Journal of Transportation Technologies, 2023, 13, 1-17
https://www.scirp.org/journal/jtts
ISSN Online: 2160-0481
ISSN Print: 2160-0473

Failure Prediction and Intelligent Maintenance of a Transportation Company's Urban Fleet

Crépin Foké, Jean-Pierre Kenné, Ngongang Somen Bill Diego

Mechanical Engineering Department, University of Quebec, École de Technologie Supérieure, Montreal, Canada
Email: jean-pierre.kenne@etsmtl.ca

How to cite this paper: Foké, C., Kenné, J.-P. and Diego, N.S.B. (2023) Failure Prediction and Intelligent Maintenance of a Transportation Company's Urban Fleet. *Journal of Transportation Technologies*, **13**, 1-17.
https://doi.org/10.4236/jtts.2023.131001

Received: July 24, 2022
Accepted: January 8, 2023
Published: January 11, 2023

Abstract

The present work deals with intelligent vehicle fleet maintenance and prediction. We propose an approach based primarily on the history of failures data and on the geographical data system. The objective here is to predict the date of failures for a fleet of vehicles in order to allow the maintenance department to efficiently deploy the proper resources; we further provide specific details regarding the origins of failures, and finally, give recommendations. This study used the Société de transport de Montréal (STM)'s historical bus failure data as well as weather data from Environment Canada. We thank Facebook's Prophet, Simple Feed-forward, and Beats algorithms (Uber), we proposed a set of computer codes that allow us to identify the 20% of buses that are responsible for the 80% of failures by mean of the failure history. Then, we deepened our study on the unreliable equipments identified during the diffusion of our computer code This allowed us to propose probable predictions of the dates of future failures. To ensure the validity of the proposed algorithm, we carried out simulations with more than 250,000 data. The results obtained are similar to the predicted theoretical values.

Keywords

Maintenance 4.0, Digital Technologies, Failureprediction, Artificial Intelligence Artificial Intelligence, Prediction Algorithm

1. Introduction

Industrial maintenance has seen a figurative expansion by mean of the advent of big data. When discussing big data in industry, the topic of predictive maintenance comes up again and again, with essentially smart sensors connected to the internet being placed in industrial products (engines, mechanisms, structures…)

and tasked with measuring and sending useful information to improve the reliability (predicting and limiting failures) of the industrial system. Historically, industrial maintenance started as a necessary evil, and therefore maintenance operations were only performed when strictly necessary. However, in industrial manufacturing environments, which are often characterised as stochastic, dynamic and chaotic, maintenance is crucial for production efficiency. Unexpected disturbances lead to a degradation of system performance, causing a loss of productivity and business opportunities, which must be optimized as a prelude to achieving competitiveness [1]. Like any other motorized object, the automobile is a device subject to various failures. The automotive sector has experienced a strong technological growth in recent years with the advent of smart sensors. But this has increased the complexity of automotive maintenance because on the one hand it involves the use of diagnostic tools such as: infrared engine analysers, spark plug checkers and compressometers. On the other hand, it involves a lot of data. In this article we make use of data analysis and processing software that will allow us to effectively exploit the history of failure data. This is called Maintenance 4.0—or intelligent maintenance, which consists of using intelligent sensors connected to the Internet on machines. And so the strategies can predict future failures and therefore reduce corrective maintenance and associated costs. Thus, the objective of this paper is to provide the automotive maintenance process with more effective and efficient based on the exploitation of historical failure data using artificial intelligence, deep learning and machine learning Therefore, we will establish a model to predict the equipment failure time through an approach similar to that of the work of Chen *et al*. [2]. The rest of the paper is organized as follows: Section 2 presents a review of the literature. Section 3 presents the methodology used. Section 4 presents the performance and choice of the algorithm used in the study, and finally, Section 5 concludes the work.

2. Literature Review

The automotive sector, like any other industrial sector, has developed over time and the practice of the two main maintenance strategies (preventive and curative) has improved the reliability of equipment. These strategies, although revolutionary, do not allow the prediction and resolution of future failures. To remedy this, new predictive and intelligent maintenance strategies have been developed. This is known as Maintenance 4.0.

According to Mobley [3], preventive maintenance is a set of actions carried out regularly on equipment that is still functional to reduce its failure. Predictive maintenance, on the other hand, is a philosophy or attitude that uses the actual operating conditions of the plant's equipment and systems to optimise the plant's operations and/or processes. The illustration of BPMN (Business Process Model and Notation), which defines the sequences of different maintenance tasks, is presented in the work of Cachada [1]. As a result, the ability to predict

the failure time of a car allows for better optimization of maintenance and fleet management.

Intelligent maintenance is little explored and implemented in the automotive sector. Some researchers have proposed an intelligent approach based on a neural network. It consists of collecting geographical and historical breakdown data. An algorithm then predicts the breakdowns of equipment in a vehicle fleet. Among these algorithms is the Multi-Grained Cascade Forest (gcForest), which was originally developed for image classification and subsequently became a useful tool for automotive maintenance [2]. These researchers conducted a study to introduce gcForest into automotive maintenance based on maintenance history data from geographic information systems (GIS). This algorithm has also been implemented for the study of prediction of structured behaviour of road vehicles. Mou *et al*. [4] proposed a structured gcForest modeling approach for the prediction of road vehicle behavior. According to Liu *et al*. [5], deep neural networks (DNNs) for intelligent machine failure diagnosis require a large amount of training data, powerful computational resources, and many hyperparameters, which must be carefully tuned to ensure maximum performance. Deep forest, as a new alternative to the deep learning framework, has the potential to overcome these shortcomings. The authors acknowledge the performance of this DNN-based diagnostic approach. However, they point out some drawbacks, including the need for a large amount of training data, as well as the fact that DNNs have many hyperparameters that need to be carefully tuned to ensure maximum performance. For Hu *et al*. [6], the essence of intelligent industrial fault diagnosis based on big data lies in the process of machine learning and feature engineering. Deep learning methods to establish the complex relationship between data and potential faults and outperform traditional machine learning methods. Hu *et al*. [6] proposed an approach that combines a deep Boltzmann machine and a multigrain forest for fault diagnosis. The deep Boltzmann machine was first used to transform raw data into a binary representation. Then, the gcForest was deployed for modeling based on the preprocessed data. Liu *et al*. [7] deployed a gcForest for the recognition of ball bearing defects based on sensor data. At the end of the experiments, they achieve good defect recognition accuracy. The previous literature illustrates the performance and efficiency of the gcForest algorithm in the manufacturing sector. Therefore, we propose to start with the Prophet, Simple Feed-forward and Beats (Uber) algorithms from Facebook, which are similar algorithms to gcForest. As for the Prophet algorithm, it is an open-source library (R and Python) for forecasting time series data based on an additive model: The difficulty of creating reliable forecasting models, because this discipline requires a particular experience. The rigidity and lack of robustness of automatic forecasting techniques Prophet has been tested in several research studies with satisfactory results [8]. The authors Žunić *et al*. [9] deployed this algorithm in a sales forecasting study based on real-world data. In their paper, they present a framework capable of accurately predicting future

sales in the retail sector and classifying the product portfolio according to the expected level of forecast reliability. The algorithm has demonstrated the ability to generate sales forecasts and a high potential to classify the product portfolio into several reasonably accurate quarterly categories; it has also shown a high potential to classify the product portfolio into several categories. The results obtained are more than satisfactory with respect to the calculated error. Zhoul-Chenl and NI [10] deployed Prophet for the prediction of the air quality index. In their paper, they designed a hybrid model Prophet-SVR and a hybrid Prophet-LSTM model to optimize the prediction accuracy of the Prophet model. The results show that the Prophet-LSTM hybrid model has the best performance, that its prediction accuracy is higher than that of the single model, and that it has clear advantages in air quality index prediction. Panicker and Valarmathi [11] showed after a study in four different regions that the Prophet model gave more reasonable results than the SARIMA model. The MAPE and RMSE values were the lowest for the Prophet algorithm and its fast execution.

Gong *et al.* [12] and Septiani *et al.* [13] deployed Prophet to analyze the power consumption trends of buildings. The comparison of the prediction results of Prophet and ARIMA algorithms shows that Prophet performs better and can consider dates and times. Septiani *et al.* [13] show that Prophet performs better than FFNN. However, the difference in the MAPE value is not too significant.

As for the simple feed-forward algorithm, its performance has been demonstrated in the work of several researchers. Melnychuk *et al.* [14] in their paper on predicting attentional focus from breathing, revealed that it is possible to predict to a moderate degree, the state of attentional control from breathing.

Cohen *et al.* [15] examined the universal approximation function for learning ranking. They have replaced costly regression forests with simple feed-forward networks feed-forward algorithms, demonstrates the advantages of feed-forward algorithms. They stipulate that this neural approach has the advantage of being easier to train than any other neural model since it can match the previously learned regression rather than learning to generalize relevance judgments directly. According to the authors, feed-forward also has runtimes of up to 400× that of traditional algorithms and up to 10× that of state-of-the-art algorithms, with no measurable loss in average accuracy.

In their study on simple fuzzy direct torque control for a DSP-based induction motor, Jat *et al.* [16] found that the fuzzy logic controller (FLC) provides better and smoother speed control. The speed control technique is implemented using a next-generation TMS320F28335 digital signal processor (DSP) with a high sampling rate to enable the sampling rate to achieve a good dynamic range. The result shows the supremacy of the FLC in the control of the three-phase induction motor when using the DTC technique.

3. Methodology

The following **Figure 1** presents the methodology that we followed throughout

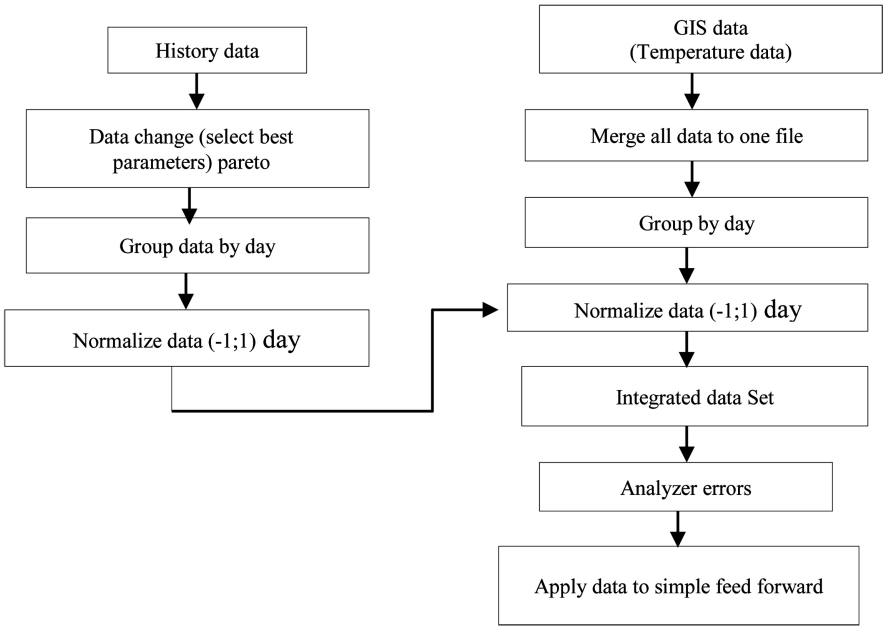


Figure 1. Study methodology.

our study.

Any time series prediction work using machine learning can be broken down into clear and easily replicated steps. We will start by performing data analysis and data mining to seek out hidden information in the data and determine which algorithms to use.

3.1. Data Presentation

The data file provided by the STM is in Excel format, with several pages that can be grouped into 2 main groups:

- The group of failures on additional periods.
- The group of failure entries by equipment, type, date.

The information we are interested in are the failure entries that constitute the raw data recorded by data entry agents or by on-board systems for monitoring the state of the monitoring of the bus status in real-time. What is interesting here is that each input is followed by a summary description and definition of an internal failure group used by the STM (engine, brakes, etc.). The Excel file contains more than 175,000 entries, which is more than enough to apply deep learning algorithms that are known to be data-intensive and whose efficiency is most often the directly linked to the data provided in the input. The procedure followed for the analysis of these raw data is described in the following subsection.

3.2. Data Analysis

To carry out our data analysis, we used BI (Business Intelligence) software and placed all the parameters of the received CSV file into sensitivity analysis graphs. The quality being mainly quantitative, we used as main parameter of compari-

son, the number of occurrences of a failure on a well-defined period of time according to several temporal granularities.

The graphs obtained by these tools are presented and analyzed next.

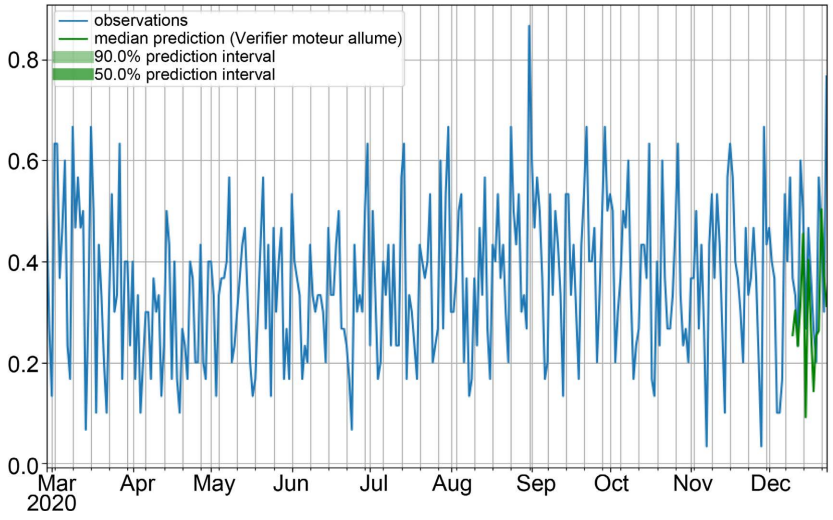
3.2.1. Graphs

The graphs below were obtained using the simple feed-forward algorithm from the Gluon toolkit for probabilistic time series modeling focused on deep learning-based models. **Figure 2(a)** represents the graphical behavior of the number of reports of the check engine parameter for the period of March to December 2020 and **Figure 2(b)** illustrates the curve of predictions made from the month of December. The blue graph represents the actual behavior, and the green represents the prediction according to the graph legend. The predictions were made over 14 days. For this, we trained the data for 135 days, and then using our code written for this purpose, the algorithm generated the forecast over the last 14 (of the 135) days. **Figure 2(c)** shows the 90% forecast graph and **Figure 2(d)** shows the other 50% graph segments. We also varied another parameter (NFP) in the forecast area at 90% and 50% (see **Figure 3**). Looking at **Figure 2(c)**, **Figure 2(d)** and **Figure 3** below, we can safely conclude that our forecast illustrates reality as all our curves fall within the real prediction area.

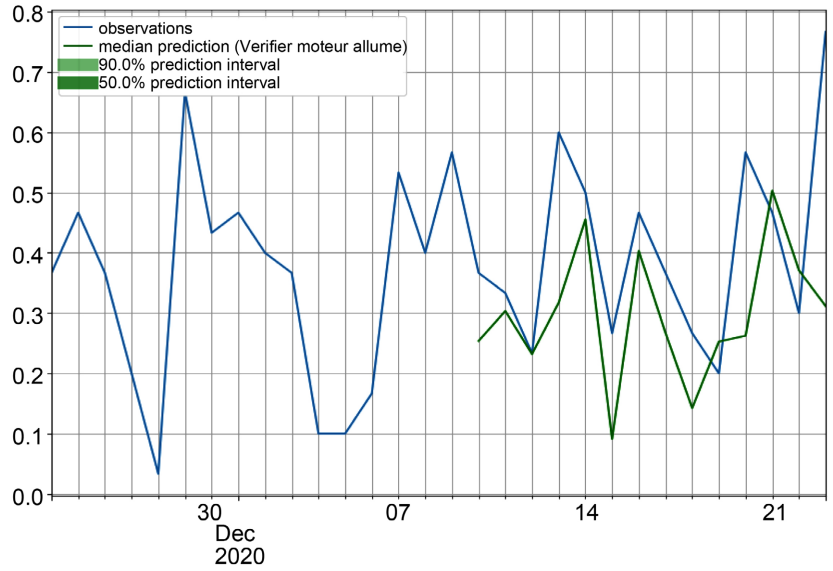
Figure 4 below presents the graphical trends of the failures obtained with Facebook's Prophet algorithm. We note a significant growth from Sunday to Monday, a slight growth from Monday to Friday, and then a significant decrease from Friday to Saturday. Currently, it is hard to make any conclusions regarding the behavior since we do not have enough information that are internal to the company. However, we believe that the increase in reporting between Sunday and Monday could be due to the large number of trips in the period, as well as to the fact that it is the start of the week. The decrease between Friday and Saturday could be explained by the fact that it is the end of the week, and therefore, there is less travel. As stated at the beginning, this hypothesis has no solid scientific basis. it is simply a proposal for a model based on the graph. From January to March, we observe a semblance of stability, and then from March to May, we reach the peak of reporting and a slight stability is seen, but it gradually decreases until January. Only information on transportation planning within the company could explain this behavior. This also applies to the daily reporting behavior. Although we do not have enough information about the graphical behavior, especially from Sunday to Monday, it would be relevant for the maintenance department to review the status of the buses in circulation during these periods.

3.2.2. Pareto

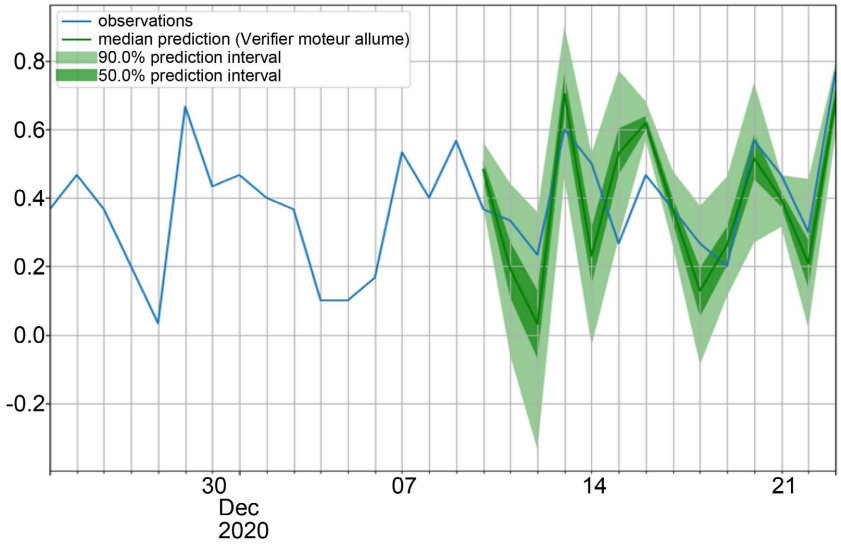
In an effort to avoid scatter in our study, we used a Pareto analysis (shown in **Figure 5**) to determine the groups of failures with the most occurrences in STM buses. The remainder of this study focuses on the group designated by Pareto as responsible for 80% of the reporting.


observations
median prediction (Verifier moteur allume)
90.0% prediction interval
50.0% prediction interval
0.8
0.6
0.4
0.2
0.0
Mar
2020
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

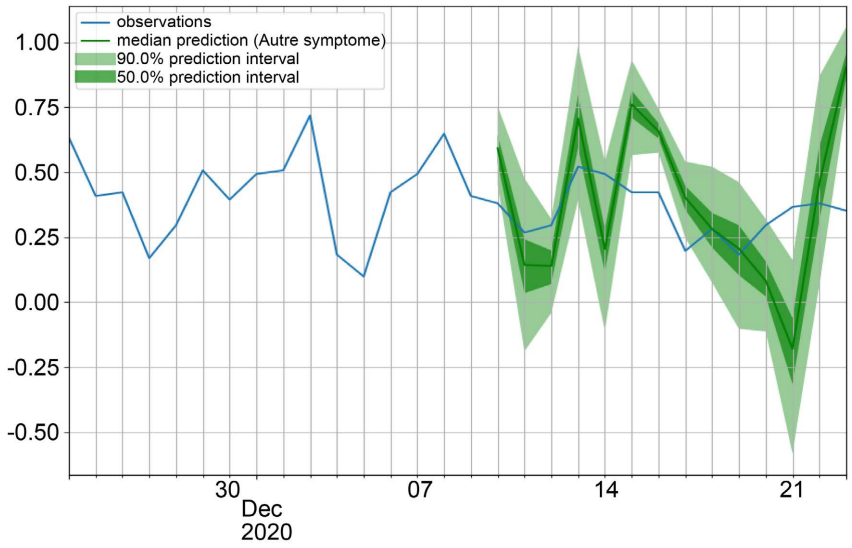
(a)


observations
median prediction (Verifier moteur allume)
90.0% prediction interval
50.0% prediction interval
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
30
Dec
2020
07
14
21

(b)


observations
median prediction (Verifier moteur allume)
90.0% prediction interval
50.0% prediction interval
0.8
0.6
0.4
0.2
0.0
-0.2
30
Dec
2020
07
14
21

(c)



(d)

Figure 2. (a) Check engine on graph, including the training period of the data. (b) Forecast graph of the check engine on parameter (November and December 2020). (c) Prediction graph of the check engine on parameter (November to December 2020, large scale). (d) Prediction graph of the other symptom parameter.

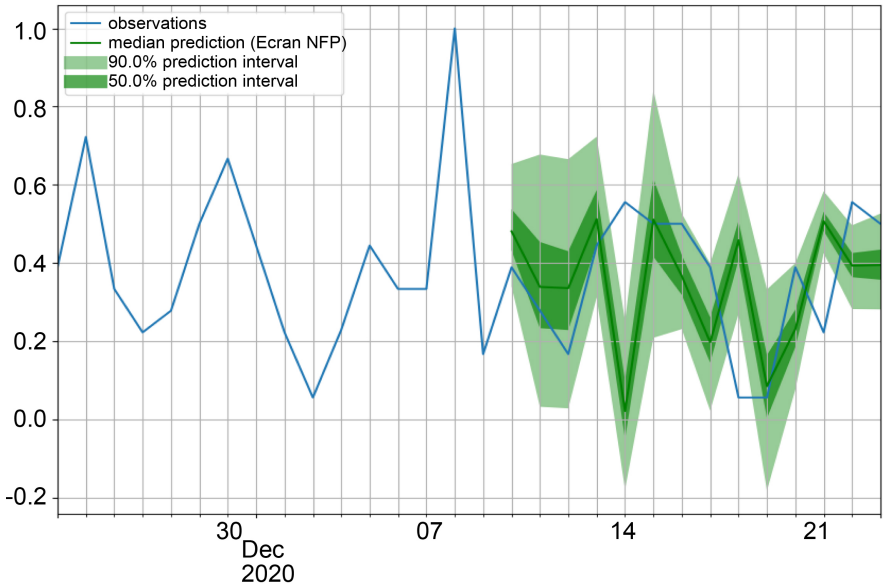


Figure 3. Forecast graph of the NFP screen parameter.

3.2.3. Selection of the Algorithms

Several algorithms are known to be efficient for predicting data of the same type as those selected above. Our data are time series with varying trends, and it is not obvious what the seasonality is. Three main criteria will allow us to select the three algorithms that we will use, namely:

- The simplicity of the implementation of the algorithm in an academic research context.
- The ability of the algorithm to adapt to variations in seasonality and to support

Figure 4. Graphical trend of failures.

the drift of concepts in the data.

- The ability of the algorithm to make predictions on several (linked) time series simultaneously.

The following three algorithms emerge from these criteria:

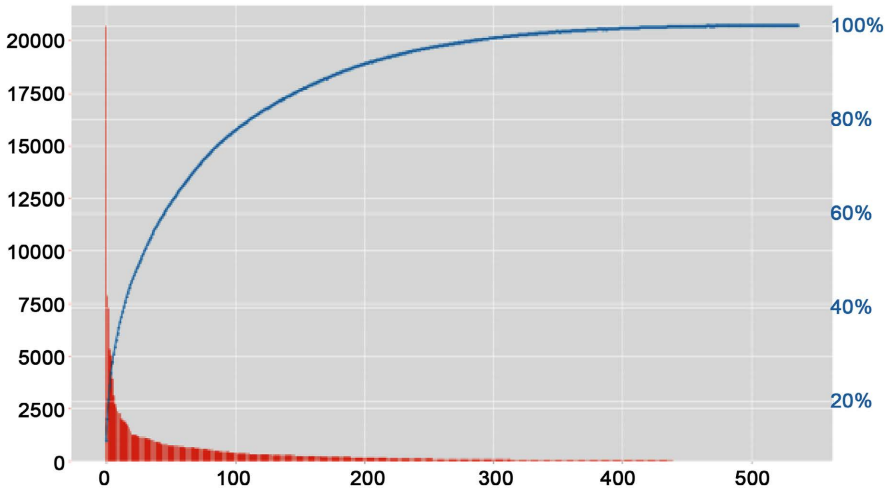


Figure 5. Pareto curve.

- Simple Feed-Forward: known for its simplicity and for its sometimes-surprising performance compared to other more complex algorithms.
- Facebook Prophet: used by Facebook; performance is always surprising even when it is not trained and does not require complex configurations.
- N BEATS (Uber): famous for having won the M8 forecasting competition and for being used by Uber; its architecture is interesting in that it allows computerizing predictions by combining residual information from past and previous predictions. To ensure that our algorithms would work, we proceeded first with a data preparation procedure, as presented next.

3.3. Data Preparation

We divided our data preparation procedure into three components. We begin by presenting the first one, which is very important, given the high number of failure occurrences in the STM data.

3.3.1. Normalization

Normalization is a procedure that allows taking data of several different orders of magnitude and then reducing them to a single identical order of magnitude. We used a simple formula for data normalization. The defined function was applied individually to each dataset, grouped by day of occurrence:

$$N_i = \frac{n_i}{Max(n_i)}, i \in [1, \text{number}_{\text{occurences}}] \quad (1)$$

3.3.2. Replacement of Null Values

It is practically impossible to not have null values when considering several study parameters. As our data was arranged by day of occurrence, there are days with no values for some parameters. To avoid divergence among our algorithms or to bias the training of the latter on our data, we decided to replace the null data by the average of the values for the parameters. Once our null data management strategy was applied, we packaged our data as a CSV file, with a partic-

ular architecture that we will discuss in the next section.

3.3.3. Pandas DF

Our dataset is a Pandas Dataframe having as index the dates per day of occurrence and as columns the normalized number of different failures that will constitute the values to be predicted for a precise time window. This file was exported to CSV format (see **Appendix A**) to make easily shareable and to avoid having to repeat our whole data transformation procedure in the event of problems during the training sessions.

3.3.4. Library Used

To consolidate our testing method, we decided to use a framework which provides an identical API for all the algorithms needing to be tested. This framework, called GluonTS, is open-source. To train our neural networks, we followed the procedure defined in the next section.

3.3.5. Training

The dataset was divided into 2 parts: training data and test data. The training data made up 80% of our data, and the test data, 20%. GluonTS allows us to define an estimator and to pass training parameters to it with a trainer. Once trained on data, this estimator is called a predictor and can be used for prediction or performance evaluation. During training, we defined several hyperparameters and selected the combination giving the best results for the accuracy metric, and during testing, for the evaluation of our algorithms, we used metrics whose strategy we present in the next section.

3.3.6. Metrics Used

One major advantage of the Gluonts framework is that it already includes some metrics, and we just had to apply them to our data.

4. Algorithm Performance

As mentioned before, we conducted this study on three distinct algorithms, namely, Facebook Prophet, Simple Feed-forward and the Nbeasts. The goal was to select the algorithm offering the best performance. To evaluate the performance of our algorithms, we focused on different error terms, namely, MSE (mean squared error), MASE (mean absolute scaled error), MAPE (mean absolute percentage error), and SMAPE (symmetric mean absolute error). The MASE defines the scaled mean absolute scaled error which is a measure of the accuracy of forecasts. It is the mean absolute error of the forecast values divided by the mean absolute error of the naive one-step forecast in the sample. The mean absolute scaled error has properties that compare very favorably against other methods of calculating forecast errors, such as the root mean square error, and is therefore recommended for determining the comparative accuracy of forecasts.

MASE has the following advantages:

- Scale invariance: the mean absolute error at scale is independent of the scale of the data, and thus can be used to compare forecasts between data sets with different scales. Not sure what is meant here.
- For predictable behavior noted that y_t tending to 0, percentage measures of forecast accuracy, such as the mean absolute percentage error (MAPE), rely on dividing y_t, which distorts the MAPE distribution. This is particularly problematic for data sets with different scales of error. Data sets with scales that do not have significant (temperature in Celsius or Fahrenheit) and for intermittent data sets, where y_t intermittent, where y_t occurs frequently.
- Symmetry: the scaled mean absolute error penalizes the same way equally the positive and negative forecast errors and penalizes the same way errors in both large and small-large forecasts and small forecasts. In contrast, the MAPE and the median absolute error (MdAPE) do not meet either criterion, while MAPE and SMAPE and "symmetric" SMdAPE do not meet the second criterion.

Interpretability: the scaled mean absolute error can be easily interpreted, as values greater than one indicate that the one-step predictions of the naive method are better than the values it is calculated by the expression by:

$$\mathrm{MASE} = \left(\frac{\frac{1}{j}\sum_j \left| e_j \right|}{\frac{1}{T-m}\sum_{t=m+1}^{T} \left| y_t - y_{t-m} \right|} \right) \tag{2}$$

where e_j is the forecast error for a given period, J is the total number of forecasts, and m is the seasonal period.

It is important to remember that this formula is only valid for seasonal time series as is the case in this study. For non-seasonal time series for non-seasonal time series, it is sufficient to replace the "m" with "1". As for MSE, the authors Lin, *et al.* [17] point out the relevance of MSE in prediction algorithms.

The mathematical expression of MSE is given by:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n} \left(Y_i - \hat{Y}_i \right)^2 \tag{3}$$

where Y is the vector of observed values of the variable to be predicted and Y is the predicted values (e.g., from a least-squares fit). Despite the multiple advantages that the MASE has in terms of prediction, our study reveals, however, that the MSE is more adequate with respect to the results obtained. This allows to state without any risk of error that the MASE is not adequate for the type of data used in this study. The performance of MASE is not being into question here. Rather, it is simply being indicated that MASE would be more appropriate for specific data.

It should be stated that some values of MASE, MAPE, and SMAPE are infinite in the context of this study, which is true from a mathematical point of view, when their denominator tends to 0. However, that is unacceptable when it comes to forecasting work such as the present one. But this can be explained by

Table 1. Comparison of the different metrics used.

	Simple feed-forward	N beasts (Uber)
MSE	5.52727484	5.56861382
MASE	1148.00867	1884.88973
MAPE	120.511948	105.325813
SMAPE	2537.04123	3980.43109

infinitesimal values close to zero. So, considering the results of Table 1, based only on the sum of the MSE, we retained the simple feed-forward algorithm for our forecasting study. Although the Prophet algorithm was used to generate our Pareto curve, we found it to be inefficient for obtaining forecast curves because the algorithm does not calculate errors. It is therefore impossible to discuss its performance moreover, unlike the simple feed, Prophet does not generate forecast percentages, as observed on the simple feed graphs, which makes the graphs difficult to interpret. Figure 6 above is the prediction graph for the "touch screen does not work" parameter obtained from the Prophet algorithm.

From the studies conducted in the context of this project, we were able to propose an intelligent maintenance approach based primarily on historical failure data and on the geographical data system. This new approach is less expensive than those used in previous models in the literature. Those models necessarily require the presence of various sensors. By exploiting historical failure data with different algorithms, we were able to predict future failures with prediction percentages ranging from 50% to 90%.

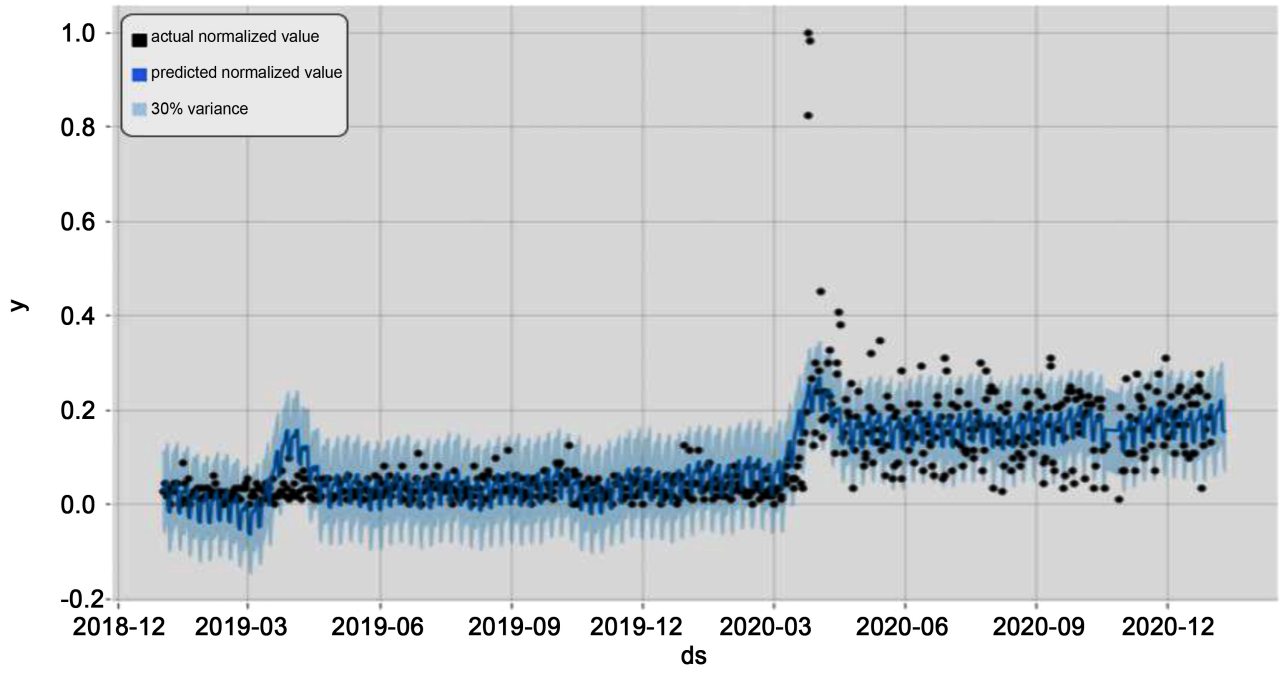


Figure 6. Prediction of "touch screen not working" parameter.

5. Conclusion

Definitely, it was a question of this study, to see how to optimize the means of the automobile maintenance in order to make it more powerful and effective. It is thus that we have by means of artificial intelligence, the deep learning, and the machine learning to establish a model of prediction of the failure of an equipment starting from historical data of maintenance. The prediction model established should be able to predict the date of failure of a vehicle using an algorithm. To meet this goal, we used the failure history of buses in the Société de transport de Montréal (STM)'s transit fleet. The simulation of its data in the simple feed-forward algorithm allowed us to obtain a prediction model. For the calculated error, *i.e.*, the MSE (mean squared error), and the forecast graphs obtained, we were able to conclude the efficiency of the results. By closely analyzing the graph of the "check engine on" failure, we note that some results are predicted at nearly 90%, which allows us to validate our prediction model. However, when this study was being conducted, we did not have information on the makes, models, and ages of the buses used in this work. The addition of this information could be the subject of future studies and could help explain the behavior of certain equipment with the greatest precision.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Cachada, A., Barbosa, J., Leitão, P., *et al.* (2018) Maintenance 4.0: Intelligent and Predictive Maintenance System Architecture. *IEEE* 23*rd International Conference on Emerging Technologies and Factory Automation* (*ETFA*), Turin, 4-7 September 2018, 139-146. https://doi.org/10.1109/ETFA.2018.8502489

[2] Chen, C., Liu, Y., Sun, X.F., *et al.* (2020) Automobile Maintenance Modelling Using gcForest. 2020 *IEEE* 16*th International Conference on Automation Science and Engineering* (*CASE*), Hong Kong, 20-21 August 2020, 600-605. https://doi.org/10.1109/CASE48305.2020.9216745

[3] Mobley, R.K. (2002) An Introduction to Predictive Maintenance. 2nd Edition, Butterworth-Heinemann, Oxford. https://doi.org/10.1016/B978-075067531-4/50006-3

[4] Mou, L.T., Mao, S.S., Xie, H.T. and Chen, Y.Y. (2019) Structured Behaviour Prediction of On-Road Vehicles via Deep Forest. *Electronics Letters*, **55**, 452-455. https://doi.org/10.1049/el.2019.0472

[5] Liu, X.L., *et al.* (2019) Deep Forest Based Intelligent Fault Diagnosis of Hydraulic Turbine. *Journal of Mechanical Science and Technology*, **33**, 2049-2058. https://doi.org/10.1007/s12206-019-0408-9

[6] Hu, G.Z., Li, H.F., Xia, Y.Q. and Luo, L.X. (2018) A Deep Boltzmann Machine and Multi-Grained Scanning Forest Ensemble Collaborative Method and Its Application to Industrial Fault Diagnosis. Elsevier, Amsterdam. https://doi.org/10.1016/j.compind.2018.04.002

[7] Liu, Q., Gao, H.L., You, Z.C., *et al.* (2018) Gcforest-Based Fault Diagnosis Method

For Rolling Bearing. 2018 *Prognostics and System Health Management Conference*, Chongqing, 26-28 October 2018, 572-577. https://doi.org/10.1109/PHM-Chongqing.2018.00103

[8] Taylor, S.J. and Letham, B. (2017) Prophet: Forecasting at Ccale. Computer Science. https://doi.org/10.7287/peerj.preprints.3190v2

[9] Žunić, E., Korjenić, K., *et al*. (2020) Application of Facebook's Prophet Algorithm for Successful Sales Forecasting Based on Real-World Data. *International Journal of Computer Science & Information Technology*, **12**, 23-36. https://doi.org/10.5121/ijcsit.2020.12203

[10] Zhoul, L., Chenl, M. and Ni, Q.J. (2020) A Hybrid Pophet—LSTM Model for Prediction of Air Quality Index. 2020 *IEEE Symposium Series on Computational Intelligence* (*SSCI*), Canberra, 1-4 December 2020, 595-601. https://doi.org/10.1109/SSCI47803.2020.9308543

[11] Panicker, V. (2021) AOD Forecasting Using Prophet Model across Four Major Urban Areas in India. 2021 6*th International Conference on Wireless Communications, Signal Processing and Networking* (*WiSPNET*), Chennai, 25-27 March 2021, 400-405. https://doi.org/10.1109/WiSPNET51692.2021.9419423

[12] Gong, F.X., Han, N.H., Li, D.Z. and Tian, S.M. (2020) Trend Analysis of Building Power Consumption Based on Prophet Algorithm. 2020 *Asia Energy and Electrical Engineering Symposium* (*AEEES*), Chengdu, 29-31 May 2020, 1002-1006. https://doi.org/10.1109/AEEES48850.2020.9121548

[13] Septiani, R., *et al*. (2021) Applied of Feed-Forward Neural Network and Facebook Prophet Model for Train Passengers Forecasting. *Journal of Physics*: *Conference Series*, **1776**, Article ID: 012057. https://doi.org/10.1088/1742-6596/1776/1/012057

[14] Melnychuk, M.C., Murphy, P.R., Robertson, I.H., *et al*. (2020) Prediction of Attentional Focus from Respiration with Simple Feed Forward and Time Delay Neural Networks. Springer Science and Business Media, Berlin. https://doi.org/10.1007/s00521-020-04841-7

[15] Cohen, D., Foley, J., Zamani, H., *et al*. (2018) Universal Approximation Functions for Fast Learning to Rank: Replacing Expensive Regression Forest with Simple Feed-Forward Networks. Association for Computing Machinery, New York. https://doi.org/10.1145/3209978.3210137

[16] Jat, G.L., Mahajan, A., Singh, K. and Shimi, S.L. (2016) A Simple Feed Forward Fuzzy Direct Torque Control of DSP Induction Motor Drive. 2016 *International Conference on Electrical Power and Energy Systems* (*ICEPES*), Bhopal, 14-16 December 2016, 284-289. https://doi.org/10.1109/ICEPES.2016.7915944

[17] Lin, Z.N., Chen, M.Y., Zhan, J. and Yao, W. (2020) Prediction Algorithm for Micro-Loudspeaker Diaphragm Displacement Based on PG-LSTM with 44% MSE Reduction. 2020 *IEEE International Conference on Integrated Circuits, Technologies and Applications* (*ICTA*), Nanjing, 23-25 November 2020, 104-105. https://doi.org/10.1109/ICTA50426.2020.9332102

Appendix A. Panda Import Methodology

0.1 STM data analytics

[1]:
```
import pandas as pd
from datetime import datetime
from dateutil.parser import parse as date_parser
from fbprophet import Prophet
import pystan
import matplotlib.pyplot as plt
import numpy as np

%matplotlib inline

plt.rcParams['figure.figsize']=(20,10)
plt.style.use('ggplot')
```

[2]:
```
from matplotlib import pyplot as plt
from matplotlib.dates import MonthLocator, num2date
from matplotlib.ticker import FuncFormatter
```

[3]:
```
def pad_data(number):
    if number < 10:
        return f'0{number}'
    return number


def excel_to_date(excel_date):
    el = datetime.fromordinal(datetime(1900, 1, 1).toordinal() + excel_date - 2)
    return f"{el.year}-{pad_data(el.month)}-{pad_data(el.day)}"
```

[4]:
```
weather_mtl_2019 = pd.read_csv('data/meteo_mtl_2019.csv')
weather_mtl_2020 = pd.read_csv('data/meteo_mtl_2020.csv')

weather_mtl_2019_20 = weather_mtl_2019.append(weather_mtl_2020)[["Date/Time",␣
 ↪"Mean Temp (°C)"]]
stm_data = pd.read_csv('stm_data.csv', low_memory=False)[["Date de l'avis",␣
 ↪'Texte du code']]
```

Appendix B. Program to Obtain the Pareto Graph

0.1.3 List of the top 20

[8]:
```
df.head(20)
```

[8]:

	count	labels	cumpercentage
0	20672	Autre symptome	11.681406
1	7875	Verifier moteur allume	16.131438
2	7284	Ecran tactile nfp correctement (allume)	20.247507
3	5291	ORDINATEUR Ne fonctionne pas du tout ROUTEUR	23.237363
4	5021	Inoperant(s)	26.074648
5	3923	N'indique pas correctement	28.291470
6	3108	Ne fonctionne pas du tout	30.047750
7	2690	Ecran NFP	31.567824
8	2512	Avant centre lache	32.987314
9	2352	Arret moteur allume	34.316390
10	2273	Ecran tactile indiqu un defaut system	35.600825
11	2262	Ne communique pas	36.879044
12	1973	Donne des coups entre les vitesses	37.993954
13	1933	Bouton demande d'arret inoperant	39.086260
14	1878	N'essuie pas bien	40.147487
15	1865	BPA probleme electrique	41.201368
16	1750	Fuite de liquide refroidissement	42.190264
17	1687	A arrete sur la route, ne demarre plus	43.143559
18	1565	Bruyant(e)	44.027915
19	1400	Batterie allume	44.819032

[9]:
```
for code in list(sorted_dict.keys())[1:20]:
    tmp = stm_data.loc[(stm_data[code_title] == code)].copy()
    tmp.columns = [index_name, 'ticket_code']

    for index, row in tmp.iterrows():
        item = row[index_name]
        if '/' in item:
            dt = date_parser(item)
            row[index_name] = f"{dt.year}-{pad_data(dt.month)}-{pad_data(dt.
↪day)}"
        if '/' not in item[1]:
            row[index_name] = f'{excel_to_date(int(item))}'

    grouped = tmp.groupby(['ticket_code', index_name]).size().
↪reset_index(name=f'count_{code}')
    grouped = grouped.drop(['ticket_code'], axis=1)

    if merged_df is None:
        merged_df = grouped.copy()
    else:
        if index_name in merged_df.columns:
```