Journal of Computer Science

Original Research Paper

Machine Learning Approaches for the Prediction of Gas Turbine Transients

[1]Arnaud Nguembang Fadja, [2]Giuseppe Cota, [2]Francesco Bertasi, [3]Fabrizio Riguzzi, [1]Enzo Losi, [1]Lucrezia Manservigi, [1]Mauro Venturini and [4]Giovanni Bechini

[1]Department of Engineering, University of Ferrara, Via Saragat 1, 44122, Ferrara, Italy
[2]Exalens Srl, Via Aldo Moro 6, 45100, Rovigo, Italy
[3]Department of Mathematics and Computer Science, University of Ferrara, Via Machiavelli 30, 44121, Ferrara, Italy
[4]Siemens Energy, Srl, Via Vipiteno, 4, 20128, Milan, Italy

Article history
Received: 7-11-2022
Revised: 06-11-2023
Accepted: 15-12-2023

Corresponding Author:
Arnaud Nguembang Fadja
Department of Engineering,
University of Ferrara,Via
Saragat 1, 44122, Ferrara
Email: ngmrnd@unife.it

Abstract: Gas Turbine (GT) emergency shutdowns can lead to energy production interruption and may also reduce the lifespan of a turbine. In order to remain competitive in the market, it is necessary to improve the reliability and availability of GTs by developing predictive maintenance systems that are able to predict future conditions of GTs within a certain time. Predicting such situations not only helps to take corrective measures to avoid service unavailability but also eases the process of maintenance and considerably reduces maintenance costs. Huge amounts of sensor data are collected from (GTs) making monitoring impossible for human operators even with the help of computers. Machine learning techniques could provide support for handling large amounts of sensor data and building decision models for predicting GT future conditions. The paper presents an application of machine learning based on decision trees and k-nearest neighbors for predicting the rotational speed of gas turbines. The aim is to distinguish steady states (e.g., GT operation at normal conditions) from transients (e.g., GT trip or shutdown). The different steps of a machine learning pipeline, starting from data extraction to model testing are implemented and analyzed. Experiments are performed by applying decision trees, extremely randomized trees, and k-nearest neighbors to sensor data collected from GTs located in different countries. The trained models were able to predict steady state and transient with more than 93% accuracy. This research advances predictive maintenance methods and suggests exploring advanced machine learning algorithms, real-time data integration, and explainable AI techniques to enhance gas turbine behavior understanding and develop more adaptable maintenance systems for industrial applications.

Keywords: Gas Turbines, Multi-Variate Time Series, Decision Tree, Extra Trees, K-Nearest Neighbour, Transient Prediction

Introduction

Gas Turbines (GTs), also known as combustion turbines, are used to convert energy by means of monitoring GTs not only help in making the service turbomachinery. The extracted energy is made available in always available but also reduces the number of failures in the form of electricity, compressed air, thrust, or a combination thereof and is used in planes, trains and ships, diagnosing GTs requires techniques of machine learning generators, tanks and many more. In order to guarantee the availability and reliability of GT assets, a robust, efficient, and flexible maintenance strategy should be designed and implemented (Tahan *et al*., 2017).

Monitoring GTs not only helps in making the service turbomachinery. The extracted energy is made available in always available but also reduces the number of failures a GT asset due to unpredictable situations. Monitoring and diagnosing GTs requires techniques of machine learning or data mining to predict their future conditions using a large set of current and past operating data.

The predictive system can exploit data provided by many sensors of many GTs in different years, thus posing big data challenges. Building such predictive systems helps in understanding the degradation trend of some components of GTs and therefore allows engineers to pay more attention to machines that have symptoms of degradation. In addition, understanding the machine degradation behavior can also help in the design of future GT components.

The term big data generally indicates a collection of data so extensive in terms of volume, speed, and variety that specific technologies and analytical methods are required for extracting value or knowledge (De Mauro *et al*., 2016). A recent study (Tahan *et al*., 2017) demonstrated that analytics methods based on large amounts of data collected from GTs can lead to significant savings in the oil and gas sector.

Data mining (Fu, 1997) and Machine Learning (ML) (El Naqa and Murphy, 2015) have been successfully applied to predictive maintenance systems whose aim is to evaluate the health state of equipment by performing regular condition monitoring. They rely on the past and actual state of the equipment to predict future conditions. Many of these methods have been applied successfully to energy-based systems using large volumes of sensor data. Zhao *et al*. (2019a) the authors present a large range of ML techniques for health monitoring as well as ML-based methods for analyzing health data. Further applications of ML for predicting the remaining useful life are presented by Zhang *et al*. (2018); Yang and Wu (2006). Naderi and Khorasani (2018) the authors outline a process that relies on Markov parameters to detect, isolate, and estimate faults on actuators and sensors in GT systems. A study by Taylor *et al*. (2019) examined the effects of damaged blades on compressor performance using ML. Wong *et al*. (2014) a real-time fault diagnostics of GTs using ML is presented. The diagnosis is done by extracting useful patterns from vibration signals. Zhao *et al*. (2019b) an improved extreme learning machine tool detects the faults of an aircraft engine.

The current paper investigates different ML models based on Decision Trees (DTs) and K-Nearest Neighbors (KNNs) to predict the future trend of the rotational speed, with an advance of 4 h. Analyses are performed without and with feature engineering and provide efficient and explainable models. Unlike other approaches, we propose an efficient preprocessing pipeline in order to not only clean the data but also to improve the prediction accuracy. The preprocessing pipeline includes efficient data extraction and selection of transient and state steady-state examples, custom stratifying sampling methods, and a broad set of feature engineering steps.

In the framework of an extended research activity conducted by the authors about the prediction of GT future operations (Losi *et al*., 2021a-b; Bechini *et al*., 2022) the current paper tackles the same challenge from a different perspective. The paper (Losi *et al*., 2021a) develops a methodology suitable to classify transients into two clusters (i.e., shutdown or trip). The methodology is applied to field data covering three years of operation and the obtained values of precision, recall, and accuracy are higher than 90% in almost all cases. The paper Losi et *al*. (2021a-b) presents a methodology dealing with data selection and feature engineering, which allows the identification of the best target examples and set of features for the setup of an ML model aimed at predicting GT trips. Such a model was developed by Losi *et al.* (2022a-b) as a random forest model, which predicts gas turbine trips based on information gathered during a timeframe of historical data acquired from multiple sensors. The model is tested by means of field data taken during three years of operation of two fleets of GTs located in different regions. In this case, precision, recall, and accuracy values are in the range of 75-85%. The paper (Losi *et al*., 2022a-b) investigates the fusion of five data-driven base models by means of voting and stacking, in order to increase model accuracy and improve prediction robustness. The five ML classifiers are K-nearest neighbor, support vector machines, naive bayes, DTs, and Long Short-Term Memory (LSTM) neural (Hochreiter and Schmidhuber, 1997). The paper Losi *et al*. (2022a-b) presents a comprehensive data-driven methodology aimed at investigating and disclosing the onset of trip symptoms, i.e., the most likely time point at which trip symptoms trigger. A (LSTM) neural network is employed as the classification model. The methodology provides the most likely trigger position for four clusters of trips within two days before trip occurrence with at least 80% confidence. The paper Bechini *et al*. (2022) first applies a systematic statistical analysis to identify the most important variables and then uses a novel ML technique known as temporal DTs, which differs from the regular DTs because it allows a native treatment of the temporal component. The learned models are finally employed to extract statistical rules.

While previous works focus on classifying trips (i.e., sudden arrests) and shutdowns (i.e., intentional shutdowns) through clustering techniques and predicting them through ensemble ML approaches, in this study we adopt a different approach in which we use ML models, DTs, Extremely Randomized Trees (ERTs) and KNNs to predict the rotational speed. Moreover, data processing and feature engineering techniques implemented in the current paper are different from our previous works. In particular, the trained models use data from the last 20 h to predict the trend of the rotational speed of the following 4 h. Based on the rotational speed trend, the proposed models allow us to predict the following GT operating conditions:

(i) Transient state (a trip or a shutdown) and
(ii) Steady state (a normal operating condition in which the rotational speed is almost constant)

Although the proposed models do not perform a direct distinction between trips and shutdowns, in real-case scenarios, if the system is predicting a transient and no human operator has intentionally sent a shutdown signal then a trip is in progress. It is worth noting that in the data used for training and testing, we do not consider explicit signals that represent shutdown commands.

Similar work has been explored by Goyal *et al.* (2020); Li *et al.* (2021); Liu and Karimi (2020). In Goyal *et al.* (2020) the authors used multiple machine learning algorithms to predict the normal behavior of operational parameters including power generated and blade path temperature spread. The predictions can be used to identify anomalies and probable failures in the gas turbine performance. The data used in the study is taken from multiple heavy-duty gas turbine units of combined cycled utility power plants which are known to contain operational failures. The predictors include operational parameters such as fuel flow, various thermodynamic variables, etc. The models are trained and parameters are selected based on the overall prediction performance on the validation set. Li *et al.* (2021) present a cross-disciplinary study on the combustion tuning of an F-class gas turbine that combines machine learning with physics understanding. A machine learning-based method was developed by Liu and Karimi (2020) to predict gas turbine performance for power generation. Two surrogate models based on Dimensional Model Representation (HDMR) and Artificial Neural Network (ANN) are developed from real operational data to predict the operating characteristics of the air compressor and turbine.

Various approaches that predict the future conditions of GTs have been investigated in the literature, (Bangert, 2020; Li and Nilkitsaranont, 2009). Bangert (2020) the authors presents an ML algorithm based on an LSTM to predict the future conditions of GTs. To train the LSTM model, the paper relies on measured variables similar to those used in the present paper. Unlike Bangert (2020) our approach makes use of tree-based models to predict future GTs conditions and classification accuracy for evaluating the models, thus it is more explainable and interpretable. Li and Nilkitsaranont (2009) the authors investigate how to predict GT future conditions by estimating the remaining useful life of GT engines before their next major overhaul based on historical health information. Unlike the present work, the authors investigate a technique that combines linear and quadratic regression to predict the trend of some measured variables such as air compressors. Unlike Bangert (2020); Li and Nilkitsaranont (2009), our approach relies on a more powerful model such as DTs to forecast GTs' future conditions. Various other GT prognostic techniques have been investigated, (DePold and Gass, 1999; Brotherton *et al.*, 2000; Roemer and Kacprzynski, 2000; Byington *et al.*, 2002; Brotherton *et al.*, 2002; Roemer *et al.*, 2006; Hess *et al.*, 2006). Most of these techniques are experience-based prognostics, model-based prognostics, evolutionary prognostics, neural networks, state estimator prognostics, rule-based expert systems, and fuzzy logic-based methods. They apply different methods to predict GT future conditions. Unlike these approaches, the current paper develops a data processing and feature engineering pipeline aimed at forecasting the trend of GT rotational speed, with the final goal of raising a warning sign about future trip occurrence.

Lastly, recent studies by Hong and Kim (2023); De Castro-Cros *et al.* (2021) employed machine learning techniques to predict gas turbine engine performance. Hong and Kim (2023) utilized operational parameters like fuel flow and compressor discharge pressure, employing artificial neural networks and support vector regression algorithms. Their approach accurately predicted the gas turbine's performance. Similarly, Hong and Kim (2023); De Castro-Cros *et al.* (2021) developed an artificial neural network model using data from a heavy-duty gas turbine unit, incorporating various operational parameters. The resulting model successfully predicted the gas turbine engine's performance. Both studies demonstrate the effectiveness of machine learning-based methods in accurately forecasting gas turbine engine performance using operational data.

Materials and Methods

In this section we present the data we considered, the preprocessing we applied to it and the ML algorithms we applied.

Available Data

The rotational speed is predicted in this study by using data from various sensors collected from GTs produced by Siemens. A measured variable is a reading from a sensor acquired with a given frequency during machine operation. In this study, 128 measured variables are considered, Table 2. The set of variables includes different types of system variables typically considered during GT monitoring and diagnostics, e.g., ambient conditions, gas path measurements, vibrations, rotational speed, power output, as well as operational parameters of external systems. More precisely, gas path measurements comprise compressor inlet and outlet pressures, compressor inlet and outlet temperatures, exhaust gas temperatures (i.e., a ring of sixteen equally spaced thermocouples placed circumferentially at the turbine outlet section, each with two redundant thermocouples), and turbine exhaust pressure. Vibration data consist of both bearing vibrations and combustion chamber pulsations. Finally, operational parameters such as fuel temperature, fuel valve position lube oil pressure, and temperature are also taken into account. Therefore, this study employs a massive and comprehensive set of measured variables. The high number of measured variables, exploited to predict the future trend of GT rotational speed, poses the challenge of handling big data. Moreover, since different measured variables are sampled at different frequencies, an interpolation operation is done in order to have data for each measured variable at each minute. Therefore, each feature considered in the rest of the paper is the value of a measured variable at a specific time (minute).

We use a dataset, Table 1 that includes 6490 transients (trip and shutdown) and steady-state examples acquired over three years from machines in different countries. The dataset is proprietary and it is not possible to make it freely available. It should be noted that the two categories are balanced, the transient category includes 70 trips and 3221 shutdowns. Sampling data randomly from this distribution for training an ML model could be problematic. Hence, it is necessary to develop an ad hoc strategy for selecting representative examples for training an ML model.

Table 1: Dataset distribution

Category	#Example
Transient	3291
Steady-state	3199

Table 2: List of measured variables

Measured variable description	Number of sensors
Air intake differential pressure	1
Ambient air humidity	1
Ambient air temperature	1
Differential pressure at compressor inlet section	1
Compressor inlet pressure	1
Compressor inlet temperature	1
Compressor outlet pressure	3
Compressor outlet temperature	3
Power output	1
Differential pressure at exhaust duct section	1
Fuel temperature	1
Turbine outlet pressure	1
Bearing vibration	6
Bearing vibration 1× N Amp	6
Turbine exhaust temperature	48
Gearbox casing vibration	2
Gearbox casing vibration 1× N Amp	2
Generator DE bearing temperature	1
Generator DE vibration	3
Generator NDE vibration	3
Generator stator temp pH	12
High-frequency pulsation	3
Low-frequency pulsation	3
Lube oil supply pressure	3
Lube oil supply temperature	4
Fuel valve position	1
Medium frequency pulsation	3
Narrow frequency pulsation	3
Combustor chamber pressure	1
Radial bearing temperature	4
Thrust bearing temperature	2
Turbine casing temperature	1
Rotational speed	1

Different ML models are trained using data associated with those 128 measured variables. The models predict the rotational speed of the next 4 h using the data of the past 20 h (including the rotational speed), Fig. 1. Therefore, the input to the models consists of 128·20·60 = 153600 features while the output includes 4·60 = 240 values of rotational speed.

Machine Learning Algorithms

This section describes the different ML algorithms used to predict the rotational speed in GTs. Three ML models are investigated: DTs (Breiman *et al*., 1984) ERT also called Extra Trees (ETs) (Geurts *et al*., 2006) which are a modification of Random Forests (RFs) (Ho, 1995; Breiman, 2001) and KNNs (Aha *et al*., 1991). DTs are ML algorithms that can be used in predictive tasks. A DT is a tree-like structure in which each node represents a test on a specific feature. Each leaf node represents an expected value (list of values) in the case of (multi-output) regression. Figure 2 for an example of a multi-output regression DT. Given an unknown example, a DT performs inference by evaluating the conditions on the nodes from the root to a leaf. The numerical list of values associated with the leaf is assigned to the example. One of the main characteristics of DTs is that they are explainable, in the sense that the model can be interpreted by a human. In fact, a path from the root node to a leaf can be translated into a human-readable logical rule. The training algorithm for building DTs Fig. 2 Example of DT. Adapted from (Losi *et al*., 2021a-b) works in a top-down manner. The algorithm starts with the whole set of training examples that are associated with the root and splits them into two partitions by greedily selecting the feature and the threshold that create the purest partitions according to an impurity measure. Then two child nodes are created, one for each partition of the training examples. The algorithm stops when the considered subset of examples is pure or a limitation imposed by a hyperparameter is reached. The most common hyperparameters are the size of the set associated with the leaf called, min sample leaf, and the depth of the tree, called max depth.

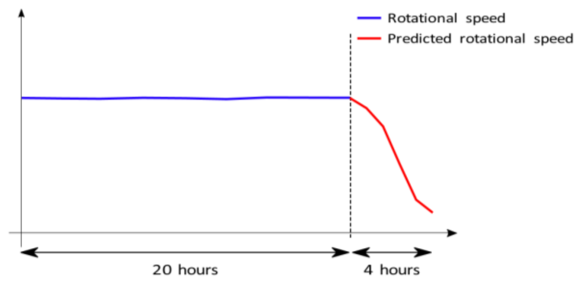


Fig. 1: Rotational speed prediction

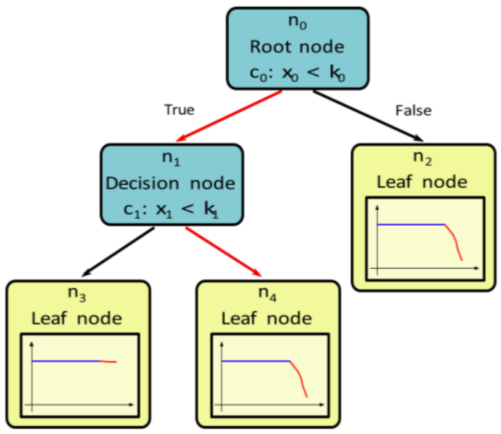


Fig. 2: Example of DT. Adapted from (Losi *et al.*, 2021b)

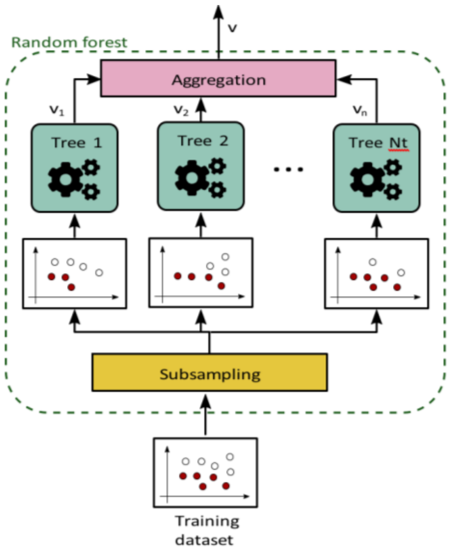


Fig. 3: Example of RF, (Losi *et al.*, 2021b)

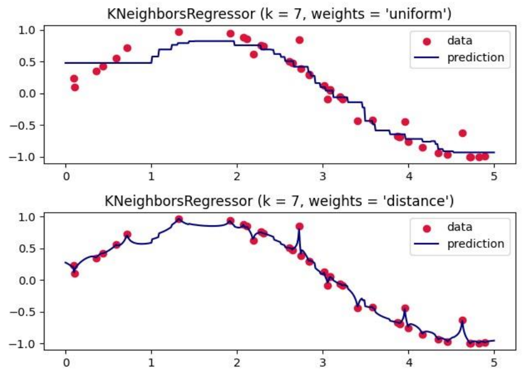


Fig. 4: Example of KNNs regression in 1 dimension: The X-axis consists of 40 sorted random values between 0 and 5 and the Y-axis is the sine values of X with added noise. It applies k-nearest neighbors regression with uniform and distance weight respectively. The data points are visualized in crimson and predicted in blue

Ensemble methods are a group of ML techniques that consist of aggregating several estimators (possibly of different types). These techniques are useful because an ensemble estimator could lead to better performance compared to that of any of the individual estimators that compose the ensemble.

An RF, Fig. 3. is an ensemble of DTs. The DT estimators are independently trained on different (randomly sampled) subsets of the training dataset and/or with different (randomly sampled) subsets of features. The prediction of an RF is a combination of the predictions obtained by the DTs that compose the ensemble. If the random forest is performing regression, the predicted value is the mean of the outputs of the DTs.

ETs are another ML ensemble method similar to RFs that introduce more randomness during tree building. Figure 4 Example of KNNs regression in 1 dimension. The X-axis consists of 40 sorted random values between 0 and 5 and the Y-axis is the sine values of X with added noise. It applies the k-nearest neighbor's regression with uniform and distance weight respectively. The data points are visualized in crimson and the predicted in the blue main difference between ETs and RFs is that, during the training of each inner DT estimator, RFs choose the optimum threshold for each feature whereas ETs choose it randomly, hence the name extremely randomized trees. This makes ETs faster to train than RFs because finding at every node the best threshold for each feature is one of the most computationally expensive tasks of the training process.

KNN is a versatile algorithm applied to both classification and regression tasks. It computes distances between the input data point and every entry in the training set, relying on metrics such as the Euclidean or Manhattan distance. By doing so, it identifies the k data points nearest to the input. For classification, the algorithm identifies the most common class among these k-neighbors, attributing it to the input data point. Conversely, regression calculates the weighted average of the target values of these k nearest neighbors. The choice of k and the distance metric significantly influence the algorithm's performance. Smaller k values heighten sensitivity to noise, whereas larger ones enhance stability, albeit at the expense of computational complexity. In regression tasks, Fig. 4. several hyperparameters come into play, including weights and p. weights, which can be either uniform or distance, define how contributions of neighboring points are weighted. In the uniform setting, all points within a neighborhood hold equal weight, while in the distance setting, points are weighted in a way inversely proportional to their distances from the target point. The parameter p represents the power parameter for the Murkowski distance metric. For instance, p = 1 corresponds to the Manhattan distance (L_1 norm), while p = 2 to the Euclidean distance (L_2 norm).

The ML workflow used in this study is illustrated in Fig. 5. All the measured variables acquired from Siemens GTs located worldwide are sent to a data warehouse, which is in charge of data storage. Given the large amount of gathered data, lossy compression techniques are needed and, as a result, the measured variables are stored in the data warehouse with different sampling frequencies, depending on the considered sensor.

Data Preprocessing

The first step that must be performed is to extract raw data for the machines of interest from the data warehouse. Then, the raw data are grouped by machines, sorted by timestamps, piecewise linearly interpolated, and then new data points are created by resampling the interpolated curves. Finally, the resulting time series of the rotational speed is stored as Comma Separated Values (CSV) files to be used for detecting transients and steady-state cases. In the following, we illustrate these steps in detail.

Raw Data Extraction

Given a set of measured variables and a GT, the aim is to extract data from user-defined time windows as fast as possible. We developed a script that creates a pool of threads running in parallel, so that each of them sends an SQL query to the cloud data warehouse that executes it, Fig. 6. The pool of threads is managed by a single client computer and, in order to avoid high memory usage, SQL server cursors were used and the data rows were concatenated in CSV files.

Each CSV file contains the data concerning the set of the selected GT-measured variables in a given time window. Finally, the resulting CSV file is zipped in order to reduce storage consumption.

Data Elaboration and Storage

As mentioned before, the data warehouse contains data sampled with different frequencies; thus data must be elaborated in order to be sampled with the same frequency since the training models used in this study are aimed at processing multi-variate time series data. The GTs taken into account include different sensors with different sampling frequencies, e.g., high-frequency sensors such as vibrations or low-frequency sensors such as ambient temperature and air humidity. Each CSV file obtained in the previous step is unzipped and read. Then, for each measured variable related to a low-frequency sensor, the raw data are ranked by timestamps, piecewise linearly interpolated, and then new data points are created by resampling the interpolated curve as shown in Fig. 7. For each measured variable related to a high-frequency sensor, the raw data are ranked by timestamps and, for every minute, the average is computed. Finally, for each CSV file that contains the raw data retrieved from the data warehouse, the result of the elaboration is a Multi-Variate Time Series (MTS), which is stored as a CSV file that in turn is zipped to save storage space (Losi *et al*., 2023a-b).

Identification of Transient Operation

In order to build a dataset that includes examples of trip and shutdown, an algorithm for identifying transients is implemented. Since the database does not always include the operational reports regarding past occurrence of transients and their type for all fleets of GTs, an appropriate technique to identify transients only based on field data is required. Based on an analogous approach used to identify starting events (Fontes and Pereira, 2016; Losi *et al*., 2023a-b) we introduced an approach that applies the subsequent matching algorithm (Fu, 2011) to identify transient events from rotational speed data.

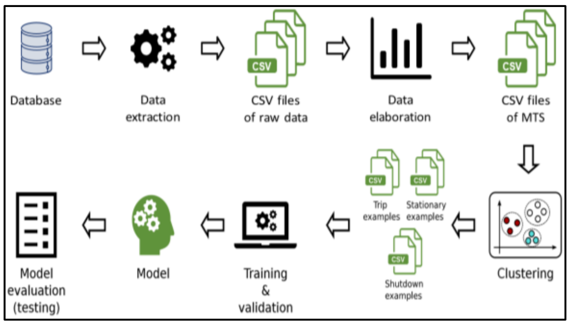


Fig. 5: Machine-learning workflow. Adapted from (Losi *et al*., 2021b)

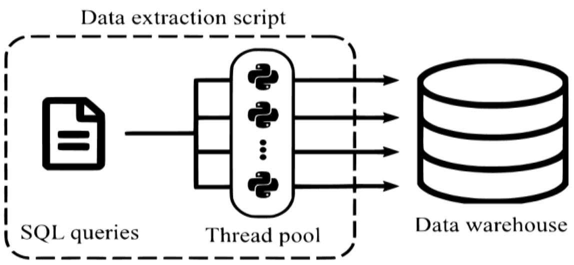


Fig. 6: Parallel query execution during raw data extraction, (Losi *et al*., 2021b)

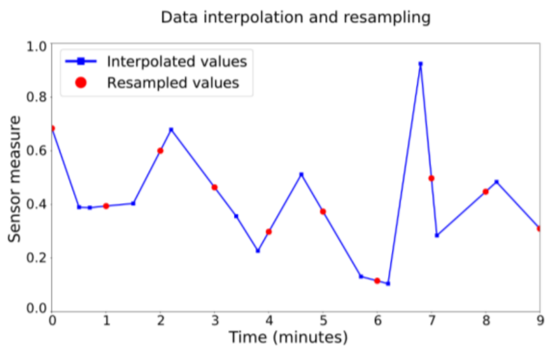


Fig. 7: Interpolation and resampling of measures of a generic asynchronous sensor, (Losi *et al*., 2021b).

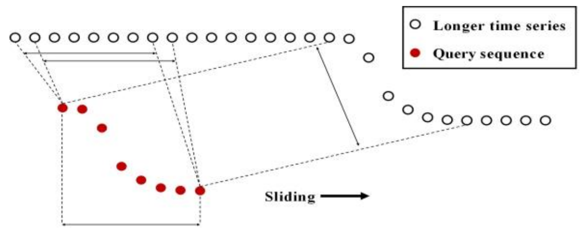


Fig. 8: Subsequence matching algorithm

The subsequent matching algorithm considers a two-time series, usually defined as a query sequence and a longer time series (Fu, 2011), and identifies the subsequences in the longer time series that match the query sequence. In our case, the query sequence is computed by analyzing 8 min of the rotational speed of 50 transient events (Losi *et al*., 2021a-b). That 8 min is selected because the rotational speed decreases from over 6000 rounds per minute (rpm) to a few rpm. We obtain a 50×8 matrix, where each row represents a transient event. Then we compute the column-wise mean of this matrix obtaining a vector that represents the typical profile of the rotational speed during a transient of the GT and consists of 8 data points, i.e., 8 min of rotational speed values. Instead, the longer time series is composed of the historical data of the rotational speed.

Figure 8 the subsequence matching algorithm computes the Euclidean distance between the query sequence and the longer time series within a moving time window that has the same length as the query sequence. The moving window slides forward one data point at a time, i.e., 1 min at a time.

For each machine, the output of this algorithm is a list of tuples ordered by timestamp of the form $\langle T_i, E_i i\rangle$, where T_i is the timestamp associated with the last data point of the time window and E_i the Euclidean distance of the time window. If the Euclidean distance of a given tuple is lower than the distances of the previous tuple and the following one, then that Euclidean distance is a local minimum. Tuples, where the Euclidean distance is a local minimum, represent those time windows where the longer time series is similar to the query reference. Moreover, if the local minimum is lower than a threshold, then the associated time window is considered as a transient event and the last data point of the time window is added to a list of transient timestamps. The value 2000 rpm is chosen as a threshold that leads to a clear distinction between steady-states and transients.

Moreover, for training and testing our machine learning models, we consider as valid transient examples those cases where, during the 23 h and 52 min before the start of the transient event, the rotational speed was greater than 60% of the maximum nominal speed, which corresponds to the minimum GT load. Therefore, from the list of transient timestamps, those timestamps associated with transient events that do not meet this constraint are removed.

The final output of the transient identification step is a CSV file containing a list of tuples of the form $\langle M_j, T_j i\rangle$, where M_j is a code that identifies a GT and T_j is a transient timestamp, i.e., the timestamp of the end of a transient event.

Identification of Steady-State Operation

For each machine, we compute the time intervals during which the rotational speed is greater than 60% of the maximum nominal speed and we filter out all the intervals that are shorter than 24 h. Then we split these intervals into a set of 24 h intervals such that they do not overlap. The output is a CSV file containing a list of tuples of the form $\langle M_j, T_j i\rangle$, where M_i is a code that identifies a GT and S_i is a timestamp that denotes the end of the interval.

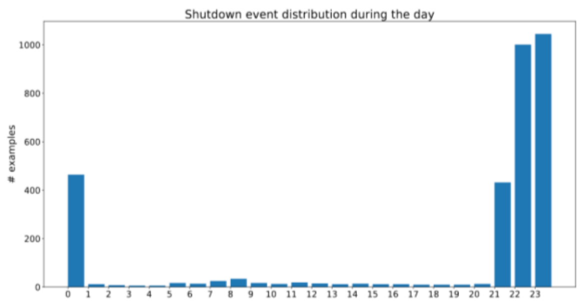
Selection of Training Examples

Before performing training, we analyze the MTS examples extracted from the data warehouse. First, we analyze the transient examples and we distinguish the trip events from the normal shutdowns by exploiting the clustering approach defined in Losi *et al*. (2023a-b). The clustering approach applies feature-based clustering embedding the unsupervised fuzzy c-means algorithm, see (Yang and Wu, 2006). The aim of clustering is to partition a number N of GT transients into a number, significantly smaller than N, of clusters that represent different operating conditions. This process allows the discrimination of trips from normal shutdowns. In the last step, cluster labels are assigned to classify each transient. For more details on the clustering approach (Losi *et al*., 2021a-b).

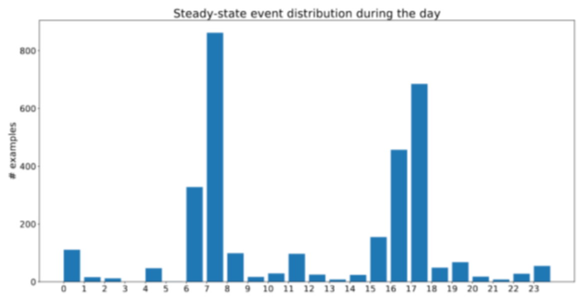
Second, we analyze the distribution of the normal shutdown events over the h of the day. In particular, for each MTS shutdown example, we extract the timestamp of the last row, which corresponds to the last minute of a shutdown event, then we bin them into the h of the day and analyze the resulting distribution.

Figure 9a shows that more than 91% of shutdown events terminate overnight between 21:00-00:59. This happens because there are some GTs which are often intentionally shutdown between 21:00 and 00:59. If we randomly sample the shutdown examples for training, we can reasonably think that around 91% of sampled examples correspond to shutdown events that fall in the interval between 21:00-00:59.

Figure 9b shows the distribution over the h of the day of the steady-state events extracted from the database. In this case, roughly 80% of the steady state events terminate during daylight between 06:00-08:59 and between 15:00 and 17:59.



(a)



(b)

Fig. 9: Shutdown and steady-state distribution; (a) Distribution of shutdown events over the hours of the day; (b) Distribution of steady-state events over the hours of the day

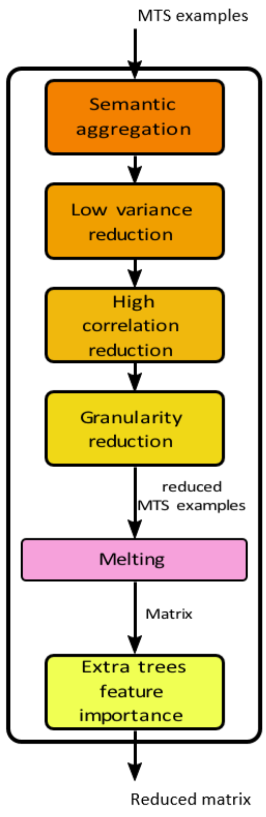


Fig. 10: Feature engineering pipeline

Feature Engineering

Feature engineering is a critical aspect of the success of an ML project (Domingos, 2012) because it helps to reduce the training cost and improves prediction accuracy (Kuhn and Johnson, 2019). The two main steps of feature engineering are feature selection (Cai *et al*., 2018) and feature extraction (Guyon *et al*., 2008). Feature selection allows reducing the number of features by keeping only the relevant ones. This helps to significantly reduce training time. It could simplify the trained model and make it more understandable and interpretable. Moreover, it also avoids the so-called curse of dimensionality Koutroumbas and Theodoridis (2008); Köppen (2000) which states that learning in higher dimensions is harder than in lower dimensions. Reducing the number of features eases the training process and makes the model more likely to find optimal patterns in the data. Feature extraction creates new features by combining the existing ones. Figure 10 shows all the steps of our feature-engineering pipeline that combines the measured variables, reduces their number, and then, after the melting process, reduces the number of features.

In the following subsections, the steps of the feature-engineering pipeline are described in detail.

Semantic Aggregation

Semantic aggregation is a method in ML that combines similar features that potentially share information. It generally summarizes those features in a reduced set by keeping only information that the reduced features have in common. In this first step of feature engineering, we aggregate the measured variables obtained from the sensors "Exhaust Temperature *n*", where n = 1⋯16. Those measured variables were grouped into 4 other groups as follows:

(i) Exhaust temperature 1-4
(ii) Exhaust temperature 5-8
(iii) Exhaust temperature 9-12
(iv) Exhaust temperature 13-16

Each group is aggregated by computing, at each minute, the mean, standard deviation, kurtosis, and skewness of the tags included in the group. Then the aggregated features (measured variables, minutes) replace the original ones.

Low Variance Reduction

One common approach of feature selection used in ML is low variance reduction, which consists of removing all features with low variance.

In our case, measured variables that are (almost) always constant in all the MTSs do not provide any informative content, hence they can be removed. Therefore, as shown in Eq. 1, for each measured variable X_j, the variance on the i^{th} MTS is computed. Then, we summed all the variances and we averaged the result over N, the number of the MTS examples. Finally, all measured variables whose variances are below a certain threshold are removed:

$$\frac{1}{N}\sum_{i=1}^{N} Var_i\left[X_j\right] \forall \, measure\ variable\ j \tag{1}$$

By analyzing the variance of all the measured variables, we set the mean-variance threshold at 0.0002. This threshold allows removing more than 40% of features. Figure 11 shows the mean-variance of the measured variables. The measured variable with the highest mean variance is ambient air humidity. This means that the environmental conditions at which the measured variables were recorded are considerably different.

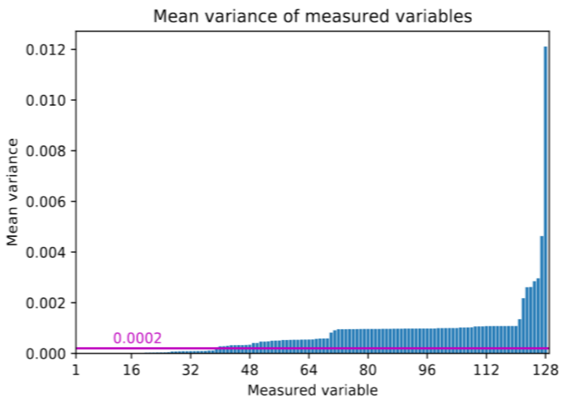


Fig. 11: Mean-variance of all the measured variables averaged over all the examples

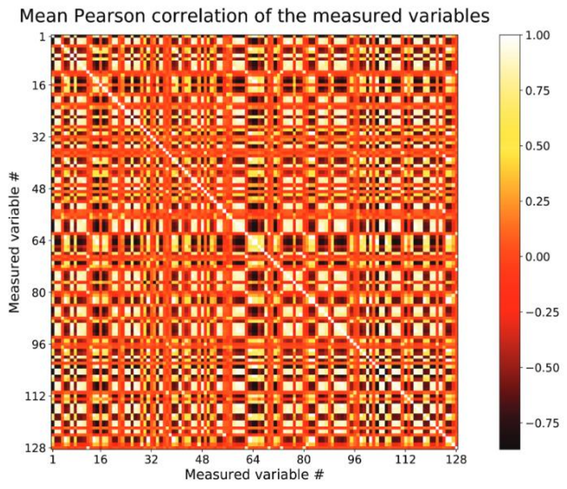


Fig. 12: Matrix of mean Pearson correlation between all couples of measured variables

High Correlation Reduction

A feature that is strongly correlated with another one can be seen as duplicated information. For this reason, we compute the mean Pearson correlation between each pair of variable time series averaged over the number of examples.

Figure 12 shows the mean Pearson correlation of the variables. It can be observed that several variables are highly correlated. In order to reduce the number of features, for each couple of variables, if the absolute value of the correlation between two variables is larger than Fig. 12. Matrix of mean Pearson correlation between all couples of measured variables. 0.85, we remove one of them. Note that Pearson correlations greater than 0.8 are often considered as very strong correlations, (Akoglu, 2018).

Granularity Reduction

The granularity reduction step consists of reducing the number of points in the MTSs. For each measured variable, instead of having a value every minute, we can have one or more values every *n* minute. In our training process, we set *n* to 10, and for every 10 min the following statistics are computed:

(i) Mean
(ii) Standard deviation
(iii) Kurtosis
(iv) Skewness

The original 10 columns for the variable are replaced with 4 new columns. Note that, to reduce the number of features, *n* should be greater than 4. We set *n* to 10 because it is a tradeoff between reducing the number of features and maintaining relevant knowledge inside the data.

Melting

In MTS problems, the melting operation converts a dataset into a format ready for the ML algorithms. In our case, each example consists of 128 measured variables and each measured variable has 20·60 = 1200 values. Therefore, each MTS example, *X* data, represented by a matrix, is transformed into a vector *X* data transformed as follows:

$$X\_data = \begin{pmatrix} MV_1 & MV_2 & \dots & MV_{128} \\ x_{1\_1} & x_{1\_2} & \dots & x_{1\_128} \\ x_{2\_1} & x_{2\_2} & \dots & x_{2\_128} \\ \dots & \dots & \dots & \dots \\ x_{1200\_1} & x_{1200\_2} & \dots & x_{1200\_128} \end{pmatrix}$$

$$\Downarrow$$

$$X\_data\_transformed = \left[\underbrace{x_{1\_1}\dots x_{1200\_1}}_{MV_1}\ \underbrace{x_{1\_2}\dots x_{1200\_2}}_{MV_2}\dots\underbrace{x_{1\_128}\dots x_{1200\_128}}_{MV_n}\right]$$

where, *MV* means measured variable, $x_{i\text{-}j}$ is the value of a measured variable *j* at the time (minute) *i*. *X data transformed* represents a single example in the dataset.

Extra-Trees Feature Importance Reduction

In order to further reduce the number of features, we use an ET al algorithm for identifying the most important features. We measure the importance of a feature by looking at how much the tree nodes that use a particular feature reduce the impurity (Mean Square Error (MSE) in our case) on average across all the trees. More precisely, for each feature, the associated MSE importance is computed. The feature is ordered according to the associated MSE importance and the most important features are selected.

Results and Discussion

This section presents the results of the experiments made in this study for predicting the trend of the rotational speed in GTs. Moreover, we discuss the results. The implementations of the machine learning algorithms used in the experiments are available in the scikit-learn suite (https://scikit-learn.org/).

DTs, ET sand KNNs are trained using a grid search that finds the best hyper-parameters by relying on accuracy. The best hyperparameters are then used to train the final models. The following sections present and discuss the accuracy adopted, the training, and the evaluation process respectively.

Accuracy

We use accuracy to measure how many transients and steady-state cases are correctly identified. For every example, the Mean Absolute Error (MAE) of the last 10 predicted points is computed: If the MAE is lower than a fixed threshold, then the predicted curve is correct. The accuracy is given by the following equation:

$$Accuracy = \frac{\#correct\ predictions}{\#example}$$

The threshold plays a crucial role in determining whether the prediction is correct or not. For this reason, a sensitivity analysis about the influence of the value of the threshold is carried out on ET and DT models with different configurations: The first case uses only examples of trip and shutdown, the second only examples of trips and stationary and the last uses all examples.

After training, the MAE of the last 10 points is computed and the result is included in a scatter plot, with the aim of analyzing the distribution of errors.

The analysis of the scatter plots highlighted a trend toward the distribution of the error in two distinct regions (Fig. 13): A region with a centroid at MAE = 200 (perfect prediction) and another with a centroid at MAE = 2500 (wrong prediction).

The intermediate cases are analyzed individually by looking at the predicted curves and identifying the optimal threshold of 1000 as a discriminant between the correct and incorrect examples.

Training and Evaluation

Two learning pipelines are investigated as described in the following sections.

Learning without Feature Selection

We first train different ML models using a simple (shallow) training pipeline that does not include feature selection, Fig. 14.

These distributions can cause serious issues in our training process, because our training examples are, in terms of time, highly biased. In order to avoid biasedness, a stratified sampling based on the h of the day is performed in such a way that the training dataset is uniformly distributed in terms of time as much as possible, Figs. 15a-b.

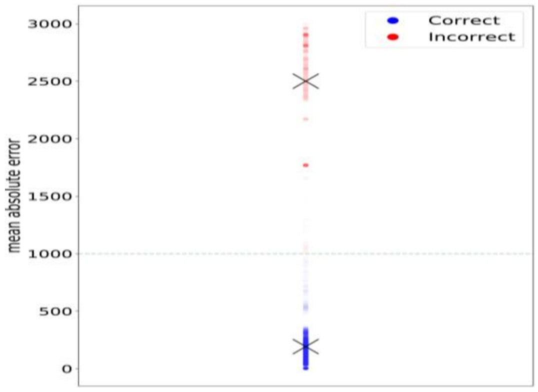


Fig. 13: Threshold analysis: Error distribution

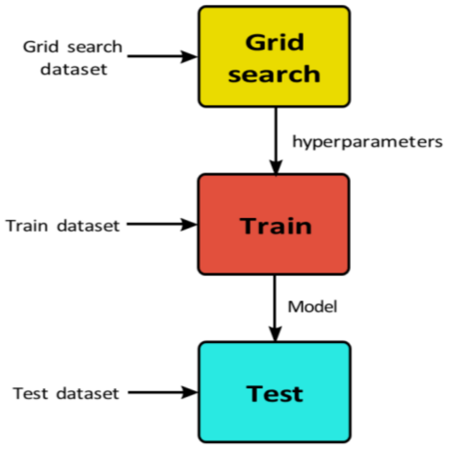
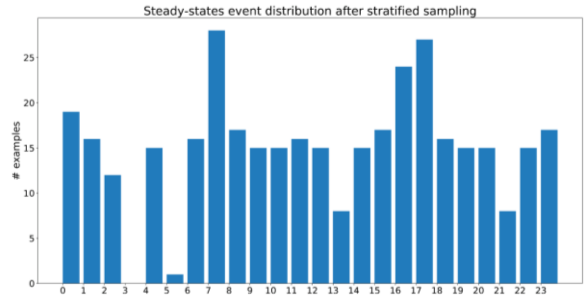
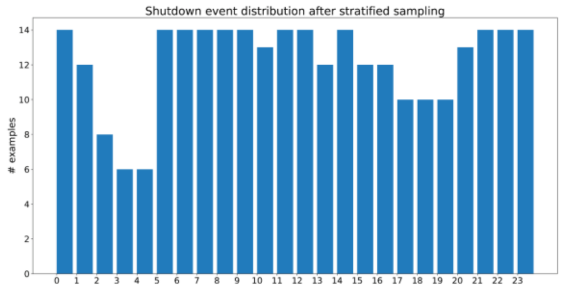


Fig. 14: Learning pipeline without feature selection

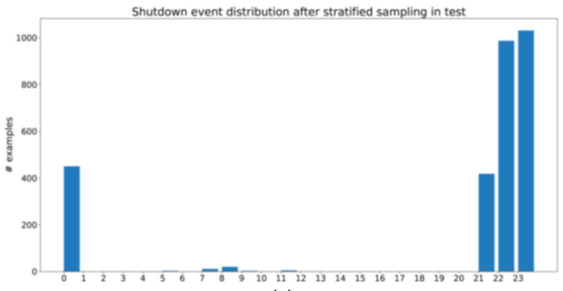


(a)

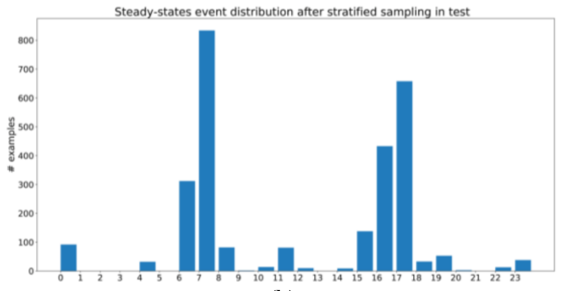


(b)

Fig. 15: Training distribution; (a) Distribution of steady-state events used for training over the hours of the day after performing the stratified sampling; (b) Distribution of shutdown events used for training over the hours of the day after performing the stratified sampling



(a)



(b)

Fig. 16: Testing distribution; (a) Distribution of shutdown events used for testing over the hours of the day; (b) Distribution of steady state events used for testing over the hours of the day

In the first experiment, we perform, as already explained, a stratified sampling of examples based on the h of the day. After performing the stratified selection of examples, we obtained the following training distribution: 362 transient examples, including 292 shutdown examples 70 trip examples, and 362 steady-state examples. All the remaining examples are used as a test set. Detailed distribution with respect to the h of the days after performing the stratified sampling for shutdown and steady-state training examples are depicted in Fig. 16a-b respectively.

Two models, DT and ET, are trained following the learning pipeline in Fig. 13. We chose ET, instead of RF because it is faster to train. Initially, a grid search is performed to find the best hyperparameters. The following hyperparameters are explored for DT: Max depth: [2, 5, 10, 20], min samples leaf: [1, 5, 10, 20] and the following for ET: Max depth: [2, 5, 10, 20], max features: [100, 500, 1000, 10000], min samples leaf: [5] and n estimators: [100, 500].

To perform the grid search, a four-fold cross-validation is used for each combination of hyperparameters. The choice of a four-fold cross-validation adopted is a compromise between the number of hyperparameters to explore and the time to perform the experiment. The following abbreviations are used in the following tables: Max-depth (m d), min-samples leaf (m s l), max features (m f), and n estimators (n e). Tables 3-5 show the values of the accuracy of the top three best combinations for DT, ET, and KNN respectively. Note that in Table 4 by feature we mean an attribute of the form (measured variable, minute) at a given minute.

Table 3: Learning without feature selection setting: Grid search for DTs

Hyperparameters		Evaluation
M_d	M_s_l	Accuracy
2	20	0.867
5	5	0.849
10	20	0.796

Table 4: Learning without feature selection setting: Grid search for ETs

Hyperparameters				Evaluation
M_d	M_f	M_s_1	N_e	Accuracy
5	500	5	100	0.894
5	500	5	500	0.898
5	10000	5	500	0.874

Table 5: Learning without feature selection setting: Grid search for KNNs

Hyperparameters			Evaluation
K	Weights	P	Accuracy
1	uniform	1	0.896
1	distance	1	0.896
3	distance	1	0.890

After performing a grid search, hyperparameters showing good performance are identified to train the final model. In our case, four combinations of hyperparameters are chosen for training: Two DT models, one ET and one KNN as follows: DT_1: Max depth = 2, min samples leaf = 20; DT_2: Max depth = 5, min samples leaf = 5; ET: Max depth = 5, max features = 500, min samples leaf = 5, n estimators = 500; KNN: K = 1, weights = uniform, p = 1. Two DTs are chosen: The one with the best accuracy and another one with slightly worse performance (used to check the usefulness of the grid search). Among all the ETs, the one with the best accuracy is chosen. Regarding the KNN, the first two combinations yield identical accuracy. However, the first one was selected because it is faster. The examples for training and testing the models are distributed as shown in Table 6. Note that the number of steady-state examples for training is equal to the number of transient examples which is the sum of the number of trips and shutdown, 348 = 292+56. Moreover, after training, the model is tested on the rest of the available data.

Considering transients as positive examples and the steady state as negative examples, in addition to accuracy, other metrics such as *Precision*, *Recall*, *F1-score,* and *Specificity* are also computed. Let us denote by True Positive (*TP*) the number of transients correctly classified, False Positive (*FP*) the number of transients wrongly classified as steady states, True Negative (*TN*) the number of steady states correctly classified and False Negative (*FN*) the number of stead-states wrongly classified as transients. The *Precision*, Eq. 2, computes the proportion of examples classified as positive that are correctly classified:

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

The *Recall* (also called sensitivity), Eq. 3, computes the proportion of the actual positive examples correctly classified:

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

The *F*1*-score* is a way of combining the precision and recall of the model. It is defined as the harmonic mean of the precision and *recall,* Eq. 4 and reaches its optimum 1 only if precision and recall are both at 100%:

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (4)$$

The specificity, Eq. 5, computes the proportion of examples classified as negative that are correctly classified.

Table 6: Distribution of training and testing examples

-	Training	Test
#transient	348	2943
#steady-state	348	2851

Table 7: Detailed test results

Model	Precision	Recall	F1-score	Specificity	Accuracy
DT_1	0.835	0.847	0.841	0.827	0.837
DT_2	0.782	0.829	0.805	0.761	0.796
ET	0.888	0.842	0.865	0.891	0.866
KNN	0.919	0.924	0.922	0.912	0.920
DT_1 with feature selection	0.878	0.877	0.878	0.875	0.876
KNN with feature selection	0.924	0.926	0.925	0.921	0.924

$$Specificity = \frac{TN}{TN + FN} \quad (5)$$

Table 7 shows the final results using the combinations of previously chosen hyperparameters.

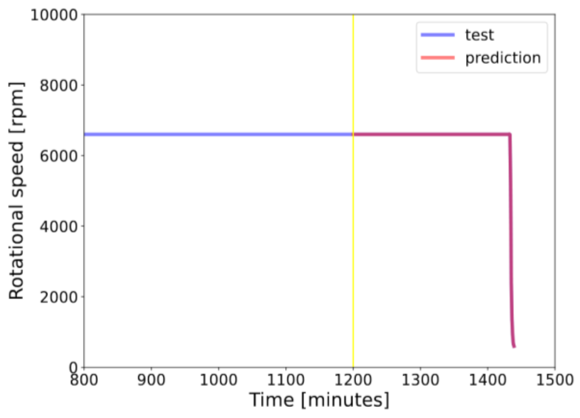
Overall, Tables 3-5 and 7 show that the models are able to generalize as well as extract patterns in the data that are useful for predicting the trend of the rotational speed.

Learning with Feature Selection

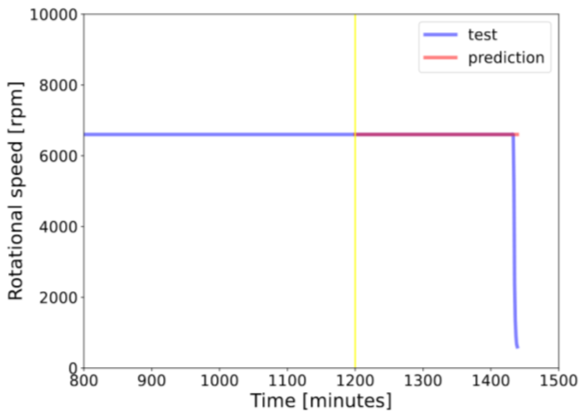
In order to improve the results obtained in the previous section, we extend the training pipeline by adding the steps of feature selection described in Section 5. We start with 128, 20, 60 = 153600 features and obtain the following number of features after each step: After performing Semantic aggregation, the number of features becomes 115200 i.e., 25% of the input features are removed. After applying low variance reduction, with a variance threshold = 0.0002 we obtain 69600 i.e., ~ 40% of features are removed. The mean Pearson correlation reduction brings the number of features to 37200 which means that more than ~ 47% of features are being removed. Granularity reduction reduces the number of features to 14880, i.e., 60% of features are removed. The melting operation does not reduce the number of features but prepares the data for training as explained. Finally, an ET, used for feature selection, reduces the number of features to 3720, i.e., a reduction of 75%. Overall ~ 98% of the initial features are removed.

We train DT_1 and KNN after feature selection and the results are shown in the last two rows of Table 7. It can be observed that feature selection contributes to improving the accuracy. Examples of transients (trip and shutdown) and steady-state classifications are depicted in Figs. 17-18 by showing the normalized rotational speed trend over one day (normalized data are reported because of confidentiality reasons).

It is worth noting that KNN provides better accuracy with respect to other classifiers trained with or without feature selection. However, DT1 trained with feature selection is a good tradeoff between prediction quality and explainability.

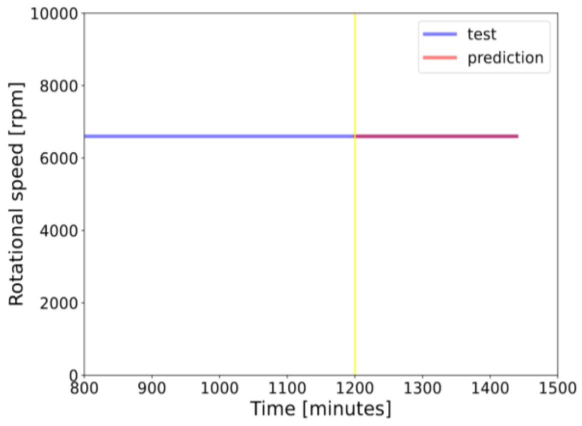


(a)

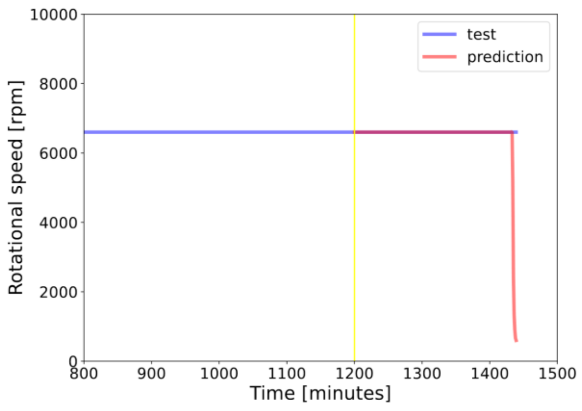


(b)

Fig. 17: Examples of correct and wrong transient predictions; (a) Transient correct prediction; (b) Transient wrong prediction



(a)



(b)

Fig. 18: Examples of correct and wrong steady-state predictions; (a) Steady-state correct prediction; (b) Steady-state wrong prediction

Conclusion

In this study, we investigated machine-learning approaches based on DTs, ETs, and KNNs for predicting the rotational speed of a GT. The models developed predict the rotational speed of the next 4 h by using data from multiple measured variables of the previous 20 h. An ML workflow starting from sensor data extraction from the Siemens database to an ML pipeline for training and predicting the rotational speed trend of GTs was applied. Three ML pipelines, with and without feature selection, were investigated to predict the rotational speed of GTs. KNNs with and without feature selection provide the best results with an accuracy of more than 92%. However, DTs with feature selection provide a good compromise between prediction quality and explainability. The trained models are not only able to correctly predict transients but are also able, in the case of DTs, to give an explanation of their decision.

As a future work, we first plan to try to predict with a 24 h advance in order to timely detect incipient GT transient symptoms.

Moreover, we plan to focus on interpretability and model explainability, especially in the context of Decision Trees (DTs). Exploring techniques Such as Shapley Additive Explanations (SHAP) (Nohara *et al*., 2019) values and LIME (local interpretable model-agnostic explanations) (Plumb *et al*., 2018) can provide deeper insights into how our models arrive at specific predictions. Understanding the rationale behind the predictions is crucial, especially in critical industrial applications where decision-making transparency is paramount.

Additionally, we plan to integrate anomaly detection algorithms with predictive modeling to identify unusual patterns and outliers in data, not only detecting transient symptoms but also highlighting anomalous behaviors indicating underlying issues. Second, they aim to incorporate real-time data streams and sensor information into their prediction models, creating dynamic and adaptive systems capable of responding promptly to changing operating conditions. This integration will enable timely alerts and preventive maintenance suggestions, ensuring uninterrupted gas turbine operation and preventing potential failures.

Lastly, we plan to design and implement a user-friendly interface for industry professionals and operators. This interface will allow users to visualize predictions, explore model explanations, and receive actionable insights in an easily understandable manner. The focus on usability and accessibility is crucial, ensuring the practical applicability and adoption of their predictive models in real-world industrial settings. Through these efforts, the authors aim to advance gas turbine predictive maintenance, offering valuable tools to the energy sector. Their goal is to enhance the operational efficiency, cost-effectiveness, and overall reliability of gas turbine systems, contributing significantly to the industry's advancements.

Acknowledgment

The authors gratefully acknowledge Siemens Energy for the permission to publish the results. The authors also wish to express sincere gratitude to Dr. Giuseppe Fabio Ceschini for inspiring and supporting the research activity about gas turbine trip prediction.

Funding Information

The research was funded by Siemens Energy Srl.

Author’s Contributions

Arnaud Nguembang, Fadja Giuseppe Cota, and Francesco Bertasi: Made significant contributions to the paper across various dimensions. Their involvement encompassed conceptualization, methodological design, software development, validation procedures, formal analysis, and the creation of visualizations.

Additionally, they played a pivotal role in crafting the original draft and contributed extensively to the review and editing phases, ensuring the overall quality and coherence of the manuscript.

Fabrizio Riguzzi: Demonstrated a comprehensive and multifaceted involvement in the project.

From the initial conceptualization and methodological framework development to the creation of necessary software, rigorous validation, and formal analysis, Fabrizio’s contributions were foundational.

Furthermore, Fabrizio actively engaged in the entire writing process, from crafting the original draft to meticulous review and edited.

Although no impactful visualizations were produced, Fabrizio’s role extended to effective supervision and adept project administration, showcasing a well-rounded and fluent presence throughout the project lifecycle.

Enzo Losi and Lucrezia Manservigi: Made significant contributions throughout the research process. Their involvement spanned from the conceptualization of ideas to the meticulous validation of results.

They actively participated in crafting the original draft and played a crucial role in the comprehensive review, editing, and visualization stages.

Their contributions have been integral to the development and refinement of this study.

Mauro Venturini and Giovanni Bechini: Played a vital role in shaping and guiding this project. Their involvement extended from the initial conceptualization of ideas to the meticulous validation of results.

They further contributed to the creation of the original draft and engaged in thorough review and editing processes. Additionally, their valuable supervision ensured the project’s integrity and quality, showcasing adept project administration skills and contributing to the overall success and efficiency of the endeavor.

Ethics

No human or animal subject was involved and we do not envision any potential negative societal impacts of the research.

References

Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, *6*, 37-66. https://link.springer.com/article/10.1007/BF00153759

Akoglu, H. (2018). User's guide to correlation coefficients. *Turkish Journal of Emergency Medicine*, *18*(3), 91-93. https://doi.org/10.1007/BF00153759

Bangert, P. (2020). Predictive maintenance for gas turbines. In *International Petroleum Technology Conference*, D023S039R003. IPTC. https://doi.org/10.2523/IPTC-19864-Abstract

Bechini, G., Losi, E., Manservigi, L., Pagliarini, G., Sciavicco, G., Stan, E. I., & Venturini, M. (2022). Temporal random forest applied to gas turbine trip prediction. *In ASME*. https://doi.org/10.1115/1.4053194

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5-32. https://link.springer.com/article/10.1023/a:1010933404324

Breiman, L., Friedman, J. H., Olshen, R. A. and Stone, C. J. (1984). Classification and regression trees. *Chapman and HalI.* https://www.geeksforgeeks.org/cart-classification-and- egression-tree-in-machine-learning/

Brotherton, T., Jahns, G., Jacobs, J., & Wroblewski, D. (2000). Prognosis of faults in gas turbine engines. *In 2000 Aerospace Conference, Proceedings*, (6),163-171. https://doi.org/10.1109/AERO.2000.877892

Brotherton, T., Grabill, P., Wroblewski, D., Friend, R., Sotomayer, B., & Berry, J. (2002). A testbed for data fusion for engine diagnostics and prognostics. In *Proceedings, IEEE Aerospace Conference*, (6), 6-6. https://doi.org/10.1109/AERO.2002.1036145

Byington, C. S., Roemer, M. J., & Galie, T. (2002). Prognostic enhancements to diagnostic systems for improved condition-based maintenance. *In Proceedings, Aerospace Conference* (6), 6-6. https://doi.org/10.1109/AERO.2002.1036120

Cai, J., Luo, J., Wang, S., & Yang, S. (2018). Feature selection in machine learning: A new perspective. *Neuro Computing*, *300*, 70-79. https://doi.org/10.1016/j.neucom.2017.11.077

De Mauro, A., Greco, M., & Grimaldi, M. (2016). A formal definition of big data based on its essential features. *Library Review*, *65*(3), 122-135. https://doi.org/10.1108/LR-06-2015-0061

DePold, H. R., & Gass, F. D. (1999). The application of expert systems and neural networks to gas turbine prognostics and diagnostics. *(121*) (4), 607-612. https://doi.org/10.1115/1.2818515

De Castro-Cros, M., Velasco, M., & Angulo, C. (2021). Machine-learning-based condition assessment of gas turbines-A review. *Energies*, *14*(24), 8468. https://doi.org/10.3390/en14248468

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM, 55*(10), 78-87. https://doi.org/10.1145/2347736.2347755

El Naqa, I., & Murphy, M. J. (2015). What is machine learning*? Theory and Applications*, 3-11. https://doi.org/10.1007/978-3-319-18305-3_1

Fontes, C. H., & Pereira, O. (2016). Pattern recognition in multivariate time series: A case study applied to fault detection in a gas turbine. *Engineering Applications of Artificial Intelligence*, *49*, 10-18. https://doi.org/10.1016/j.engappai.2015.11.005

Fu, T. C. (2011). A review on time series data mining. *Engineering Applications of Artificial Intelligence*, *24*(1), 164-181. https://doi.org/10.1016/j.engappai.2010.09.007

Fu, Y. (1997). Data mining. *Potentials*, *16*(4):18-20. https://doi.org/10.1109/45.624335

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, *63*, 3-42. https://doi.org/10.1007/s10994-006-6226-1

Goyal, V., Xu, M., Kapat, J., & Vesely, L. (2020). Prediction of gas turbine performance using machine learning methods. *In Turbo Expo: Power for Land, Sea and Air*, (84157), V006T09A004. https://doi.org/10.1115/GT2020-15232

Guyon, I., Gunn, S., Nikravesh, M., & Zadeh, L. A. (2008). Feature extraction: Foundations and applications. *Springer*, *207*. ISBN-10: 3540354883.

Hess, A., Frith, P., & Suarez, E. (2006). Challenges, issues and lessons learned implementing prognostics for propulsion systems. *In Turbo Expo: Power for Land, Sea and Air*, *42371*, 927-935. https://doi.org/10.1115/GT2006-91279

Ho, T. K. (1995). Random decision forests. *In Proceedings of 3rd International Conference on Document Analysis and Recognition*, *1*, 278-282**.** https://doi.org/10.1109/ICDAR.1995.598994

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hong, C. W., & Kim, J. (2023). Exhaust temperature prediction for gas turbine performance estimation by using deep learning. *Journal of Electrical Engineering and Technology*, 1-9. https://doi.org/10.1007/s42835-023-01488-x

Köppen, M. (2000, September). The curse of dimensionality. In *5th online world conference on soft computing in industrial applications (WSC5)* (Vol. *1*, pp. 4-8).

Koutroumbas, K., & Theodoridis, S. (2008). Pattern recognition. *Academic Press*. ISBN-10: 0080949126.

Kuhn, M., & Johnson, K. (2019). Feature engineering and selection: A practical approach for predictive models. *Chapman and Hall/CRC, (1)*, 310. https://doi.org/10.1201/9781315108230

Li, S., Zhu, H., Zhu, M., Zhao, G., & Wei, X. (2021). Combustion tuning for a gas turbine power plant using data-driven and machine learning approach. *Journal of Engineering for Gas Turbines and Power*, *143*(3), 031021. https://doi.org/10.1115/1.4050020

Li, Y. G., & Nilkitsaranont, P. (2009). Gas turbine performance prognostic for condition-based maintenance. *Applied Energy*, *86*(10), 2152-2161. https://doi.org/10.1016/j.apenergy.2009.02.011

Liu, Z., & Karimi, I. A. (2020). Gas turbine performance prediction via machine learning. *Energy*, *192*, 116627. https://doi.org/10.1016/j.energy.2019.116627

Losi, E., Venturini, M., Manservigi, L., Ceschini, G. F., Bechini, G., Cota, G., & Riguzzi, F. (2021a). Data selection and feature engineering for the application of machine learning to the prediction of gas turbine trip. *In Turbo Expo: Power for Land, Sea and Air, (*85017), V008T20A004. https://doi.org/10.1115/GT2021-58914

Losi, E., Venturini, M., Manservigi, L., Ceschini, G. F., Bechini, G., Cota, G., & Riguzzi, F. (2021b). Structured methodology for clustering gas turbine transients by means of multivariate time series. *Journal of Engineering for Gas Turbines and Power*, *143*(3), 031014. https://doi.org/10.1115/1.4049503

Losi, E., Venturini, M., Manservigi, L., Ceschini, G. F., Bechini, G., Cota, G., & Riguzzi, F. (2022a). Prediction of gas turbine trip: A novel methodology based on random forest models. *Journal of Engineering for Gas Turbines and Power*, *144*(3), 031025. https://doi.org/10.1115/GT2021-58916

Losi, E., Venturini, M., Manservigi, L., Ceschini, G. F., Bechini, G., Cota, G., & Riguzzi, F. (2022b). Prediction of gas turbine trip: A novel methodology based on random forest models. *Journal of Engineering for Gas Turbines and Power*, *144*(3), 031025. https://doi.org/10.1115/1.4053194

Losi, E., Venturini, M., Manservigi, L., & Bechini, G. (2023a). Detection of the onset of trip symptoms embedded in gas turbine operating data. *Journal of Engineering for Gas Turbines and Power*, *145*(3), 031023. https://doi.org/10.1115/1.4055904

Losi, E., Venturini, M., Manservigi, L., & Bechini, G. (2023b). Ensemble learning approach to the prediction of gas turbine trip. *Journal of Engineering for Gas Turbines and Power*, *145*(2), 021009. https://doi.org/10.1115/1.4055905

Naderi, E., & Khorasani, K. (2018). Data-driven fault detection, isolation and estimation of aircraft gas turbine engine actuator and sensors. *Mechanical Systems and Signal Processing*, *100*, 415-438. https://doi.org/10.1016/j.ymssp.2017.07.021

Nohara, Y., Matsumoto, K., Soejima, H., & Nakashima, N. (2019). Explanation of machine learning models using improved shapley additive explanation. *In Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics,* 546-546. https://doi.org/10.1145/3307339.3343255

Plumb, G., Molitor, D., & Talwalkar, A. S. (2018). Model agnostic supervised local explanations. *Advances in Neural Information Processing Systems*, *31*.

Roemer, M. J., Byington, C. S., Kacprzynski, G. J., & Vachtsevanos, G. (2006). An overview of selected prognostic technologies with application to engine health management. *In Turbo Expo: Power for Land*, *Sea and Air*, 42371, 707-715. https://doi.org/10.1115/GT2006-90677

Roemer, M. J., & Kacprzynski, G. J. (2000). Advanced diagnostics and prognostics for gas turbine engine risk assessment. *In* 2000 *Aerospace Conference*, *Proceedings*, *6*, 345-353. https://doi.org/10.1109/AERO.2000.877909

Tahan, M., Tsoutsanis, E., Muhammad, M., & Karim, Z. A. (2017). Performance-based health monitoring, diagnostics and prognostics for condition-based maintenance of gas turbines: A review. *Applied Energy*, *198*, 122-144. https://doi.org/10.1016/j.apenergy.2017.04.048

Taylor, J. V., Conduit, B., Dickens, A., Hall, C., Hillel, M., & Miller, R. J. (2019). Predicting the operability of damaged compressors using machine learning. *In Turbo Expo: Power for Land*, *Sea and Air*, 58554, V02AT39A027. https://doi.org/10.1115/GT2019-91339

Wong, P. K., Yang, Z., Vong, C. M., & Zhong, J. (2014). Real time fault diagnosis for gas turbine generator systems using extreme learning machine. *Neurocomputing*, *128*, 249-257. https://doi.org/10.1016/j.neucom.2017.05.063

Yang, M. S., & Wu, K. L. (2006). Unsupervised possibilistic clustering. *Pattern Recognition*, *39*(1), 5-21. https://doi.org/10.1016/j.patcog.2005.07.005

Zhang, J., Wang, P., Yan, R., & Gao, R. X. (2018). Deep learning for improved system remaining life prediction. *Procedia Cirp*, *72*, 1033-1038. https://doi.org/10.1016/j.procir.2018.03.262

Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., & Gao, R. X. (2019a). Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, *115*, 213-237. https://doi.org/10.1016/j.ymssp.2018.05.050

Zhao, Y. P., Huang, G., Hu, Q. K., Tan, J. F., Wang, J. J., & Yang, Z. (2019b). Soft extreme learning machine for fault detection of aircraft engine. *Aerospace Science and Technology*, *91*, 70-81. https://doi.org/10.1016/j.ast.2019.05.021