

# Machine learning for predicting thermal power consumption of the Mars Express Spacecraft

Matej Petković<sup>1,2,✉</sup>, Redouane Boumghar<sup>3</sup>, Martin Breskvar<sup>1,2</sup>, Sašo Džeroski<sup>1,2</sup>,  
Dragi Kocev<sup>1,2</sup>, Jurica Levatić<sup>4</sup>, Luke Lucas<sup>5</sup>, Aljaž Osojnik<sup>1</sup>, Bernard Ženko<sup>1</sup>,  
Nikola Simidjievski<sup>1</sup>

e-mail: name.surname@ijs.si<sup>1</sup>

Jožef Stefan Institute, Slovenia<sup>1</sup>

Jožef Stefan International Postgraduate School, Slovenia<sup>2</sup>

e-mail: redouane.boumghar@esa.int<sup>3</sup>

European Space Agency, ESOC, Data Analytics Team for Operations, Germany<sup>3</sup>

e-mail: jurica.levatic@irbbarcelona.org<sup>4</sup>

Institute for Research in Biomedicine, Spain<sup>4</sup>

e-mail: luke.lucas@lsspace.com<sup>5</sup>

Mars Express, Mission Planning & Spacecraft Operations, Germany<sup>5</sup>

## Abstract

The thermal subsystem of the Mars Express (MEX) spacecraft keeps the on-board equipment within its pre-defined operating temperatures range. To plan and optimize the scientific operations of MEX, its operators need to estimate in advance, as accurately as possible, the power consumption of the thermal subsystem. The remaining power can then be allocated for scientific purposes. We present a machine learning pipeline for efficiently constructing accurate predictive models for predicting the power of the thermal subsystem on board MEX. In particular, we employ state-of-the-art feature engineering approaches for transforming raw telemetry data, in turn used for constructing accurate models with different state-of-the-art machine learning methods. We show that the proposed pipeline considerably improve our previous (competition-winning) work in terms of time efficiency and predictive performance. Moreover, while achieving superior predictive performance, the constructed models also provide important insight into the spacecraft's behavior, allowing for further analyses and optimal planning of MEX's operation.

## Index Terms

machine learning, Mars Express spacecraft, ensemble learning, predictive modeling, random forest, gradient boosting, feature engineering

## I. INTRODUCTION

**M**ARS EXPRESS (MEX), a spacecraft operated by the European Space Agency (ESA), is Europe's first spacecraft that orbits Mars. During its science operations, since the beginning of 2004, it has provided evidence of the presence of water above and below the surface of the planet [1], an ample amount of three-dimensional renders of the surface as well as the most complete map of the chemical composition of Mars's atmosphere [2].

✉ Corresponding author

Jožef Stefan Institute, Jamova Cesta 39, 1000 Ljubljana, Slovenia  
phone: +386 1 477 3635

MEX is powered by electricity generated by its solar arrays and stored in batteries to be used during the eclipse periods. The scientific payload of the MEX consists of seven instruments that provide global coverage of the planet's surface, subsurface and atmosphere. The instruments and on-board equipment have to be kept within their operating temperature ranges, spanning from room temperature for some instruments, to temperatures as low as  $-180^{\circ}\text{C}$  for others. In order to maintain these predefined operating temperatures, the spacecraft is equipped with an autonomous thermal system composed of 33 heater lines as well as coolers. The thermal system, together with the platform units, consumes a significant amount of the total generated electric power, leaving a fraction to be used for science operations.

Predicting the power consumption of the thermal system is a non-trivial task. However, due to the aging of the spacecraft and the decaying capacity of its batteries, it is a very crucial one for optimal planning and execution of science operations on MEX. The power consumption is a dynamic process that changes through time, depending on various external and internal factors, such as long-term exposure of the spacecraft to the Sun or heat generated by the on-board instruments. For instance, Figure 1 shows the effect of the radio transmitter during a communication pass, with interpolation between different temperature sensors of the  $-Y$  face of the spacecraft. Temperatures fluctuate by up to  $28^{\circ}\text{C}$  due to these two different on/off conditions. Current attempts at modeling and predicting the power consumption, involve manually constructed models that are based on simplified first-principle models, expert knowledge and experience. Given MEX's current condition, this prompts for a more accurate predictive model of the thermal power consumption, which would yield prolonged operating life.

This motivated the organization of the first ESA's data mining competition – the Mars Express Power Challenge [3]. The focus of the challenge was the development of specialized approaches for constructing models that are able to accurately estimate and predict the MEX's thermal power consumption (TPC) given only measured telemetry data. For this task of predictive modeling, machine learning approaches offer a different, yet more accurate solution to modeling the complex relationship between the telemetry and the power consumption than a human expert.

Machine learning is an area in the realm of artificial intelligence, which studies algorithms with the ability to learn, i.e., algorithms that improve their performance through knowledge gathered from experience (data). Their ability to capture and describe patterns in complex data makes them a valuable asset for studying a variety of phenomena in different domains from life

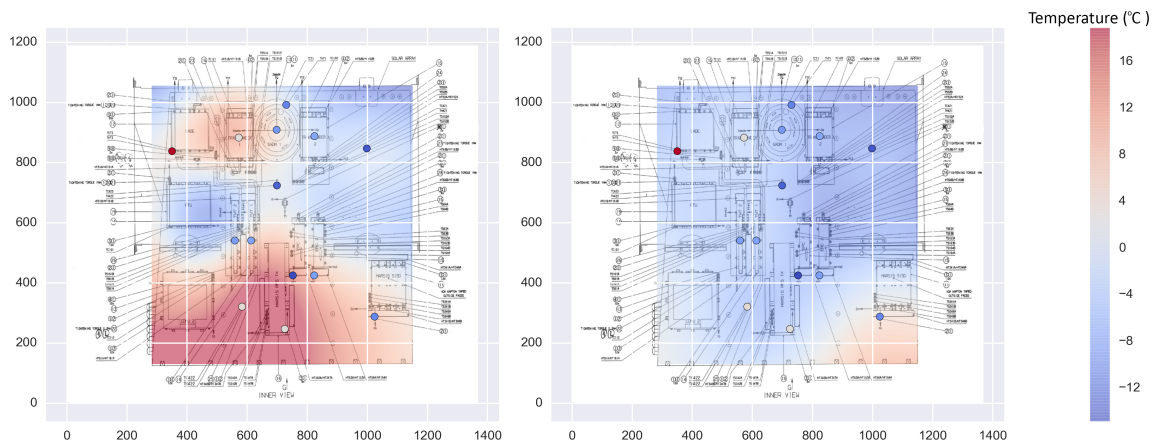


Fig. 1: Interpolated thermal effects on the MEX's  $-Y$  face, with radio transmitter turned on (left) and off (right)

sciences, earth sciences, social and behavioral sciences. In the context of the MEX challenge, machine learning algorithms for predictive modeling aim at constructing a model that can capture complex relationships in the data. In turn, such models accurately estimate future values of power consumption for each of the 33 heater lines and coolers (target variables/features) given measured telemetry data (descriptive variables/features).

In our previous work [4], we presented the machine learning pipeline solution that won the Mars Express Power Challenge. The winning solution first transforms the raw telemetry data into carefully constructed features with 1 minute time resolution between values, rendering a massive data set. Next, it uses the method of Random Forest of Predictive Clustering Trees (RF-PCTs) [5] to construct 33 predictive models for each of the 33 target variables. Finally, it outputs a predicted value of each target variable for every hour of one Martian year in the future (1.88 Earth years). The proposed solution performed better than the  $\sim 40$  other competing solutions while being more accurate than the models currently in use at ESA by an order of magnitude.

However, the premium predictive accuracy of the winning solution came at a cost of substantial computational overhead. In this paper, we extend the work presented in [4] to address this issue. In particular, we propose an update to the winning solution which aims at efficiently constructing predictive models of MEX's TPC, while still being able to maintain good predictive performance. More specifically, we consider updates of the pipeline along two dimensions: (1) constructing data with different data granularity in the learning process and (2) using different machine learning methods which can efficiently learn accurate predictive models. The former considers engineering features from the raw telemetry data at different time resolutions coarser than 1 minute thus reducing the size of the data set used in the learning phase. The latter considers both local and global methods for multi-target regression. Local methods construct a model for each target variable separately. Here, besides the winning method RFs of PCTs [5], we also consider XGBoost [6], a recent efficient implementation of Stochastic Gradient Boosting [7]. In contrast, global methods produce a predictive model able to predict several target variables simultaneously [8]. To this end, we consider global RFs of PCTs for multi-target regression, an extension of the local version, which can construct single model for all 33 target variables, therefore substantially reducing the computational time needed for obtaining a solution [5], [9].

In sum, the main task that we address is: Given three Martian years of telemetry data (August 22, 2008 to April 14, 2014), use machine learning to efficiently construct predictive model that accurately predicts the values of the electric current through the 33 thermal power consumers for the subsequent Martian year (April 14, 2014 to March 1, 2016).

The remainder of the paper is organized as follows. In Section 2, we provide an overview of the work related to machine learning applications for space-exploration research. Section 3 presents and discusses the tasks of data preparation and pre-processing. Section 4 presents the machine learning methods used in this study. In Section 5, we present the experimental setup for evaluating the proposed extensions of the machine learning pipeline. Section 6 presents and discusses the results of the empirical evaluation. Finally, Section 7 concludes the paper and suggests directions for further work.

## II. RELATED WORK

Machine learning offers an ample amount of methods that tackle predictive tasks in real life domains [10], [11]. These methods have been applied for predicting discrete output values (classification), continuous output values (regression), even structured outputs as in gene networks, image classification, text categorization etc [12].

The challenges typically addressed in space-exploration research are associated with high-cost of failure [13]. For instance, the remote spacecraft are typically equipped with processors and

memory lagging decades behind the state of the art. Next, the development and launch of a space mission is expensive and there is little or no opportunity for repair. In this context, the utility of machine learning approaches has been proven to be a valuable asset. In particular, many applications of machine learning address the task of anomaly detection in spacecraft, where a typical task considers monitoring the status of the on-board equipment. Such analyses of telemetry data are performed using neural networks [14], relevance vector machine [15] or by applying seasonal decomposition methods (linear regression together with the nearest neighbours)[16].

Machine learning can be used not only for estimating the current state of a spacecraft, but also predicting its future ones and therefore allowing for autonomous decisions. In our previous work [4], we propose a solution for predicting spacecraft's power consumption, which can be used for optimizing its operation. This work closely relates to the one of [17], where the authors propose Random Forests for predicting temperature of the instruments to optimize battery usage during eclipses.

Another important challenge relates to safe ground movement of autonomous (space) rovers or robots. The authors of [18] address this issue by using support vector machines to recognize and avoid dangerous objects. In similar context, [19] addresses the task of image analysis for automated navigation systems. Here, with deep neural Networks as the underlying machine learning method, autonomous drones are utilized for tracking forests paths.

Finally, machine learning can be also utilized to learn or simplify a physical model of a spacecraft or a model of its environment. In [20], the authors employ Random Forests to simplify the exact physical model for complex and dynamic radiation environment in the Van Allen belts. In a similar context, [13] outlines several studies which address challenges of spacecraft operating in high-radiation environments (beyond Earth's magnetosphere and ionosphere) and the reliability of machine learning algorithms applied in these scenarios. In particular, these studies propose variants of traditional ML approaches (k-means and SVMs) robust to potential data corruption on the disks due to the various levels of radiation.

### III. THE DATA

In the typical machine learning setting, the input in a learning algorithm is (training) data which embodies the experience. The data consists of training examples (also referred to as instances or measurements) and their properties (also referred to as features or attributes). The features, numerical (i.e., continuous) or nominal (i.e., discrete), can either describe the data or specify the desired output of the algorithm. In the context of predicting MEX's TPC, an example is a time period, while features are derived from context and observations data.

In this paper, we use data provided by ESA [3], that consists of raw telemetry data (context data) and measurements of the electric current of 33 thermal power lines (observation data), for three Martian years of MEX operations. We refer to these data as the training data  $\mathcal{D}_{\text{TRAIN}}$ . For the fourth Martian year of the operation, the context part of the data was used for generating the predicted values, that in turn were evaluated using the real measured observation data. We refer to these data as the test data  $\mathcal{D}_{\text{TEST}}$ .

The *observation data* consists of the electric current measurements of the 33 power consumers, recorded once or twice per minute. The *context data* consists of five components:

- **SAA** (Solar Aspect Angles) data contain the angles between the Sun–MEX line and the axes of the MEX's coordinate system.
- **DMOP** (Detailed Mission Operations Plans) data contain the information about the execution of different subsystems' commands at a specific time.
- **FTL** (Flight dynamics TimeLine events) data contain the pointing and action commands that impact the position of MEX, such as pointing the spacecraft towards Earth or Mars.

- **EVTF** (Miscellaneous Events) data contain time intervals during which MEX was in Mars's shadow or records of the time points when the MEX is in apsis of its elliptical orbit.
- **LTDATA** (Long Term Data) contain the Sun–Mars distances and the solar constant.

All raw data entries are time-stamped (expressed in milliseconds) indicating when the entry was logged. The time span between the two consecutive entries varies from less than a minute (SAA) to several hours (LTDATA). For a detailed description of the task and the data, we refer the reader to [3], [21].

The raw data is not directly applicable to a machine learning algorithm due to two main reasons: (i) incompatible time resolutions of the different components of the raw data, and (ii) unstructured format of some of the entries, such as text, that are not readily usable for machine learning algorithms. Therefore, to construct an appropriate data set for a learning algorithm, we pre-process the raw data in two phases: (i) conveying data time resolution (time interval between two consecutive examples) and (ii) engineering new (more informative) features from different parts of the context data that may yield to better predictive performance.

The first phase relates to choosing an appropriate time resolution  $\Delta t$  of the data set, and divide the time span  $[t_{\text{FIRST}}, t_{\text{LAST}})$  into subintervals  $[t_i, t_{i+1})$ , where  $t_{i+1} = t_i + \Delta t$ . Here,  $t_{\text{FIRST}}$  is the first time-stamp in the  $\mathcal{D}_{\text{TRAIN}}$ , and  $t_{\text{LAST}}$  is the last time-stamp in the  $\mathcal{D}_{\text{TEST}}$ .

The second phase considers constructing more informative features. The value of a given feature for a particular time interval is obtained by aggregating measurements from the time interval at hand that correspond to one or more components from the raw telemetry data.

Due to issues with the spacecraft communication in some periods, some measurements are missing from the both the context and observations data. In principle, the machine learning methods employed in this study can handle data with missing values. However, longer periods with contiguous missing values can substantially damage the accuracy of the learned predictive models as well as add an additional computational overhead. For this reason, we remove examples with missing observation data for time periods longer than 10 minutes. On the other hand, in the context data, we interpolate the examples with missing values for time periods shorter than 10 minutes, or leave them intact otherwise.

In the following subsections, we describe the groups of features constructed in the pre-processing step of the pipeline.

#### A. Energy influx features

There are seven features in this group: one for solar panels and one for each of the six sides of the cuboid of MEX. The features describe the amount of solar energy that is collected through a given surface in a given time interval  $[t_i, t_{i+1})$ . The solar energy collected by a side of cuboid directly influence the amount of the energy used by the thermal lines that maintain the temperature in that part of the spacecraft. The solar energy collected by the solar panels influence the amount of available energy that can be stored in spacecraft's batteries.

The amount of energy collected by a given surface is proportional to the product of the effective area  $A_{\text{eff}}$  of the surface and the solar constant  $c$ . If the area  $A$  is given, we compute  $A_{\text{eff}}$  as  $A_{\text{eff}} = A \max\{\cos \alpha, 0\}$ , where  $\alpha$  is the angle between the Sun–MEX line and the outer normal  $\vec{n}$  to the surface (see Figure 2). Without any loss of generality, we assume  $A = 1$  for all surfaces, as the machine learning methods that we use, are invariant to monotonic transformations of features. The values of  $\alpha$  for each of the seven surfaces were computed directly from the SAA data, while  $c$  was given in LTDATA. In addition to the the effective area and the solar constant,

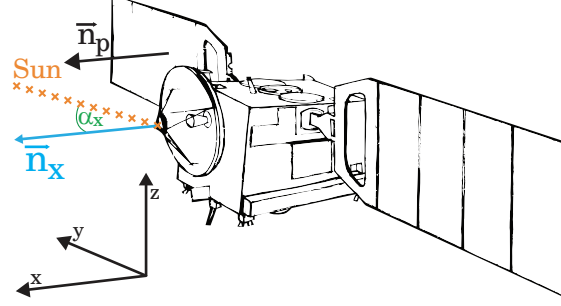


Fig. 2: Illustration of the MEX spacecraft and its coordinate axes  $x$ ,  $y$  and  $z$ , that correspond to *front*, *left* and *up* sides of MEX, respectively.  $\alpha_x$  denotes the solar aspect angle of the front side, i.e., the angle between the normal  $\vec{n}_x$  and the Sun-MEX line.  $\vec{n}_p$  denotes the normal of the panels.

(pen)umbras have a considerable impact on the energy influx. We define the amount of the energy  $E_S^i$  that pass through the surface  $S$  at time interval  $[t_i, t_{i+1})$  as

$$E_S^i = \int_{t_i}^{t_{i+1}} A_{\text{eff}}(t) c(t) U(t) dt,$$

where  $U$  is the *umbra coefficient*, an approximation of the proportion of Sun visible from the spacecraft.  $U$  takes the value  $U(t) = 0$  if the spacecraft is in an umbra,  $U(t) = 0.5$  if the spacecraft is in a penumbra, and  $U(t) = 1$  otherwise. Instead of calculating exact integrals for  $E_S^i$ , we approximate the values using the trapezoid-rule.

### B. Historical energy influx features

The thermal state of the spacecraft depends not only on the current energy influx, but also on the energy influx in the past. To capture this, we construct historical energy influx features for each of the seven surfaces. A given historical feature for surface  $S$  at time  $t_i$  is computed as a sum of energy influx during given historical time-frame:

$$H_S^i = \sum_{j=1}^H E_S^{i-(j-1)}, \quad (1)$$

where  $H$  is the number of time-intervals included in the historical feature. To account for different impacts of the historical energy influx we construct historical features with different time-frames, for different values of  $H$ , given in Sec. V-A.

### C. DMOP features

The DMOP data contain log of commands issued to different MEX sub-systems. The names of commands have been obfuscated, however the available documentation reveals two variants of events: (1) events that contain information about the subsystem and command that has been executed (e.g., ASXX383C) and (2) events that represent flight dynamics events (e.g., MAPO.000005).

The first four characters of the first variant represent the subsystem while the rest represent the command and its parameters. In the second variant the first four characters represent the name of the event, followed by a number that indicates the number of occurrences. Given that these events have different impact on the temperatures of various subsystems of the spacecraft, it is safe to assume that they impact the thermal subsystems differently.

More specifically, we assume that there is a significant delay between triggering of a sub-systems' command and its actual effect on the thermal state of the spacecraft. Therefore, from the raw DMOP data, we construct features that encode this information of delayed effect in terms of "time since last activation" of a specific sub-system command.

The values of the DMOP features are calculated as follows:

$$f_k^i = \begin{cases} 0 & \text{if } k \text{ is activated at } t_i, \\ \min(f_k^{i-1} + \Delta t, \theta) & \text{otherwise} \end{cases},$$

where  $f_k^i$  denotes the value of feature corresponding to event  $k$  at time  $t_i$ . Note that, here we also assume that all of the subsystems were deactivated at the first time point (i.e.,  $f_k^0 = \theta$ ). The  $\theta$  regulates the effect of a given event diminishing with time, rendering its influence unimportant at some point. We selected this threshold to be 1 day ( $\theta = 1440$ , the number of minutes in a day). Table I presents these calculations of features.

TABLE I: Illustration of the DMOP features that encode the time since last activation of a given subsystem command.

Raw data		DMOP features			
t	Command	APSF28A1	ASXX383C	ATTTF030A	ASXX303A
$t_1$	<i>none</i>	1440	1440	1440	1440
$t_2$	ASXX383C	1440	0	1440	1440
$t_3$	ATTTF030A	1440	$\Delta t$	0	1440
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$t_{14}$	ASXX303A	1440	$12\Delta t$	$11\Delta t$	0

We construct such features for each flying dynamic event (17 features), each subsystem-command pair (345 features) and each sub-system in case different commands are issued to it (15 features). We also construct binary indicators for each subsystem and flying dynamic event (34 features in total), where a feature  $f_k^i$  has value of 1 if the subsystem was triggered within the time-step  $t_i$ , and 0 otherwise.

#### D. FTL features

The FTL data contain logs of pointing events and their time ranges, where simultaneous events are also possible. For each pointing event in time interval  $[t_i, t_{i+1})$ , a feature has value that equals the proportion of the time in  $[t_i, t_{i+1})$ , during which the event is in progress. Since the duration of events is typically longer than  $\Delta t$ , the values of the features are mostly 1 (event is in progress), or 0 (event not in progress). This approach renders 23 FTL features in total.

#### E. The final data sets

Table II presents the important details regarding the final constructed data sets used further in the experiments\*.

### IV. MACHINE LEARNING METHODS

Considering predictive modeling, many machine learning methods struggle with the problem of overfitting. Overfitting occurs when a method learns a model with a very good performance on the

\*The data sets will be publicly available after the completion of the review process.

TABLE II: The properties of the final data sets given different granularity  $\Delta t$ .

$\Delta t$ [min]	number of examples	number of features	size [MB]
1	3922895	459	10090
5	784773	459	2048
10	392474	459	1045
15	261697	459	702
30	130900	459	353
60	65493	445	168

provided training data, but has limited generalization power and performs poorly on data unseen during learning. In machine learning, there is a long tradition of developing methods that address this problem of, by learning multiple (diverse) models and combining their outputs instead of just learning a single model. These methods are referred to as ensemble methods or ensembles. An ensemble is a set of (base) predictive models constructed with a given algorithm, that is expected to lead to predictive performance gain over an individual model by combining the predictions of its constituents. In this study, we employ two types of ensembles: (i) Random Forest of Predictive Clustering Trees (both local and global version of the algorithm for multi-target regression) [5], [22] and (ii) Stochastic Gradient Boosted Trees (XGBoost) [6], [7].

#### A. Random Forest of Predictive Clustering Trees (PCTs)

Random Forest (RF) [22] is an ensemble method that learns a set of tree-based predictive models and combines their prediction. The base models are learned from random bootstrap samples of the training set, where for each tree at each tree node a random feature subset (with a user defined size) is considered for selecting the best split. Such approach allows for constructing a set of diverse predictive models that can differ both in size and performance.

In this study, the Random Forest ensembles consist of Predictive Clustering Trees (PCTs) [23]. PCTs have a tree structure that includes internal nodes and leaves. The internal nodes contain tests on the descriptive variables (i.e., the different features extracted with pre-processing), while leaves give predictions for the target variable (i.e., power consumption of a thermal line). PCT refers to a hierarchy of clusters with each node corresponding to a cluster. In particular, the top-node of a PCT corresponds to one cluster (group) containing all data points. This cluster is then recursively partitioned into smaller clusters while moving down the tree. The leaves represent the clusters at the lowest level of the hierarchy and each leaf is labeled with its cluster's centroid/prototype (the average of the target variable is the prediction made by the leaf).

Random Forest of PCTs is a generalization of the traditional Random Forest ensemble of regression trees [24], in terms of addressing structured output prediction tasks [5], [25]. While the traditional RF is able to predict values of a single numeric target variable at a time (i.e., is a local method for Multi-Target Regression), the RF of PCTs ensemble allows also for predicting several target variables simultaneously (i.e., is a global method for Multi-Target Regression). The algorithm for learning a RF of PCTs is presented in Alg. 1. Namely, it takes three inputs: (1) training data  $\mathcal{D}_{\text{TRAIN}}$ , as well as two hyper-paramaters denoting (2) the number of trees  $M$  in the ensemble and (3) the size of the feature subset considered at each node split  $f$ . Each PCT in the ensemble is learned with greedy recursive top-down induction algorithm on a random bootstrap sample of the input data set (line 3 of the Alg. 1)<sup>†</sup>.

<sup>†</sup>The RF of PCT framework is implemented in the CLUS system available at <http://source.ijs.si/ktclus/clus-public>



The PCT-induction algorithm (lines 8-15 of the Alg. 1) starts in the root node of the tree by selecting a test from set of candidate tests that are generated by a random feature subset  $\mathcal{F}$  of size  $f$ . The best test is greedily chosen by a heuristic function that typically measures the impurity of the target values of the examples in the subsets  $E_i$  of the set  $E$  that this test results in. The goal of the heuristic function is to guide the algorithm towards small trees with good predictive performance. In the global MTR setting, this heuristic function is the average variance of the targets. In our case, with numeric features, every test is of form  $x_i < \vartheta$ , for some threshold  $\vartheta$ , and partitions the set  $E$  into two subsets. These are the set  $E_1$  of test-positive instances for which  $x_i < \vartheta$  and the set  $E_2$  of test-negative instances for which  $x_i \geq \vartheta$ , i.e.,  $\mathcal{P}^* = \{E_1, E_2\}$  (line 9).

The procedure is recursively repeated on the subsets  $E_i$  to construct the sub-trees (line 12) until a stopping criterion is satisfied (e.g., the minimal number of examples in a leaf is reached or the heuristic score no longer changes, etc.). In turn, a leaf node is created and its prototype is computed (line 15). In the global MTR setting, the prototype is a vector of average target values of the examples in the leaf. The prototypes are used for prediction.

Finally, the RF of PCTs algorithm outputs a set of PCTs, which predictions are combined (averaged per value) to obtain the final ensemble prediction. The reasons for using RF of PCTs are twofold: (i) its state-of-the-art predictive performance [5], and (ii) ability to calculate feature importance scores, i.e., ranking of the features w.r.t. their importance for the target variables.

Namely, random forests can measure how much each feature contributed to the quality of the predictive model. For this purpose, we used the Genie3 algorithm [26], for which the motivation is the following. If a relevant feature  $x_i$  is part of a test  $x_i < \vartheta$ , then the heuristic score  $h^*$  (reduction of the variance) of this split is high. Additionally, the features that appear in the tests of nodes at lower depths, e.g., in the root, influence more examples compared to the ones appearing deeper in the tree, so the former are intuitively more important. Therefore, the Genie3 importance score is defined as

$$importance_{\text{GENIE3}}(x_i) = \frac{1}{|\mathcal{RF}|} \sum_{\mathcal{T} \in \mathcal{RF}} \sum_{\mathcal{N} \in \mathcal{T}(x_i)} h^*(\mathcal{N}) |E(\mathcal{N})|, \quad (2)$$

---

**Algorithm 1** Random Forest of PCTs ( $\mathcal{D}_{\text{TRAIN}}$ ,  $M$ ,  $f$ )

---

```

1:  $\mathcal{RF} = \emptyset$ 
2: for  $m = 1, 2, \dots, M$  do
3:    $E = \text{bootstrapSample}(\mathcal{D}_{\text{TRAIN}})$ 
4:    $\mathcal{T} = \text{inducePCT}(E, f)$ 
5:   add  $\mathcal{T}$  to  $\mathcal{RF}$ 
6: return  $\mathcal{RF}$ 

7: where  $\text{inducePCT}(E, f)$ 
8:    $\mathcal{F} = \text{random sample of } f \text{ features}$ 
9:    $(t^*, \mathcal{P}^*) = \text{findBestTest}(E, \mathcal{F})$ 
10:  if  $t^* \neq \text{none}$  then
11:    for each  $E_i \in \mathcal{P}^*$  do
12:       $\text{subtree}_i = \text{inducePCT}(E_i, f)$ 
13:    return  $\mathcal{N}(t^*, \bigcup_i \{\text{subtree}_i\})$  // internal node
14:  else
15:    return  $\mathcal{L}(\text{Prototype}(E))$  // leaf node

```

---

where  $\mathcal{T}(x_i)$  is the set of nodes in the tree  $\mathcal{T}$  in which  $x_i$  is part of the test,  $h^*(\mathcal{N})$  is the value of the variance reduction function in the node  $\mathcal{N}$  and  $|E(\mathcal{N})|$  is the number of examples that come to the node  $\mathcal{N}$ .

Further in the paper, we denote the local and the global version of Random Forest PCTs for multi-target regression with *L-RF* and *G-RF*, respectively.

### B. Gradient Boosted Trees

Gradient boosting [27] refers to a class of boosting ensemble methods [10] which aims at learning a set of predictive models focusing on difficult observations in the data. In contrast to Random Forest ensembles, that first learn the ensemble constituents from random parts of the data set and in turn combine their outputs, boosting ensembles are constructed iteratively. At each boosting iteration a new weak base model that corrects the error made by the ensemble thus far is learned and added to the set creating a stronger model at each step. Typically, such weak models have simple structure and thus perform slightly better than random models.

In general, boosting methods differ in the type of base models they employ and how the learning is performed. The former is task related, where typical base models considered include: logistic regressors (for classification tasks), linear regressors (for regression tasks) or decision trees (for both classification and regression tasks). Regarding how the learning is performed, boosting base models are either constructed to focus on hard examples identified in previous iterations [28] or by minimizing the empirical risk via steepest gradient descent [27]. In this paper we focus on the latter category, referred to as Gradient Boosting.

An outline of the gradient boosting algorithm for constructing ensembles with decision trees is presented in Alg. 2. The algorithm takes three inputs: (1) a training data set  $\mathcal{D}_{\text{TRAIN}}$ , (2) number of boosting iterations  $M$ , (3) a loss function  $\mathcal{L}$  and (4) a learning rate  $\eta$ . First, an initial default (weak) model is learned on the whole data set that minimizes the loss function  $\mathcal{L}$ . Given that the gradient boosted method aims at optimizing the loss function, it is important that  $\mathcal{L}$  should be convex and differentiable. A typical choice of  $\mathcal{L}$  is  $L_2$  square-loss function [10]. In turn, gradient boosting at each step  $m$  learns a new model on the pseudo-residuals, i.e. the discrepancy value between the true and the predicted value of the ensemble in the previous iterations (line 4).

However, such straightforward approach can very easily overfit to the training data. In order to address the problem of overfitting, more sophisticated gradient boosting methods implement two different mechanisms: a learning rate and random data sampling procedure. The former regulates the influence of the prediction of each subsequent model added in the ensemble set (line 6). The latter, referred to as Stochastic Gradient Boosting [7], employs additional random data sampling procedure: Each model is learned and evaluated on different random subsamples of the training data (line 3).

In this study, we employ XGBoost [6] – a recent efficient implementation of Stochastic Gradient Boosting [7] that employs regression trees as proposed in [24] as base models. In the paper, we denote the XGBoost ensembles with *XGB*.

## V. EXPERIMENTAL SETUP

### A. Parameter Instantiation

*Granularity.* The data granularity is defined by the length  $\Delta t$  of the time interval that corresponds to one example in the data set. We construct the predictive models using data sets with  $\Delta t \in \{1, 5, 10, 15, 30, 60\}$  (measured in minutes) where  $\Delta t = 1$  corresponds to the data set used in [4].

*Historical features.* For the data set with finest granularity  $\Delta t = 1$ , we consider the following numbers  $H$  of historical intervals from (1):  $H \in \{4, 16, 32, 64, 128\}$ . Consequently, the historical

**Algorithm 2** Stochastic Gradient Boosted Trees ( $\mathcal{D}_{\text{TRAIN}}$ ,  $M$ ,  $\mathcal{L}$ ,  $\eta$ )

---

```

1:  $GBT_0 = \text{defaultModel}(\mathcal{D}_{\text{TRAIN}}, \mathcal{L})$ 
2: for  $m = 1, 2, \dots, M$  do
3:    $E = \text{randomSample}(\mathcal{D}_{\text{TRAIN}})$ 
4:    $\mathcal{R}_m = \frac{\partial \mathcal{L}(E, GBT_{m-1}(E))}{\partial GBT_{m-1}}$  // compute pseudo-residuals
5:    $\mathcal{T} = \text{induce}(\mathcal{R}_m, \mathcal{L})$ 
6:    $GBT_m = GBT_{m-1} + \eta \mathcal{T}$ 
7: return  $GBT$ 

```

---

time span ranges from 4 minutes to 128 minutes. The choices of the historical intervals in the data sets with courser granularity are presented in Table III. In total, these values yield 35 features in the data set with  $\Delta t \leq 30$  and 21 features data set with  $\Delta t = 60$ .

TABLE III: Values of the number of historic intervals  $H$  and the corresponding historic time spans, for different granularities.  $\Delta t$

$\Delta t$	values of $H$	time spans
1	{4, 16, 32, 64, 128}	{4, 16, 32, 64, 128}
5	{1, 3, 6, 13, 25}	{5, 15, 30, 65, 125}
10	{1, 2, 3, 6, 13}	{10, 20, 30, 60, 130}
15	{1, 2, 3, 4, 9}	{15, 30, 45, 60, 135}
30	{1, 2, 3, 4, 5}	{30, 60, 90, 120, 150}
60	{1, 2, 3}	{60, 120, 180}

*Random Forest parameters.* To constrain the size of the trees in the Random Forests, we specify a minimal number of examples in the leaves  $m_{\text{LEAF}}$  for each tree. Since the number of instances in the data sets is inversely proportional to  $\Delta t$ , we set the minimal number of instances for the  $\Delta t = 1$  experiments to 500, while for the others we set them to  $m_{\text{LEAF}} = 500/\Delta t$ . Additionally, we set the total number of trees in the random forests to 200, where one quarter of the features is considered at every split when growing the trees, i.e.,  $f = 0.25|\mathcal{F}|$  in Alg. 1.

*XGBoost parameters.* Analogously, to constrain the size of the trees we set maximal depth of each tree in the ensemble to 11. The learning rate parameter is set to 0.1. Additionally, to address potential over-fitting issues, for every boosting iteration 60% of the features and 60% of the examples are randomly chosen for training. The maximum number of boosting iteration (ensemble constituents) is set to 200 with an early-stop option, i.e., if the newly added trees in the ensemble do not improve the performance of the ensemble over five consecutive boosting iterations the algorithm stops.

### B. Evaluation procedure

The data set  $\mathcal{D}$  consists of examples  $(\mathbf{x}, \mathbf{y})$  where  $\mathbf{x}$  is a vector of feature values (features are described in Sec. III), and  $\mathbf{y}$  is a vector of 33 target values, i.e., the electrical currents trough the heaters and coolers.

The data set is divided into two parts:  $\mathcal{D}_{\text{TRAIN}}$  that describes the state of the spacecraft throughout the first three Martian years, and  $\mathcal{D}_{\text{TEST}}$  that describes the state of the spacecraft throughout the fourth Martian year. All predictive models, i.e., the approximations  $\hat{\mathbf{y}} : \mathbf{x} \mapsto \hat{\mathbf{y}}(\mathbf{x})$

of the true mappings  $\mathbf{y} : \mathbf{x} \mapsto \mathbf{y}(\mathbf{x})$ , were learned on  $\mathcal{D}_{\text{TRAIN}}$ . The  $i$ -th component of vectors  $\hat{\mathbf{y}}(\mathbf{x})$  and  $\mathbf{y}(\mathbf{x})$ , i.e., the predicted and true value for  $i$ -th target, are denoted by  $\hat{y}_i(\mathbf{x})$  and  $y_i(\mathbf{x})$ .

The quality of the predictions  $\hat{\mathbf{y}}(\mathbf{x})$  is evaluated on a separate test set  $\mathcal{D}_{\text{TEST}}$ , not used for learning the models. It is measured in terms of the average root mean squared error  $\overline{RMSE}$ , defined via the root mean squared errors of each target variable, as follows:

$$\overline{RMSE}(\hat{\mathbf{y}}) = \sqrt{\frac{1}{T} \sum_{i=1}^T RMSE^2(\hat{y}_i)} \quad (3)$$

$$RMSE(\hat{y}_i) = \sqrt{\frac{1}{|\mathcal{D}_{\text{TEST}}|} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_{\text{TEST}}} (y_i(\mathbf{x}) - \hat{y}_i(\mathbf{x}))^2} \quad (4)$$

where  $|\mathcal{D}_{\text{TEST}}|$  denotes the size of  $\mathcal{D}_{\text{TEST}}$  and  $T = 33$  is the number of target variables.

We also compare the machine learning methods in terms of their time efficiency (for constructing a predictive model). By doing so, we estimate the trade-off between the predictive performance of the models and time needed for constructing them, in turn determining the optimal combination of time resolution in the data and machine learning method. The time efficiency refers to single-threaded runs of the algorithms, computed as follows. First, only an  $\alpha$  portion of the ensemble is constructed where we measure the learning time  $t_\alpha$ . Subsequently, the total learning time is estimated as  $t = t_\alpha/\alpha$ . Such estimations were necessary, since some methods do not allow for single threaded runs (as reported in the next section).

## VI. RESULTS

The goal of the experiments is to determine whether we can improve the efficiency of our approach for predicting thermal power consumption, while retaining good predictive performance. In particular, we evaluate three different algorithms in terms of time efficiency (for learning a model) and predictive performance.

First, we report on the running times of the three algorithms given training data with different granularity as well as the impact on their predictive performance. Next, we discuss different alternatives for improving both the efficiency and the predictive performance of the algorithms. Finally, we discuss the quality of the learned predictive models using feature importance diagrams.

### A. Performance

We first focus on the learning times for different algorithms given different data granularity  $\Delta t$ . Figure 3 presents the time needed for each algorithm to construct a model for each target as well as the total time to complete the task.

As expected, in general, for all approaches the learning times decrease with the granularity  $\Delta t$  increasing. Nevertheless, the *L-RF* approach, that constructs an ensemble model for each of the 33 target variables, has the longest learning time of approximately 22000 hours ( $\sim 2.5$  years). In particular, learning an *L-RF* ensemble from the  $\Delta t = 1$  data set takes approximately 60-times longer than constructing an *XGB* ensemble which takes 450 hours.

While the constituents of an *XGB* ensemble are considerably smaller than the ones of an *RF*, constructing a *XGB* ensemble still takes approximately twice as much time as constructing a *G-RF* ensemble (220 hours). Note that, the constituents of both types of *RF* ensembles can be constructed in parallel, thus additionally reducing the learning time for *L-RF* and *G-RF* by a factor of 100. In contrast, the most resource friendly is the  $\Delta t = 60$  data set, where the estimated learning times of *L-RF*, *XGB* and *G-RF* are 330, 4.1 and 2.3 hours respectively.

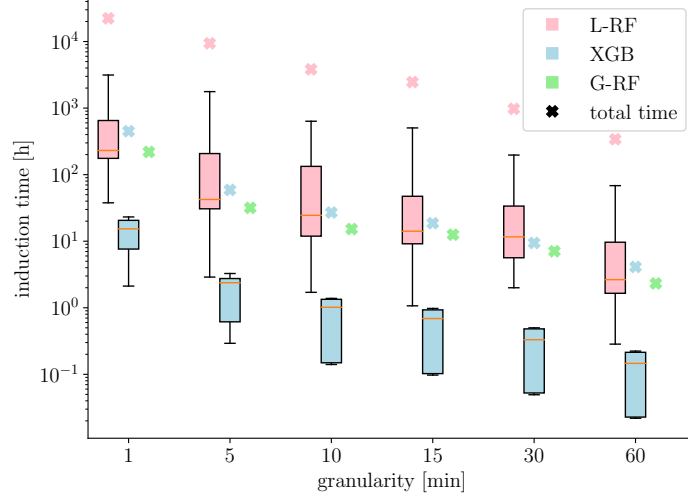


Fig. 3: Learning times for different algorithms and time granularity  $\Delta t$ . The crosses present the total learning time of the algorithm. Since *L-RF* and *XGB* build a model for each target (power line) separately, the distribution of learning times per power lines is additionally presented with the box plot. The per-target times add up to the total running time.

Next, Figure 4 presents the impact of the data granularity on the predictive performance of the models constructed with the three different methods. Note that there is a trade-off between learning-time efficiency and predictive performance. In particular, the ability of *G-RF* to construct predictive models in the shortest amount of time comes at a cost of decreased predictive performance compared to the other two methods. In this context, *XGB* performs best in two cases ( $\Delta t \in \{15, 30\}$ ) while being substantially more efficient than *L-RF* which performs best in the remaining four cases ( $\Delta t \in \{1, 5, 10, 60\}$ ).

The predictive error of both *RF* methods, increases substantially with  $\Delta t > 15$ . In the case of *XGB*, however, the best predictive performance is obtained with the medium grained data sets  $\Delta t \in \{15, 30\}$ . Note that, the *L-RF* trained on  $\Delta t = 1$  data set (as used in [4]) still has some competitive advantage in terms of predictive performance. However, similar (or slightly better) performance can be obtained either by learning from the  $\Delta t \in \{5, 10\}$  which is considerably more efficient, or by constructing *XGB* ensemble using the  $\Delta t = 15$  data set.

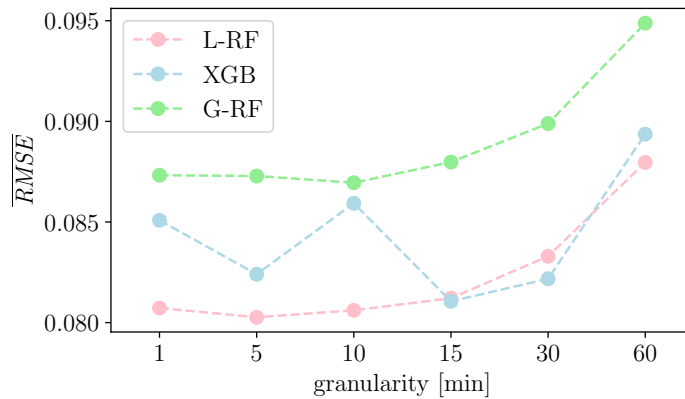


Fig. 4: Predictive performance of the three ensemble methods given data with different granularity.

### B. Further optimization

So far, we reported on ensembles consisting of 200 constituents. In general, the size of the ensemble has different effects on the quality of the predictions in the case of Random Forest ensembles and Boosting ensembles. In the former case, a general rule-of-thumb is that the predictive performance increases with the ensemble size, until it (effectively) saturates at some point. The reason for this is that every tree in RF is grown independently of the others. Thus, a RF ensemble may only be *unnecessarily* large.

In contrast, the boosting ensemble creation is different. Here, every additional tree focuses on minimizing the errors of the previously grown trees, therefore such ensembles have a tendency to overfit to the training data. As a consequence, the ensemble size can have substantial effect on the predictive performance.

We conjecture that such artefacts are also present in the models evaluated thus far, and therefore we aim to further optimize the learning methods. To estimate the optimal number of trees in the ensembles, we take the  $\Delta t = 60$  data set and perform 5-fold cross validation on  $\mathcal{D}_{\text{TRAIN}}$ . More specifically, we randomly divide  $\mathcal{D}_{\text{TRAIN}}$  into five equally sized parts  $P_i$ ,  $1 \leq i \leq 5$ . We train ensembles with different sizes (1, ..., 200) on every group of four parts, and estimate their performance on the remaining part not used for training. The average error of the five attempts (so called *cross-validated* error) is the final score of a given ensemble. We perform the same procedures for the RF (*G-RF*) and *XGB* ensembles.

The results in Figure 5 show that both types of methods can achieve good performance with considerably smaller ensembles. In particular, the RF method is able to achieve good performance very early (which does not drastically improve over time) due to accurate and deeper trees. On the other hand, as expected, *XGB* starts with very poor performance which considerably improves after cca. 50 iterations.

Given these evidence, we once again evaluate the predictive performance of the three methods on the test data, however instead of constructing ensembles with 200 trees we construct them with only 50.

Table IV confirms our conjecture: The ensemble size in the case of RF methods (*L-RF* and *G-RF*) has in general insignificant effect on the predictive performance. Note that, the Random Forest method is comprised of random trees, hence in some cases, like the *G-RF* constructed on the  $\Delta t = 15$ , small ensemble sizes hurt the predictive performance. Further analysis shows that in this case the performance stabilizes with at least 100 constituents to  $\overline{RMSE} = 0.087$ . In in the case of *XGB*, the effect of the ensemble size is more prominent and constant. More specifically,

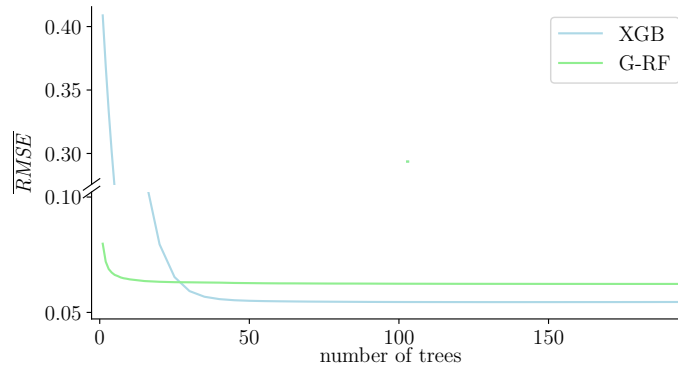


Fig. 5: The effect of the ensemble size on the performance of a Random Forest and a Gradient Boosting ensemble, evaluated with 5-fold cross validation on the  $\Delta t = 60$  data.

TABLE IV: Predictive performance ( $\overline{RMSE}$ ) of the three ensemble methods with 50 constituents.

$\Delta t$ Method	1	5	10	15	30	60
<i>L-RF</i>	0.0825	0.0804	0.0812	0.0808	0.0844	0.0876
<i>G-RF</i>	0.0865	0.0869	0.0881	0.0989	0.0899	0.0954
<i>XGB</i>	0.0835	0.0796	0.0835	<b>0.0785</b>	0.0816	0.0889

the predictive performance of an *XGB* ensemble learned on a data set with granularity  $\Delta t = 15$  yields the best overall performance we obtained so far. Note that obtaining such performance is also considerably faster than the one reported in our previous study, i.e., *L-RF* learned on  $\Delta t = 1$ .

Nevertheless, in terms of time efficiency, all ensemble methods benefit from the reduction of the ensemble size. In particular, on average the learning time is reduced by factor of 4 in the case of RF ensembles and by a factor of almost 5 in the case of boosting. Figure 6 presents the results of the overall learning time and the predictive performance of all three methods with reduced ensemble size.

More specifically, here we aim at identifying the optimal choice of algorithm given both criteria of predictive performance and time efficiency. Therefore, the optimal solution should be as close as possible to the lower left corner of the graph. A point *A* in the graph is *dominated* by another point *B* in the graph if *B* is better than *A* in both criteria. For example, the performance of *L-RF* on  $\Delta t = 10$  is dominated by the performance of *XGB* with both  $\Delta t \in \{5, 15\}$ . The non-dominated points form a so called *Pareto front*. In our case, the Pareto front consists of the two *XGB* points ( $\Delta t \in \{15, 30\}$ ) and four *G-RF* points ( $\Delta t \in \{5, 10, 30, 60\}$ ). All these points are considered optimal, unless we further specify our preferences over the criteria: If one aims at obtaining the fastest (but less accurate) solution one should consider *G-RF* ( $\Delta t = 60$ ). On the other hand, *XGB* ( $\Delta t = 15$ ) yields the most accurate (but less efficient) solution.

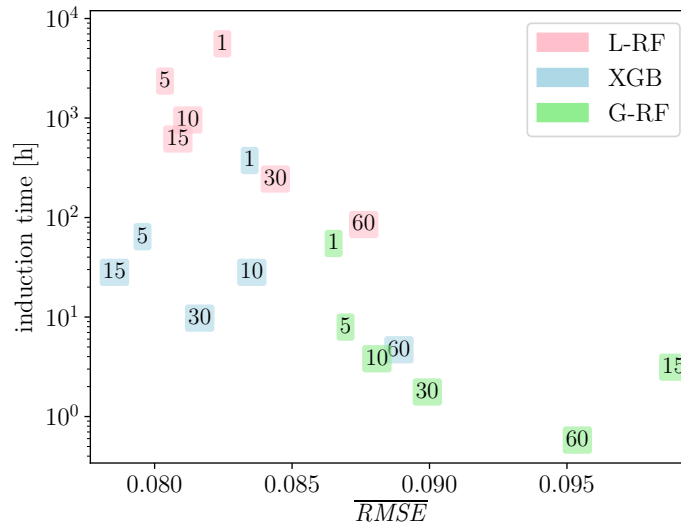


Fig. 6: The trade-off between the learning time and the predictive performance ( $\overline{RMSE}$ ) of the three ensembles with 50 : Colors of the rectangles denote the induction algorithm, while the numbers in the rectangles determine the time granularity  $\Delta t$ .

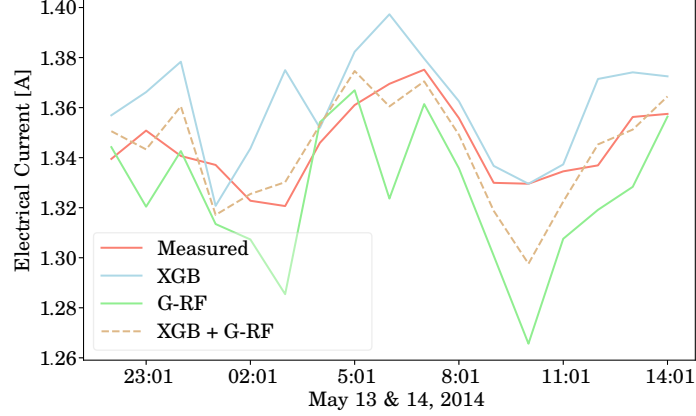


Fig. 7: Measured (red line) and predicted behavior on a sample time-period of 2 days for the 12th power line, obtained from *G-RF* and *XGB* ensembles as well as their combination.

### C. Ensemble of Ensembles

The performance of an ensemble is a consequence of the performance and diversity of its constituents. Moreover, ensembles usually perform better when compared to each individual constituent [29]. Given that in this study we consider different types of ensembles (*RFs* and *XGB*) with good predictive performance, to further improve the overall performance we can also combine their outputs in an Ensemble of Ensembles (EoE). As a proof-of-principle, Figure 7 presents the predictions of *XGB* and *G-RF* during sample time-period of 2 days for the 12th power line. We can see that, while the *XGB* overestimates and *G-RF* underestimates the measured data (red line), their combination performs better.

Given the results from Figure 6 we construct two EoEs. The member ensembles are trained on  $\Delta t = 15$  data set, since all results so far point to a good trade-off between efficiency and performance in this case. Moreover, both EoEs have one *XGB* member combined either with *G-RF* (both methods are efficient) or *L-RF* (both methods are accurate).

The results are summed up in Table V. While the *XGB&G-RF* ensemble is very efficient to construct, its performance is only better than one of the members, *G-RF*. On the other hand, the *XGB&L-RF* achieves the best performance overall of  $\overline{RMSE} = 0.07779$ . However, in practice, obtaining such an ensemble takes 10 times more than learning only a *XGB* ensemble that has practically similar performance ( $\overline{RMSE} = 0.07853$ ).

TABLE V: Comparison of the two ensembles of ensembles (EoE) in terms of predictive performance ( $\overline{RMSE}$ ) to the performance of the individual ensembles. All models are learned on the  $\Delta t = 15$  data set.

Ensembles	$\overline{RMSE}$
<i>XGB&amp;G-RF</i>	0.0803
<i>XGB&amp;L-RF</i>	<b>0.0778</b>
<i>G-RF</i> (100)	0.087
<i>L-RF</i> (50)	0.0808
<i>XGB</i> (50)	0.0785



#### D. Feature Importance

In our last set of experiments, we assess the importance of the features obtained from the three methods. Typically, different features have different influence on the target variables, which in turn affects the predictive performance of the constructed models. Figure 8, illustrates how different groups of features (energy influx, DMOP and FTL) influence the 33 power consumers. The feature importance diagrams were calculated using the  $\Delta t = 15$  data set. More specifically, in the case of *L-RF* and *XGB* (Figs. 8b and 8c), the proportions in the diagrams for each target were computed according to Eq. (2). On the other hand, in the case of *G-RF* (Figure 8a) where the model predicts all targets simultaneously, Eq. 2 leads only to the global feature importance (*all* diagram). The per-target importance were computed from the local models, considering one target when computing the heuristic function. Note that, for some targets, the feature importance diagrams are blank since all ensemble constitutes in these cases are constant models (trees without internal nodes).

In the case of *G-RF*, the most important feature group overall is DMOP. In particular, in the majority of the power lines (27) their influence is at least 50%. The Energy influx features have a major influence on five power lines, while the FTL features have a considerable impact only on the 13th power line. Similarly, *L-RF* finds the DMOP features as most important overall. However, as opposed to *G-RF*, here we can see find more power lines which are almost exclusively influenced by energy influx features. These differences in the computed feature importance between *G-RF* and *L-RF* is a consequence of how the ensembles are constructed: While the former is able to capture the global phenomena across all power lines, the latter captures more detailed behaviour that relates to each individual power line.

On the other hand, *XGB* mostly relates to the DMOP features. Compared to the other two methods, in this case their influence is considerably greater. In particular, the Energy Influx features have some significant importance ( $> 20\%$ ) on only 4 power lines, while the FTL features do not contribute greatly. Additionally, *XGB* was not able to produce feature importance diagrams for 5 of the target variables.

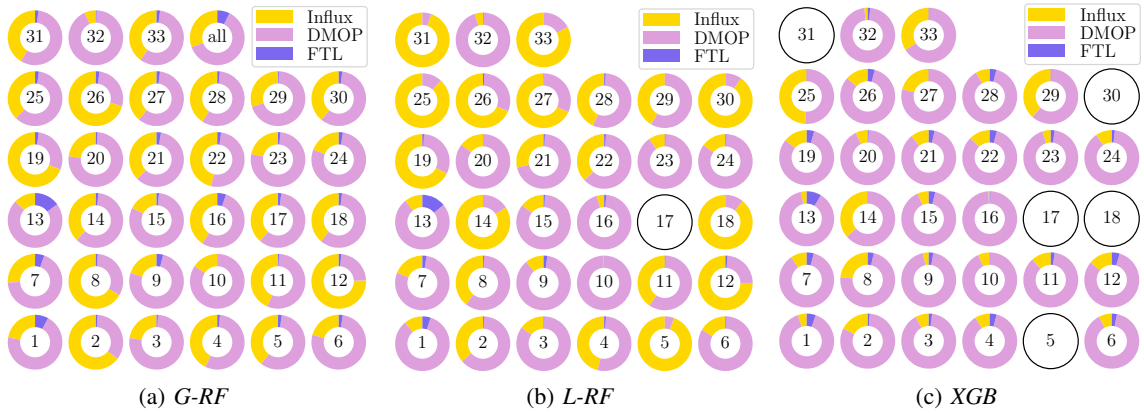


Fig. 8: Distribution of different feature groups in the feature rankings produced by (a) *G-RF*, (b) *L-RF* and (c) *XGB* ensembles, for the 33 power lines. In each of the individual diagrams, the presence of a given feature group type is proportional to the sum of Genie3 relevance over the features from the group. The empty diagrams denote ensembles constructed from constant models. The *G-RF* method allows for the additional *global* feature importance denoted as *all*.

## VII. CONCLUSION

In this paper, we propose a machine learning pipeline for predicting the power of the thermal subsystem on board the Mars Express spacecraft. More specifically, we propose novel solutions in the machine learning pipeline that focus on efficiently constructing predictive models of MEX's TPC, while still being able to maintain high predictive performance. More specifically, we employ state-of-the-art feature engineering approaches for transforming raw telemetry data, which in turn is used for constructing accurate predictive models with different machine learning methods. These solutions are the main contribution of our paper, since they considerably improve our competition-winning solution [4] in two directions: efficiency and accuracy.

The proposed improvements in the pipeline consider (1) preparing training data with different time granularity, as well as (2) employing different machine learning methods for constructing accurate predictive models. Regarding the former, we carefully transformed the raw telemetry data at different time resolutions ( $\Delta t \in \{1, 5, 10, 15, 30, 60\}$  min) which resulted in significant reductions of the size of the data sets used in the learning phase. Regarding the latter, we considered different state-of-the-art local and global machine learning ensemble methods for multi-target regression. These methods include: Local and Global Random Forests of Predictive Clustering Trees (*L-RF* and *G-RF*) as well as Stochastic Gradient Boosted Trees (*XGB*). We evaluated our proposed solutions on the task of predicting hourly values of the electric current through the 33 thermal power consumers on board MEX for one Martian year, given raw telemetry data of three preceding Martian years.

In terms of time efficiency, our empirical study shows that the time resolution of the data has a significant impact on both the construction time of the predictive models as well as on their accuracy. The former is an expected result, given that coarser granularity yields a reduced data set and therefore shorter learning time. However, learning methods using coarser data usually yield less accurate models. The latter result though, provides a significant insight into this problem: Given a data set with moderate granularity ( $\Delta t \in \{10, 15\}$ ), all three methods are able to obtain models with comparable (or better) predictive performance, in substantially shorter time as compared to models learned on data set with finer granularity.

In terms of predictive performance, the local ensemble methods perform better than the global method. More specifically, in most cases, *L-RF* and *XGB* have comparable performance, with *XGB* being slightly better. While both methods perform better than *G-RF*, the difference in performance is neither substantial nor significant. Note that learning a global model also takes considerably less time than learning a local model. In the same context, our results show that, while the size of the ensembles has a significant effect on the learning time, it can also improve the predictive performance. In particular, we showed that, with all methods we can obtain similar or better (*XGB*) predictive performance with smaller ensembles. Moreover, we also demonstrated that by further combining the predictions of the different ensembles into an ensemble-of-ensembles, we can additionally improve the predictive performance and obtain premium accuracy ( $\overline{RMSE}_{XGB\&L-RF} = 0.07779$ ).

Finally, our feature importance analysis indicates that, for this particular problem of predicting thermal power consumption, the Detailed Mission Operations Plans (DMOP) have a significant role in the quality of the predictive models. Their importance is more prominent in the *XGB* ensembles models than in the *RF* ensembles, which also rely on the Energy Influx features when constructing a model.

There are several directions to extend the work presented in this paper. Considering the data, note that DMOP information is available only after a certain command is executed on the spacecraft and its effect measured subsequently. This means that using these data for predicting longer time horizons is not possible. Moreover, given the findings of our paper, omitting them

from the learning process might have a severe consequence on the performance of the predictive models. Therefore, an immediate continuation of the work presented here is to further optimize the constructed features as well as investigate different approaches for engineering new (informative) features. Finally, while the proposed methodology focuses on the thermal subsystem of the MEX spacecraft, it can also be readily applied to the other subsystems. Moreover, it can also be extended to other spacecraft such as the XMM Newton [17], Integral [20] and ExoMars as well as rovers (such as Curiosity and ExoMars) exploring Mars.

#### ACKNOWLEDGMENT

We thank the Slovenian grid initiative & the Academic and Research Network of Slovenia for support with the computer infrastructure. We acknowledge the financial support of the Slovenian Research Agency (via the grants P2-0103, J4-7362, L2-7509 and the young researcher grants to MP and MB), as well as the European Commission via the grants HBP (The Human Brain Project) SGA1 and SGA2.

#### REFERENCES

- [1] R. Orosei, S. E. Lauro, E. Pettinelli, A. Cicchetti, M. Coradini, B. Cosciotti, F. Di Paolo, E. Flamini, E. Mattei, M. Pajola, F. Soldovieri, M. Cartacci, F. Cassenti, A. Frigeri, S. Giuppi, R. Martufi, A. Masdea, G. Mitri, C. Nenna, R. Noschese, M. Restano, and R. Seu, "Radar evidence of subglacial liquid water on mars," *Science*, 2018.
- [2] A. Chicarro, P. Martin, and R. Trautner, "The Mars Express mission: An overview," in *Mars Express: The Scientific Payload*, vol. 1240, 2004, pp. 3–13.
- [3] L. Lucas and R. Boumghar, "Machine learning for spacecraft operations support - The Mars Express Power Challenge," in *Sixth International Conference on Space Mission Challenges for Information Technology, SMC-IT 2017*, 2017.
- [4] M. Breskvar, D. Kocev, J. Levatić, A. Osojnik, M. Petković, N. Simidjievski, Ženko, R. Bernard Boumghar, and L. Lucas, "Predicting thermal power consumption of the mars express satellite with machine learning," in *2017 6th International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, 2017, pp. 88–93.
- [5] D. Kocev, C. Vens, J. Struyf, and S. Džeroski, "Tree ensembles for predicting structured outputs," *Pattern Recognition*, vol. 46, no. 3, pp. 817–833, 2013.
- [6] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16, 2016, pp. 785–794.
- [7] J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367 – 378, 2002, nonlinear Methods and Data Mining.
- [8] H. Borchani, G. Varando, C. Bielza, and P. Larrañaga, "A survey on multi-output regression," *Data Mining and Knowledge Discovery*, vol. 5, no. 5, pp. 216–233, 2015.
- [9] E. Spyromitros-Xioufis, G. Tsoumakas, W. Groves, and I. Vlahavas, "Multi-target regression via input space expansion: treating targets as inputs," *Machine Learning*, vol. 104, no. 1, pp. 55–98, 2016.
- [10] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, ser. Springer Series in Statistics. Springer New York, 2013.
- [11] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.
- [12] G. H. Bakır, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. V. N. Vishwanathan, *Predicting structured data*, ser. Neural Information Processing. The MIT Press, 2007.
- [13] A. McGovern and K. L. Wagstaff, "Machine learning in space: extending our reach," *Machine Learning*, vol. 84, no. 3, pp. 335–340, 2011.
- [14] Z. Li, "Machine learning in spacecraft ground systems," in *2017 6th International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, 2017, pp. 76–81.
- [15] T. Yairi, Y. Kawahara, R. Fujimaki, Y. Sato, and K. Machida, "Telemetry-mining: A machine learning approach to anomaly detection and fault diagnosis for space systems," pp. 476–483, 07 2006.
- [16] M. Muñoz, Y. Yue, and R. Weber, "Telemetry anomaly detection system using machine learning to streamline mission operations," in *2017 6th International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, 2017, pp. 70–75.
- [17] G. De Canio, T. Godard, R. Boumghar, and U. Weissmann, "Optimization of the battery usage during eclipses using a machine learning approach," in *15th International Conference on Space Operations*, Marseille, France.

- [18] A. C. Hernández, C. Gómez, J. Crespo, and R. Barber, "Adding uncertainty to an object detection system for mobile robots," in *2017 6th International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, 2017, pp. 7–12.
- [19] A. Giusti, J. Guzzi, D. C. Cireşan, F.-L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, D. Di Caro, Gianni Scaramuzza, and L. M. Gambardella, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, 2016.
- [20] T. J. Finn, R. Boumghar, J. Martinez, and A. Georgiadou, "Machine learning modeling methods for radiation belts profile predictions," in *15th International Conference on Space Operations*, Marseille, France.
- [21] R. Boumghar, L. Lucas, and A. Donati, "Machine learning in operations for the mars express orbiter," in *15th International Conference on Space Operations*, Marseille, France.
- [22] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [23] H. Blockeel, L. De Raedt, and J. Ramon, "Top-down induction of clustering trees," in *The 15th International Conference on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 1998, pp. 55–63.
- [24] L. Breiman, J. Friedman, R. Olshen, and C. J. Stone, *Classification and Regression Trees*. Chapman & Hall/CRC, 1984.
- [25] H. Blockeel, "Top-down induction of first order logical decision trees," Ph.D. dissertation, Katholieke Universiteit Leuven, Leuven, Belgium, 1998.
- [26] V. A. Huynh-Thu, A. Irrthum, L. Wehenkel, and P. Geurts, "Inferring regulatory networks from expression data using tree-based methods," *PLOS ONE*, vol. 5, no. 9, pp. 1–10, 09 2010.
- [27] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.
- [28] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119 – 139, 1997.
- [29] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 993–1001, 1990.