Case Study

Institution of MECHANICAL ENGINEERS
Journal of RAIL AND RAPID TRANSIT

Proc IMechE Part F:
J Rail and Rapid Transit
2022, Vol. 0(0) 1–10

DOI: 10.1177/09544097221127781
journals.sagepub.com/home/pif
SAGE

Machine learning based prediction of rail transit signal failure: A case study in the United States

Junyan Dai and Xiang Liu

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 1 month in advance, based on records of past failures and maintenance events. 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 1 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.

Keywords

urban rail transit, signal, machine learning, imbalanced data mining

Introduction

Urban rail transit is an important, safe, efficient, and environmentally friendly mode of passenger transport. Signal system, 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). Considering New York City Transit (NYCT) as an example, 80% of 125 non-holiday working days in the first half of 2021 had signal problems that caused a delay during the morning rush hours (6 am–10 am).[1] 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.[2] To give another example, in the Bay Area Rapid Transit (BART) system, in-service signal failures account for 50% of infrastructure-related delays and slow service for about 400 hours per year.[3] 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.

To ensure a safe and reliable operation, rail transit companies must conduct routine inspection and maintenance on all equipment, including signals, and record the work orders as well as trouble calls. The recorded inspection, maintenance, or trouble calls can be considered as event-based data, which consists of a set of events (i.e., inspection, maintenance work, or trouble calls) and a set of participating entities (i.e., signal equipment). Event-based data is very common in the real world, including email traffic, telephone calls, and research publications.[4] 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. In our dataset, about 3% of signals in the study region were 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 are designed based on the assumption that the class distribution is equal or slightly imbalanced.[5] For rare event prediction, imbalanced data mining (IDM) techniques can be implemented.[6] Resampling is a widely used IDM technique, in which the class distribution of the training data is changed by either increasing the minority data samples or removing some majority data samples. A number of resampling techniques were proposed and validated in the previous studies.[6–8] However, no resampling method can guarantee superior performance over others.[9] In this research, we conduct a

Rutgers University, Piscataway, NJ, USA

Corresponding author:
Xiang Liu, Rutgers University, 500 Bartholomew Road, Piscataway, NJ 08854, USA.
Email: xiang.liu@rutgers.edu

comparative experiment in this signal failure prediction problem using various resampling methods, which includes random oversampling, random undersampling,[8] Synthetic Minority Oversampling Technique (SMOTE),[6] and ADAptive SYNthetic sampling approach (ADASYN).[7]

This study aims to predict urban rail transit signal failures 1 month in advance using records of past failures and maintenance events. We apply a machine learning algorithm, Extreme gradient boosting (XGBoost), to build the predictive model. XGBoost is a scalable implementation of the tree boosting algorithm, which has been widely used as the most powerful machine learning approach to solve data mining problems.[10] The implementation of XGBoost has several advantages. Firstly, it is computationally faster than the standard gradient boosting machine as it utilizes parallel processing. Second, it supports L1 (Lasso Regression) and L2 (Ridge Regression), which both prevent overfitting. Moreover, XGBoost can efficiently handle missing data and has a built-in cross validation function. In this study, we compare XGBoost model with Random Forest, Neural Network, and Logistic Regression models, which shows that XGBoost has a better performance.

The remainder of this paper is organized as follows. The next section discusses related works and identifies knowledge gaps in the existing literature. The following section introduces the dataset utilized in our experiments. The proposed approach for failure prediction of rail transit signals is presented in the "Methodologies" section. In the "Results" section, we test the effectiveness of various approaches in addressing imbalanced data and discuss the model performance in comparison with the empirical data. Our conclusions as well as future work are elaborated in the final section.

Literature review and knowledge gap

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.[11] Zhang et al. presented a new risk assessment method called Fuzzy-FMECA (Failure Mode, Effects and Criticality Analysis) for railway signal systems.[12] 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.[13] The above studies focus on the design of the signal system rather than the safe operation of each signal. There were a few previous studies on predicting failures of each signal equipment. Yang et al. proposed a novel model for railway signal equipment fault classification using SMOTE and ensemble learning.[14] Their model is applied to unstructured fault text data with limited features. Gao et al. proposed an improved feature representation method in combination with multilevel classification model to predict high-speed railway signal equipment fault using fault text data.[15] Both of the prior studies on railway signal equipment fault prediction used text data, which contains limited features and requires sophisticated text mining technologies.

XGBoost was originally developed from the Classification And Regression Trees (CART) algorithm.[16] Then, in 1996, Freund and Schapire proposed AdaBoost which combines many relative weak trees to create a highly accurate classifier.[17] Friedman et al., in 2000, interpreted the boosting as an additive logistic regression model (i.e., AdaBoost) which aims to minimize exponential error,[18] and soon proposed a gradient boosting machine to address the general supervised problem by combining boosting and CART.[19] Recently, Chen and Guestrin proposed a scalable and efficient implementation of the gradient boosting machine, which has been widely accepted and used to address many machine learning challenges.[10] In the field of railway, several prior studies have implemented XGBoost and revealed that it outperforms other machine learning algorithms. Zhang et al. proposed a broken rail prediction method using XGBoost and validated that XGBoost achieves a better score than logistic regression and random forests.[20] Shi and Xu applied XGBoost in combination with Bayesian Optimization algorithm to predict train arrival delays.[21] Their proposed method outperforms several other machine learning models, which includes random forests and gradient boosted decision trees.

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.[22] 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.[23] SMOTE, proposed by Chawla et al.,[6] and ADASYN, proposed by He et al.,[7] both create new samples for minority class instead of duplicating the existing ones. There is also significant literature on the effects of random undersampling.[24,25] 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.[26]

Knowledge gaps

Although many previous studies have developed various approaches for analyzing the safety degrees of rail transit signal systems, very few have studied the failure prediction for each single signal unit or equipment. 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 past failures and maintenance records.

Table 1. Variables retrieved from the datasets.

Variables	Descriptions
Division	Historic operating company (division) where the work is to be performed
Subdivision	Defined subdivision of the study region where the signal is located
Line	Line (track) 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)
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

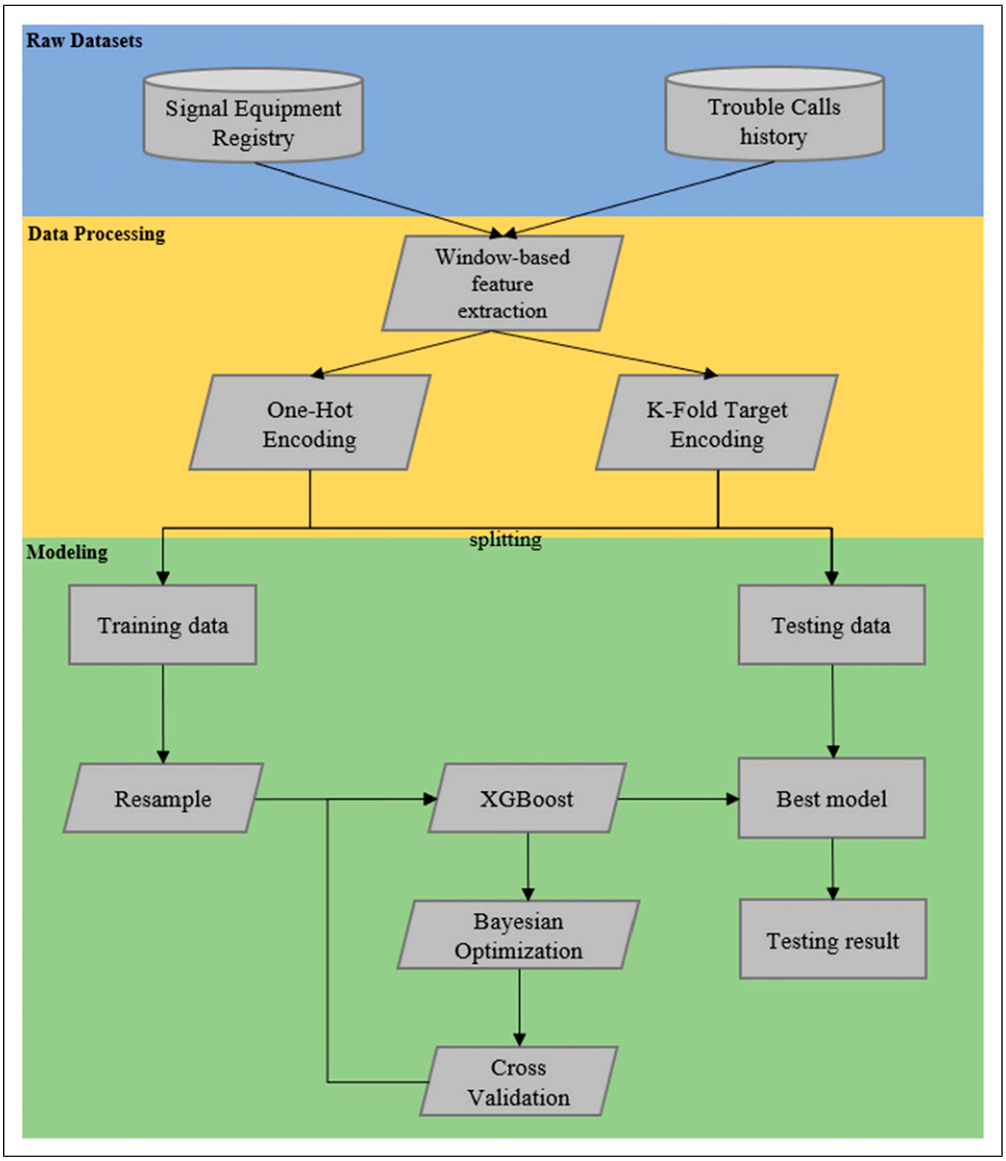

Figure 1. Framework of the proposed method.

Data

The signal equipment registry and the trouble call history from a major rail transit agency in the United States 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 installed in the field (e.g., Class, Division, Subdivision, Line). In this dataset, signal head, track circuit, and insulated joint are all defined as signal equipment. 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. Table 1 displays a detailed list of variables gathered

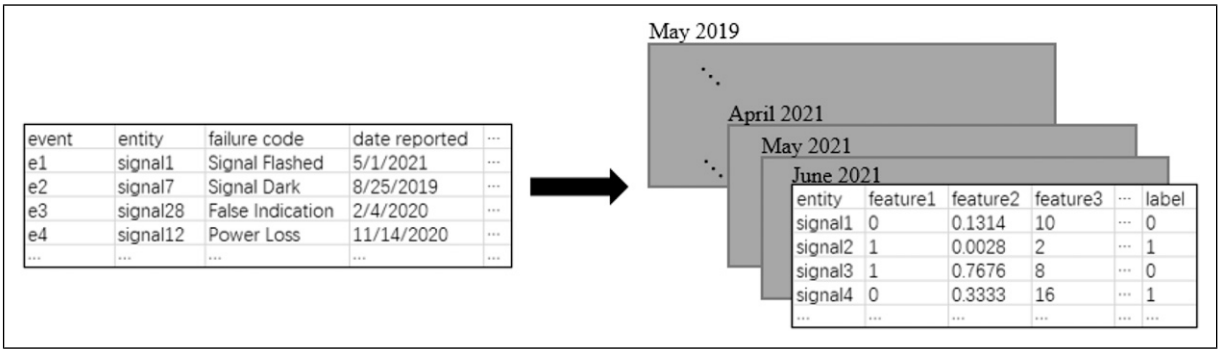

Figure 2. Input and output of the data processing approach.

from the two 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 signal unit. For example, a failure that occurred on the insulated 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 unit in May 2021. In total, 18,623 signal units on 66 lines are collected and used in our experiments.

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 1 demonstrates the framework of our proposed method.

The datasets utilized in this study include the signal equipment registry and the trouble calls history. 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 propose a window-based method to extract important features monthly and use One-Hot Encoding and K-Fold Target Encoding to process categorical features.

Figure 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 signal unit, 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 Oversampling Technique (SMOTE), the Adaptive Synthetic sampling approach (ADASYN), and random undersampling, are 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.

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.[27]

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.[27] Let $X = \{x_1, x_2, x_3, \ldots, x_n\}$ 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 $Z = \{z_1, z_2, z_3, \ldots, z_l\}$, 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.[28] Assume a feature x has l distinct values, that is $x \in S$ where $S = \{s_1, s_2, s_3, \ldots, s_l\}$. For each s_i, the replacement value can be calculated using the below equation (1)

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

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.[29] Therefore, we apply its extension, K-Fold Target Encoding, to reduce the risk of overfitting.[30] 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.

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 1 month as the minimum time-period for prediction. The proposed feature extraction approach contains three parts, as follows.

In a k -month window, (1) we find the latest event for each entity, then extract some event features (e.g., failure cause, maintenance action) with One-Hot Encoding and calculate the number of days since the last event occurred. Assuming entity v_q has p events $\{e_1, e_2, e_3, \ldots, e_p\}$ (in chronological order) recorded in a k -month window $[j, j + k - 1]$, where j is the starting month, our method generates features from the p^{th} event and the number of days since the p^{th} event occurred (first day in the month $(j + k) - t_{e_p}$). (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 1 month to $[j + 1, j + k]$ and repeat the above steps until $j + k$ reaches the latest month recorded in the original dataset.

Algorithm 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 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 ep from events_for_v
feature_v.append(Categorical_Encoding(ep))
days_past_ep = (target_month – ep.date).days
feature_v.append(days_past_ep)
# 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)
```

Given the dataset described in section 3, 12 possible input variables are generated using the categorical data encoding and window-based feature extraction methods. The variables and their descriptions are listed in Table 2. 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. 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).

Resampling

Most machine learning algorithms were developed based on the assumption that the number of data samples in different classes are similar.[22] However, in rare event analysis, the distribution of the data samples 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 samples or by creating synthetic minority samples, whereas the undersampling technique rebalances the training dataset by deleting some majority samples. Despite their advantages, both resampling methods could also negatively affect model performance.[8] 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.[8] 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.[6] 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.[7] When

Table 2. Input variables.

No.	Variables	Type	Cardinality (size of *i*)	Descriptions
1	D_i	Categorical	4	Operating company where the work is to be performed
2	SD_i	Categorical	6	Defined subdivision of the study region where the signal is located
3	L_i	Categorical	66	Line (track) where the signal is located
4	$type_i$	Categorical	7	Type of work being performed for the last failure (e.g., corrective, preventive)
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

Table 3. 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

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.[31]

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.[8] 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 3 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.

Machine learning algorithm

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 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.

Table 4. Class distribution of the dataset.

Dataset	Class	Sample amount	Percentage, %
Training	Failure	12,298	3.3
	Normal	360,162	96.7
Testing	Failure	3270	2.9
	Normal	108,468	97.1

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.[10] The following Equation (2) demonstrates the regularized objective, which is to be minimized in the learning process.

$$L(\phi) = \sum_i l\left(\hat{y}_i, y_i\right) + \sum_k^K \Omega(f_k) \tag{2}$$

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.[6] 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 a preferred single-valued measure of model performance.[26] 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.

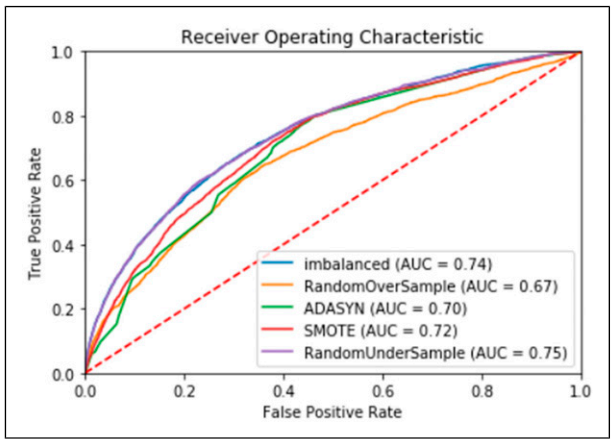

Figure 3. AUCs of XGBoost models using different resampling techniques.

Results

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 samples 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 4. Both datasets are highly imbalanced and contain around 3% of the failure class.

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 3 (below). It shows that the random undersampling approach has the highest tested AUC value (0.75) among all the test cases.

In addition, we apply Random Forests, a fully connected 2-layer Neural Network, and Logistic Regression to the non-resampled dataset and the random undersampled dataset. Bayesian optimization is applied to all machine

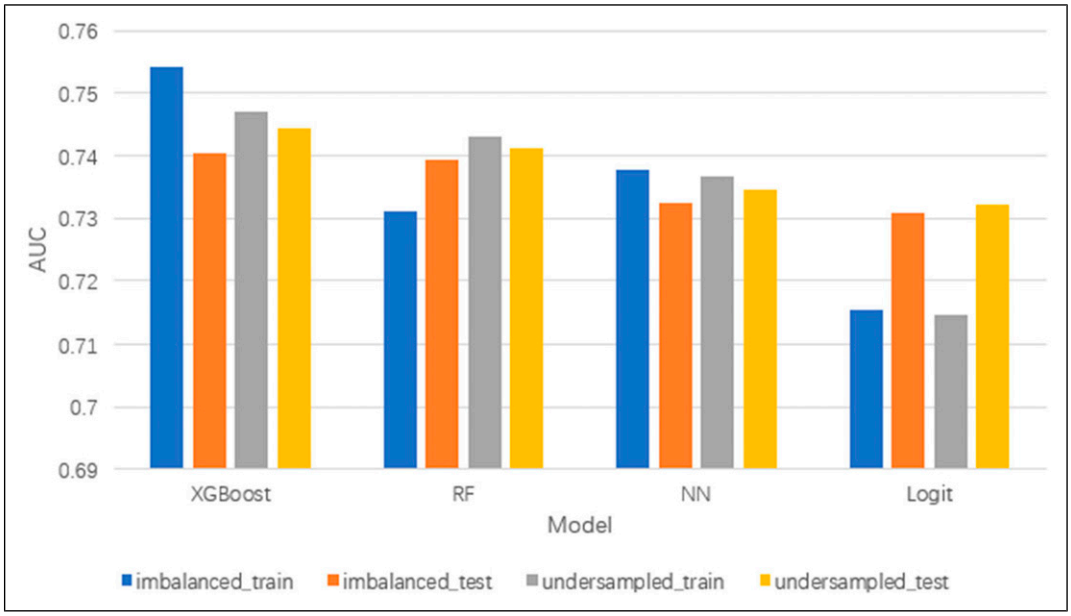

Figure 4. AUCs of various models using random undersampled and original datasets.

learning methods for hyperparameter selection. The results of each model are displayed in Figure 4 below. In all models, random undersampling slightly increases the AUC value in the testing set. The XGBoost algorithm produces a better result for both the non-resampled dataset and the undersampled dataset than Random Forests, Neural Network, and Logistic Regression models in our experiments.

To visualize the performance of the selected model (XGBoost and random undersampling), we. Draw 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) in Figure 5, which takes January 2021 as the predicting month. First, we rank the signals based on the predicted probabilities (output of the predict_proba() function according to python xgboost library) in descending order. Next, we start screening according to the ranked signal list and record the number of signals has been screened and the number of signals actually had failures. Then, the failure percentage curve is drawn in Figure 5, where the y-axis represents the percentage of actual signal failures and the x-axis stands for the percentage of signals that have been screened in order. According to the failure percentage curve for January 2021, screening the top 10% of signals may identify around 35% of signal failures in advance.

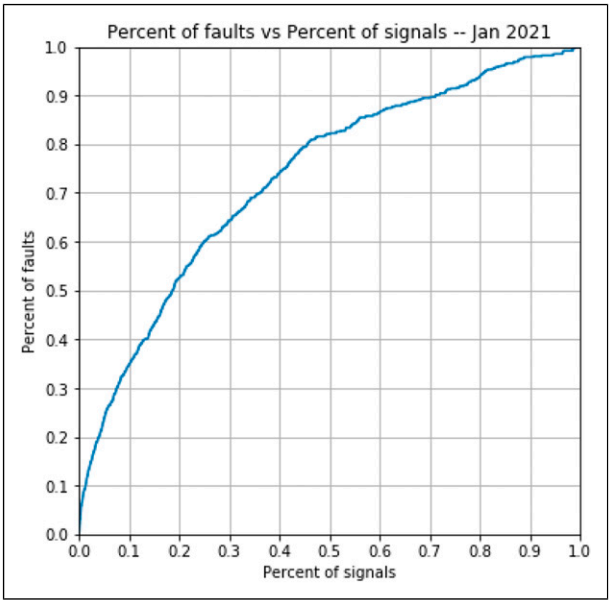

Figure 5. Failure percentage curves for January 2021 prediction.

Figure 6 (below) shows the ranking of the top 5 important features 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 6, 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 the most useful in constructing the model. This may indicate that some location-specific characteristics might affect signal failure occurrence. The feature "month" represents which month of the year we predict the signal failure. This is considered the second most important by the model and may be due to the extreme weather or equipment life span. The third feature, "days since last occurrence", refers to the number of days between the first day of the predicting month and the date when the most recent failure occurred. The fourth feature, "number of failures in past year", 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 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. 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,

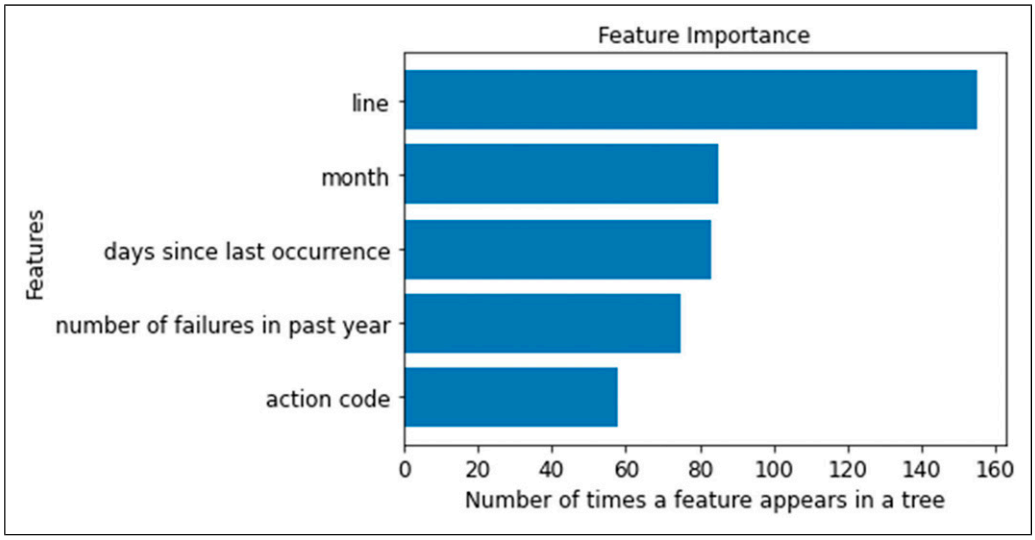

Figure 6. The most important 5 features for the proposed model.

Type) are all considered more important than the features generated by One-Hot Encoding.

Conclusions

This paper proposed an event-based data analysis method to generate features that can be better used in a machine learning model to predict urban rail transit signal failures. The proposed data processing approach consists of One-Hot Encoding, 5-fold Target Encoding, and window-based feature extraction. We validated this data processing approach using the XGBoost algorithm and Bayesian Optimization for hyperparameter selection. Four resampling methods: random oversampling, random undersampling, SMOTE, and ADASYN are also tested and compared to the model using the original imbalanced dataset. The random undersampling leads to the best AUC scores in combination with the XGBoost algorithm in this study. Our proposed method can capture about 35% of total failures from 10% of screened signal locations. By predicting rail transit signals that are prone to failure, rail companies can make a better inspection plan to achieve a balance between safety, efficiency, and economy. In future research, if preventative maintenance data is available, it would be interesting to compare the results of the following month when preventative maintenance is performed on the 10% of identified signals for the next month with those that do nothing. Furthermore, we have tested the random undersampling on Random Forests, Neural Network, and Logistic Regression, where the AUC scores were all slightly increased. This indicates that we could try some more sophisticated undersampling techniques in future studies. Additional information (e.g., equipment age, sensory information) may also be incorporated into model improvements for our next step.

Acknowledgement

This research is sponsored by a grant from the Center for Advanced Infrastructure and Transportation (CAIT), which is a USDOT University Transportation Center, at Rutgers University.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by USDOT Transportation Center

ORCID iD

Xiang Liu https://orcid.org/0000-0002-4348-7432

References

1. Alliance R. *The Bad, Old Normal: Subway Signal Problems Still Delayed Trains during 80. % of Morning Rush Hours in the First Half of 2021 [Internet]*. 2021 p. 3. https://static1.squarespace.com/static/61033b9bd377817f5bcc6db9/t/614e54a63c9104704b083efc/1635951289660/The+Bad%2C+Old+Normal%3A+Subway+Signal+Delays+-+Riders+Alliance
2. Fitzsimmons EG. *Key to Improving Subway Service in New York? Modern Signals*. The New York Times [Internet]. 2017; https://www.nytimes.com/2017/05/01/nyregion/new-york-subway-signals.html (Accessed on 2021 Nov 15).
3. Wiedmann W. *As Railroad Systems Advance, Wayside Signals Fade Away*. [Internet]. Burns Engineering. 2021. https://insights.burns-group.com/2021/07/29/as-railroad-systems-advance-wayside-signals-fade-away/ (Accessed on 2021 Oct 29).
4. O'Madadhain J, Hutchins J and Smyth P. Prediction and ranking algorithms for event-based network data. *ACM SIGKDD Explor Newsl* 2005; 7(2): 23–30.
5. Fernández A, García S, Galar M, et al. *Learning from Imbalanced Data Sets [Internet]*. Cham: Springer International Publishing, 2018. http://link.springer.com/10.1007/978-3-319-98074-4 (Accessed on 2021 Sep 29).
6. Chawla NV, Bowyer KW, Hall LO, et al. SMOTE: Synthetic Minority Over-sampling Technique. *J Artif Intell Res* 2002; 16: 321–357.
7. He H, Bai Y, Garcia EA, et al. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *IEEE International Joint Conference on Neural Networks*. IEEE World Congress on Computational Intelligence), 2008, pp. 1322–1328.
8. Menardi G and Torelli N. Training and assessing classification rules with imbalanced data. *Data Min Knowl Discov* 2014; 28(1): 92–122.
9. Provost F. Machine learning from imbalanced data sets 101. In: *Proceedings of the AAAI'2000 workshop on imbalanced data sets*. AAAI Press, 2000, pp. 1–3.
10. Chen T and Guestrin C. XGBoost: A Scalable Tree Boosting System. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining [Internet]*. San Francisco California USA: ACM, 2016. https://dl.acm.org/doi/10.1145/2939672.2939785 (Accessed on 2021 Oct 20).
11. Tu J, Tao Q and Deng Q. Safety evaluation of urban transit signal system based on the improved TOPIS. *Proced Eng* 2011; 15: 4558–4562.
12. Zhang YP, Xu ZJ and Su HS. Risk assessment on railway signal system based on Fuzzy-FMECA method. *Sens Transducers* 2013; 156(9): 203.
13. Ren W, Ma L and Wang Y. Monte Carlo analysis for safety and reliability of rail transit signal system based on Cloud Computing. *J Phys Conf Ser* 2020; 1654: 012065.
14. Yang L, Li P, Xue R, et al. Intelligent classification model for railway signal equipment fault based on SMOTE and ensemble learning. *IOP Conf Ser Mater Sci Eng* 2018; 383: 012042.
15. Gao F, Li F, Wang Z, et al. Research on Multilevel Classification of High-Speed Railway Signal Equipment Fault Based on Text Mining. *J Electr Comput Eng* 2021: 2021: e7146435.
16. Breiman L, Friedman JH, Olshen RA, et al. *Classification and Regression Trees*. Boca Raton: Routledge, 1984, p. 368.
17. Freund Y and Schapire RE. *Experiments with a New Boosting Algorithm*. icml. Citeseer, 1996, pp. 148–156.

18. Friedman J, Hastie T and Tibshirani R. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *Ann Stat* 2000; 28(2): 337–407.
19. Friedman JH. Greedy function approximation: a gradient boosting machine. *Ann Stat* 2001: 1189–1232.
20. Zhang Z, Zhou K and Liu X. *Broken Rail Prediction with Machine Learning-Based Approach*. American Society of Mechanical Engineers Digital Collection, 2020. https://asmedigitalcollection.asme.org/JRC/proceedings/JRC2020/83587/V001T08A014/1085551 (Accessed on 2022 Jun 27).
21. Shi R and Xu X. A Train Arrival Delay Prediction model using XGBoost and Bayesian Optimization In *IEEE 23rd International Conference on Intelligent Transportation Systems*. ITSC), 2020, pp. 1–6.
22. Krawczyk B. Learning from imbalanced data: open challenges and future directions. *Prog Artif Intell* 2016; 5(4): 221–232.
23. Ling CX and Li C. Data mining for direct marketing: Problems and solutions. In: *Kdd*, 1998, pp. 73–79.
24. Hasanin T and Khoshgoftaar T. The effects of random undersampling with simulated class imbalance for big data In *IEEE International Conference on Information Reuse and Integration (IRI)*. IEEE, 2018, pp. 70–79.
25. Prusa J, Khoshgoftaar TM, Dittman DJ, et al. Using random undersampling to alleviate class imbalance on tweet sentiment data In *IEEE International Conference on Information Reuse and Integration*. IEEE, 2015, pp. 197–202.
26. Swets JA. Measuring the Accuracy of Diagnostic Systems. *Science* 1988; 240(4857): 1285–1293.
27. Potdar K, Pardawala TS and Pai CD. A comparative study of categorical variable encoding techniques for neural network classifiers. *Int J Comput Appl* 2017; 175(4): 7–9.
28. Micci-Barreca D. A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. *ACM SIGKDD Explor Newsl* 2001; 3(1): 27–32.
29. Grover P. *Getting Deeper into Categorical Encodings for Machine Learning*. https://towardsdatascience.com/getting-deeper-into-categorical-encodings-for-machine-learning-2312acd347c8 (Accessed on 2021 Oct 19).
30. *Pourya. K-Fold Target Encoding [Internet]*. Medium. 2019 https://medium.com/@pouryaayria/k-fold-target-encoding-dfe9a594874b (Accessed on 2021 Oct 19).
31. Dattagupta SJ. *A Performance Comparison of Oversampling Methods for Data Generation in Imbalanced Learning Tasks [PhD Thesis]*, 2018.
32. *XGBoost Documentation — Xgboost 1.5.0-dev Documentation* https://xgboost.readthedocs.io/en/latest/index.html#
33. *scikit-learn: machine learning in Python — scikit-learn 1.0 documentation [Internet]*. https://scikit-learn.org/stable/index.html

APPENDICES

Appendix 1: Hyperparameters and bayesian optimization ranges

Model	Hyperparameter	Range
XGBoost	learning_rate	(0, 0.5)
	n_estimators	(100, 1000)
	max_depth	(3, 15)
	min_child_weight	(0, 10)
	Gamma	(0, 10)
	Subsample	(0.6, 1)
	colsample_bytree	(0.6, 1)
RN	n_estimators	(100, 1000)
	max_depth	(3, 15)
	min_samples_split	(2, 20)
	max_features	(0.1, 0.99)
NN	learning_rate_init	(0, 0.5)
	neurons_layer_1	(10, 200)