CAIT-UTC-REG 43

Artificial Intelligence-Aided Rail Transit Infrastructure Data Mining

FINAL REPORT
March, 2022

Submitted by:

Xiang Liu, Ph.D.
Associate Professor
Rutgers, The State University
Department of Civil & Envir. Eng.
500 Bartholomew Road
Piscataway, NJ, 08854

Junyan Dai
Graduate Research Assistant
Rutgers, The State University
Department of Civil & Envir. Eng.
500 Bartholomew Road
Piscataway, NJ 08854

External Project Manager
David Kraft
Sr. Director, Enterprise Asset Management Program Administration
Metropolitan Transportation Authority

In cooperation with

Rutgers, The State University of New Jersey
And
U.S. Department of Transportation
Federal Highway Administration

Disclaimer Statement

1. Report No. CAIT-UTC-REG 43	2. Government Accession No.	3. Recipient’s Catalog No.

4. Title and Subtitle Artificial Intelligence-Aided Rail Transit Infrastructure Data Mining	5. Report Date MARCH, 2022
	6. Performing Organization Code CAIT/Rutgers University
7. Author(s) Xiang Liu https://orcid.org/0000-0002-4348-7432 Junyan Dai https://orcid.org/0000-0003-2647-4178	8. Performing Organization Report No. CAIT-UTC-REG 43
9. Performing Organization Name and Address Center for Advanced Infrastructure and Transportation Rutgers, The State University of New Jersey 100 Brett Road, Piscataway, NJ 08854	10. Work Unit No.
	11. Contract or Grant No. 69A3551847102
12. Sponsoring Agency Name and Address Center for Advanced Infrastructure and Transportation Rutgers, The State University of New Jersey 100 Brett Road, Piscataway, NJ 08854	13. Type of Report and Period Covered Final Report 02/01/2021-03/31/2022
	14. Sponsoring Agency Code

15. Supplementary Notes
U.S. Department of Transportation/OST-R
1200 New Jersey Avenue, SE
Washington, DC 20590-0001

16. Abstract
Signals are an important part of the urban rail transit system. Signals being in functioning condition is key to rail transit safety. Predicting rail transit signal failures ahead of time has significant benefits with regard to operating safety and efficiency. This paper proposes a machine learning method for predicting urban rail transit signal failures one month in advance, based on records of past failures and maintenance events, as well as other information (e.g., location and weather). Because signal failure is a relatively rare event, imbalanced data mining techniques are used to address its prediction. A case study based on data provided by a major rail transit agency in the United States is developed to illustrate the application of the proposed machine learning method. The results show that our model can be used to identify approximately one third of signal failures one month ahead of time by focusing on 10% of locations on the network. This method can be used by rail transit agencies as a risk screening and ranking tool to identify high-risk hot spots for prioritized inspection and maintenance, given limited resources.

17. Key Words Urban Rail Transit, Signal, Machine Learning, Imbalanced Data Mining	18. Distribution Statement

19. Security Classification (of this report) Unclassified	20. Security Classification (of this page) Unclassified	21. No. of Pages Total #25	22. Price

Form DOT F 1700.7 (8-69)

Acknowledgments
This project was financially supported by the Center for Advanced Infrastructure and Transportation (CAIT), a University Transportation Center (UTC) supported by USDOT-OST-R. We are grateful to the anonymous rail agency for providing their infrastructure asset data for this research.

Contents

1. Introduction

Urban rail transit is an important, safe, efficient, and environmentally friendly mode of passenger transport. Signal lights, as an important part of the rail transit system, can convey specific instructions to drivers to ensure the safe operations of trains. However, damage or failure of the signal often results in unpredictable delays and even safety issues (note that due to the "fail-safe" design of rail signals, a failure will typically lead to the "stop" indication, resulting in service delays). For example, in New York City, "the percent of morning rush hours scrambled by subway signal problems declined from 92% in 2018 to 78% in 2019 but is back up to 80% for the first half of 2021 and a very troubling 88% for the second quarter, April through June 2021", according to Riders Alliance (Riders Alliance, 2022). Since many signals were installed in the early 20th century, aging equipment was recognized as one of the major causes of the transit crisis in New York City in 2017 (Fitzsimmons, 2017). To give another example, in the Bay Area Rapid Transit (BART) system, in-service signal failures account for 50% of infrastructure-related delays that result in slowed service for about 400 hours per year (Wiedmann, 2021). In addition, the simultaneous malfunctioning of multiple devices will lead to heavy maintenance tasks and significant economic losses due to transit shutdowns. By predicting rail transit signals that are prone to failure, one can move toward predictive asset management, achieving a balance between safety, efficiency, and economy.

The widely used machine learning algorithm XGBoost (eXtreme Gradient Boosting) is a scalable implementation of the tree boosting algorithm. It is a state-of-art machine learning method which has good applications in many areas. Take Kaggle competition as an example: XGBoost is the machine learning method that appears most frequently in the winning solutions (Chen & Guestrin, 2016). In this report, we develop an XGBoost-based model to predict rail transit signal failures for the following month (prediction of failure one month in advance).

Rail transit companies have recorded large amounts of event-based data, such as maintenance logs of signal equipment. Event-based data consists of a set of events and a set of participating entities. Event-based datasets are very common, including email traffic, telephone calls, and research publications (O'Madadhain et al., 2005).

Because it is impractical to install sensory devices on every single piece of signal equipment to collect real-time equipment condition data, there is a practical value to the recorded event data being able to predict signal failure by location. For this type of prediction, one particular challenge is dealing with the rarity of failure events. There were only a small portion of signals reported to have failures in the study period, while most other signals operated normally. In the context of classification in the machine learning field, this poses difficulties since many machine learning algorithms used in classification prediction models are designed based on the assumption that the class distribution is equal or slightly imbalanced (Fernández et al., 2018). For rare event prediction, imbalanced data mining (IDM) techniques can be used (Chawla et al., 2002). For IDM, resampling is a widely used statistical technique, in which the class distribution of the training data is changed by either increasing the minority data sample or removing the majority data sample. A number of resampling techniques were proposed and validated in previous studies (Chawla et al., 2002; He et al., 2008; Menardi & Torelli, 2014). However, no resampling method can guarantee superior performance over others (Provost, 2000). Therefore, we conduct a comparative experiment for various resampling methods, including random oversampling, Synthetic Minority Oversampling Technique (SMOTE), ADAptive SYNthetic sampling approach (ADASYN), and random undersampling.

The remainder of this report is organized as follows. Firstly, Chapter 2 discusses related works and identifies knowledge gaps in the existing literature. Chapter 3 introduces the dataset utilized in our experiments. Then, the proposed approach for failure prediction of rail transit signals is presented in the Chapter 4. In Chapter 5, we test the effectiveness of various approaches in addressing imbalanced data and discuss the model performance in comparison with the empirical data. Finally, our conclusions as well as future work are elaborated in the last chapter.

2. Literature Review and Knowledge Gaps

2.1. Literature Review

The literature includes many prior studies which have been conducted on rail transit signal systems. Tu et al. proposed a Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) method for evaluating safety degrees of transit signal systems according to engineering practices and a questionnaire survey (Tu et al., 2011). Zhang et al. presented a new risk assessment method called Fuzzy-FMECA (Failure Mode, Effects and Criticality Analysis) for railway signal systems (Zhang et al., 2013). Ren et al. studied the application of cloud computing technology in rail transit signal systems using the Monte Carlo method for safety and reliability analysis (Ren et al., 2020). The above studies focus on the design of the signal system rather than the safe operation of each signal.

XGBoost was originally developed from the Classification And Regression Trees (CART) algorithm (Breiman et al., 1984). Then, in 1996, Freund and Schapire proposed AdaBoost which combines many relative weak trees to create a highly accurate classifier (Freund & Schapire, 1996). Friedman et al., in 2000, interpreted the boosting as an additive logistic regression model (i.e., AdaBoost) which aims to minimize exponential error (J. Friedman et al., 2000), and soon proposed a gradient boosting machine to address the general supervised problem by combining boosting and CART (J. H. Friedman, 2001). Recently, Chen et al. proposed a scalable and efficient implementation of the gradient boosting machine, which has been widely accepted and used to address many machine learning challenges (Chen & Guestrin, 2016).

The issue of data imbalance has also been studied in much of the literature. Krawczyk discussed major challenges for developing a method to treat imbalanced data (Krawczyk, 2016). He mentioned that imbalanced data can be tackled from the data level and the algorithm level. On the data level, many resampling methods have been proposed to modify the distribution of data samples and have been validated as effective in the field of classification problem. Ling and Li proposed the random oversampling method to duplicate the existing minority samples (Ling & Li, 1998). SMOTE, proposed by Chawla et al. and ADASYN, proposed by He et al. both create new samples for minority class instead of duplicating the existing ones (Chawla et al., 2002; He et al., 2008). There is also significant literature on the effects of random undersampling (Hasanin & Khoshgoftaar, 2018; Prusa et al., 2015). On the algorithm level, one prevalent approach is to use appropriate evaluation metrics. Swets proposed a new measure of model performance that can better reflect the degree of accuracy for binary classification for imbalanced data (Swets, 1988).

2.2 Knowledge Gaps

Although many previous studies have developed various approaches for analyzing safety degrees of rail transit signal systems, very few have studied the failure prognosis for each single signal unit. In addition, few studies have used event-based data for rail transit signal data analysis. It is also challenging to develop a machine learning based model for imbalanced data in such a rare event data analysis. Since a resampling method that is well-validated on one dataset may not have the same effect on others, it is also intriguing to apply various resampling methods comparatively to the signal failure event dataset. This knowledge gap has motivated the development of this research, which aims to develop a machine learning based approach to predict the failure of each signal, using event data from maintenance records.

3. Data

The signal equipment registry and the trouble call history from a major rail transit agency in the United States and the weather condition data from NOAA's National Centers for Environmental Information are utilized to confirm the validity of our proposed failure prognosis method. The signal equipment registry dataset contains primary information pertaining to the signals when they were registered (e.g., Class, Division, Subdivision, Line). The trouble calls dataset records the signals' failure histories and corresponding maintenance work (e.g., Date Reported, Problem Code, Cause Code, Action Code) from May 2018 to June 2021. The weather data includes average temperature and total precipitation for each month from May 2019 to June 2021. Table 3.1 displays a detailed list of variables gathered from the three datasets as well as their descriptions. To reduce the model complexity while keeping the effectiveness of the predictive model, multiple records from the signal equipment registry and the trouble call history datasets, where various pieces of signal equipment work jointly at the same location, are combined into one observation. For example, a failure that occurred on the insulted joint of a signal on May 1, 2021 and a failure that occurred on the signal head of the same signal on May 15, 2021 will be counted as 2 failures of the signal in May 2021. A total number of 18,623 observations are collected and used in our experiments.

Table 3.1. Variables Retrieved from the Datasets

Variables	Descriptions
Division	Historic Operating Company (Division) where the work is to be performed
Subdivision	Subdivision where the signal is located
Line	Line where the signal is located
Date Reported	Date and time that the Transit employee reported the trouble
Type	Type of work being performed (e.g., Corrective, Preventive, Capital)
Problem Code	The reported issue – the symptom observed
Failure Code	Which equipment failed and its malfunction
Cause Code	What caused the signal failure
Action Code	Step taken to resolve the failure
Temperature	Monthly average temperature in degrees Fahrenheit
Precipitation	Monthly total precipitation in millimeters

4. Methodologies

We define an event-based dataset containing a set of events (i.e., failures and corresponding maintenance actions) $E = \{e_1, e_2, e_3, \ldots, e_m\}$ and a set of entities (i.e., signal units) $V = \{v_1, v_2, v_3, \ldots, v_n\}$. Every event has a timestamp t_i. For example, the trouble calls dataset mentioned in the above section records a specific date t_i for a maintenance activity. Figure 4.1 demonstrates the framework of our proposed method.

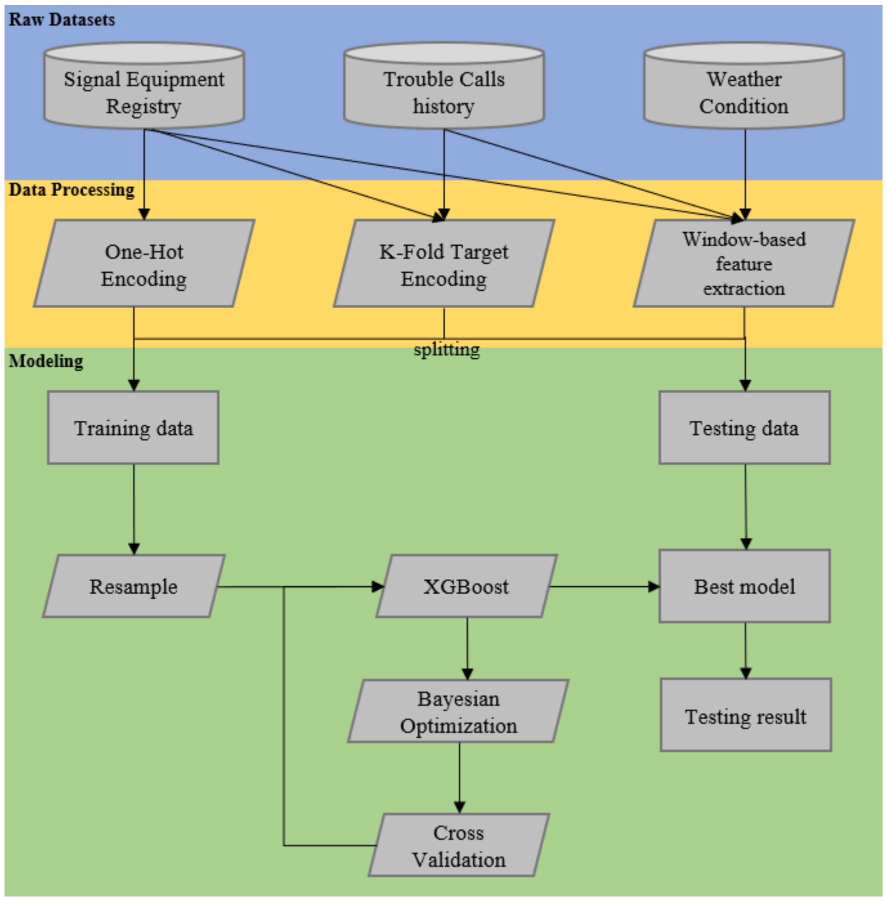
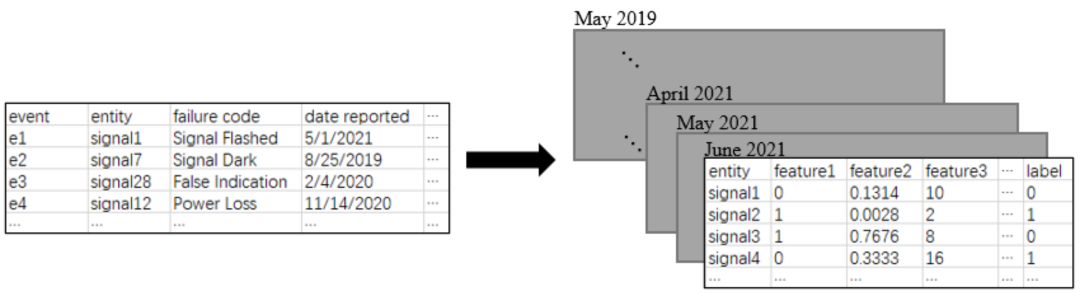


Figure 4.1. Framework of the Proposed Method

The datasets utilized in this study include the signal equipment registry, the trouble calls history, and the weather condition data. In the data processing step, we aim to convert event-based data into a data form that could be better used in the machine learning model. We use One-Hot Encoding and K-Fold Target Encoding to process categorical data in the original dataset and a window-based method to extract important features monthly.

Figure 4.2 illustrates the input and output of the data processing method. The categorical data are either transformed into binary values through One-Hot Encoding or decimals in between 0 and 1 using Target Encoding. More features are generated via the window-based feature extraction method, which will be explicated later in this section. The label in Figure 2 refers to the class label for a given observation, indicating whether a signal failure occurred in the given month (1) or not (0). In the modeling step, due to the rarity of failure events, we resample the highly imbalanced dataset before training. Various resampling strategies, including random oversampling, the Synthetic Minority Over-sampling Technique (SMOTE), the Adaptive Synthetic sampling approach (ADASYN), and random undersampling, are each tested in experiments. After resampling the training dataset, we train multiple models using Bayesian Optimization and 5-fold Cross Validation, and calculate the average AUC scores for the trained models with different combinations of hyperparameter settings. The best model (with the highest average AUC score after 5-fold Cross Validation) is selected and applied to the testing dataset.



event	entity	failure code	date reported	···
e1	signal1	Signal Flashed	5/1/2021	···
e2	signal7	Signal Dark	8/25/2019	···
e3	signal28	False Indication	2/4/2020	···
e4	signal12	Power Loss	11/14/2020	···
···	···	···	···	···

entity	feature1	feature2	feature3	···	label
signal1	0	0.1314	10	···	0
signal2	1	0.0028	2	···	1
signal3	1	0.7676	8	···	0
signal4	0	0.3333	16	···	1
···	···	···	···	···	···

Figure 4.2. Input and Output of the Data Processing Approach

4.1. Categorical Data Encoding

In a machine learning model, the output variable is affected not only by quantitative (numerical) variables, but also qualitative (categorical) variables. In order to use categorical variables in machine learning models, it is necessary to transform the categorical data into numerical values using encoding techniques (Potdar et al., 2017).

One-Hot Encoding is one of the most common encoding techniques in categorical data processing. It converts categorical variables into multiple lists of binaries indicating the presence (1) or absence (0) of the variable (Potdar et al., 2017). Let $X = \{x_1, x_2, x_3, \dots, x_n\}$ denote n categorical features and l_i represent the number of distinct values of feature x_i. One-Hot Encoding transforms a single x with l distinct values to $Z = \{z_1, z_2, z_3, \dots, z_l\}$, where $z \in \{0,1\}$. However, as the cardinality of the categorical variable increases, using One-Hot Encoding may create too many predictors, which can reduce the model's performance and be computationally expensive.

Target Encoding is an alternative encoding scheme for high-cardinality categorical data that does not increase the dimensionality of the original dataset. This scheme replaces the categorical feature with the posterior probability of the target label, conditioned by the categorical value and the prior probability of the target label over all data samples (Micci-Barreca, 2001). Assume a feature x has l distinct values, that is $x \in S$ where $S = \{s_1, s_2, s_3, \dots, s_l\}$. For each s_i, the replacement value can be calculated using the below equation (1):

$$P(t = 1 \mid x = s_i) = \frac{P(t=1 \,\&\, x=s_i)}{P(x=s_i)} \qquad (1)$$

Where $t \in \{0, 1\}$ is the target label, indicating the presence ($t = 1$) of a failure or not ($t = 0$). Since Target Encoding uses some information from the target to predict the target, it has a tendency of overfitting to the training dataset, especially when the distribution of the categorical features in the training dataset and the testing dataset are significantly different (Grover, 2019). Therefore, we apply its extension, K-Fold Target Encoding, to reduce the risk of overfitting (Pourya, 2019). We divide the dataset into K-stratified folds, where $K = 5$. Then, we replace the categorical values in fold i with a mean target using the equation (1) for the rest of the $K - 1$ folds.

4.2. Feature Extraction

We propose a window-based feature extraction method to obtain useful information from past events in a k time-period window. In this study, we use one month as the minimum time-period for prediction. The proposed feature extraction approach contains three parts, as follows.

In a k-month window, we can: 1) find the latest event for each entity, then extract some event features (e.g., failure cause, maintenance action) using One-Hot Encoding and calculate the number of days since the last event occurred. Assuming entity v_q has p events $\{\mathrm{e_1, e_2, e_3, \dots, e_p}\}$ (in chronological order) recorded in a k-month window $[\mathrm{j, j+k-1}]$, our method generates features from the p^{th} event and the number of days since the p^{th} event occurred (first day in $(\mathrm{j} + \mathrm{k})$ month $- \mathrm{t_{e_p}}$). 2) We search through the k-month window and count the number of events that occurred in each month. k features will then be generated by the second part. 3) Then, we can move forward the k-month window one month to $[\mathrm{j+1, j+k}]$ and repeat the above steps until $\mathrm{j + k}$ reaches the latest month recorded in the original dataset.

Algorithm 4.1 (below) shows a pseudocode of steps 1) and 2) for target month May 2019 (i.e. predicting rail signal failures in May 2019) based on a 12-month window.

Algorithm 4.1:

```
target_month = 2019-05
k = 12
features = empty list
for v in V:
    feature_v = empty list
    events_for_v = empty list
    for e in E:
        if e.month < target_month-k or e.month >= target_month:
            continue
        if e.entity == v:
            events_for_v.append(e)
    # do step 1)
    find the latest event $e_p$ from events_for_v
    feature_v.append(Categorical_Encoding($e_p$))
    days_past_$e_p$ = (target_month – $e_p$.date).days
    feature_v.append(days_past_$e_p$)
    # do step 2)
    for i from 1 to k:
        num = 0
        for e in events_for_v:
            if e.month == target_month-i:
                num = num + 1
        feature_v.append(num)
    # generate label
    for e in E:
        label = 0
        if e.entity==v and e.month==target_month:
            label = 1
    feature_v.append(label)
    features.append(feature)
```

4.3. Resampling

Most machine learning algorithms were developed based on the assumption that the number of observations in different classes are similar (Krawczyk, 2016). However, in rare event analysis, the distribution of the observations is largely skewed, having a very small number of failures and a large proportion of normal events.

This leads to a problem in that machine learning algorithms may ignore the minority class (i.e., failures). An approach for addressing the issue of data imbalance is to resample the training dataset.

There are two main types of resampling techniques: oversampling and undersampling. Oversampling expands the minority by randomly duplicating the minority observations or by creating synthetic minority examples, whereas the undersampling technique rebalances the training dataset by deleting some majority observations. Despite their advantages, both resampling methods could also negatively affect model performance (Menardi & Torelli, 2014). Oversampling can increase the likelihood of overfitting and undersampling can discard useful information from the majority class. In our experiments, we test four different resampling methods, as follows.

Random Oversampling is the most common oversampling method. It expands the dataset by simply replicating data samples of the minority class (Menardi & Torelli, 2014). Different from other synthetic oversampling methods, random oversampling does not generate new samples. Although this technique is simple to implement and widely used, it can cause overfitting on the duplicated samples of the minority class and be ineffective for the classifier to find a borderline between the majority class and the minority class.

SMOTE is a state-of-art oversampling technique that rebalances the dataset by creating synthetic examples of the minority class. Instead of oversampling with replacement, this method takes each sample of the minority class and generates new examples by joining the k ($k = 5$ in our experiments) minority class's nearest neighbors (Chawla et al., 2002). However, SMOTE may not deal well with high dimensional data and may lead to over-generalization.

ADASYN is another state-of-art synthetic oversampling approach that was inspired by SMOTE. In addition to rebalancing the distribution of the original dataset, this method can adaptively shift the classification decision boundary towards the difficult-to-learn samples (He et al., 2008). When rebalancing a multi-label dataset, SMOTE provides equal opportunities for increasing each minority class, whereas ADASYN oversamples the dataset according to the distribution of the minority class. Our study focuses on binary classification, where the minority class contains only one label. When rebalancing a two-label dataset, ADASYN creates samples with a little more variance to make it more realistic as compared to SMOTE. A drawback of ADASYN is that it is unable to identify noisy instances, indicating that outliers in the dataset may affect this method's performance (Dattagupta, 2018).

Random Undersampling is also widely used to address imbalanced data. It rebalances the dataset by randomly removing a portion of samples from the majority class (Menardi & Torelli, 2014). This method can accelerate the learning process because it decreases the size of the training dataset, but some useful information from the majority class may be missed.

Table 2 compares the advantages (except balancing the dataset) and limitations of the resampling techniques mentioned above. Each method has its limitations. A comparative study and its results are demonstrated in the RESULTS section based on our datasets.

Table 4.1. Comparison of Alternative Resampling Techniques

Method	Advantages	Limitations
Random Oversampling	Easy to implement	Overfitting on the minority; increase the training time
SMOTE	Generate synthetic data	Poor performance on high dimensional data; over generalization
ADASYN	Generate synthetic data; More realistic than SMOTE	Unable to deal with outliers
Random Undersampling	Reduce the training time	Miss certain information from the majority class

4.4. Machine Learning Algorithm

The Machine Learning aims to map a list of input variables $X = \{x_1, x_2, x_3, \ldots, x_n\}$ to an output variable Y. In this case, X represents the feature variables that are generated from Table 3.1 after data processing, and Y is a binary label that indicates the presence of a signal failure (1) or not (0) in a particular month for prediction.

We use XGBoost, a widely used scalable implementation of the tree boosting algorithm, in this study. This method assembles a considerable number of weak but complementary CARTs to create a more robust classifier. Compared to Gradient Boost Decision Trees, improvements were made in the regularized learning object of XGBoost, which is simpler and easier to parallelize (Chen & Guestrin, 2016). The following Equation (1) demonstrates the regularized objective, which is to be minimized in the learning process.

$$L(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k^K \Omega(f_k) \qquad (1)$$

The l function denotes a loss function that measures the difference between the ground truth y_i and the estimated value $\hat{y}_i$. The Ω function is the regression tree function that penalizes the model complexity. This can smooth the final learned weight to reduce the risk of overfitting. More detailed explanations can be found in Chen & Guestrin (2016).

In our experiments, the XGBoost algorithm is implemented using the XGBoost python package and the Scikit-Learn python library. The performance of the model is evaluated using the Area Under the receiver operating characteristic Curve (AUC) because typical evaluation metrics, such as accuracy, may not be appropriate when the data is imbalanced (Chawla et al., 2002). The Receiver Operating Characteristic (ROC) curve illustrates a binary classifier performance by plotting the true positive rate against the false positive rate over a range of threshold settings of the decision criterion. The AUC is the proportion of the area under the ROC curve compared to the entire graph, which is considered as a preferred single-valued measure of model performance (Swets, 1988). When the AUC is 0.5, that is, when the ROC curve is on the diagonal, it means that the classification ability of the model is as poor as random guessing. When the AUC is 1, it indicates a "perfect" model. Furthermore, Bayesian optimization with 5-fold cross validation is applied for hyperparameter selection. For each combination of hyperparameters, we conduct a 5-fold cross validation and calculate the average AUC score. After the full iterations of Bayesian optimization, an optimal model with the highest score is chosen to use for the testing dataset.

5. Results

Given the dataset described in section 3, 14 possible input variables are generated based on the proposed methods. The variables and their descriptions are listed in Table 5.1. Variables 1, 2, and 3 represent the location information of the signal unit, which are retrieved from the Signal Equipment Registry dataset. Variables 4-8 represent the failure and corresponding maintenance work from the Trouble Calls history dataset. Variables 9-12 are generated by the proposed window-based feature extraction method. Variables 13 and 14 are the weather condition of the region. To deal with the eight categorical variables, we apply One-Hot Encoding to Variable 1 and 2, and apply 5-Fold Target Encoding to the other variables (i.e., Variables 3-8).

Table 5.1. Input Variables

No.	**Variables**	**Type**	**Cardinality (size of i)**	**Descriptions**
1	D_i	Categorical	4	Division
2	SD_i	Categorical	6	Subdivision
3	L_i	Categorical	66	Line where the signal is located
4	$type_i$	Categorical	7	Type of work being performed for the last failure (e.g., Corrective, Preventive, Capital)
5	pc_i	Categorical	96	Problem Code (the symptom observed) of the last failure
6	fc_i	Categorical	144	Failure Code (which equipment failed and its malfunction) of the last failure
7	cc_i	Categorical	103	Cause Code (what caused the failure) of the last failure
8	ac_i	Categorical	35	Action Code (step taken to resolve the failure) of the last failure
9	m	Numerical		The month in which failures are being predicted
10	day	Numerical		Number of days since the last failure occurred
11	nf_i	Numerical	k	Number of failures in each month of the k-month window
12	tnf	Numerical		Total number of failures that occurred in the k-month window
13	t	Numerical		Average temperature of the last month in Fahrenheit
14	p	Numerical		Total precipitation of the last month in millimeters

The original dataset contains all rail transit signal failures that occurred from May 2018 to June 2021. We apply a 12-month window feature extraction method. Due to data limitations, we only consider the predicting months from May 2019 to June 2021. In this case, the processed dataset contains 26 months of data, with 18,623 observations for each month.

We then split the dataset into a training dataset and a testing dataset. Since the past data may contain important information for predicting future events, we use the dataset from May 2019 to December 2020 as the training dataset and the others (i.e., from January 2021 to June 2021) as the testing dataset. The distributions of each class for both training data and testing data are displayed in Table 5.2. Both datasets are highly imbalanced and contain around 3% of the failure class.

Table 5.2. Class Distribution of the Training Data

Dataset	**Class**	**Sample amount**	**Percentage**
Training	**Failure**	12,298	3.3%
	Normal	360,162	96.7%
Testing	**Failure**	3,270	2.9%
	Normal	108,468	97.1%

We applied various resampling techniques to the training dataset to balance the failure class and the normal class to equal sample size. Therefore, the training dataset contains 360,162 samples of each class after random oversampling, SMOTE oversampling, and ADASYN oversampling, while containing only 12,298 samples of each class after random undersampling.

The model performances of each resampling technique used as well as original imbalanced data are illustrated using ROC curves in Figure 5.1 (below). It shows that the random undersampling approach has the highest tested AUC value (0.75) among all the test cases. It also reduced model training time in comparison to the oversampling methods.

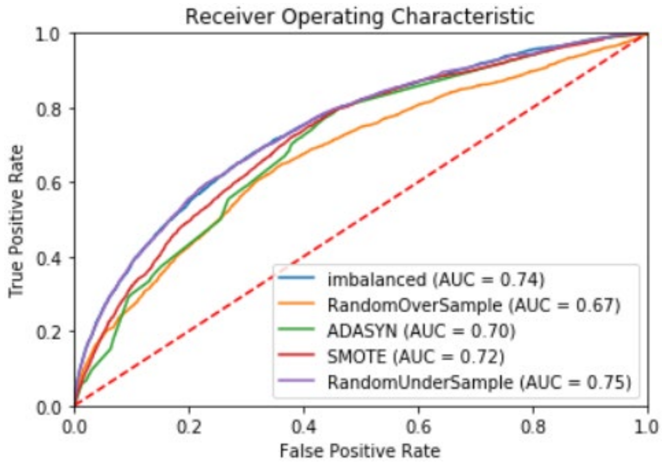


Figure 5.1. AUCs of XGBoost Models using Different Resampling Techniques

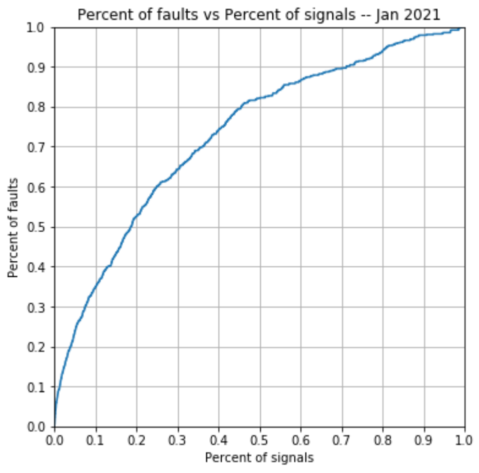
In addition, we apply Random Forest and traditional fully connected Neural Network algorithms to the non-resampled dataset and the random undersampled dataset. The results of each model are displayed in Table 5.3 below. On the Random Forest model and the Neural Network model, random undersampling not only shortens the model training time, but also slightly increases the AUC value. Further, the XGBoost algorithm produces a better result for both the non-resampled dataset and the undersampled dataset than Random Forest and Neural Network in our experiments.

Table 5.3. AUCs of Various Machine Learning Models using Random Undersampled Dataset and Original Dataset Without Resampling

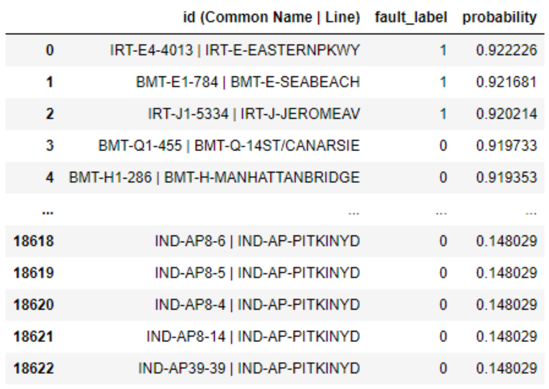
		XGBoost	**RF**	**NN**
Not Resampled	**Training**	0.7498	0.8086	0.7477
	Testing	0.7447	0.7391	0.7325
Random Undersampled	**Training**	0.7437	0.8423	0.7499
	Testing	0.7451	0.7412	0.7365

To visualize the performance of the selected model (XGBoost and random undersampling), we visualize the relationship between the percentage of signals screened using our machine learning algorithm (prediction), and the percentage of the total number of rail signal failures that could be found in those locations (reality).

For example, if we take January 2021 as the predicting month, Figure 5.2(a) shows the failure percentage curve, which is plotted using the failure probability table shown in Figure 5.2 (b). The y-axis represents the percentage of actual failures in January 2021 and the x-axis stands for the percentage of signals among all 18,623 signal units. The points on the plots are calculated by ranking the signals based on estimated failure probability in descending order, as is displayed in Figure 5.2 (b). According to the failure percentage curve for January 2021 (shown in Figure 5.2 (a)), screening the top 10% of signals may identify around 35% of signal failures in advance.



(a)



	id (Common Name | Line)	fault_label	probability
0	IRT-E4-4013 | IRT-E-EASTERNPKWY	1	0.922226
1	BMT-E1-784 | BMT-E-SEABEACH	1	0.921681
2	IRT-J1-5334 | IRT-J-JEROMEAV	1	0.920214
3	BMT-Q1-455 | BMT-Q-14ST/CANARSIE	0	0.919733
4	BMT-H1-286 | BMT-H-MANHATTANBRIDGE	0	0.919353
...	...	...	...
18618	IND-AP8-6 | IND-AP-PITKINYD	0	0.148029
18619	IND-AP8-5 | IND-AP-PITKINYD	0	0.148029
18620	IND-AP8-4 | IND-AP-PITKINYD	0	0.148029
18621	IND-AP8-14 | IND-AP-PITKINYD	0	0.148029
18622	IND-AP39-39 | IND-AP-PITKINYD	0	0.148029

(b)

Figure 5.2. Failure Percentage Curves (a) and Estimated Probability of Failed Signal Table (b) of January 2021 Prediction

Figure 5.3 (below) shows the ranking of the feature importance in the selected model. The importance of each feature is calculated by its “weight,” which is the number of times the feature appears in a tree. As shown in Figure 5.3, the y-axis displays the features used in the model and the x-axis represents the “weight.” The feature “line,” which was processed using target encoding, is most useful in the construction of the boosted decision trees within the model. This may indicate that some location-specific characteristics might affect signal failure occurrence. The “pmonth_total” variable represents the total number of failures that happened to the signal within the past 12 months. This may indicate that the locations with large numbers of signal failures may be likely to experience failures again in the future. The third feature “lastOccurDays” refers to the number of days between the first day of the predicting month and the date when the most recent failure occurred. The different actions operated for the past failure may also influence the probability of signal failure in the future. For example, if the equipment is replaced, it is supposed to have better durability than the one that is simply repaired or rechecked in the same conditions. Temperature is also important for predicting signal failures. For example, in extremely hot weather, failures may occur due to blown fuses and cable faults (Greenham et al., 2020). In addition, it is interesting to observe that the features produced by target encoding (i.e., Line, Action Code, Failure Code, Cause Code, Problem Code, Type) are all considered more important than the features generated by One-Hot Encoding.

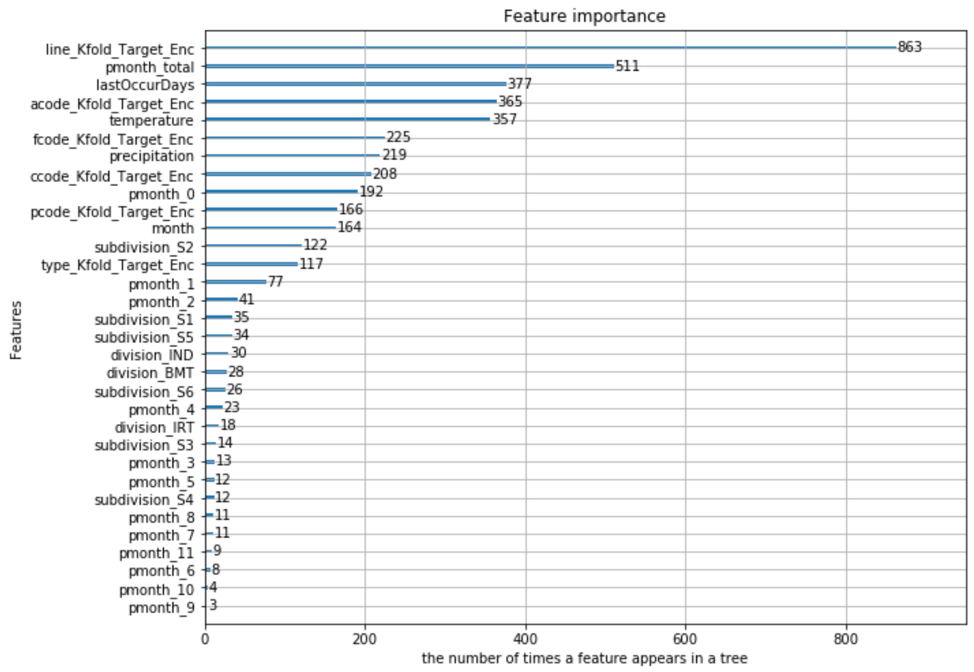


Figure 5.3. Feature Importance Plot for the Proposed Model

6. Conclusions

This report proposed an event-based data analysis method to generate data that can be better used in a machine learning model to predict failures of urban rail transit signals. The proposed data processing approach consists of a combination of One-Hot Encoding, 5-fold Target Encoding, and window-based feature extraction. We validate the approach using the XGBoost algorithm and Bayesian Optimization for hyperparameter selection. Four resampling methods: random oversampling, random undersampling, SMOTE, and ADASYN were also tested and compared to the model using the original imbalanced dataset. Our proposed method can capture about 35% of total failures from 10% of screened signal locations. In the future, additional information (e.g., equipment age, sensory information) may be incorporated into model improvements.

REFERENCES

[1] Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). Classification And Regression Trees. Routledge. https://doi.org/10.1201/9781315139470.

[2] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, 321–357. https://doi.org/10.1613/jair.953.

[3] 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.

[4] Dattagupta, S. J. (2018). A performance comparison of oversampling methods for data generation in imbalanced learning tasks [PhD Thesis].

[5] Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B., & Herrera, F. (2018). Learning from Imbalanced Data Sets. Springer International Publishing. https://doi.org/10.1007/978-3-319-98074-4.

[6] Fitzsimmons, E. G. (2017, May 1). Key to Improving Subway Service in New York? Modern Signals. The New York Times. https://www.nytimes.com/2017/05/01/nyregion/new-york-subway-signals.html

[7] Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. Icml, 96, 148–156.

[8] Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. Annals of Statistics, 1189–1232.

[9] Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting (with discussion and a rejoinder by the authors). The Annals of Statistics, 28(2), 337–407.

[10] Greenham, S., Ferranti, E., Quinn, A., & Drayson, K. (2020). The impact of high temperatures and extreme heat to delays on the London Underground rail network: An empirical study. Meteorological Applications, 27(3), e1910. https://doi.org/10.1002/met.191

[11] Grover, P. (2019, July 8). Getting Deeper into Categorical Encodings for Machine Learning. https://towardsdatascience.com/getting-deeper-into-categorical-encodings-for-machine-learning-2312acd347c8

[12] Hasanin, T., & Khoshgoftaar, T. (2018). The effects of random undersampling with simulated class imbalance for big data. 2018 IEEE International Conference on Information Reuse and Integration (IRI), 70–79.

[13] He, H., Bai, Y., Garcia, E. A., & Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), 1322–1328. https://doi.org/10.1109/IJCNN.2008.4633969

[14] Riders Alliance (Accessed in March 2022). https://static1.squarespace.com/static/61033b9bd377817f5bcc6db9/t/614e54a63c9104704b083efc/1635951289660/The+Bad%2C+Old+Normal%3A+Subway+Signal+Delays+-+Riders+Alliance.

[15] Krawczyk, B. (2016). Learning from imbalanced data: Open challenges and future directions. Progress in Artificial Intelligence, 5(4), 221–232. https://doi.org/10.1007/s13748-016-0094-0

[16] Ling, C. X., & Li, C. (1998). Data mining for direct marketing: Problems and solutions. Kdd, 98, 73–79.

[17] Menardi, G., & Torelli, N. (2014). Training and assessing classification rules with imbalanced data. Data Mining and Knowledge Discovery, 28(1), 92–122. https://doi.org/10.1007/s10618-012-0295-5

[18] Micci-Barreca, D. (2001). A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. ACM SIGKDD Explorations Newsletter, 3(1), 27–32. https://doi.org/10.1145/507533.507538

[19] O'Madadhain, J., Hutchins, J., & Smyth, P. (2005). Prediction and ranking algorithms for event-based network data. ACM SIGKDD Explorations Newsletter, 7(2), 23–30. https://doi.org/10.1145/1117454.1117458

[20] Potdar, K., Pardawala, T. S., & Pai, C. D. (2017). A comparative study of categorical variable encoding techniques for neural network classifiers. International Journal of Computer Applications, 175(4), 7–9.

[21] Pourya. (2019, February 5). K-Fold Target Encoding. Medium. https://medium.com/@pouryaayria/k-fold-target-encoding-dfe9a594874b

[22] Provost, F. (2000). Machine learning from imbalanced data sets 101. Proceedings of the AAAI'2000 Workshop on Imbalanced Data Sets, 68(2000), 1–3.

[23] Prusa, J., Khoshgoftaar, T. M., Dittman, D. J., & Napolitano, A. (2015). Using random undersampling to alleviate class imbalance on tweet sentiment data. 2015 IEEE International Conference on Information Reuse and Integration, 197–202.

[24] Ren, W., Ma, L., & Wang, Y. (2020). Monte Carlo analysis for safety and reliability of rail transit signal system based on Cloud Computing. Journal of Physics: Conference Series, 1654, 012065. https://doi.org/10.1088/1742-6596/1654/1/012065

[25] Scikit-learn: Machine learning in Python—Scikit-learn 1.0 documentation. (n.d.). Retrieved October 20, 2021, from https://scikit-learn.org/stable/index.html

[26] Swets, J. A. (1988). Measuring the Accuracy of Diagnostic Systems. Science, 240(4857), 1285–1293. https://doi.org/10.1126/science.3287615

[27] Tu, J., Tao, Q., & Deng, Q. (2011). Safety evaluation of urban transit signal system based on the improved TOPIS. Procedia Engineering, 15, 4558–4562.

[28] Wiedmann, W. (2021, July 29). As Railroad Systems Advance, Wayside Signals Fade Away. Burns Engineering. https://insights.burns-group.com/2021/07/29/as-railroad-systems-advance-wayside-signals-fade-away/

[29] XGBoost Documentation—Xgboost 1.5.0-dev documentation. (n.d.). Retrieved September 6, 2021, from https://xgboost.readthedocs.io/en/latest/index.html#

[30] Zhang, Y.-P., Xu, Z.-J., & Su, H.-S. (2013). Risk assessment on railway signal system based on Fuzzy-FMECA method. Sensors & Transducers, 156(9), 203.