of samples (or total weight) in a neighborhood for a point to be considered as a core point: 4

Method advantages

- Computational efficiency and memory: DBSCAN is also lauded for its efficiency and speed in comparison to similar methods.
- Can identify data patterns that are not easily detected visually

Method disadvantages

- Epsilon has to be determined for each metric
- Model sensitivity: The model is sensitive and has a tendency to flag false positives in comparison to the median shift method.

4.3.5 Bayes

A Bayesian changepoint analysis was also deployed on the data. This implementation is an offline approach. [Fea06]

Selected model features

- Prior probability: Constant
- Likelihood: Gaussian

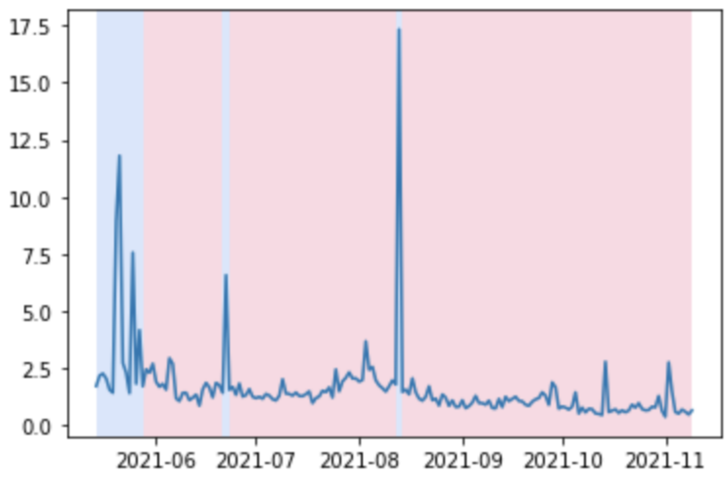


Figure 11: An example Bayes run for tower 100-1, with changepoints identified by a change in color (red/blue)

- Threshold probability: 0.5

Method advantages

- Does not require large amounts of data

Method disadvantages

- Makes assumptions about prior distribution
- Model sensitivity: The model is sensitive and has a tendency to flag false positives in comparison to the median shift method.
- Offline approach requires entire data set to be available

4.3.6 ARIMA

An ARIMA approach was also taken on the data. This model was partially built from scratch and flagged anomalies as data points falling outside of a 99-percent confidence interval. This model was unsuccessful due to a narrow confidence interval that led to many data points being flagged as anomalies. This model was quickly ruled out.

Selected model features

- P (order of AR term): 2
- D (difference factor): 1
- Q (order of MA term): 2

Method advantages

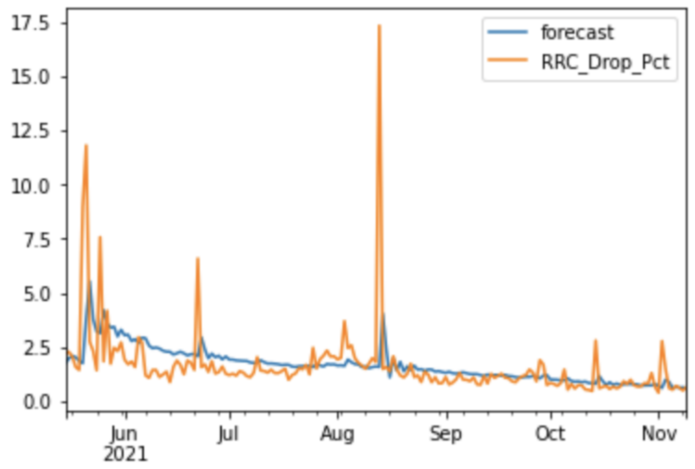


Figure 12: An example ARIMA run for tower 100-1, with the ARIMA forecast plotted alongside the metric

- Takes seasonality into consideration
- Addresses noise in the data

Method disadvantages

- Requires configuration of model features and assumptions
- Model sensitivity: The model was extremely sensitive and had a tendency to flag false positives in comparison to the median shift method.

4.3.7 SVM

The team used the One-class SVM algorithm as a anomaly detection method on the data. By training the model on tower data without anomalies, the goal was to be able to detect anomalies by predicting with the trained model. In testing, the team trained with the first half of the generated data and predicted using the last three-quarters of the generated data. This method was implemented using the Python package scikit-learn [Ped+11].

Selected model features

- Kernel: RBF

Method advantages

- The model is unsupervised, which makes it easy to apply to individual towers
- The computation cost is relatively low

Method disadvantages

- SVM is parametric, so its distribution assumptions may not line up with the data.

- Because the method looks for outliers, a sequence of anomalies may not register. Additionally, gradual tower degradation may not register as well.

4.4 Model Evaluation

To evaluate the models, the team looked at the following three metrics: accuracy, precision, and recall. Out of the three, precision is the most valued, as the goal is to reduce the amount of false positives so that Commnet engineers are not overloaded with false alarms. Additionally, a delay in flagging (false negative) was stated to be a lesser concern than a false alarm (false positive), so the team was not as concerned with recall alone, though it was included to ensure that a model's false negative performance is still reasonable.

In addition, testing was conducted using both the simulated noisy data and the regular data, as these were labeled. Two testing filters were applied: All Flags and First Flags. All Flag evaluates all rows in the data. First Flags excludes all flags where a flag occurred on the previous day. This is because most anomalies occur over a span of multiple days, so the team avoided testing instances where the same anomaly is flagged repeatedly over several days. An example of First Flag is provided in Table 4.

Day	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9
Flag	0	1	0	1	1	0	1	1	1

Table 4: A demonstration of the First Flag filter. Note that Days 5, 8, and 9 are excluded because they are subsequent flags, likely for the same anomaly.

The code for creating a confusion matrix, as part of the calculations, was used from a SciPy package. [Vir+20]

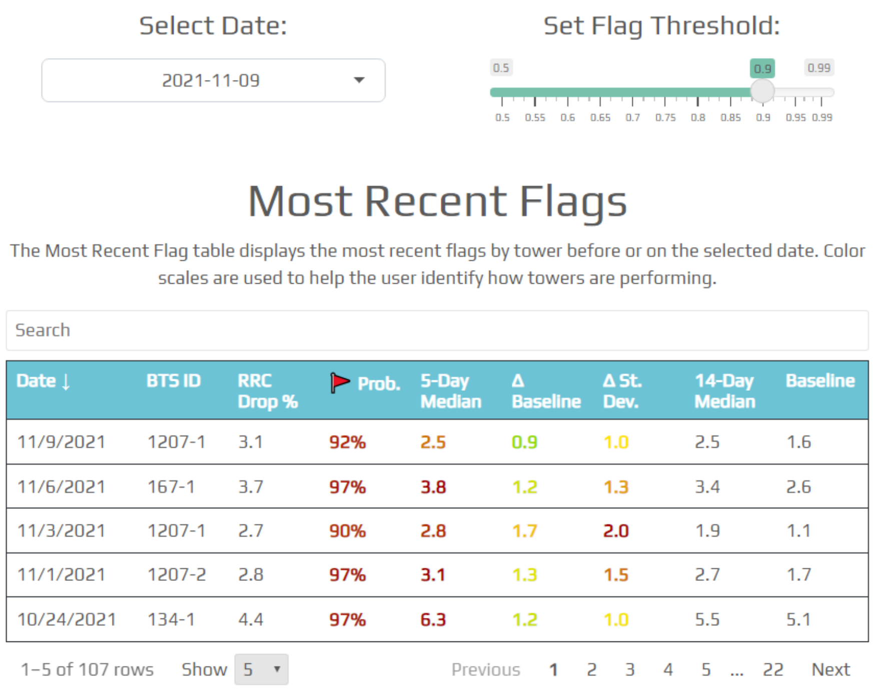
4.5 Web Application

A supporting demo application is available at: `https://bit.ly/3uVhDX8`.

A demo model was developed to demonstrate the team's change detection model on a simulated and real data in R Shiny [R C13]. The data, flags, flag rules, and algorithm can be inspected and adjusted through a web application. It monitors the performance of the cell tower sites and and provides the probability of a required intervention (flag). Users can inspect the most recent flags, daily performance, and review tower performance over time by selecting the date, changing the flag threshold, and examining additional anomaly models. Users can also review the performance of the change detection model using the simulated plot and table tools.

4.5.1 Overview Page

The overview page enables the user to inspect the most recent flags which occurred before, or on, the selected date at the selected flag threshold as well as reviewing the performance for every tower on the selected date. Users can adjust the sensitivity of the change detection algorithm by changing the flag threshold.



Select Date: 2021-11-09

Set Flag Threshold: 0.9

Most Recent Flags

The Most Recent Flag table displays the most recent flags by tower before or on the selected date. Color scales are used to help the user identify how towers are performing.

Search

Date ↓	BTS ID	RRC Drop %	Prob.	5-Day Median	Δ Baseline	Δ St. Dev.	14-Day Median	Baseline
11/9/2021	1207-1	3.1	92%	2.5	0.9	1.0	2.5	1.6
11/6/2021	167-1	3.7	97%	3.8	1.2	1.3	3.4	2.6
11/3/2021	1207-1	2.7	90%	2.8	1.7	2.0	1.9	1.1
11/1/2021	1207-2	2.8	97%	3.1	1.3	1.5	2.7	1.7
10/24/2021	134-1	4.4	97%	6.3	1.2	1.0	5.5	5.1

1–5 of 107 rows   Show 5   Previous 1 2 3 4 5 … 22 Next

Figure 13: A screenshot of the Most Recent Flag Table within the application.

Additionally, users can inspect which towers are flagged for the highest percentage of time and how many times a flag was triggered at the selected flag threshold.

4.5.2 Commnet Tower Plot Page

The By Tower (Commnet) page plots the performance of the tower over time and displays key metrics such as flags, flag probability, and the deviation of performance from the expected baseline. Users can select a tower, set a custom flag threshold, and choose to add in an additional anomaly detection model.

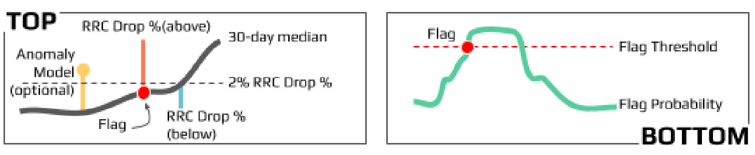


Figure 14: The historical performance, anomalies, and flags are easily reviewed on the plot page.

4.5.3 Simulated Tower Plot Page

The By Tower (Simulation) page plots the performance of the simulated towers over time, as well as displaying key metrics such as flags, flag probability, and the deviation of performance from the expected baseline. Users can select a tower and set a custom flag threshold. The Model Performance On Simulated Data Table provides the user with examples of how the model performs on various anomaly and noise scenarios.

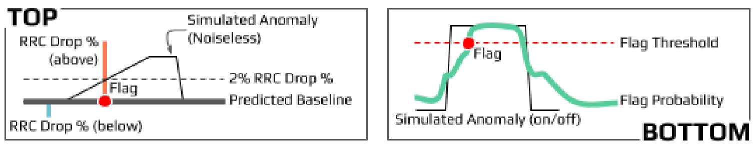


Figure 15: The simulated data can be used to visually review the change detection algorithm performance in various scenarios.

4.5.4 Suggested Production Implementation

Currently, this application uses local data (a live data link was not provided by Commnet). The team provided this tool as a lightweight example which can be adapted for use in Commnet's workflow and automated reporting. The team suggests that a rules database is implemented to edit the sensitivity and alarm settings of high-baseline-high-variance towers such as 128-1. Additionally, the team suggests implementing a reset threshold or cool-down after a flag has occurred.

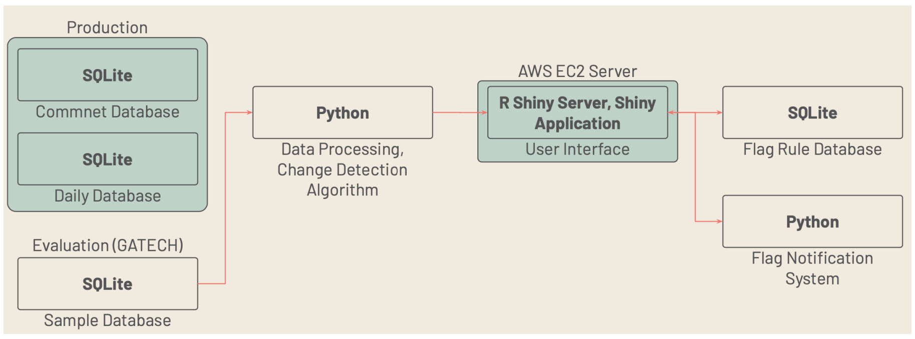


Figure 16: Suggested implementation for an automated reporting dashboard.

5 Analysis and Findings

5.1 Change Detection Algorithms

The median-shift model correctly had a significantly lower incident rate of false positives than the alternative methods explored. The CUSUM and SVM models both correctly identified a higher proportion of true anomalies in the population, but at the expense of many additional false positives. This can be identified from the precision metrics of each tower. Methods such as PELT, DBSCAN, and Bayes did not have acceptable results to merit recommendation. Tables with the results for each method are recorded in the appendix.

Model	Accuracy	Precision	Recall	Discussion
Median-Shift	0.983	0.895	0.308	Single model, requires training data, computationally inexpensive model, trade-off between precision and 2-4 day lag
CUSUM	0.936	0.124	0.105	Flexible and easy to interpret, but requires configuration for each KPI, accuracy highly dependent on first flag versus all flags
PELT^a	0.809	0.099	0.089	Overly sensitive, leading to a high occurrence of false positives
DBSCAN^a	0.895	0.122	0.042	Identifies clusters that are not easily detected visually, but overly sensitive and requires configuration for each KPI
Bayes^a	0.886	0.052	0.002	Offline model, poor precision
SVM^a	0.810	0.080	0.190	Low-cost computation and has relatively good recall but poor precision. Not able to detect anomalies well, even with exponential smoothing.

Table 5: Summary of the change detection algorithms on the simulated data (first flag generation, threshold = 0.5), ^{a} exponential smoothing

5.1.1 Median-Shift

Multiple versions of the median-shift model were evaluated. Feature set, model type, and threshold were all evaluated [6]. The team recommended the smaller feature set (2-windows vs. 4-windows) due to computational requirements, model simplicity, and relative performance. Non-parametric models, such as a random forest, or models that allow for interaction between features outperformed linear combinations of features. Since Commnet has a strong preference against false positives, the team suggests a starting threshold of 0.90.

The team found that a features and windows could be selected so that the model would ignore, or detect, different types of changes. For example, including a standard deviation ratio feature reduced the model's sensitivity to serially correlated samples with high variances such as tower 128-1. If detecting this type of scenario is deemed worth the trade off, this feature could can removed and the overall performance of the methodology does not significantly decrease based on the team's evaluations.

Model	Threshold	Accuracy	TP Rate	TN Rate
Random Forest (1-window)	90%	0.915	0.957	0.913
Random Forest (3-windows)	95%	0.914	0.970	0.912
Random Forest (1-window)	95%	0.907	0.973	0.906
Logistic Regression (3-windows)	95%	0.899	0.849	0.899
Logistic Regression (1-window)	95%	0.898	0.810	0.899

Table 6: Median-shift model performance on withheld test data.

Visual inspection of the real (unlabeled) data provided by Commnet confirmed that the algorithm correctly identified changes-worth-flagging exceeded a rate of 8-in-10 instances (50 sampled flags). Since the data is trained on simulated data without serial correlation, the difference in performance when compared to the simulated data was regarded as acceptable. For example, the algorithm correctly identified two instances where an review should have been conducted on tower 167-1 in figure 17.

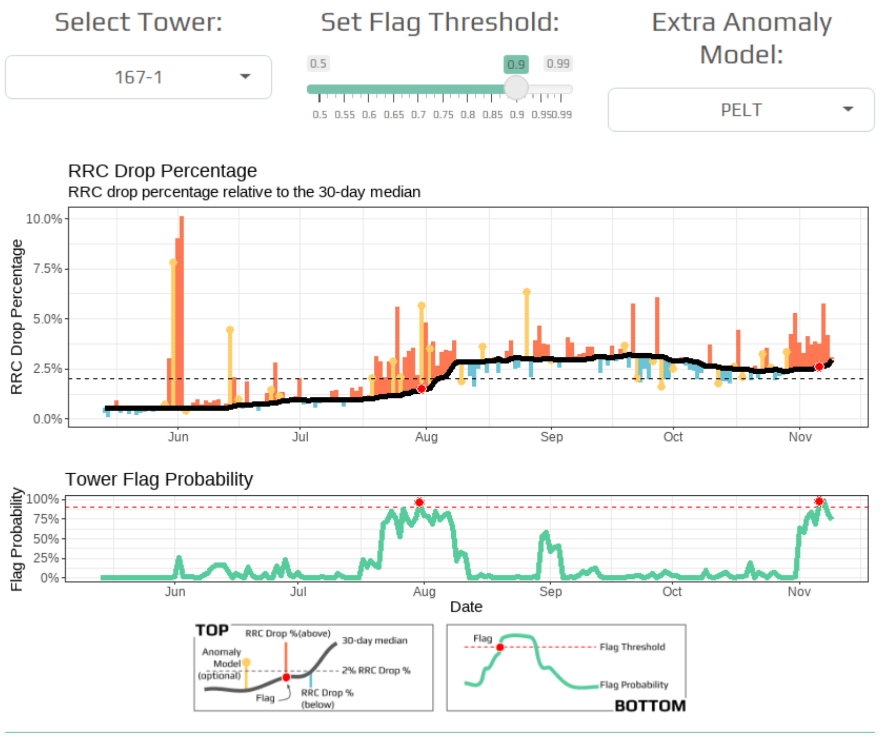


Figure 17: Identifying performance changes with the median-shift algorithm in tower 167-1 .

5.2 Weather

After looking at both correlation and linear regression models, weather data was determined to not have a significant impact on noise to be used in the modeling process. The R^2 values for each of the attempted models are displayed in Table 7.

In addition, as shown in Tables 8a and 8b, the individual factors for daily and hourly weather data have a very weak correlation with RRC_Drop_Pct. This is independent of date and tower, though such results suggest that too much of the variability in drop percentage data comes from other sources for weather data to be useful. While there was some correlation with weather condition code (-0.22), because the code is essentially qualitative data factorized, it is not directly useful.

5.3 Seasonality

The team investigated but did not pursue seasonality adjustments by week, month, or holidays. Some seasonality did seem to exist as shown by the 0.4% improvement in the tower population performance that occurs by mid-August. Additionally, some sites did demonstrate significant differences between weekdays and the weekend. Additional analysis and recommendations are discussed in the conclusion.

Response Variable	R^2 Adjusted
RRC_Drop_Pct	0.023
Log of RRC_Drop_Pct	0.015
Noise	0.005
Normalized Standard Deviation Change	0.005
Log of Normalized Standard Deviation Change	0.002
Smoothed RRC_Drop_Pct	0.023
Log of Smoothed RRC_Drop_Pct	0.029

Table 7: Adjusted R^2 for linear regression models concerning RRC_Drop_Pct vs Daily Weather

	RRC_Drop_Pct
Temperature Average	0.020
Temperate Minimum	-0.018
Temperature Maximum	0.046
Precipitation	-0.003
Wind direction	-0.060
Wind Speed	0.071
Pressure	-0.124

(a) Correlation between Daily Weather and RRC_Drop_Pct

	RRC_Drop_Pct
Temperature	0.006
Dew Point	0.005
Relative Humidity	0.002
Precipitation	-0.011
Wind Direction	-0.0001
Wind Speed	0.021
Pressure	-0.065
Weather Condition Code	-0.225

(b) Correlation between Hourly Weather and RRC_Drop_Pct

Table 8: Weather data correlation

5.4 Noise Reduction

Since the data has a significant positive skew, exponential smoothing without a transformation and moving average techniques were not as effective as median or percentile-based methods. Windows of various length were examined with 5-day and 6-day windows performing the best.

Smoothing Method	Direction	R^2
Friedman's SuperSmoother (transformed)	Forward-Backward	0.99
Median (5-day window)	Backward	0.98
Mean (5-day window, transformed)	Backward	0.96
Exponential Smoothing (0.05 level)	Backward	0.46

Table 9: A smoothing algorithm using a 5 -day median had acceptable performance

6 Conclusions

6.1 Change Detection Algorithms

Overall, the team's research demonstrates that models trained on the simulated data are effective at detecting changes on a wide degree of tower characteristics including typical operating range, variance, serial correlation, and anomaly type and scale. Additional analysis, reflection, and recommendations for model design are included in the analysis and findings section.

6.2 Model Evaluation

The current evaluation method utilizes the entire simulated data. The team realizes this could lead to information leakage between training data and testing data. To ensure the suggested model was not over-fit, it was evaluated independently using appropriate controls and focused on simple models and limited feature sets to reduce the risk of over-fitting.

For future iterations of the model development and evaluation, the team suggests that a sub-sample of simulated towers reserved for evaluation are withheld from the training data in their entirety. This method will ensure that there is not data leakage between the training and testing data, as well as ensure that there is not leakage due correlation in the time series data. This approach will enable accurate evaluation for a wide degree of algorithms.

6.3 Seasonality

As discussed in section 5.3, the team investigated but did not pursue seasonality adjustments by week, month, or holidays. The team only noted these observations for future investigation and did not factor them in the analysis due to the following reasons:

- **Limited data**: 5+ years of data would be required to accurately predict seasonality impact.
- **Variable seasonality**: Seasonality performance may be driven by dates that are not consistent year-to-year, such as holidays and ski resort openings.
- **Data limitations**: The team also investigated usage features to predict changes in usage. However, Commnet informed students that the majority of the data is consumed

by stationary site users and the majority of connections are generated by mobile cell users. Currently, Commnet does not have the capability to identify the distribution of these types of users.

6.4 Future Work

6.4.1 Model Development

The team's research demonstrates that a simple model trained on the simulated data presents a promising solution. The simulated data, basic model, and evaluation code provides a framework for Comment to continue to improve their processes.

6.4.2 Usage Study

The team investigated usage features to predict changes in traffic volume on the network with the assumption that changes in mobile volume would impact the performance statistics. However, Commnet informed the team that the majority of the data is consumed by stationary site users and the majority of unstable connections are due to mobile users. Currently, Commnet does not have the capability to distinguish between these types of users.

6.4.3 Correlation Study

Certain towers, such as 128-1, demonstrate cyclical serial correlation that does not fit a seasonality profile. The team suggests that Commnet investigate these anomalies as, if controlled, they would improve the performance of the monitoring tools.

6.4.4 Application Update

As previously mentioned, the mock application uses local data, as a live data link was not provided by Commnet. The team provided this tool as a lightweight example which can be adapted for use in Commnet's workflow and automated reporting. The team suggests that Commnet tailor rules and flags to their preferences.

References

[AC16] S. Aminikhanghahi and DJ Cook. *A Survey of Methods for Time Series Change Point Detection*. 2016, pp. 339–367. DOI: 10.1007/s10115-016-0987-z.. URL: https://pubmed.ncbi.nlm.nih.gov/28603327/.

[al12] Killick et al. *Optimal Detection of Changepoints With a Linear Computational Cost*. Informa UK Limited, 2012, pp. 1590–1598.

[Arc] ArcGIS. *How change point detection works*. URL: https://pro.arcgis.com/en/pro-app/latest/tool-reference/space-time-pattern-mining/how-change-point-detection-works.

[Est+96] M. Ester et al. *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*. Portland, OR: AAAI Press, 1996, pp. 226–231.

[Fea06] Paul Fearnhead. *Exact and Efficient Bayesian Inference for Multiple Changepoint problems*. 2006, pp. 203–213.

[Lam] Christian Lamprecht. *Meteostat*. URL: https://meteostat.net/en/.

[Ped+11] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[R C13] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: http://www.R-project.org/.

[Tsa88] Ruey S. Tsay. “Outliers, Level Shifts, and Variance Changes in Time Series”. In: *Journal of Forecasting* 7 (1988), pp. 1–20.

[Vir+20] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[Wic+19] Hadley Wickham et al. “Welcome to the tidyverse”. In: *Journal of Open Source Software* 4.43 (2019), p. 1686. DOI: 10.21105/joss.01686.

7 Appendix

7.1 Model Evaluation

type	tp	fp	tn	fn	accuracy	precision	recall
overall	198	2874	93177	11751	0.865	0.064	0.017
flat	0	1026	34974	0	0.972	0.0	
sharp	79	1026	31393	3502	0.874	0.071	0.022
slope	119	822	26810	8249	0.748	0.126	0.014

Table 10: All flag generation with Bayes on regular data. No exponential smoothing.

type	tp	fp	tn	fn	accuracy	precision	recall
overall	10683	44416	33635	1266	0.492	0.194	0.894
flat	0	16467	13533	0	0.4511	0.0	
sharp	3378	15558	10861	203	0.475	0.178	0.943
slope	7305	12391	9241	1063	0.552	0.371	0.873

Table 11: All flag generation with CUSUM on regular data. No exponential smoothing.

type	tp	fp	tn	fn	accuracy	precision	recall
overall	1226	4067	91984	10723	0.863	0.232	0.103
flat	0	1795	34205	0	0.950	0.0	
sharp	830	1376	31043	2751	0.885	0.376	0.232
slope	396	896	26736	7972	0.754	0.307	0.047

Table 12: All flag generation with DBSCAN on regular data. No exponential smoothing.

type	tp	fp	tn	fn	accuracy	precision	recall
overall	7860	1335	94716	4089	0.950	0.855	0.658
flat	0	178	35822	0	0.995	0.0	
sharp	2902	584	31835	679	0.965	0.832	0.810
slope	4958	573	27059	3410	0.889	0.896	0.592

Table 13: All flag generation with Median Shift on regular data. No exponential smoothing.

type	tp	fp	tn	fn	accuracy	precision	recall
overall	1066	9705	86346	10883	0.809	0.099	0.089
flat	0	3548	32452	0	0.901	0.0	
sharp	321	3304	29115	3260	0.818	0.089	0.090
slope	745	2853	24779	7623	0.709	0.207	0.090

Table 14: All flag generation with PELT on regular data. No exponential smoothing.

type	tp	fp	tn	fn	accuracy	precision	recall
overall	10187	41578	27473	1762	0.465	0.197	0.853
flat	0	15786	11214	0	0.415	0.0	
sharp	3106	14281	9138	475	0.453	0.179	0.867
slope	7081	11511	7121	1287	0.526	0.381	0.846

Table 15: All flag generation with SVM on regular data. Exponential smoothing applied.

type	tp	fp	tn	fn	accuracy	precision	recall
overall	198	2874	93177	11751	0.865	0.064	0.017
flat	0	1026	34974	0	0.972	0.0	
sharp	79	1026	31393	3502	0.874	0.071	0.022
slope	119	822	26810	8249	0.748	0.126	0.014

Table 16: First flag generation with Bayes with regular data. No exponential smoothing.

type	tp	fp	tn	fn	accuracy	precision	recall
overall	148	1048	33635	1266	0.936	0.124	0.105
flat	0	429	13533	0	0.969	0.0	
sharp	64	320	10861	203	0.954	0.167	0.240
slope	84	299	9241	1063	0.873	0.219	0.073

Table 17: First flag generation with CUSUM with regular data. No exponential smoothing.

type	tp	fp	tn	fn	accuracy	precision	recall
overall	435	3599	91984	10723	0.866	0.108	0.039
flat	0	1559	34205	0	0.956	0.0	
sharp	165	1177	31043	2751	0.888	0.123	0.057
slope	270	863	26736	7972	0.753	0.238	0.033

Table 18: First flag generation with DBSCAN with regular data. No exponential smoothing.

type	tp	fp	tn	fn	accuracy	precision	recall
overall	427	243	94716	4089	0.956	0.637	0.095
flat	0	92	35822	0	0.997	0.0	
sharp	196	82	31835	679	0.977	0.705	0.224
slope	231	69	27059	3410	0.887	0.770	0.063

Table 19: First flag generation with Median Shift with regular data. No exponential smoothing.

type	tp	fp	tn	fn	accuracy	precision	recall
overall	1066	9705	86346	10883	0.809	0.099	0.089
flat	0	3548	32452	0	0.901	0.0	
sharp	321	3304	29115	3260	0.818	0.089	0.090
slope	745	2853	24779	7623	0.709	0.207	0.089

Table 20: First flag generation with PELT with regular data. No exponential smoothing.

type	tp	fp	tn	fn	accuracy	precision	recall
overall	414	4781	27473	1762	0.810	0.080	0.190
flat	0	1909	11214	0	0.855	0.0	
sharp	172	1604	9138	475	0.817	0.097	0.266
slope	242	1268	7121	1287	0.742	0.160	0.158

Table 21: First flag generation with SVM with regular data. Exponential smoothing applied.

type	tp	fp	tn	fn	accuracy	precision	recall
overall	3577	2964	93087	8372	0.895	0.547	0.299
flat	0	1073	34927	0	0.970	0	
sharp	3358	1087	31332	223	0.964	0.755	0.938
slope	219	804	26828	8149	0.751	0.214	0.026

Table 22: All flags for DBSCAN on regular data. Exponential smoothing applied.

type	tp	fp	tn	fn	accuracy	precision	recall
overall	19	348	95703	11930	0.886	0.052	0.002
flat	0	124	35876	0	0.997	0	
sharp	8	119	32300	3573	0.897	0.063	0.002
slope	11	105	27527	8357	0.765	0.095	0.001

Table 23: All flags for Bayes on regular data. Exponential smoothing applied.

type	tp	fp	tn	fn	accuracy	precision	recall
overall	1693	12680	83371	10256	0.788	0.118	0.142
flat	0	4649	31351	0	0.871	0	
sharp	487	4304	28115	3094	0.795	0.102	0.136
slope	1206	3727	23905	7162	0.698	0.244	0.144

Table 24: All flags for PELT on regular data with exponential smoothing

type	tp	fp	tn	fn	accuracy	precision	recall
overall	5773	13709	82334	6184	0.816	0.296	0.483
flat	0	4732	31268	0	0.869	0	
sharp	3533	4398	27859	210	0.872	0.446	0.944
slope	2240	4579	23207	5974	0.707	0.328	0.273

Table 25: All flags for DBSCAN on noisy data with exponential smoothing

type	tp	fp	tn	fn	accuracy	precision	recall
overall	1	56	95987	11956	0.889	0.018	0.000
flat	0	4	35996	0	0.999	0.000	
sharp	1	20	32237	3742	0.896	0.048	0.000
slope	0	32	27754	8214	0.771	0.000	0.000

Table 26: All flags for Bayes on noisy data with exponential smoothing

type	tp	fp	tn	fn	accuracy	precision	recall
overall	4671	36622	59421	7286	0.593	0.113	0.391
flat	0	13738	22262	0	0.618	0.000	
sharp	1464	12282	19975	2279	0.596	0.107	0.391
slope	3207	10602	17184	5007	0.566	0.232	0.390

Table 27: All flags for PELT on noisy data with exponential smoothing

type	tp	fp	tn	fn	accuracy	precision	recall
overall	663	78	95668	1490	0.984	0.895	0.308
flat	0	20	35979	0	0.999	0	
sharp	219	34	32233	417	0.986	0.866	0.344
slope	444	24	27456	1073	0.962	0.949	0.293

Table 28: First flags for Median Shift on regular data with exponential smoothing

type	tp	fp	tn	fn	accuracy	precision	recall
overall	369	2654	93087	8372	0.894	0.122	0.042
flat	0	1008	34927	0	0.971	0	
sharp	183	865	31332	223	0.967	0.175	0.451
slope	186	781	26828	8149	0.752	0.192	0.022

Table 29: First flags for DBSCAN on regular data with exponential smoothing

type	tp	fp	tn	fn	accuracy	precision	recall
overall	19	348	95703	11930	0.886	0.052	0.002
flat	0	124	35876	0	0.997	0	
sharp	8	119	32300	3573	0.897	0.063	0.002
slope	11	105	27527	8357	0.765	0.095	0.001

Table 30: First flags for Bayes on regular data with exponential smoothing

type	tp	fp	tn	fn	accuracy	precision	recall
overall	1693	12680	83371	10256	0.788	0.118	0.141
flat	0	4649	31351	0	0.871	0	
sharp	487	4304	28115	3094	0.795	0.102	0.136
slope	1206	3727	23905	7162	0.698	0.244	0.144

Table 31: First flags for PELT on regular data with exponential smoothing

type	tp	fp	tn	fn	accuracy	precision	recall
overall	1542	444	95443	3630	0.960	0.776	0.298
flat	0	184	35788	0	0.995	0	
sharp	437	154	32029	923	0.968	0.739	0.321
slope	1105	106	27626	2707	0.911	0.912	0.290

Table 32: First flags for Median-shift on noisy data with exponential smoothing

type	tp	fp	tn	fn	accuracy	precision	recall
overall	1405	10050	82334	6184	0.838	0.123	0.185
flat	0	3487	31268	0	0.900	0	
sharp	205	3492	27859	210	0.883	0.055	0.494
slope	1200	3071	23207	5974	0.730	0.281	0.167

Table 33: First flags for DBSCAN on noisy data with exponential smoothing

type	tp	fp	tn	fn	accuracy	precision	recall
overall	1	56	95987	11956	0.889	0.018	0
flat	0	4	35996	0	0.999	0	
sharp	1	20	32237	3742	0.896	0.048	0
slope	0	32	27754	8214	0.770944	0	0

Table 34: First flags for Bayes on noisy data with exponential smoothing

type	tp	fp	tn	fn	accuracy	precision	recall
overall	4671	36622	59421	7286	0.593	0.113	0.391
flat	0	13738	22262	0	0.618	0	
sharp	1464	12282	19975	2279	0.596	0.106	0.391
slope	3207	10602	17184	5007	0.566	0.232	0.390

Table 35: First flags for PELT on noisy data with exponential smoothing