17th International Conference on Space Operations, Dubai, United Arab Emirates, 6 - 10 March 2023.

SpaceOps–23, ID # 331

Utilizing Machine Learning methods for classifying Telemetry of Human Spaceflight Systems

Carsten Hartmann[a]*, Franca Speth[b], Udo Kebschull[c], Dieter Sabath[a], Florian Sellmaier[a]

[a] *DLR e.V., Münchener Str. 20, 82234 Weßling, Germany*
[b] *SVA System Vertrieb Alexander GmbH, Borsigstraße 26, 65205 Wiesbaden, Germany, Franca.Speth@sva.de*
[c] *Goethe-Universität, 60323 Frankfurt am Main, Germany*
* Corresponding Author

Abstract

All past and present crewed outposts in space, most prominently the International Space Station (ISS), rely on one particular mode of operation: real-time operation ("ops" for short). Here, the vehicle, and it's subsystems, are almost entirely monitored and controlled from ground. The astronauts on-board instead focus their time and effort on achieving the scientific mission objectives. However, within the next decade, almost all international space agencies have defined goals to achieve human presence around and on the Moon, as well as prepare for deep space missions beyond the Moon, and towards Mars. For such missions, it is obvious that, with increasing distance from earth, the signal delay between a vehicle and a ground station also increases. As a consequence, beyond a certain distance, a ground controller is no longer capable to monitor and control the vehicle in real time. Therefore, a new operational concept is needed, shifting the responsibility of evaluating and classifying real time data of the on-board systems. Shifting the responsibility to crew is not feasible, since available crew size and crew time are limited resources on-board a crewed spacecraft. So, in order to enable the crew to still achieve the mission objectives, without significantly increasing their duties or size, a shift is needed to an intelligent on-board system instead. This paper will examine how this can be achieved by analyzing the data that is generated on-board the Columbus Module of the ISS, as well as introducing different Machine Learning (ML) methods used for classification. For that purpose, we first highlight how time series data is generated on-board, how it's arriving at the ground controllers' console, as well as what properties the data has. Afterwards, we focus on the different ML methods capable of dealing with the given data. To validate and compare the different algorithms and preparation methods we classified nominal and unexpected operating states of the Columbus module using archive data from the Columbus Training, Qualification and Verification System (TQVS). Reports from an anomaly database were used to map unexpected states and a mission timeline was used to map nominal states. Our research demonstrates the benefits and drawbacks of different supervised machine learning algorithms in the context of telemetry classification and system monitoring in Human Spaceflight. On top of that, this work offers valuable insights for further developments on smart systems in the context of human deep space exploration.

Keywords: Human Spaceflight Operations, Data Processing, Telemetry Monitoring, Classification Algorithms, Operation Concepts,

Acronyms/Abbreviations

AI Artificial Intelligence
CMU Command and Measurement Unit
CNES Centre National d'Etudes Spatiales
COL-CC Columbus Control Center
COMMS Communications System
CRISP-DM Cross Industry Standard Process for Data Mining
DaSS Data Service Subsystem

DMS Data Management System
ECLSS Environmental Control and Life Support System
EPDS Electrical Power Distribution System
ESA European Space Agency
ETCS External Thermal Control System
FDIR Fault Detection, Isolation, and Recovery
GER Global Exploration Roadmap
GSOC German Space Operations Center
HOSC Huntsville Operations Support Center
IGS Interconnection Ground Segment
ISECG International Space Exploration Coordination Group
ISS International Space Station
JSC Johnson Space Center
KOMPSAT-2 Korea Multi-Purpose Satellite 2
LEO Low-Earth-Orbit
LSTM Long-Short Term Memory
MAE Mean Absolute Error
MCC Mission Control Center
MCC-H Mission Control Center Houston
MCS Monitoring and Control System
METIS Mars Exploration Telemetry-driven Information System
ML Machine Learning
MSFC Marshall Space Flight Center
NASA National Aeronautics and Space Administration
OOL Out-Of-Limit
PDU Power Distribution Unit
REST Representational State Transfer
RMSE Root Mean Square Error
SAN Storage Area Network
TCS Thermal Control System
TDRS Tracking and Data Relay Satellite
TQVS Training, Qualification and Verification System

1. Introduction

Today, crewed spaceflight is mostly limited to space stations in Low-Earth-Orbit (LEO). Examples are of course the ISS, but also other stations like the Chinese' Tiangong Station, or stations which were decommissioned by now, like the MIR. What all those space stations in LEO have in common is, that a multitude of sensors exists on-board generating telemetry in real-time for the complete lifetime of the respective station. This telemetry is then downlinked to a ground station, and used by a ground controller to evaluate and control the state of the station, and its subsystems, down to each single sensor. This concept of operation is called "real-time operation", and it is proven to work well for crewed vehicles. Among its many advantages are: 1. direct feedback and support during science runs, leading to increases in crew efficiency, 2. direct response to off-nominal situations by ground, alleviating crew responsibilities and helping to ensure adequate crew health and safety at all times, as well as 3. having uninterrupted ground support helps ease training and subject matter expertise requirements for the crew. On the other hand, real-time ops has a few significant drawbacks. Among them are: 1. ground stations and control centers need to be available and operated 24/7, putting significant strain on the ground personnel, 2. continuous space-to-ground communications need to be maintained via tracking and relay satellites and antenna dishes on Earth, increasing overall system complexity and susceptibility to errors, as well as 3. building and maintaining

real-time capable ground infrastructure and training and employing ground personnel adds a substantial amount of cost to the overall mission. For currently ongoing missions, like the ISS, those are not serious concerns at the moment, however, in light of the latest objectives set forth by National Aeronautics and Space Administration (NASA) and European Space Agency (ESA), those issues could become critical show-stoppers for future missions, if not addressed adequately. Thus, both agencies are working on ways to improve and ease operations, specifically for missions, where real-time operation is either no longer possible, due to signal delays between ground and station, or where real-time ops is no longer feasible, due to overall mission complexity and cost. As proposed by Söllner et al., this paper represents the next step in developing a new operational concept, by shifting the task of telemetry monitoring from the ground controller to an automated tool. This work aims to contribute to the development of an automated tool utilizing ML. For this purpose, the paper is organized as follows: hereafter, a short background and introduction to ISS operations are given. Section 2 contains a brief overview of the related work including latest research in the direction of telemetry processing and ML utilization for space missions and particularly mission concepts of human spacecraft operations. Section 3 deals with a description of the experimental setup and methods, followed by section 4 containing the results and comparison of ML methods. Finally, the key takeaways are summarized and outlooks for future research are given in chapter 5.

1.1 Background

In addition to the three advantages of real-time ops mentioned above, two main contributions can be identified, why real-time ops is still in use today for crewed spaceflight missions. First, historically, this mode of operation has always been the safest way to operate a crewed spacecraft. This was proven time and time again, since the very first crewed space missions, Mercury and Apollo [2]. Since then the concept has persisted due to its high reliability and the close cooperation between the crew and ground personnel, especially in off-nominal situations. As a short side note, we do acknowledge and recognize that this is not applicable for un-crewed spacecraft (i.e. satellites). Here, a high level of autonomy and time-tagged commands are state of the art. However, since un-crewed vehicles and space-to-ground commanding are not subjects of this paper, we will only discuss the topic of on-board monitoring for satellites in 2. Second, due to the high cost of launching and hosting humans in space and the resulting limited availability of crew time, the second goal, after safety, is the goal of maximizing crew utilization. During a mission, the time needed for vehicle and system maintenance, the time for housekeeping and the time for personal well-being of the crew have to be weighted against the time available for scientific experiments. It follows logically, that if ground personnel can perform certain tasks (mostly vehicle and system maintenance) from ground, crew time is freed up for other more important duties.

While the provided benefits of real-time ops, as mentioned above, have a tremendous positive impact on crew and crew performance, there are certain drawbacks that come with this mode of operation. A few of them were already mentioned, however, they are revisited here to provide additional context. The first and most important drawback is cost. There are several key factors contributing to the annual cost of operations alone: the need to have ground personnel available 24 hours a day, 7 days a week, and 365 days a year, the cost for the ground infrastructure connecting the different control centers with each other and the infrastructure to connect the control centers to the ISS, the additional cost to provide secondary engineering and manufacturing support, to mention just a few. Thus, long-duration crewed space missions usually are only reserved for entities with enough budget to provide operational support over the whole lifetime of a mission. While the current plan foresees the decommissioning of the ISS by 2030, the problem of cost holds true for all current and future crewed missions utilizing the real-time ops concept (i.e. all of them at the moment). Thus, looking into the future, any new crewed outposts in space currently proposed or already under development, have to take this factor into account during design and development. Examples of such outposts are the Lunar Gateway, as part of NASAs Artemis Program [3] and the SciHab, as part of ESAs Terrae Novae 2030+ Strategy Roadmap [4]. The Gateway will orbit the Moon and have un-crewed periods. It is supposed to be an intermediate habitation and support station to ease transportation to and from the Lunar surface. SciHab on the other hand is planned as a commercial station, providing a platform for science and habitation in LEO. In addition to their proposed stations around Earth and the Moon, NASA and ESA both have ambitions to send humans to Mars in the 2030-2040 timeframe, as outlined in their respective strategy documents [3, 4]. While the specific vehicles for such missions are either still in the concept or (at

most) the design phase, it is already a known fact, that the traditional way of real-time operation is not feasible for such deep space missions. The simple reason for this is the signal delay between the vehicle and the potential ground station. Once the signal delay becomes too great, real-time interaction between the ground and the vehicle becomes impossible, even with advanced communication concepts like laser relay communication systems.
With the advantages and disadvantages of real-time ops in mind, we can now identify the need for a new operational concept: In order to overcome physical limitations like signal delays, to support crew to the fullest extend without significantly increasing their duties or size, and to keep crewed spacecraft operation cost manageable for all future missions.

1.2 Columbus Operations

In order to support 24/7 real-time operations for the ISS, various Mission Control Center (MCC) were established, which monitor and control specific systems or whole modules of the ISS. This way the safety of the crew, along with the safety of the vehicle, is guaranteed at all times. For the Columbus module, the Columbus Control Center (COL-CC), located at the German Space Operations Center (GSOC) in Munich Germany, is the MCC tasked with performing real-time monitoring and control for the Columbus module and its interfaces to the Harmony module (also called Node 2), which it is attached to. This MCC is further described in [5–8]. Initially flown into orbit in early 2008, the Columbus Module was docked to the ISS on the 11th February 2008. All missions and increments since then were prepared and supported by COL-CC successfully. Details about the preparation and support of these missions can be found in [9–22]. Assuming that the basic setup will not change, COL-CC is capable to operate Columbus until the end of this decade [23]. The latest achievement at COL-CC was the successful completion of Samantha Cristoforettis' Mission "Minerva", covering ISS Expeditions 67, as shown in figure 1.

Fig. 1: ISS Expedition 67 (Photo: NASA)

During her mission she spend a total of 170 days on board the ISS and complete numerous scientific experiments. Furthermore, COL-CC managed to stay operational during the Covid-19 pandemic, without any interruptions to ongoing ISS operations, as detailed in [24].

2. Related Work

In 2018 the International Space Exploration Coordination Group (ISECG) published the Global Exploration Roadmap (GER), an exploration strategy that captured a shared vision from space agencies participating in the group. The GER defines a set of exploration goals, objectives and identifies benefits to humanity of future missions [25]. The central vision is to increase the human presence in our solar system

moving research and missions from LEO, like the ISS, to the Moon and eventually to Mars. With the lunar exploration program, Artemis, NASA is planning a to send astronauts to the Moon by 2026 for the first time in more than 50 years. The center of the program is the development of the Lunar Gateway, to function as a new operations and research outpost in deep space [26]. The outpost around the Moon serves as an intermediate destination to Mars. The GER highlights technologies identified by the ISECG as critical for future exploration missions. One critical technology under the category Autonomous Systems is beyond-LEO Crew Autonomy. On the ISS operations are secured through instant communication from the astronauts to experts on the ground in different MCCs. The 24/7 support is not only administrating, monitoring and maintaining the spacecrafts state but further reacts in case of time critical emergencies. This concept has been securing the astronauts safety and giving the crew enough time to concentrate on doing research and conducting experiments. Due to increasing communication times adapting the same mission concept for human missions conducted to a destination like Mars is not an option. The delay of communication with the Moon will already increase to 2.7s, which stays manageable for real-time ops, even in case of emergencies. Whereas the delay of communication with Mars can go to up to 42 min for a round-trip exchange, which would make a real time communication impossible [22]. Hence for future missions to Mars increasing crew autonomy is a necessity.

Increasing the level of autonomy requires automating the majority of operational tasks currently carried out from MCCs, since crew cannot take over those tasks, without significantly increasing either crew tasks or size. ML algorithms have the potential to implement the needed automation. ML is a branch of Artificial Intelligence that extracts knowledge from data to replicate the way humans learn [27]. These computer systems take advantage of algorithms and statistical models capable of learning and adapting without following explicit instructions [28]. The majority of published ML applications in the space industry are based on satellite or rover data. The fundamental insights are summarized in the following section.

In 2016 ESA organized the Mars Express Power Challenge with the goal to predict the thermal power consumption based on historic telemetry data [29]. A huge part of research in the field of space telemetry analysis is therefore based on the data that ESA made publicly available [30, 31]. An overview of the methods used by the submitted models within the challenge shows that more than half are Random Forest models. The second most popular method used were Recurrent Neural Networks [32].

Another core research area, that can benefit from utilising ML, is the detection of anomalies in satellite telemetry data [33]. Up to now operations are secured using a Fault Detection, Isolation, and Recovery (FDIR) system. The FDIR logic has the aim of detecting, isolating and recovering faults at unit, subsystem or equipment level. For monitored parameters the general approach is to set hard upper and lower limits. If the parameter in question breaches one of those limits, a so called Out-Of-Limits (OOLs) is displayed and further recovery actions are triggered if applicable. At GSOC a statistical outlier detection tool, called ATHMoS, is used to differentiate between nominal data and suspicious data from satellites [34]. For the detection of anomalies Centre National d'Etudes Spatiales (CNES) developed a ML tool called NOSTRADAMUS. Their system is based on a One-Class Support Vector Machine algorithm [35]. Another approach by Tariq et al. uses Long-Short Term Memory (LSTM)-Networks for multiple telemetry channels at once. More specifically they suggest one model for each subsystem [36]. Their analysis is based on data from the Korea Multi-Purpose Satellite 2 (KOMPSAT-2). Their model, dubbed CL-MPPCA, consists of a multivariate Convolutional LSTM, combined with a complementary mixtures of Probabilistic Principal Component Analyzers [36]. In comparison to other state of the art methods their model outperforms them on both precision and F1-score.* Different prepossessing methods were also evaluated with Archive Sampling leading to the best results [36]. They show that using an effective prepossessing method for feature vectors storage space can be reduced significantly without losses in accuracy of the model [36].

Another approach to increase the autonomy of astronauts is the development of assistance robots. As a contribution to the GER in 2018 CNES described a roadmap for the development of a virtual assistant named AI4U. The core focus is taking care of the human-computer interaction on the planned habitats and bases for the Moon and Mars. According to Shashkova et al. [37] AI4U is an emotional meta-bot acting as an interface between the astronauts and their environment. Overall the three core objectives of AI4U are

*In statistical analysis of binary classification, the F-score or F-measure is a measure of a test's accuracy. Source: https://en.wikipedia.org/wiki/F-score

supervising the habitat and relieving the astronauts from repetitive tasks, supporting the astronauts work and increasing crew autonomy, as well as serving as a companion in stressful and isolating situations. In their paper Shashkova et al. mainly focus on the character design. The technical requirements and Artificial Intelligence (AI) methods are not specified in detail. Besides AI4U other projects like CIMON-2 are already being tested on the ISS. The current research objective of this robot is to support the astronauts with the procedural steps of experiments and fly autonomously in zero gravity [38]. The current development of robot assistants shows that the primary benefit is the human-computer interaction and verbal assistance provided to the crew. They are currently not capable of solving the issue of not having real-time ground support. In perspective, however, these robots have the potential to provide an interface for the astronauts to other decision-making systems, like the one developed in this paper, via voice as well as facial recognition.

3. Method and Setup

A reference commonly used in Data Mining and ML projects is the Cross Industry Standard Process for Data Mining (CRISP-DM) [39]. The process consists of six sequential phases shown in figure 2. The core focus of this paper is based on phases one to five. Deployment strategies and tools were discussed in a previous publication [40]. Therefore the Deployment phase will not be discussed further in the following.

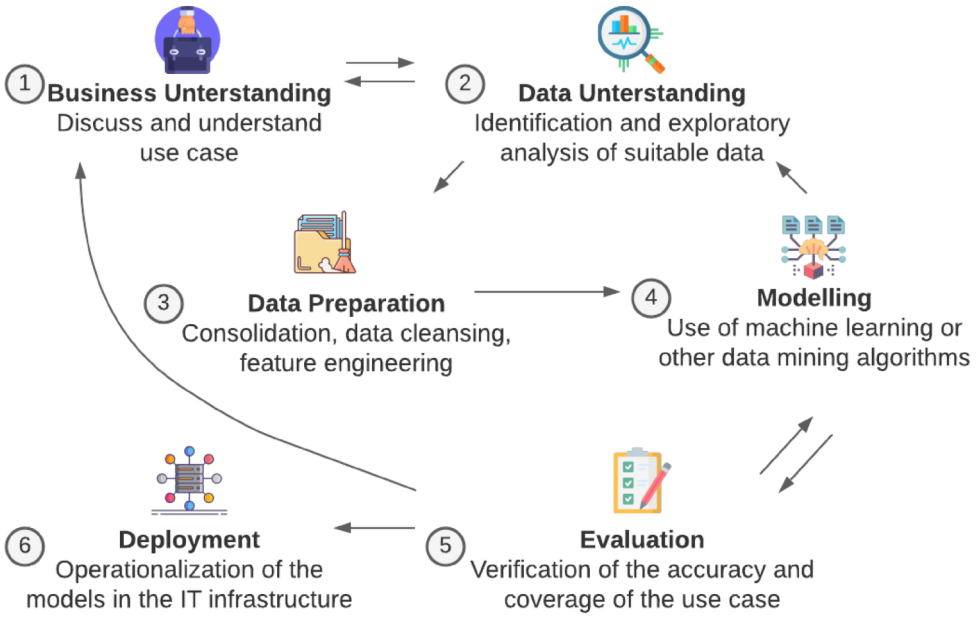

Fig. 2: CRISP-DM Model Phases [39]

The *Business Understanding* phase focuses on defining the goals and requirements from the company's perspective. A concrete task is derived from the knowledge and, based on this, a rough procedure is planned to achieve the set goals [39]. Within the *Data Understanding* phase, the available data is collected, followed by an analysis and evaluation of the data quality. Primarily, data quality problems are to be identified and, if necessary, hypotheses about the hidden information can already be formed on the basis of the data [39]. In the *Data Preparation* phase, a final data set is created from the original raw data, which will serve as the basis for the modelling phase. Data preparation tasks include the selection of tables, records, and attributes that would be useful for modelling. In addition, the data will be transformed and cleaned [39]. The data preparation phase is followed by the *Modelling* phase, where the modelling techniques are selected and applied. In this process, there are several techniques for the same type of data mining problem, so it may be necessary to revert to the Data Preparation phase, since some techniques specify a different data form as a basis. When the technique is applied, the parameters are calibrated to optimal values [39]. The

model(s) will then be evaluated and their quality will be compared [39]. Since the *Evaluation* phase can offer further insights in business understanding and potential models may not offer sufficient results the process can be initiated from the top several times [39]. If the results are sufficiently satisfactory the model will be deployed in the *Deployment* phase.
In this paper, the following sections will focus on the first four stages by giving an overview on the classification task, the processing and transformation of the raw data and algorithms tested. Chapter 4 then evaluates the results.

3.1 Business Understanding: Definition of telemetry formats

To achieve proper data quality for the modelling phase, the prerequisites for the data are as follows:

- The data consists of simulated telemetry from the Columbus module of the ISS. The telemetry is time-series data, meaning a measurement is taken at regular intervals, saved on-board, encoded, downlinked, decoded and saved on ground.
- The frequency at which measurements are taken is either a) 1Hz (once per second) or b) 0.1Hz (every 10 seconds). Details on which parameters are taken at which frequency are given in the section hereafter.
- After the parameters are saved on ground, they have to be retrieved from an archive server before they can be processed further. During retrieval, the data is not altered in any way.
- Once the parameters are retrieved, they have to be fitted to a certain format for later processing.

With these prerequisites in mind, several steps need to be implemented in the *Data Preparation* phase, in order to:

1. extract the data in large quantities from the archive for pre-defined timeframes,
2. post-process the extracted data to assign each parameter in each timeframe a certain class,
3. bring the data into the following format:
    - Parameter Name:
        - List of timestamps
        - List of values
    - Parameter Class
4. remove any duplicate entries, convert timestamps from UNIX epoch to UTC, and map discrete values to numerical values.
5. Furthermore, due to the different frequencies with which measurements are generated, certain timestamps exist where not all parameters have a value, but are instead assigned "NaN". In this case, the parameter needs to assume the value from the last timestamp a measurement was taken.

Now that we defined the prerequisites coming from the data and goals how to transform the data, we will have a deeper look at how data is generated, followed by a detailed description of the scripts that were developed to prepare the data for the *Modelling* phase.

3.2 Data Understanding: How data is generated

This section is split into two parts: first, we will have a look at how data is generated on-board, how data is stored, how it is encoded, and how it is finally downlinked to ground. Second, we will have a look at how the data is handled on ground until it arrives in the archive. This includes how the data is forwarded to the ground-node where it is decoded, how the data is transferred to the ground controllers' workstation who is responsible for real-time monitoring of the parameters, and how it is finally archived afterwards.

3.2.1 Telemetry Data Generation on-board the Columbus module

First of all, data generation on-board is not trivial due to the complexity of the engineering that went into the design and assembly of the Columbus module. We therefore have to consider two aspects separately. The first one is the physical devices or sensors, which are generating measurements. For this, we consider the Columbus module to be an integrated system consisting of several key sub-systems. Those subsystems are:

- Environmental Control and Life Support System (ECLSS)
- Electrical Power Distribution System (EPDS)
- Thermal Control System (TCS)

The following paragraphs will give a short summary of each subsystem, their purpose, which devices and sensors they contain, and their interfaces to the Harmony module.

ECLSS: Environmental Control and Life Support System The ECLSS is the most complex of the Columbus systems. The ECLSS main purpose is to provide air conditioning and pressure control, as well as supply vacuum, venting and nitrogen to dedicated experiments (also called payloads). It has a multitude of devices and sensors to fulfill those main purposes. The main parts involved in air conditioning are the devices for condensate removal, the control units for maintaining the set cabin temperature, multiple fans providing inter- and intra-module ventilation, as well as numerous shut of valves to control the airflow. Pressure control and payload supply all work by means of pneumatically or electronically actuated valves. In addition a multitude of sensors are build into the devices or are stand-alone in the cabin to analyse e.g. air, airflow, air constituents, amount of generated condensate, and many more. Regarding ECLSS interfaces to Node 2, the subsystem receives air supply for ventilation and nitrogen supply for payloads. In turn, Columbus returns air for ventilation, as well as cabin atmosphere samples and condensate water. For the ECLSS, a total of 864 on-board parameters and measurements are available. Of those 864, 309 are analog, and 555 are digital values. In this context analog means that those measurements are analog signals (e.g. voltages), which are measured and digitized on-board before downlink. In addition, 354 are generated at a frequency of 1Hz and 510 are generated at a frequency of 0.1Hz.

EPDS: Electrical Power Distribution System The EPDS is responsible for voltage conversion, power distribution, power protection and module illumination. All other components in the module are powered from either of two Power Distribution Unit (PDU)s, which are the main devices providing the EPDS functionality in Columbus. The maximum power distributed inside Columbus is 18kW. For this, incoming voltage is first converted from 120V to 28V, and after conversion, power is distributed in either 120V or 28V to the subsystems, module illumination units, payloads (external and internal), and several power outlets placed throughout the cabin. Power protection is achieved by means of solid state power controllers inside each PDU, as well as special fuses for the external payloads, housed in a dedicated box. Columbus is illuminated by 8 lighting units, identical to lights in other parts of the station. Switching of lights can be done in groups of four from control switches, situated aft and front of the hatch for easy access by crew. Each light can also be switched on and off individually. The only interface of the EPDS to Node 2 are two power feeder lines connecting the PDUs to the 120V plane of the remaining station. The EPDS generates a total of 1132 on-board parameters, split into 422 analog and 710 digital, of which 336 have a 1Hz and 796 have a 0.1Hz frequency.

TCS: Thermal Control System The TCS purpose is to maintain Columbus equipment and payloads within their required temperature ranges. This is achieved through three functions. 1. Cooling: Here, heat is collected using a water loop that rejects the heat to the ISS External Thermal Control System (ETCS). 2. Heating: Where the module shell and critical pressure control equipment are heated to prevent condensation and freezing. 3. Insulation: To minimizes temperature fluctuation and prevent condensation from forming on the cooling pipes. The TCS cooling function works with a single water cooling loop with two temperature sections: a low temperature section at 4-6^{o}C and a moderate temperature section at 16-18^{o}C. The low temperature section cools one condensate collection device at a time, and the moderate temperature section cools the system components and payloads. Water circulation is achieved by a water pump, with a redundant

pump available as back-up. Heat rejection to the ETCS is performed by two heat exchangers installed in series. The water temperature is maintained by mixing cold water from the heat exchangers and warm water from a bypass. This process is automated and controlled by the active pump, with two dedicated valves controlling how much water gets mixed between the cold water and the bypass. Various other valves allow for isolation of single branches or entire loop section. Shell heaters are used to prevent condensation on the module shell. Further heaters are used to prevent freezing of the depressurisation valves in case the cabin atmosphere has to be vented overboard. Two other control units provide power to and control the shell and depressurisation valves heaters. Last but not least, insulation limits heat gains and leaks. The module shell is insulated using multi-layer-insulation and surface coatings. Also, the low temperature water lines are insulated to prevent condensation in the module. Interface wise, the TCS rejects heat via the two previously mentioned heat exchangers, placed at the outside of Node 2. The TCS consists of a total of 486 on-board parameters, 132 of them analog and 354 digital. 104 of the TCS parameters are generated at 1Hz, and 382 are generated at 0.1Hz.

The second aspect is how the generated data and measurements are handle on-board after generation. For this purpose, the Columbus module has a dedicated Data Management System (DMS) and a dedicated Communications System (COMMS). In the context of this paper, the DMS and COMMS systems play only a secondary role, since their main purpose is to ensure the proper functionality of the remaining subsystems (ECLSS, EPDS, and TCS). However, since they are also responsible of routing, storing and processing data on-board, as well as to and from ground, they play a significant role when it comes to data processing. Therefore certain aspects of those two subsystems will be highlighted hereafter.

DMS: Data Management System The Columbus DMS is split into two layers: The "vital" DMS, which ensures safety critical functions are available at all times, and the "nominal" DMS, which covers functions not safety critical and include the majority of Columbus subsystem functions. The nominal layer is independent and physically separated from the vital layer. The nominal layer cannot issue vital layer commands to ensure that the nominal layer cannot interfere with the vital layers functions. On the vital DMS, data is collected directly via two vital on-board computers and packaged for downlink, if required. On the nominal DMS on the other hand, special Command and Measurement Unit (CMU)s serve as input/output units, collecting the incoming electrical or thermal sensor signals transforming them into bus compatible bit codes and vice versa for commanding. For this, they connect to other Columbus subsystem devices via hardwired connections. Furthermore, a dedicated bus interconnection allows vital data to be sent to the nominal DMS for downlink, which is the usual way vital telemetry is downlinked. All computers of the nominal DMS are participants to the so called "Shared Data Pool". This means that these nodes constantly exchange data to make it available on each node. One of the nominal computers then generates telemetry packets for downlink, using data from the Shared Data Pool, and sends them via serial lines to the COMMS for downlink. Telemetry packets are a formatted block of data which can be exchanged between computers on the network. For the Columbus module there are several types of packets, however we will focus on S-band and Ku-band telemetry packets which are downlinked to ground. During nominal operation, vital ground packets are packets generated on one of the two vital computers and sent to the nominal DMS for downlink. These packets can only be activated at 1Hz and only on S-band. Therefore, every parameter of the ECLSS, EPDS and TCS polled at 1Hz can be considered vital telemetry. Nominally there are no vital ground packets active, as it is very resource intensive and the data is also available in the nominal DMS. As mentioned, nominal S-band ground packets are generated from the Shared Data Pool. These packets can be activated at 1Hz on Ku-band or 0.1Hz on S-band. This is where the 0.1Hz data of the subsystems is coming from and can be considered nominal telemetry. Depending on the packet configuration on-board, not all on-board telemetry is downlinked at all times. If e.g. equipment is powered off, no telemetry is generated by this piece of hardware and thus the entries in the Shared Data Pool are empty. On the ground controllers console this telemetry is then displayed as either "not received", "not acquired", or "static".

COMMS: Communications System Columbus COMMS functions comprise audio and video communication, as well as data routing and bandwidth allocation. All previously mentioned telemetry packets go via the COMMS to the US side for downlink. Once the packets are routed from Columbus to the Destiny module, the packets are sent via dedicated S- and Ku-band antennas towards an available Tracking and Data Relay Satellite (TDRS). Ground packets sent via S-band are referred to as Low Rate Data, and

ground packets sent via Ku-band are referred to as Medium Rate Data. From the TDRS data is forwarded to the ground stations' antenna dishes in White Sands, New Mexico, which then sends the data to the two main NASA control centers: Mission Control Center Houston (MCC-H) at Johnson Space Center (JSC) in Houston, Texas, or Huntsville Operations Support Center (HOSC) at Marshall Space Flight Center (MSFC) in Huntsville, Alabama.

3.2.2 Telemetry Data Processing from the ISS to Ground Consoles

Now that we covered how a) data is generated on-board, b) data is processed and packaged on-board, and c) downlinked to ground, we will have a short introduction of what happens to the data once it arrived on ground. To be able to perform Columbus operations COL-CC needs different elements that can provide all the services and interfaces needed between NASA, COL-CC, the different research centres, and the Russian control centre. All of these elements together form the ESA Ground Segment. It provides COL-CC with the capabilities to continuously monitor telemetry and send telecommands to Columbus in real-time, to ensure the health and safety of the module, and to talk to or see the inside of the module. The ESA Ground Segment is grouped into two categories: 1. the infrastructure refers to everything, which is related to the transfer and storage of data and services, and 2. the (ground) subsystems refer to software and hardware supplying and receiving data. Here, we will focus on how data is exchanged between NASA and COL-CC (recall, after downlink, telemetry packets first arrive at NASA facilities at JSC and HOSC), and how data is process, once it has arrived at COL-CC.

On the infrastructure side, to transport data on the ESA Ground Segment, COL-CC is connected to several other centres in Europe and over the world via the Interconnection Ground Segment (IGS). The data that is exchanged on the IGS includes telemetry packets, telecommands, science data, files, voice and video. The different services that are carried over the IGS are carried over dedicated virtual or physical links. Redundancy is generally realized via a secondary connection. Once the data arrives at COL-CC it is stored in a dedicated Storage Area Network (SAN), which is also part of the infrastructure. From there data can be retrieved by the (ground) subsystems.

The Data Service Subsystem (DaSS) is the central interface for COL-CC to receive and distribute telemetry data. Therefore it has to be able to receive the telemetry from several facilities in parallel (especially the main MCC in Houston, Texas, for S-band data and the HOSC, in Huntsville, Alabama for Ku-band data), archive a copy and distribute it simultaneously to all users that request data. The DaSS is only implemented in software, so it runs on servers provided by the SAN, where its archive areas are also located. A complete set of DaSS software installed on the required hardware is referred to as an instance. DaSS instances can always only support one mission mode at a time, i.e. operation, simulation or test. Therefore three DaSS instances have been installed at COL-CC to allow parallel activities. The software for telecommanding and telemonitoring is called the Monitoring and Control System (MCS). Its core is running on servers also located in the SAN, but it also has its own storage area. The MCS in turn is made up of different clients. Those clients communicate with the servers and the MCS then communicates with the DaSS and MCC-H via networks provided by the IGS. Another parallel to the DaSS is that an MCS instance can only support one mission mode, therefore three instances exist to allow parallel activities and redundancy. To summarize, once the data arrives on ground, data is routed via White Sands to MCC-H or HOSC, then via IGS to the DaSS, and from there through the SAN to MCS, where the telemetry packets are finally available for use by the flight controllers. Currently the SATMON tool is also utilized on top of MCS for visualizing and archiving the incoming telemetry stream.

3.3 Data Preparation: Pre-processing and Classification of the Data

Now that we understand how data is generated on-board and how it arrives at COL-CC we can finally focus on which data is used, how data is retrieved from the archive, and which steps are necessary to bring it into the correct shape for later processing and modelling.

First of all, let's take a look at which data is used. In order to generate a viable set of training data, we did not use historical archive data from the Columbus module, but instead generated a set manually using the Columbus TQVS, which is the highest fidelity simulator available at COL-CC for Columbus. While the three key subsystems are modeled in the Simulator, the behavior is still slightly different to real life. E.g. in the actual Columbus module, the pressure in the water loop varies around a specific value, since the pressure

sensors have a certain absolute error and due to other physical effects influencing the water loop, while in the simulator the value stays constant, because those effects are not implemented. However, the main advantage of using the TQVS is that we are able to acquire data for certain activities and anomalies, where no or only very limited historical data from the real Columbus module is available. An overview about the activities that were performed, and how many minutes of telemetry we generated this way, is given in table 1.

Activity Class	Number of activities	Total data generated in [min]
Nominal Timeline	9	40
Nominal Maintenance	26	476
Anomaly Caution	7	104
Anomaly Warning	16	119
Anomaly Emergency	2	47

Table 1: Activities performed using the Columbus TQVS

Nominal Timeline activities refer to activities, which would normally be planned in advance in the NASA mission and timeline planning tool OPTIMIS. Those activities are "regular" tasks either performed by ground or by crew, depending on the specific activity. For example whenever a crew member is performing scientific experiments with a certain payload, the payload is activated from ground prior to experiment execution. The activation is planned for in advance and placed on the timeline, so that crew and ground controllers know at what time the activity starts and what has to be performed as part of the activity. *Nominal Maintenance* refers to activities that are also planned, but are ground only. Meaning those activities can be performed by the ground controller, without the need for crew assistance or intervention, by sending software commands to the on-board computers, observing changes in downlinked telemetry and verifying expected system behavior. Those activities are (mostly) to perform vehicle or system maintenance, like periodic valve cycling to prevent the valve from getting stuck at a certain position if it remains there for too long. The three anomaly classes represent the way anomalies are announced and handled on board. Depending on severity of the anomaly event, different types of warning tones and visual messages are displayed on ground and/or the on-board computers. Activities belonging to the *Anomaly Caution* class are those, that are performed in response to on-board cautions, the lowest severity incident on-board. An example here would be the loss of redundancy of a specific device, e.g. one of the two delta pressure sensors for the cooling loop. *Anomaly Warning* activities are more critical and activities belonging to this class have a higher impact to on-board functionality, then their Caution counterparts. Warnings are triggered for example if sensors fail that are actively used by the on-board software for closed loop control, like PID controllers tasked with maintaining a predefined cabin temperature. Lastly, *Anomaly Emergency* events are events of the highest severity. Here, the crew needs to react immediately to ensure their own safety above all else. An obvious example is a fire breaking out on-board the station.

In addition to the total telemetry we generated by performing different activities (ref. table 1), we also collected telemetry for idle operations, i.e. for times where the subsystem are running but are not actively commanded by ground or crew, e.g. during the night. This yielded another 950 minutes of telemetry data.

Once activities were performed, the generated telemetry had to be extracted from the SATMON archive at COL-CC. However, before data can be extracted, a script had to be developed to make sure only the relevant parameters, sorted by subsystem and type, are extracted. For this, the script first checks which parameters are known to the system (around 26k in total) and compares them to the parameters which are generated on-board (recall: 864 for ECLSS, 1132 for EPDS, and 486 for TCS). This is to make sure the parameters exist in the system and are available for extraction. Afterwards it checks which data types (1Hz vs. 0.1Hz and analog vs. digital) are represented in the parameters and finally writes the parameters to be extracted to a dedicated .json file[†] for each data type and subsystem. With this, another special scripts was developed, interfacing with SATMONs own http-servers' Representational State Transfer (REST) interface.

[†]JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays (or other serializable values). Source: https://en.wikipedia.org/wiki/JSON

The purpose of this script is to pick up the previously generated .json files containing the parameter names, and request those parameters from the SATMON archive REST interface via http for specific timeframes. The extracted data is then saved to yet another .json file. However, due to limitations in the http interface, the response is just a list of the requested parameters, in the form:

- Parameter Name
- Timestamp
- Value

Therefore, after extraction, a third script was developed to process the http response and bring it into the shape as mentioned in 3.1. This script also assigns the classes from table 1 to each parameter and converts the timestamps from UNIX epoch to UTC, rounding the timestamps from micro-seconds to seconds (e.g. 0.5s gets rounded to 1.0s). Furthermore, in this script, discrete values are converted to binary ones, using one-hot encoding, so they can be processed by the used ML algorithms. In addition, due to the different sampling frequency, certain timestamps exist where not every parameter also has a value, i.e. their value for this timestamp is set to "NaN". Here, the script inserts the value from the previous timestamp, where a value was last present. Also, as previously mentioned, depending on the state of the on-board systems, certain parameters can be either "not received", "not acquired", or "static". If a parameter for the requested timeframe has one of those three states it is ignored by the SATMON REST interface and not included in the response. So, last but not least, in addition to sorting and formatting the response from the server, a fourth script also checks which parameters were not extracted, by comparing the extracted data to the expected parameters, and saves them to a dedicated .json file as a list sorted by data type. This is to make sure the number of parameters stays consistent throughout all activities and timeframes.
Now that we have shown how data is generated, packaged, downlinked and processed on ground (i.e. Data Understanding), and how data is extracted and processed afterwards (i.e. Data Preparation), we can finally move on to the modelling phase of the CRISP-DM model in figure 2.

3.4 Modelling: Comparison of Machine Learning Algorithms

Depending on the learning task and the problem definition, ML can be divided into two types: supervised and unsupervised learning. In supervised leaning next to a feature vector the data consists of expert knowledge in form of classifications or labels. The goal of supervised learning algorithms is to build a model based on the data. This model should take the feature vector as input and return the label for this feature vector as output. In order to learn the chosen algorithm is fed with data containing the feature vector and the classification learning their representation. The model is then evaluated by classifying previously unseen data and making a comparison of the predicted and real classes or values. Some well-known algorithms used for supervised learning are: k nearest neighbors, decision trees, random forests, support vector machines, or neural networks [41]. In contrast, the characteristic of unsupervised learning is that the data is unclassified. Unsupervised learning involves the systematic learning process of a system to represent and display the input signals in a way that reflects the structure of the entire collection of input signals [42]. With the help of unsupervised learning, data can be classified without having prior information on the classification. The results must therefore usually be interpreted again by experts in the respective domain [41].
For this work we generated data based on activities using the TQVS. The data entails classes that were described in chapter 3.3. The learning task can therefore be categorized as supervised learning, and more specifically a classification problem. Previous work like [32] shows the potential of random forest models. Random forests are based on the foundation of decision trees. The following chapters give a theoretical overview of both methods.

3.4.1 Decision Trees

Decision trees are one of the most popular type of supervised learning method. In this process, directed and ordered trees are created that represent decision rules. These decision rules are derived based on past data. These are usually simple binary decisions that have only two possible answers. The algorithm for building decision trees is constructed using the divide-and-conquer method. In this method, a complex

severe problem is divided into smaller partial solutions. The data is differentiated according to a certain criterion at each node until all data can be assigned to one leaf node. This process is described in more detail below [41]:

1. First a root node is created, where the training data is given as input to the decision tree.
2. Then the best selected attribute is chosen to split the data according to a certain criterion. There are several methods for selecting the splitting criterion.
3. Accordingly, a branch is added to the node for each value of the split.
4. The data is divided into subsets according to the respective division.
5. Finally, the second and third steps are repeated for each leaf node until a stop criterion is reached.

Figure 3 represents a simple partitioning. Here the first split occurred on $x2 \geq a2$. Then, the two subspaces were again partitioned: The left branch was split on $x1 < a4$. The right branch was first split on $x1 \geq a1$, and one of its subbranches was split on $x2 > a3$ [43]. The first node equals the root node and the last nodes represent leaf nodes.

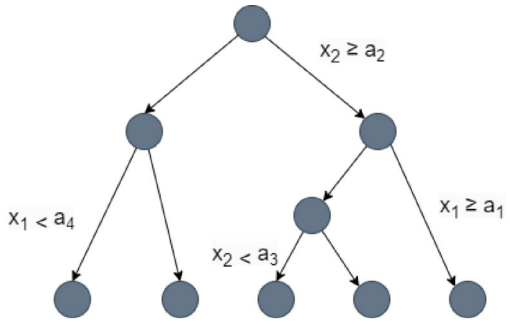

Fig. 3: A graphical representation of a decision tree [43]

The algorithms for the implementation of decision trees differ in three core elements: Splitting criterion, stop criterion, and pruning [41]. With the splitting criterion, there are several measures for selecting variables. For example, the variables could be selected according to the largest information gains, the lowest error rate, or be chosen according to the "Gini" coefficient. The Gini coefficient calculates the amount of probability that a given feature will be misclassified when randomly selected. For the stop criterion, there are two ways of termination of the decision tree, either a tolerance level for the error rate is set, or a certain depth of the tree is reached. Pruning is a technique that prevents a decision tree to over-fit on the training data. In the process of pruning parts that do not provide the power to classify instances are removed [41]. The structure of decision trees is relatively simple and transparent. These properties bring the advantage that the results can be interpreted and the decisions are comprehensible. In addition, decision trees have a very high predictive power compared to other machine learning methods. During model building, the decision tree automatically selects the most relevant key attributes [41]. However, there are also some disadvantages. Multidimensional data sets can produce complex decision trees. This may increase the prediction accuracy due to the large number of variables considered, but decrease the interpretability and simplicity of the decision tree. In addition, there is a risk of overfitting the model on the training data. This could lead to the decision tree being well fitted on the training data, but not suitable for unknown data. An option to decrease the risk of overfitting is the adjustment of hyperparameters. Moreover in 2001 Breiman random forests were introduced to make decision trees more robust and correct the problem of overfitting [44].

3.4.2 Random Forest

Random Forest uses the concept of decision trees as a basis. Let us assume we have a training set of N training examples, and for each example we have M features. The special feature of random forest is that the algorithm grows N_{tree} different randomized decision trees, or estimators [45]. The randomness is injected in two ways [44]. The first component is to bootstrap an independent sample from the original data. Prior to the construction of each tree, an a subset of size n is randomly drawn from the original dataset N, with or without reclamation. These - and only these - observations are considered in the tree construction. Second, when splitting a node, the best split is found over a randomly selected subset of m predictor variables instead of all M features, independently at each node. Therefore no estimator is presented with the complete training set, since the size is $m < M$ [44]. We can now pass a new incoming example to the random forest model where the feature vector is used as an input for the N_{tree} estimators [45]. Each of the N_{tree} estimators will then make a predication. In the case of classification, we will use majority voting to decide on the predicted class. Figure 4 makes the process visible using three decision trees. In this particular random forest classification tree 1 and tree 3 predict Class A for our incoming example and tree 2 predicts Class B. Leading to a majority vote and final classification to Class A.

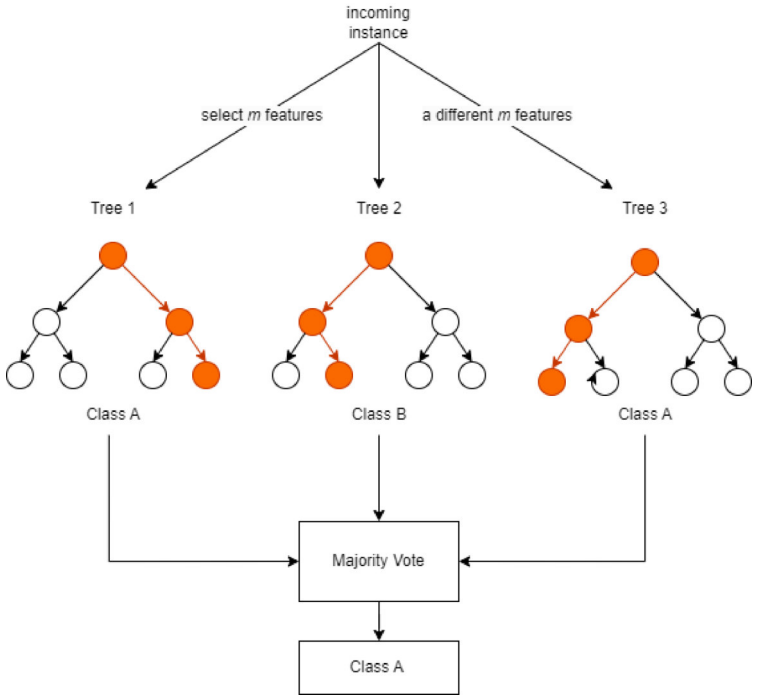

Fig. 4: Random forest inference for a simple classification example with $N_{tree} = 3$ [45]

An advantage of the algorithm is that it can be used to identify the most important features from the training dataset [46]. Moreover, random forests mitigate the problem of overfitting, which often occurs with decision trees [46]. Arising disadvantages are, on the one hand, that the algorithm requires more memory and computing time for the creation of the trees [47]. Moreover, the model, as well as the decisions that are subsequently made, remain a black box and cannot be interpreted [47].
In the following chapter the results of training both methods, classical decision trees and random forests, with the data generated by the TQVS, are summarized and evaluated.

4. Results

The goal of the ML model is to correctly differentiate and classify the predefined classes, explained in depth in chapter 3.3. To learn the differentiation the model is trained using a training set. Once the model is

trained the performance can be assessed by classifying a previously unseen amount of data with the model. The predicted class is then compared to the real class making visible how good the model can generalize when confronted with new data. Various metrics can be considered for the evaluation of the model quality. The used metrics are briefly explained by the following equations:

$$\text{Accuracy} = \frac{\text{true positive} + \text{true negative}}{\text{total number of observations}} \tag{1}$$

$$\text{RMSE} = \sqrt{(\frac{1}{n})\sum_{i=1}^{n}(Predicted_i - Actual_i)^2} \tag{2}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{D}|Predicted_i - Actual_i| \tag{3}$$

Accuracy is the ratio between the number of correct predictions and the total number of predictions. Therefore the closer the accuracy is to one, the higher the performance of the model is. However, it is not a good measure if the target variable classes in the data set are unbalanced. Due to this drawback further metrics on the error rates are reviewed. In particular the Root Mean Square Error (RMSE) and the Mean Absolute Error (MAE) were measured. Generally the smaller the value of the error rates the better the performance of the model. The acceptable error size strongly depends on the business context and requirements.
The input data format, described in chapter 3.3, was transformed based on standard operations necessary for the calculation of random forest and decision tree models. The lists containing the values and timestamps was transformed into a dataframe where each row contains the telemetry measurements of each unique timestamp. Most ML models are not capable of dealing with missing variables, therefore the frequency differences are interpolated using the previously sent measurement as a filler. The resulting dataframe contains 488 columns including telemetry data from the TCS as well as the timestamp and the class. The dataframe consists of 101663 timestamps. The classes, described in depth in chapter 3.3, are distributed as follows: Idle: 57472 (56%), Nominal Timeline: 2418 (2%), Nominal Maintenance: 28764 (28%), Anomaly Caution: 6327 (6%), Anomaly Warning: 3824 (4%) and Anomaly Emergency: 2858 (2%). The telemetry comprises measurements stored as float values such as the temperature of the water in the water pumps. Besides the numerical values the data contains categorical values, for instance the enablement status for the system control of a valve, which could either be enabled or disabled. These telemetry items were encoded using One-Hot-Dummy Encoding [48]. Therefore the previous 488 variables were increased to 648 variables. When features have different ranges, which is the case here, the data should be normalized. The goal of normalization is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information. Prior to the modelling the dataframe containing only the telemetry measurements were normalized. Thereupon the data is split into a training and test dataset with a proportion of 70 % training data and 30 % test data. The training data is then used for modelling and the test data for the evaluation.
The data splits and models contain random factors. Due to this a seed value controls these factors. Setting a seed makes the results reproducible. A robust model should produce similar results based on different seed values. For the decision tree and random calculation we therefore tested 50 different randomly picked seed values between 1 and 8000. The following table 2 gives an overview on the prediction results for decision trees and random forests:

Table 2: Summary of the results for 50 seed values

Metric	Random Forest	Decision Tree
minimal Accuracy	0.999869	0.999934
maximal Accuracy	1.000000	0.999967
minimal MAE	0.000000	0.000066
maximal MAE	0.000230	0.000098
minimal RMSE	0.000000	0.011452
maximal RMSE	0.020646	0.012804

The results presented in table 2 show that for both models the metrics are relatively stable and only vary slightly. The random forest models have a slightly lower error rate and higher accuracy than the decision tree models. Depending on the seed value all, in case of an accuracy of 1, or nearly all, in case of an accuracy < 1, test data could be correctly classified to one of the six classes.

5. Conclusion and Outlook

Regarding the data, we highlighted how data is generated and the steps necessary to process the data after extraction from the archive. For this paper, we used the Columbus TQVS to generate a total of 1736 minutes of telemetry, sampled at 0.1Hz or 1Hz, for 2482 on-board parameters. The results show promising potential for the task of classifying the data into the six activity classes. With this in mind, the next step is to verify these results for real historical on-board data. For this COL-CC has around 15 years of archived data, which equates to roughly 30TB of data (around 5GB per full 24h). From this archive we need to define a viable set of training data and then test it against a set of test data. Keeping the results of this paper in mind, our hypothesis is that with a large enough training set, the classification task can be performed with the same or better accuracy then a human flight controller would be able to.

As mentioned in the introduction, the reason why we need a computer system with such capabilities is to ultimately shift the task of telemetry monitoring from a flight controller on ground to an on-board system. While the reasons why this shift is necessary was also covered in the introduction, hereafter we also want to highlight other aspects of operations that need to change for crewed spaceflight systems where real-time operation is not a viable option. First of all, commanding and monitoring are usually directly related to each other. Whenever an operator sends a command as part of an activity or procedure, the command will result in changes in the on-board telemetry. So not only do we need to shift the task of monitoring, but we also need to shift the task of commanding to an on-board system. For this, we already proposed a novel way of commanding in [49]. Once we are confident enough both tasks can be performed well enough on their own, we need to combine these two functions to verify if this is a viable option for commanding and monitoring, where a human operator is not available.

These two aspects are also only a part in an larger effort by GSOC and COL-CC to develop a novel concept of operations as detailed in [1], where an on-board assistant takes over the role of the ground personnel (or certain aspects of their responsibilities). We call the overall and complete system Mars Exploration Telemetry-driven Information System (METIS). While METIS is still in the prototyping phase, two of the four envisioned functions are already addressed: commanding in [49] and monitoring in this paper. A detailed description of the envisioned architecture and the remaining core functionalities of METIS will be given in future publications.

Nevertheless, in conclusion, we have shown that already at this stage, shifting telemetry monitoring from ground to a potential on-board system for a crewed spacecraft shows tremendous potential, when utilizing decision trees or random forests for classification. Future tests with additional data from real operations on the Columbus Module and on other subsystems will be required secure that the models are generalizing. Upcoming publications will describe and evaluate these tests. This can help bridge the gap between ground and crew for future missions, where real-time support from Earth is not guaranteed or available.

References

[1] Söllner, G., Sabath, D., & Müller, T. (2018). Columbus operation as basis for future exploration. *International Astronautical Congress (IAC)*, (69).

[2] Kranz, G. (2000). *Failure is not an option*. Berkley Publishing.

[3] NASA. (2020). *Nasa's lunar exploration program overview*. National Aeronautics; Space Administration.

[4] ESA. (2022). *Terrae novae 2030+ strategy roadmap*. The European Space Agency.

[5] Kuch, T., Sabath, D., & Fein, J. (2005). Columbus-cc - a german contribution to the european iss operations. *International Astronautical Congress (IAC)*, (56).

[6] Sabath, D., Nitsch, A., & Hadler, H. (2006). Columbus operations - joint undertaking between dlr and industry. *AIAA SpaceOps Conference*, (2006-5807).

[7] Kuch, T., & Sabath, D. (2007). The columbus-cc - operating the european laboratory at iss. *International Astronautical Congress (IAC)*, (58).

[8] Sabath, D., & Hadler, H. (2008). Management and shift planning of the col-cc flight control team for continuous columbus operations. *AIAA SpaceOps Conference*, (3395).

[9] Sabath, D., & Schulze-Varnholt, D. (2008). First experience with real-time operations of the columbus module. *International Astronautical Congress (IAC)*, (59).

[10] Schlerf, A., Sabath, D., Söllner, G., & Verzola, I. (2017). Implementation of an additional command system, pathing the way for new tasks at col-cc. *International Astronautical Congress (IAC)*, (68).

[11] Bach, J. M., & Sabath, D. (2018). Horizons mission – challenges and highlights. *International Astronautical Congress (IAC)*, (69).

[12] Campan, J., Sabath, D., & Söllner, S. (2019). Increment 56/57 iss events put focus on safety role of the columbus flight director. *International Astronautical Congress (IAC)*, (70).

[13] Stoelzle, A., Sabath, D., Soellner, G., & Versola, I. (2020). System upgrades prepare columbus for a new decade. *International Astronautical Congress (IAC)*, (71).

[14] Bender, F., Boere, M., Goettfert, T., Pruefer, S., Sabath, D., & Soellner, G. (2021). First experience with columbus dms modernization, col ka, operations and ip-based communication. *International Astronautical Congress (IAC)*, (72).

[15] Sabath, D., & Schulze-Varnholt, D. (2009). One year of columbus operations and first experience with 6 persons crew. *International Astronautical Congress (IAC)*, (60).

[16] Sabath, D., Nitsch, A., & Schulze-Varnholt, D. (2010). Highlights in columbus operations and preparation for assembly complete operations phase. *International Astronautical Congress (IAC)*, (61).

[17] Sabath, D., Nitsch, A., & Schulze-Varnholt, D. (2011). Changes in columbus operations and outlook to long-term operations phase. *International Astronautical Congress (IAC)*, (62).

[18] Sabath, D., Söllner, G., & Schulze-Varnholt, D. (2012). Development and implementation of a new columbus operations setup. *International Astronautical Congress (IAC)*, (63).

[19] Sabath, D., Söllner, G., & Schulze-Varnholt, D. (2013). First experience with new col-cc console setup. *International Astronautical Congress (IAC)*, (64).

[20] Baklanenko, M., Sabath, D., & Söllner, G. (2014). New col-cc operations concept and new challenges. *International Astronautical Congress (IAC)*, (65).

[21] Leuoth, K., Sabath, D., & Söllner, G. (2015). Consolidating columbus operations and looking for new frontiers. *International Astronautical Congress (IAC)*, (66).

[22] Bach, J., Sabath, D., Soellner, G., & Bender, F. (2016). Consolidating columbus operations and looking for new frontiers. *International Astronautical Congress (IAC)*, (67).

[23] Sabath, D., Kuch, T., Soellner, G., & Mueller, T. (2014). The future of columbus operations. *International Astronautical Congress (IAC)*, (2014).

[24] Campan, J., & Zoeschinger, G. (2022). Columbus operations throughout the covid-19 pandemic. *International Astronautical Congress (IAC)*, (73).

[25] Group, I. S. E. C. (2018). The global exploration roadmap.

[26] Foust, J. (2021). Nasa inspector general warns of further delays in returning humans to the moon. Retrieved January 21, 2022, from https://spacenews.com/nasa-inspector-general-warns-of-further-delays-in-returning-humans-to-the-moon/

[27] IBM. (2022). Machine learning. Retrieved April 21, 2022, from https://www.ibm.com/cloud/learn/machine-learning

[28] LEXICO. (2022). Machine learning. Retrieved April 21, 2022, from https://www.lexico.com/definition/machine_learning

[29] European Space Agency. (2016). Mars express power challenge. Retrieved October 2, 2021, from https://kelvins.esa.int/mars-express-power-challenge/home/

[30] Gu, Y., Gowda, G. M., Jayanna, P. K., Boumghar, R., Lucas, L., Bernardi, A., & Dengel, A. (2019). The added value of advanced feature engineering and selection for machine learning models in spacecraft behavior prediction. In *Space operations: Inspiring humankind's future* (pp. 439–454). Springer.

[31] Boumghar, R., Lucas, L., & Donati, A. (2018). Machine learning in operations for the mars express orbiter. *2018 SpaceOps Conference*, 2551.

[32] Lucas, L., & Boumghar, R. (2017). Machine learning for spacecraft operations support-the mars express power challenge. *2017 6th International Conference on Space mission challenges for information technology (SMC-IT)*, 82–87.

[33] Hülsmann, M., Haser, B., & Förstner, R. (2021). Artificial intelligence in space: Current status and future challenges – a review. *International Astronautical Congress (IAC)*, (72).

[34] OMeara, C., Schlag, L., Faltenbacher, L., & Wickler, M. (2016). Athmos: Automated telemetry health monitoring system at gsoc using outlier detection and supervised machine learning. *14th International Conference on Space Operations*, 2347.

[35] Delande, P., Lambert, P.-B., Bouayad, M., Zaroubian, M., Baron, A., & Jalabert, E. (2022). Ai for satellite anomaly detection: On-ground operational feedback and development of on-board experiments. *International Astronautical Congress (IAC)*, (73).

[36] Tariq, S., Lee, S., Shin, Y., Lee, M. S., Jung, O., Chung, D., & Woo, S. S. (2019). Detecting anomalies in space using multivariate convolutional lstm with mixtures of probabilistic pca. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2123–2133.

[37] Shashkova, E., Navarro, G., Roy, R. N., Paillet, A., & Truntzler, L. (2022). Study and development of an ai assistant for future moon and mars stations. *International Astronautical Congress (IAC)*, (73).

[38] DLR. (2020). Cimon-2 makes its debut on the iss. Retrieved November 20, 2022, from https://www.dlr.de/content/en/articles/news/2020/02/20200415_cimon-2-makes-its-debut-on-the-iss.html

[39] Wirth, R., & Hipp, J. (2000). Crisp-dm: Towards a standard process model for data mining. *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, *1*.

[40] Speth, F., Hartmann, C., Kebschull, U., Sabath, D., & Sellmaier, F. (2022). Towards transparent ai-systems: Benefits of mlops pipelines for space system development. *International Astronautical Congress (IAC)*, (73).

[41] Maheshwari, A. (2014). *Business intelligence and data mining*. Business Expert Press.

[42] Dike, H. U., Zhou, Y., Deveerasetty, K. K., & Wu, Q. (2018). Unsupervised learning based on artificial neural network: A review. *2018 IEEE International Conference on Cyborg and Bionic Systems (CBS)*, 322–327.

[43] Schonlau, M., & Zou, R. Y. (2020). The random forest algorithm for statistical learning. *The Stata Journal*, *20*(1), 3–29.

[44] Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

[45] Wood, T. (2022). What is a random forest? Retrieved December 21, 2022, from https://deepai.org/machine-learning-glossary-and-terms/random-forest

[46] Synced. (2017). How random forest algorithm works in machine learning. Retrieved December 10, 2022, from https://medium.com/@Synced/how-random-forest-algorithm-works-in-machine-learning-3c0fe15b6674

[47] Great Learnings Team. (2020). Random forest algorithm- an overview. Retrieved December 10, 2020, from https://www.mygreatlearning.com/blog/random-forest-algorithm/

[48] Pramoditha, R. (2021). Encoding categorical variables: One-hot vs dummy encoding. Retrieved December 10, 2022, from https://towardsdatascience.com/encoding-categorical-variables-one-hot-vs-dummy-encoding-6d5b9c46e2db

[49] Hartmann, C., Speth, F., Sabath, D., & Sellmaier, F. (2022). The road to on-board crew autonomy: Using iss' columbus module as basis for ground procedure automation. *International Astronautical Congress (IAC)*, (73).