Chapter 6

Explainable PHM

In chapter 5, we touched upon a topic of high significance for PHM and specifically data-driven PHM, namely uncertainty quantification (UQ). We proposed a technique for uncertainty quantification (UQ) based on Bayesian deep learning (BDL). The hyperparameters of the developed framework were tuned using a novel bi-objective Bayesian hyperparameter optimization (HPO) method with objectives the predictive performance and predictive uncertainty, to account for conflicts between these two objectives. The method was validated on the widely used C-MAPSS dataset against a single-objective baseline, that aggregates the two objectives through the harmonic mean (HM). We demonstrated the existence of trade-offs between the predictive performance and the predictive uncertainty and showed that the bi-objective HPO might be more suitable for a larger and more diverse set of hyperparameters compared to the single-objective baseline. Lastly, we saw that the proposed approach exhibits better or comparable performance to the single-objective baseline when validated on the test sets.

This chapter[1] leaves the topic of RUL prediction and follows a parallel track to discuss the importance of explainability in data-driven methods in PHM, a subject that has not yet received much attention, despite its value. Through a case study with real-world data from the aerospace industry, we will motivate the criticality of explainability, the advantages that accompany such methods, and opportunities for further research in this direction. Lastly, in this chapter, we present to the user the notion of symbolic regression (SR) and how it can assist as a tool for explainability.

[1]Contents of this chapter are based on [113]; Marios Kefalas, Juan de Santiago Rojo, Asteris Apostolidis, Dirk van den Herik, Bas van Stein, and Thomas Bäck. Explainable Artificial Intelligence for Exhaust Gas Temperature of Turbofan Engines. Journal of Aerospace Information Systems (JAIS), Volume 19, Issue 6, pages 447-454, 2022. American Institute of Aeronautics and Astronautics; reprinted with permission of the American Institute of Aeronautics and Astronautics, Inc.

6.1 Introduction

Data-driven modeling is a crucial tool in various industrial applications, including many applications in the sectors of aeronautics and commercial aviation. By data-driven modeling, we do not imply a conceptual model that is based on the data requirements of an application being developed but rather a model of the underlying data-generating process. Such predictive models perform the task of identifying complex patterns in multimodal data, something that can also be loosely termed as reverse engineering [228]. These models are in charge of providing key insights, such as which parameters (covariates) are essential on a specific measured outcome or which parameter values we should expect to observe given a set of input parameters. In addition, such models can infer future states of the system and distill new or refine existing physical models of nonlinear dynamical systems [228].

A physics-based model that adequately fits the data requires a thorough understanding of the system's physics and processes, which can be prohibitively costly in terms of time and resources. On the other hand, there are cases where such models are necessary, especially in applications where no sufficient data have been generated yet. A good example is the design and certification phase of new aeronautical systems. Linear and nonlinear statistical models rely on assumptions that might not hold (e.g., stationarity for ARMA [31] models in case of time-series). In contrast to those, non-parametric machine learning (ML) algorithms, such as the more recent deep neural networks (DNNs) [98] are considered to be "black box" models, referring to processes that lack interpretability of their internal workings and can be viewed only in terms of their inputs and outputs. This means that these models do not explain their predictions/outputs in a way that is understandable by humans, and as a result, this lack of transparency and accountability can have severe consequences [191], especially in safety-critical systems. However, model explainability is very important in a variety of engineering applications.

An *interpretable* alternative to the "black box" models and with considerably less assumptions is symbolic regression (SR). SR is a method for automatically finding a suitable algebraic expression that best describes the observed/sampled data [125]. It is different from conventional regression techniques (e.g., linear regression, polynomial regression) in that SR does not rely on a specific a-priori model structure but instead searches for the optimal model structure while simultaneously optimizing the model's parameters. The only assumption made by SR is that the response surface *can* be described algebraically [158]. SR can be achieved by various methods, such as genetic programming (GP) [125, 200], Bayesian methods [105] and physics inspired artificial intelligence (AI) [225].

Minimal human bias and low complexity of the modeling process that allows function expressiveness and insights into the underlying data-generating process is of paramount importance. For aeronautical applications, safety is of the foremost significance, and the consequences of

failure or malfunction may be loss of life or serious injury, severe environmental damage, or harm to plant or property [151]. In aviation, properly understanding the data generating process can lead to developing and improving existing physical models for nonlinear dynamical systems that could lead to new insights, as well as indicate faults and failures that can save lives and money in the context of prognostics and health management (PHM) [230].
Nowadays, with the growing generation of large amounts of data in the aviation industry (e.g., passing from snapshot to continuous data collection), many applications have been developed and improved. Some of them[234] are focused on engine health monitoring (EHM), as this is a central topic for engine manufacturers and operators. Continuous engine operating data (CEOD) are collected at high frequencies in newer aircraft types, a development which -in combination with suitable algorithms- can improve the predictive capabilities for engine operators. EHM monitors the state of individual engines or engine fleets by using historical operational data or data generated during past events to improve the availability and operability of assets. By optimizing maintenance operations, safety is improved, and asset utilization can be optimized, leading to reduced costs and improved operational efficiency. This is an area of interest not only for engine operators and maintenance providers but also for engine manufacturers. The aim of these data-driven solutions is primarily to avoid imminent failures by identifying possible anomalies in the engine operation, and secondly, to prevent over-maintenance of parts and components, exploiting their entire life span.
From an operational context, the use of models like the one presented in this chapter can assist engine users to understand in depth the evolution of the deterioration of their engines while making more reliable predictions about the time for maintenance actions and mitigating the possible disruptions in their flight and passenger operations. At the same time, maintenance providers can predict the deterioration in detail and anticipate the physical state of the engines they will inspect and repair in the near future, without first waiting for the real asset to be inducted in the shop. This way, they can streamline the maintenance process and provide more accurate quotations to their customers. Last, engine manufacturers – apart from benefiting in their maintenance business, for the reasons mentioned above – can use this type of work to understand in a better way the performance of their global fleet. This way, they can identify the influence of the different operating environments (e.g., presence of sand particles, salty water, air pollution, etc.) in the evolution of engine's health and incorporate their findings in the design of either newer versions of the same engines or even to future engine generations.
The temperature of the exhaust gases of an engine, known as Exhaust Gas Temperature (EGT), has evolved to become the standard industrial indicator of the health of an aircraft engine [236]. This is because it can capture the cumulative effect of deterioration in the isentropic efficiency

[2]Predix Platform:`https://www.ge.com/digital/iiot-platform`
[3]IntelligentEngine:`https://www.rolls-royce.com/products-and-services/civil-aerospace/intelligentengine.aspx`
[4]The MRO Lab - Prognos: `https://www.afiklmem.com/en/solutions/about-prognos`

of gas path components.
The above information motivates our main research question: Can we uncover a *meaningful* algebraic relationship between the EGT and the other measured parameters present in the CEOD data, using SR? By *meaningful* we mean that the analytic expression between the EGT and the other parameters should also be justifiable. The longtime industrial standard in engine health monitoring is the analysis of static, snapshot data. Despite being computationally lighter, this approach cannot capture the dynamics of continuous engine operation during the different flight phases. Having said that, our main contribution in this work is the first, to our knowledge, attempt to use real-world, *continuous* data collected during the entire flight duration at a recording frequency of 1Hz in order to model the EGT analytically against the rest of the monitored flight parameters. These data, termed continuous engine operational data (CEOD), allow for a more complete digital representation of the operational history of an engine.

6.2 Symbolic Regression

Symbolic regression (SR) is a methodology for finding a suitable algebraic expression that best describes the observed data [125]. In symbolic regression, no a-priori assumptions on the possible form of the expression are made, as in, for example, conventional regression models (e.g., linear regression). We could say that the latter class of models constrains the space of available expressions. The only assumption made by SR is that the relationship between the input and the output data *can* be described analytically (or in a symbolic form) [125]. In order to find the most appropriate solution, SR searches the space of mathematical expressions and estimates the corresponding parameters simultaneously [125, 105].
Performing this *data-to-function* regression [125] is a sophisticated task. Various frameworks have been developed to tackle this problem, such as genetic programming (GP) [125, 200], Bayesian methods [105] and physics inspired artificial intelligence (AI) [225]. In this work, we use GP as our framework to perform SR on our data, as with the progress in the field of GP [125], new ideas and methodologies have made GP a tool that could outperform more traditional techniques when solving modeling and identification problems, such as autoregressive moving-average (ARMA) models [240]. Furthermore, GP provides a relatively straightforward solution to the problem of SR.

6.2.1 Genetic Programming

Genetic Programming (GP), first introduced by Koza [125] in 1992, is a biologically inspired machine learning method that evolves computer programs to perform a specific task. When that task is building an empirical mathematical model then GP is called symbolic regres-

sion (SR). GP is a specialized form of genetic algorithms (GA) [70]. GA is likely the most widely known type of evolutionary algorithms (EA), which comprises a larger class of direct, probabilistic search and optimization algorithms inspired from the model of organic structure evolution [34, 91]. The idea is to evolve randomly generated initial solutions (or chromosomes, as they are more commonly referred to) on a given problem following Darwin's theory of evolution and to find the fittest solution after a number of generations or other user-specified termination criteria [70]. Solution candidates are evolved through what are called genetic operators, which include crossover or recombination and mutation, as well as selection [34, 70]. Each individual solution is evaluated using a fitness function, which essentially tailors the evolutionary algorithm to the specific problem. In essence, solutions are selected in a way that reflects their evaluation (better solutions have a higher chance of getting selected), recombined to make offspring solutions and in turn mutated, and replace the parent population for the next generation. For more information on EA, we refer the interested reader to [34].
Instead of using strings of binary digits as chromosomes to represent solutions, as in GA [70], solutions in GP are represented as tree-structured chromosomes, formed by nodes called operators and terminals. As an example, Figure 6.1 represents the simple expression:

$$(\cos(x_1) + (x_2 \cdot 0.5)) \tag{6.1}$$

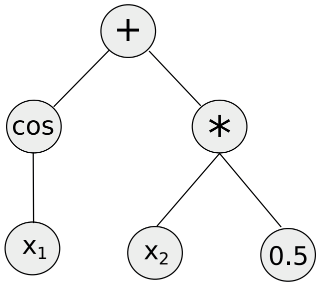

Figure 6.1: Basic GP tree representation.

Terminals are variables or values that the operator can process. These include input variables like x_i or coefficients to be used. The operators correspond to all those functions that can be applied to terminal nodes. These could be the fundamental arithmetic operators, such as $\{+, -, \cdot, /, \exp, \log, \sin, \cos, \ldots\}$, Boolean logic functions (AND, OR, NOT, etc) or any other mathematical functions. An individual (tree) is the hierarchical combination of operators and terminals, which is equivalent to an algebraic expression. When generating these tree structures, their computational complexity will depend on the method used for building them (hybrid, declarative, procedural, mathematical). A more detailed description of these tree building methodologies, as well as the algorithmic execution of a GP workflow, can be found in [203] and is illustrated in Figure 6.2.

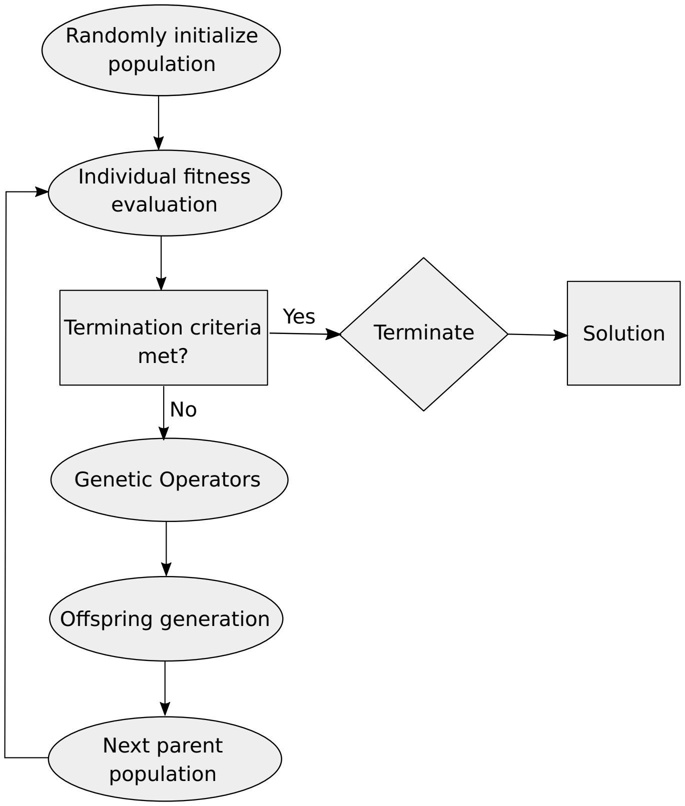
Randomly initialize population
Individual fitness evaluation
Termination criteria met?
Yes
Terminate
Solution
No
Genetic Operators
Offspring generation
Next parent population

Figure 6.2: Genetic programming algorithm flowchart.

The standard framework of GP, however, suffers from high complexity and overly complicated output expressions in SR [124]. In order to mitigate these side effects, multi-gene genetic programming (MGGP) has been developed as a robust variant of GP [76]. While the standard representation of a GP algorithm is based on the evaluation of one single tree structure, MGGP is designed to generate individual members of the GP population (mathematical models of predictor response data) that are *multi-gene* in nature, i.e., linear combinations of low-order nonlinear transformations of the input variables [202, 76]. The user can specify the maximum allowable number of genes and any gene's maximum tree depth. This facilitates a remarkable control over the maximum complexity of the evolved models [202, 76].

Mathematically, a multi-gene regression model can be expressed as:

$$\widehat{y} = d_0 + d_1 \cdot \mathrm{Tree}_1 + \ldots + d_n \cdot \mathrm{Tree}_n \tag{6.2}$$

where d_0 represents the bias term, n is the number of genes which constitutes a certain individual and $d_1, \ldots, d_n$ are the gene weights. Figure 6.3 represents an example of a multigene genetic programming model that represents the mathematical expression in Equation 6.3:

$$d_0 + d_1 \cdot (\cos(x_1) + (x_2 \cdot 0.5)) + d_2 \cdot (x_1/x_2 + 2) \tag{6.3}$$

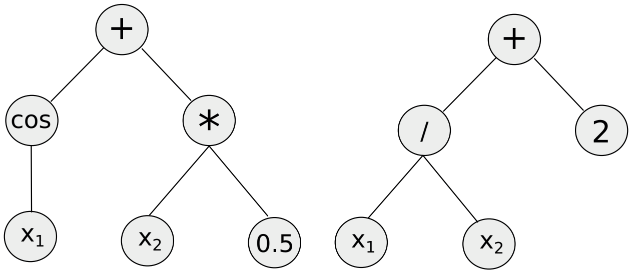

Figure 6.3: Multigene genetic programming model example.

6.3 Related Work

Although to the best of our knowledge, SR by means of GP has not been applied to the modeling of the EGT from real-life continuous flight data, there have been certain related studies. A study closely resembling our work is from Nayyeri et al. [167] who proposed an offline health monitoring system by simulating the EGT using SR by means of GP for the take-off and cruising phases of simulated data. The results returned an error of less than 0.5% and 2.5% for the take-off and cruising phases, respectively, indicating good performance. However, the material used was simulated snapshot data, and the authors did not use regularization to reduce model complexity. Martìnez-Arellano et al. [11] developed an SR approach by means of GP to predict future values of EGT, amongst other jet engine parameters, for control design. The data were collected from a small-scale jet engine that operates on the same principles as the commercial jet engines. In [144] the authors modeled the start-up process of an aero-engine by performing SR using a specialized GP that generates models that are linear combinations of nonlinear functions of the inputs and produces more parsimonious solutions. The main idea is to apply orthogonal least squares to estimate the contribution of the branches of the tree to the accuracy of the model. The models outperform the results returned from the support vector machine (SVM) algorithm and can generally identify the dynamic system characteristics correctly, even without system knowledge. GP has further been used in the field of aviation to nonlinear identification of aircraft engine [12, 64, 190].

There have also been numerous contributions of SR in engineering in general, apart from aviation. An example is [63] where the authors used SR by means of a specially designed GP to predict the fuel flow and the EGT of a gas turbine in an electrical power setting. Their approach outperformed machine-learning techniques and other symbolic regression techniques, such as fast function extraction (FFX) and multivariate adaptive regression splines (MARS), on the EGT problem. The results showed that standard GP algorithms could be used to address complex real-world problems. In [223] the authors present the first approach for the formulation of a gasoline engine performance parameters (torque and brake specific fuel consumption) using an extension of GP called gene expression programming (GEP) that evolves computer programs encoded in linear chromosomes of fixed length. Their results demonstrate that GP can be effectively used to obtain formulations for highly nonlinear function approximation problems, in general. Bongard and Lipson [30] generated symbolic equations for nonlinear coupled dynamical systems in the fields of mechanics, systems biology, and ecology. They also noted the differences between symbolic and numerical models in terms of complexity, making the former easier to interpret. In [199] the authors developed a deterministic SR method to derive algebraic Reynolds-stress models for the Reynolds-Averaged Navier-Stokes (RANS) equations for turbulence modeling.

Genetic programming has also provided solutions to various problems such as classification problems [255], telecommunications problems [65] and manufacturing process modeling [66].

The aforementioned list of applications is by no means exhaustive. It shows, nevertheless, that GP can be successfully applied to real-world industrial problems, with better, comparable, and interpretable results, compared to “black box” machine learning (ML) and artificial intelligence (AI) methods. What is more, we can see that there is also a lot of potential and growing opportunities for GP applications in the field of aviation still to come. This work stands as an example of such an application on real-life turbo-fan engine data.

6.4 Experimental Setup and Results

Our objective is to see if SR can uncover meaningful relationships in complex engineering problems. Driven by this aim, we performed the following experiments on a real aircraft operational dataset to uncover relevant dependencies between the EGT and other measured parameters of a flight.

6.4.1 Data

The data used in this study came from a specific GEnx turbofan engine mounted on a Boeing $787-10$ and were recorded during four flights in July 2019[5]. The collected data are termed continuous engine operational data (CEOD) [69] and are a data stream made out of several hundred parameters (696) which are measured during the entire flight duration at a recording frequency of 1Hz. Due to onboard computational limitations, the data have been off-loaded post-flight via gate link. The four different flights were anonymized for confidentiality and security purposes.
An important point is that the recording of CEOD is a relatively new technical development, so its use in engine health monitoring is still very limited from an operational standpoint. The longtime industrial standard is still the use of snapshot data, which are recorded only once during every flight phase. In other words, snapshot data contain only one point for takeoff and cruise and, depending on the aircraft type, for the remaining flight phases. The advantage of CEOD for diagnostics and prognostics is obvious when combined with ML algorithms since their training can be more effective.
The selected target parameter that will be modeled is termed in the CEOD dataset as *Selected Exhaust Gas Temperature (DEG_C)*. In the remainder of this study, we will call this simply EGT.

6.4.2 Experimental Setup

All experiments were executed on an off-the-shelf PC with a processor running at 1.8 GHz and 8 GB of RAM. The source code has been developed using `Python` version 3.8.3 and `MATLAB` version R2019b. We used `GPTIPS` version 2 and `Pandas` version 1.0.3.

Data Pre-processing

We decided to select the most stable phase of the flight for this study, as exhibited by the data. This phase is assumed to be the cruising phase due to the data's lack of labeled phase segmentation. Field experts further validated this assumption. This decision was made to allow for accurate modeling of the underlying process, as the distribution of EGT measurements does not exhibit extreme fluctuations, since during cruise the operational and the environmental conditions are more stable compared to other flight phases. Thus, phases such as taxiing, take-off, climb, descent, or landing were not investigated as they constitute a transient part of a flight, where engine performance and thermal effects vary with time and with mission characteristics. Furthermore, the cruising phase allowed for a larger data sample since it covers

[5]The data used is proprietary material of the Koninklijke Luchtvaart Maatschappij N.V. (KLM) and cannot be shared in the public domain.

the longest in duration part of a long-haul flight. A large sample is vital to uncover any meaningful relationships between the EGT and the rest of the monitored engine parameters.

For our experiments, three of the flights, henceforth known as *flight 1, flight 2, and flight 3* were concatenated into a single dataset. From this dataset, we discarded parameters providing little to no information. Specifically, we removed parameters containing at least one NaN (6.9% of the total parameters) value or string (alphanumeric), retaining only numeric data. Subsequently, we split the remaining data into training and validation sets by randomly selecting 80% of the data for training and the remaining 20% for validation[6]. We further pre-processed the training data by performing a correlation analysis with different conditions that result in the different experiments (see Section 6.4.2). The training data will allow the SR algorithm to *learn* patterns from the data and, as a result, estimate the model's parameters. The validation set is used to reduce any over-fitting of the SR algorithm to the training data by estimating the generalization capability of the fitted model on the validation data. This will reduce the possibility of the resulting algebraic expression reflecting *only* the training data from which it was generated. The training and validation process can be considered the training phase of the SR algorithm. A fourth flight (*flight 4*) was selected for testing purposes in order to measure the final performance of our method on unseen data (the test data). For the validation and test data we only used the parameters that were retained on the training set after all the pre-processing steps performed on it. The final pre-processing step involved normalizing each of the parameters of the training, test, and validation data as follows:

$$x' = \frac{x - \mu}{\sigma}, \tag{6.4}$$

where x, x' are the data item and the transformed data item, respectively, and μ, σ are the population (or sample) mean and standard deviation, respectively. It should be pointed out here that μ, σ which were used to normalize the validation and test parameters, are the same μ, σ learnt from the training data. The last step is standard practice in ML. Furthermore, we should note here that the selection of the flights to be used for training and validation has been done randomly, i.e., not taking into consideration flight details or characteristics, e.g. departure airport, or duration of the flight. Finally, we would like to point out that there are different potential reasons for the presence of NaN values in the dataset. In general, NaN values can be attributed to recording and synchronization issues and to the fact that not all data capturing takes place at the exact same frequency, even if the *recording* takes place at 1Hz in the CEOD. In addition, not all parameters are recorded during all the phases of a flight, so a part of the missing values could be attributed to this reason. Moreover, some secondary

[6]Please note that the terminology here is different from the original publication [113]. In this chapter and its corresponding Appendix, we note as validation set (test set) what we noted as test set (validation set) in the original publication. This has been done for consistency of the terminology between the chapters of this dissertation.

systems might not be functional for operational reasons during specific segments or the totality of the flight, and temporary recording issues cannot be excluded. For calculated parameters, some required inputs might not be available at the time that specific entries are recorded for the aforementioned reasons, resulting in NaN values. Finally, due to data ownership agreements between the original equipment manufacturer and the aircraft operator, some parameters are missing altogether.

Methodology and System Setup

We decided to use the *GPTIPS* [202, 201] to perform our experiments because of its ease of use, as well as its multigene GP approach that was discussed earlier. Additionally, *GPTIPS* takes into account the trade off surface of model performance and model complexity [202, 201]. In the multi-gene approach complexity is defined as the simple sum of the expressional complexities of its constituent trees [201]. For *each* experiment, 10 final models were independently created, *each* of which used 10 independent runs *internally.* The models resulting from the multiple, *internal* runs are automatically merged at the end of the execution and the best model is selected in terms of predictive performance (R^2 - see also Performance Metrics 6.4.2 below) among models from a Pareto front of model performance and model complexity. This internal, multi-start approach mitigates issues with the possible loss of model diversity over a single run and with the GP algorithm getting stuck in local minima [201]. The repetition (10 times) of the previously mentioned process, per experiment, is performed to have an estimate of the centrality and dispersion of the performances in each experiment. The population size was chosen to be 250 individuals, while the number of generations was at maximum 150 generations. The tournament size is set to 20, Tournament Pareto, which encourages less complex models, was set to 0.3. Elitism = 30% of the population. The maximum tree depth was set to 5, and the maximum number of genes was selected to be 10. Finally, the function set contained these operators = $\{\cdot, -, +, /, x^2, \sqrt{}, \exp, x^3, x^a, \exp(-x), -x, |x|, \log\}$. In essence, these operators define our alphabet. See Table 6.1 for a quick reference of the hyperparameters used.

Table 6.1: System setup hyperparameters.

Hyperparameter	**Value**
Runs (internal)	10
Population size	250
Number of generations	150
Tournament size	20
Tournament Pareto	0.3
Elitism	0.3
Maximum tree depth	5
Maximum number of genes	10
Function set	$\cdot, -, +, /, x^2, \sqrt{}, \exp, x^3, x^a, \exp(-x), -x, \|x\|, \log$

By default *GPTIPS* provides a multigene symbolic regression fitness function, which was used in order to minimize the root mean squared prediction error on the training data. We used $p = 0.1$ for the mutation probability and $p = 0.85$ for the crossover probability for the genetic operators. The chosen hyperparameters were based on values suggested from literature, in combination with execution time and preliminary experiments. More specifically, the mutation/crossover rates are equal to the values in [202] (default values). The same is also true for other non-mentioned hyperparameters of the algorithm. Lastly, the RMSE (see also Performance Metrics below) is used as the objective function that is minimised by the process.

Performance Metrics

To measure the performance of our approach against the ground truth, we decided to use the common error metrics for regression [205], namely, root mean squared error (RMSE), mean squared error (MSE), mean absolute error (MAE), and R^2:

$$\begin{aligned} RMSE(y,\widehat{y}) &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i-\widehat{y}_i)^2}\ , && (6.5)\\ MSE(y,\widehat{y}) &= \frac{1}{n}\sum_{i=1}^{n}(y_i-\widehat{y}_i)^2\ , && (6.6)\\ MAE(y,\widehat{y}) &= \frac{1}{n}\sum_{i=1}^{n}|y_i-\widehat{y}_i|\ , && (6.7)\\ R^2(y,\widehat{y}) &= 1-\frac{\sum_{i=1}^{n}(y_i-\widehat{y}_i)^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2}\ , && (6.8)\end{aligned}$$

where y_i are the ground truth values, $\widehat{y}_i$ the predicted values, $\bar{y}$ is the mean of the observed data, and n the number of samples.

6.4.3 Experimental Results

For the experiment, a correlation analysis was performed on the input parameters, as a dimensionality reduction step. Specifically, parameters that were highly correlated (over 0.90) were discarded, retaining only the first representative. After this step, 114/696 CEOD parameters remained to be used as final input, in the GP framework, in addition to the target EGT. We performed this experiment 10 times to account for the stochastic nature of GP by combining internally in each of these executions, 10 independent runs. This resulted in 10 different algebraic expressions. In Table 6.2 we show the average error metrics (over 10 runs) for the training, the validation, and the test datasets. The results show us that SR has managed to account for the variability of the EGT against the used CEOD parameters on both the training and validation sets ($R^2 = 1$). As a result, the average deviation from the ground truth is less than 1 degree

Celcius (MAE= 0.77°C and 0.76°C) for the training and validation sets, respectively, which is negligible from an engineering perspective. We see a larger average error regarding the test set compared to the training and validation sets. In particular, we see an average error (MAE) of 3°degrees Celcius compared to the ground truth EGT values. This slight increase in the error is also backed up by the slight decrease of the average $R^2 = 0.86$, indicating a small degree of overfitting. The slight error increase is, in general, expected. However, considering that we did not correct for parameters such as the duration of the cruising phase or the flight level, the current error is within an acceptable range.

Table 6.2: Average error metrics (over 10 runs) of the training error, the validation error and the test error, on experiment 1. All numbers have been rounded up to the nearest hundredth.

Errors	**R^2**	**RMSE**	**MAE**	**MSE**
Training Error	1 ± 0	1.3 ± 0.06	0.77 ± 0.03	1.69 ± 0.16
Validation error	1 ± 0	1.27 ± 0.05	0.76 ± 0.03	1.62 ± 0.15
Test error	0.86 ± 0.08	8.44 ± 3.27	3.01 ± 1.01	80.81 ± 46.8

In Figure 6.4 we show a plot of the EGT predictions on the validation set (displayed in orange) overlaid against the observed EGT values (displayed in blue). The x-axis represents the number of used data points. We should note here that the y axis represents the *scaled* EGT measures. The results show that the resulting algebraic expression has managed to learn the underlying relationship between the EGT and the other CEOD parameters very well.

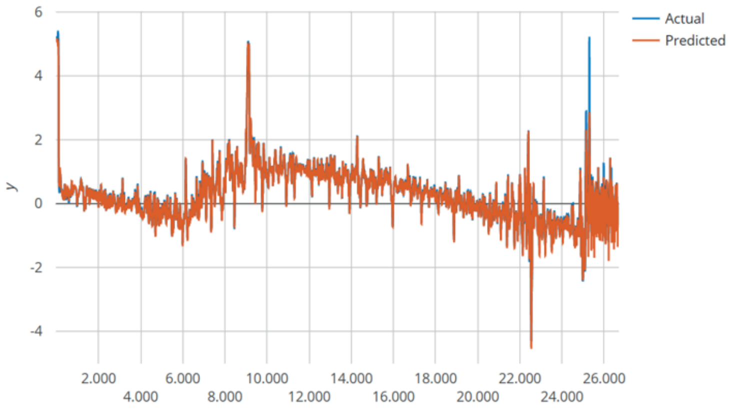

Figure 6.4: Scaled EGT predictions (red) (y-axis) on the test set vs. observed (blue) values (Model 1 - Experiment 1). x-axis shows the data sample index in consecutive order according to their sampling over time.

In addition, Equation 6.9 is the algebraic expression for the first of the 10 resulting models.

$$\begin{aligned}Y_1^1 &= 0.141 \cdot x_4 + 0.123 \cdot x_5 + 0.8 \cdot x_6 + 0.0214 \cdot x_{12} - 0.123 \cdot x_{18} \\ &+ 0.751 \cdot x_{21} + 0.0261 \cdot x_{60} + 0.0405 \cdot x_{74} - 0.0371 \cdot |\log(x_{39})| \\ &+ 1.32 \cdot 10^{-4} \cdot e^{(2 \cdot x_{21})} - 0.0428 \cdot |x_4| - 0.0261 \cdot e^{(x_{21})} \\ &+ 0.00762 \cdot x_{74}^2 + 0.133\end{aligned} \tag{6.9}$$

In Table 6.3, we show the input variables that appear in the resulting 10 models as well as their percentage of appearance. Here, each variable is represented by an x with an index. The reader

Table 6.3: Percentage of appearance per variable over all models (Experiment 1).

Input variables (Index)	**% of appearance**
x_4, x_6, x_{21}	10,6 %
x_{43}	8,5 %
x_{12}, x_{74}, x_{113}	5,1 %
x_5, x_{110}	4,08 %
x_{11}, x_{14}, x_{39}, x_{59}	3,06 %
x_{18}, x_{23}, x_{24}, x_{94}	2,04 %
x_1, x_8, x_{13}, x_{25}, x_{46}, x_{52}, x_{57}, x_{60}, x_{96}, x_{111}, x_{112}	1,02 %

might find it interesting to know which of the variables have resulted from this experiment. In the following list, we provide the technical meaning of the most frequently occurring variables based on Table 6.3. In the technical explanations, we considered only parameters with more than 5% occurrence in Table 6.3.

- **Actual Calculated HPT Clearance** – x_4
  The tip clearance of the high pressure turbine (HPT) is directly related to its isentropic efficiency and the gas enthalpy drop through the blade stages. The higher the clearance, the less efficient the expansion process is, and thus the EGT is higher.

- **Average Gas Temperature at Station 25** – x_6
  This is the gas temperature at the inlet of the high pressure compressor (HPC). A higher temperature here indicates a less efficient compression process through the engine booster, which for a given pressure ratio requires increased power input from its corresponding turbine, the low pressure turbine (LPT). This high power output can only be achieved via higher fuel flows that lead to increased EGT.

- **Corrected Fan Speed to Station 12** – x_{21}
  A higher Fan Speed also corresponds to a higher EGT, since the power required from the interconnected LPT is higher, leading to an increased fuel flow.

- **FSV Minimum Main Fuel Split Regulator** – x_{43}
  As the fuel splitting valve (FSV) influences the amount of fuel directed to the combustion chamber, there is a direct relation between this variable and the resulting EGT.

- **BPCU 1 GCU Generator Load** – x_{12}
  This parameter is related to the load control of the engine generators. The higher the load required from the generators, the higher the power extraction from the engine, which leads to higher fuel flow to cover the increased energy needs. The higher fuel flow results in a higher EGT.

- **Selected Variable Bleed Valve (VBV) Position** – x_{74}
  The position of the VBV controls the amount of air that is bled from the engine. With an increasing degree of bleed, the HPC compresses air that does not contribute to the power generated by the turbines, resulting in a reduced overall thermal efficiency. This reduction means that the engine needs to consume a higher amount of fuel for the same thrust output, which results in an increased EGT.

- **WF/(P3 · RTH25) Base (PPH/PSIA)** – x_{113}
  This is an expression for the non-dimensionalized fuel flow of the engine, which is directly related to a higher EGT.

In the list above, the station numbering (e.g., "Average Gas Temperature at Station 25") is standardized and follows the SAE Aerospace Standard AS755 (Aircraft Propulsion System Performance Station Designation)[7]. Under this standard, station 25 (see "Average Gas Temperature at Station 25") is the interface between the low pressure compressor (LPC) and the high pressure compressor (HPC), while station 12 (see "Corrected Fan Speed to Station 12") is the inlet fan tip station.
Moreover, the coefficients multiplied by the variables in Equation 6.9 indicate the relative importance (contribution) of that parameter to the output. For example, the coefficients of variables x_4, x_6, x_{21} are the largest among the coefficients of the other variables, showing their importance to the EGT. This is also backed up by the percentage of appearance of these variables throughout the repetitions and the nature of these variables, as mentioned before.
In addition, we performed two control experiments to investigate the effect that certain parameters might have on estimating the EGT. In particular, in the first, we removed the parameter *Average Temperature at Station 25 (DEG_C)*, which, even though it did not exceed the 0.9 correlation threshold, is in direct relation with the EGT. After removing it we performed the same experiment described before, which resulted in Table 6.4. The results show a similar pattern to those of our initial experiments. In addition, we see a slight decrease (by about 6%) in the

[7] `https://www.sae.org/standards/content/as755g/`

average R^2 value of the test set and a small increase (by about 8%) in the average MAE value. Despite the error increase, the deviation from the ground truth is still minimal, indicating that the EGT can be evaluated non-trivially. With this, we mean that despite dropping parameters that are closely related to the nature of our target output (e.g., *Average Temperature at Station 25 (DEG_C)*), we still get satisfying results.

Table 6.4: Average error metrics (over 10 runs) of the training error, the validation error, and the test error, on experiment 2. All numbers have been rounded up to the nearest hundredth.

Errors	**R^2**	**RMSE**	**MAE**	**MSE**
Training Error	1 ± 0	1.43 ± 0.06	0.88 ± 0.03	2.03 ± 0.17
Validation error	1 ± 0	1.4 ± 0.07	0.87 ± 0.04	1.97 ± 0.2
Test error	0.81 ± 0.04	10.27 ± 1.24	3.26 ± 0.26	106.93 ± 23.73

The second experiment involved discarding all the highly correlated (more than 90%) input parameters with the EGT before performing the correlation analysis of our initial experiment. The results of this experiment are summarized in Table 6.5. Here we see the results resembling more closely those of our initial experiment. However, it is interesting to see a decrease (by about 7%) of the average MAE of the test data, compared to the same value of the initial experiment.

Table 6.5: Average error metrics (over 10 runs) of the training error, the validation error, and the test error, on experiment 3. All numbers have been rounded up to the nearest hundredth.

Errors	**R^2**	**RMSE**	**MAE**	**MSE**
Training Error	1 ± 0	1.23 ± 0.04	0.75 ± 0.02	1.5 ± 0.1
Validation error	1 ± 0	1.24 ± 0.04	0.75 ± 0.02	1.55 ± 0.09
Test error	0.86 ± 0.08	8.44 ± 3.23	2.8 ± 0.95	80.65 ± 44.63

The resulting models and plots from all experiments can be found in the Appendix A.

6.5 Discussions and Conclusions

In this chapter, we investigated the use of symbolic regression (SR) by means of genetic programming (GP) on a real engineering problem. Specifically, we examined the use of SR on real aircraft operational data with the aim of uncovering meaningful relationships between the exhaust gas temperature (EGT) - a standard industrial indicator of the health of an aircraft engine - and the rest of the monitored engine parameters. Our main contribution is the first, to our knowledge, analytical model of EGT against the rest of the monitored flight parameters, which has been automatically derived from real-world *continuous* data collected during the entire flight time at a recording frequency of 1Hz (and been assessed by engine experts to provide valuable insights). These data, termed continuous engine operational data (CEOD), allow for a more complete digital representation of the operational history of an engine, while the longtime industrial standard is still the use of snapshot data, which are recorded only once during every flight phase.

The experimental results are promising, both in terms of model accuracy, as well as in explainability. In more detail, the trained models exhibited on average a small amount of overfitting and an absolute difference of 3°C compared to the ground truth EGT values, a small difference from an engineering perspective. Furthermore, the resulting formulas demonstrated consistency from a physics/engineering point of view between the predictor-parameters and the EGT, which field experts validated. This indicated that the proposed method could uncover meaningful relationships in the data that the end-user can interpret. In addition, we performed two more experiments to investigate the effect that specific parameters might have on the estimation of the EGT. The results showed a similar pattern to our initial experimental output.

The importance of our study lies in the fact that with little or no field knowledge, we were able to generate models that relate the EGT accurately and meaningfully to other monitored parameters. Such algebraic expressions can assist field practitioners in diagnosing faults or failures and even uncover new relationships between parameters previously unknown to engineers or field experts.

At this point, we should also mention some of the limitations of our work. Firstly, we only considered the cruising phase of the flight, ignoring the others. Having said that, we expect different behavior in phases such as take-off, where the engine performance is transient and thermally unstable. Moreover, we did not take into account or correct the data in any way, based on information such as the cruising flight-level (altitude) or the flight duration, or the aircraft's weight during cruising. For example, EGT might increase with increasing HPT tip clearance since its isentropic thermal efficiency drops. What is more, even though the data pre-processing that we did, proved to be effective, we had to discard certain data because of the NaN values. Lastly, we only modeled the EGT as a function of the rest of the observed parameters. Modeling other parameters might be more difficult or even impossible. However, as EGT is the standard industrial indicator for the overall engine thermal efficiency, this is not the main concern.

Our limitations mentioned above clearly pave the road into future directions. Initially, we would like to model transient flight phases, such as take-off, which constitutes a very intensive time for the engine. Additionally, it is worth looking into pre-processing the data with minimum loss of information (e.g., NaN value imputation) and incorporating additional information (data augmentation), such as weather conditions (e.g., when modeling parameters during climb or landing). Regarding the CEOD data specifically, we should emphasize that they can play a significant role since their higher sampling rate can capture, for example, early issues and pinpoint the exact moment they took place. However, since CEOD contains a larger amount of information than, e.g., snapshot data, the amount of data for training and testing needs to be equally high. In addition, as the engine conditions might vary significantly during take-off due to different ambient conditions, airport elevation, engine derate, etc., data representativeness is a key-point for successfully applying the methods we used and any other ML method in essence. Regarding the modeling, it would be very interesting to perform hyperparameter optimization on the GP to select the optimal hyperparameters that will allow high accuracy and low generalization error. It would also be worth building a meta-model that combines all of the formulas derived from the experiments or an ensemble model by, for example, taking the average or other aggregation function of the predictions provided by each of the models. Also, as mentioned before, such models interpretable by the end-users can lend themselves for predictive maintenance. For example, any substantial deviation between the predicted value of the model and the monitored parameter(s) can indicate a(n) (imminent) fault or malfunctioning sensor and can, thus, assist in maintenance planning. This, of course, would be possible if the model is built from *healthy* data. These formulas can also be used to generate more data, healthy or faulty, by tuning the range of the predictor parameters to simulate various conditions. Lastly, by proper data pre-processing, one can also derive formulas that allow forecasting of parameters into the future, enabling prognostics. This list is by no means exhaustive, but it is clear that there are a lot of opportunities.