Performance evaluation of machine learning techniques for fault diagnosis in vehicle fleet tracking modules

Luis Sepulvene[1], Isabela Drummond[1], Bruno Kuehne[1], Rafael Frinhani[1], Dionisio Leite Filho[2], Maycon Peixoto[3], Stephan Reiff-Marganiec[4] and Bruno Batista[1]

[1]*Federal University of Itajubá, Itajubá, Brazil*
[2]*Federal University of Mato Grosso do Sul, Ponta Porã, Brazil*
[3]*Federal University of Bahia, Salvador, Brazil*
and University of Campinas, Campinas, Brazil
[4]*University of Derby, Derby, United Kingdom*
Email: brunoguazzelli@unifei.edu.br

With industry 4.0, data-based approaches are in vogue. However, extracting the essential features is not a trivial task and greatly influences the final result. There is also a need for specialized system knowledge to monitor the environment and diagnose faults. In this context, the diagnosis of faults is significant, for example, in a vehicle fleet monitoring system, since it is possible to diagnose faults even before the customer is aware of the fault, minimizing the maintenance costs of the modules. In this paper, several models using Machine Learning (ML) techniques were applied and analyzed during the fault diagnosis process in vehicle fleet tracking modules. Two approaches were proposed, "With Knowledge" and "Without Knowledge", to explore the dataset using ML techniques to generate classifiers that can assist in the fault diagnosis process. The approach "With Knowledge" performs the feature extraction manually, using the ML techniques: Random Forest, Naive Bayes, Support Vector Machine (SVM) and Multi Layer Perceptron (MLP); on the other hand, the approach "Without Knowledge" performs an automatic feature extraction, through a Convolutional Neural Network (CNN). The results showed that the proposed approaches are promising. The best models with manual feature extraction obtained a precision of 99.76% and 99.68% for detection and detection and isolation of faults, respectively, in the provided dataset. The best models performing an automatic feature extraction obtained respectively 88.43% and 54.98% for detection and detection and isolation of failures.

Keywords: Machine Learning; Fault Diagnosis; Feature Extraction; Convolutional Neural Networks

Received 31 October 2019; revised 06 April 2021

1. INTRODUCTION

Fault diagnosis has its importance due to the demand for reliability in systems subject to possible abnormalities and component failures, and these faults can result in a malfunction or a system shutdown. In this way, it is crucial to detect and isolate possible anomalies and failures as early as possible to minimize system degradation, avoid hazardous situations and financial losses.

There are several ways of fault diagnosis, such as those based on models, signals and knowledge [**?**]. However, with a large amount of data available, fault diagnosis methods are making use of this data to detect, isolate and identify faults in systems. Also, it is noted that traditional approaches to fault diagnosis require a mathematical model of the system, which is not necessary when using a machine learning (ML) technique to diagnose faults [**?**].

Fault diagnosis is vital for the most diverse systems since this process aims to ensure a robust, efficient and durable system. In this way, when designing a system, it is always interesting to develop a method to perform this diagnosis.

Traditional knowledge-based approaches typically have a feature extraction phase, in which there is

a demand for specialized knowledge of the system [**?**]. This step consists of the selection of the most representative attributes and the creation of new handcrafted attributes [**?**]. However, there are currently methods that do not extract features and work with all possible data, such as Convolutional Neural Network (CNNs) [**?**].

Indeed, one of the biggest challenges in fault diagnosis using ML is that the data must be carefully analyzed and adequately treated so that the method can converge to get an efficient model. [**?**] list some challenges of this approach such as defining a single diagnostic procedure, robust fault separation, noise sensors, missing data, the interdependence of system processes, and lack of specialized knowledge.

Many transportation and logistic companies use modules to track their vehicles, where it is common to find that some modules are not working as expected. Thus, it is significant to remotely diagnose failures of these modules, since, in the event of failures, financial losses may occur. The vehicle may have to stand; still, wrong vehicle data can be collected and also technical support can be sent to analyze the problem over a long distance, for example.

1.1. Motivation

This research is motivated by the fact that the diagnosis of failures is a task of great importance when it is desired to maximize the utilization of the projected system. From this importance, combined with the amount of data available and the difficulty of mathematically modeling specific systems, data-driven fault diagnosis approaches are increasingly important.

While traditional ML approaches such as Decision Trees, Naive Bayes, and SVM do not work well with large amounts of data, they require feature extraction that is efficient to deliver accurate results [**?**]. In the meantime, Deep Learning (DL) approaches can work with large amounts of data and do not require manual feature extraction. In this way, DL can extract relevant system characteristics automatically [**?**]. Another motivational factor is to quantitatively analyze the influence of a proper extraction of characteristics concerning the extraction of automatic features through an automated method.

Another important factor is that machine learning techniques can be used in many areas besides fault diagnosis, such as medical diagnosis [**?**], genetics [**?**], robotics, and several other areas [**?**]. Thus, analyzing applications of these techniques can contribute to the other application areas of ML.

The goal of this work is to apply machine learning techniques to diagnose faults in vehicle fleet tracking modules. For this, a company named DDMX provided a dataset containing data from several tracking modules. Two approaches were proposed, one using a system expert knowledge, and another without any instruction from the system expert, both were analyzed and compared.

The remainder of the paper is organized as follows: the preliminary knowledge is described in Section 2; related works are discussed in Section 3; Section 4 shows the problem description and proposed approaches; the experiments and results analysis are carried out in Section 5, and finally, the conclusions and suggestions for future works are presented in Section 6.

2. PRELIMINARY KNOWLEDGE

Faults are an unallowed deviation of at least one property or a characteristic parameter of the system [**?**]. According to [**?**], there are three steps to diagnose faults:

- **Fault Detection**: is the most basic task of fault diagnosis, it is used to check for malfunction or failure in the system and to determine when the fault occurs;
- **Fault Isolation**: the isolation serves to detect the location of the fault or which is the defective component;
- **Fault Identification**: the identification is used to determine the type, format, and size of the fault.

Generally, the fault diagnosis process consists only of Fault Detection and Isolation (FDI). This does not remove the utility of fault identification, however this process may not be essential if no reconfiguration action is involved [**?**].

In data-driven approaches to fault diagnosis, it is common for the dataset to be unbalanced, that means the dataset has a large quantity of flawless data and only a small amount of data with faults [**?**, **?**]. For example, in a problem with two classes, if a positive class has 95% of the data and the negative class has 5%, this database is considered unbalanced. In the fault diagnosis scenario, this behavior is quite common since faulty systems are not as likely as working systems. Generally, a database is considered unbalanced when one of the classes has double the elements concerning any other class [**?**].

An unbalanced dataset can cause ML classifier algorithms to create a tendency to classify for the class with the most massive amount of data, called the majority class. The traditional approach to avoid this problem is to undersample the majority classes or oversample minority classes [**?**, **?**].

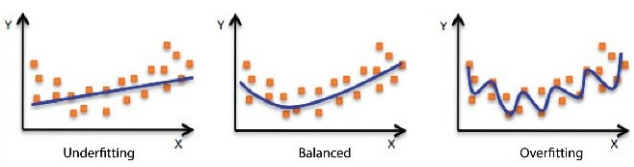

FIGURE 1. Fitting explanations [**?**]

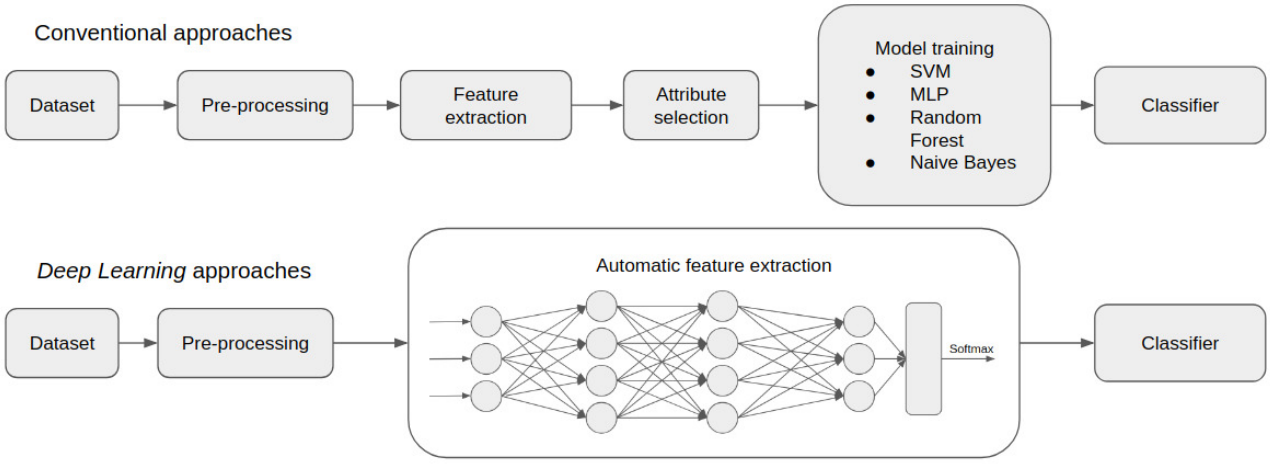

FIGURE 2. Flowchart of traditional approaches and approaches with DL.

In statistics, a fit means how well a function ($f(x)$) approximates a target function ($y(x)$). Overfitting happens when a model learns the noise and details of the training data; consequently, the model will not present good results in the test set. Underfitting happens when a model does not learn from the training data. Then it also has a low performance on the test data. A good or balanced fit occurs when the model fits the training and the test data [**?**]. These concepts are illustrated in Figure 1.

Figure 2 shows a comparison between the methods that make the automatic extraction of characteristics with the DL and conventional methods, performing the extraction of characteristics manually. Also, it is possible to observe the steps of feature extraction and attribute selection present in conventional approaches. These steps are fundamental for dimensionality reduction and aim to ensure that the ML model will represent the system well. The pre-processing in the figure refers to the process of cleaning, normalizing and standardizing the data.

Precision, recall and F_1-*score* metrics are used to compare the models and choose which is better. In the fault diagnosis scenario, precision expresses the ability to avoid false positives, and recall expresses the capability to avoid false negatives. Equations (1) and (2) define precision and recall, where TP is the number of True Positives, FP is the number of False Positives and FN is the number of False Negatives.

$$Prec(\hat{f}) = \frac{TP}{TP + FP} \tag{1}$$

$$Recall(\hat{f}) = \frac{TP}{TP + FN} \tag{2}$$

In some circumstances, it is desired to find the optimal combination of precision and recall. In those cases, these two metrics are combined using the F_1-score, which is the harmonic average of precision and recall. Equation (3) defines this metric.

$$F_1\text{–}score(\hat{f}) = \frac{2 \times Prec(\hat{f}) \times Recall(\hat{f})}{Prec(\hat{f}) + Recall(\hat{f})} \tag{3}$$

Also, to better analyze the models, the confusion matrix is used to provide an overview of the model accuracy and recall, and also allows an analysis of the model's performance for individual classes.

With the growing amount of data, many researchers are developing applications to find patterns in a data set, including fault detection and isolation. Thus, with the advancement of computing, the data-driven approach to fault diagnosis is becoming more common. Next section discusses about some studies.

3. RELATED WORK

Several studies available in the literature make use of machine learning for the fault diagnosis. [**?**] proposed a new framework for fault diagnosis of electric machines (DTS-CNN). The framework is fed up directly by raw signals and is less dependent on prior knowledge. It uses a dislocation time series operation to convert 1-D signals to 2-D form in order to apply the Convolution Neural Network technique to diagnose the faults. The framework is considered a powerful fault diagnosis method for large testing samples in non-stationary conditions. Nevertheless, DTS-CNN operates as a black box, so there is no way to retrieve knowledge from the models.

[**?**] proposed a model for the classification of multiple faults in bearings. In the methodology, there is a feature extraction (time and frequency domains) phase, where the signal input is converted to just 8 attributes. Then, a SVM technique was used to diagnose the faults.

To optimize the parameters of the SVM, a particle swarm algorithm was used. The results were compared with other traditional methods and, according with the authors, they had a better generalization and robustness. However, this approach demanded manual feature extraction and, compared to studies that extract features automatically, they present superior results as can still be seen in this section.

[**?**] made a fault diagnosis model to detect and isolate faults on wind turbines. The authors used data input from four time series acquired by four sensors, and the data in the input for a Long Short-Term Memory (LSTM) network, without any manual feature extraction, to perform the fault diagnosis. Comparisons with other methods were carried out to evaluate the model, including MLP, SVM, CNN, and Recurrent Neural Network (RNN), showing that the LSTM model was superior in almost all metrics.

Also to diagnose bearing failures, [**?**] proposed a Hierarchical Diagnosis Network (HDN) collecting Deep Belief Networks (DBNs) by layer for hierarchical identification of the mechanical system. Wavelet Packet Transform (WPT) was used to extract features for the DBNs. The results were compared to an SVM and a Back Propagation Neural Network (BPNN) to confirm the capability of the HDN, showing that the HDN is reliable and can overcome problems caused by noise and other disturbances.

[**?**] performed a fault diagnosis in several databases using the CNN method, including bearing databases, centrifugal and axial piston hydraulic pumps. The proposed method transforms data from 1-D signals to 2-D images. In order to evaluate the algorithms, the results were compared with other architectures of CNNs and an SVM.

For the intelligent diagnosis of rotating machines, [**?**] identified the shortcomings of past approaches and also proposed a DL classifier model to work without feature extraction. The authors discussed the improvements in the use of several hidden layers and their potential with a massive amount of data. A comparison was also carried out with the traditional methods without feature extraction to demonstrate the superiority of the deep model.

[**?**] proposed a combined Deep & Shallow model (DSM) to take advantages from Deep Neural Networks (DNNs) and shallow ones. This model structure is a shallow linear neural network combined with a deep feed-forward dense network. The results showed that joint training significantly improved the performance for both deep and shallow parts. Also, for the test case scenario, the DSM performed better than CNN, LSTM, Naive Bayes, Random Forest, and other methods.

[**?**] used an SVM to classify multiple faults in an electric generation system from hydrogen. As it is a system with many variables, a PCA (Principal Component Analysis) was also used to decrease the dimensionality of the problem. According to the authors, the results were as expected and sufficient to significantly increase the safety and reliability of the system.

Most of these analyzed papers make automatic feature extraction, presenting substantial insights. However, in this work, we analyze models using manual and automatic feature extraction to study its advantages and shortcomings. Furthermore, no study was found in the literature considering a performance evaluation of machine learning techniques for fault diagnosis in vehicle fleet tracking modules. In the next section, the case study of this research is detailed.

4. PROBLEM DESCRIPTION AND PROPOSED APPROACHES

As pointed out in Section 1, one of the objectives of this work is to apply machine learning techniques for the fault diagnosis in vehicle fleet tracking modules. For this, two approaches were applied, one using the knowledge of a system expert to perform the feature extraction, and the other without making use of any specific knowledge about the application, working directly with raw data. The approach that makes use of the knowledge of the specialist is called "With Knowledge" and the approach which does not make use of knowledge is called "Without Knowledge".

The case study of this research is a problem of fault diagnosis of vehicle fleet tracking modules, where several failures may occur. All data used is real and has been granted by DDMX[1].

4.1. Problem description

The modules installed in the vehicles send data during their entire period of operation. A registry composed of all the data sent by one module in one day period was defined. On average, each module sends 1116.67 points per day. The aim is to classify the faults that a registry can have or if it is working correctly.

The DDMX company provided 12586 registries. Each point that a module sends has 62 attributes, of which a system expert discards the majority after a rigorous analysis, such as data that are not configured on all modules and temporary data used for other applications. After the analysis of the system expert, just eight attributes left, which are described in Table 1. Note that, the approach "Without Knowledge" will use the data which is discarded by the system specialist.

Based on the company expertise, seven faults were mapped considering their occurrence in the modules fixed in the vehicles. In order to perform this mapping, each fault has been isolated in a way that is independent of each other; that is, nothing prevents multiple faults in the same module. This mapping is presented in Table 2.

[1] http://ddmx.com.br

TABLE 1. Attribute descriptions

Attribute	Description
Battery Voltage	Float value for voltage of the vehicle battery.
Longitude	Float value for the vehicle longitude.
Latitude	Float value for the vehicle latitude.
Ignition	Boolean value indicating whether the ignition is switched on.
Odometer	Float value for the vehicle odometer.
GPS signal	Boolean value indicating whether the GPS (Global Positioning System) signal is valid.
Velocity	Float value for the vehicle speed.
Memory Index	Integer value for the memory position that the point is saved in the module.

TABLE 2. Failure descriptions

Failure	Description
Failure 1	Wrong pulse set up.
Failure 2	Odometer locked.
Failure 3	GPS locked.
Failure 4	Ignition wire released.
Failure 5	Accelerometer defective.
Failure 6	Module buffer with problem.
Failure 7	GPS with jumps in location.

Since the modules send data constantly and on different frequencies, each registry has a different size. In Equation (4), P_k is the number of points a module sends during one day period, and eight is the number of attributes per point, so a given module k sends N_k total attributes.

$$N_k = P_k \times 8 \tag{4}$$

From these attributes and the ones discarded, it is possible to formulate supervised classification models to predict which failure a registry has or if it is working correctly.

4.2. "With Knowledge" Approach

A process methodology was developed and presented in Figure 3, which was proposed for the approach With Knowledge, due to the particularity of the used data. This procedure is simple and straight forward, containing the basics of every machine learning process and some specific details for the case study.

4.2.1. Data set Acquisition

The relation of the number of registries collected and their failures can be seen in Table 3. It is essential to consider that, in the collected data, some failures occurred less frequently than others. Failures that did not reach 3% of the total registry number were all labeled as "Others".

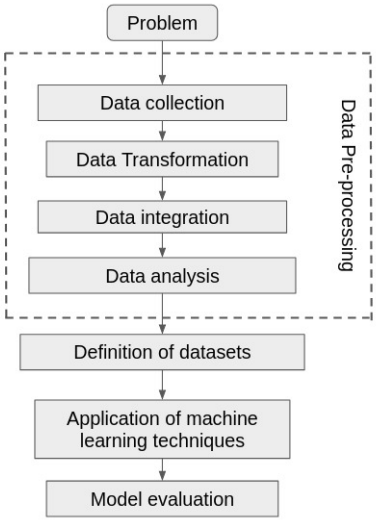

FIGURE 3. Methodology flowchart in the "With Knowledge" approach.

TABLE 3. Failure quantities

Failure	Number of registries
Flawless	5106
Failure 2	2170
Failure 3	1573
Failure 4	1702
Others	2035
Total	12586

4.2.2. Data Transformation

Since the feature extraction process is an extensive piece of work and significantly impacts the final result, the data transformation uses the knowledge provided by a system specialist [**?**]. From the expert knowledge, a dimensionality reduction on the data was performed. The initial attributes are time series (battery voltage, latitude, etc.). Twenty-one new attributes were created, shown in Table 4, using the knowledge of the system expert. After this, data transformation process converted the original data from 2-D to just one dimension.

Most of these new attributes can be calculated directly and, therefore, there is no complex formula or algorithm for this. The only attribute that needs formula is the RMS (Root Mean Square) value of the battery voltage, which is calculated by Equation (5) [**?**]. In the equation, N is the number of points sent in the day and v_i is the value of each battery voltage sent during the day.

$$V_{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} v_i^2} \tag{5}$$

The choice of each of these attributes was made based on company employees' manual fault diagnosis

TABLE 4. Description of the new attributes.

Attribute	Attribute description
Date	Date of the registry
Serial	Code that identifies the module
Longitude Variation	Boolean indicating whether the longitude varied
Latitude Variation	Boolean indicating whether the latitude varied
Ignition Variation	Boolean indicating whether the ignition varied
Odometer Variation	Boolean indicating whether the odometer varied
Velocity Variation	Boolean indicating whether the velocity varied
Maximum battery voltage	Float for higher voltage recorded in the battery
Minimum battery voltage	Float for lower voltage recorded in the battery
Maximum battery voltage with ignition on	Float for higher voltage recorded on battery with the ignition on
Maximum battery voltage with ignition off	Float for higher voltage recorded on battery with the ignition off
Minimum battery voltage with ignition on	Float for lower voltage recorded on battery with the ignition on
Minimum battery voltage with ignition off	Float for lower voltage recorded on battery with ignition off
Battery voltage variation	Float for maximum battery voltage minus minimum battery voltage
RMS (Root Mean Square) of battery voltage	Float for RMS value of the battery voltage signal
Higher positive memory index jump	Float for highest positive jump in memory index
Higher negative memory index jump	Float for highest negative jump in memory index
Number of memory index jumps	Float for quantity of jumps in memory index
Number of points with valid GPS	Float for amount of points with valid GPS
Number of points with invalid GPS	Float for amount of points with invalid GPS
Number of jumps in latitude or longitude	Float for amount of jumps on the GPS

experience. For example, to check if the GPS of the module is defective, check if the speed of the vehicle varied without latitude and longitude variation.

4.2.3. Data Integration

The label of each registry is in the $fault$ collection. Then, it was necessary to make queries to find which fault each registry has. For this, the date and serial attributes were used. Thus, two class attributes were defined: $faults$ and has_faults, which are respectively a string indicating the faults that the module has and a Boolean value indicating if there is a failure.

4.3. Data Analysis

Most of the attributes are binary, so they do not require normalization. However, attributes such as the number of points, data regarding battery voltage and jumps in the memory index can be normalized. Since outliers can be symptoms of a fault, there is no outlier analysis. A traditional normalization can cause the data to be suppressed by some high value, so *RobustScaler* is used according to Equation (6) [**?**]. In this equation, the first and third quartiles are used as constraints and not the maximum and minimum values, so this normalization is more robust and maintains the outliers.

$$S_i = \frac{x_i - Q_1(x)}{Q_3(x) - Q_1(x)} \tag{6}$$

Since some attributes can have a high correlation to others, all the possible correlations were calculated. The Longitude and the Latitude Variation have a Pearson correlation of 0.996315, which in most ML projects suggest we should discard one of these attributes. However, since the objectives of the models are diagnosing faults in the GPS of the module, both attributes are useful despite the high correlation.

Furthermore, there is a high correlation between the number of points and the number of points with invalid GPS, 0.962769. This occurs because the number of points is the sum of the number of GPS points that are valid with the number of invalid GPS points. Therefore, due to this linear combination, the attribute "number of points" was discarded.

Finally, the date and serial were discarded since they were just used to integrate the classes, as described on Subsection 4.2.3. In this way, now there are just 19 attributes.

These steps complete the data pre-processing phase shown in Figure 3, which was fundamental to standardizing the input data and ensuring that the data were organized. After this, ML techniques can be applied without significant difficulties. In this way, experiments regarding this approach are detailed in Section 5 and the rest of this section describes the approach "Without Knowledge".

4.4. "Without Knowledge" Approach

In the "Without Knowledge" approach, the ML models work directly with the raw data of the tracking modules, without performing manual feature extraction as in the "with information" approach. For this, deep neural network models were used, capable of automatically extracting the features of the dataset.

The methodology adopted in this approach is

described in the Figure 4, where it is possible to note that there is still a data pre-processing process. However, it maintains the entire original set of data without creating new attributes.

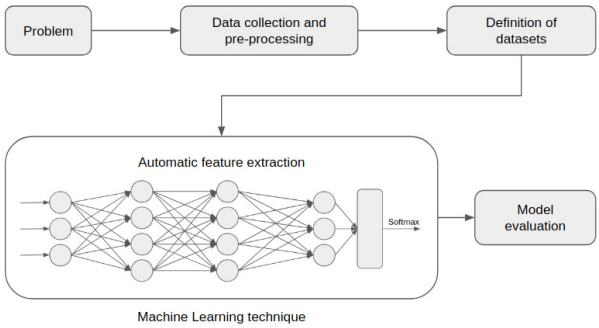

FIGURE 4. Methodology flowchart in the "Without Knowledge" approach.

As described in Subsection 4.2.1, at the time of data extraction, they were collected from the database and standardized. As shown in Section 4, each point sent by the modules has 62 attributes, of which eight have been selected as useful by the system expert. However, as the objective of this work is to perform the feature extraction Without Knowledge of the expert, all the collected data were pre-processed, not only these eight.

The first step in this pre-processing was to discard data that does not make sense for classification, redundant data, and null data. After that, there were 41 attributes per point sent.

The second step was to normalize all the data, again the robust normalization was used, not suppressing the data by the presence of outliers.

Since the number of attributes per registry is different, this amount can be defined by Equation (7), where P_k is the number of points a module sends over a one-day period, and 41 is the number of attributes per point, so a given module K sends a total of N_k attributes.

$$N_k = P_k \times 41 \tag{7}$$

Considering the registries have variable length, it was not possible to directly construct the ML models with this data set. In this way, it was necessary to set a default size for all retries. Knowing that the highest number of points present in a registry is 6410, the mean points is 1116.67, and the standard deviation is 1155.75, the registries with a smaller amount of data were completed with "0s" until they have the same amount of data as the largest registry. In this way, all registries have total attribute amounts equal to 262810 (6410×41).

Different approaches to set the default size of a registry could be performed, such as truncating the amount of data from above-average registries, and completing with "0s" those that are below average, or truncating the amount of data from all registries with more data than the smaller registry. However, these approaches were tested and considered unfeasible due to the divergence of the ML models.

After processing the data, it was possible to train the ML models. Next section presents the experimentation and results discussion.

5. EXPERIMENTATION AND DISCUSSION

In this section, the planning of the experiments together with their respective results and discussions are presented.

5.1. Experiments "With Knowledge"

5.1.1. Definition of datasets

It is significant to improve the analysis of the results and to discover whether any faults were mapped in the wrong way, besides the possibility to compare the efficiency of the techniques in different tasks. From the original dataset, two structures of the same dataset were defined: one to detect faults (Structure 1) and other to detect and isolate faults (Structure 2). These structures are described as follows:

- Structure 1 was used in the experiments to detect if the system had faults or not. It contains 12586 records, of which 7480 are flawed, and 5106 are flawless;
- Structure 2 was used in the experiments to detect and identify faults. It contains 5106 records flawless, 2170 with Failure 2, 1573 with Failure 3, 1702 with Failure 4, and 2035 with "Others" Failure, totaling 12586 records.

5.1.2. Application of ML methods

In this subsection, the design of the experiments is described considering the vehicle fleet tracking modules dataset to diagnose faults. All the models were written in Python 3.6 with *Sk-learn* 0.20.2 and ran on Ubuntu 18.04 with an Intel i7-7700HQ, 16Gb RAM, GTX 1050Ti.

In order to run the algorithms, the datasets were separated into a training and a test set. For every experiment, 20% of the registries were for testing and 80% for training. This percentage is considered a common practice in the literature [**?**].

A full factorial experimental planning was used for the experiments. This model is outlined by [**?**] and is suitable for the analysis of the response variables. In this methodology, the planning and analysis of experiments include both factors and levels, where the factors correspond to environmental characteristics, and the levels are the possible environmental variations. Table 5 shows the factors and levels. All combinations of techniques, structures, and sampling were performed,

TABLE 5. Planning of experiments for the "With Knowledge" approach.

Factors	Levels
ML Technique	Random Forest, Naive Bayes, SVM, MLP
Sampling	Oversampling, Undersampling
Structure	1, 2

totaling 16 experiments. The response variables considered were: precision, recall, F_1-*score* and also the confusion matrix to see if there is any confusion between the classes.

Considering the ML techniques, all Random Forests were made using 100 estimators, and all MLPs were done with architecture (5, 10, 5), $lr = 0.001$ and 200 maximum iterations. The following section presents the analysis of the results.

5.1.3. Models evaluation

The response variables need adequate interpretation. In the case study in question, it is important to minimize the number of false positives, as this would involve sending a technician to perform unnecessary maintenance, while a false positive implies receiving a complaint of some malfunctioning module. Thus, from the financial point of view, the recall has greater relevance than precision. As the tie-breaking criterion, the F_1-*score* is used. This measure is the harmonic mean between precision and recall. Thus it is presented as a balanced evaluation criterion, evaluating false positives and false negatives.

In the whole "With Knowledge" approach, it is observed that the average between all the precision of a model and the average of all the recalls of the same model, tends to cancel the exchange between these metrics. It is also possible to analyze the number of false positives and false negatives directly in the confusion matrix. In this way it is simple to analyze if the model tends to classify false positives or false negatives. This analysis reveals that, in general, the models of this approach tend to classify more false negatives than false positives, which is seen negatively from the financial point of view.

All models indicate confusion between the faults named "Others", and all other faults. This is due to the "Others" class being a cluster of several flaws. Consequently, there was confusion between the "Other" class and the other classes. Besides, metrics related to Failure 2 always remain above 95%, which demonstrates that this is a fault that is more easily detected by the proposed approaches.

In Structure 2, all techniques, except Random Forest, presented superior performance with oversampling. This indicates that Random Forest was able to learn with a smaller amount of data.

In Table 6 there is a summary of all experiments showing the precision, recall and F_1-*score* of all the experiments. By "US", "OS", "S-1" and "S-2", we mean under-sampling, over-sampling, structure 1 and structure 2, respectively.

The undersampling presented better results in both sets of experiments. Also, among the four techniques used, Random Forest stood out, followed by SVM in all experiments. Naive Bayes had low performance compared to other techniques, mainly because it assumes complete independence between attributes, which does not exist. Also, to better evaluate the models, some analyses were made specifically for each structure of data.

5.1.4. Experiment 1

The Random Forest technique with undersampling had the best results considering the identification of faults in the dataset. The normalized confusion matrix for this experiment can be seen in Figure 5.

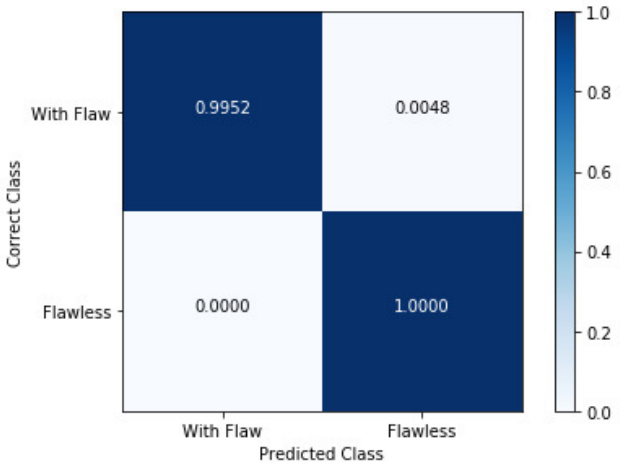

FIGURE 5. Confusion Matrix for the Random Forest in Experiment 1 with undersampling.

In general, the false positive rate was smaller than the false negative rate, which is not good from a financial point of view. Also, the difference between the undersampled model and the oversampled model was small; however, undersampling was more efficient. The worst results were observed with Naive Bayes technique. The rates of false positives and false negatives were close, and the training set was submitted to a result score of 82.78% in the F_1-*score*, which indicated the underfitting in these models. Also, the undersampled model was more efficient than the oversampled model. After Random Forest, the best overall efficiency was recorded by SVM. Like Naive Bayes, the undersampled model obtained better results than the oversampled. Finally, MLP had a performance similar to SVM, but with higher rates of false negatives. Despite being a simple neural network architecture, it was able to adapt to the problem.

TABLE 6. Evaluative metrics of every model in "With Knowledge" approach (%)

		Random Forest			Naive Bayes			SVM			MLP		
		Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
S-1	US	**99.76**	**99.76**	**99.76**	82.22	82.18	82.19	98.83	98.83	98.83	97.19	97.16	97.16
	OS	99.68	99.68	99.68	81.45	80.94	81.06	98.89	98.89	98.89	98.17	98.13	98.14
S-2	US	**99.68**	**99.68**	**99.68**	79.19	78.70	78.79	97.87	97.84	97.84	97.39	97.33	97.33
	OS	99.53	99.53	99.53	82.45	80.18	80.81	98.09	98.09	98.08	97.78	97.78	97.77

5.1.5. Experiment 2

There is a factor in common among all confusion matrices referring to Experiment 2. All indicate confusion between faults named "Others" and " Flawless". That is due to the class "Others" being an agglomeration of several faults. As a result, it was expected that there would be confusion between the "Others" class and the other classes. Besides, the metrics related to Failure 2 always remain above 95%. This demonstrates that this is a fault that was easily detected by the proposed methods.

The best result for Experiment 2 during the identification of a specific fault was the Random Forest with undersampling. The normalized confusion matrix for this experiment can be seen in Figure 6. All techniques, except for Random Forest, presented superior performance with oversampling. This indicates that Random Forest was able to learn with fewer data.

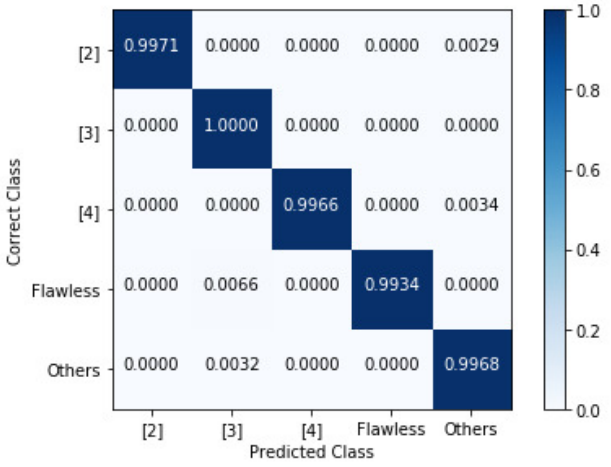

FIGURE 6. Confusion Matrix for the Random Forest in Experiment 2 with undersampling.

The Random Forest technique presented the best performance, only presenting confusion between the "Others" and "Flawless" classes. Considering Naive Bayes, this technique only efficiently classified the Failure 2, showing confusion among all the other classes. Again, Naive Bayes was underfitting with 82.19% F_1-*score* in the training set. SVM, as well as Random Forest, also presented good results during the fault identification process. Finally, in MLP, even with a simple architecture and five classes, the neural network was able to learn and adapt to the problem in question.

TABLE 7. Time (s) to train and evaluate the models.

		Random Forest		Naive Bayes		SVM		MLP	
		Train	Test	Train	Test	Train	Test	Train	Test
S-1	US	0,77	0,31	0,04	0,31	269,00	0,50	3,06	0,36
	OS	1,00	0,20	0,01	0,31	565,00	0,54	3,27	0,35
S-2	US	0,53	0,31	0,08	0,32	1,00	0,55	0,54	0,28
	OS	1,21	0,38	0,11	0,47	865,00	0,94	2,51	0,27

5.1.6. Time evaluation

In addition to the evaluation metrics for ML techniques, the training and evaluation times of the models were also calculated. These data can be seen in Table 7.

The test time refers to the evaluation of the entire test set, so the average evaluation time of a sample by dividing the total test time by the number of samples. This time was, in the worst case, 0.37 milliseconds, what can be considered a real-time response for the application in this case study.

In Table 7, it can be observed that the training time was generally higher than the test time, except for Naive Bayes. That is due to the way of training of this technique which is naturally statistical and requires fewer operations than any of the three other techniques. Also, it is noted that the SVM had a substantially longer training time than the other techniques, which is mainly due to the grid search performed to find the parameters used.

The "With Knowledge" approach is robust, and despite the need for specialized knowledge of the system was effective and presented a positive evaluation. In Subsection 5.2, the "Without Knowledge" approach is described and evaluated.

5.2. Experiments "Without Knowledge"

5.2.1. Machine learning method

The evaluation factors and the levels used in the planning of these experiments can be seen in Table 8; are just two factors with two levels in each. All combinations of factors were performed, totaling four experiments, one for each structure. The response variables considered were: precision, recall, F_1-*score* and confusion matrix.

All experiments were performed using an environment with the same settings described in 5.1.2, adding the following libraries: *Tensorflow-gpu* 1.12 and *Keras* 2.2.4. Thus, as in the "With Knowledge" approach, the dataset was separated into a training set and test set to execute the algorithms. For each experiment, 20% of the records were used for testing, and 80%

TABLE 8. Planning of experiments for the "Without Knowledge" approach.

Factors	Levels
Objective	Detect faults, detect and isolate fault
Attribute selection	With attribute selection Without attribute selection

for training.

The ML technique used in this approach was CNN, as described in Subsection 5.2.3, due to the high adaptability and generalization that this network has.

5.2.2. Definition of the datasets

As in the "With Knowledge" approach, structures of the original data set were defined for each experiment, and in this approach, the same records that were used in the "With Knowledge" approach were used. In total, four structures have been defined, and in addition to these structures have different objectives, they also work with different amounts of data. Two use data without any attribute selection and two other databases select only eight attributes per sent point. These eight are those selected as useful by system expert. Also, due to the superior performance of sub-sampling in the "With Knowledge" approach, and difficulty in creating synthetic data with a high amount of attributes, an undersampling is performed to define these test bases, selecting the data randomly. In this way, four structures were defined, described as follows:

- **Structure 1**: made only to detect failures, without attribute selection. It contains 10212 records, 5106 of which have failures and 5106 without failures. Each registry has 262810 attributes;
- **Structure 2**: made to detect and isolate failures without attribute selection. It contains 7865 records, 1573 with Failure 2, 1573 with Failure 3, 1573 with Failure 4, and 1573 with " Others" Failures. Each registry has 262810 attributes;
- **Structure 3**: made only to detect failures, with attribute selection. It contains 10212 records, 5106 of which have failures and 5106 without failures. Each registry has 51280 attributes;
- **Structure 4**: made to detect and isolate faults, with attribute selection. It contains 7865 records, 1573 with Failure 2, 1573 with Failure 3, 1573 withFfailure 4, and 1573 with "Others" Failures. Each registry has 51280 attributes.

5.2.3. CNN architecture

The data of structures 1 and 2 have dimensions (6410, 41), and the data of structures 3 and 4 have dimensions (6410, 8). That is due to the selection of attributes, implying the need for two CNNs, one for structures 1 and 2, and another for structures 3 and 4. In this way, two CNN architectures were designed, both based on the VGG-16 architecture [**?**].

TABLE 9. CNN architecture for structures without attribute selection.

Layer name	Layer description
Conv-1	128 filters, (3, 3), ReLU
Conv-2	128 filters, (3, 3), ReLU
Pool-1	*Avg-Pooling*, (9, 2)
Drop-1	*Dropout*, 10%
Conv-3	256 filters, (3, 3), ReLU
Conv-4	256 filters, (3, 3), ReLU
Pool-2	*Avg-Pooling*, (7, 2)
Drop-2	*Dropout*, 10%
Conv-5	512 filters, (3, 3), ReLU
Conv-6	512 filters, (3, 3), ReLU
Pool-2	*Avg-Pooling*, (5, 1)
Drop-3	*Dropout*, 10%
Conv-7	768 filters, (3, 3), ReLU
Conv-8	768 filters, (3, 3), ReLU
Pool-2	*Avg-Pooling*, (3, 1)
Drop-4	*Dropout*, 10%
Flatten-1	Flatten
FC-1	4096 neurons, ReLU
Drop-5	*Dropout*, 60%
FC-2	768 neurons, ReLU
Drop-6	*Dropout*, 20%
FC-3	n neurons, *Softmax*

In the Table 9, all the CNN architecture used in structures 1 and 2 is described. The acronyms and abbreviations Conv, Pool, Drop and FC mean, respectively, Convolutional Layer, Layer sub-sampling, Dropout layer, and fully connected layer. The denotation of "128 filters, (3, 3), ReLU" means that it is a convolution layer, there are 128 filters, the filter size is 3×3 and there a ReLU activation function. AvgPooling(3, 3) means that it is an average-pooling layer with 3×3 filters.

In Table 10, the architecture used in structures 3 and 4 is described. The changes made were due to the dimensionality of the input data.

In the last layer, in each of the architectures, the number n corresponds to the number of classes of the problem involved. That is two classes for structures 1 and 3, and five for structures 2 and 4. The undersampling layers are of the type Average-Pooling because of the data collected.

As an optimizer, the study carried out by Nadam et al. [**?**] was used, which is Adam along with the Nesterov moment, with a learning rate equal to 0.0001. The size of the batch size used was one due to the computational costs.

5.2.4. Models evaluation

All the models were used to evaluate the training and the testing set. Table 11 shows the results of all models of the "Without Knowledge" approach.

It is possible to observe the importance of selecting

TABLE 10. CNN architecture for structures with attribute selection.

Layer name	Layer description
Conv-1	128 filters, (4, 4), ReLU
Conv-2	128 filters, (4, 4), ReLU
Pool-1	*Avg-Pooling*, (9, 1)
Drop-1	*Dropout*, 10%
Conv-3	256 filters, (3, 3), ReLU
Conv-4	256 filters, (3, 3), ReLU
Pool-2	*Avg-Pooling*, (7, 1)
Drop-2	*Dropout*, 10%
Conv-5	512 filters, (3, 3), ReLU
Conv-6	512 filters, (3, 3), ReLU
Pool-2	*Avg-Pooling*, (5, 1)
Drop-3	*Dropout*, 10%
Conv-7	768 filters, (3, 3), ReLU
Conv-8	768 filters, (3, 3), ReLU
Pool-2	*Avg-Pooling*, (3, 1)
Drop-4	*Dropout*, 10%
Flatten-1	Flatten
FC-1	4096 neurons, ReLU
Drop-5	*Dropout*, 60%
FC-2	768 neurons, ReLU
Drop-6	*Dropout*, 20%
FC-3	n neurons, *Softmax*

attributes for the learning process of the model. In the fault detection and the fault detection and isolation, there are, respectively, two and five classes. In this way, it can be said that the model was not able to extract features without attribute selection (AS), concluding that all models without the attribute selection suffer from underfitting.

The number of attributes per record in the registry, with the attribute selection, is more than five times smaller, and yet, most of the data present in the dataset without attribute selection is zero, which makes the feature extraction process difficult. Already with the attribute selection, a right amount of data is non-zero, except for the final data of each registry.

Furthermore, in Figure 7, the confusion matrix for fault detection using CNN shows much more false positives than false negatives, which is not the best from the financial point of view. Also, this confusion matrix confirms the model underfitting.

The confusion matrix of the CNN used to detect and isolate faults can be seen on Figure 8. There is a confusion between Failure 2 and all the other faults, and a smaller confusion between the "Others" Failures and all the remaining faults. That is probably due to the "Others" being a cluster of several faults, and Failure 2 confusion can be a symptom of the model underfitting.

The evaluative metrics of the models with attribute selection demonstrate that there was learning, and automatic feature extraction is the performance of the models that only detects faults higher, due to the greater ease of the proposed problem. The difference

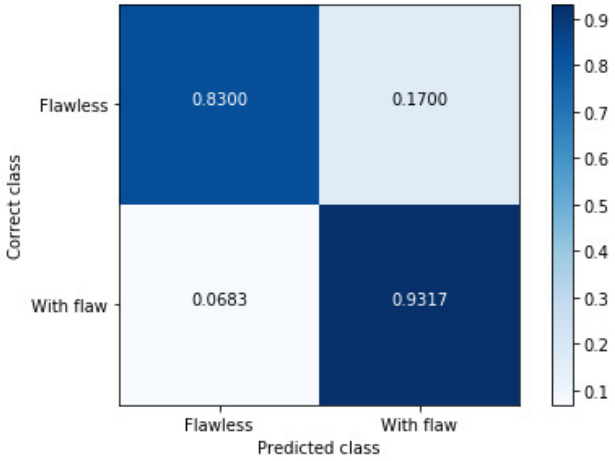

FIGURE 7. Confusion Matrix for the CNN without feature selection, used to detect faults.

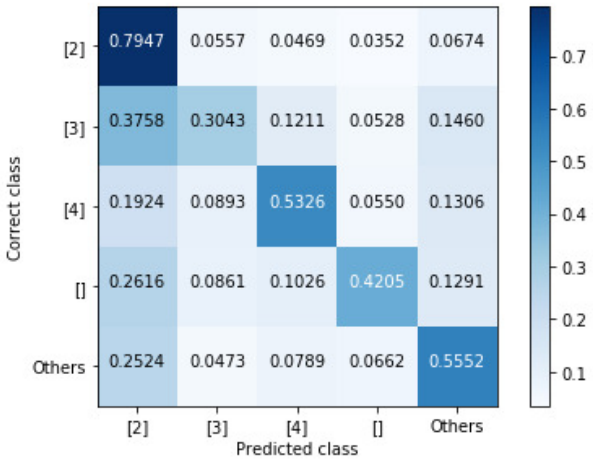

FIGURE 8. Confusion Matrix for the CNN without feature selection, used to detect and isolate faults.

of 2.25% and 19.43% between the recalls of the training and test sets show a possibly avoidable variance, with a more regularizing architecture, that is, an architecture that aims to avoid over-adjustment.

The models without attribute selection were trained for 50 epochs and did not show any signs of learning that would justify training these models for longer. On the other hand, models with attribute selection were trained for 200 epochs, presenting the best results in epochs 137 and 157. Each epoch has a training period close to 3000 seconds for fault detection, and 2000 seconds for detection and isolation.

The evaluation time can be viewed in Table 12. In the worst case, the time for the evaluation of a record is 14.3 milliseconds, which does not preclude the application of these models in practice.

TABLE 11. Evaluation metrics for models of the "Without Knowledge" approach. AS means attribute selection.

	Fault detection						Fault detection and isolation					
	Train			Test			Train			Test		
	Prec	TVP	F_1	Prec	TVP	F_1	Prec	TVP	F_1	Prec	TVP	F_1
Without AS	48,15	48,48	46,87	21,31	32,99	25,07	4,06	20,15	6,76	3,98	19,96	6,64
With AS	**90,87**	**90,21**	**90,16**	**88,43**	**87,96**	**87,94**	**79,06**	**72,00**	**72,58**	**54,98**	**52,57**	**51,55**

TABLE 12. Evaluation time of "Without Knowledge" approach.

Models	Time(s)
Fault detection, with attribute selection	29,2
Fault detection and isolation, without attribute selection	19,7
Fault detection, with attribute selection	26,4
Fault detection and isolation, without attribute selection	14,8

TABLE 13. Comparison between the approaches, using the recall. [%]

Objective	With Knowledge	Without Knowledge
Fault detection	**99,76**	87,96
Detection and isolation	**99,68**	52,57

5.3. With Knowledge vs Without Knowledge

This section, presents a final comparison between the approaches, comparing only the highest metrics, using recall as it was defined as a significant response variable.

From the inferences made in Subsection 5.1.3, it was possible to infer that the extraction of feature performed manually was shown to be active and obtained results higher than expected. Based on Subsection 5.2.4, it was also possible to say that the extraction of automatic characteristics was feasible. However, this one obtained results inferior to the "With Knowledge" approach.

In Table 13, it is observed that the "With Knowledge" approach obtained a recall 9.55% greater than the "Without Knowledge" approach, which means superior performance.

Furthermore, since the CNNs of the "Without Knowledge" approach is not optimized and more data can also be collected for the training, these results can be improved with an adjustment in the architecture, neural network parameters and changes in data pre-processing.

6. CONCLUSION

In this work, the application of a set of ML techniques to diagnosis faults on vehicle fleet tracking modules was presented. Two approaches were proposed, one making use of the knowledge of a system expert to perform a manual extraction of features and another approach employing deep neural networks for automatic feature extraction.

For the analysis of the "With Knowledge" approach, which considers the expert's knowledge, traditional techniques of ML were considered: Random Forest, Naive Bayes, SVM, and MLP. A total of 16 models were trained and tested using real data relating to vehicle fleet tracking modules. These models using the Random Forest in the undersampled dataset were able to reach 99.79% and 99.68% in recall to detect faults and detect and isolate faults. In the "Without Knowledge" approach, CNNs were built to diagnose faults extraction features automatically. Four models were trained and tested using raw data. These models obtained 87.96% and 52.57% in recall to detect faults and to detect and isolate faults, both models performing the attribute selection for data cleaning.

From the analysis performed, it is essential to note that, although most of the current fault diagnosis models use automatic feature extraction approaches, this does not necessarily apply to any system. In the data referring to the vehicle fleet tracking modules, the manual feature extraction proved to be competent, however, in order to carry it out, it is necessary to know the operation of the modules in question.

The main contribution of this work involves the definition of two approaches for the construction of ML models capable of diagnosing data failures from vehicle fleet tracking modules. One of these approaches performs the extraction of features manually, through the knowledge of a system expert, and another approach by extracting features automatically. Also, this work performs detailing the operation of each model, and a qualitative analysis comparing each model and its approach.

All proposed models are not able to discover faults that the company did not map, so any faults that have not been mapped would be erroneously classified as one of the known faults. Besides, the feature extraction is done with specialized knowledge of the system, in such a way that it can be improved. The "Without Knowledge" approach models can be optimized by changing the architecture of CNNs and other parameters. Other models such as LSTM can be used to perform this classification. In this way, developing models capable of discovering new flaws, enhancing the feature extraction from the information approach, optimizing proposed CNNs in the "Without Knowledge" approach, and developing models using different techniques are promising proposals for future work.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the infrastructure provided by UNIFEI, UFMS and

UFBA. The Python codes are available at https://github.com/lmeazzini/Tracker-fault-diagnosis if someone wants to reproduce this work.