INTELLIGENT DATA UNDERSTANDING FOR ENTRY, DESCENT, AND LANDING, ARCHITECTURE ANALYSIS

A Dissertation

by

SAMALIS SANTINI DE LEON

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Daniel Selva Valero
Committee Members,	Nancy Currie-Gregg
	Raktim Bhattacharya
	Theodora Chaspari
Head of Department,	Srinivas Vadali

August 2021

Major Subject: Aerospace Engineering

ABSTRACT

Designing Planetary Entry, Descent, and Landing Systems requires analyzing a wide range of architectures and scenarios with high fidelity Monte Carlo simulations of performance under uncertainty. Given the complexity of these systems, datasets contain tens of thousands of parameters describing the system and the environment. These datasets are generally manually analyzed by subject matter experts, trying to find interesting correlations and couplings between parameters that explain the behaviors observed. Such analysis work is critical, given that it could lead, for example, to the discovery of major flaws in a design. While the subject matter experts can leverage their knowledge and expertise with past systems to identify issues and features of interest in the current dataset, the next generation of EDL systems will make use of new technologies to address the issue of landing larger payloads, and may present unprecedented challenges that may be missed by the human.

In this work, we present Daphne, a cognitive assistant, into the process of EDL architecture analysis to support EDL experts by identifying key factors that impact EDL system metrics. Specifically, this work describes the current capabilities of Daphne as a platform for EDL architecture analysis by means of a case study of a sample EDL architecture for an ongoing NASA mission, Mars 2020. Given that the work presented in this work is in its early development, the thesis focuses on the description of the expert knowledge base and historical database developed for the cognitive assistant, as well as on describing how experts can use it to obtain information relevant to their EDL analysis process by means of natural language or web visual interactions, thus reducing the effort of searching for relevant information from multiple sources.

A popular approach to automate the extraction of explanation rules of data is association rule mining, in which rules with high statistical strength are mined from a dataset. However, current rule mining algorithms (e.g., a priori, FP-growth) generate too many rules that are redundant or not useful because they are too complex, too obvious, or don't make sense to the user. In this work, we propose a new approach to improve the comprehensibility, insightfulness, and usefulness

of the association rules generated during the analysis of an EDL dataset by leveraging a user-provided knowledge graph. The knowledge graph captures the user knowledge about EDL and the specific problem at hand. We then use a statistical relational learning framework based on probabilistic soft logic to assess the degree of consistency of the rule with our knowledge of the system. We hypothesize that rules that are considered more consistent with the knowledge graph will be perceived by the user as being more comprehensible (making more sense) than rules that are less consistent with the knowledge graph. We test this hypothesis – and more generally the relation between our proposed metric and the perceived usefulness and insightfulness of a rule – in a small study with N=6 subject matter experts. Results support our primary hypothesis and also show interesting relationships between comprehensibility, usefulness and insightfulness of the extracted rules. These findings can enable a more personalized and adaptive approach to intelligent data understanding, a key enabling technology to help aerospace organizations make sense of the large and heterogeneous datasets that are becoming available in many areas of science and engineering.

DEDICATION

To my mother and my father for your love and support.

ACKNOWLEDGMENTS

First, I would like to thank my advisor Dr. Daniel Selva for his time and patience. The SEAK lab has been like a second family and I am very thankful for the support from everyone.

To Dr. William Warmbrodt at the NASA Ames Research Center all I can say is:"How cool is that ?" One milestone down and many more to go. This research was funded by the NASA Science Research and Technology Fellowship (NSTRF) and thanks to this opportunity I have had the opportunity to collaborate with Dr. David Way. I am especially thankful for his patience and the time he has taken to expose me to the field of Entry, Descent, and Landing and all of his perspective as a researcher in EDL on how my research can contribute to the EDL teams. Dr. Way incorporated me into the EDL team at Langley and JPL from day one and I am very thankful for the opportunity to work on a mission up to landing day.

I would like to thank my two special groups of friends. First, my childhood friends. We have been friends since we were only 12 years old and we still stay in touch every day. Every single one of these women is a fighter and each went through their own set of obstacles and now we have doctors, scientists, physicists, and adventurers. I admire you all greatly. Second, I want to thank the family of "Maya" friends I got during my undergraduate studies. I never thought I'd make lifelong friends in a single place.

Last but not least, I would like to thank my family and my life partner, Markus Guerster. My mom and dad have been through more than anyone I know and they are a true source of inspiration. I am eternally grateful for all of the support they have given me from day one. Science was always a passion of mine since Iwas little and my biggest fear was not being able to obtain an advanced degree due to the hardships my family went through. Nevertheless, my parents and my partner have always reminded me that I can achieve whatever I propose myself. This encouragement helped me get through rough times and made me the person I am today.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor Daniel Selva, Raktim Bhattacharya and Nancy Currie-Gregg of the Department of Aerospace Engineering and Professor Theodora Chaspari of the Department of Computer Science and Engineering.

The data analyzed used in this dissertation was provided obtained during case studies and Operational Readiness Tests for the Mars 2020 mission. The analysis shown in the dissertation was conducted by the main author. These datasets, however, were also analyzed with the Mars 2020 EDL team for mission purposes.

Funding Sources

This work is funded by a NASA Space Technology Research Fellowship (NSTRF), grant number 80NSSC18K1635.

NOMENCLATURE

ARM	Association Rule Mining
EDL	Entry, Descent, and Landing
IDU	Intelligent Data Understanding

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION AND LITERATURE REVIEW*

1.1 Motivation

The success of landing rovers and humans on Mars relies significantly on the capabilities of planetary Entry, Descent, and Landing (EDL) technology developments. Up to the present day, fundamental EDL technologies used for robotic landings on Mars have been derived and extended from the Apollo program's capabilities developed during the 1960s and 1970s. The Mars 2020 (M2020) mission defines the current state of the art for EDL -which successfully landed a 1-ton rover on Mars. The M2020 architecture consisted of seven segments: exo-atmospheric flight, Guided Entry, Parachute Descent, Terrain-Relative Navigation, Powered Descent, Sky Crane, and Flyaway. Many M2020 architecture elements were derived and extended from the Viking Pathfinder (MPF) and the Mars Exploration Rover (MER), resulting in improved target precision and successful landing [4]. However, the next generation of M2020-class landing vehicles will be delivering larger payloads at tighter delivery ellipses that are pushing the limits of current EDL technologies [5].

Earth-based testing of Mars Entry, Descent, and Landing (EDL) systems is limited. For this reason, EDL system analysis relies heavily on simulation techniques. These high-fidelity simulations simulate the operation of the system under various entry conditions and model parametrizations (e.g., gravity, planetary geometry, atmospheric, aerodynamic, control system, guidance, and navigation models). Results produced by simulations support trade studies, system development, testing, and operations [6]. They also provide a means for analysis of performance under uncertainty. Given the complexity of these systems, datasets produced by these simulations are often extensive. For example, a Mars 2020 Entry, Descent, and Landing (EDL) simulation evaluates 8,000 trajectories and generates over 15,000 output variables. These large datasets are manually analyzed by the subject matter expert, who searches for conspicuous correlations and couplings

*Parts of this chapter have been adapted from "A Cognitive Assistant for Entry, Descent, and Landing Architecture Analysis" (2019)[2] and "Interactive Explanation of Entry, Descent, and Landing Simulations" (2020)[1] by Santini De Leon, S., Selva, D., and Way, D. with permission from IEEE and AIAA, respectively.

between parameters and assesses the sensitivity of figures of merit to key parameters. This analysis work is crucial since it may lead, for example, to discovering a significant flaw in a design. Since the analysis is manual, it is subject to human limitations such as information processing or biases due to experience and expertise. Nevertheless, the current approach suffers from an important limitation. While the subject matter expert can leverage his or her knowledge and expertise with past systems to identify issues and features of interest in the dataset, the next generation of EDL systems may present **unprecedented challenges** that may be missed by the human.

Because of the limitations of the manual approach, NASA has suggested that end-to-end EDL architecture analysis can benefit from computational advances to reduce analysis cycle time, reduce cost, identify areas of risk, and ensure mission success [7]. More specifically, NASA has suggested incorporating **Intelligent Data Understanding (IDU)** in multiple domains to aid subject matter experts in the task of **knowledge discovery** from data. IDUs are a form of information processing that seeks to find **high-value and/or actionable content** in large datasets such as, identifying targets or events, for example, and take the appropriate action [8]. Up to the present day IDUs have been employed in spacecraft for on-board detection, data prioritization, planning and scheduling [9]. **However, it's use in performance analysis of complex systems has not been explored**. Furthermore, the main challenge with these technologies is the ability to identify **high-value content that makes use of existing knowledge about the system** (e.g. pre-specified rules about interesting events) without relying on subject-matter experts actively providing information to the system [7]. Given these limitations, NASA's Technology road map for Technology Area Breakdown Structure (TABS) 9.4.1 (Architecture Analysis) states that future EDL Technologies could benefit from computational advances. These capabilities can help reduce analysis cycle time, minimize architecture life cycle cost and ensure mission success [7].

The problem described above is fundamentally a situation where a user has to make sense of a large dataset to make a decision. State-of-the-art data mining algorithms and tools are already helping engineers in similar data-driven decision-making problems tackle the challenges mentioned above. One common application is tradespace exploration studies during early mission

formulation [10, 11]. These tools use data mining to help system engineers explore thousands of designs and obtain insights[12]. Another related important application is spacecraft anomaly detection and diagnosis [13, 14, 15, 16, 17, 18], where the goal is to identify outliers given a time series dataset (detection), and then identify a root cause for a given anomalous signature (diagnosis). For this paper, we restrict ourselves to situations where the decision-maker is a human, since that exacerbates the importance of the explainability of the rules mined. However, this need not be the case. For example, data mining algorithms are used for spacecraft on-board data processing tasks such as cloud masking [9, 19]. In all these cases, some aspect of explainability are still important for a posteriori analysis and reconstruction of system behavior.

A typical method for knowledge discovery in large datasets is association rule mining. This family of algorithms extracts patterns expressed as logical rules using *if-then* statements. These statements are used to map observations to outcomes such as: "IF entry mass increases, THEN Mach at parachute deploy increases." Logical rules have been widely employed as knowledge representations in artificial intelligence, since foundational work by Newell and Simon proposed them as a model to mimic how humans reason [20, 21]. Association rule mining algorithms extract logical rules from a dataset by looking for frequent patterns in the data (e.g., “many” simulations show high entry mass and high Mach at parachute deploy). One drawback of rule mining algorithms is that they often produce many association rules [22]. Furthermore, often times these rules are too complex, too obvious, or make little sense to subject matter experts [23, 24, 25]. These limitations make it difficult for experts to process the content in all of the rules, identify interesting rules, interpret findings, and use the information for decision making.

Most algorithms tackle this limitation by adjusting statistical measures of a rule (e.g., support and confidence of a rule). However, support thresholds can neglect rules with small support and high confidence, which might contain more interesting rules. Other approaches have developed objective interestingness measures (e.g., heuristics and constraints). However, they do not account for context in the domain nor do they consider background knowledge. On the other extreme, some work has proposed incorporating subjective interestingness measures to identify surprising,

novel, or actionable rules. This typically involves users actively providing “general impressions" to the data mining algorithm so that the algorithm can identify rules that deviate from what users expect to see with the goal of finding rules that contradicts previous beliefs, for example. Although approaches like that one may help uncover more interesting rules, they require that experts continually provide information to improve the data mining process and uncover interesting rules. Although some hybrid approaches exist, there is still a **large dependence on experts** to further prune mined rules.

This thesis aims to employ IDU for the analysis of EDL simulations. More specifically, this work seeks to use a cognitive assistant as a platform for IDU to achieve three main functions: 1) summarize the statistics of large datasets (summarization), 2) identify input variables that appear to be driving the output variables (sensitivity analysis), and 3) identify features and behaviors that appear to be common among failing cases or some other region of interest of the design space (association rule mining).

Given that rule mining algorithms often produce many association rules and exacerbate the task of identification of interesting rules, **we propose combining knowledge graphs and statistical relational learning to improve the comprehensibility, insightfulness and usefulness of mind association rules**.

The remainder of this chapter is organized as follows. Section 1.2 discusses how EDL systems are studied via Monte Carlo simulations. This section also introduces several approaches to analyzing Monte Carlo simulation outputs in other aerospace domains. Section 1.3 presents an overview of Intelligent Data Understanding technologies. Section 1.4 introduces knowledge discovery methods and tools used to extract information from a domain. This section includes an introduction on knowledge representation and frameworks useful for extracting information from them. Section 1.6 discusses the approach and goals of the work presented in this theses given the limitations discussed.

1.2 EDL Analysis

This section, provides an overview of how EDL system analysis is carried out. This section also discusses approaches used to identify driving features and methods used to help explain simulation outputs.

1.2.1 EDL Analysis Under Uncertainty

EDL system analysis requires analysis using high fidelity simulations to assess performance, and risk under uncertainty. In addition to Earth-based testing limitations, Mars EDL trajectories are highly coupled to major sources of uncertainty that include but are not limited to vehicle aerodynamics, launch window, and atmospheric conditions during day-of-entry events. NASA uses the Program to Optimize Simulated Trajectories (POST-2) to simulate different entry conditions under many model parametrizations (e.g., gravity, planetary geometry, atmospheric, aerodynamic, control system, guidance, and navigation models)[26]. POST-2 uses Monte Carlo dispersion analysis techniques to help users evaluate performance under uncertainty, assess mission-level feasibility, identify off-nominal behavior, and support system design trades, among other capabilities [27].

POST-2 based Monte Carlo dispersion analysis are employed as early as Pre-Phase A of a mission lifecycle [26, 28]. During the early phases of a mission's lifecycle, manual analysis of Monte Carlo results is relatively straightforward. The models are relatively simple, and the number of parameters used to describe the system is relatively few. However, as the mission's lifecycle progresses, the fidelity and complexity of models increases. For example, recent statistics showed that the Mars 2020 mission conducts roughly 257 simulations each year. Each simulation contains 8,000 random trajectories and generates around 15,000 output variables. For MSL, the number of trajectories generated in each Monte Carlo was the same; however, the number of outputs was merely 4,545 variables. This increase is interesting given that the Mars 2020 EDL architecture is largely replicated from MSL but with a new algorithm for parachute deploy and terrain-relative navigation. This fact helps highlight how much fidelity and complexity of POST-2 has evolved in the past decade. Regardless, the analysis of these complex simulations remains done manually

by experts. Although past successes are an indicator that this analysis is possible, it is **time and resource consuming**. Furthermore, there is a risk of **information overload**, which often leads to time and money waste and missed scientific opportunities, as reported in the literature [29], [30]. Nevertheless, wile this works for systems with some heritage, the **next generation of EDL technologies may present new unprecedented challenges** for EDL experts that may significantly increase the challenge of data analysis.

1.2.2 EDL Analysis Process

The typical analysis of an EDL system under uncertainty has three main components: simulation case setup, data processing, and performance assessment. During the simulation case setup, experts establish the objectives of the study. For example, a simulation case may be case where the system has a known failure (e.g., an instrument does not collect measurements), whereas in other instances, a simulation case may be simply a Monte Carlo with updated models (e.g., atmospheric model). Once objectives are established, experts set up their models in POST-2 accordingly and generate input data using random sampling techniques that are provided as an input deck to the simulation, as depicted in Figure 1.1.

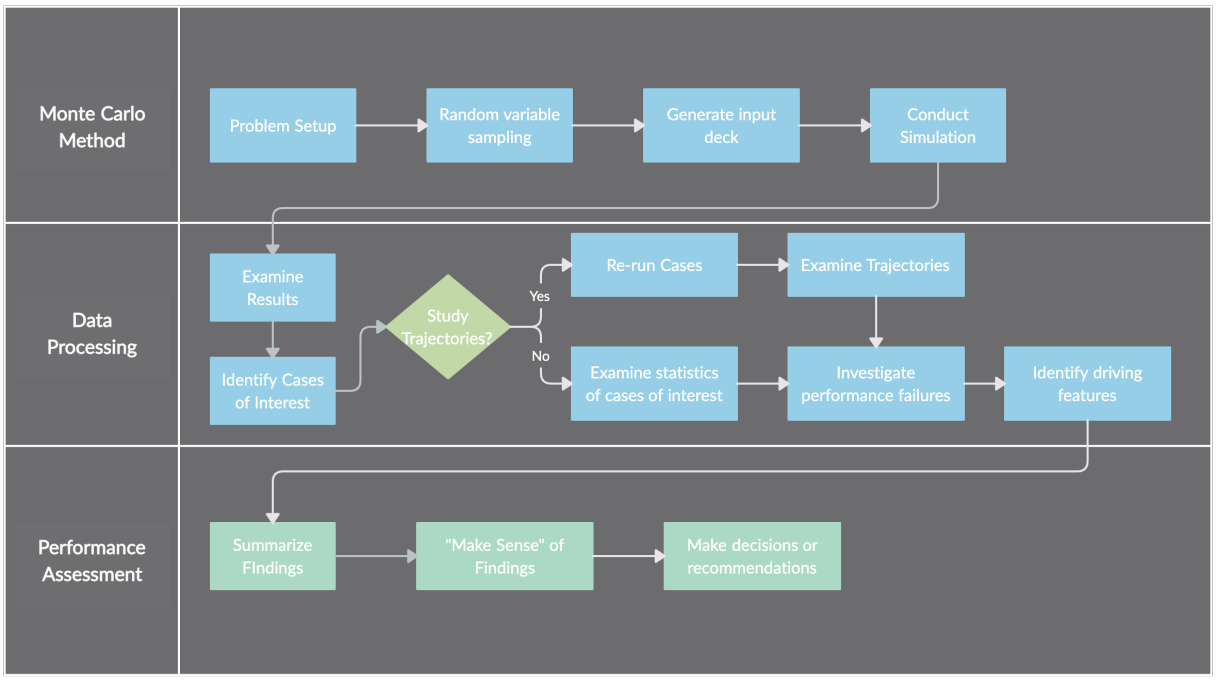


Figure 1.1: EDL analysis process (Adapted with permission from [1]).

Once simulation cases are executed and completed, experts face the laborious task of data processing. This portion of the analysis process is the most important and the most time-consuming portion of the analysis cycle [31]. Here, experts analyze results with the objective of characterizing system performance. To partially tackle the task of examining all 15,000 output variables across all of the trajectories generated and reducing the cognitive workload of this process, experts use a Scorecard. This document is a type of summary report that describes mission-specific system performance metrics, the main simulation results (e.g., various percentiles, means), threshold values, and whether the results satisfy system requirements. Although this documents contains information deemed most relevant, this task still involves examining about 1,000 system performance metrics.

Up to the present day, there is no prescription for the data processing task, and unfortunately, it often relies on the team's expertise of the system under study [2, 31]. A typical approach to tackle the data processing task is to identify changes in the Scorecard metrics. This task often involves

identifying changes the statistic of a metric and whether it is labeled as “flagged" or “out of spec." For example, if the 99%-tile of fuel consumption does not satisfy system requirements (e.g., out of spec), experts search in the dataset what trajectories fall outside the established thresholds. In some instances, as depicted in Figure 1.1, experts study the trajectories of the cases of interest selected [1]. In such scenario, it is necessary to compute the trajectories in POST-2 and examine each. This approach is often insightful, given that they may point experts to a particular region in the design space, which is often a particular point in time (i.e., parachute deploy, backshell separation). In other instances, however, anomalous behavior in the trajectory is not apparent. If experts opt not to look at trajectories, they must examine the statistics of interesting cases or standard package plots generated by their software. Regardless of the path taken in the outlined process, this task is critical given that understanding performance failures and combinations of conditions that degrade the system’s performance is necessary to characterize the system in terms of driving features. **In EDL, and for this work, we refer to driving features as a parameter or a combination of parameters that largely determine a metric’s value**. For example, driving features for peak deceleration include mass and entry velocity.

With a fully characterized system, as depicted in Figure 1.1, the objective is to summarize discoveries and make sense of the different outcomes. The manual “making sense" task is the most challenging of all given that it involves examining the implications of all of the findings on the system and how it affects performance and probability of success. These findings also enable decision-making and may highlight an issue that was not considered a priori. Although the manual approach to data processing may help identify some commonalities and patterns (e.g., similar trajectories with anomalies in a similar region) that summarize system performance, they do not always make any emerging behavior in the system apparent. Hence, on many occasions, performance assessment relies on previous system expertise and may fail to capture unknown possibilities.

1.2.3 Monte Carlo Analysis Techniques and Tools

As systems become more complex (e.g., non linearity and number of models) and more data becomes available, the task of making sense of large volumes of data can become increasingly challenging. This increase in size and complexity was evident between the MSL to Mars 2020 simulations. Furthermore, we argue that models used in EDL simulations are so complex that the idea of seeing the problem as a black box model is not too far-fetched [32]. Consequently, understanding how uncertain inputs contribute to the uncertainty of outputs can become a challenge. Nevertheless, assessing system performance and understanding its behavior remains critical to ensure mission success.

The most common approach to understanding the performance of complex systems, such as EDL, relies on sensitivity analysis techniques. This type of analysis is carried out with the goal of understanding how uncertainties in input variables propagate and affect the uncertainties of outputs [33, 34, 35]. In the context of EDL, we can apply this method to identify how aerodynamic uncertainty affects the landing ellipse, for example [36]. This information is valuable given that it can serve to confirm, for example, that aerodynamic uncertainty remains the biggest contributor to uncertainty in the landing ellipse size. In can also provide information on how much is landing ellipse affected by aerodynamic uncertainty relative to other input criteria.

Most sensitivity analysis algorithms can be classified as either local or global. Local sensitivity analysis methods encompass derivative-based approaches that employ one-at-a-time (OAT) techniques. In other words, sensitivities are identified by varying one parameter at a time while maintaining all other inputs fixed. Hence, they provide information on the influence of individual parameters and fail to consider interactions between inputs. Global sensitivity analysis methods, on the other hand, assesses parameter influence by allocating uncertainty across all input ranges. The most widely employed methods are density-based methods [37, 38], variance-based [35, 39] and surrogate model-based methods [40, 41, 42]. Density-based measures are used to study the empirical distributions (e.g. PDF and CDFs) of model outputs. These are useful for given that they are moment-independent, thus, they are simple to compute. However, to obtain accurate distribu-

tions, they require sufficient simulation runs [38]. Variance and surrogate-based methods, on the other hand, have demonstrated good performance in capturing interactions between input parameters. However, they are computationally expensive for exploring high-order interactions as they rely on evaluating the model with a larger number of samples than typical simulation run.

1.2.3.1 Sensitivity Analysis in EDL

Specific to the field under study, most of the assessment of sensitivities in the system is achieved using the OAT approach. Given that varying individual input parameters is unfeasible considering input uncertainties approach 1,000 varibales, the most common approach is to employ grouping methods. For example, a typical example in EDL is to logically group variables together by model to avoid the curse of dimensionality. For example, to assess model contributions to a metric, parameters are classified by model (e.g., guidance and control, thermal, aerodynamics, atmosphere) [36]. If we were to assess the contribution of atmospheric uncertainty on the landing ellipse, all variables in each model are fixed (i.e., held constant) while atmospheric variables are dispersed. This process is repeated until all individual contributions are quantified.

The OAT approach is useful for obtaining an intuition on how system performance is affected by individual or groups of variables [35]. However, this task often involves **running many Monte Carlo simulations**. However, due to size and complexity, in the EDL domain this approach is constrained to grouping and neglects to reflect the influence of individual parameters. Hence, it is not particularly useful for diagnosing. In other words, it does not provide any information on whether high/low values of a particular input cause a performance failure. Furthermore, the OAT approach neglects to capture the effects of these particular interactions in the overall system. Consequently, **there is a need for tools and methods that help explain these analyses**.

1.2.3.2 Other Approaches to Analyzing and Explaining Monte Carlo Simulation Results

Many other fields in the aerospace domain face similar difficulties and can benefit from computational advances to identify sensitivities and performance drivers. Nevertheless, there has been some work for analyzing these simulations with the aim of identifying likely causes of system

failure that use methods that go beyond the OAT approach.

Some common approaches to sensitivity analysis include analysis of variance (ANOVA) [43], scatter plots [33], regression, and variance-based sensitivity analysis [39]. Some problems have employed the ANOVA method to compare population means, showing the effects of interactions between design variables and objective functions [44]. Scatter plots on the other hand, are a graphical method that presents the relationship between individual inputs to outputs [33]. Although very simple, they can be useful to identify linear relations and trends between input to output mappings.

Regression analysis is a statistical method used for investigating the relationship between variables. The resulting coefficients from a linear regression model are indicators of sensitivity [45]. For non-linear regression, ordinary least squares can be used to estimate a surface response model. Response surface models can be useful for identifying what conditions of input variables lead to maximum or minimum response of a target metric [46, 47]. Furthermore, variance-based sensitivity analysis decompose variance of the model into terms attributable to each input. One advantage of this method is that it quantifies contributions of individual as well as combinations of parameters [39].

Other approaches in the aerospace domain have made use of test-vector generation and hypothesis testing to identify driving features. This approach relies on deriving test vectors from the original uncertain vector. For example, if a vector caused failure, a combination of N test vectors are generated by generating combinations of the elements of the original uncertain vector [48]. This method requires conducting further Monte Carlo simulations with the new vectors and performing hypothesis testing to determine whether a particular parameter is significant or not. In their framework, the null hypothesis is that each uncertain parameter has no influence on whether the results from the Monte Carlo simulation are satisfactory or not. Hypothesis testing is achieved by means of an upper-tailed test on the probability of x failures occurring, given that if a parameter is influential, it is expected that the number of unsatisfactory cases increases. To explain the results, this method provides a ranking of influential variables based on the upper cumulative probability (i.e. the cumulative probability of those failed cases).

Although the aforementioned approaches are widely used for sensitivity analysis, there are several inherent disadvantages with these approaches. Namely, there are three critical limitations discussed by Restrepo et al. [49]: 1) some analysis techniques require re-running simulations and/or manipulating data (test vector generation, variance-based sensitivity, response surface method, scatter plots), 2) statistical techniques (e.g., ANOVA) allocate amount of variance rather than explaining interactions, and 3) other analysis techniques operate as black boxes so the user no longer has control over the model's internal representations.

Re-running simulations in the EDL domain is computationally expensive. With the usual setups, a simulation takes about six hours to run, and usually about one additional hour to setup. For grouping of variables, and considering that EDL simulations contain 12 models, this equals to 84 hours of runtime to obtain the needed data to asses contributions of each model to critical EDL metrics.

When using statistical techniques, for example, the variable ballistic coefficient can be classified by one of these methods as a driving feature for failed cases given that the touchdown velocity exceeded the requirements, but the method does not provide any notion of the effect of its magnitude on the target metric (high/low) nor any indication as to the cause of this correlation. One alternative could be to use an unsupervised neural network technique to classify and predict failure regions in a low dimensional space; however the neural network does not explain which parameters are driving the calculated predictions [44].

For some applications, explanations to such level of detail might seem unnecessary. However, we argue that high-stake scenarios, such as the EDL phase of a mission, deserve explainable models that help experts understand what is occurring in the predictive model (i.e., physics-based Monte Carlo Simulation). Especially if the resulting data products and interpretations are intended to support mission-critical decisions.

Some notable work in the aerospace domain that partially overcome the limitations just mentioned are the Tool for Rapid Analysis of Monte Carlo Simulations (TRAM) and the work conducted by Gundy et al. [49, 50, 51]. TRAM provides a ranking of individual variables and com-

binations of influential variables by means of density-based sensitivity analysis. TRAM compares the distribution between simulation cases that passed and failed system requirements. The tool also employs k-nearest neighbors to cluster data points and distinguish between feasible and infeasible regions(e.g., success/fail regions) in the design space. Gundy-Burlet et al., on the other hand, employ treatment learning as a data mining technique for finding good heuristics that generate particular outcomes. Treatment learning relates to association rule mining except it searches for patterns that distinguish a target class from contrasting classes [52].

Outside the aerospace domain, some of the most notable work is the Automated Statistician (www.automaticstatistician.com). This tool explores plausible models on data, generates explanations and delivers a summary of the findings in written natural language form. However, some of the drawbacks of the work discussed is that most techniques and tools focus on **fully automated analysis** and explanation of data and analysis techniques are isolated from one another. Given that these tools are fully automated and explore the entire design space, they often **provide more information than the one the user deems necessary**. Another inherent disadvantage is that the work discussed focuses on identifying driving features for pass/fail criteria scenarios exclusively. In the EDL domain, often times experts are more interested in specific regions of the data set. For example, during a divert maneuver after backshell separation occurs, experts might be interested in identifying what drives the direction of the maneuver selected. Other times, experts might be interested in what is influencing cases that fall above the 99%-tile range, for example. Finally, these tools only tackle one aspect of the explanation task: **explaining how the model behaves rather than explaining why particular outcomes occur**.

1.3 Intelligent Data Understanding

Intelligent Data Understanding is a subdivision of NASA's Intelligent Systems Program- which aims to incorporate practices from the field of computer science into the aerospace domain [8]. These efforts arise from the need to provide the necessary means to help experts cope with data growth as systems become increasingly complex [53]. Some motivation for IDU also arises from the fact that missions rely heavily on ground control of on-board spacecraft procedures [7]. This

task is achievable for Earth observation, but deep space missions cannot rely on ground control due to communication delays [54][55]. Nevertheless, having systems that accelerate the data processing task and provide more autonomy is a step forward to great advancements in IDUs. This will be achieved by reducing analysis cycle time while providing good representations of data that lead to a better understanding of the system under study.

Most applications of IDUs in the aerospace domain are designed to provide on-board information processing capabilities to spacecraft [7]. The goals of on-board data processing are to analyze data, perform data reduction, identify high-value content and prioritize tasks accordingly or take intelligent action. The state of the art of spacecraft IDUs is defined by NASA's SpaceCube [56] and the Earth Observing-1 satellite [19].

IDU's are closely, if not completely related to Intelligent Data Analysis, which aims to automatically analyze data and extract useful information [57]. As with Intelligent Data Analysis, IDUs seek to combine elements from the fields of data mining and knowledge discovery [8]. However, the end-to-end capability this program seeks to achieve has 4 elements that are key to advancing IDUs [8].

The first element is **transforming data to information**. A good example of this task is the NASA Scorecard, which takes raw data from a Monte Carlo simulation and generates tables with critical EDL metrics, the statistics of each metric, and whether the values satisfy requirements or not [2]. The second element is **transforming information to knowledge**, which is highly dependent on context, personal experience, and internal representations individuals have about a system [58, 59]. The third element seeks to provide the capability of performing **knowledge discovery with less human intervention**. This aspect requires that systems possess knowledge that is shared among experts so that it can employ common heuristics without relying on humans. The final element is obtaining **knowledge that is actionable**.

The end product we envision for IDUs are tools that contain a portfolio of algorithms that aid experts in the data processing, data analysis and understanding tasks of data. This paradigm shift will help reduce analysis cycle time, reduce the cognitive load imposed onto experts by the task,

and help ensure that all relevant information is available. This advancement, in turn, provides more time and space to focus on scientific opportunities [60, 8, 61].

1.3.1 State of the Art for IDU

Up to the present day, IDUs have been primarily employed for on-board data processing and analysis [62]. These technologies have been incorporated on several spacecraft, with NASA's SpaceCube 2.0 being one notable technology development [9]. This spacecraft incorporated a hybrid science data processing system that provides the system with first-responder real-time awareness, enables multi-platform collaboration and conducts on-board data processing [56]. Data reduction is done on-orbit, which allows for instruments to collect more data without increasing on-board storage. On-board detection capabilities can detect events/features such as fires and ocean color, for example, and can broadcast the data products.

Another notable example is NASA's Earth Observing-1 (EO-1) [19]. This spacecraft has featured IDU technology for on-board planning and scheduling tasks. on-board processing capabilities enable the system to detect science events and respond accordingly. For example, EO-1 can perform on-board cloud detection and data targeting. NASA's EO-1 performs data acquisition, downlinks data to a processing center and scientific events are detected. In the event that an event is detected, a request is forwarded to its planning system which enables data collection and downlinks relevant data [63, 64].

Another paradigm explored by IDU in the Earth Observation domain is to employ data mining for estimating incomplete data obtained by remote sensing [65]. This approach seeks to employ Virtual Sensors and data mining techniques to obtain rich data from incomplete records. The idea is to use old records and available rich information with the objective of obtaining results sooner and minimize cost by using alternative methods to obtain the desired data [66].

While most of these technologies have demonstrated advancements in the automation aspects, they all **lack on-board intelligence**[62]. Current technologies employ on-board processing and analysis but do not incorporate autonomous decisions making to optimize this process. Developing more “intelligent" tools may help increase the percent of decisions made autonomously. At the

moment the NASA SpaceCube 2.0 and EO-1 can conduct about 30% of decisions autonomously which means that on-board intelligent action still relies heavily on human intervention [60]. Another element that deserves attention is that IDUs have been mostly employed for operational processes to define the knowledge in a given process, but **its use in performance analysis has not been studied**.

1.3.2 Limitations of IDU

While automation reduces the need for experts to be involved in the data analysis task, we argue that in some domains, like EDL, can benefit from **interactivity from the user as well as automation of certain tasks**. One drawback of systems that only possess autonomous functions is that the end user has to navigate through a large amount of information to find aspects of the data she/he is interested in, potentially resulting in information overload [67]. We argue that tools such as the Automated Statistician [68] can benefit from more interactivity with the user, for example to allow him or her to interactively specify the type of information, family of models, or region of interest in a dataset for further analysis. The interaction between the human and the tool can thus be enhanced by means of cognitive assistants with advanced dialogue and interactive capabilities. Others have also emphasized the potential of human-in-the-loop data mining tools to improve the process of knowledge extraction [69].

On the other hand, we acknowledge that certain tasks during the analysis cycle time become repetitive. Using EDL as an example, the task of generating a scorecard, and examining statistics, for example, can benefit from automation. Although these tasks are more procedural, some aspects of the discovery process can also be automated. Every domain, like EDL, employs heuristics and rules of thumb that help assess the validity, or importance of information extracted by algorithms [70]. Hence, on-board intelligence can help further prune information extracted by these systems as opposed to having to rely on experts repetitively searching in the data and identifying interesting or non-interesting information.

Another challenge with IDU is obtaining machine-readable information and the necessary knowledge representation relevant to the analysis. This task often becomes increasingly com-

plex given that most of the analysis of mission elements is done in isolation- which often results in different modeling techniques and heterogenous data formats [71]. In many instances mission-critical information comes in the form of semi-unstructured information (e.g., Json, HTML) and in many instances, unstructured raw text [72]. This makes the task of information extraction from individual sources and data fusion challenging. Consequently, there is a need to develop systems that can obtain data from multiple sources and fuse it to derive knowledge [60].

1.3.3 Cognitive Assisstants as a Platform for IDU

Cognitive assistants (CA) have been explored as a viable platform to provide decision-making support to experts in the face of uncertainty. Furthermore, they provide one of the capabilities IDU technologies lack: user interactivity. Unlike other artificial intelligence tools and applications, CAs can obtain domain-specific knowledge in ways that follow a teacher-apprentice approach [73]. Hence, a CA can learn from "rules of thumb" of dos and don'ts for a domain-specific application. CAs can also incorporate a knowledge base with pre-specified knowledge of a domain. However, they can still **exploit AI and data analysis techniques that conform the essence of IDU technologies**, to quantify the probabilities and states of a particular decision [74]. A CA can be useful for identifying features of interest in a design; analyzing and communicating the findings to team members; providing historical or contextual information; and more generally reducing cognitive load on the team members [75]. In the context of EDL, CAs can help experts identify anomalies, **features of interest**, and **extract knowledge** that could potentially not be attainable by manually examining a data of the architecture under evaluation, for example. To exploit the interactivity features CAs are characterized for, EDL teams can specify to the assistant aspects of the data they are interested in and what type of analysis to conduct.

At the moment, CAs in the aerospace domain have been mostly created with the intent of providing situational awareness and subsequent operational decisions making tasks. For example, COGAS, a CA, supports NAVY ships in air target identification. COGAS makes use of sensor information and a priori expert knowledge contained in their models (e.g., operator activities in their work domain) to process acquired data, identify and analyze the system's state, establish sys-

tem goals, and activate the appropriate procedure [75]. Other CAs within this application domain include the Crewed Assistant Military Aircraft (CAMA) and the Digital Copilot [76][77]. Along these lines, interest has arisen in integrating CAs for supporting astronaut crew during missions beyond Low Earth Orbit (LEO), especially in off-nominal conditions when there is a long communication delay between Earth and the space vehicle. As for the previous examples presented, a CA for space crew support- with some level of automation- should have the capabilities to diagnose a problem, provide recommendations to the the crew during emergency situations based on previous knowledge, evaluate the diagnoses, perform risk trade-offs, and evaluate and generate procedures [78]. At the moment, virtual assistants, like Daphne [74, 79], have been explored as a platform to aid with this task [13].

1.4 Knowledge Discovery

Knowledge discovery is the *"non-trivial extraction of implicit, previously unknown and potentially useful information from data [80]."* This process often involves some form of data analysis that provides information about the data, such as patterns, for example. Data analysis can include, sensitivity analysis, statistical analysis, or data mining [81]. In some instances and applications, algorithms discover knowledge by either implicit or explicit methods. Implicit methods depict information in the form of visualizations, whereas explicit methods use formal notations to convey information. A popular approach is to express knowledge explicitly is in the form of logical rules. The most common algorithms are decision trees [82], contrast set learning [83], and association rule mining [84].

Logical rules have been widely employed as knowledge representations in artificial intelligence, since foundational work by Newell and Simon proposed them as a model to mimic how humans reason [20, 21]. This also makes them suitable for cognitive assistants if the goal is to communicate and reason how humans do. An example in the context of EDL is a rule that expresses: "if entry mass increases, then super sonic parachute deploy altitude decreases." However, what makes the task of knowledge discovery challenging is identifying discovered patterns that are considered interesting to subject matter-experts [23, 24]. This challenge is exacerbated by the fact

that many data mining algorithms provide vast amount of information that relies on expert judgement to evaluate results, identify valuable and actionable information. For example, the number of association rules discovered by most algorithms increases exponentially with the number of items [85][86].

1.4.1 Survey of Interestingness Measures for Knowledge Discovery

Most association rule mining algorithms extract rules using the support and confidence statistical measures [87][88]. High support thresholds guarantee that only rules that appear frequently are mined, whereas confidence quantifies how often the rule is found to be true. While these measures help prune the rule set, there is no prescription to tunning thresholds since both metrics conflict with one another [89]. For example, setting a high support threshold may eliminate attributes that are infrequent but could be potentially useful for identifying high-confidence rules. Furthermore, reducing both thresholds would result in a large number of rules with high redundancy. Because of this, assessing the interestingness of discovered patterns has become an active research area in data mining.

The goal of applying interestingness measures in rule mining is two-fold. First, they help reduce the number of rules provided by the algorithm, making it easier for subject matter experts to manually examine results. Second, they can help steer the subject matter expert's attention towards rules that are more likely to be potentially interesting and useful to them. The explanation of EDL simulations can be seen as an iterative hypothesis testing process, in which candidate explanations for a behavior seen in the data are generated and examined or tested. In this context, a rule can be useful to provide support for a hypothesis from the user (e.g., "I think what is driving these cases landing so far down-range is that higher tailwinds during parachute deploy are increasing its drift"), or to identify a new candidate hypothesis for the subject-matter expert to examine.

Interestingness measures are primarily divided into two classes: objective and subjective [23]. Most existing interestingness measures are objective, and focus on pruning rules with low statistical significance. The most common are support (Equation 1.4.1 and confidence (Equation 1.4.1) [87, 88]. In both equations, t is a transaction, T is the set of transactions, X is a feature (i.e., an

itemset) and $|\{t \in T; X \subseteq t\}|$ is the number of transactions in T that contain the itemset X. A high value of support is an indicator that X is found often in the dataset. This measure guarantees that only rules for which the antecedent and the consequent appear simultaneously in a significant number of examples in the data are mined. In contrast, confidence quantifies the fraction of examples satisfying the antecedent of the rule that also satisfy the consequent.

$$supp(X) = \frac{|\{t \in T; X \subseteq t\}|}{|T|} \tag{1.1}$$

$$conf(X \rightarrow Y) = \frac{supp(X \cap Y)}{supp(X)} \tag{1.2}$$

Note the similarity of support and confidence with the joint and conditional probabilities respectively of the antecedent and consequent. While these measures help prune the rule set, there are no universal guidelines for tuning these thresholds as they trade off with one another[89]. For example, setting a high support threshold may eliminate infrequent attributes appearing in high-confidence rules. However, a low threshold could be potentially useful for identifying high-confidence rules that are infrequent but some of those rules could be over-fitting the data. Furthermore, reducing both thresholds would result in a large number of rules with high redundancy [90]. Other common objective measures include lift, correlation, Jaccard, Gini index, and information gain, for example. A complete literature review on these measures can be found in [91, 23, 89]. One important limitation of objective measures is that they do not account for context. In other words, they do not consider goals or background knowledge of the domain or the user. While these measures help identify strong rules with respect to attributes such as reliability, conciseness, and coverage [92], they fall short on the task of finding rules that are surprising, novel, and/or actionable [24]. This is partially attributed to the fact that many measures for novelty, surprisingness, and actionability are domain-dependent [84].

Subjective measures, on the other hand, help identify interesting rules based on the expert's knowledge or beliefs. With this approach, a rule's novelty, surprisingness, and actionability are identified by employing some comparison mechanism between the subject-matter expert's belief against the discovered rules. Common measures include syntactic distance measures between rules [93, 94], logical contradictions [95], and probabilistic measures to evaluate conditional probabilities of soft beliefs [96]. Soft beliefs are often provided by experts in the form of rules (e.g., rules experts believe could be interesting).

The largest drawback of mechanisms employed to identify rules based on current user-driven knowledge is that they are often query-based [97]. The system searches for rules that satisfy the user's request. This process may be tedious given that it relies on subject matter experts continuously providing information to the system. In some instances, requests can contain repetitive information.

To partially tackle the limitations of both objective and subjective measures, some work has been done towards incorporating domain knowledge for identifying interesting rules. The work depicted in [98] uses semantic distance between concepts in an ontology present in a rule to prune mined rules. The framework uses general impressions provided by users to further prune the rules. General impressions refers to beliefs a user may have about the association between items. For example, a user may believe there exists a relationship between eggs, bread, and butter among the transactions in a database. Semantic distance is estimated by evaluating the minimum distance between the concepts in the antecedent and the consequent of a rule (i.e, the minimum path that connects the two items in an ontology) [99]. Additional rules are pruned by applying an operator that eliminates all rules that match general impressions provided by the user. The second operator keeps rules that do not conform to the rule schema and discard all other. For example, let us consider an a general impression *AmotsphericWinds* $\rightarrow$ *ParachuteDeploy* where:

$$AtmosphericWinds = west - east, north - south \tag{1.3}$$

and

$$ParachuteDeploy) = dynamicpressure, parachutedrift, mach \tag{1.4}$$

This rule, or general impression, implies that we would expect for parameters that correspond to atmospheric winds have some effect on parachute deploy conditions. Equation 1.3 shows items in an ontology that correspond to the class Atmospheric Winds, whereas Equation 1.4 shows the items that correspond to the parachute deploy segment class.

However, if we were to find a rule with the form:

$$\textit{west-east} \rightarrow \textit{backshell-recontact} \tag{1.5}$$

this rule would be kept as interesting given that it does not conform with the rule schema specified with respect to the consequent. Furthermore, does not consider that a large distance between two items may imply that there is no real relationship between them, thus, the rule might not make sense. In the work discussed in [25, 98], rules that match beliefs provided by the user are pruned, considering that the user would not find those rules interesting since they already know them.

The work described in [100] uses reliability as an objective measure and identifies novel rules. In their work, reliability measures the independence between the antecedent and the consequent of a rule, as shown in Equation. If a rule has a high reliability, but the relationship between items is not explicit in the ontology, the rule is considered interesting. Most other efforts make use of concept generalization using ontologies. Srikant and Agrawal, who first introduced the concept of generalization, proposed that items in a taxonomy with the same parent will have similar associations [101]. For example, a general impression could be: *Fruits* $\longrightarrow$ *DairyProducts*. If there is a rule whose item has a different classification than *Fruits* in the antecedent and *DairyProducts* in the consequent, it could be considered interesting [92, 102, 103, 25, 90].

$$\frac{P(X \bigcap Y)}{P(X))} - P(Y)) \tag{1.6}$$

While these efforts partially tackle the limitation of using domain knowledge for discovering interesting rules, they have several limitations. One limitation is that knowledge provided by user and knowledge encoded in the ontology is deterministic. With the current approaches [93, 98] all beliefs are treated with the same level of confidence when in reality, some beliefs are stronger than others. Furthermore, knowledge captured by the ontology is assumed to be complete. Consequently, they only explore explicit links between entities and neglect possible interactions and emerging relationships across seemingly distant entities. This is a great limitation because complex knowledge representations of domains are often created manually- making them prone to human error. Additionally, information incorporated into knowledge graphs is often incomplete and comes with some degree of uncertainty [104]. Furthermore, with the current approaches [93, 98] all user-provided beliefs are treated with the same level of confidence when in reality, some beliefs are stronger than others.

Another significant drawback of interestingness measures rarely discussed in the literature is the comprehensibility of a rule in the context of its domain. Comprehensibility relates to how easy a rule is to understand. Most past works assess rule comprehensibility based on its number of attributes exclusively. The literature argues that if a rule is concise (i.e., few attributes), it is easily comprehensible [23, 105]. While conciseness can contribute to understanding a pattern, this measure neglects human learning factors such as chunking [106, 107]. Therefore, if the elements in a set of rules do not appear to have association with one another, a human is less likely to extract any useful information.

Nevertheless, we believe that IDU can benefit from some form of knowledge representation in the system that enables automatic inference capabilities. Knowledge can be in the form of general impressions, as employed in the literature or in the form of constraints. This reduces the need for expert to repetitively provide general impressions and knowledge to further prune rules since it can be done automatically. We also believe that domain knowledge captured should go beyond general representation. Ontologies are a powerful tool but they only capture relationships between

concepts and do not include data about elements in the ontology. To exploit the broad variety of data analysis and data mining algorithms knowledge bases should contain information from multiple elements of a system to enrich the analysis.

1.4.2 Domain Knowledge Representations for Knowledge Discovery

To be able to achieve the goals of the development of intelligent systems, much effort has been placed on evaluating ways to incorporate human knowledge into expert systems and machine learning methods [11, 98]. For many decades, ontologies have been widely employed to capture domain knowledge that is useful for data mining. Similar to ontologies, knowledge graphs have recently become a popular tool for capturing such knowledge, fueled by the advent of knowledge graph embeddings and other new efficient machine learning algorithms. Knowledge graphs represent real-world data (e.g., social networks, e-commerce sites) in the form of graphs. In KGs, entities, or nodes, represent objects like people, places, things, for example, and their interrelations [108] as a graph. KGs are often associated with ontologies given that they are both built on semantics. However, what distinguishes them from ontologies is that they incorporate assertions about the schema [109]. In other words, knowledge graphs are ontologies that contain data. Furthermore, with the rapid growth of data and sparsity of information, relational database that contain data useful for data mining have become difficult to expand and maintain. Consequently, there has been a shift from tabular to graphical data storage. This is achieved by using knowledge graphs (KG), or graph databases, to store domain data from multiple sources [110].

Common applications of knowledge graphs in the field of artificial intelligence include tracking networks of people and the connections between them and recommendation systems [111][112]. In recent years, NASA developed a knowledge graph for lessons learned with the objective of tackling the issue of information accessing across groups, and centers. NASA uses text analysis to identify documents that contain similar information across documents [113].

Going beyond accessing information across centers, NASA faces multiple challenges in other domains. NASA's satellites, for example, downlink terabytes of data per day that rely on expert-based analysis [8, 66]. Furthermore, we have evidenced the increase in complexity of models used

to analyze EDL systems- which has led to an increase in the volume of data that needs to be examined. This will only continue to increase as EDL architectures and missions continue to evolve. Consequently, we believe there is some benefit to using knowledge graphs to represent domain knowledge and data for analysis. To the best of our knowledge this technology has only been employed for storing document and satellite data and **have not been employed for performance analysis of a complex system**.

1.4.2.1 Challenges with Knowledge Graph Construction

Although KGs are a versatile tool for machine learning (e.g., inference and logical reasoning), generating a knowledge graph can become very challenging, given that it often relies on manual effort. This task becomes increasingly convoluted when managing large volumes of data; thus, it is no surprise that this approach may lead to human error and information loss. Because of this, knowledge graph construction makes frequent use of link prediction methods to predict missing facts in a graph. Most of the link prediction methods serve two purposes: to complete missing information and uncovering interactions between entities that were previously unknown [114].

Another challenge in KG construction arises due to the fact that information and data come from distributed data holdings. In many instances this data comes in the form of semi-unstructured information (e.g., Json, HTML) and in many instances, unstructured raw text. This makes the task of information extraction from individual sources for knowledge graph construction challenging. Most literature employs named entity recognition (NER) for extracting entities (i.e., people, locations, organizations) from document using grammar-based and machine learning methods. These are then represented in knowledge graphs as nodes[115].

1.4.2.2 Domain Specific Methods for tackling the distributed data problem

One area under exploration by NASA is to employ Model-Based Systems Engineering (MBSE) throughout a mission lifecycle to minimize systems engineering practices that rely on document tracking and standalone analysis- which reduces the issue of distributed data holdings [116]. For example, the Mars Sample Return mission efforts plan to use MBSE to build an integrated model

of concepts of operations, which includes: functional decomposition, operational scenarios, structural decomposition, requirements and traces to other model elements, and authority delegation. For example, for modeling operational scenarios, they used an activity diagram to depict component activities and how a they perform functions [72]. The block definition diagram, on the other hand, provides a functional decomposition of activities and invocations of functions in the system [117]. NASA then maps semantics captured by models into their Integrated Model-Centric Engineering(IMCE) ontology framework [118]- which acts as a centralized knowledge base. With the appropriate tools available, the goal is to perform logical reasoning to identify “logical fallacies" and other other information across mission elements that is critical and hard to detect with the current approaches (e.g., manual and distributed data). **We believe that cognitive assistants are a viable platform for bringing relevant information from multiple sources very quickly at the relevant time.** We also believe that integrating MBSE artifacts, such as lessons learned databases and SysML models into knowledge graphs can be useful for extracting high-value content during knowledge discovery.

1.4.3 Knowledge Extraction from Knowledge Graphs

The task of extracting information from a knowledge graph is commonly framed as a link prediction problem given that often times, the goal is to discover new relationships. Furthermore, graphs are so large and complex and unstructured that manual discovery of relationships is unfeasible. Looking at knowledge extraction from the perspective of a complex system, a knowledge graph may be used to represent an EDL architecture, for example, where nodes represent subsystems, components, and metrics. Furthermore, relationships may represent interactions between components and subsystems. Given the flexibility of knowledge graphs, relationships can also used to capture correlations between metrics and whether they are coupled to any known component or subsystem. One potential discovery, for example, could be that we believed that atmospheric winds only contributed to parachute deploy and recontact based on the explicit links in the graph, however, they turn out to possibly be linked to rover separation and touchdown conditions.

The most common methods employed for link prediction are path-based methods and embedding-

based methods [119]. Path-based methods examine connectivity to quantify similarity between entities in a graph [120]. The most common measures are number of common neighbors, Jaccard's coefficient, preferential attachment, and graph distance. All of these measures are based on the premise that nodes are likely to interact if they have a similar network structure[121].

The common neighbors metric examines the topology of the graph and quantifies the number of shared entities between a node pair [122]. The Jaccard coefficient and the Adamic/Adar also examine the number of shared entities, however, the former evaluates the ratio of common neighbors to the number of distinct adjacent nodes (i.e., intersection over union), whereas the latter computes the inverse log the degree of the node. Unlike the other measures, the Adamic/Adar penalizes nodes with high degree of centrality and places a higher value on a common neighbor if they have fewer neighbors [123]. Furthermore, preferential attachment poses the idea that nodes with many relationships will gain more relationships. This measure proposes that the probability of a relationship between two entities is correlated to the product of the number of direct relationships each entity has [124]. Finally, the graph distance measure, as the name implies, measures the shortest path between a pair a nodes in the graph. Most of the prunning of association rules discussed in Section 1.4.1 that make use of ontologies make use of the shortest path metric to measure the distance between concepts that appear in a rule. The idea is that concepts that are distant in the graph but are present in a rule are more interesting than concepts that are close neighbors.

One advantage of these methods is that local techniques like these are fast to implement and evaluate. However, the most considerable drawbacks are that they only explore the similarity between entities that are "neighbors of neighbors". This means that these metrics, especially the graph distance, do not consider the emergence of links formed at large distances [124] [121].

One of the most popular link prediction methods is graph embeddings. This method "embeds components of a KG including entities and relations into continuous vector spaces, so as to simplify the manipulation while preserving the inherent structure of the KG [125]." The intuition for the link prediction problem is that similar nodes have similar embeddings [126]. Most methods initialize embeddings randomly and improve them by some optimization algorithm. Most approaches make

use of back propagation with gradient descent [114] or matrix decomposition [127] to improve embeddings. Given the embeddings, the problem of link prediction now become a binary classification problem [128]. Binary classification is primarily accomplished by performing logistic regression to predict the likelihood of a link [129]. This is typically achieved by transforming the node embeddings to edge embeddings. Edge embeddings are used to train the model and product “1" or “0" if both entities are related or not, respectively.

Graph embeddings for link prediction has demonstrated exemplary performance in the knowledge base completion task, as it preserves the semantic meanings of entities and relations [130]. However, there are two considerable drawbacks to this approach. First, they do not capture uncertainty of unseen relation facts[104]. Second, they have limited reasoning capabilities. Graph embeddings use algebraic manipulations as opposed to logical inference [131].

Another form of inference on relational data is known as statistical relational learning (SRL). This general approach employs probabilistic inference methods to determine to what extent, or probability, a relationship exists between two entities [132]. The field of SRL has been primarily motivated by the explosive growth of heterogeneous data collected in many domains, given that information is often incomplete or uncertain [133]. What makes SRL robust is that it combines elements from statistical learning, logical reasoning, and relational learning [134].

Nevertheless, unlike graph embedding methods, they capture uncertainty in the model. The most common methods are Bayesian logic programs (BLP)[135], ProbLog [136], and Markov Logic Networks [137]. BLPs integrate the concepts of Bayesian Networks and Logic programming and logic programs. In these programs, rules are defined using first-order logic that capture the structure of a Bayesian network, which is an acyclic graph. BLP create a one-to-one mapping of logical relationships between random variables, and the dependency relation with the logical consequence and the respective dependency relation [138]. This way, they are able to combine qualitative and quantitative components [134].

ProbLog extends logic programming by assigning a probability that a ground atom is true (i.e., random variables and the respective relationship) and assumes that the probabilities are mutually

independent[136]. MLNs are similar but they differ primarily in the fact that MLNs accompany their first-order logic rules with weights as opposed to probabilities. In MLNs weights represent the degree of belief that a rule holds [137]. Together, all pairs of rules and their respective weight are used to ground a Markov Network. Once all groundings are discovered, MLNs are used to predict links between entities by employing probabilistic inference methods such as Markov Chain Monte Carlo (MCMC) Gibss Sampling [139]. In MLN, atoms take boolean truth values $0, 1$.

One of the main drawbacks of BLPs is that they do not use logical connectives, so negations are not permitted. Another drawback is that the language used for defining clauses in BLPs is challenging to read and write [140]. The most notable limitation of ProbLog is that it relies on a closed-world assumption. In other words, it assumes that facts that are not known are false. Both of these limitations are tackled by MLNs. MLNs use rules to capture the relational dependency of the entities under study and use an open-world assumption [141]. Although MLNs are popular, they suffer from the drawback that atoms can only take 0 or 1 truth values. PSL, like MLN, creates a program from first-order weighted rules that capture the relational dependency between entities. However, in PSL, atoms take continuous truth values in the interval of [0,1]. PSL conducts inference using MPE to find the truth values of atoms that maximize the likelihood of rules being satisfied. This framework is much faster given that it frames the inference task as a convex optimization problem that can be solved in linear time [142].

1.4.3.1 Probabilistic Soft Logic

PSL is an SRL framework that supports reasoning about similarities in relational domains [134]. A PSL program is formed by a set of weighted rules. Each rule in a PSL program serves as a template for hinge-loss potentials that when grounded, induces a Hinge-Loss Markov Random Field (HL-MRF). HL-MRFs are like Markov Random fields (i.e., undirected probabilistic graphical models) with the exception that they are defined over continuous variables in the [0,1] interval [143]. In PSL, a rule's distance to satisfaction function is considered a hinge-loss potential.

Rules in PSL use predicates to express a relationship that takes a variable number of arguments. The predicate HasEvent/2, for example, takes two arguments. In the context of EDL, this predicate

represents whether an EDL variable (e.g., parachute deploy altitude) corresponds to a particular event (e.g., parachute deploy) in the EDL sequence, for example. When a predicate is combined with its arguments such as HasEvent(V,M), then it is called a ground atom. When constants are are assigned to an atom such as HasEvent('parachute deploy altitude', 'parachute deploy'), then it is considered a ground atom. In PSL, predicates can be either open or closed. If a predicate is defined as closed, it makes a closed-world assumption. In other words, it assumes that facts that are not known are false. If a predicate is open, on the other hand, PSL will attempt to infer unobserved atoms [143].

In PSL, a rule is a disjunctive clause of atoms. If a rule is unweighted, it is interpreted as a hard linear constraint. A weighted rule, on the other hand, is interpreted as a template for a hinge-loss potential [143]. The weighted logical rule:

$25 : IsCorrTo(V1, M)\&VarCorrsVar(V1, V2)\&(V1! = V2) \implies$
$IsCorrTo(V2, M)$

indicates that if a variable 1 is correlated to a metric, and that variable is correlated to a variable 2, then variable 2 is also likely correlated to the metric. The weight of 25 will induce the potential of a second variable being related to a metric given its association to another variable. When all possible substitutions of a rule are made, the rule is grounded and is considered a potential function or hard constraint if it is unweighted.

In PSL, atoms are mapped to soft truth values in what is called an interpretation. Interpretations are found by finding the probability distribution over interpretations that maximizes the likelihood of satisfying more ground rules. Given that PSL deals with Soft Logic, the framework uses Lukasiewicz Logic to determine the degree to which a grounded rule is satisfied [142]. Lukasiewicz logic extends Boolean logic by allowing propositions to take intermediate truth values in the [0,1] interval [144]. Given a truth value for each atom in a rule, the Lukasiewicz t-norm is used to estimate the truth values of the body (antecedent of a rule) and the head (consequent of a rule). For the aforementioned example, we can use the equation for the logical relaxation of a conjunction between atoms $IsCorrTo$ and $VarCorsVar$ in the rule body (Equation 1.4.3.1). In

this framework, a rule can only be satisfied if the truth value of the head is at least as large as the truth value of the body. If a rule is not satisfied, PSL uses distance to satisfaction to estimate how far a rule is from being satisfied:

$$T_{Luk}(a,b) = max\{0, I(a) + I(b)1\} \tag{1.7}$$

$$d_r(I) = max\{0, I(r_{body}) - I(r_{head})\} \tag{1.8}$$

In Equation 1.4.3.1 I is the interpretation (i.e., the mapping from atoms to soft truth values in a rule r).

After all groundings are done, each rule in PSL will be coupled to a distance to satisfaction. To identify the best assignments for truth values, PSL selects all ground rules *R* that mention atoms in *l* and defines a probability distribution over interpretations:

$$\begin{aligned} f(I) &= \frac{1}{Z} exp[-\sum_{i \in R} \lambda_r (d_r(I))^p] \\ Z &= \int_I exp[-\sum_{r \in R} \lambda_r (d_r(I))^p] \end{aligned} \tag{1.9}$$

In Equation 1.4.3.1, the probability density function over interpretations is a function of the weight of a PSL rule λ_r, the distance to target d_r, a normalization constant *Z*, and the loss function *p*. The loss function can take the values $p = \{1, 2\}$ and establishes how the model will trade off between competing assignments. Using $p = 1$ results takes more of a winner take all approach (i.e., linear). This choice will result in favoring interpretations that satisfy a rule entirely, which results in a larger distance from satisfaction for conflicting rules. A $p = 2$ (e.g., quadratic), on the other hand, is more flexible and will favor interpretations that satisfy all rules up to some degree [143, 145].

PSL performs the inference task using Most Probable Explanation (MPE), which means of maximizing the probability density function $f(I)$. Part of what makes PSL computationally efficient is attributed to the fact that the the resulting optimization problem is convex, specifically a Second Order Cone Program (SOCP), which can achieve linear scalability, as discussed in [143].

1.5 General Problem Statement

Most of the analysis done to study complex simulation results, such as EDL systems, relies on sensitivity analysis methods. However, most methods applied to complex dynamical systems are screening-based. In other words, they only provide a ranking of influence of parameters [50, 49], or fraction of contribution over system metrics [36, 35]. While these methods provide a good intuition and help identify driving features of the system, **they do not explain how the system behaves**. Furthermore, the literature review revealed that there are few tools that allow the users to automate aspects of the data processing task.

Intelligent Data Understanding is still an emerging field. Nevertheless, literature has shown that data mining and machine learning methods are useful for on-board event and anomaly detection, event classification, and climate modeling ([19, 9, 66]). However, they all **lack intelligence**. Therefore, identification of potentially interesting results from data mining relies on the subject-matter expert. This is a great limitation given that most algorithms, like association rule mining, generate hundreds and even thousands of rules with a large number of features [90]. The cognitive load imposed by this task makes it difficult for users to interpret all of the information, create internal representations and “make sense" of the information given. Consequently, we want to enable intelligent data understanding for EDL to help **reduce the volume of data** given to users and **provide high-value content information**. We propose using tools like **cognitive assistants** as a platform for data analysis and propose using knowledge graphs and statistical relational learning to help improve the comprehensibility, insightfulness and usefulness of discovered patterns.

1.6 Approach and Research Goals

This thesis develops a tool that serves as a platform for intelligent data understanding of EDL simulations. This tool automates basic tasks of the data processing portion of the analysis to help reduce cognitive load, and provides visualization of analysis in response to verbal or written requests. Given the interactive nature of the tool, it allows users to guide the data analysis process by user-specified constraints. This helps tailor the analysis to features in the design space the user is interested in. In addition, this thesis proposes incorporating a new method to help identify comprehensible, insightful and useful rules. The following paragraphs addresses each contribution.

- ***Domain Contribution*** - Aligned with the goals of the Intelligent Systems Program and NASA's Technology Roadmaps, this thesis proposes a **tool that can aid EDL experts** in the task of evaluating end-to-end vehicle performance. Due to the nature of cognitive assistants, we believe that this technology is an ideal platform for incorporating IDU technology that enables data search and discovery for distributed data holdings. We believe that IDU can benefit from flexibility- allowing users to intervene in the knowledge discovery process when necessary but performing **autonomous analysis when desired**. The proposed cognitive assistant helps automate common tasks conducted during the analysis of EDL simulations. This include, generating data products and summarizing results (e.g. scorecard, scatter plots histograms) and explaining simulation results. We propose two analysis forms for explaining simulation results: identifying driving features (sensitivity analysis) and discovering common features and behaviors that appear to be common among a region of interest in a dataset (association rule mining). This tool provides visualizations and makes use of information from multiple data sources to conduct analysis.

- ***Theoretical Contribution*** - We propose a new method for identifying comprehensible, insightful and useful association rules during data mining. To the best of our knowledge, the literature considers the number of items in a rule as the main (and in fact only) factor affecting comprehensibility of the rule. Therefore, the proposed approach serves as an initial

advancement for enhancing rule comprehensibility for a given domain. To help tackle comprehensibility, and the limitations discussed in 1.4.1, we propose integrating knowledge in the form of knowledge graphs and logical reasoning to help improve the comprehensibility, insightfulness, and usefulness of mined association rules. The knowledge graph is used to capture knowledge in the context of the domain or study. With this information, statistical relational learning is used to infer whether pairs of items in a rule are related in some way. We then derive a measure of consistency between a rule and the knowledge graph based on those probabilities. We hypothesize that rules that are more consistent with the knowledge graph will be more comprehensible than rules with lower consistency with the knowledge graph, and therefore more useful.

- ***Methodological Contribution*** Aligned with our domain contribution, we want to demonstrate a framework for integrating data from distributed data holdings into one framework that can be used for analysis of a mission. We focus on EDL system performance but the vision is to expand tool capabilities so that it is useful for studying all aspects of a mission (e.g.,architecture development and operations). Furthermore, most elements incorporated for the tool and methods used for identifying insightful and useful information are more case-specific. However, we believe that given current trends in model-based systems engineering, knowledge graphs can make use of MBSE artifacts such as lessons learned databases and SysML models generated during the mission development process. Knowledge graphs and along with reasoning capabilities can be very powerful tools to help experts perform end-to-end analysis of EDL systems.

1.7 Structure

Chapter 2 proposes using a cognitive assistant as a platform for EDL. The first part of the chapter describes Daphne, the cognitive assistant adapted for EDL analysis. We describe the different layers in Daphne and describe the user interface. We also discuss the algorithms used to perform analysis of EDL simulations. Furthermore, we discuss how Daphne can be used to interactively

study a dataset. We then present a case study to demonstrate Daphne's capabilities.

Chapter 3 describes a framework for reasoning about the relationships between EDL metrics and parameters and identifying new ones. The first part of this chapter describes the approach used for constructing a knowledge graph. The graph represents knowledge about EDL performance metrics and parameters. We then describe how we integrated a SRL framework to infer relationships between entities in our knowledge base.

Chapter 4 describes our approach for using inferences made by the SRL framework to help identify interesting rules. This chapter first describes what inferences about association are extracted from the SRL framework. We then describe how we compare statistical measures derived from data mining to inferences made by the SRL framework to help identify interesting rules. This chapter includes a human survey experiment conducted to study three hypothesis posed about rule interestingness in our framework.

Chapter 5 provides a summary of the thesis, and its contributions. We also discuss the limitations with our approach and discuss future research opportunities.

2. A COGNITIVE ASSISTANT AS A PLATFORM FOR EDL ANALYSIS*

2.1 Introduction

Cognitive assistants have been explored as a viable decision-support tool given that they can provide experts with high-quality recommendations that can help enable informed decisions [146, 12]. Cognitive assistants are a viable platform for IDU given that they provide one capability IDU's in isolation lack: user interactivity [147]. CAs can store domain-knowledge in the form of databases and rules, for example, that can be used to provide experts with responses to specific queries using a teacher-apprentice approach [73]. In its simplest form, knowledge can be technical information about spacecraft instruments, or historical information about spacecraft missions [74]. In the work described in [12], rules of thumb, or heuristics, are used to evaluate spacecraft designs. For example, rules of thumb can be used to determine which orbits and time of the day are best to operate particular spacecraft instruments.

Asides from obtaining knowledge available, cognitive assistants can exploit AI and data analysis techniques that conform the essence of IDU: to quantify the probabilities and states of a particular decision [73]. A CA can be useful for identifying features of interest in a design; analyzing and communicating the findings to team members; providing historical or contextual information; and more generally reducing cognitive load on the team members [74]. In the context of EDL, CAs can help experts identify anomalies, features of interest, and extract knowledge that could potentially not be attainable by manually examining a simulation data set of the architecture under evaluation, for example. To exploit the interactivity features CAs are characterized for, EDL teams can specify to the assistant aspects of the data they are interested in and what type of analysis to conduct.

Given the opportunities CAs provide, the aerospace domain has explored several ways CAs can be used as decision support tools. Initial work focused on exploring CAs for providing situa-

*Parts of this chapter have been adapted from "A Cognitive Assistant for Entry, Descent, and Landing Architecture Analysis" (2019)[2] and "Interactive Explanation of Entry, Descent, and Landing Simulations" (2020)[1] by Santini De Leon, S., Selva, D., and Way, D. with permission from IEEE and AIAA, respectively.

tional awareness and subsequent operational decisions making tasks. Tools like COGAS [75], for example, were developed to support, supports NAVY ships in air target identification. Other tools like CAMA, on the other hand, were developed to aid pilots operating in military environments.

For space missions, preliminary work has been done for using virtual assistants for supporting astronaut crew during missions beyond Low Earth Orbit (LEO), especially in off-nominal conditions when there is a long communication delay between Earth and the space vehicle [78]. The most recent work described in [13] uses a CA to trouble shoot spacecraft anomalies, characterizing anomalies, identifying root causes of the anomaly and recommend a course of action.

Up to the present day, these technologies have been commonly employed to support mission operations, but their use in performance analysis during mission development has not been explored. However, recently some work has been done on using CAs for system architecting problems for Earth observing (EO) satellite systems. The work described in [148, 79, 12] uses Daphne, a cognitive assistant, to help experts in the architecture analysis process by providing relevant information, advice, and feedback that address strengths and weaknesses of a particular design. These capabilities help minimize the cognitive load on experts by reducing the need to manually search through multiple sources of information.

Largely motivated by this work, we propose using Daphne as a tool that can help experts in the data processing and performance assessment tasks of EDL analysis process depicted in Figure 1.1. We believe that Daphne can help experts during data processing by: 1) automating tasks and summarizing data and 2) conducting analysis to investigate performance and identify driving features.

In this chapter we provide an overview of Daphne, a cognitive assistant, and how we have adapted the system for the analysis of EDL simulations. We also discuss the different layers of Daphne-EDL: front end, backends, and data sources. With Daphne-EDL, experts can make requests using verbal or written natural language and Daphne provides data products and/or responses to specific queries through a web interface. We also describe how Daphne uses different analysis capabilities to help experts with investigating performance failures and identify driving

features, as depicted in Figure 1.1.

2.2 Overview of Daphne

Originally, Daphne was developed for Earth Observing (EO) architecture analysis of satellite systems [79]. Daphne's main goal was to provide experts with relevant information, advice, and feedback that address the strengths and weaknesses of a design [74, 148]. As depicted in Figure 2.1, Daphne's architecture consists of four layers. The first layer, the front end, serves as a platform for the user to interact and communicate with Daphne. Requests are made either in natural language or through a web visual interface. The second layer is referred to as the 'Daphne brain', which acts as the front-end server. This layer forwards requests to the respective backend modules in the third layer. Depending on the form of the request, these are directed using HTTP or on Websockets (e.g. from buttons on the web interface or natural language). The third layer contains all of the roles. Roles act as software snippets that use available microservices [12] in the data sources available in the fourth layer. For example, a question such as "what missions were launched in 2018?" would be classified as a question for the Historian role and would extract a response from the historical database in the data sources. Additional details about Daphne's architecture and its skills can be found in [12].

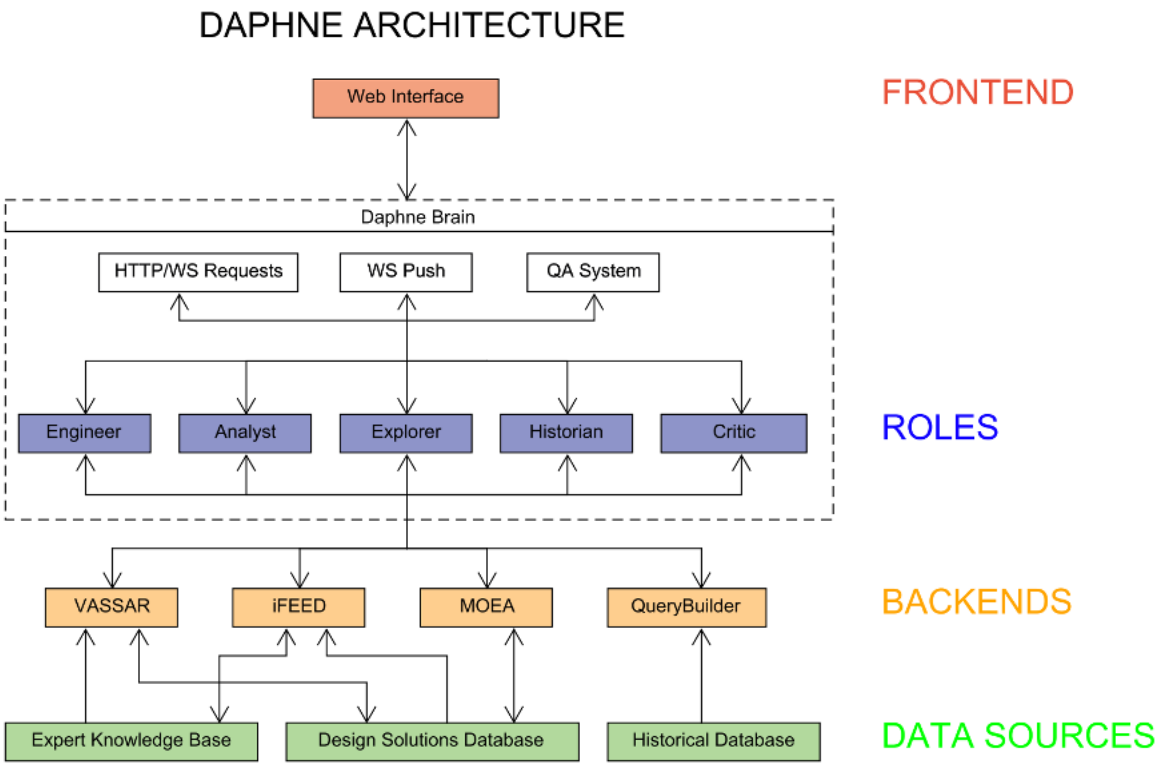


Figure 2.1: Daphne-EO architecture (Reprinted from [2])

2.3 Tailoring Daphne for EDL

This section describes how we established what capabilities are necessary to help experts during the analysis of EDL simulations. We also discuss the backends and data sources developed to help address the needs of EDL experts.

2.3.1 Survey of Information and Capabilities of Interest to EDL Experts

Establishing what type of information subject matter experts deem relevant during the EDL analysis process was done by two lines of investigation: literature reviews and discussions with our NASA collaborator, an EDL expert. Knowledge and information extracted during this process were used to develop a set of questions/commands Daphne should respond in order to assist experts.

A survey of analysis techniques used for studying outputs of Monte Carlo simulations [149, 150], and the EDL analysis process discussed in 1.2, suggest that subject matter experts are often

predisposed to acquire a sense of the statistics of variables of interest and their sensitivities. This is often done by examining histograms of key variables and metrics. Experts are also inclined to identify stressing cases (e.g. flags and out of spec) for further investigation. In the process of identifying features of interest, the analysis is driven by comparing stressing cases to nominal cases in an attempt to identify commonalities and differences between them. During this process, experts make use of visual aids (e.g. variable plots and statistical plots) and conduct extensive search of the dataset for identifying distinctive features that explain the system's behavior.

From the frequent discussions held with an expert in EDL end-to-end simulation analysis, we generated a set of preliminary question types (QT) and actions (AC) that emerge during the analysis process discussed. Some of these include:

- QT: What are the statistics (e.g. mean, min, max, 99of parameter X ?
- QT: Is parameter X correlated with parameter Y ?
- QT: How is the result from mission/simulation A different from mission/simulation B ?
- QT: Why is case X failing ?
- QT: What do cases A to C have in common ?
- AC: Find the value of parameter X for a mission/simulation.
- AC: Plot statistics (e.g. histogram, quad-quad plot, CDF).
- AC: Plot parameter X vs. parameter Y.
- AC: Identify a stressing case.
- AC: Plot the evolution of a parameter over time, possibly across missions.

2.3.2 Use Cases for Daphne-EDL

A survey of the EDL analysis process in 1.2 and discussions with an EDL expert helped identify two use cases in which Daphne can be of aid to experts: 1) by reducing the cognitive load and the manual labor of having to search through multiple sources of information and 2) by providing analysis and insights about a dataset.

The first item mentioned is relevant for individual and collective analysis of EDL architectures. For example, due to the human-like nature of CAs, Daphne could be incorporated into a collective

setting where experts discuss the results of metrics from multiple simulations (e.g. different landing sites) and assess system performance of each. At the moment, this task requires that experts search for the relevant simulation data set and extract the values of the metric(s) they are interested in. In some cases, additional calculations are required. This process is repeated for each simulation. Hence, we envision that Daphne could do this for the user. By means of natural language, the subject matter expert can ask Daphne for the results she/he is interested in without going through the manual labor of searching and loading each data set and calculating the metric of interest.

Along these lines, we envision that Daphne can analyze and identify critical information in a simulation and communicate the findings to the user. As seen 2.3.1, some requests and/or questions go beyond extracting information that is explicitly available and require some form of analysis method. Some form of analysis capability, for example, can help identify critical parameters that drive a particular architecture's behavior as well as extract and compare features of interest across missions or simulations.

2.4 Daphne-EDL

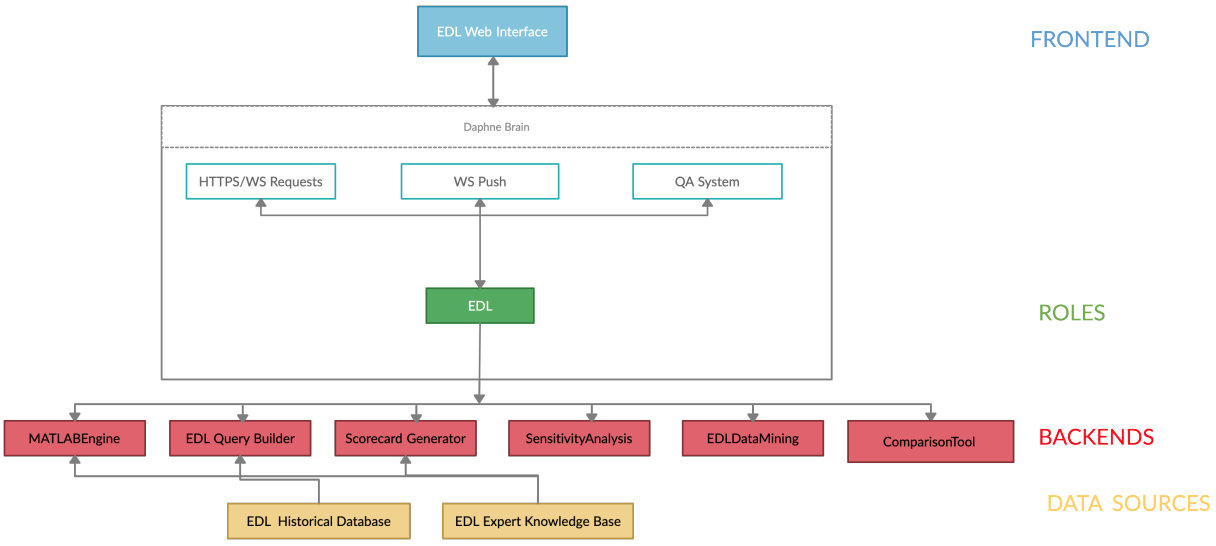


Figure 2.2: Daphne-EDL architecture (Reprinted from [1]).

Figure 2.2, shows Daphne-EDL's implementation and the respective front-ends, backends, and data sources that belong to the EDL role. In the current implementation of Daphne, all EDL-related requests made in the user interface are directed to the Daphne server through HTTP/Websockets, a bi-directional line of communication established between the client and the Daphne brain. EDL-related queries or commands are processed by the Sentence Processor's CNN and classified as an EDL role. Requests are then processed by the EDL query builder. The Query Builder uses JSON file templates to identify the type of query, extract the features of interest in the query (e.g., mission name, parameter name), and direct the query to the respective executable functions and data sources used to generate the response. A response is then created and directed back to the client. Mechanisms in place for how requests are handled through HTTP/Websockets, is detailed in [12].

2.4.1 Data Sources

Daphne-EDL has two primary data sources: an EDL historical database and an EDL Expert Knowledge Base.

2.4.1.1 Historical Database

The EDL historical database was created to provide subject matter experts with information about previous EDL missions. However, unlike for Earth observing satellite missions, there is no online database of previous EDL mars missions to support the coordination of EDL architecture analysis for future planetary missions. Furthermore, creating a database in the EDL domain is challenging due to the number of variables involved in these complex multi-body vehicle systems. Consequently, we established two requirements for the implementation of the EDL database. First, the database shall contain descriptive information about mechanisms employed during the EDL sequence of each mission. Some of these are, for example, type of entry (direct/orbit), entry lift control (center of-mass offset/no offset), entry guidance (unguided/guided), and descent attitude control (RCS roll rate/none), among others. Such information can provide experts with contextual data when examining metrics of different architectures. And second, the database should contain

information that is shared across EDL architectures. This consideration is driven by the fact that limited information is available from past missions and that comparison across missions can only be achieved if different vehicle systems can be described using common performance metrics. For example, although different missions have employed different mechanisms for entry lift control, common performance metrics include peak deceleration and peak heat rate, among others. Figure 2.3 shows the schema used for the database creation.

The EDL database was implemented as an object-relational database management system (ORDMS). Such database provides a bridge between relational and object-oriented paradigms. The standard selected for managing information in the database is the Structured Query Language (SQL) through the PostgreSQL software.

The resulting database was built in Python using the SQLAlchemy toolkit and served as an interface between the database and PostgreSQL. In the current model, one-to-many relationships were incorporated to connect fields in a given class (i.e., table) to another table. In other words, the current model uses hierarchical relationships. However, considering that not all relationships are of this nature (e.g., different missions or segments can share the same attitude control mechanism), many-to-many relationships can be discovered. Thus, many-to-many relationships can be incorporated to account for additional complexity.

2.4.1.2 EDL Expert Knowledge Base

The Scorecard discussed in Section 1.2 was used as the expert knowledge base for the EDL skill given that it provides a standardized knowledge repository that is shared among all EDL groups. This document is a type of summary report that provides natural language-form descriptions and mathematical models Daphne can make use of for analysis and calculations. For example, the metric described as fuel consumption contains a fixed number of entries, each containing the flag and out of spec values, units, description of the metric, the POST2 results, and the calculation required to obtain the metric. value. In addition, it contains thresholds and conditions that can be translated into rules by employing "if-then" statements and quickly identify mission-specific requirements that are not satisfied - or close. These thresholds are evaluated when an expert requests which

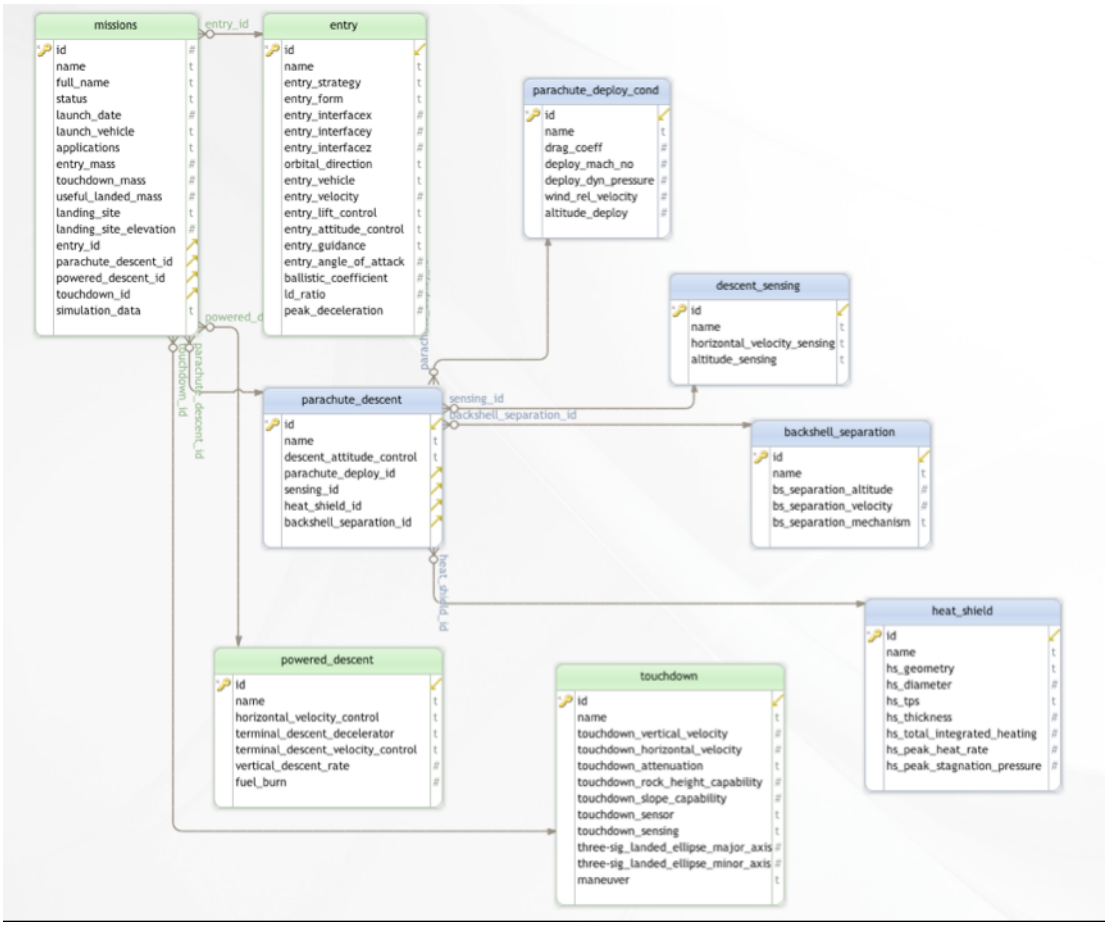


Figure 2.3: EDL historical database.

metrics in the scorecard do not satisfy system requirements. A basic representation of a scorecard used to assess the performance of the EDL pre-flight simulation predictions based on reconstructed flight data from MSL is available in Reference [26].

2.4.2 Back Ends

The backends in Daphne communicate with the data sources to extract information- when needed. Some of the backends, however, only perform data analysis or invoke external modules.

2.4.2.1 MATLAB Engine

The MATLAB Engine in Daphne is used to perform EDL calculations when requested. Most EDL scripts and models are defined using the MATLAB language, hence, we communicate with

the engine when any EDL data estimate is made.

2.4.2.2 EDL Query Builder

The query builder is in charge of translating a natural language request into SQL queries, which in turn, extract the parameter of interest. As discussed in [12], queries are classified by the Natural Language Processing (NLP) module. With the current implementation, Daphne performs question classification by type and searches for the information requested in the query. JSON file templates available in Daphne specify the name/value pairs required to respond to a particular query and are used to search for the features requested. For the query "What as the entry velocity for MSL?", we want to extract two features: mission name and parameter. Feature extractors match the sentences to lists of known values for the requested information. Daphne's implementation of the statistical model provided by Sellers et al., algorithm accounts for mistakes (e.g. typos) in the users request. In this case, features are extracted from the historical database in the query section of the template. We will discuss the historical database for EDL in the following subsections. Finally, after features are extracted, results are embedded into the template response. The response is then returned to the user at the front end through voice or through the visual response template.

Based on the question types and actions EDL experts often conduct, as discussed in 2.3.1, we trained Daphne to process four types of queries and commands shown in Table 2.1: 1)file loading and scorecard generation commands, 2)summarization queries, 3) visualization commands, and 4) parameter value extraction queries. Given that Daphne's context stores the information of the dataset under study, experts tell ask Daphne "from the scorecard, what metrics are flagged?" as opposed to having to re-specify the dataset of interest.

2.4.2.3 Scorecard Generator

The scorecard generator is in charge of generating a scorecard for the dataset under study in the event it has not been created yet. The scorecard creator invokes the executable file and generates data products that are stored in Daphne's database.

File commands
For {mission}, load {file}.
Generate a scorecard for {file}.
Generate a scorecard for this matfile.
Summarization Queries
From the {scorecard file}, what metrics are flagged?
From the current scorecard, what metrics are flagged?
From the {scorecard file}, what metrics are out of spec?
From the current scorecard, what metrics are out of spec?
Visualization commands
From the {matfile}, plot {parameter 1} vs {parameter 2}.
From the current matfile, plot {parameter 1} vs {parameter 2}.
For {mission}, calculate the statistics for {parameter} in {file}.
Calculate the statistics for {parameter}.
Parameter Value Extraction
From {scorecard file}, what are the POST results for the {metric name} metric?
From the current scorecard, what are the POST results for the {metric name} metric?
From the {matfile}, calculate {metric name}.
From the current matfile, calculate {metric name}.
For {mission}, calculate the statistics for {parameter} in {file}.
Calculate the statistics for {parameter}.

Table 2.1: Queries available in Daphne-EDL.

2.4.2.4 Sensitivity Analysis

Sensitivity analysis was implemented to aid experts in the task of identifying driving features in a dataset. Such capability could be helpful for experts to get an understanding of the workings of the EDL models. As discussed in 1.2.3, variance-based and surrogate model-based sensitivity methods are computationally expensive for complex models, such as EDL. This drawback is largely attributed to the number of input parameters that are to evaluate EDL trajectories. Consequently, we have opted for density-based sensitivity analysis measures, given that EDL simulations produce sufficient runs to generate accurate distributions.

The most common methods for model inspection include but are not limited to, partial dependence plots and sensitivity analysis methods [151]. Partial dependence plots are a visual explanation method that depict the marginal effect of two variables on a particular outcome [152]. However, this method relies on the assumption of independence, which does not necessarily hold for EDL simulations. Consequently, in this work we have opted for using non-parametric sensitivity analysis approaches. In particular, we selected the Kolmogorov-Smirnov (K-S) d-statistic test. In short, the K-S test is a non-parametric goodness-of-fit statistical test that is used to verify whether two sets of observations belong to the same distribution. The K-S test estimates the maximum distance between two cumulative distribution functions (CDFs) conditioned on user-specified criteria. For example, given a simulation with 8,000 cases, one scenario is to examine what drives velocity navigation errors. User-specified criteria could be: cases that fall outside of the requirement ellipse size (Fail) against those that fall inside the ellipse (Pass).

$$F1_{X|fail} = P(X < x|fail), F2_{X|pass} = P(X < x|pass) \tag{2.1}$$

If the d-statistic between pass and fail cases is large, then this parameter is influential. Indeed, sensitivity by means of the K-S test can be assessed by ranking of influential variables as a function of the d-statistic (Equation 2.2), which is the maximum distance between the two sample cumulative distribution functions and by means of hypothesis testing (p-value). In the K-S test, a small

distance ($D \approx 0$), is an indicator that both samples come from the same distribution. In the context of sensitivity analysis this indicates a lack of influence. For $D > 0$, one can perform a statistical test, as in other methods, but using the d-statistic to obtain the p-value. The hypotheses for the test are that the two samples come from the same distribution (null hypothesis, H_0), or that they do not come from the same distribution (alternate hypothesis, H_a). Consequently, a large d-statistic and low p-value are indicators of parameter influence. Figure 2.4 presents two examples. examples of influential parameters and non-influential parameters in the context of the K-S test. This example illustrates the effect of two parameters on touchdown velocity conditioned on pass/fail criteria. Both plots in the figure illustrate the conditional probabilities of both classes. The plot on the left illustrates a parameter that is influential given that the maximum vertical distance (d-statistic) between both distributions is large. The plot on the right, however, illustrates an example on a non-influential parameter.

$$D = sup_x|F1(x) - F2(x)| \tag{2.2}$$

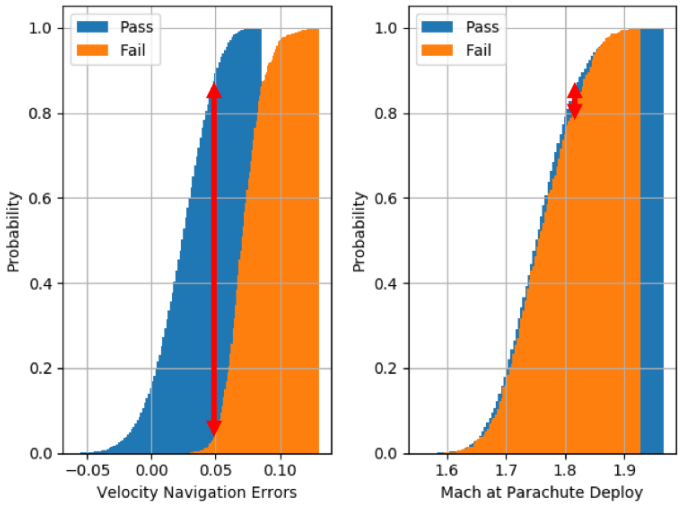


Figure 2.4: Example of influential and non-influential parameters using the K-S Test.

2.4.2.5 EDL Data Mining

One of the main goals of using a cognitive assistant was to provide experts with explanations of simulation outcomes. For example, an outcome explanation could be explaining why the 99%-tile of touchdown velocity is high. Common ways to explain outcomes is by using decision trees, and decision rules [151]. As discussed in Section 1.4, Decision Trees are very popular, however, interpretability decreases as the size of the tree increases. Given that this method follows a greedy approach, it is very sensitive to any changes in the data, making it unstable. Decision rules make use of *if-then* statements to map observations to outcomes. They take the form: if *condition 1* and *condition 2* and ... , then *outcome*, where *condition 1*, *condition 2*, and *outcome* are all binary features. In EDL, an example of a rule could be:

$$\{entryMass > 1,000kg, efpa < -15\,\mathrm{deg}\} \rightarrow \{machPD > 2.2\}$$

In natural language, this rule means: *if* the entry mass of the vehicle exceeds 1,000 kg and the entry flight path angle is steeper than -15 deg, *then* the threshold for the Mach number at parachute deploy (2.2) will (likely) be exceeded. Given the nature of cognitive assistants, we believe that these rules resemble more how experts reason- making it a suitable algorithm for explaining outcomes in a simulation [20, 21].

A common method for mining rules of this form is Association Rule Mining (ARM). In essence, given a set of binary features, ARM aims to find regions of high joint probability density and generates rules based on those. The most common example is a transaction list from a grocery store. Bread, butter and eggs are commonly found together on shopping lists. Therefore, ARM will generate rules such as $\{bread, eggs\} \rightarrow butter$. An example of a transaction set *T* in the context of EDL and a list of itemsets is depicted in Table 2.2. Assuming $X \rightarrow$Y is an association rule where X and Y are itemsets, the effectiveness of the rule can be measured using three metrics: support, confidence, and lift. For example, the transaction depicted in the table contains the feature $X = \{mass > 1,000kg, efpa < -15\,\mathrm{deg}\}$. The support of this feature is the proportion of the

occurrences of X among all items in the transaction. In other words, it answers the question "how much does the historical data support this statement?"

$$supp(X) = \frac{|\{t \in T; X \subseteq t\}|}{|T|} \tag{2.3}$$

Here, t is a transaction, T is the set of transactions, X is a feature (i.e., an itemset) and $|\{t \in T; X \subseteq t\}|$ is the number of transactions in T that contain the itemset X. A high value of support is an indicator that X is found often in the dataset. For the example illustrated above ($X = \{mass > 1,000kg, efpa < -15\,\mathrm{deg}\}$), the support of X in Table 2.2 is $2/4 = 0.5$. Confidence, on the other hand, quantifies the conditional probability of the association rule. In other words, it quantifies how likely is it that the Mach threshold (Y) was exceeded when mass was over 1 ton and the entry flight path angle was steeper than -15 deg (X). In this example, the confidence for this example would be $2/2 = 1$. A high confidence is an indicator of high strength of the association rule.

$$conf(X \rightarrow Y) = \frac{supp(X \cap Y)}{supp(X)} \tag{2.4}$$

Finally, lift measures the distance between $P(Y|X)$ and $P(Y)$. This measure determines the extent to which X and Y are dependent [153]. In this example, we have a support of $2/(2x3) = 1/3$. A lift that approximates 1 is an indicator of independence between X and Y. A lift higher than 1 is an indicator of dependency between the antecedent and the consequent.

$$lift(X \rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)supp(Y)} \tag{2.5}$$

The most popular algorithms for mining frequent item sets and association rules are Apriori and FP-growth [87, 154]. Both methods mine for frequent itemsets and generate rules in the form $X \rightarrow Y$. Rules that satisfy the threshold for minimum confidence and support are kept. Rules with high support are considered statistically significant whereas rules with high confidence are considered "strong" rules.

Despite the popularity of these algorithms, there are numerous disadvantages. Namely, despite

the application of the greedy Apriori rule, these methods find rules by brute force (e.g., they require many database scans), making them very slow in general. Furthermore, the finding of rules is dependent on the threshold values of support and confidence set by the user. Low support and confidence thresholds may find too many irrelevant rules and high thresholds might lead to no rules at all. To overcome these disadvantages, other search algorithms such as evolutionary algorithm (EAs) have also been used for rule learning. This way, the search of rules is framed as an optimization problem. In this paper the task of rule mining is done by means of a multi-objective evolutionary algorithm developed by Bang et al. [90]. This algorithm uses the ϵ-MOEA algorithm for rule mining framed as a binary classification problem. In this method, rules are encoded as trees, a method typically employed in Genetic Programming (GP). The fitness function for the EA employed is multi-dimensional and consists of two measures of confidence and complexity. Confidence and support have already been defined. In the context of classification algorithms, the two measures of confidence are known as precision and recall, whereas complexity is the number of literals in the antecedent of the rule. A rule with high complexity could result in difficulty in interpreting the rule. Hence, this algorithm optimizes the search for low-complexity rules. This algorithm was employed for the outcome explanation task.

Table 2.2: Example of an EDL itemset.

ID	Items
1	$\{entryMass > 1,000kg, efpa < -15deg, machPD > 2.2\}$
2	$\{entryMass < 1,000kg, efpa < -15deg, entryVel > 6.5km/s, machPD > 2.2\}$
3	$\{entryMass < 1,000kg, efpa > -15deg, machPD < 2.2\}$
4	$\{entryMass > 1,000kg, efpa < -15deg, entryVel < 6.5km/s, machPD > 2.2\}$

2.4.2.6 Comparison Tool

The EDL dataset comparison tool in Daphne-EDL is used to test whether the data between two simulations is significantly difference. Often times, EDL experts compare scorecards between simulation cases with the goal of identifying significant changes in EDL metrics. Often times,

assessing whether a value is truly significantly different or not is difficult to attest given that, in some instances, small changes in metric values are commonly due to statistical noise.

Given two user-selected datasets, the comparison tool performs a K-S test to verify whether the shapes of the distribution of a metric between two simulations differs. The output provided to the user is a Table that lists scorecard metrics, the results in both datasets, the D-statistic, and the p-value of the statistical test. Results are sorted by the D-statistic in descending order so that the most significant are highlighted first.

2.4.3 Front End

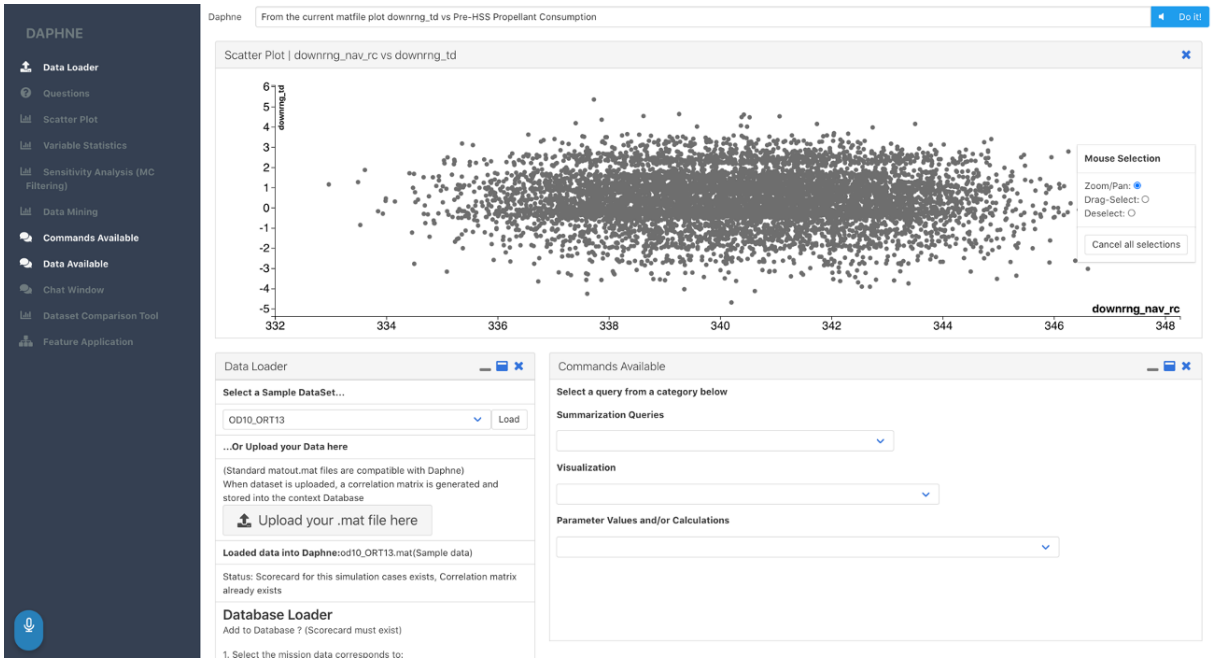


Figure 2.5: Daphne web interface.

Daphne-EDL uses a web interface as a front-end, shown in Figure 2.5. Here, experts can interact with Daphne using verbal and/or written requests. Experts can request Daphne to load a dataset, generate a scorecard, provide a summary of results (e.g., list of metrics flagged or out of spec), visualization of results (scatter plots or statistics), and can ask Daphne to extract or compute

the value of a variable or metric of interest. To help users, Daphne-EDL has a component with queries available. Figure 2.6 shows a screenshot of this component. Upon loading a dataset, users can also see what metrics are available in the scorecard and what variables are in the simulation file.

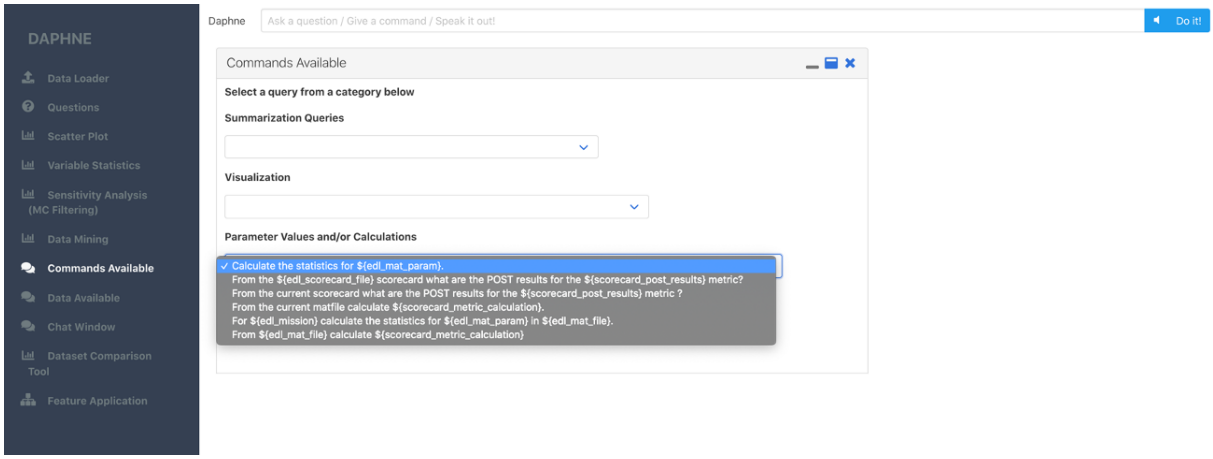


Figure 2.6: Commands available helper.

For analysis, the interface has a designated component for sensitivity analysis, data mining, and dataset comparison. Given that specifying criteria for these analysis can vary from user to user significantly, users can select the options and/or constraints for the analysis in each module. For sensitivity analysis, Daphne provides a ranking of influential parameters based on the D-statistic, as shown in Figure 2.7.

2.5 Using Daphne for Explaining EDL Simulations

As discussed in Section 1.2, everyday tasks in the EDL simulation analysis process involve generating summary reports (such as scorecards), examining results, identifying metrics or cases of interest, investigating performance failures, and identifying driving features. Our goal with Daphne-EDL is to help in this process by automating common data processing tasks and summarize data. Furthermore, we also wanted an interactive framework that analyzes performance and

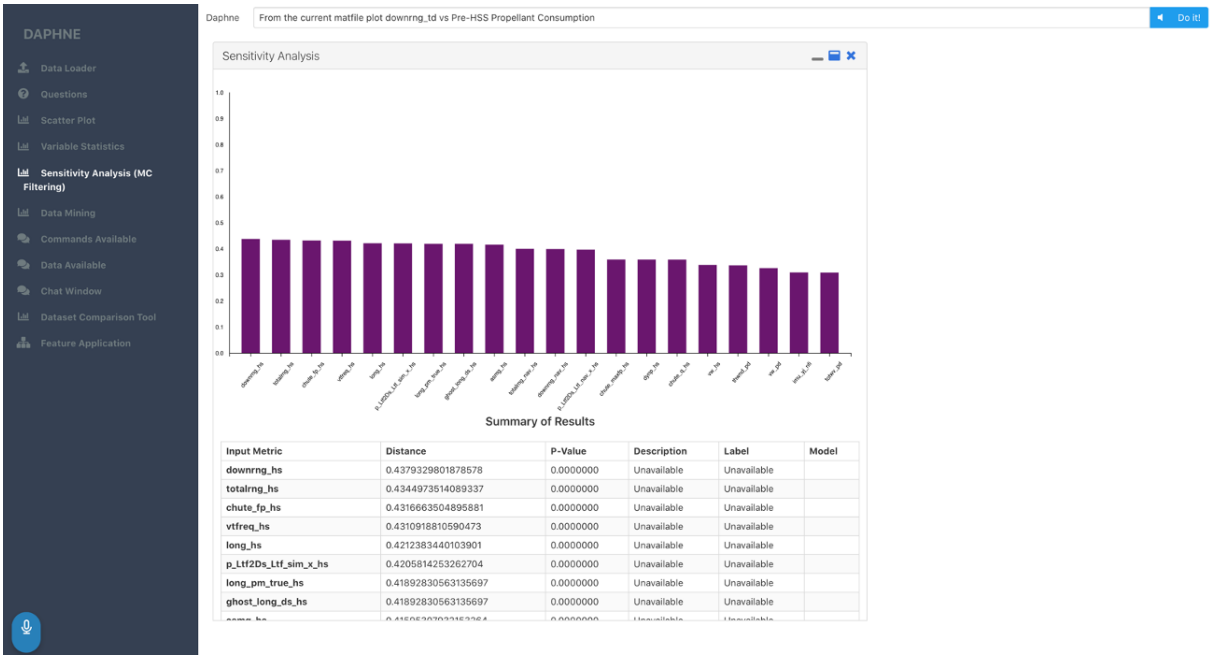


Figure 2.7: Output from sensitivity analysis (Adapted from [1]).

helps experts identify driving features. Given the nature of CAs, we also wanted to provide a platform that allows users to incorporate knowledge in data constraints to tailor the analysis to their interests. In this section, we discuss how Daphne can be used interactively for the analysis of EDL datasets.

2.5.1 Interactive Strategy

Figure 2.8 shows the framework for interactively communicating with Daphne. As seen in the figure, communication between users and Daphne is bidirectional. Following the typical approach experts conduct to study simulation outputs, as a first step, experts can request Daphne to generate a scorecard for the new simulation. As a second step, experts can request Daphne to provide them with simulation results. For example, a common task done manually by experts is to examine the values of critical metrics (e.g., probability of success), their statistics, and whether a metric is flagged or out of spec. With Daphne, the user can simply ask: “What are the POST-II Results for peak deceleration?" or “what are the statistics for peak deceleration?" Another way users can request results is by using the data comparison component in the interface. There, experts can

select two datasets to see whether any metric value has changed significantly. Once results are provided by Daphne, as a third step, experts can visually examine the results of the variables or metrics requested. This includes, inspecting the statistics provided in the chat, for example, or by visualizing the histogram that accompanies the results. If the user executed a dataset comparison, they can examine the table of results. This table provides a ranking of variables whose distribution differs the most between two datasets.

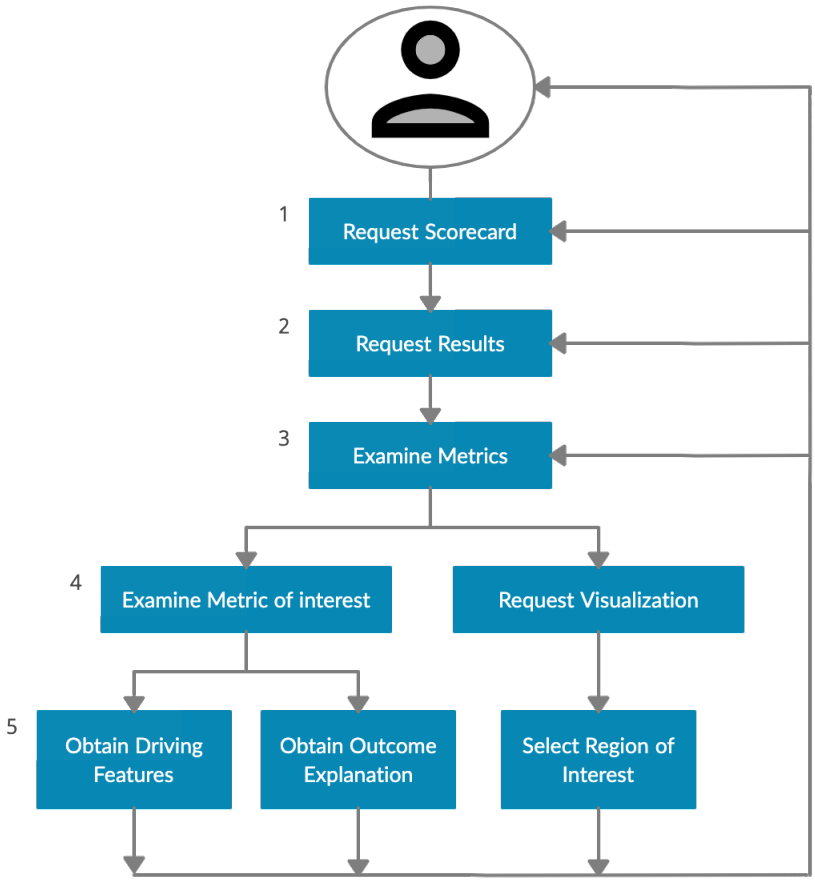


Figure 2.8: Interactive framework (Reprinted from [1]).

Given the nature of cognitive assistants, the user can continue to request additional results, as in step 2, and examine these results. After examining results, the path for identifying cases of interest

or identifying performance drivers can go two ways. One analysis path is for the expert to select one metric of interest and conduct sensitivity analysis to identify the most influential parameters driving their metric of interest. Alternately, the user can employ ARM to obtain explanations for a particular range of values. For example, data mining can be conducted to find patterns that drive high touchdown velocities, for example. Another approach is to request a scatter plot of the metric of interest against any other variable or metric. One example is to plot downrange and crossrange at touchdown to and select points outside of the landed ellipse. With this method, Daphne-EDL conducts ARM on a subset of the design space and generates explanations for those outcomes. With either path taken in step 5, the user will obtain a type of explanation and in some cases, they might choose to further examine the results of particular metrics. Flexibility is main advantages of using a cognitive assistant given that, experts can follow the steps they see fit to investigate simulation outputs. With Daphne-EDL experts can conduct analysis and then request additional information from the simulation understudy. They can also request Daphne-EDL to load a new simulation dataset for further analysis or comparison. In the following section, we present a case study where we demonstrate Daphne's capabilities for explaining simulation outputs.

2.5.2 Case Study

To demonstrate Daphne's capabilities, we selected a Mars 2020 dataset used in a Terminal Descent Sensor (TDS) failure study. This study was aimed to examine the effect of a single radar failure on critical EDL metrics [3]. The TDS, in short, is a doppler radar that has six line of sight altimeter/velocimeter narrow beam antennas that collect instantaneous altitude and velocity measurements after backshell separation and throughout the powered descent segment. Figure 2.9 shows a bottom view of all six beams placed on the terminal descent stage. One of the beams (beam 1) is also referred to as the nadir-pointing beam given that it points straight down to the ground. This beam has the most visibility of all six. Beams 2,4, and 6, are the "canted" beams whereas beams 3 and 5 are the "headlight beams." The canted beams are three evenly spaced beams that form an equilateral triangle. These beams are also canted 20 degrees. The headlight beams, on the other hand, point in front of the rover and are canted 50 degrees from the nadir angle and 20

degrees off the spacecraft forward axis.

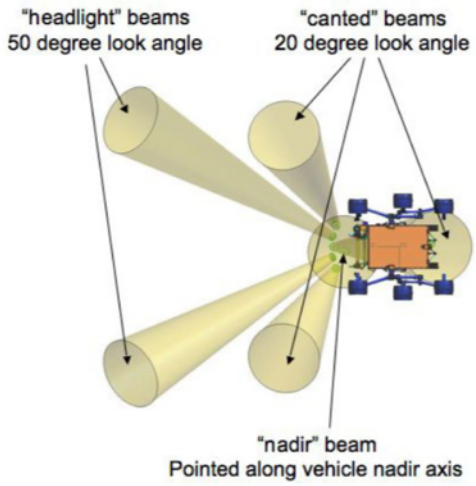


Figure 2.9: TDS beam layout (Reprinted from [3]).

For the purpose of this case study, we will demonstrate Daphne's capabilities using two datasets used in the aforementioned study. The first dataset is a simulation that mimics a nadir beam failure. The second datastet mimics a nadir beam failure.

In the following subsections we will present the interactive approach discussed in 2.5.1. In the context of this case study, the objective is to discover parameters and metrics that are affected by a beam failure and their impact on critical EDL metrics. More broadly, the objective of the case study is to obtain explanations that could lead experts to arrive to similar conclusions in Reference [3] whilst removing the steps of manually searching, plotting, and examining individual parameters that change and affect key metrics. Given that horizontal and vertical velocity were affected the most by a beam failure, this paper will focus on examining influential factors for the increased values of these metrics whilst knowing that in each simulation a particular beam does not operate.

2.5.3 Step1: Requesting Information from a Simulation

The first step required by users in Daphne-EDL is to load the simulation data they wish to analyze. Otherwise, Daphne-EDL will continue to use the latest used in its working memory. This task is evidently required to conduct analysis, however, in Daphne-EDL this task is automated by means of written or verbal communication. As discussed in Section 1.2, experts must manually load, configure the dataset, and generate a Scorecard so that they can proceed with studying performance metrics. Here, experts can simply speak or write a query such as: *Daphne, for Mars 2020, load the TDS1-beam dataset*. The corresponding follow up question, would be: *Generate a Scorecard for this file*. Otherwise, the user can manually select the dataset of interest using the Data Loader, as depicted in Figure 2.10. Once tasks are complete, or if the solicited data is available, Daphne displays a status message indicating that the files were created or already existed.

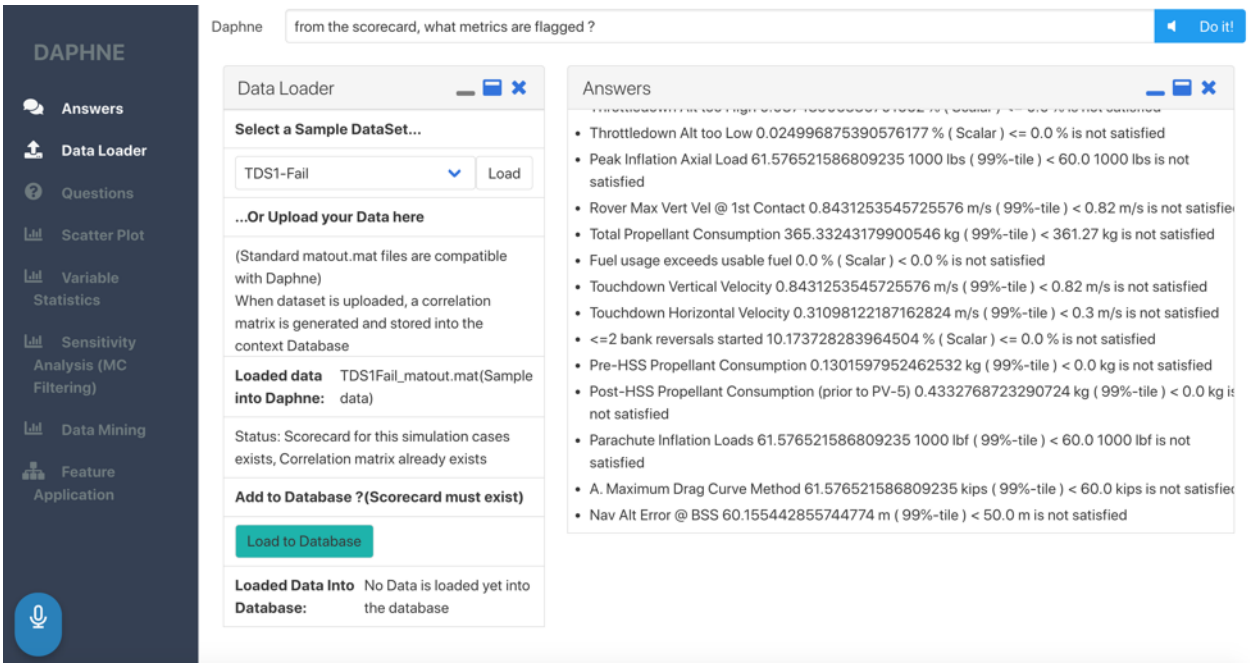


Figure 2.10: Data loader and data summarization in Daphne (Reprinted from [1]).

[hbt!]

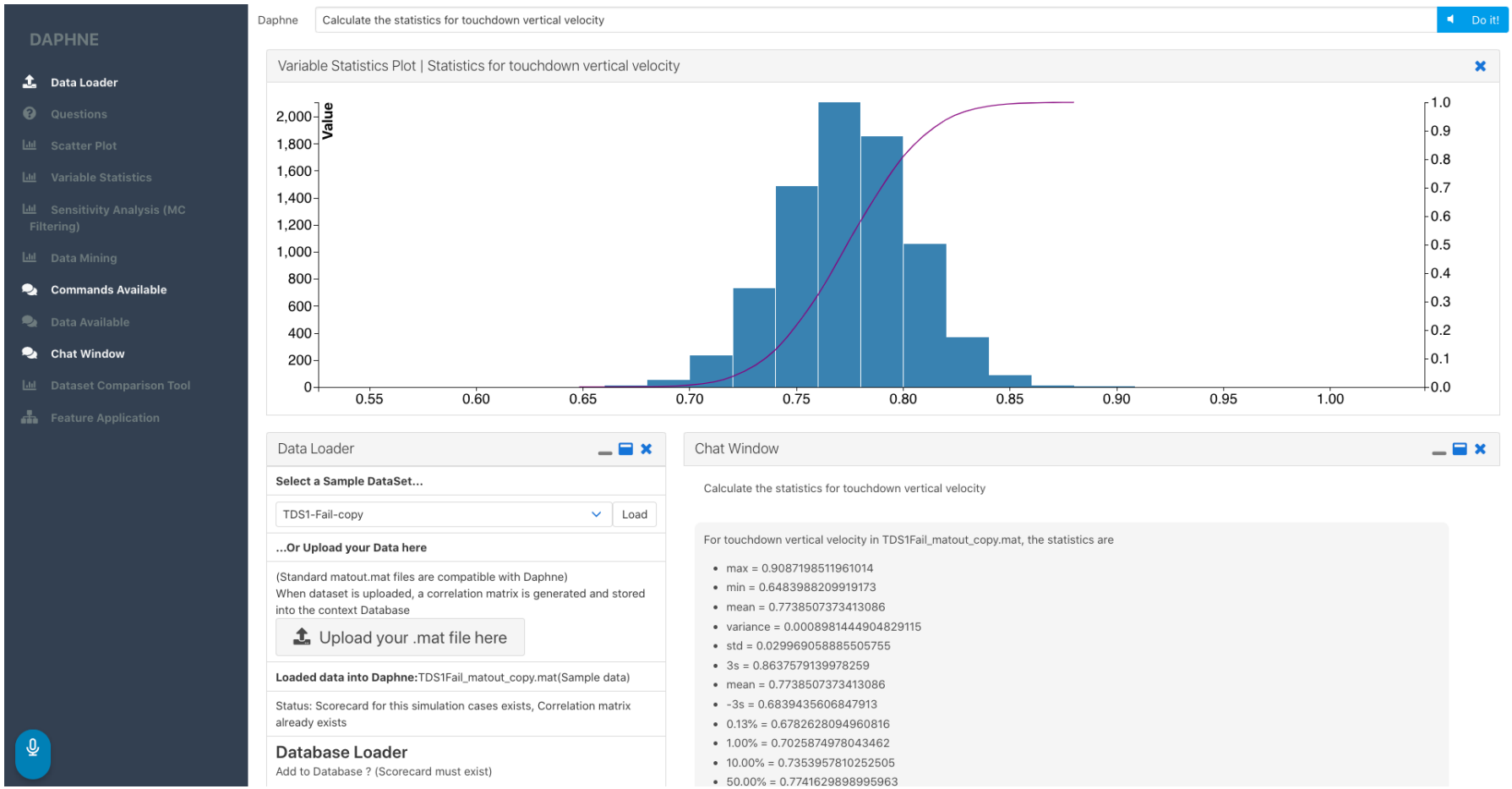


Figure 2.11: Visualizations in Daphne (Reprinted from [1]).

2.5.4 Step 2: Requesting Simulation Results

As discussed in Section 1.2, users have to manually examine simulation results, compute statistics, or generate statistical plots with the objective of identifying metrics of interest. To reduce the cognitive load and manual labor involved in this process, users can request outcomes of interest several ways. One way to get a summary of potentially interesting metrics is to ask Daphne which metrics in the scorecard are flagged or out of spec. Otherwise, the user can ask for the value of a specific metric. For example, in this case study, a beam failure would be expected to affect metrics in the powered descent segment. The user could ask through the chat window: "*What is the touchdown vertical velocity?*". The user could also ask: "*From the scorecard, what metrics are flagged?*" or "*What are the statistics for touchdown velocity?*". For the latter query, Daphne returns a list that indicates metrics of interest and explains why a metric is not satisfied.

2.5.5 Step 3: Examining Results

As shown in 2.10, Daphne provides a list of EDL performance metrics that exceed the requirements flag threshold. As seen in the figure, touchdown vertical velocity at first contact exceeds the 0.82 m/s requirement. Figure 2.11 shows the statistical information Daphne provides. At the top of the screen, Daphne shows a histogram and cumulative distribution function generated from empirical data. In the chat window, Daphne shows additional statistics of the metric under study. This includes various percentiles, minimums and maximums, standard deviation, variance, and mean.

Asides from examining numerical values of critical EDL metrics, a task commonly done is to manually compare new simulation results to previous simulation runs to see if any metric value has changed significantly. In some instances, the difference between two metrics can be evident. For example, an increase in timeline margin of 5 seconds is a significant difference in performance. However, a peak deceleration difference of 0.02 g's in the 99-percentile might not be significant and could be attributed to statistical noise. To help with this task, the user can perform a dataset comparison and can examine what metric's distribution is significantly different across two datasets. In the context of this experiment, the user can select the nadir beam failure simulation case and

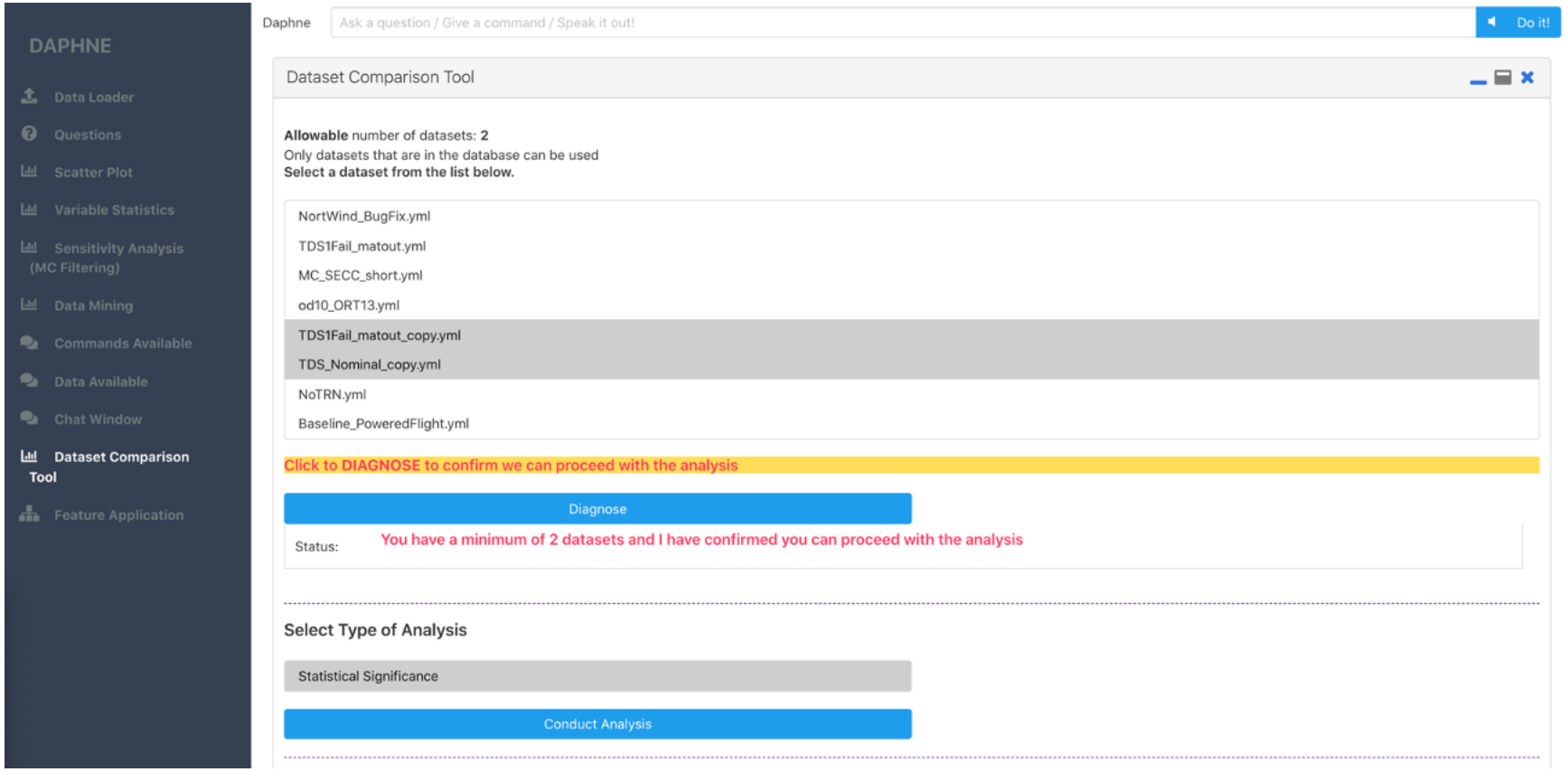


Figure 2.12: Comparison tool in Daphne.

a nominal simulation to see what metrics change the most. Figure 2.12 shows the two datasets selected. The component will display a prompt indicating that enough datasets were selected for the analysis. Once the user has selected their datasets of interest, they can select statistical analysis and Daphne will compare the simulation outputs. As a response, Daphne provides a table like the one depicted in Figure 2.13. The first column shows the metric name, followed by the type of metric, its value type, and the units of measure. The following four columns show the POST results and the standard deviation of the metric in both datasets side by side. The last two columns show the p-value and the D-statistic (i.e. maximum vertical distance between both CDFs).

Examination of the comparison table shows that some of the most affected metrics when a nadir beam fails include Navigation filter propagation errors, Terrain-relative navigation performance, altitude of first TDS ground solution, navigation errors at the start of the constant deceleration during powered flight, timeline margins, and vertical velocity. Intuitively, it should not come as a surprise that these metrics are affected the most given that the navigation filter uses measurements collected by the TDS to estimate altitude and velocity. Thus, the navigation relies solely on IMU-informed estimates. Here, we see that the vertical touchdown velocity examined previously is significantly different to a nominal scenario. This metric is critical given that even a small increase in touchdown vertical velocity can affect the structural integrity of the rover and its instruments. In the following steps we will investigate what parameters affect that increase in velocity the most.

2.5.6 Step 4: Analysis

2.5.6.1 Identification of most influential features driving Vertical Touchdown Velocity

To help with the task of identifying what parameters are affected by a nadir beam failure and also drive the increase in touchdown velocity observed, we will use the sensitivity analysis feature in Daphne. Here, the analysis is largely driven by data constraints given by the user. For this study, it was assumed that the expert was interested in finding out what parameters following heatshield separation cause the increase in the 99%-tile in touchdown velocity. We selected events from heatshield separation onward given that this is when the TDS begins to collect altitude and

Dataset Comparison Tool

				TDS1Fail_matout_copy.yml		TDS_Nominal_copy.yml		Statistical Significance	
Metric Name	**Sheet Name**	**Value Type**	**Units**	**Results**	**sigma**	**Results**	**sigma**	**P-Value**	**D-statistic**
NAV Filter Propogation Error (from decision)	TRN	m	99%-tile	14.886	0.000	14.900	0.000	0.0000000000	1.000
Residual Error	TRN	m	99%-tile	2.019	0.000	1.987	0.000	0.0000000000	1.000
Surface Distortion	TRN	m	99%-tile	109.829	0.000	111.150	0.000	0.0000000000	1.000
Horizontal Distortion due to DEM Elevation	TRN	m	99%-tile	0.202	0.000	0.206	0.000	0.0000000000	1.000
TRN End-to-End Performance	EDL Metrics	m	99%-tile	46.986	0.000	47.349	0.000	0.0000000000	1.000
Altitude of First Ground Solution (strength 5)	Parachute Descent	m	1%-tile	2637.610	991.224	3667.215	840.358	0.0000000000	0.489
TRN Timeline Margin	TRN	s	1%-tile	10.778	5.318	16.423	5.333	0.0000000000	0.391
Altitude of First Ground Solution (strength 4)	Parachute Descent	m	1%-tile	3663.763	854.060	5568.937	429.569	0.0000000000	0.346
Altitude of First Ground Solution (strength 3)	Parachute Descent	m	1%-tile	5560.268	452.400	6091.153	365.385	0.0000000000	0.343
Nav Alt Error @ CD Start	Powered Flight	m	99%-tile	2.403	1.478	1.577	0.662	0.0000000000	0.343
Time in GNC Mode 21	Powered Flight	s	1%-tile	23.939	6.236	29.625	6.303	0.0000000000	0.338
Timeline Margin	EDL Metrics	s	1%-tile	23.939	6.236	29.625	6.303	0.0000000000	0.338
Time in GNC Mode 20	Powered Flight	s	50%-tile	33.125	7.764	27.125	7.409	0.0000000000	0.282
DS Max Vert Vel @ Rover Sep	Powered Flight	m/s	99%-tile	0.886	0.047	0.852	0.041	0.0000000000	0.263
DS Min Vert Vel @ Rover Sep	Powered Flight	m/s	1%-tile	0.672	0.047	0.667	0.041	0.0000000000	0.263
Att Comp. Vert Velocity Knowledge Error @ Rover Sep	Powered Flight	m/s	99%-tile	0.085	0.025	0.065	0.021	0.0000000000	0.208
Absolute Value of Rover Velocity in Descent Stage - Z	Powered Flight	m/s	99%-tile	0.846	0.031	0.829	0.027	0.0000000000	0.161
Touchdown Vertical Velocity	EDL Metrics	m/s	99%-	0.843	0.030	0.824	0.027	0.0000000000	0.161

Figure 2.13: Results of dataset comparison.

velocity measurements.

Figure 2.14 shows the logical flow used to specify user constraints for the sensitivity analysis. The first step in this process is for the user to select a metric of interest.

For a user-selected metric of interest, Daphne finds the array in the simulation dataset or calculates (if scorecard metric) the value of the metric for the current simulation case. To reduce the amount of data involved in the process and make sure only relevant data is used, the user has the option of selecting the range in the dataset they are interested in. Given that EDL experts are mostly familiar with ranges of values of typical metrics, they can simply type the values. The example histogram shown next to this step highlights an example region of interest in the dataset. In that case, the user simply specifies the minimum and maximum value to only use that highlighted range in the analysis. Once the data of interest is selected, the user has the option of dividing the dataset into two classes by one of three criteria: percentile, user-specified cutoff value, or pass/fail criteria. If the user specifies a percentile, the dataset is divided into two classes based on that value. Class 0 corresponds to the lower range and class 1 corresponds to the upper range. Along the same lines, the user can divide the dataset into two based on a threshold value. For example, assuming the second plot in Figure 2.14 ranges from 0 to 360 and the user is interested in conducting SA between 0 to 90 vs 90 to 360, by simply typing the value of 90, the data is divided into two classes accordingly. There is an additional option for users to specify two values as cutoff criteria, this is done if the user wants to compare median values against values in the tail of a distribution, for example. The selection in the third plot on the figure illustrates a scenario where users are interested in identifying what is driving that center peak values against the other two. In the case where the user wants to use pass/fail criteria, the division of the dataset will be driven by the thresholds available in the scorecard. It must be noted that this would only be feasible if the threshold already exists in the scorecard.

For this particular analysis, we used the following criteria: 1) Touchdown vertical velocity as the target metric; 2) use the entire dataset; 3) divide the dataset into two classes using a cutoff value (with the value of the 99%-tile of touchdown velocity); 4) run sensitivity analysis against

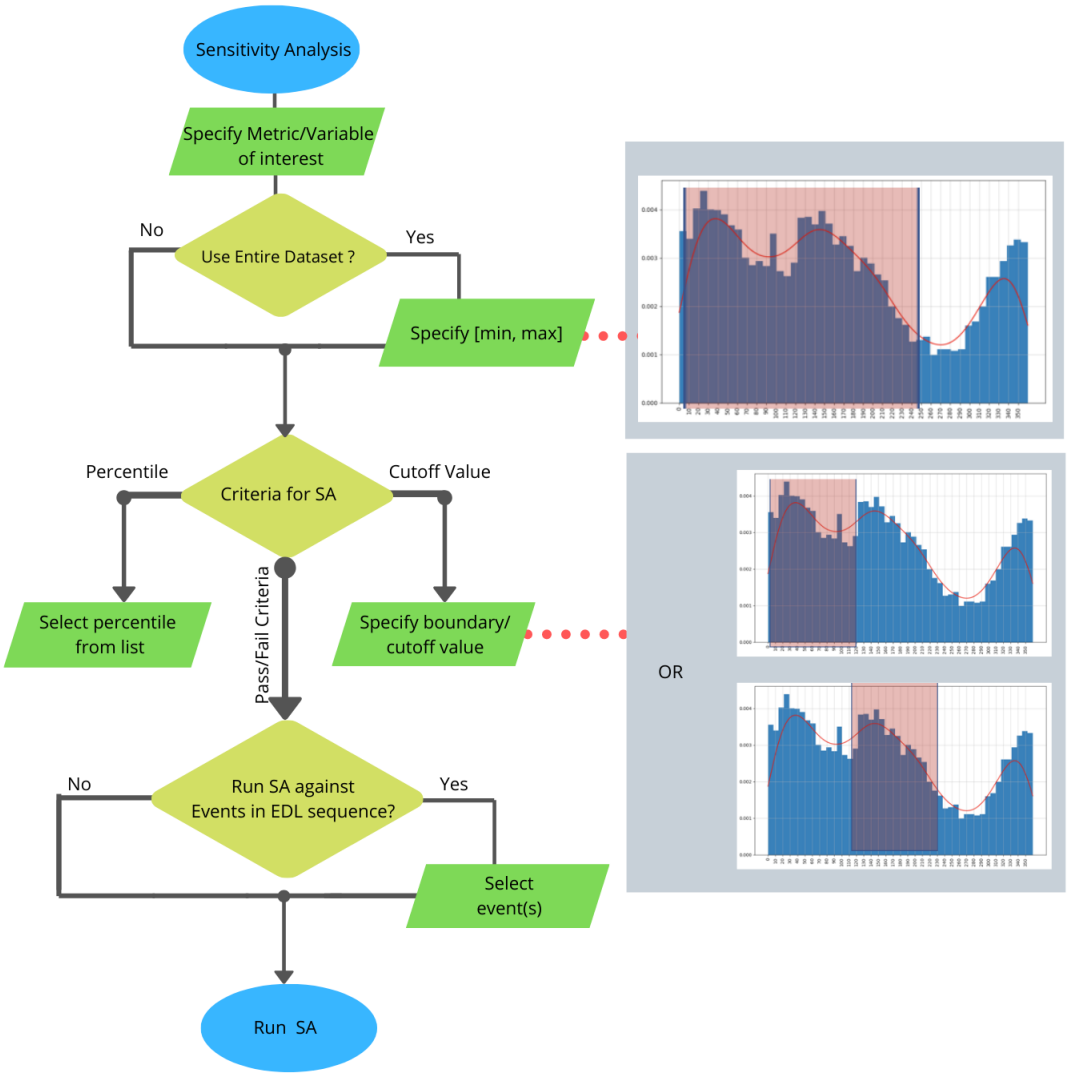


Figure 2.14: Steps involved in preparing data for sensitivity analysis (Reprinted from [1]).

all relevant variables in the events following heatshield separation until the Sky Crane maneuver. Figure 2.15 shows the component in the interface where the user can specify their data constraints for the analysis.

Once the sensitivity analysis is initiated, Daphne will load the data against which to run the SA and will divide the dataset into two classes depending on the criteria selected. For each parameter/variable, the CDFs are estimated and the maximum distance between both classes is calculated along with its p-value. Daphne then ranks parameters by decreasing d-statistic and presents the

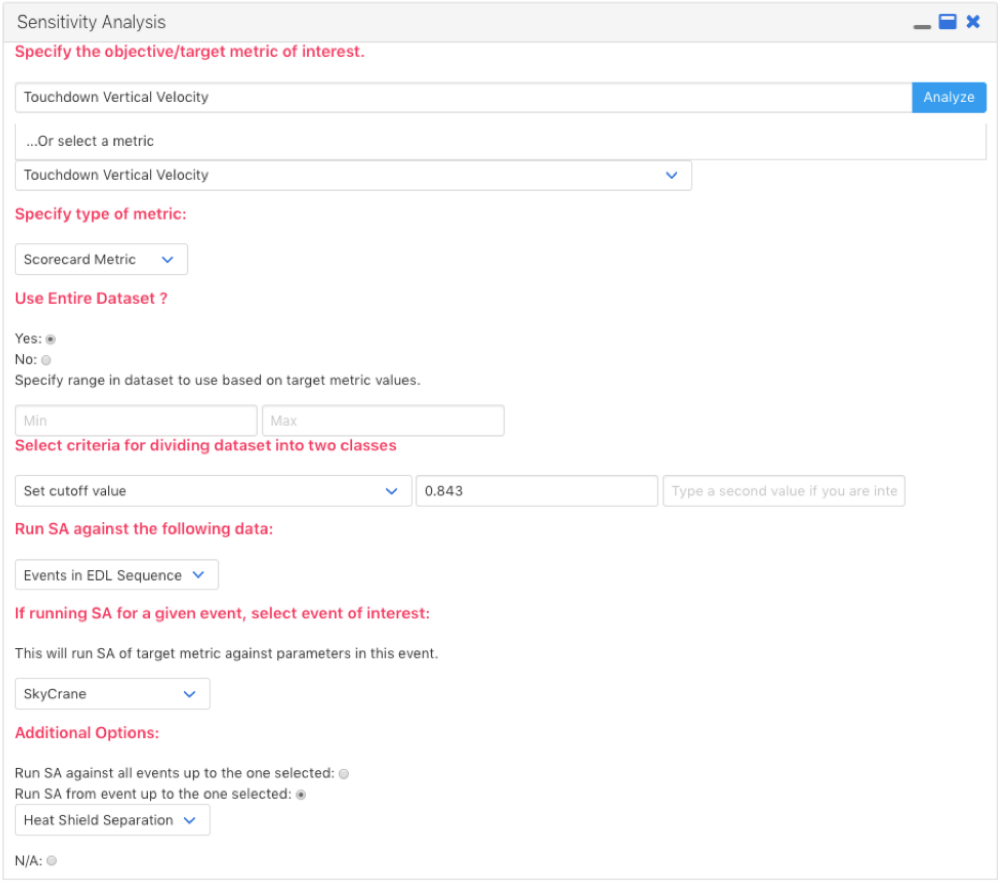
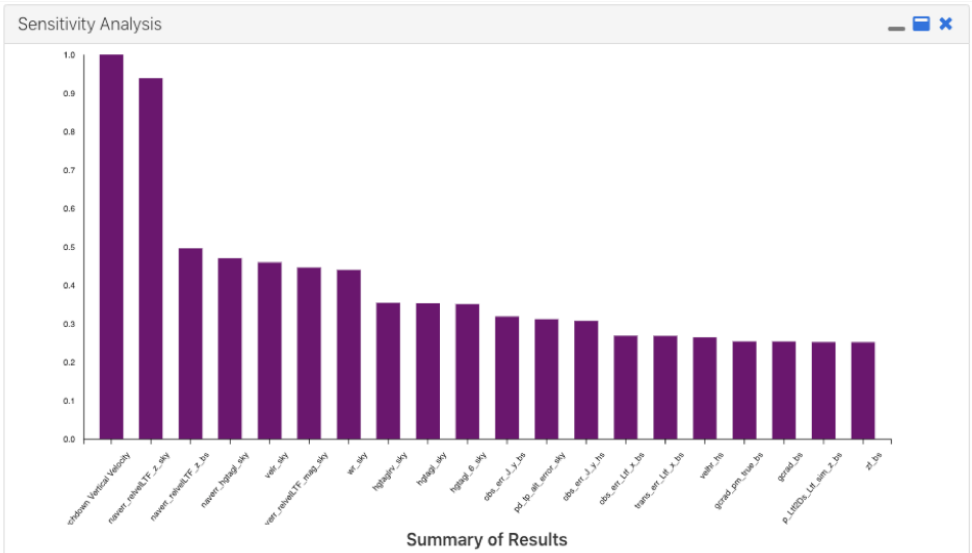


Figure 2.15: User-specified constraints for the data analysis (Reprinted from [1]).

results in a graph and table format, as presented in Figure 2.16.

The top portion of the figure shows a visualization of the ranking of influential variables from most influential to less influential (based on the K-S d-statistic). In the context of the study, this graph showed that in the event of a known nadir beam failure, touchdown vertical velocity is affected by the vertical component of the velocity navigation errors from Backshell Separation to Sky Crane event, with Sky Crane (the last event prior to touchdown) being the most influential. Followed are altitude navigation errors at Sky Crane maneuver as well as altitude above ground level at Sky Crane. In the event of a canted beam failure, we repeated the same procedure. In this case, sensitivity analysis showed that velocity knowledge errors affect the increase in the magnitude of horizontal touchdown velocity. However, unlike a nadir beam failure, altitude navigation



Input Metric	Distance	P-Value	Description	Label	Model
Touchdown Vertical Velocity	1	0.0000000	Unavailable	Touchdown	
naverr_relvelLTF_z_sky	0.9384399551066218	0.0000000	Velocity knowledge error	SkyCrane	
naverr_relvelLTF_z_bs	0.49631032547699216	0.0000000	Velocity knowledge error	Backshell Separation	
naverr_hgtagl_sky	0.4698653198653199	0.0000000	Altitude Navigation Errors	SkyCrane	
velr_sky	0.4594135802469136	0.0000000	Relative velocity	SkyCrane	
naverr_relvelLTF_mag_sky	0.4463524130190797	0.0000000	Velocity knowledge error	SkyCrane	
wr_sky	0.44031986531986533	0.0000000	Vertical velocity	SkyCrane	
hgtaglrv_sky	0.35422278338945007	0.0000000	Marsgram altitude	SkyCrane	
hgtagl_sky	0.35296015712682377	0.0000000	Altitude	SkyCrane	

Figure 2.16: Results for a nadir-beam failure (Reprinted from [1]).

errors were not classified as influential. These results initially go in accord with what EDL experts identified, without the extensive search of individual parameters.

2.5.6.2 Explanation of outcomes

As seen in the previous subsection, altitude and velocity navigation errors drive the increase in touchdown vertical velocity. However, even though a ranking of influential parameters gives a good representation of the underlying behavior and drivers of the system, they do not inform us under what circumstances they drive the behavior described. For example, we know that navigation errors affect touchdown velocities. However, we do not know if this happens only when navigation errors are high extremely high. To help explain the trends among features in the increase in touchdown vertical velocity, the EDL Data Mining Section was employed.

To help explain rules obtained, Daphne provides a table listing rules discovered in the form of $X \rightarrow VelocityInRange$. Here the range can take values between 0 and 4, where X is a conjunction of 3 binary variables obtained from one-hot encoding. For example, in a rule where a variable is labeled with a 4 indicates that values above the 99%-tile of this parameter yield the outcome under study (i.e.high touchdown vertical velocity).

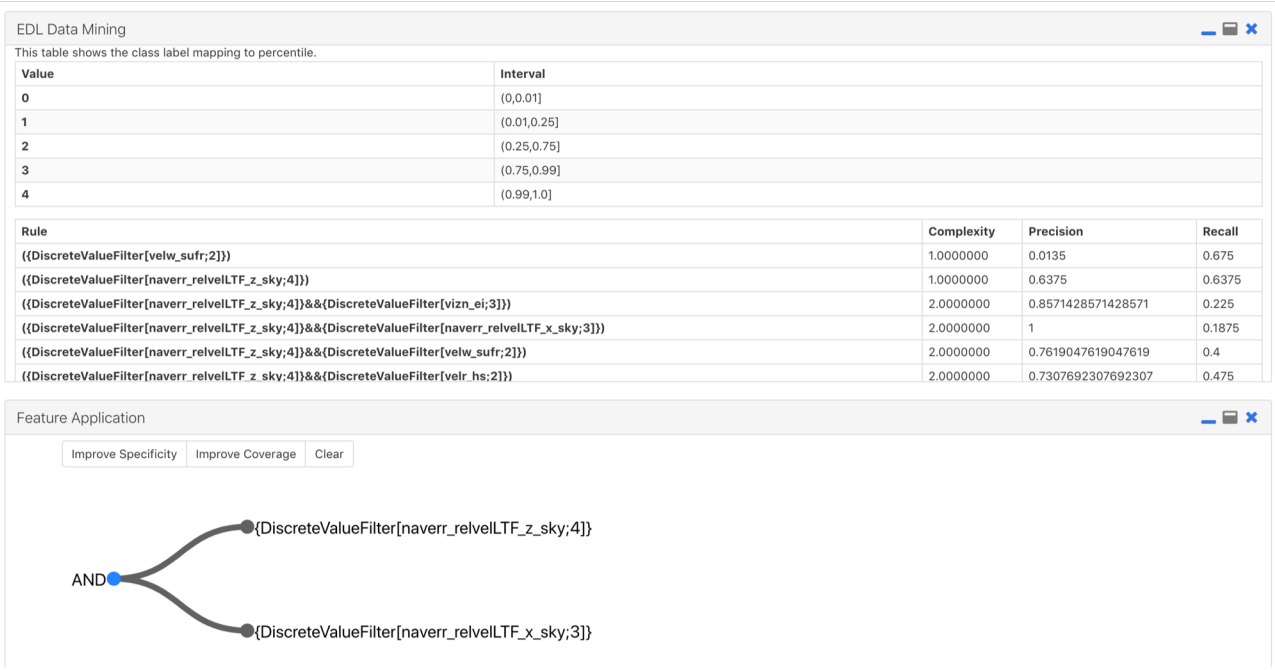


Figure 2.17: Data mining results for a nadir beam failure (Reprinted from [1]).

Results are shown in Figure 2.17. Here, rules are listed along with their complexity, recall, and precision. For a different visualization, the user can click on the rule and visualize a feature tree. Based on the tree depicted in the Figure, a rule exists that indicates that velocity navigation errors in the z-axis above the 99%-tile and velocity knowledge errors in the x-axis between the 75 and 99%-tile yield values in the 99%-tile of the touchdown vertical velocity when a nadir beam fails.

When running the same algorithm for the case of a canted beam failure, rules generated showed that the magnitude of vertical touchdown velocity drivers above the 99%-tile, as opposed to components, drive the increase in horizontal velocity. Other rules showed that, this range of velocity

magnitude errors and navigation filter attitude errors between the 75 and 99%-tile drive the increase in vertical velocity. For both simulation cases, results presented here agree with the conclusions reached by experts "manually."

2.6 Conclusions

This chapter introduced Daphne-EDL, a cognitive assistant designed to help EDL experts with data processing and performance assessment of simulation outputs. Daphne is designed to help experts in two ways: 1) by automating some parts of the data processing task, and 2) by conducting analysis to help investigate and identify performance features. Experts can conduct all tasks through a web interface. Requests like summarizing results, obtaining metrics from previous missions, requesting metric values, and statistics can be done interactively using written or verbal natural language form. Experts can also conduct three types of analysis: dataset comparison, sensitivity analysis, and data mining. The data comparison section helps verify where the value of a metric is significantly different across two datasets. The sensitivity analysis tool helps identify the most important performance drivers. Data mining, on the other hand, is useful for explaining a simulation outcome.

We believe that this interactive approach reduces the burden of navigating through large amounts of data by providing the correct information quickly when needed. While the approach does not automatically provide the solutions to problems in EDL, we hope it can help reduce the cognitive load imposed by the data analysis process. Given that this task is very time-consuming, experts can only explore a fraction of the data by traditional manual approaches. Therefore, tools like Daphne can help tackle these limitations. We envision that flexible platforms, like cognitive assistants, can help EDL experts with test reporting and data management in addition to analysis.

The advantage of this approach is that this platform allows for experts to easily incorporate data constraints into sensitivity analysis and rule mining processes. Incorporating constraints helps reduce the volume of data, as opposed to fully automated approaches. Furthermore, incorporating data constraints by the user helps eliminate redundancies in the data and information that might not be of interest.

With Daphne, explanations are provided in visual and textual format, making them more comprehensible than raw results. Since the goal of CAs is to facilitate and embrace a human-machine collaborative approach to problem-solving, information is delivered in an understandable human way, trying to mimic how many experts would provide such explanations.

One limitation of this work is that it is tailored for data analysis tasks and does not contain general knowledge about EDL systems. Consequently, at the moment, Daphne is primarily helpful for analyzing simulation data products instead of acting as a recommendation system. Nevertheless, Daphne-EDL could benefit from a knowledge base that contains rules of thumb and figures of merit about key performance drivers that can be useful for the analysis and development of future EDL architectures. Rules of thumb could be useful for designing and evaluating candidate architectures and configurations for future missions. Furthermore, a database of tests and reports that evaluate EDL criteria could prove very useful for operational scenarios. In EDL operations, automation plays a key role. During these days before landing, experts are 24/7 obtaining data from other teams, such as Navigation, updating their simulation setup and assess EDL performance given the spacecraft's onboard knowledge of its position and its actual state. Given the fast pace of all tasks required, experts face physical and emotional human factors that can lead to human error. Common factors are fatigue, pressure, and communication- among many others. Hence, experts tend to automate tasks like file generations and simulation execution automatically to minimize error.

Daphne-EDL can also benefit from mixed-initiative approaches to enhance the teacher-apprentice vision of the system. It would be helpful if an expert can provide feedback on the results generated. For example, if analysis identifies some patterns that do not make sense (i.e., there is no manner in which items in the pattern can affect each other), experts can indicate Daphne. This way, Daphne can learn this information and make use of it for future analysis. Another limitation is that the interface has not been tested with subject matter experts. Daphne's skills have been used in several case studies at NASA to help identify performance drivers and sensitivities. However, how well the tool and the design help experts achieve their goals has not been assessed. Future work should assess these qualities of Daphne-EDL and should explore ways the system can be improved.

3. KNOWLEDGE GRAPHS AND STATISTICAL RELATIONAL LEARNING FOR REASONING ABOUT RELATIONSHIPS BETWEEN EDL PARAMETERS*

3.1 Introduction

As discussed in Sections 1.1, EDL systems are highly complex for various reasons. First, increased performance requirements have pushed the limits of EDL technology, resulting in systems with a much higher technical complexity than Viking-era EDL systems [155]. Second, EDL systems and are coupled to many major sources of uncertainty. These uncertainties include, but are not limited to, vehicle aerodynamics, launch window, and atmospheric conditions during day-of-entry events.

For EDL systems, analyzing and identifying key performance drivers remains a manual approach and often, assessments are made using previous systems expertise [2]. This approach makes verification and validation time consuming and does not ensure that all relevant performance drivers are identified. Consequently, only a handful of quantities/parameters are examined. In the light of these challenges, EDL experts have recommended using tools that enable tractability and facilitate data analysis [31]. Although previous chapters have discussed using a cognitive assistant to help with this task. We believe that EDL can benefit from new forms of knowledge representation and reasoning capabilities to help analyze EDL systems. These analysis capabilities can help inform EDL decisions, reduce lifecycle costs, identify areas of risk, and ensure mission success.

Primarily motivated by data growth and complexity of systems, fields like engineering, healthcare, and social network sciences have opted to use knowledge graphs as a form of knowledge representation to gain valuable insight from data. Knowledge graphs represent real-world data (e.g., social networks, e-commerce sites) as graphs. In KGs, entities, or nodes, represent objects like people, places, things, for example, and their interrelations [108] as a graph. In the engi-

*Parts of this chapter have been adapted from "A Cognitive Assistant for Entry, Descent, and Landing Architecture Analysis" (2019)[2] and "Interactive Explanation of Entry, Descent, and Landing Simulations" (2020)[1] by Santini De Leon, S., Selva, D., and Way, D. with permission from IEEE and AIAA, respectively.

neering sector, the work described in [156] uses a knowledge graph to improve ecotoxicological effect predictions for the Norwegian Institute for Water Research (NIVA). In the healthcare system, knowledge graphs have been used to help identify drugs with both antiviral and anti-inflammatory properties that could be used to treat severe cases of COVID-19 virus [157]. In the social network analysis domain, knowledge graphs are commonly used to identify relationships between people, for example, [158, 111, 112].

In the aerospace domain, NASA has also implemented such technology to deal with big data growth and information sharing issues. For example, NASA uses a knowledge graph designed to help experts access documents and information about lessons learned across groups and centers [113]. Furthermore, NASA has also been placing efforts on developing a dynamic knowledge base useful for research in the Earth science community [159].

Primarily motivated by these advances, we believe that this technology could help represent different aspects of EDL architectures and how they interact. This technology could be beneficial for modeling inter-dependencies between EDL elements and their environment. For example, in an EDL knowledge graph, nodes could represent subsystems, components, and metrics. Furthermore, relationships may describe interactions between components and subsystems. These could be simulation-specific or could depict general beliefs. Given the flexibility of knowledge graphs, relationships can also capture correlations between metrics and whether they are coupled to any known component or subsystem. KGs coupled with a machine learning algorithm could be helpful to discover relationships between EDL elements.

This chapter introduces a method that combines expert knowledge and data-driven analysis to infer new relationships between EDL variables and key metrics. With the proposed framework, we use a knowledge graph to relate variables and metrics that are known to be correlated to each other. The proposed method uses a statistical relational learning (SRL) framework to infer relationships between EDL variables and metrics using expert knowledge. The framework is applied to a real-world EDL dataset and is used to infer relationships between EDL variables and key performance metrics.

3.1.1 Statistical Relational Learning

As discussed in 1.4.3, the task of extracting new and valuable knowledge from a knowledge graph is commonly framed as a link prediction problem. With such a framework, the goal is to discover relationships that may uncover risks and/or opportunities. Furthermore, graphs are so large, complex, and unstructured that manual discovery of relationships is unfeasible. One potential discovery, for example, could be that we believed that atmospheric winds only contributed to parachute deployment and recontact. However, they could be linked to rover separation and touchdown conditions.

Statistical relational learning (SRL) is a sub-field from machine learning that uses a model to support learning from relational data. SRL provides answers to queries based on probabilistic inference [134]. One quality that distinguishes SRL from other methods is that it integrates probabilistic reasoning, first-order logic, and machine learning [138].

Most SRL approaches express probabilistic semantics using either directed or undirected graphical models. Furthermore, the most common representation formalisms of SRL are either logic-based (e.g., rule-based) or frame-based (e.g., object-oriented). As described by [134, 135], directed models (e.g., Bayesian Networks) represent generative models. In contrast, undirected models (e.g., Markov Networks) help identify non-causal dependencies. As discussed in 1.4.3, the two most widely applied formalisms of SRL are Markov Logic Networks (MLN) and Probabilistic Soft Logic (PSL)[160]. Both MLNs and PSL use logic-based formalisms and use undirected graphical models in a single representation. Both frameworks create programs from first-order weighted rules that capture the relational dependency between entities.

The weight of a rule is an indicator of the degree of belief in a rule. A large weight will result in a more significant difference in the log probability between the worlds that satisfy and do not satisfy the formula [137, 103]. In MLNs, all pairs of rules and their respective weight are used to ground a Markov Network. Once all groundings are discovered, MLNs are used to predict links between entities by employing probabilistic inference. The current methods employed are Markov Chain Monte Carlo (MCMC) Gibss Sampling [139]. PSL, however, was designed to be

more computationally efficient during the inference process. PSL, like MLN, creates a program from first-order weighted rules that capture the relational dependency between entities. PSL conducts inference using the most probable explanation (MPE) to find the truth values of atoms that maximize the likelihood of rules being satisfied [143]. This framework is much faster. It frames the inference task as a convex optimization problem solvable in linear time [145]. Asides from the inference mechanisms and computational cost difference between MLNs and PSL, one distinguishing characteristic is that in MLNs, atoms take boolean truth values of 0 or 1. In PSL, they take continuous truth values in the interval of [0,1].

Given the computational advantage of PSL over MLNs, we believe that PSL is a suitable framework for making inferences about the relationship between parameters in an EDL knowledge graph. Furthermore, PSL can infer continuous truth values- which permits the discovery of partial levels of truth as opposed to true (1) or false (0) exclusively [145].

3.2 Method

This section demonstrates how we integrate knowledge graphs and PSL to infer new relationships between EDL variables and critical EDL metrics. The proposed method is unique given that it combines knowledge and data-driven information. The knowledge graph contains a combination of both. Many metrics and variables in the KG are related to each other if there exists any linear correlations between them. However, given there are many non-linear interactions in the system, many relationships were added manually. Similarly, PSL incorporates a combination of knowledge and data driven information. Rules used in PSL represent expert beliefs about how variables and metrics interact. However, PSL inferences use the data available as evidence to quantitatively make new inferences.

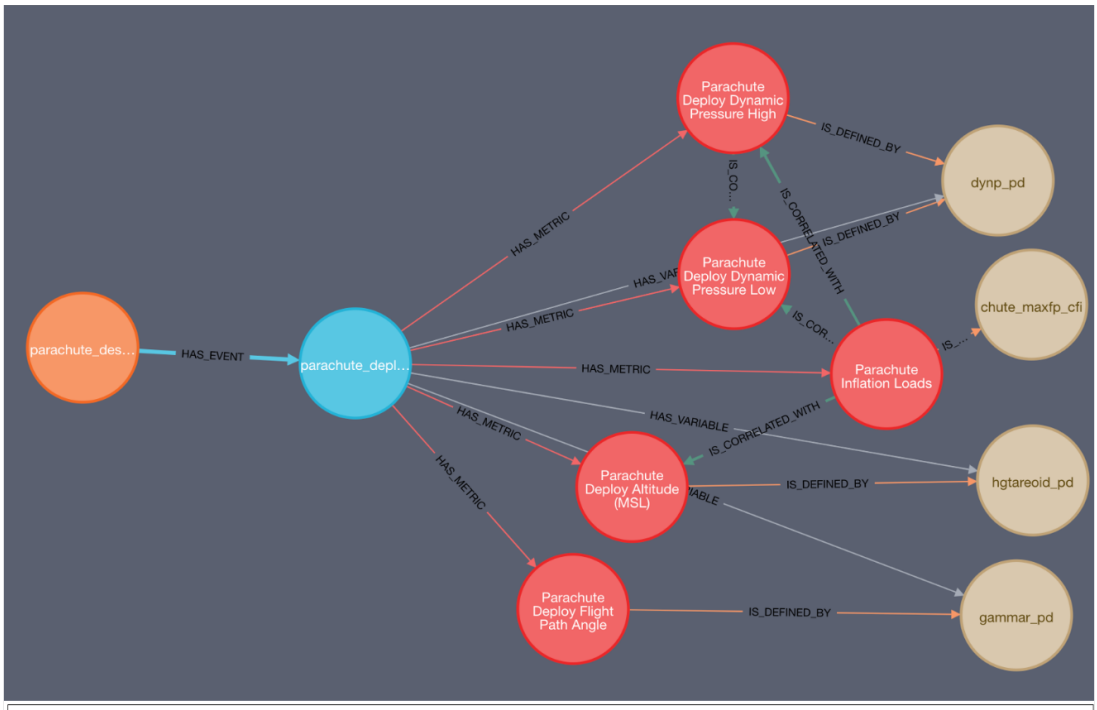
3.2.1 EDL Knowledge Graph

The baseline of the knowledge graph was created using an EDL scorecard- and using Neo4j as a platform. The EDL Scorecard is a document that is structured as a tabular summary report for a particular simulation. This document contains simulation results (e.g., percentiles, means) for

commonly examined system performance metrics, together with the corresponding requirement thresholds, and whether the results satisfy those requirements. We selected this document as a starting point to add to the knowledge graph information about the key metrics for all segments (e.g., entry, parachute descent, and powered descent) and all of the events that comprise them. The number of parameters that define an EDL architecture can reach the tens of thousands, and the scorecard provides a more concise representation that captures the most relevant information. A graphical form of the information contained in the Scorecard is depicted in Figure 3.1.

For example, Figure 3.1 shows the segment of parachute descent, and one of the events that characterize it: parachute deployment. In the Scorecard, each event contains a table with the metrics that characterize it and the calculation/model that defines it. For example, the parachute deploy segment is characterized by parameters such as inflation loads and mach number. In these instances, the metrics are defined by a single variable that is generated in the simulation. However, other metrics are calculated by more complex models (e.g., probability of success, slopes landed on). In these instances, we used regular expression operations to extract all parameters that define the metric.

The scorecard provides the knowledge graph with structural information about the relations between variables and metrics but does not give any information about relations of these variables with EDL subsystems, components or functions. For this purpose, we created a simple SysML block definition diagram and activity diagram that describes the EDL Guidance Navigation and Control subsystem. The idea was to encapsulate relationships between key components such as the inertial measurement unit and the navigation filter, what they do, and how they map to scorecard metrics. Figure 3.3 shows the block diagram of the EDL GNC subsystem and its components. This diagram shows that the navigation filter receives information from both the IMU and the TDS radars. With this information, the navigation filter generates altitude, velocity, and attitude estimates. Attitude estimates produced by the navigation filter are used by the attitude controller to create correction torques. The thruster logic converts these into commands that are executed by the Reaction Control System (RCS) [161]. Figure 3.4 shows the component for the terminal descent



Parachute Deploy Conditions

Metric	Type	Units	Simulation Case Results	Description	Calculation	<,>	Yellow Limit	Red Limit
Angle of attack at parachute deploy	99%-tile	degrees	XXXXX	XXXXX	ans = alptot_pd	<	5	12
Parachute Deploy Mach Number	99%-tile	n/a	XXXX	XXXX	ans = mach_pd	<	2.15	2.25
Parachute Deploy Mach Number	1%-tile	n/a	XXXX	XXXX	ans = mach_pd	>	1.6	1.6
Parachute Inflation Loads	99%-tile	1000 lbf	XXXX	XXXX	ans = chute_kpfp_cfi / 4448.22	<	45	50
Parachute Deploy Dynamic Pressure High	99%-tile	Pa	XXXX	XXXX	ans = dynp_pd	<	632	700
Parachute Deploy Dynamic Pressure Low	1%-tile	Pa	XXXX	XXXX	ans = dynp_pd	>		239
Parachute Deploy Altitude (MSL)	1%-tile	km	XXXX	XXXX	ans = hgtareoid_pd / 1e3	>		
Parachute Deploy Altitude (MSL)	99%-tile	km	XXXX	XXXX	ans = hgtareoid_pd / 1e3	>		
Parachute Deploy Flight Path Angle	1%-tile	deg	XXXX	XXXX	ans = gammar_pd	>	-30	-35
Parachute Deploy Flight Path Angle	99%-tile	deg	XXXX	XXXX	ans = gammar_pd	>		

Tab: Parachute Descent

Figure 3.1: Graph representation of Scorecard data.

sensor and information flows from each radar to the NAV Filter. Figure 3.5 shows the graphical representation of the EDL GNC block diagram. We see, for example, that the navigation filter generates two data products: velocity ($naverr_relvellf$) and altitude ($naverr_hgtagl$) errors. Figure 3.6 shows how those two data products map to altitude and navigation errors during different phases of the powered descent segment. The SysML model was exported as a JSON file and was passed on to the KG. In the KG, nodes created from the block definition diagram inherit their class label (e.g.

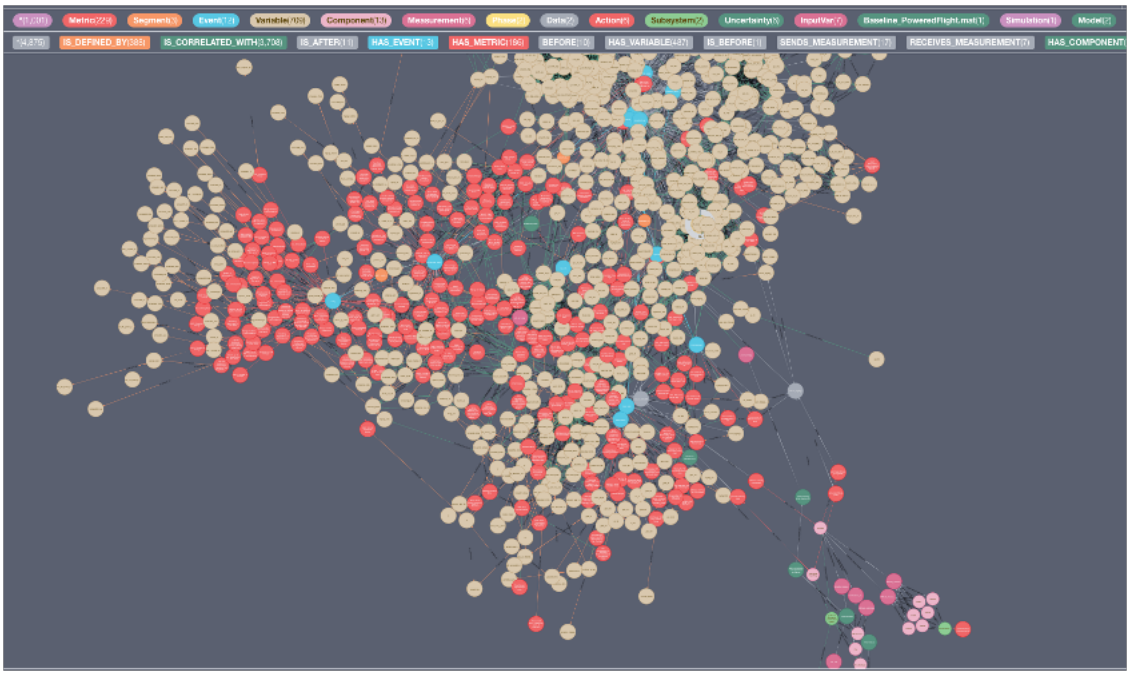


Figure 3.2: EDL knowledge graph.

component, measurement) and information flows are represented by links in the graph. For example, subsystem elements like the Navigation Filter are related to data products in the scorecard such as velocity navigation errors using a relationship type of "$GENERATES$." Relationship types such as "$RECEIVES\_MEASUREMENT$" are used map collected measurements from instruments (e.g., IMU, Radar) to subsystem components.

To further connect metrics and variables, we computed linear correlation coefficients between variables and metrics in the graph using a "nominal" (no failures) Monte Carlo simulation dataset. Pairs of variables/metrics that had strong linear relationships ($r > 0.75$) were connected in the graph via an isCorrelatedWith relation.

Finally, given the lack of additional readily available structured sources of information to enhance the knowledge graph, several of the remaining relationships between parameters were added manually by the authors. Figure 3.2 shows a snapshot of the resulting knowledge graph.

To incorporate additional context into the knowledge graph, variables were mapped to the model, or subsystem, they describe. For example, all variables in the graph related to fuel con-

sumption parachute descent, were mapped to a node named "fuel used". The same was done for variables related to times of events, descent stage impact, thermal parameters, entry guidance consumption, mars lander engine (MLE) parameters, etc.

In the near future, we hope that rich knowledge graphs could be created almost fully automatically by leveraging model-based systems engineering artifacts and perhaps even automated knowledge extraction techniques for unstructured documents such as system requirements documents or design review documents.

To connect metrics and variables, we computed linear correlations and those pairs that had high correlation values were connected in the graph. Given the lack of information available to enhance the knowledge graph and our inability to computationally identify non-linear correlations, many relationships between parameters were added based on domain expert knowledge. Figure 3.2 shows a snapshot of the knowledge graph.

To incorporate additional context into the knowledge graph, variables were mapped to their respective types. For example, all variables in the graph related to fuel consumption, were mapped to a node named fuel using the following syntax: $(m : Variable) - [IsParamType] - > (n : ParamTypename :' fuel')$. The same was done for variables related to times of events, descent stage impact, thermal parameters, entry guidance consumption, mars lander engine (MLE) parameters, etc. An example is shown in Figure 3.7.

3.2.2 PSL Model

PSL is a framework for modeling dependencies in relational domains in the presence of uncertainty. In our work, we use a PSL model to infer the probability of there being a relationship between pairs of items in a rule, given the knowledge graph. Table 3.2.2 presents the rules used to construct the PSL model. For this study, the goal was to explain cases where the spacecraft lands far from the intended landing target, and thus the model tries to find associations between variables and the offset of the landing ellipse from the target. Although our knowledge graph contains a variety of links, we will focus on inferring relationships of the type *IsCorrTo*.

The first four rules in Table 3.2.2 essentially encode a transitivity relation for the correlation

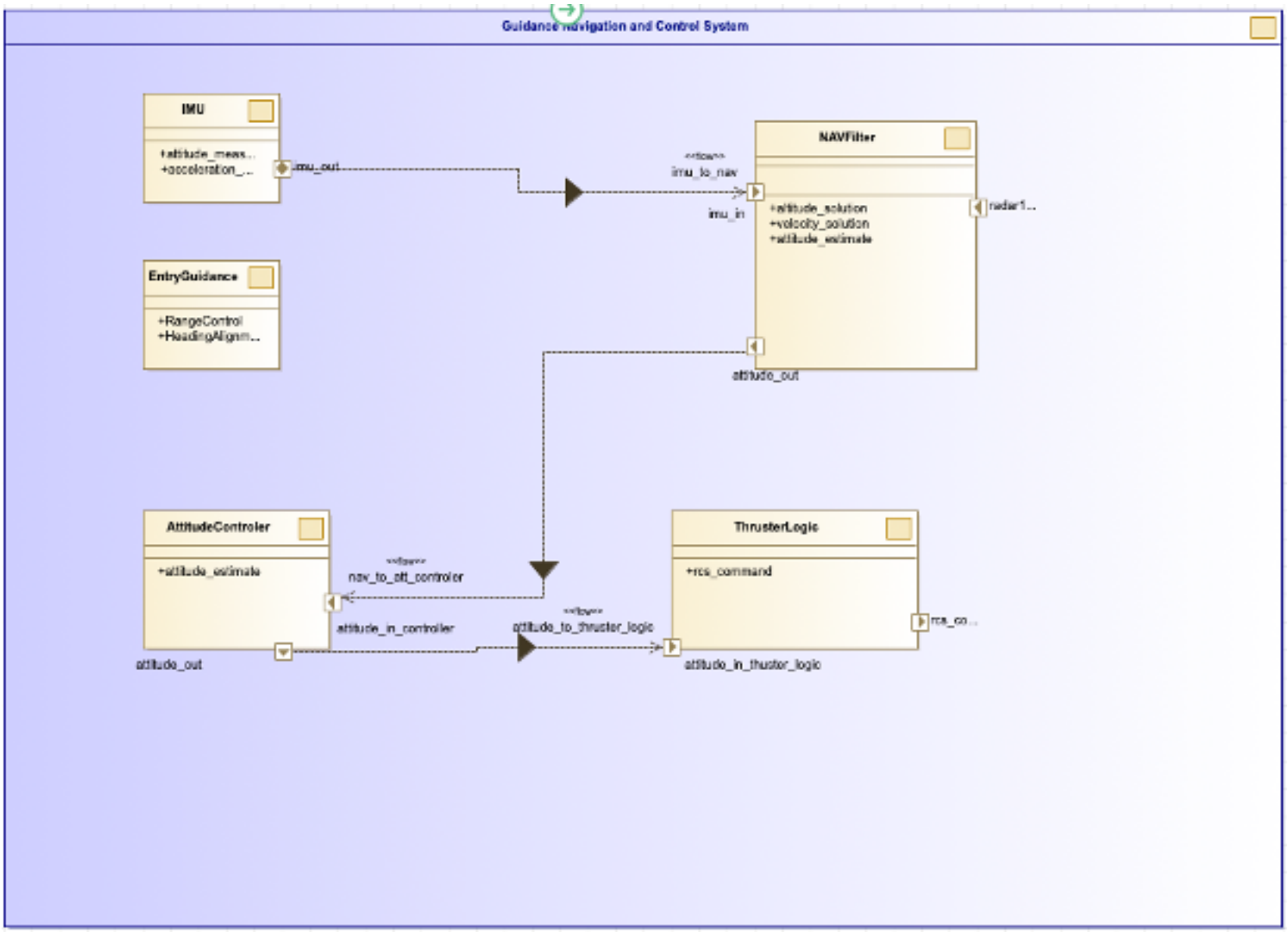


Figure 3.3: GNC block diagram I.

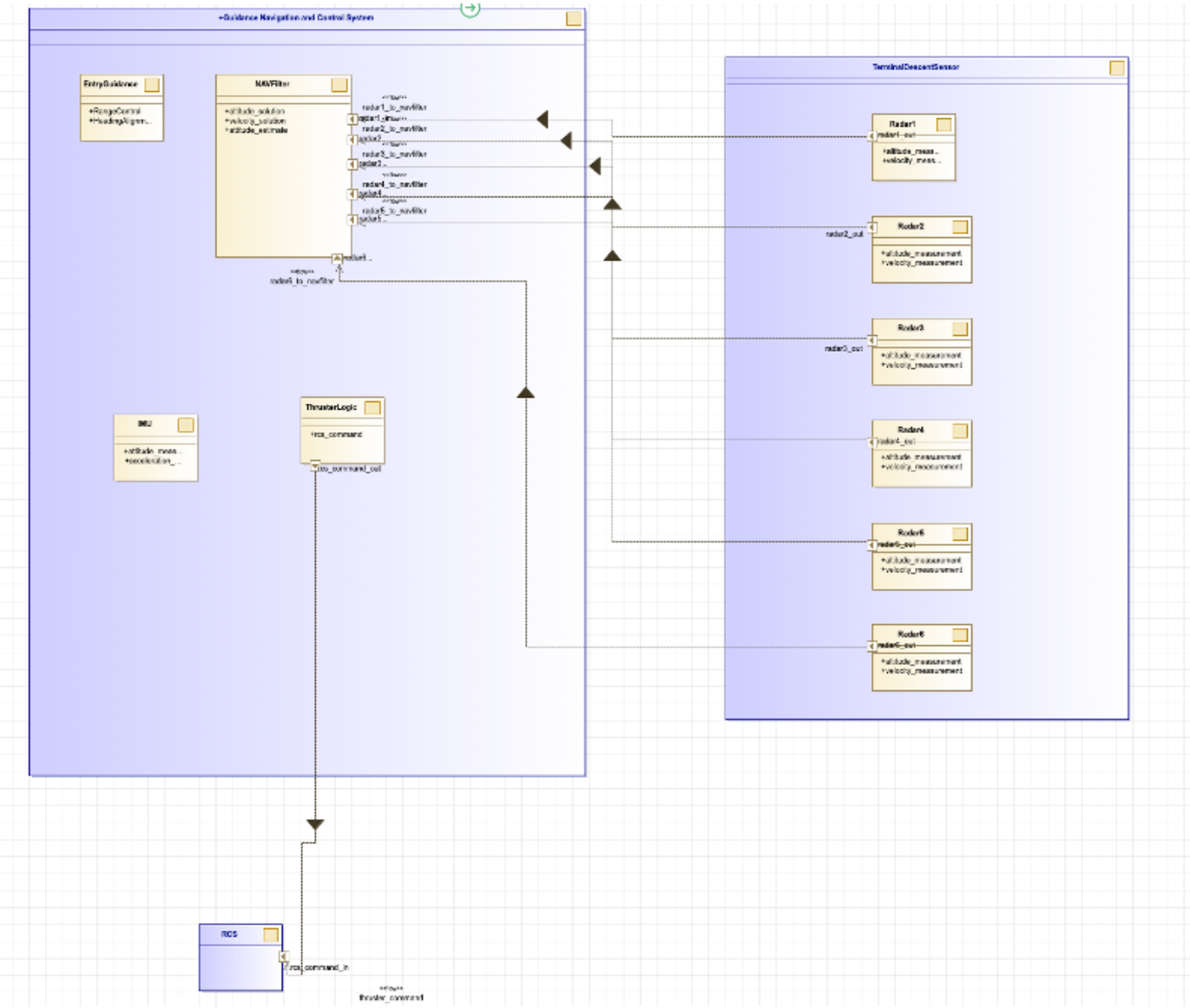


Figure 3.4: GNC block diagram II.

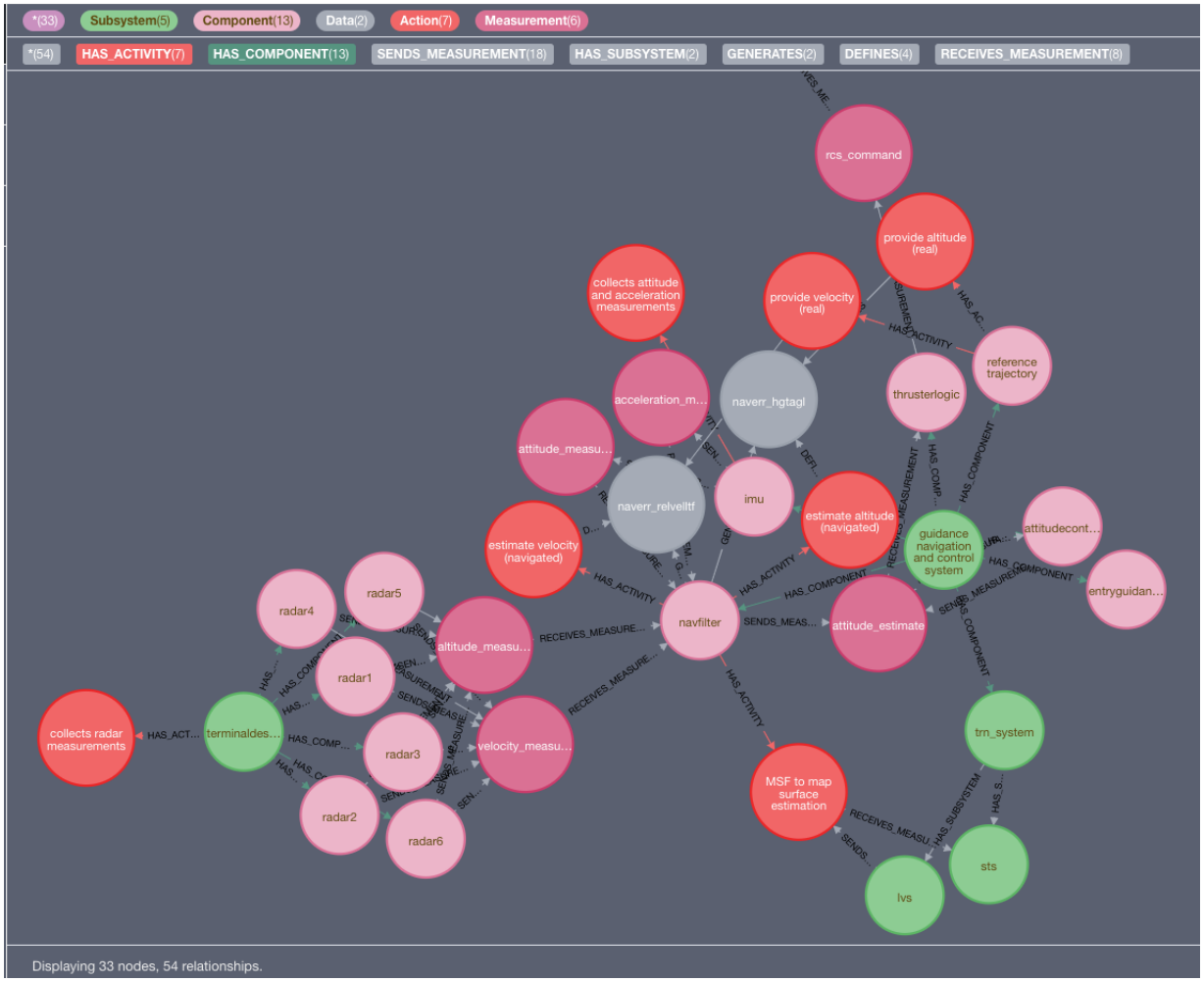


Figure 3.5: Graphical representation of GNC block diagram.

Figure 3.6: Mapping from data products to EDL variables.

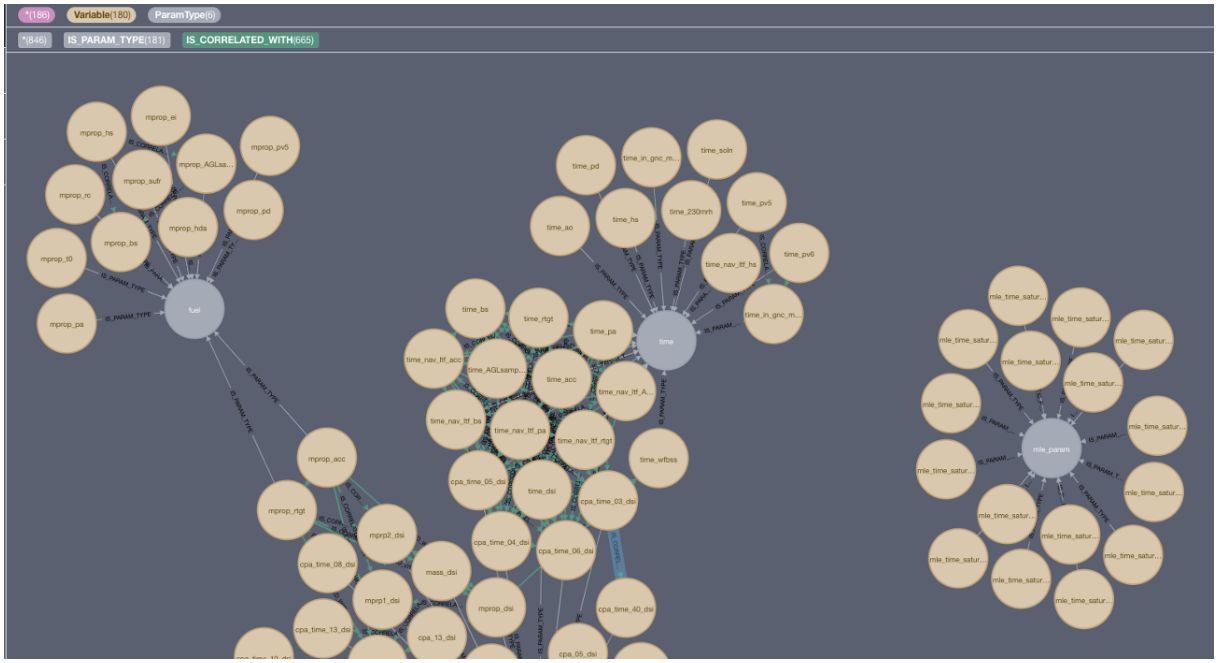


Figure 3.7: Mapping of variable to parameter type.

Rule ID	Rule	Weight
1	$IsCorrTo(V1, M)\&VarCorrsVar(V1, V2)\&(V1! = V2) \Longrightarrow$ **IsCorrTo**(V2, M)	25
2	$VarCorrsVar(V1, V2)\&VarCorrsVar(V2, V3)\&(V1! = V3)$ $\Longrightarrow$ **VarCorrsVar**$(V1, V3)$	5
3	$MetricCorrsMetric(M1, M2)\&IsCorrTo(V1, M1)$ $\Longrightarrow$ **IsCorrTo**$(V2, M1)$	10
4	$IsCorrTo(V1, M)\&IsCorrTo(V2, M)\&$ $\Longrightarrow VarCorrsVar(V1, V2)$	50
5	$MachVar(V2)\&ParMetric(M) \Longrightarrow IsCorrTo(V2, M)$	75
6	$DownrngPdVar(V2)\&ParMetric(M) \Longrightarrow IsCorrTo(V2, M)$	10
7	$RelToFuel(V2)\&EllipseMetric(M) \Longrightarrow !IsCorrTo(V2, M)$	100
8	$RelToGuidanceCons(V)\&EllipseMetric(M) \Longrightarrow !IsCorrTo(V, M).$	95
9	$HasEvent(V, "touchdown", N)\&IsAnalogous(S)\&(V! = S)\&$ EllipseMetric(M) $\longrightarrow$!**IsCorrTo**(V, M)	100
10	$RelToThermal(V2)\&EllipseMetric(M) \Longrightarrow !IsCorrTo(V2, M)$	
11	$RelToRecontact(V2)\&EllipseMetric(M) \Longrightarrow !IsCorrTo(V2, M)$	
12	$RelToMLE(V2)\&EllipseMetric(M) \Longrightarrow !IsCorrTo(V2, M)$	
13	$NavErrTds(V2)\&EllipseMetric(M) \Longrightarrow !IsCorrTo(V2, M)$	
14	$PoweredFlightVar(V)\&EllipseMetric(M)\&HasEvent(V, E, N)$ $\&E! = poweredApproach\&E! = backshellSeparation \Longrightarrow$ **!IsCorrTo**(V, M)	
15	$!IsCorrTo(V, M)$	0.1
16	$!VarCorrsVar(V1, V2)$	0.1
17	$VarCorrsVar(V1, V2) = VarCorrsVar(V2, V1)$	
18	$IsTheSame(V, M) \Longrightarrow IsCorrTo(V, M)$	

Table 3.1: Rules used to create EDL PSL model.

between variables and metrics, and variables to other variables. The first rule expresses the belief that if variable $V1$ is correlated to a metric M, and $V1$ is also correlated to another variable $V2$, then $V2$ is also likely to be correlated to M. Along similar lines, the second rule expresses the relationship between pairs of correlated variables. The third rule looks at the relationship between two metrics and the relationship between one of those metrics to a variable. The fourth rule indicates that if two variables are correlated to the same metric, then they are also correlated to each other. Each rule in Table has a weight that indicates the relative importance of satisfying this rule.

Our model also incorporates some known facts about the EDL sequence to drive the inference. Rules 5 and 6 express the belief that if a variable describes X subsystem or model, such as aero-

dynamics, and the metric defines parachute deploy conditions, then the variable and the metric are likely correlated. For example, we know that Mach, downrange, entry velocity, and entry flight path angle are believed to affect parachute deployment. These nodes in the graph were added as observations for this rule. Observations are provided to PSL as evidence, as discussed in Section 4.2. Opposite to that statement, rules 7-9 state that a variable of a certain type and a metric of another type are observed, then they are unlikely to correlate. For example, rule 8 indicates that parameters related to guidance consumption should not affect our ellipse center distance to target. Observations for these predicates were extracted from the knowledge graph by means of database querying. Where parameter types include fuel usage during powered flight, guidance consumption, and thermal parameters, for example.

Rule 9 expresses the belief that if a variable V corresponds to the event touchdown and is not labeled as analogous it is not correlated to an ellipse-type metric. Variables considered “analogous" were manually identified and included parameters considered to be equivalent to the variables that define the ellipse center distance from target but use a different nomenclature. Some of these include vehicle states represented in different reference planes at touchdown. This way, parameters at touchdown equivalent to ellipse center distance to target are assigned a high probability given that they contain the same information.

Rules 10-14 are similar to rules 7-9. However, these rules have no weight specified. Consequently, they are treated as hard constraints. In the case where we look at navigation errors derived from the terminal descent sensor (rule 14), for example, observations were extracted by searching in the KG for variables that are derived from beam measurements. All observations were obtained by means of database querying.

Rule 14 is similar to rule 9 but it tells our model that any metric during powered flight that is not associated to powered approach or heatshield separation does not affect touchdown ellipse. We added this rule given that the events after backshell separation correspond to the vertical flight portion, and thus they cannot possibly have any effect on the landing ellipse. The initial portion that includes powered approach and heatshield separation could have some effect. We would expect,

however, that the effect would not be very significant.

Rules 15-16 are negative priors; hence, they assumed that ground atoms for the predicates $IsCorrTo$ and $VarCorrsVar$ are false unless they get overpowered by evidence. Items 17 and 18 depicted in 3.2.2 are hard constraints. The first indicates that $Variable1$ being correlated to $Variable2$ is the same as $Variable2$ being correlated to $Variable1$. The second constraint indicates that variables equivalent to the metric under study are correlated to that metric. Some of these variables include coordinates landed on in terms of the Terrain-Relative Navigation (TRN) coordinates landed on in terms of the Safe Target Selection (STS), total range, and divert pixels. These observations include analogous parameters during touchdown mentioned previously. However, unlike analogous parameters, it considers representations of the same parameters at different events in the EDL sequence.

Predicates in bold indicate that they are open predicates- meaning that those inferences made are added to the model and used to make further predictions. Table 3.2 includes a description of each predicate.

3.2.3 Inferences

For the model, each predicate was given a set of observations. For example, the predicate $IsCorrTo$, was given a list of triplets $< variableID, metricID, label >$. A label of 1 indicates that the specific variable and metric are connected in the graph. The pair may be connected because the variable defines the metric, or because they are correlated in some way. The number of observations of variable to metric relationships was 782. Similarly, the predicate $HasEvent$ contains a list of triplets with the form $< variableID, event, variablename >$. The first column contains the unique identifier of the variable, the event in the EDL sequence it is measured at, and its name. This predicate had about 450 observations. The set of observations that increased the computational cost of making inference was the variable-to-variable relationships. The current knowledge graph contained over 25,000 pairs, thus we had to sample from the large set of pairs. The set of observations given to PSL was roughly 10,000. All observations used in PSL were extracted from the knowledge graph using the Cypher query language. In the query,

Predicate	Description
IsCorrTo	Variable is correlated to a metric
VarCorrsVar	A variable that is correlated to another variable
MetricCorrsMetric	Two metrics that are correlated
HasEvent	Maps variables to the event they correspond to in the EDL sequence according to the KG
TrnMetric	Metric that correspond to the TRN subsystem according to the KG
PoweredFlightVar	Variables that correspond to the powered flight segment according to the KG
TrnVar	Metric that correspond to the TRN subsystem according to the KG
ParMetric	Variables that correspond to the parachute deploy event in the KG
MachVar	Variables in the KG that are linked to the node type Mach (under the label aerodynamics)
OffVertVar	Labels variables that describe the off-vertical angle of the vehicle in different events of the EDL sequence
NavErrMetrics	Variables in the KG linked to the node 'navigation error estimate'
DownrngPDVar	Variable downrange at parachute deploy
NavErrTDS	Navigation estimates derived from the TDS subsystem as per the KG
RelToFuel	Variables in the KG that are linked to the node fuel
RelToRecontact	Variables in the KG that are linked to the recontact node
RelToThermal	Variables in the KG that are linked to the thermal parameters node
RelToMle	Variables in the KG that are linked to the MLE parameter node
RelToGuidanceConsumpt	Variables in the KG that are linked to the 'guidance consumption' node

Table 3.2: Predicates used in PSL model.

we search for relationships and if it exists, then the label is 1, otherwise it is zero. For example, to get whether a variable is related to thermal parameters we use the following query: $MATCH(n:Variable),(m:ParamTypename:"thermal")return ID(n), case when (n) <--> (m) then\ 1\ else\ 0\ end\ as\ value$

Using the model presented in Table 3.2.2, we obtained a total of 273,634 inferences about variable-to-metric relationships and 106,927 variable-to-variable relationships. As mentioned, our goal was to infer what variables could affect the landed ellipse center distance from target. These inferences are done given some knowledge about what parameters are known to be correlate to the metric under study.

When examining the ellipse center distance to target in the KG, we can see that the metric is defined by downrange and crossrange from target. We can also see that it is correlated to total range at touchdown, which essentially the same, as it already has combined crossrange and downrange into one variable using the Pythagoras theorem. Furthermore, we can see that Ellipse center to distance is immediately related to metrics such as downrange at parachute deploy and Terrain relative navigation performance. For example, the Range Trigger employed by the Mars 2020 EDL sequence helps shrink the ellipse size around the landing target given that it uses its navigated position to deploy the parachute as opposed to velocity. TRN on the other hand, gives the system knowledge about its location with respect to the terrain and helps direct the spacecraft to a safe area in the vicinity.

If we expand the graph, we can see that downrange at parachute deploy is also correlated to downrange at heading alignment. This is obvious given that downrange that during preceding events will determine the downrange of subsequent events. Further exploration also shows that downrange at heading alignment is correlated to entry metrics such as angle of attack at SUFR.

To facilitate the task of manually examining inference results made by PSL, we kept all inferences that contained variables in the system prior to backshell separation. We did so given that variables and metrics during the powered approach segment of EDL are highly coupled. For example, horizontal position at the start of powered approach will not be significantly different to the

position at touchdown. This is primarily attributed to the fact that this portion of EDL is mostly vertical flight. Furthermore, the powered flight segment lasts about a minute and a half, thus the time between events is short enough that seeing significant changes is difficult. At most, navigation errors during this portion of flight could affect touchdown velocities. In more extreme circumstances, landing in a hazardous area could result in a decreased probability of success. However, this is not the scope of this example. Nevertheless, previous studies have shown that the factors that contribute the most to the landing ellipse are atmospheric conditions (downrange direction), parachute aerodynamics (downrange direction), and attitude initialization errors (crossrange direction) [36].

When looking at the inferences made by PSL relating a variable to the ellipse center distance to target, we saw that the many of the most likely variables to be related to the metric under study include position errors and vehicle state variables (e.g. position, velocity and angle of attack) during previous events in the EDL sequence. Variables less likely to be related to the ellipse center distance from target were primarily propellant consumption at the start of the entry phase, RCS thruster firings, timeline margin, and altitude of navigation filter solutions, for example.

To facilitate the analysis, we will look at inferences by event in the EDL sequence. The two primary phases of entry are: range control and heading alignment. During range control, the entry vehicle adjusts its bank angle magnitude in order to minimize downrange error at parachute deploy. During this phase, the guidance algorithm also monitors crossrange to target and commands bank reversals to manage errors in this direction. During heading alignment, the vehicle's bank angle is commanded to steer towards the target coordinates. This command helps minimize crossrange.

Table 3.3 shows some of the inferences made by PSL for the range control phase. These results indicate that both navigated crossrange and downrange at range control, are more likely correlated to ellipse center distance to target at touchdown than propellant available at this phase. By inspection, we argue that both navigated and actual crossrange and downrange are expected to have an effect in our landed point, given that navigated errors in both directions at different times during entry will undoubtedly determine the landed positions. Given our model, guidance crossrange (based on the lateral angle) at range control is is also likely to be correlated to our

Variable	Metric	Inference Value
Propellant remaining at range control	Ellipse center distance to target	0.049
Guidance crossrange at range control	Ellipse center distance to target	0.267
Downrange at range control	Ellipse center distance to target	0.292
Downrange (navigated) at range control	Ellipse center distance to target	0.307
Crossrange at range control	Ellipse center distance to target	0.367
Crossrange (navigated) at range control	Ellipse center distance to target	0.586

Table 3.3: Inferences made for variables in range control

Variable	Metric	Inference Value
Pre-bank delta at heading alighnment	Ellipse center distance to target	0.001
Propellant remaining at heading alignment	Ellipse center distance to target	0.485
Lift Margin Error at heading alignment	Ellipse center distance to target	0.499
Guidance downrange error at heading alignment	Ellipse center distance to target	0.574
Downrange at heading alignment	Ellipse center distance to target	0.578
Crossrange at heading alignment	Ellipse center distance to target	0.655

Table 3.4: Inferences made for variables in heading alignment.

objective metric given that it essentially measures trajectory distance out-of-plane with respect to the reference trajectory. Thus, as we see, it is expected that the inference is in family with the other crossrange measures. As for fuel available at range control, we do not believe (in this model) it should have any effect in our landed points given that we know it does not affect landing ellipse.

Similarly, PSL infers that during heading alignment crossrange (actual) and downrange (actual) are correlated to our target metric, which as stated previously, is expected. We also see that it infers that guidance downrange error at the start of heading alignment, and land lift margin (up) are very likely correlated to ellipse center distance to target. Again, we see that propellant available during heading alignment is less likely correlated to our target metric. Furthermore, we also see that PSL believes that the delta between pre-bank and the first bank command is less likely to be correlated to our landed location. This low probability estimate came as a surprise given that in the KG we see that the pre-bank delta defines a metric that is correlated to the variable downrange, which defines ellipse center distance to target (Figure 3.8).

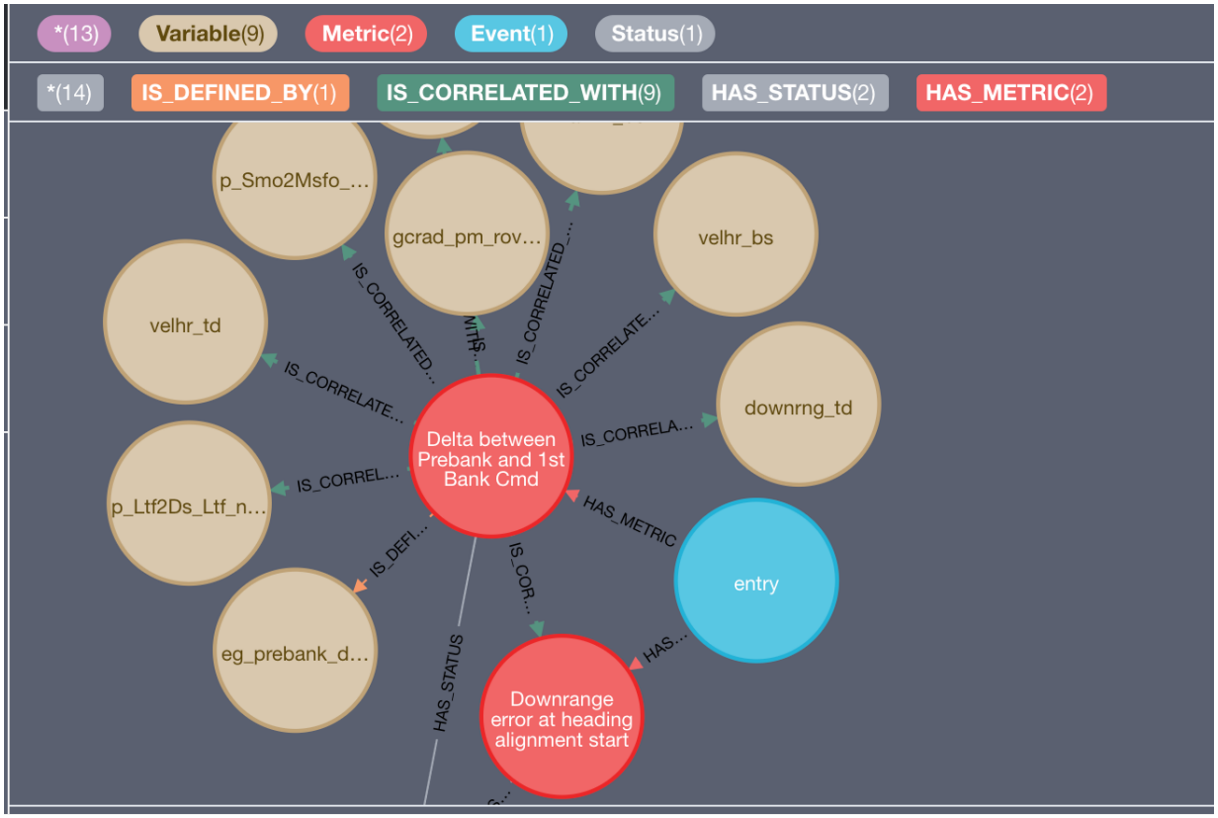


Figure 3.8: Subgraph with prebank metric.

Variable	Metric	Inference Value
Pre-bank delta at heading alignment	Downrange at touchdown	1.0
Pre-bank delta at heading alignment	Crossrange at touchdown	0.617

Table 3.5: Inference of pre-bank delta to ellipse center distance to target.

Given that our model also infers variable-to-variable relationships, we can use those inferences to further investigate variables of interest, such as the pre-bank delta. Due to computational cost, only a fraction of all possible variable-to-variable relationships were sampled and used to make inferences. Upon examination, the inferences of the pre-bank delta angle to downrange and crossrange variables did not make it to the final list of targets. Thus, this would explain why the inference value was lower than expected.

To test whether PSL could identify any relationships between the pre-bank variable and downrange and crossrange at touchdown, a second PSL model was evaluated using a lazy-inference approach. This model only contained rules dedicated to infer variable-to-variable relationships and included additional pairs of variables as targets (about 10,000 more). Having only one open predicate made it computationally feasible to increase the number of variable-to-variable inferences. Table 3.5 shows the inferences made regarding the delta between pre-bank and downrange and crossrange at touchdown- which define the ellipse center distance to target. Here, it is evident that PSL infers there is a possible relationship between the pre-bank delta and the metric under study given that the inference values for both metrics are high. Therefore, if additional computational resources would have been available, a larger sample of observations and targets could have been incorporated in our model. Including such information could have helped capture relationships that are 1) expected and 2) are new and potentially interesting. Furthermore, being able to incorporate more observations and targets could have potentially reduced the number of false negatives due to lack of evidence.

Following heading alignment, the entry vehicle conducts the “Straighten Up and Fligh-Right" Maneuver (SUFR)". Here, the vehicle ejects its balance masses to remove the angle of attack, such that the TDS looks directly to the ground. For variables in this event, PSL inferred that the system’s

Variable	Metric	Inference Value
Propellant available at SUFR	Ellipse center distance to target	0.112
Guidance crossrange capability	Ellipse center distance to target	0.364
Entry flight path angle at SUFR	Ellipse center distance to target	0.407

Table 3.6: Inferences made for variables in SUFR.

guidance crossrange capability and flight path angle at this point, are the most likely correlated to the ellipse center distance to target metric. As expected, it estimates that propellant available is not correlated to out target metric as per our model. These results are shown in Table 3.6

Following SUFR, the system is ready to deploy its parachute using the range trigger. Here, we see some more interesting inferences made by PSL. On Table 3.6, we see that downrange and crossrange at parachute deploy are among the most likely to be correlated to the ellipse center distance to target, this has been consistent in all events, and expected. Moreover, we see that there is a very high inference value for attitude rates at parachute deploy, dynamic pressure, thermal load, and altitude of parachute deploy. However, even if the aforementioned variables had strong inference values relative to other parameters during this event, all inferences can be considered fairly significant. We say this given that we have seen during previous events much lower likelihood estimates. This is also not surprising given that parachute deployment conditions are difficult to model and they are highly subjective to major sources of uncertainty, primarily atmospheric conditions. Furthermore, we know for a fact from previous studies that deployment conditions largely affect the uncertainty ellipse in the downrange direction.

Seeing attitude rates at parachute deploy with a high inference value was interesting because many possible factors affect this parameter during parachute deployment. Primary factors that influence attitude rates at parachute deploy are atmospheric winds and parachute aerodynamics. These factors could help explain why we see a high inference value for dynamic pressure at parachute deploy. According to our model, we see that this is true (inference value of 0.99). However, seeing a thermal load parameter was more of a surprise given that our PSL model had a constraint that indicated that thermal parameters should not affect our landing points.

Nevertheless, given that thermal parameters are associated with vehicle aerodynamics, PSL still gives this parameter a high likelihood of being associated with our target metric. We also see in these results that altitude at parachute deploy has a high inference value. Again, this result is expected given that the altitude at which the parachute deploys will significantly affect the loads experienced during this segment. This coupling is evident in Figure 3.9. In the figure; we see that altitude, time, and thermal loads are correlated to several metrics that measure the loads experienced during this event. This coupling would explain why many of the inference values are in family.

Furthermore, we see that other aerodynamic parameters are believed to be correlated to our landing ellipse distance to target that include, relative velocity, angle of attack, flight path angle, peak deceleration, and Mach number. Like the aforementioned, these inferences do not come as a surprise. This statement can be repeated for longitude and declination at parachute deploy given that, like crossrange and downrange, they measure the entry vehicle's position. In other words, they measure the same parameter.

3.3 Conclusion

This chapter introduced a new method to combine expert knowledge and data driven-analysis to infer relationships between EDL variables and metrics. In our framework, we used a knowledge graph to capture how EDL variables, metrics, events, and components interact. Our knowledge graph was created using multiple sources that included tabular, numerical, and object-oriented data. A Scorecard template was used to map segments to events in the EDL sequence; events to metrics; and metrics to variables that define them. On the other hand, correlation analysis was used to establish relationships between EDL variables and metrics across all events in the sequence. Pairs with a high linear correlation coefficient were connected using a relationship type of *IsCorrTo*. Furthermore, the knowledge graph included information from SysML models (originally in JSON format).

An SRL framework called PSL was employed to infer relationships between a variable and a metric. This framework uses first-order logic to capture beliefs about the relational dependency

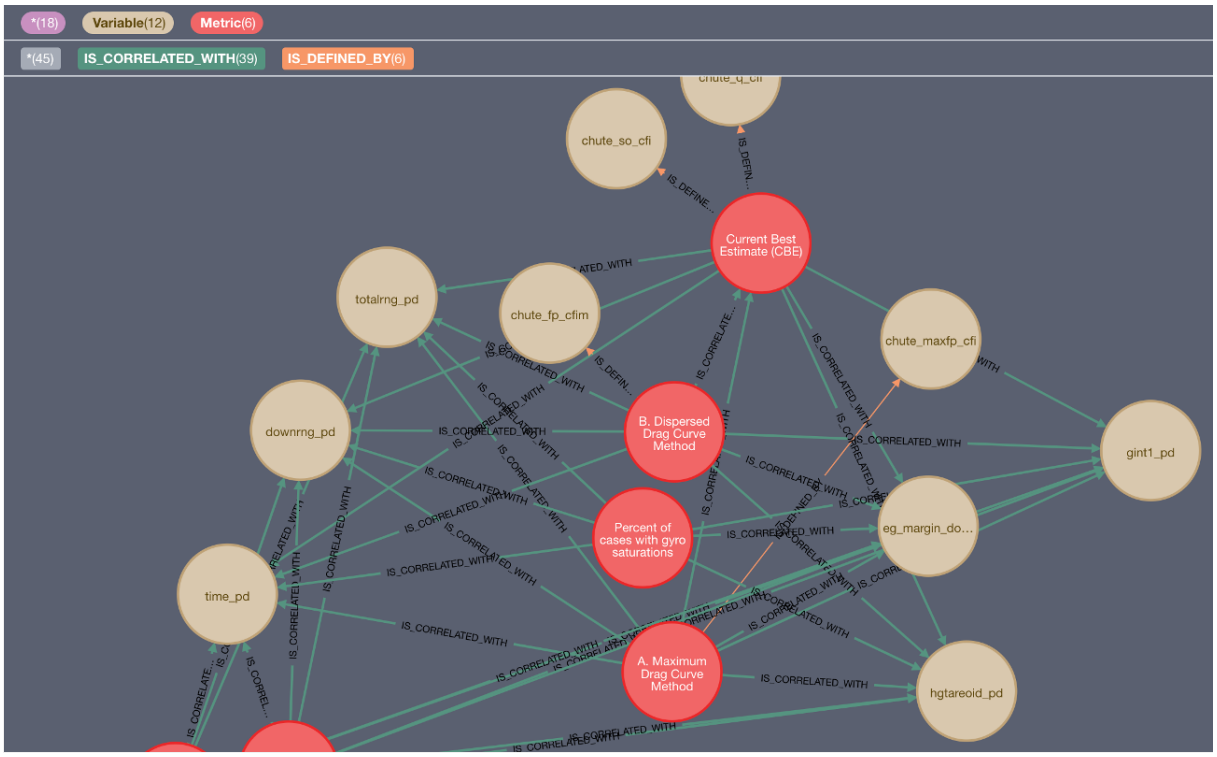


Figure 3.9: Subgraph with several parachute deploy metrics.

Variable	Metric	Inference Value
Attitude knowledge error at parachute deploy	Ellipse center distance to target	0.172
Angle of attack at parachute deploy	Ellipse center distance to target	0.192
Mach at parachute deploy	Ellipse center distance to target	0.257
Declination at parachute deploy	Ellipse center distance to target	0.267
Downrange (navigated) at parachute deploy	Ellipse center distance to target	0.274
Propellant available at parachute deploy	Ellipse center distance to target	0.295
Flight path angle at parachute deploy	Ellipse center distance to target	0.451
Peak deceleration at parachute deploy	Ellipse center distance to target	0.477
Relative velocity at parachute deploy	Ellipse center distance to target	0.511
Time at parachute deploy	Ellipse center distance to target	0.515
Longitude at parachute deploy	Ellipse center distance to target	0.584
Crossrange (navigated) at parachute deploy	Ellipse center distance to target	0.609
Altitude at parachute deploy	Ellipse center distance to target	0.612
Crossrange at parachute deploy	Ellipse center distance to target	0.632
Heat load at parachute deploy	Ellipse center distance to target	0.657
Downrange at parachute deploy	Ellipse center distance to target	0.674
Dynamic pressure at parachute deploy	Ellipse center distance to target	0.689
Attitude rates at parachute deploy	Ellipse center distance to target	0.743

Table 3.7: Inferences made for variables at parachute deploy.

between the nodes in the graph. PSL makes inferences by using MPE to find truth values of atoms in a rule that maximize the likelihood of the rule being satisfied. In the example shown, PSL was used to infer what variables during entry and parachute descent might affect the landed ellipse center distance to the target.

As discussed in Section 1.4.2.2, in recent years, NASA has placed significant effort into developing a unified framework with models that enable expressivity for evaluating complex systems, facilitate trade-off analysis and enable interoperability between disciplines. Furthermore, a unified modeling framework can also enable large-scale re-usability of entities across projects. Employing these capabilities throughout development and later lifecycle phases may help reduce lifecycle time and cost while also reducing the likelihood of late discovery of design problems.

NASA aims to use MBSE to reduce systems engineering practices that rely on document tracking and standalone analysis. For example, the Mars sample return mission will use MBSE to construct an integrated model (e.g. SysML) that contains functional decomposition of the system, operational scenarios, and requirements and traces to model elements. These models are incorporated into their IMCE ontology framework which whose goal is to perform logical reasoning across mission elements.

Here, we have taken a different approach and propose using knowledge graphs instead of ontologies to integrate domain knowledge from distributed data holdings. We selected KGs due to their scalability, interpretability, and ability to incorporate data valuable to describe relational properties of a given domain. Furthermore, there is a wide variety of frameworks, like PSL, for performing logical reasoning about relational data that exhibit uncertainty. Therefore, these frameworks can help experts make judgments in the face of imprecise, incomplete, or uncertain information.

Our work used a case-specific scenario to demonstrate that MBSE can provide domain knowledge of the interconnections of the system's structure, behavior, requirements, and parametrics across models. These models provided information on the EDL architecture (e.g., subsystems, interactions, and dataflows) and how they relate to metrics. In the case illustrated, the graph has to be provided by the user. However, we hope that given current trends in model-based systems

engineering, in the future, it could be automatically extracted from MBSE artifacts such as lessons learned databases and SysML models generated during the mission development process. An interesting application would be linking how metrics link to decision-making criteria that can affect schedule and operations. Furthermore, such a framework can incorporate mission-critical information across different disciplines and phases of a mission life cycle.

Future work should consider using Named Entity Recognition (NER) to extract entities (e.g., subsystems, components) and their relationships to other aspects of the system (e.g., events, timeline) to extract knowledge from documents. NER can help automate and facilitate the task of knowledge graph construction. The NER approach was explored during the creation of our knowledge base. However, due to the lack of documents available, the performance of the NER algorithm was poor. Consequently, constructing the KG relied heavily on a manual approach to include relationships from knowledge extracted in papers.

4. IMPROVING RULE MINING FOR ENTRY, DESCENT, AND LANDING SIMULATIONS USING KNOWLEDGE GRAPHS AND PROBABILISTIC SOFT LOGIC*

4.1 Introduction

Rule learning algorithms have become a popular method for discovering patterns in large datasets. This family of algorithms extracts patterns expressed as logical rules using *if-then* statements. These statements are used to map observations to outcomes such as: "IF entry mass increases, THEN Mach at parachute deploy increases." Logical rules have been widely employed as knowledge representations in artificial intelligence, since foundational work by Newell and Simon proposed them as a model to mimic how humans reason [20, 21]. Association rule mining algorithms extract logical rules from a dataset by looking for frequent patterns in the data (e.g., "many" simulations show high entry mass and high Mach at parachute deploy). One drawback of rule mining algorithms is that they often produce many association rules [22]. Furthermore, often times these rules are too complex, too obvious, or make little sense to subject matter experts [23, 24, 25]. These limitations make it difficult for experts to process the content in all of the rules, identify interesting rules, interpret findings, and use the information for decision making.

Given the limitations of rule mining, many attempts have been made to develop measures that reduce the number of rules provided provided by the algorithm, making it easier for subject matter experts to manually examine results. These are known as "interestingness measures." Moreover, some measures have been developed to help steer the subject matter expert's attention towards rules that are more likely to be potentially interesting and useful to them. The explanation of EDL simulations can be seen as an iterative hypothesis testing process, in which candidate explanations for a behavior seen in the data are generated and examined or tested. In this context, a rule can be useful to provide support for a hypothesis from the user (e.g., "I think what is driving these cases landing so far down-range is that higher tailwinds during parachute deploy are increasing its

*Parts of this chapter have been adapted from "Interactive Explanation of Entry, Descent, and Landing Simulations" (2020) [1] by Santini De Leon, S., Selva, D., and Way, D. with permission from AIAA

drift"), or to identify a new candidate hypothesis for the subject-matter expert to examine.

Regardless of the measure employed, learning and interpreting information encoded in logical rules remains a challenge. For example, a rule may contain many literals and combinations of conjunctions and disjunctions that make it difficult to understand. Over the years, very little work has been done to improve rule comprehensibility. Most past works assess rule comprehensibility based on its number of attributes exclusively. The literature argues that if a rule is concise (i.e., few attributes), it is easily comprehensible [23, 105]. While conciseness can contribute to understanding a pattern, this measure neglects human learning factors such as chunking [106, 107]. Therefore, if the elements in a set of rules do not appear to have association with one another, a human is less likely to extract any useful information.

To partially tackle the limitation of comprehensibility, some work has suggested incorporating domain knowledge for identifying interesting rules. Most work in the literature employs some comparison mechanism between rules and domain knowledge encoded in an ontology. The work depicted in [98] uses semantic distance between concepts in an ontology present in a rule to prune mined rules. The framework uses general impressions provided by users to further prune the rules. General impressions refers to beliefs a user may have about the association between items. For example, a user may believe there exists a relationship between eggs, bread, and butter among the transactions in a database. Semantic distance is estimated by evaluating the minimum distance between the concepts in the antecedent and the consequent of a rule (i.e, the minimum path that connects the two items in an ontology) [99]. This mechanism, however, does not consider that a large distance between two items may imply that there is no real relationship between them, thus, the rule might not make sense. In the work discussed in [25, 98], rules that match beliefs provided by the user are pruned, considering that the user would not find those rules interesting since they already know them.

To help tackle these limitations, we use a knowledge graphs and probability soft logic to improve the comprehensibility, insightfulness, and usefulness of mined association rules. With the proposed method, a knowledge graph is used to capture knowledge in the context of the domain

or study. This knowledge graph has to be provided by the user, but we hope that given current trends in model-based systems engineering, in the future it could be automatically extracted from MBSE artifacts such as lessons learned databases and SysML models generated during the mission development process. With this information, we use probabilistic soft logic, a statistical relational learning framework, to infer the probability of there being a relationship between pairs of items in a rule, given the knowledge graph. We then derive a measure of consistency between a rule and the knowledge graph based on those probabilities. We hypothesize that rules that are more consistent with the knowledge graph according to our metric will be more comprehensible than rules with lower consistency with the knowledge graph, and therefore more useful.

To formally test this hypothesis, we conducted a small human subject study (N=6 subjects) where we presented experts with various rules extracted from EDL simulations and asked them about their degree of comprehensibility. In addition, we also asked them about the insightfulness and usefulness of each rule, to test additional hypotheses related to those attributes. The subjective perceptions of these attributes for each rule were compared to our measure of consistency with knowledge of the system and with standard data-derived statistical measures of the quality of a rule (lift). Results support our primary hypothesis and further show that all three attributes (comprehensibility, usefulness, insightfulness) are correlated with one another and more correlated with our measure of consistency than with lift.

The rest of this chapter discusses how we used the model and information generated in Chapter 3 to assess different qualities of association rules. This chapter also discusses the human subject experiment and presents the results obtained in the experiment, both quantitative and qualitative. This chapter provides a summary of the work and discusses its contributions, limitations and future work.

4.2 Framework Description

Figure 4.1 shows the proposed framework for evaluating rules generated by rule mining algorithms to assess their comprehensibility, insightfulness, and usefulness. As shown in the figure, the goal our goal is to use rule mining algorithms, such as Apriori, to generate rules for an EDL

dataset. The end goal is to use these rules to help explain EDL simulation outputs [1]. I In our framework, the knowledge graph provides evidence, or observations, to the PSL model. From the rules, we extract all pairs of triplets and we provide them as targets to the PSL model. Therefore, if a relationship is not explicit in the knowledge graph, PSL will evaluate whether there exists a relationship between the entities in the target set. Once all inferences are made, we map probabilities to rules. If a rule has a single item, we extract the probability of that item being linked to the consequent from the inference set, for example. If the rule has more than one item in the antecedent, we extract probabilities between each entity in the antecedent and the consequent as well as the inference value between pairs of entities in the antecedent. These probabilities could then be combined to form metrics of how consistent a rule is with a KG. We later test hypotheses related to the correlation between this consistency metric and the comprehensibility, insightfulness and usefulness of the rule.A consistency can be the minimum, maximum, or average probability of items in a rule being linked in the knowledge graph. However, for the sake of our hypothesis, we looked at rules whose pairs of items all had high or low values.

We hypothesize that rules that are more consistent with the knowledge graph according to our metric will be more comprehensible than rules with lower consistency with the knowledge graph. We will also explore whether rules with low probabilities between its items and the consequent, might result in more insightful and useful rules given that they may contain relationships that are unexpected based on domain knowledge. However, there is also a possibility that many of the “low probability rules" are not comprehensible. How these rules are perceived will be studied in later sections.

4.2.1 Link Prediction using Probabilistic Soft Logic

In this chapter we use the PSL model discussed in Chapter 3. In our work, we use a PSL model to infer the probability of there being a relationship between pairs of items in a rule, given the knowledge graph. Table 3.2.2 presents the rules used to construct the PSL model. For this study, the goal was to explain cases where the spacecraft lands far from the intended landing target, and thus the model tries to find associations between variables and the offset of the landing ellipse from

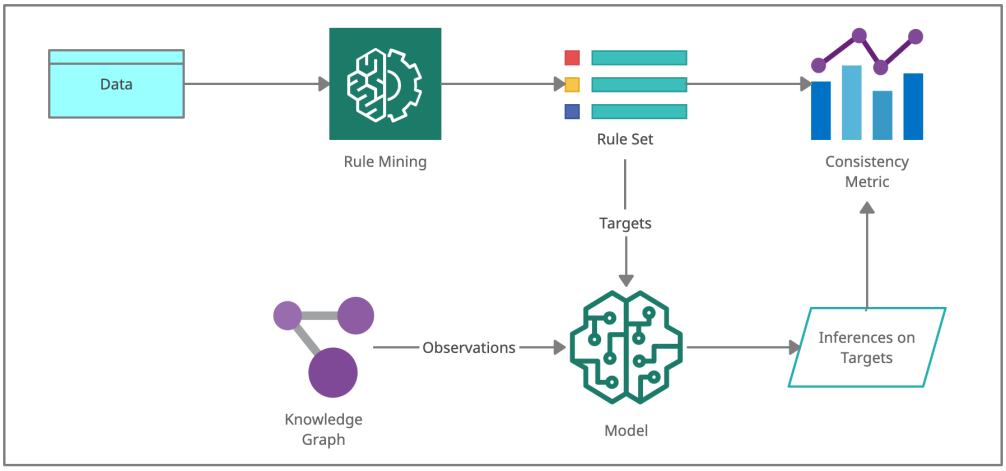


Figure 4.1: Framework overview.

the target. Although our knowledge graph contains a variety of links, we will focus on inferring relationships of the type *IsCorrTo*.

4.2.2 Combining PSL Inferences to assess consistency of a rule with the KG

Given simple association rule $A \longrightarrow B$, we use the PSL model described in the previous section to assess how items in the antecedent and the consequent of a rule are related. For rules with a single item in the antecedent, the consistency metric is simply the PSL probability estimate generated. If a rule infers a variable-to-metric relationship P_{AB}, is provided by the open predicate $IsCorrTo(A, B)$, whereas if the rule maps a variable-to-variable relationship, then the probability is inferred by the open predicate ($VarCorsVar(A, B)$). For example, for the rule $Lowdynamicpressureatparachutedeploy \longrightarrow Downrangeattouchdownislow$, is simply described by $P_{AB} = 0.69$. A compounded rules such as $A\&B \longrightarrow C$, however, is described has 3 relevant probabilities P_{AB}, P_{AC} and P_{BC}. These probabilities are obtained by ($VarCorsVar(A, B), IsCorrTo(B, C), IsCorrTo(A, C)$), respectively. For the rule, $Lowdynamicpressureat$ $Downrangeattouchdownislow$ we get $P_{AB} = 0.93, P_{AC} = 0.29$ and $P_{BC} = 0.67$. Depending on the consistency metric selected by the user, they can for example, average these probabilities or use the minimum or maximum. In this paper, we limit ourselves to rules whose that contain conjunc-

tions of two items in the antecedent and do not use a particular consistency metric and to evaluate our hypothesis, we looked at rules that had a high or low probability estimates overall. Nevertheless, the idea is extensible to more general cases with the consistency metric of choice. For rules with a larger number of conjunctions simply require aggregating more probability estimates.

Given our assumptions, if all three probabilities estimated are high, any reasonable aggregation of the probabilities would lead to a high KG-consistency metric. We hypothesize that these rules are more comprehensible rules. When at least one of the probabilities is low, there are three main cases to consider. The first case is that one of the items in the antecedent having some relationship is low – given our knowledge. This could be an indicator of a previously unknown relationship, or it could also indicate that there is actually no relationship between the items. Second, it could be that items in the antecedent have a clear relationship (P_{AB} is high) but, there is a small probability that one or more items in the antecedent is related to the consequent. Third, we can observe that all three probabilities estimated are low. All of these cases could indicate to some extent that the rule is not very comprehensible given the unclear relationship between all items.

4.3 Human Subject Study

The general hypothesis of this work is that there exists a correlation between consistency metrics, comprehensibility, insightfulness, and usefulness of a rule. To validate our approach and evaluate the hypothesis, we conducted a survey with N=6 engineers at the NASA Langley Research Center (LaRC) and the Jet Propulsion Laboratory. Some of the experts were EDL trajectory analysts (3) and the others were EDL systems engineers (3). Both groups of experts work closely together on a regular basis. Trajectory analysts are primarily in charge of executing and analyzing EDL simulated trajectories using NASA's Program to Optimize Simulated Trajectories (POST2). Systems engineers, on the other hand, use data products generated by trajectory analysts to ensure EDL system performance requirements are being met. Both systems engineers and simulation analysts collaborate to monitor and test different performance metrics for potential bottlenecks, identify possible solutions, and work with different subsystem engineers to implement solutions [26]. Examples of decision could be changing the radar sequence of the terminal descent sensor se-

quence, or deciding whether EDL parameter updates are necessary during the vehicle's trajectory to Mars.

4.3.1 Dataset and Rules

To demonstrate the framework, we used a nominal EDL Mars 2020 Monte Carlo Simulation output. This Monte Carlo dataset contains 8,001 random trajectories each containing over 15,000 variables. To reduce the complexity of the dataset, we only used variables that define metrics in the scorecard given that these are considered a good summary of system performance. For the study, we also excluded all events that occur after backshell separation given that events that occur during the last minute prior to landing have no effect on the uncertainty ellipse. These filters reduced the dataset to 96 variables. The In the study, Apriori was used to generate association rules. The algorithm was set to find frequent itemsets that occur together at least 10% of all simulation cases and rules were constrained to have 2 items in the antecedent and a minimum confidence of 0.2. The algorithm produced around 3,400 rules where the downrange (West-East) at touchdown is the consequent. A large downrange towards West (positive downrange) means that the vehicle flew short, whereas a negative downrange means that the vehicle flew past the target.

4.3.2 Experimental Design

The experiment was designed to be within-subjects. Each participant provided their opinion on comprehensbility, insightfulness, and usefulness on 15 rules, shown in Table 4.1. To test our hypothesis, we selected 5 rules where PSL provided high probability estimates between all items in the rule, and data mining provided high lift (rules 1-5). We also selected 5 rules where PSL probabilities were high and the lift was low (rules 11-15). To contrast, we selected 5 more rules where the PSL estimates were low and the lift was high (rules 6-10). This resulted in a total of 15 rules.The third column on Table 4.1 contains the estimated lift of rules produced by the Apriori algorithm. The fourth column contains the prediction on whether each variable in the antecedent could be correlated to the target metric (Ellipse center distance to target (downrange), respectively. Finally, the last column contains the inference value generated by PSL on whether the variables in

the antecedent might be correlated.

In this paper, we selected lift to be compared to KG consistency metrics because it provides a numerical value that describes how the antecedent of a rule affects the consequent. As shown in Equation 4.3.2, the lift is essentially the ratio of confidence of the rule and the support of the antecedent [153]. In other words, the lift indicates up to what extent X and Y are independent [162]. A lift value greater than 1 indicates a positive correlation between X and Y whereas a value less than 1 indicates a negative correlation. A lift value of 1 indicates that the probability of occurrence of the antecedent and the consequent are independent of each other.

$$Lift(X \longrightarrow Y) = \frac{P(Y|X)}{P(Y)} \tag{4.1}$$

To facilitate the analysis and interpretations of rules, we used low/mid-range/high to describe the numerical attribute of the metric. Originally, the dataset was discretized and variables were accompanied by a numerical range the attribute belonged to. For example, a variable/metric is *high* if its value is above the 75th percentile of all the 8,001 values in the simulation. Mid-range indicates that values lie between the 25 and 75th percentile. Low indicates that the value for that variable is in the lower 25 percentile.

It must be noted that when measuring crossrange and downrange, values may be positive or negative- depending on the direction of flight. Therefore, if either one of these metrics is in the mid-range, it means that these landing spots are at a close distance of the target point (North and South or East and West) . For example, in this particular simulation, crossrange values in the mid-range were points between 140m South and 500m North. Therefore, when looking at crossrange and we refer to mid-range, it means that it is close to the target but values go both directions. Downrange values, on the other hand, describe landing points in the West-East direction. Points that landed short of the target are described as Low/High (short) and points that fly past the target are described as Low/High (long).

When prompting experts to assess the extent to which each rule satisfied the three criteria, we

Table 4.1: Rules for EDL experiment.

Rule No.	Rule	Lift	V-M	V-V
1	"IF Crossrange at touchdown is high (south) AND Relative Velocity at Parachute Deploy is in the mid-range THEN Downrange at touchdown would likely be Low (short)"	1.49	[1.0, 0.51]	0.80
2	"IF Altitude at Parachute Deploy is in the mid-range values AND the Downrange at Parachute Deploy is High (short) THEN Downrange at touchdown would likely be high (short)"	2.46	[0.61, 0.67]	1
3	"IF Crossrange at Touchdown is in the mid-range values AND the Lift Margin (Up) at Heading Alignment is low THEN Downrange at touchdown would likely be high (short)" .	2.25	[1.0, 0.49]	1
4	"IF Downrange at Parachute Deploy is high (short) AND the Flight Path Angle at Parachute Deploy is steep THEN the Downrange at touchdown would likely be high (short)" .	2.94	[0.67, 0.45]	1
5	"IF the Downrange at parachute deploy is high (short) AND the Relative Velocity at Parachute Deploy is low THEN the Downrange at Touchdown would likely be high (short)" .	2.64	[0.67, 0.51]	1
6	"IF the Attitude Rate Watermark (at HS+ 5s BS) is in the mid-range AND the Dynamic Pressure at Heatshield Separation is high THEN Downrange at Touchdown would likely be high (short)" .	2.56	[0.08, 0.20]	0.02
7	"IF Parachute Inflation Loads are in the mid-range AND Dynamic Pressure at Heatshield Separation is in the mid-range THEN Downrange at Touchdown would likely be high (long)" .	2.64	[0.09, 0.20]	0.08
8	"IF Dynamic Pressure at Heatshield Separation is high AND the Range Rate Between Descent Stage and Heatshield (at 230 m Range) is high THEN the Downrange at Touchdown would likely be high (short)"	2.43	[0.52, 0.20]	0.19
9	"IF Relative Velocity at Parachute Deploy is low AND Propellant Used at Entry Interface is in the mid-range THEN Downrange at Touchdown would likely be high (short)" .	2.63	[0.51, 0.06]	0.88
10	"IF Peak Shear (aerothermal) is high AND the Capsule RSS Angular Acceleration Watermark is in the mid-range THEN Downrange at Touchdown would likely be low (short)" .	1.22	[0.20, 0.51]	0.04
11	"IF Navigated Downrange at Parachute Deploy is low (short) AND Relative Velocity at Parachute Deploy is high THEN Downrange at Touchdown would likely be low (short)" .	0.74	[0.27, 0.51]	1
12	"IF Flight Path Angle at Parachute Deploy is steep AND the Flight Path Angle at SUFR is steep THEN Downrange at Touchdown would likely be low (short)" .	0.89	[0.49, 0.36]	1
13	"IF Time Exposure to Aerial Oscillations is in the mid-range AND Crossrange at Parachute Deploy is in the mid-range THEN Downrange at Touchdown would likely be low(short)" .	0.90	[0.41, 0.63]	0.87
14	"IF Parachute Loads are high AND Altitude at Parachute Deploy is low THEN Downrange at Touchdown will be low(short)" .	0.89	[0.43, 0.61]	1
15	"IF Lift Margin (Down) at Heading Alignment is in the mid-range AND Downrange at Heading Alignment is in the mid-range THEN Downrange at Touchdown would likely be low (short)" .	0.95	[0.33, 0.57]	0.36

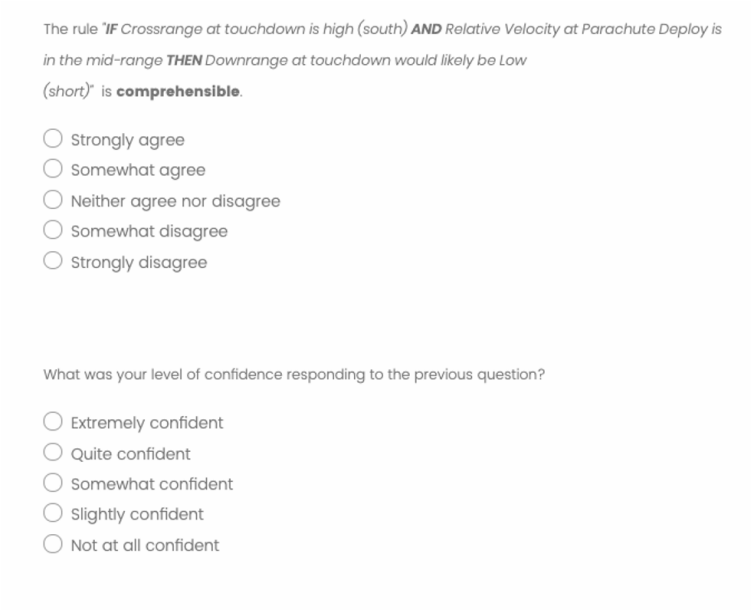


Figure 4.2: Sample from survey.

asked that they respond according to the definitions provided in Table 4.2. Each question contained a statement indicating that the rule described possessed a quality (comprehensible, insightful, or useful). Users were prompted to provide their opinion in agreement or disagreement by selecting one of the 5 Likert-type responses. We used a symmetrical scale around a neutral mid-point. Experts were also prompted to respond about the level of confidence for each response provided. The responses were also Likert-type. Given that experts had to assess the three criteria on all 15 rules and their level of confidence in each response, the survey contained a total of 90 questions. At the end of the survey, experts were asked to provide comments about the survey, or discuss any challenges they had in providing responses. Figure 4.2 shows a snapshot of the two types of questions included in the survey. The survey was distributed electronically using an anonymous link so the authors could not link survey responses to individuals.

The primary hypothesis of this study is that consistency metrics will be positively correlated to comprehensibility. Therefore, our hypothesis is: *H1: Rules with high KG-consistency are more comprehensible*. To test this hypothesis, data points from G1 and G3 were grouped to describe rules that are consistent with the KG. Therefore, G1 and G3 was compared altogether (10 rules) against

Criteria	Description
Comprehensbility	A rule is comprehensible if you can understand the pattern. A rule is comprehensible if the pattern makes sense A rule might not be comprehensible if the parameters in the rule clearly do not affect each other
Insightfulness	A pattern is insightful if it sheds light. A pattern is insightful if it shows an interesting combination of parameters that you would not think of looking at together. A pattern is insightful if the predictive power of the rule is unexpected.
Usefulness	A pattern is useful if it helps you identify important factors and provides a physical explanation of what affects the metric under study

Table 4.2: Definitions provided to users about criteria studied.

rules in G2 (5 rules)- which were rules considered less consistent with the KG. Furthermore, the two following hypothesis were studied to help identify a link between proposed consistency metrics to insightfulness and usefulness of rules as perceived by the user: H2: Rules where knowledge and data mining estimates disagree are more insightful. H3: Rules where knowledge and data mining estimates disagree are more useful. Disagreements between knowledge are instances where the rule is considered consistent with the KG (e.g., all probabilities between items are high) but the statistical measure of lift indicates that there is no positive correlation (G3). Rules in G2 show weak consistency values with the KG but high lift values- indicating a positive correlation between the antecedent and the consequent.

Although two of our hypothesis focus on discrepancies between consistency metrics and data-driven measures to assess if the rule is useful or insightful, we will also study whether rules consistent with the knowledge graph turn out to be more insightful and useful than rules with low consistency. This will be assessed independent of the data-driven measure. In Sections 1.4.1 we argued that although a rule's statistical strength is high (e.g., confidence, lift), it does not imply that the rule makes sense in the context of the domain. Consequently, we will perform additional analysis to demonstrate that in some instances, rules with strong statistical strength are not comprehensible and might not be insightful or useful.

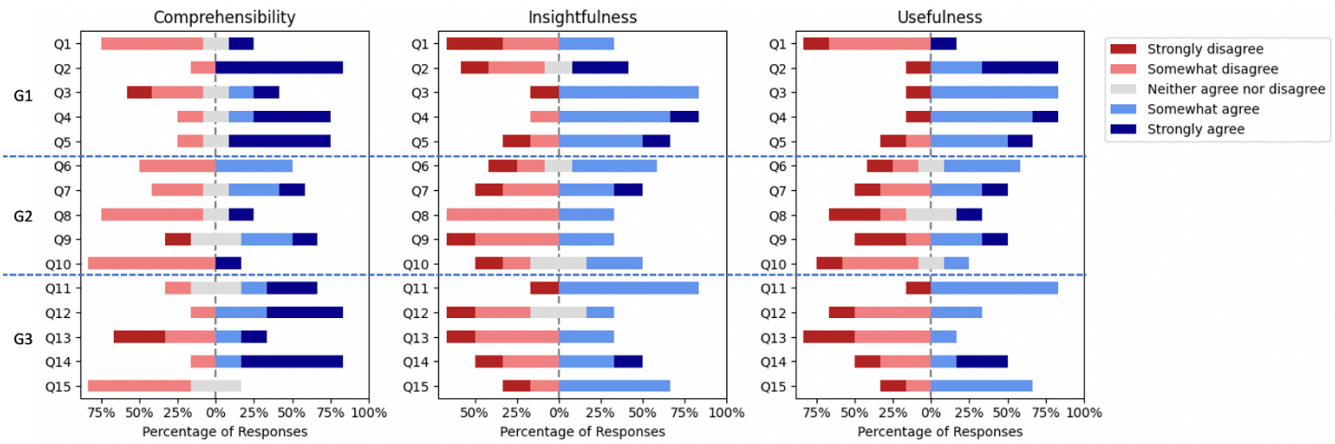


Figure 4.3: Overview of all responses.

4.4 Results

4.4.1 Overview

Figure 4.3 shows the frequency of responses among all subjects for all three criteria studied. In the first plot, we see that rules with KG-consistency (G1 and G3) are perceived as comprehensible. In G2 (low consistency with knowledge graph), however, we see that opinions on whether this group of rules are comprehensible are more balanced. Still, a fair number of subjects found rules in this group comprehensible, which is not what we expected and may indicate either that the KG is incomplete or that the lack of relation between the items in the rule was not a critical factor driving comprehensibility for those subjects.

The second plot in Figure 4.3 shows a similar trend: rules consistent with the knowledge graph (G1 and G3) are more frequently perceived as more insightful than rules not consistent with the (G2) knowledge graph. Moreover, in some instances rules in G2 were perceived as comprehensible (rules 6 and 7) and they were considered insightful and useful. A similar trend is observed for the usefulness of rules, which suggests that there might exist a correlation between comprehensibility, insightfulness, and usefulness. The following subsections will evaluate our hypotheses and will examine in more depth these results.

4.4.2 Statistical Analysis

To perform statistical analysis, we assigned numerical values to each Likert-like response. Responses where experts completely agree with the statement at hand receive the highest score (5) and the lowest score (1) was given to responses where they strongly disagreed with the statement.

4.4.2.1 Hypothesis 1: Rules with high KG-consistency are more comprehensible

To test H1, we performed a Mann-Whitney U test to assess whether the perceived comprehensibility of the rules between rules consistent with the knowledge graph (G1 and G3) is significantly different to those with lower consistency (G2). Figure 4.4 shows the box plots with the median response (red line) and range of responses (horizontal lines at the end of whiskers) for all responses obtained. Application of the Mann-Whitney U-test indicated that there is a significant difference in comprehensibility between both group of rules ($U = 689$, $p = 0.030$). Therefore H1 is supported by the data.

Although results indicate that rules with high PSL inference values are perceived as significantly more comprehensible, there were still more rules in G1 and G3 labeled as incomprehensible than we expected. Similarly, there were more subjects that found groups with weak PSL inferences (hypothesized to be less comprehensible) more comprehensible than we expected. Of note, rules 1, 3, 13 and 15 leaned more towards “incomprehensible" and one common characteristic of these rules is that they contain items whose numerical attribute is in their mid-range values. For example, rule 1 expresses that: *‘‘IF Crossrange at touchdown is high (south) AND Relative Velocity at Parachute Deploy is in the mid-range THEN Downrange at touchdown would likely be Low (short)"*. It is plausible that values with this mid-range attribute might not help explain how downrange could be at the two extremes of high or very low. Some subjects commented during and after filling out the survey that rules with statements in mid-range in the antecedent and non-midrange attributes in the consequent were confusing. Thus, these rules contain special instances where items with certain attributes, although statistically significant, might make a rule less comprehensible to some subjects. Nevertheless, this does not imply that there does not exist

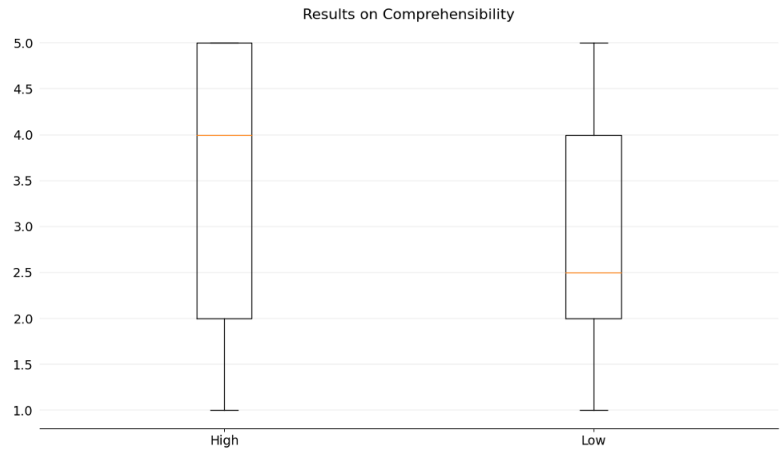


Figure 4.4: Comprehensibility for rules with high and low PSL inference values.

a relationship between these variables and the target metric. Note that some of these rules (3 and 15) were still considered insightful and useful by a majority of the experts.

4.4.2.2 *Hypothesis 2: Rules where knowledge and data mining estimates disagree are more insightful*

Our second hypothesis states that rules where PSL inference values do not align with statistical measures, such as lift from data mining, are more insightful. Our intuition was that this situation could model a rule that was unexpected, contradicting previous beliefs, which could be considered more insightful. To study this hypothesis, we compared rules in G1, where both PSL probabilities and lift are high, against all other rules, for which either PSL probabilities are low and lift is high (G2) or viceversa (G3). Results for the second hypothesis are shown in Figure 4.5. The median insightfulness was actually higher for G1 (agreement) than for G2+G3 (disagreement), which is the opposite of what we expected. In any case, the statistical test yielded a p-value of 0.108, so this effect was not significant. Consequently, H2 is not supported by the data.

Although results are not statistically significant, data seems to indicate that rules in G1 are more insightful overall. This group of rules' median response was 4 whereas rules with disagreements had a median response of 3 and 2.5, respectively. Results shown in Figure 4.6 (first plot) also seem to indicate that rules in G2 are the least insightful. Therefore, rules with poor consistency with

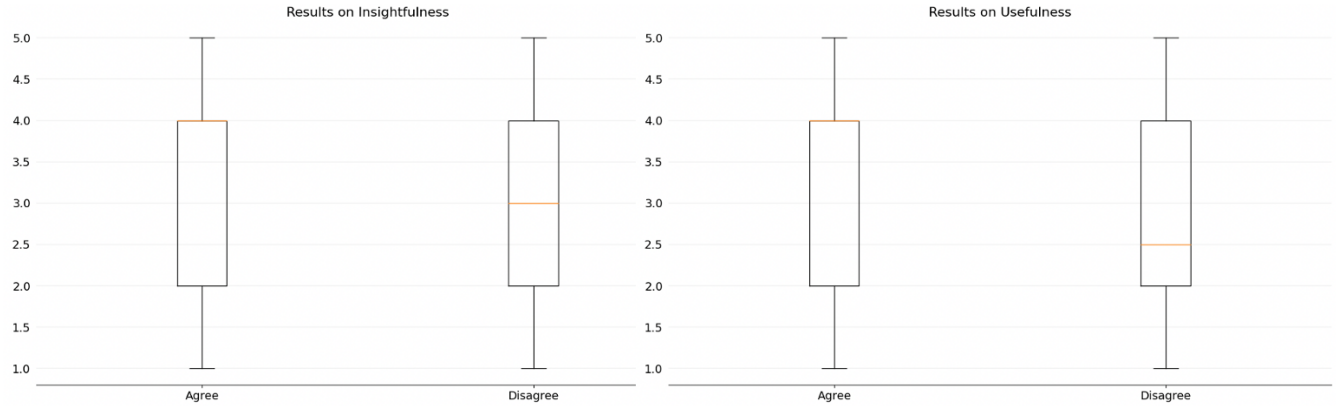


Figure 4.5: Usefulness and insightfulness of rules when knowledge-driven and data driven measure disagree.

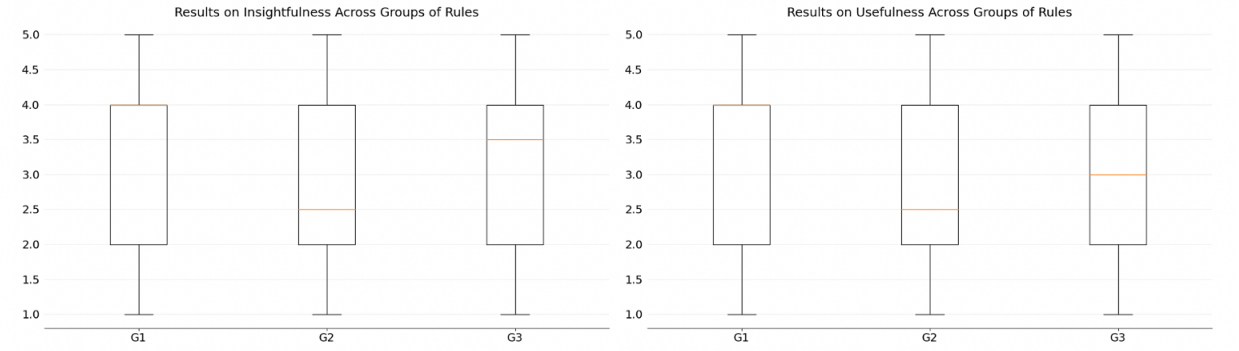


Figure 4.6: Perceived usefulness and insighfulness in all groups of rules.

the knowledge graph and high lift were considered the least insightful group of rules among all. The first plot in Figure 4.6 also indicates that rules consistent with the KG tend to be perceived as more insightful overall. Furthermore, we see that rules can be insightful when knowledge and data-driven measures disagree. This seems to be more likely for instances in which data does not support a rule statement (G3) but it is comprehensible. Thus, results continue to suggest that how insightful a rule is depends on how well it can be understood.

4.4.2.3 *Hypothesis 3: Rules where knowledge and data mining estimates disagree are more useful.*

Our third hypothesis is similar to H2, except we examine the usefulness of rules where knowledge and data disagree. The second plot in Figure 4.5 shows these results. Similar to H2, we see that on average, rules with KG-consistency and strong statistical measures are actually considered more useful than when both measures differ – i.e., again, the opposite of what we expected, but consistent with the results from H2. Thus, H3 is not supported by the data. Moreover, for this case, the Mann-Whitney U-test yields a p-value of 0.032. Thus, there is a significant difference between both groups of rules.

The second plot in Figure 4.6 also shows that rules in G2 are perceived as the least useful. This indicates that experts thought that these rules did not help them identify important factors or did not provide a good physical explanation as to why downrange is high or low. Results show that rules with high comprehensibility rating are considered the most useful.

4.4.2.4 *Additional Analysis*

The previous analysis showed that overall, rules with high KG-consistency are perceived as more comprehensible than rules with lower KG-consistency. Results also showed that rules with disagreements between knowledge-driven measures and data-driven measures are not necessarily more insightful or useful if they are not comprehensible.

Figure 4.7 shows plots examining insightfulness and usefulness for rules high and low inference values. By inspection, it remains apparent that rules hypothesized to be more comprehensible are more insightful and useful. To test whether there exists a correlation between comprehensibility, insightfulness, and usefulness, we performed Spearman-Rank Correlation test on the results depicted in Figure 4.8. By by inspecting the plot, it is already apparent that there is no obvious correlation between comprehensibility to insightfulness and comprehensibility to usefulness. Furthermore, the Spearman-Rank Correlation test yielded correlation values of 0.02 and 0.05 and pvalues of 0.8 and 0.06 for the respective comparisons. Statistical tests for the insightfulness and

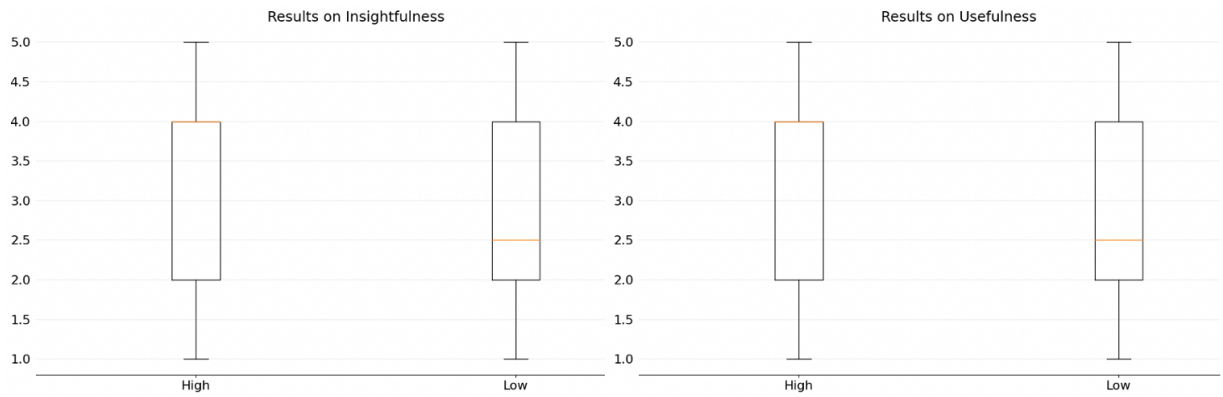


Figure 4.7: Perceived usefulness and insightfulness between rules with high and low KG-consistency (G1 & G3 vs G2).

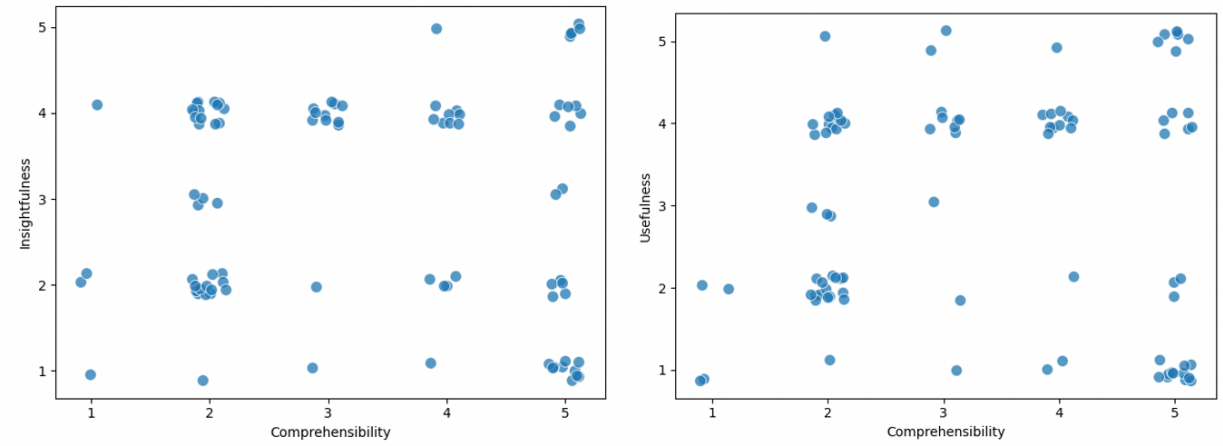


Figure 4.8: Perceived usefulness and insightfulness as a function of comprehensibility (G1 & G3 vs G2).

usefulness between groups with High and Low KG-Consistency yield pvalues of 0.16 and 0.16, respectively. Therefore, although data seems to indicate a positive correlation between attributes, our data does not quantitatively support the hypothesis that overall, rules perceived as comprehensible are more insightful and useful.

4.4.2.5 Confidence of Experts in Responses Provided

Figure 4.9 shows the frequency of responses indicating how confident experts felt after answering each question regarding rule comprehensibility, insightfulness, and usefulness, respectively.

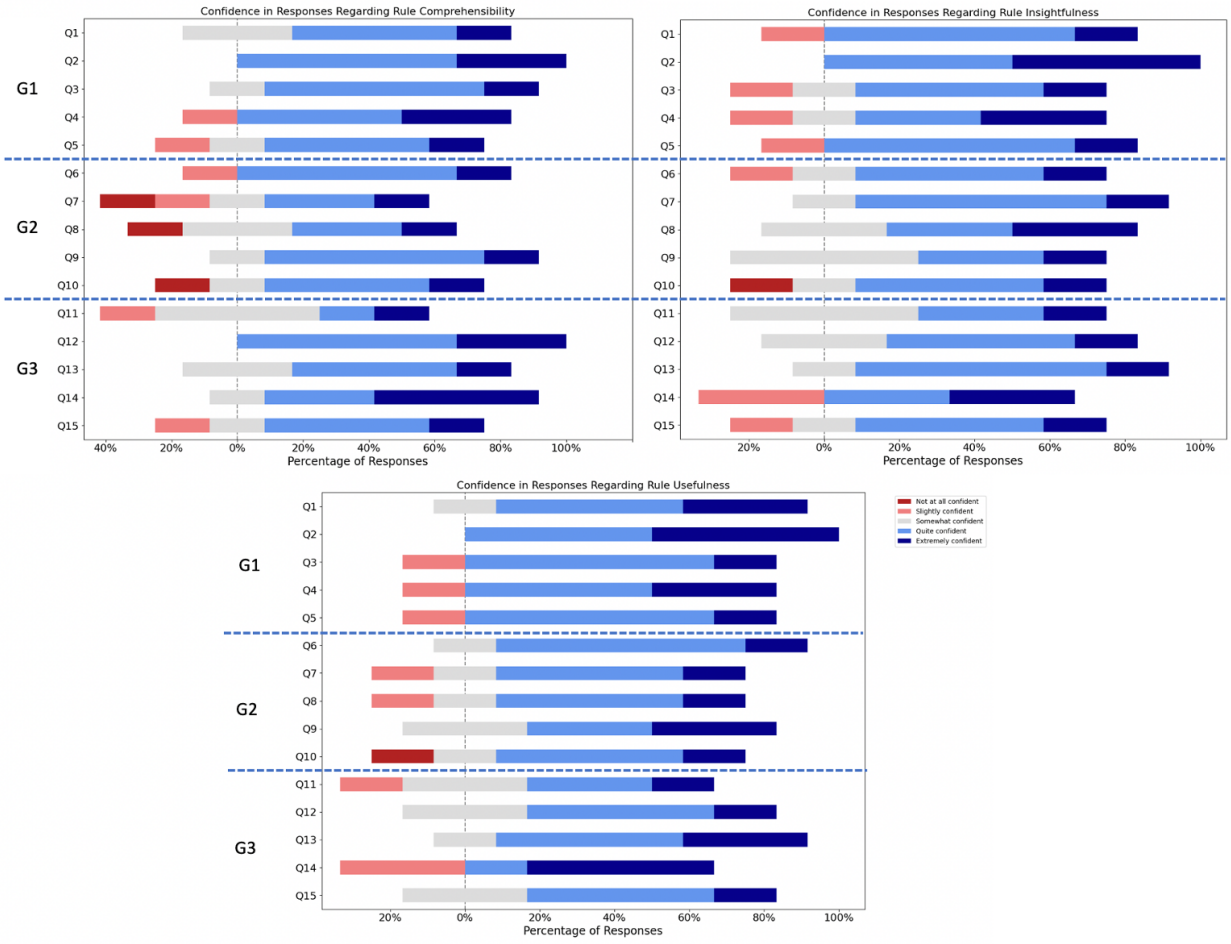


Figure 4.9: Confidence in responses obtained from experts.

Overall, the level of confidence in responses appears to be very similar across metrics. For comprehensibility, data shows that there was less confidence for responses in G2. However, statistical analysis ($U = 741$, $p = 0.0723$) indicates that there is no significant difference in confidence when evaluating comprehensibility when a rule has a high or low KG-consistency. The same statement can be done when assessing expert's confidence in responding about a rule's insightfulness ($U = 847.5$, $p = 0.316$) and usefulness ($U = 802.5$, $p = 0.185$).

4.4.3 Comparison of Results by EDL Roles

As seen in the previous sections, although our first hypothesis was supported by the data, there were some more surprising trends, particularly concerning H2 and H3. To better understand these results and assess whether there were significant differences in responses between the two groups of experts (analysts vs system engineers), we conducted additional analyses. The top row of plots in Figure 4.10 shows frequency of responses provided by the Trajectory Analysts and the bottom row shows frequency of responses provided by the Systems Engineers.

One of the distinguishing characteristics of responses was that trajectory analysts were generally in more agreement (i.e., more skewed) between them in their responses about a rule's comprehensibility, insightfulness, and usefulness. Responses provided by systems engineers, however, were more mixed. This was most noticeable for a rule's insightfulness and usefulness. When looking at responses provided by trajectory analysts, results show that rules consistent with knowledge (G1 and G3) are generally considered more comprehensible, insightful and useful when compared to rules with weak PSL inferences (G2).

We must highlight, that although overall there was no statistically significant effects of comprehensibility, insightfulness, and usefulness among all subjects. Correlation analysis indicated a potentially monotonic positive correlation between comprehensibility and usefulness of a rule. The correlation coefficient was 0.76 and a p-value of 0. However, statistical tests comparing comprehensibility and usefulness yielded a p-value of 0.1.

Overall, responses provided by system engineers differed significantly from those provided by trajectory analysts. The main difference is that results were much more mixed than those provided

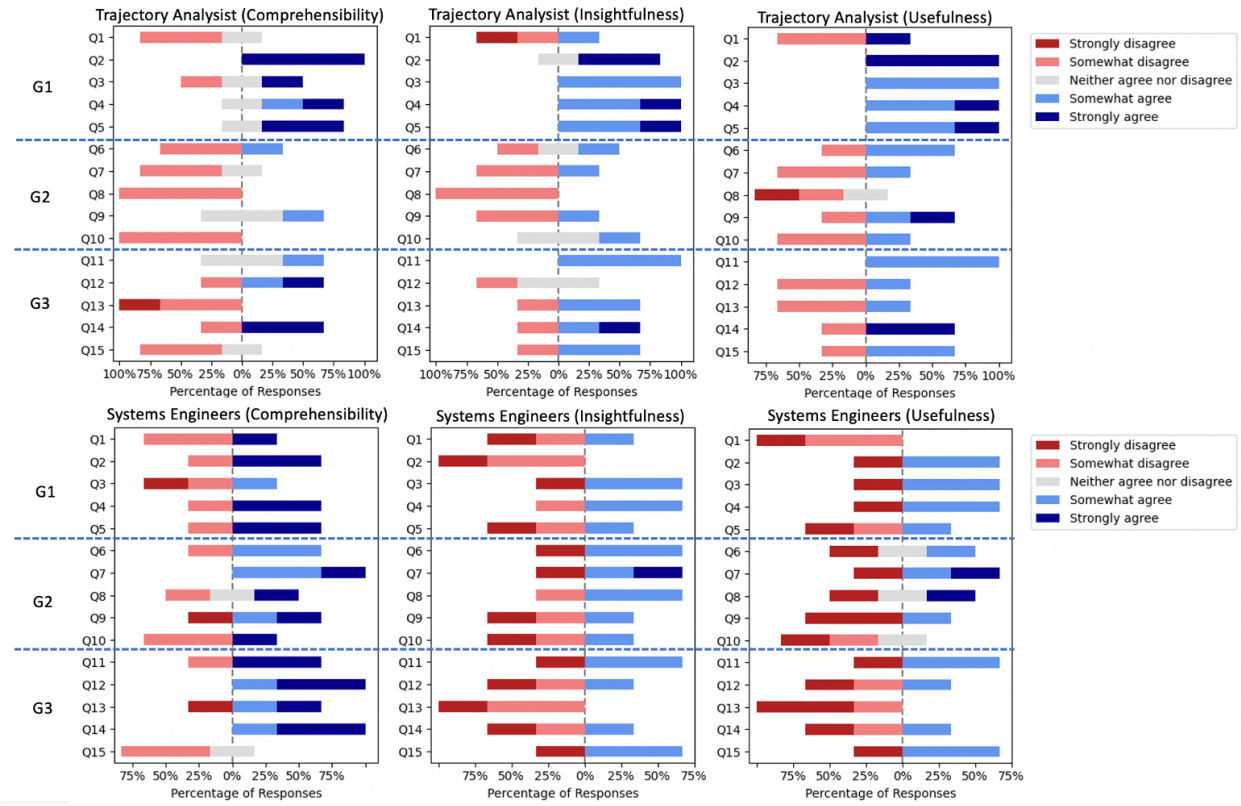


Figure 4.10: Responses regarding rule comprehensibility, insightfulness, and usefulness for both groups of subjects.

by trajectory analysts. Therefore, the seems to be more divided opinions among systems engineers. Furthermore, we see that system engineers considered rules less consistent with the knowledge graph (G2) more comprehensible than trajectory analysts. Results also show that although rules in G2 were considered quite comprehensible by systems engineers, they have strong opinions that these rules are not necessarily insightful or useful when compared to the other groups.

4.5 Conclusions

In this chapter, we proposed a method integrating knowledge and data-driven analysis to reason about inferred rules and help identify comprehensible, insightful, and useful rules. To the best of our knowledge, rule comprehensibility has not been well studied. Because of this, the state of the art remains to constrain the number of items in a rule to ensure comprehensibility. However, our results showed that even if rules have a high statistical strength, such as lift, it does not imply that it makes logical or physical sense. In the literature, it has been suggested that using knowledge (e.g., ontologies) can help ensure that comprehensible rules are found. However, existing frameworks focus on graph distance between items in a rule to assess interestingness without any underlying reasoning mechanism. Nevertheless, graph distance does consider that a large semantic distance could also indicate that there is no real relationship between items in the rule.

To help partially tackle these limitations, we used a knowledge graph to represent EDL knowledge about metrics, variables, and events in the EDL sequence. We also used a statistical relational learning framework named Probabilistic Soft Logic to reason about relationships under uncertainty. This allows assessing the likelihood of items in a rule being related using a rigorous mathematical framework. The study conducted revealed that rules whose items have strong inference values (i.e., high KG-consistency) between each other tended to be more comprehensible. Results also showed that in some instances, rules with weak inference values were considered comprehensible, which is somewhat surprising but could be due to an incomplete KG among other things. We believe that the lack of subjects and sample points contributed to this outcome. We also believe that contradicting perceptions of comprehensibility between groups of experts affected these results.

Results further revealed that rules with discrepancies between knowledge-driven and data-

driven measures are not necessarily more insightful or useful if they are not perceived as comprehensible. The tendency was that rules with high comprehensibility might be more insightful and useful.

Comments provided by experts indicated that although some rules were non-intuitive (especially in G2), this did not imply that they did not make sense in their entirety. This is especially useful given that one of the goals of our framework and the study, was to be able to discern between intuitive and non-intuitive rules. However, in the comments they mentioned that they found it interesting that the rule had high predictive power. Some experts argued that this type of rule could be useful. However, an audience like themselves will be skeptical when rules contain non-intuitive information and would thus, need convincing in order to investigate the information provided by the rule. This was mostly the case with trajectory analysts. Other experts, however, argued that the more trivial or complicated the relationship, the less useful it will tend to be. It was seen more from the systems engineers that this type of rule is not useful for them but it is potentially insightful.

We believe that lack of familiarity of the subjects with this type of data mining algorithms might have partially led to these results. Future work should investigate how familiarity with rule mining (or other data mining) algorithms affect the usefulness and insightfulness of discovered rules. We also believe that, although all experts specialized in EDL, there is a fundamental difference in how they interpret rules given their different backgrounds (analysts vs system engineers). Given that our KG was originally intended to help with the data analysis task, most relationships between variables and metrics exist assuming there is some underlying physical relationship between parameters in a rule (e.g., fuel consumption to high thermal loading). This is why trends in responses provided by trajectory analysts came less as a surprise than systems engineers. With this fact in mind, discrepancies between groups of experts are not necessarily surprising. Future work should examine ways to integrate systems engineering artifacts to the KG that can be useful for both teams. Knowledge graphs along with probabilistic reasoning capabilities can be very powerful tools to help experts perform end-to-end analysis of EDL systems.

5. CONCLUSIONS

5.1 Summary

This thesis presented new approaches and tools to improve current knowledge discovery process in the EDL domain. The goal was to provide interactive analysis capabilities whilst automating common tasks conducted during the analysis of EDL simulations. The end goal was to extract high-value information and help reduce the cognitive load imposed by the data analysis process. This thesis also proposed a new framework address the issue of comprehensibility of association rule mining data products.

Chapter 1 of this thesis began by introducing how EDL analysis is typically done and its limitations. One of the main limitations discussed is that there is a lack of tools useful for helping experts analyze EDL datasets. Consequently, most of the analysis is done manually by experts. Thus, experts must select only a handful of metrics to investigate every time a new simulation is run. While often times, experts may leverage their knowledge and expertise with past systems to identify issues and features of interest in a dataset, the next generation of EDL systems may present new challenges, especially for data analysis. At the moment, the most common analysis approaches used to identify driving features in a dataset is to conduct OAT sensitivity analysis or employ response surface methodology. However, most of these approaches rely on re-running simulations. Nevertheless, these methods are useful for obtaining a good understanding of the general behavior of the system. However, they do not help explain why particular outcomes occur. To help tackle some of the limitations of analyzing and explaining how complex systems such as EDL, this thesis introduced intelligent data understanding (IDU). The goal of using intelligent data understanding is to aid experts in the knowledge discovery process. IDUs can provide computational advances that can potentially help reduce analysis cycle time, identify areas of risk, and ensure mission success. This chapter states that, although IDUs are useful, they lack interactivity. Interactivity can be beneficial given that systems with these qualities, can use knowledge from ex-

perts (e.g., constraints, suggestions) to guide the analysis process. This can also be useful to scope the analysis to what users deem relevant. To help tackle this limitation, this chapter introduced cognitive assistants as a platform for IDU useful for analyzing EDL systems. This chapter also discussed how knowledge discovery tools help discover new information in large datasets. One of the largest drawbacks identified is that these algorithms, like rule mining algorithms, produce too many results, making it difficult for experts to navigate through pools of information in order to identify useful information. It is pointed out that there exist many interestingness measures aimed to prune data mining products. However, most approaches overlook how well discovered patterns and/or information are understood when studied in the context of a domain. The chapter proposed using domain knowledge (knowledge graphs) and statistical relational learning to reason about patterns and assess comprehensibility of a rule.

Chapter 2 of the thesis presented Daphne-EDL, a cognitive assistant used to help experts with the task of data analysis of EDL simulation outputs. This chapter first introduced Daphne-EO, the original version of the cognitive assistant and how it was adapted to handle natural language in the context of EDL. This chapter also discussed how queries and use cases for Daphne were selected- this was primarily done by communications with an EDL expert. This chapter then introduced all of the backends incorporated into Daphne-EDL: the query builder, scorecard generator, sensitivity analysis, data mining, and the dataset comparison tool. This chapter then discussed how each back ends obtains the relevant information from the data sources: historical database, expert knowledge base. With the interactive strategy introduced, Daphne's functions for EDL simulation explanations were discussed by means of a case study. Visualization of responses were provided for all instances. The main functions discussed were:

- Summarizing the statistics of large datasets (summarization)
- Identification of driving features (sensitivity analysis)
- Identification features and behaviors that appear to be common among failing cases or some other region of interest of the design space (association rule mining)

- : Comparing different simulation outputs (summarization sensitivity analysis)

This chapter also discussed how Daphne-EDL can help experts by reducing the burden of manually navigating through distributed data holdings for the analysis. Furthermore, the chapter also discussed how semi-automation of the analysis task can help explore more information in the data than what experts are able to do manually due to workload, time constraints, and the inherent complexity of the system.

Chapter 3 starts by discussing the challenge of tractability knowledge representation useful for analyzing EDL complex systems. Most expert have their own internal representation about how EDL systems behave and how many metrics and/or parameters interact. However, there is no framework that captures this information in a way it can be useful for other experts. Therefore, this chapter introduced knowledge graphs and Probabilistic Soft Logic (PSL). Knowledge graphs were proposed as a platform for representing knowledge about how w EDL variables, metrics, events, and components interact. The knowledge graph was created from multiple data sources that included tabular, numeric, and object-oriented data. This chapter then discussed how to use PSL to infer new relationships between variables and metrics in the KG. Using a case study, this chapter illustrated how the framework could be used to infer what parameters during entry and parachute descent affect the landing position of a Mars 2020 EDL architecture. Although the KG and PSL model were over-simplified, the goal was to demonstrate how to use these tools to integrate distributed information and knowledge into a unified framework that can be used to help evaluate complex systems. The proposed method is different to the state of the art given that the chapter proposed using knowledge graphs, as opposed to ontologies, given their scalability, interpretability, and relational nature. Given the relational nature of this tool, many frameworks from the statistical relational field can be useful for reasoning about information that exhibits uncertainty. These frameworks can be very useful for helping experts make judgements in the face of imprecise, incomplete, or uncertain information.

Chapter 4 discussed one of the challenges in studying results produced association rule mining algorithms: the number of rules generated is too large for manual examination and often contain

information that is redundant or incomprehensible. Although there exist many measures that help experts prune rules with the goal of identifying interesting patterns, how well the pattern is understandable by subject matter experts has been overlooked. Therefore, this chapter discussed how to use the statistical relational learning framework based on probabilistic soft logic to assess the degree of consistency of the rule with our knowledge of the system quantitatively. In the chapter it was hypothesized that rules that are considered more consistent with the knowledge graph will be perceived by the user as being more comprehensible (making more sense) than rules that are less consistent with the knowledge graph. We test this hypothesis and study links of the proposed consistency metric to the insightfulness and usefulness of rules as perceived by the user by means of a human study with N=6 subject matter experts. Results also showed a difference in how comprehensibility, insightfulness, and usefulness is perceived by experts with different backgrounds. Results support our primary hypothesis. We believe that the lack of subjects and sample points contributed to this outcome. We also believe that contradicting perceptions of comprehensibility between groups of experts affected these results. However, results indicated that comprehensibility is a necessary condition for usefulness and insightfulness given that we did not see any cases of rules no comprehensibility considered useful or insightful.

5.2 Main Contributions

This thesis explored tools for helping experts in the EDL domain and developed a new approach to improve rule mining algorithms. The main contributions and findings of this thesis are summarized below.

- A cognitive assistant that can aid EDL experts in the task of analyzing EDL simulations. Typical analysis of EDL systems has relied on manual analysis- making it time consuming and difficult to explore the entire dataset. Engineers use reports to help summarize simulation outcomes but data exploration and identification of driving features still relies on manual approaches. Because of this, NASA has been exploring IDU to help tackle the challenges of analysis of complex systems. Cognitive assistants, like Intelligent Data Understanding

Technologies harness machine learning and data mining. However, they provide one capability IDU's in isolation lack: user interactivity. Interactivity allows the user to intervene in the knowledge discovery process when necessary but can also perform autonomous analysis when desired. The congitive assistant used helps automate common tasks conducted during the analysis of EDL simulations. This includes, generating data products and summarizing results (e.g. scorecard, scatter plots histograms) and explaining simulation results. We propose two analysis forms for explaining simulation results: identifying driving features (sensitivity analysis) and discovering common features and behaviors that appear to be common among a region of interest in a dataset (association rule mining). This tool provides visualizations and makes use of information from multiple data sources to conduct analysis.

- Proposed using knowledge graphs to integrate data from distributed data holdings into one framework. We also proposed integrating systems engineering artifacts to such as MBSE to capture domain knowledge in a single platform. We created an EDL knowledge graph to represent and infer relationships between EDL variables and critical performance metrics. This knowledge graph has to be provided by the user, but we hope that given current trends in model-based systems engineering, in the future information can automatically extracted from MBSE artifacts such as lessons learned databases and SysML models generated during the mission development process.

- Implemented a new framework for to help improve comprehensibility in rule mining algorithms. To the best of the author's knowledge, there has not been significant attempts to develop methods and measures to assess the comprehensibility of mined association rules. Thus, most work largely considers the number of items in a rule as the main (and in fact only) factor affecting comprehensibility of the rule. However, in some instances, there is no logical or physical relationship between entities in a rule, making it difficult for the subject matter expert to make sense of it, regardless of the number of items in the rule. This thesis proposed using knowledge graphs to capture the user knowledge about EDL the specific

problem at hand. We then use Probabilistic Soft Logic to assess the degree of consistency of the rule with our knowledge of the system quantitatively. We hypothesized that rules that are considered more consistent with the knowledge graph will be perceived by the user as being more comprehensible (making more sense) than rules that are less consistent with the knowledge graph. We tested this hypothesis and study links of the proposed consistency metric to the insightfulness and usefulness of rules as perceived by the user by means of a human study with N=6 subject matter experts. Reresults indicated that comprehensibility is a necessary condition for usefulness and insightfulness given that we did not see any cases of rules no comprehensibility considered useful or insightful.

5.3 Discussion

The tools, methods, and frameworks presented in this thesis seek to help experts extract high-value information from EDL datasets that can be used for future decision-making. Given how time consuming the task of data analysis is in this domain, we hope that the work presented here can help reduce the cognitive load imposed on experts by the aforementioned task.

Using cognitive assistants as a decision support tool can maximize the information gain during data analysis. Interaction is useful for specifying-domain specific information that can be qualitative or quantitative. Furthermore, cognitive assistants, like Daphne-EDL bring relevant information from multiple sources very quickly at the relevant time. Chapter 2 can be used to help experts with automating common aspects of data analysis in addition to provide analysis capabilities. Chapter 2 can be expanded by incorporating test databases for generating evaluating data and generating reports useful of operational scenarios, for example. Automation can help reduce cognitive load on experts during high-stake scenarios. Chapter 2 can also be expanded by incorporating a framework that allows experts to provide feedback information extracted by simulation results. This can help reduce the redundant information provided by Daphne.

The work described in Chapter 3 presents a framework for encoding domain knowledge. Furthermore, when combined with statistical relational learning, experts can discover potentially insightful and useful relationships useful for decision making. Chapter 3 can be expanded by incor-

porating systems engineering artifacts, like MBSE, into knowledge graphs. As stated in Chapter 1, NASA has been exploring the idea of using MBSE throughout a mission lifecycle to minimize systems engineering practices that rely on document tracking and standalone analysis- which reduces the issue of distributed data holdings. Their goal is to use MBSE to build an integrated model of concepts of operations, which includes: functional decomposition, operational scenarios, structural decomposition, requirements and traces to other model elements, and authority delegation. Integrating MBSE artifacts, such as lessons learned databases and SysML models into knowledge graphs can be useful for extracting high-value content during knowledge discovery. With the appropriate tools available, like PSL and other statistical relational frameworks, logical reasoning can me employed to identify “logical fallacies" and other other information across mission elements that is critical and hard to detect with the current approaches (e.g., manual and distributed data).

The work in Chapter 4 addressed the issue of comprehensibility in association rule mining. The framework proposed improved comprehensibility of rules by leveraging domain knowledge and statistical relational learning for reasoning about dependencies in the graph. Results showed that rules with high KG-consistency are comprehensible and are thus useful, and insightful. For the sake of demonstrating the hypothesis, the work done did not perform a comparison on consistency metrics and whether one of them was better at identifying comprehensible rules. The work in Chapter 4 can also be extended for developing KG-consistency metrics for more complex rules. The study only considered conjunctions of items in the antecedent. Future work can extend the proposed metrics by developing consistency metrics and formulations for rules that contain arbitrary combinations of conjunctions and disjunctions. The work in this chapter can be further extended to incorporate simulation data into the KG and the PSL model. In our model, parameters, events, and metrics were evaluated did not represent their relationships across different simulation cases and was intended to capture general knowledge and/or beliefs. Therefore, with the current formulation, PSL only reasons about entities (e.g., navigation errors) without considering its numerical attribute. In the future, information about variables and simulation outputs can be useful to make more specific inference tasks. For example, PSL can be used to do reasoning about critical

EDL metrics for simulation cases in the KG where navigation errors are high, for example. These inferences might provide much more insightful information.

Although the human-subject experiment indicated that association rules consistent with the knowledge graph are perceived as comprehensible by EDL experts, more work should be done to study how experts asses this quality given their background. Although teams within EDL collaborate and work together, the study showed a significant difference in their perception about the rules presented. This work is also limited given that it does not evaluate how comprehensibility is perceived in the presence of conjunctions and disjunctions present.

5.4 Limitations and Future Work

This thesis was a first step towards incorporating intelligent tools to the EDL domain. There are many opportunities for future work to find ways to incorporate expert's knowledge to a cognitive assistant to help improve the analysis. We also believe that integrating system engineering artifacts can help evaluate EDL architectures throughout a mission's lifecycle.

Effort should be made to explore how mixed-initiative approaches can help enhance the teacher-apprentice vision of the system. It would be helpful if an expert can provide feedback on the results generated. For example, if analysis identifies some patterns that do not make sense (i.e., there is no manner in which items in the pattern can affect each other), experts can indicate so to Daphne. This way, Daphne can learn this information and make use of it for future analysis.

In this work, Daphne-EDL was tailored to helping experts with exploring simulation data products in an interactive manner. Therefore, perform human factors validation studies. However, if Daphne-EDL were to be adapted for helping experts during operational scenarios, it is critical that verification and validation (V&V) is done. V&V will be critical for identifying any usability or design issues with the tool that could increase the likelihood of human errors as opposed to mitigating them.

A significant effort should be made to develop knowledge graphs with systems engineering artifacts that can be shared across engineering groups. Furthermore, a unified modeling framework can also enable large-scale re-usability of entities across projects. Employing these capabilities

throughout development and later lifecycle phases may help reduce lifecycle time and cost while also reducing the likelihood of late discovery of design problems.

Future work should continue to explore models that help improve the comprehensibility of mined association rules since comprehensible rules can be more easily studied by engineers and designers. Surveys conducted demonstrated skepticism by designers regarding trust about simulation outputs when they are non-intuitive. Future work should also consider reasoning about combinations of conjunctions and disjunctions in a rule. For the sake of demonstration, in thesis we only explored conjunctions of two items.

Future work should investigate how familiarity with rule mining (or other data mining) algorithms affect the usefulness and insightfulness of discovered rules. We also believe that, although all experts specialized in EDL, there is a fundamental difference in how they interpret rules given their different backgrounds (analysts vs system engineers). Given that our KG was originally intended to help with the data analysis task, most relationships between variables and metrics exist assuming there is some underlying physical relationship between parameters in a rule (e.g., fuel consumption to high thermal loading). This is why trends in responses provided by trajectory analysts came less as a surprise than systems engineers. With this fact in mind, discrepancies between groups of experts are not necessarily surprising. However, given that both groups of experts actively collaborate, it was interesting how their perceptions varied.

REFERENCES

[1] S. Santini De Leon, D. Selva, and D. W. Way, "Interactive explanation of entry, descent, and landing simulations," in *AIAA Scitech 2020 Forum*, p. 2094, 2020.

[2] S. S. De León, D. Selva, and D. W. Way, "A cognitive assistant for entry, descent, and landing architecture analysis," in *2019 IEEE Aerospace Conference*, pp. 1–12, IEEE, 2019.

[3] C. A. W. D. Santini, Samalis and P. Brugarolas, "Assessment of the robustness of the mars 2020 terminal descent sensor in the event of beam failures," *Unpublished Manuscript*, 2019.

[4] A. A. Wolf, B. Acikmese, Y. Cheng, J. Casoliva, J. M. Carson, and M. C. Ivanov, "Toward improved landing precision on mars," in *2011 Aerospace Conference*, pp. 1–8, IEEE, 2011.

[5] R. Braun, M., Manning, "Mars Exploration Entry, Descent, and Landing Challenges," *Journal of Spacecraft and Rockets*, vol. 44, no. 2, pp. 310–323, 2007.

[6] A. Halder and R. Bhattacharya, "Dispersion analysis in hypersonic flight during planetary entry using stochastic liouville equation," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 2, pp. 459–474, 2011.

[7] M. Adler, M. Wright, C. Campbell, I. Clark, W. Engelund, and T. Rivellini, "Entry, descent, and landing roadmap," *NASA TA09, April*, 2012.

[8] D. E. Cooke, "An overview of nasa's intelligent systems program," in *2001 IEEE Aerospace Conference Proceedings (Cat. No. 01TH8542)*, vol. 7, pp. 7–3664, IEEE, 2001.

[9] T. Flatley, "Advanced hybrid on-board science data processor-spacecube 2.0," in *AGU Fall Meeting Abstracts*, vol. 2011, pp. IN43B–1438, 2011.

[10] J. Le Moigne, P. Dabney, O. de Weck, V. Foreman, P. Grogan, M. Holland, S. Hughes, and S. Nag, "Tradespace analysis tool for designing constellations (tat-c)," in *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 1181–1184, IEEE, 2017.

[11] N. Hitomi, H. Bang, and D. Selva, "Adaptive knowledge-driven optimization for architecting a distributed satellite system," *Journal of Aerospace Information Systems*, vol. 15, no. 8, pp. 485–500, 2018.

[12] A. V. i Martin and D. Selva, "Daphne: A virtual assistant for designing earth observation distributed spacecraft missions," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 30–48, 2019.

[13] P. Dutta, O. Balcells-Quintana, A. Viros Martin, R. Whittle, P. K. Josan, N. Beebe, B. J. Dunbar, R. KW Wong, A. Diaz-Artiles, and D. Selva, "Virtual assistant for anomaly treatment in long duration exploration missions," in *AIAA Scitech 2020 Forum*, p. 2255, 2020.

[14] B. Matthews, S. Das, K. Bhaduri, K. Das, R. Martin, and N. Oza, "Discovering anomalous aviation safety events using scalable data mining algorithms," *Journal of Aerospace Information Systems*, vol. 10, no. 10, pp. 467–475, 2013.

[15] S. Alimo, B. Kahovec, D. Divsalar, D. Sam, and A. Chowdhury, "Data accountability and uncertainty analysis for the mars science laboratory," 2020.

[16] R. Fujimaki, T. Yairi, and K. Machida, "An anomaly detection method for spacecraft using relevance vector learning," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 785–790, Springer, 2005.

[17] 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," in *2nd IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT'06)*, pp. 8–pp, IEEE, 2006.

[18] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 387–395, 2018.

[19] S. Chien, B. Cichy, A. Davies, D. Tran, G. Rabideau, R. Castano, R. Sherwood, D. Mandl, S. Frye, S. Shulman, *et al.*, "An autonomous earth-observing sensorweb," *IEEE Intelligent Systems*, vol. 20, no. 3, pp. 16–24, 2005.

[20] C. Breazeal, C. D. Kidd, A. L. Thomaz, G. Hoffman, and M. Berlin, "Effects of nonverbal communication on efficiency and robustness in human-robot teamwork," in *2005 IEEE/RSJ international conference on intelligent robots and systems*, pp. 708–713, IEEE, 2005.

[21] A. Newell, H. A. Simon, *et al.*, *Human problem solving*, vol. 104. Prentice-hall Englewood Cliffs, NJ, 1972.

[22] E. García, C. Romero, S. Ventura, and T. Calders, "Drawbacks and solutions of applying association rule mining in learning management systems," in *Proceedings of the international workshop on applying data mining in e-learning (ADML 2007), Crete, Greece*, pp. 13–22, sn, 2007.

[23] L. Geng and H. J. Hamilton, "Interestingness measures for data mining: A survey," *ACM Computing Surveys (CSUR)*, vol. 38, no. 3, pp. 9–es, 2006.

[24] K. McGarry, "A survey of interestingness measures for knowledge discovery," *Knowledge Eng. Review*, vol. 20, no. 1, pp. 39–61, 2005.

[25] C. Marinica, F. Guillet, and H. Briand, "Post-processing of discovered association rules using ontologies," in *2008 IEEE International Conference on Data Mining Workshops*, pp. 126–133, IEEE, 2008.

[26] S. A. Striepe, D. W. Way, A. M. Dwyer, and J. Balaram, "Mars Science Laboratory Simulations for Entry, Descent, and Landing," *Journal of Spacecraft and Rockets*, 2006.

[27] R. W. Maddock, A. M. Dwyer-Cianciolo, D. Litton, and C. H. Zumwalt, "Insight entry, descent, and landing pre-flight performance predictions," in *AIAA Scitech 2020 Forum*, p. 1269, 2020.

[28] S. J. Kapurch, *NASA systems engineering handbook*. Diane Publishing, 2010.

[29] N. Hirschi and D. Frey, "Cognition and complexity: an experiment on the effect of coupling in parameter design," *Research in Engineering Design*, vol. 13, no. 3, pp. 123–131, 2002.

[30] D. Keim, "Mastering the information age: Solving problems with visual analytics," 2010.

[31] A. Stehura, M. Rozek, and D. Isla, "Managing complexity: Solutions from the mars science laboratory entry, descent, and landing flight system verification and validation campaign," *Journal of Spacecraft and Rockets*, vol. 51, no. 4, pp. 1270–1287, 2014.

[32] H. J. Escalante, S. Escalera, I. Guyon, X. Baró, Y. Güçlütürk, U. Güçlü, and M. van Gerven, *Explainable and interpretable models in computer vision and machine learning*. Springer, 2018.

[33] B. Iooss and P. Lemaître, "A review on global sensitivity analysis methods," in *Uncertainty management in simulation-optimization of complex systems*, pp. 101–122, Springer, 2015.

[34] F. Pianosi and T. Wagener, "A simple and efficient method for global sensitivity analysis based on cumulative distribution functions," *Environmental Modelling & Software*, vol. 67, pp. 1–11, 2015.

[35] A. Saltelli, M. Ratto, S. Tarantola, F. Campolongo, E. Commission, *et al.*, "Sensitivity analysis practices: Strategies for model-based inference," *Reliability Engineering & System Safety*, vol. 91, no. 10-11, pp. 1109–1125, 2006.

[36] S. Dutta and D. W. Way, "Comparison of the effects of velocity and range triggers on trajectory dispersions for the mars 2020 mission," in *AIAA Atmospheric Flight Mechanics Conference*, p. 0245, 2017.

[37] W. Castaings, E. Borgonovo, M. D. Morris, and S. Tarantola, "Sampling strategies in density-based sensitivity analysis," *Environmental Modelling & Software*, vol. 38, pp. 13–26, 2012.

[38] A. Wang and D. P. Solomatine, "Practical experience of sensitivity analysis: Comparing six methods, on three hydrological models, with three performance criteria," *Water*, vol. 11, no. 5, p. 1062, 2019.

[39] A. Saltelli, P. Annoni, I. Azzini, F. Campolongo, M. Ratto, and S. Tarantola, "Variance based sensitivity analysis of model output. design and estimator for the total sensitivity index," *Computer physics communications*, vol. 181, no. 2, pp. 259–270, 2010.

[40] B. Sudret, "Global sensitivity analysis using polynomial chaos expansions," *Reliability engineering & system safety*, vol. 93, no. 7, pp. 964–979, 2008.

[41] G. Blatman and B. Sudret, "Efficient computation of global sensitivity indices using sparse polynomial chaos expansions," *Reliability Engineering & System Safety*, vol. 95, no. 11, pp. 1216–1229, 2010.

[42] D. Shahsavani and A. Grimvall, "Variance-based sensitivity analysis of model outputs using surrogate models," *Environmental Modelling & Software*, vol. 26, no. 6, pp. 723–730, 2011.

[43] R. DeLoach, "Analysis of variance in the modern design of experiments," in *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, p. 1111, 2010.

[44] S. Jeong, K. Chiba, and S. Obayashi, "Data mining for aerodynamic design space," *Journal of aerospace computing, information, and communication*, vol. 2, no. 11, pp. 452–469, 2005.

[45] C. B. Storlie, L. P. Swiler, J. C. Helton, and C. J. Sallaberry, "Implementation and evaluation of nonparametric regression procedures for sensitivity analysis of computationally demanding models," *Reliability Engineering & System Safety*, vol. 94, no. 11, pp. 1735–1763, 2009.

[46] C. Zhang, L. Song, C. Fei, C. Lu, and Y. Xie, "Advanced multiple response surface method of sensitivity analysis for turbine blisk reliability with multi-physics coupling," *Chinese Journal of Aeronautics*, vol. 29, no. 4, pp. 962–971, 2016.

[47] R. DeLoach, "The role of hierarchy in response surface modeling of wind tunnel data," in *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, p. 931, 2010.

[48] T. Motoda and Y. Miyazawa, "Identification of influential uncertainties in monte carlo analysis," *Journal of spacecraft and rockets*, vol. 39, no. 4, pp. 615–623, 2002.

[49] C. I. Restrepo and J. E. Hurtado, "Tool for rapid analysis of monte carlo simulations," *Journal of Spacecraft and Rockets*, vol. 51, no. 5, pp. 1564–1575, 2014.

[50] C. Restrepo and J. Hurtado, "Pattern recognition for a flight dynamics monte carlo simulation," in *AIAA Guidance, Navigation, and Control Conference*, p. 6590, 2011.

[51] K. Gundy-Burlet, J. Schumann, T. Menzies, and T. Barrett, "Parametric analysis of antares re-entry guidance algorithms using advanced test generation and data analysis," in *Proc. iSAIRAS*, vol. 2008, 2008.

[52] T. Menzies, E. Chiang, M. Feather, Y. Hu, and J. D. Kiper, "Condensing uncertainty via incremental treatment learning," in *Software Engineering with Computational Intelligence*, pp. 319–361, Springer, 2003.

[53] M. Chi, A. Plaza, J. A. Benediktsson, Z. Sun, J. Shen, and Y. Zhu, "Big data for remote sensing: Challenges and opportunities," *Proceedings of the IEEE*, vol. 104, no. 11, pp. 2207–2219, 2016.

[54] J. D. Frank and G. B. Aaseng, "Transitioning autonomous systems technology research to a flight software environment," in *AIAA SPACE 2016*, p. 5530, 2016.

[55] J. Frank, L. Spirkovska, R. McCann, L. Wang, K. Pohlkamp, and L. Morin, "Autonomous mission operations," in *2013 IEEE Aerospace Conference*, pp. 1–20, IEEE, 2013.

[56] D. Petrick, A. Geist, D. Albaijes, M. Davis, P. Sparacino, G. Crum, R. Ripley, J. Boblitt, and T. Flatley, "Spacecube v2. 0 space flight hybrid reconfigurable data processing system," in *2014 IEEE Aerospace Conference*, pp. 1–20, IEEE, 2014.

[57] M. R. Berthold, C. Borgelt, F. Höppner, and F. Klawonn, *Guide to intelligent data analysis: how to intelligently make sense of real data*. Springer Science & Business Media, 2010.

[58] B. Hjørland, "Information: Objective or subjective/situational?," *Journal of the American society for information science and technology*, vol. 58, no. 10, pp. 1448–1456, 2007.

[59] D. J. Hand, "Intelligent data analysis: Issues and opportunities," in *International Symposium on Intelligent Data Analysis*, pp. 1–14, Springer, 1997.

[60] M. Shafto, M. Conroy, R. Doyle, E. Glaessgen, C. Kemp, J. LeMoigne, and L. Wang, "Modeling, simulation, information technology & processing roadmap," *National Aeronautics and Space Administration*, 2012.

[61] D. E. Cooke, "An overview of nasa's intelligent systems program," in *2001 IEEE Aerospace Conference Proceedings (Cat. No. 01TH8542)*, vol. 7, pp. 7–3664, IEEE, 2001.

[62] N. R. Council *et al.*, *NASA space technology roadmaps and priorities: restoring NASA's technological edge and paving the way for a new era in space*. National Academies Press, 2012.

[63] S. Chien, R. Sherwood, D. Tran, B. Cichy, G. Rabideau, R. Castano, A. Davies, R. Lee, D. Mandl, S. Frye, *et al.*, "The eo-1 autonomous science agent," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pp. 420–427, Citeseer, 2004.

[64] E. M. Middleton, S. G. Ungar, D. J. Mandl, L. Ong, S. W. Frye, P. E. Campbell, D. R. Landis, J. P. Young, and N. H. Pollack, "The earth observing one (eo-1) satellite mission: Over a decade in space," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 6, no. 2, pp. 243–256, 2013.

[65] M. J. Way and A. Srivastava, "Novel methods for predicting photometric redshifts from broadband photometry using virtual sensors," *The Astrophysical Journal*, vol. 647, no. 1, p. 102, 2006.

[66] A. N. Srivastava, N. C. Oza, and J. Stroeve, "Virtual sensors: Using data mining techniques to efficiently estimate remote sensing spectra," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, no. 3, pp. 590–600, 2005.

[67] P. Maes, "Agents that reduce work and information overload," in *Readings in human–computer interaction*, pp. 811–821, Elsevier, 1995.

[68] C. Steinruecken, E. Smith, D. Janz, J. Lloyd, and Z. Ghahramani, "The automatic statistician," in *Automated Machine Learning*, pp. 161–173, Springer, Cham, 2019.

[69] Z. Ghahramani, "Probabilistic machine learning and artificial intelligence," *Nature*, vol. 521, no. 7553, pp. 452–459, 2015.

[70] D. B. Lenat, "Eurisko: a program that learns new heuristics and domain concepts: the nature of heuristics iii: program design and results," *Artificial intelligence*, vol. 21, no. 1-2, pp. 61–98, 1983.

[71] J. Z. Pan, G. Vetere, J. M. Gomez-Perez, and H. Wu, *Exploiting linked data and knowledge graphs in large organisations*. Springer, 2017.

[72] T. Bayer, S. Chung, B. Cole, B. Cooke, F. Dekens, C. Delp, I. Gontijo, K. Lewis, M. Moshir, R. Rasmussen, *et al.*, "11.5. 1 early formulation model-centric engineering on nasa's europa mission concept study," in *INCOSE International Symposium*, vol. 22, pp. 1695–1710, Wiley Online Library, 2012.

[73] R. Nayak, "Intelligent data analysis: Issues and challenges," in *6th World Multi Conferences on Systemics, Cybernetics and Informatics*, 2002.

[74] H. Bang, A. Virós Martin, A. Prat, and D. Selva, "Daphne: An intelligent assistant for architecting earth observing satellite systems," in *2018 AIAA Information Systems-AIAA Infotech@ Aerospace*, p. 1366, 2018.

[75] B. Döring *et al.*, "A cognitive assistant for supporting air target identification on navy ships," *IFAC Proceedings Volumes*, vol. 45, no. 2, pp. 469–474, 2012.

[76] R. Onken and A. Walsdorf, "Assistant systems for aircraft guidance: cognitive man-machine cooperation," *Aerospace Science and Technology*, vol. 5, no. 8, pp. 511–520, 2001.

[77] S. A. Wilkins, "Examination of pilot benefits from cognitive assistance for single-pilot general aviation operations," in *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*, pp. 1–9, IEEE, 2017.

[78] G. Tokadlı and M. C. Dorneich, "Development of a functionality matrix for a cognitive assistant on long distance space missions," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 61, pp. 247–251, SAGE Publications Sage CA: Los Angeles, CA, 2017.

[79] A. Viros Martin and D. Selva, "Explanation approaches for the daphne virtual assistant," in *AIAA Scitech 2020 Forum*, p. 2254, 2020.

[80] W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus, "Knowledge discovery in databases: An overview," *AI magazine*, vol. 13, no. 3, pp. 57–57, 1992.

[81] N. Dedić and C. Stanier, "Towards differentiating business intelligence, big data, data analytics and knowledge discovery," in *International Conference on Enterprise Resource Planning Systems*, pp. 114–122, Springer, 2016.

[82] L. Rokach and O. Z. Maimon, *Data mining with decision trees: theory and applications*, vol. 69. World scientific, 2008.

[83] S. D. Bay and M. J. Pazzani, "Detecting group differences: Mining contrast sets," *Data mining and knowledge discovery*, vol. 5, no. 3, pp. 213–246, 2001.

[84] C. Zhang and S. Zhang, *Association rule mining: models and algorithms*, vol. 2307. Springer, 2003.

[85] Z. Zheng, R. Kohavi, and L. Mason, "Real world performance of association rule algorithms," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 401–406, 2001.

[86] M. Z. Ashrafi, D. Taniar, and K. Smith, "A new approach of eliminating redundant association rules," in *International Conference on Database and Expert Systems Applications*, pp. 465–474, Springer, 2004.

[87] R. Agrawal, R. Srikant, *et al.*, "Fast algorithms for mining association rules," in *Proc. 20th int. conf. very large data bases, VLDB*, vol. 1215, pp. 487–499, 1994.

[88] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pp. 207–216, 1993.

[89] M. Jalali-Heravi and O. R. Zaïane, "A study on interestingness measures for associative classifiers," in *Proceedings of the 2010 ACM Symposium on Applied Computing*, pp. 1039–1046, 2010.

[90] H. Bang and D. Selva, "Discovering generalized design knowledge using a multi-objective evolutionary algorithm with generalization operators," *Unpublished Manuscript*, 2019.

[91] S. Guillaume, D. Grissa, and E. M. Nguifo, "Categorization of interestingness measures for knowledge extraction," *arXiv preprint arXiv:1206.6741*, 2012.

[92] A. A. Freitas, "On objective measures of rule surprisingness," in *European Symposium on Principles of Data Mining and Knowledge Discovery*, pp. 1–9, Springer, 1998.

[93] B. Liu, W. Hsu, L.-F. Mun, and H.-Y. Lee, "Finding interesting patterns using user expectations," *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 6, pp. 817–832, 1999.

[94] B. Liu, W. Hsu, S. Chen, and Y. Ma, "Analyzing the subjective interestingness of association rules," *IEEE Intelligent Systems and their Applications*, vol. 15, no. 5, pp. 47–55, 2000.

[95] B. Padmanabhan and A. Tuzhilin, "Unexpectedness as a measure of interestingness in knowledge discovery," *Decision Support Systems*, vol. 27, no. 3, pp. 303–318, 1999.

[96] A. Silberschatz and A. Tuzhilin, "On subjective measures of interestingness in knowledge discovery.," in *KDD*, vol. 95, pp. 275–281, 1995.

[97] A. S. Al-Hegami, "Subjective measures and their role in data mining process," in *Proceedings of the 6th International Conference on Cognitive Systems, New Delhi, India*, Citeseer, 2004.

[98] C. Marinica and F. Guillet, "Knowledge-based interactive postmining of association rules using ontologies," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 6, pp. 784–797, 2010.

[99] B. Liu, W. Hsu, and Y. Ma, "Mining association rules with multiple minimum supports," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 337–341, 1999.

[100] G. Mansingh, K.-M. Osei-Bryson, and H. Reichgelt, "Using ontologies to facilitate post-processing of association rules by domain experts," *Information sciences*, vol. 181, no. 3, pp. 419–434, 2011.

[101] R. Srikant, Q. Vu, and R. Agrawal, "Mining association rules with item constraints,"

[102] R. G. Miani, C. A. Yaguinuma, M. T. Santos, and M. Biajiz, "Narfo algorithm: Mining non-redundant and generalized association rules based on fuzzy ontologies," in *International Conference on Enterprise Information Systems*, pp. 415–426, Springer, 2009.

[103] M. A. Domingues and S. O. Rezende, "Using taxonomies to facilitate the analysis of the association rules," *arXiv preprint arXiv:1112.1734*, 2011.

[104] X. Chen, M. Chen, W. Shi, Y. Sun, and C. Zaniolo, "Embedding uncertain knowledge graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 3363–3370, 2019.

[105] B. Minaei-Bidgoli, R. Barmaki, and M. Nasiri, "Mining numerical association rules via multi-objective genetic algorithms," *Information Sciences*, vol. 233, pp. 15–24, 2013.

[106] W. G. Chase, "Simon. ha (1973)," *Perception in chess. Cognitive psychology*, vol. 4, no. 1, pp. 55–81, 81.

[107] F. Gobet, P. C. Lane, S. Croker, P. C. Cheng, G. Jones, I. Oliver, and J. M. Pine, "Chunking mechanisms in human learning," *Trends in cognitive sciences*, vol. 5, no. 6, pp. 236–243, 2001.

[108] L. Ehrlinger and W. Wöß, "Towards a definition of knowledge graphs.," *SEMANTiCS (Posters, Demos, SuCCESS)*, vol. 48, pp. 1–4, 2016.

[109] M. Krötzsch, M. Marx, A. Ozaki, and V. Thost, "Attributed description logics: Ontologies for knowledge graphs," in *International Semantic Web Conference*, pp. 418–435, Springer, 2017.

[110] C. Strauch, U.-L. S. Sites, and W. Kriha, "Nosql databases," *Lecture Notes, Stuttgart Media University*, vol. 20, p. 24, 2011.

[111] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, and M. Guo, "Ripplenet: Propagating user preferences on the knowledge graph for recommender systems," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 417–426, 2018.

[112] E. Palumbo, G. Rizzo, and R. Troncy, "Entity2rec: Learning user-item relatedness from knowledge graphs for top-n item recommendation," in *Proceedings of the eleventh ACM conference on recommender systems*, pp. 32–36, 2017.

[113] D. Meza, "How nasa finds critical data through a knowledge graph," *Online], https://neo4j. com/blog/nasa-critical-data-knowledge-graph/, Accessed February*, vol. 19, p. 2019, 2017.

[114] A. Rossi, D. Firmani, A. Matinata, P. Merialdo, and D. Barbosa, "Knowledge graph embedding for link prediction: A comparative analysis," *arXiv preprint arXiv:2002.00819*, 2020.

[115] R. B. Tchoua, A. Ajith, Z. Hong, L. T. Ward, K. Chard, A. Belikov, D. J. Audus, S. Patel, J. J. de Pablo, and I. T. Foster, "Creating training data for scientific named entity recognition with minimal human effort," in *International Conference on Computational Science*, pp. 398–411, Springer, 2019.

[116] B. K. Muirhead, A. K. Nicholas, J. Umland, O. Sutherland, and S. Vijendran, "Mars sample return campaign concept status," *Acta Astronautica*, vol. 176, pp. 131–138, 2020.

[117] S. J. Herzig, D. Velez, B. Nairouz, B. Weatherspoon, R. Tikidjian, T. M. Randolph, and B. Muirhead, "A model-based approach to developing the concept of operations for potential mars sample return," in *2018 AIAA SPACE and Astronautics Forum and Exposition*, p. 5389, 2018.

[118] T. J. Bayer, L. A. Cooney, C. L. Delp, C. A. Dutenhoffer, R. D. Gostelow, M. D. Ingham, J. S. Jenkins, and B. S. Smith, "An operations concept for integrated model-centric engineering at jpl," in *2010 IEEE Aerospace Conference*, pp. 1–14, IEEE, 2010.

[119] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, and Q. He, "A survey on knowledge graph-based recommender systems," *arXiv preprint arXiv:2003.00911*, 2020.

[120] J. Li, B. Ge, K. Yang, Y. Chen, and Y. Tan, "Meta-path based heterogeneous combat network link prediction," *Physica A: Statistical Mechanics and its Applications*, vol. 482, pp. 507–523, 2017.

[121] V. Martínez, F. Berzal, and J.-C. Cubero, "A survey of link prediction in complex networks," *ACM computing surveys (CSUR)*, vol. 49, no. 4, pp. 1–33, 2016.

[122] S. Li, J. Huang, Z. Zhang, J. Liu, T. Huang, and H. Chen, "Similarity-based future common neighbors model for link prediction in complex networks," *Scientific reports*, vol. 8, no. 1, pp. 1–11, 2018.

[123] F. Gao, K. Musial, C. Cooper, and S. Tsoka, "Link prediction methods and their accuracy for different social networks and network metrics," *Scientific programming*, vol. 2015, 2015.

[124] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American society for information science and technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[125] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

[126] M. PALMONARI and P. MINERVINI, "Knowledge graph embeddings and explainable ai," *Knowledge Graphs for Explainable Artificial Intelligence: Foundations, Applications and Challenges*, vol. 47, p. 49, 2020.

[127] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *arXiv preprint arXiv:1709.05584*, 2017.

[128] T. Li, J. Zhang, S. Y. Philip, Y. Zhang, and Y. Yan, "Deep dynamic network embedding for link prediction," *IEEE Access*, vol. 6, pp. 29219–29230, 2018.

[129] A. Mara, J. Lijffijt, and T. De Bie, "Evalne: a framework for evaluating network embeddings on link prediction," *arXiv preprint arXiv:1901.09691*, 2019.

[130] M. Qu and J. Tang, "Probabilistic logic neural networks for reasoning," in *Advances in Neural Information Processing Systems*, pp. 7712–7722, 2019.

[131] S. Dumancic, A. Garcia-Duran, and M. Niepert, "A comparative study of distributional and symbolic paradigms for relational learning," *arXiv preprint arXiv:1806.11391*, 2018.

[132] D. Sileo, T. Van-De-Cruys, C. Pradel, and P. Muller, "Composition of sentence embeddings: Lessons from statistical relational learning," *arXiv preprint arXiv:1904.02464*, 2019.

[133] L. D. Raedt, K. Kersting, S. Natarajan, and D. Poole, "Statistical relational artificial intelligence: Logic, probability, and computation," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 10, no. 2, pp. 1–189, 2016.

[134] K. Kersting, L. De Raedt, and S. Kramer, "Interpreting bayesian logic programs," in *Proceedings of the AAAI-2000 workshop on learning statistical models from relational data*, pp. 29–35, 2000.

[135] D. Koller, N. Friedman, S. Džeroski, C. Sutton, A. McCallum, A. Pfeffer, P. Abbeel, M.-F. Wong, D. Heckerman, C. Meek, *et al.*, *Introduction to statistical relational learning*. MIT press, 2007.

[136] L. De Raedt, A. Kimmig, and H. Toivonen, "Problog: A probabilistic prolog and its application in link discovery.," in *IJCAI*, vol. 7, pp. 2462–2467, Hyderabad, 2007.

[137] M. Richardson and P. Domingos, "Markov logic networks," *Machine learning*, vol. 62, no. 1-2, pp. 107–136, 2006.

[138] M. Kastrati and M. Biba, "Statistical relational learning: A state-of-the-art review," *Journal of Engineering Technology and Applied Sciences*, vol. 4, no. 3, pp. 141–156, 2019.

[139] P. Wittek and C. Gogolin, "Quantum enhanced inference in markov logic networks," *Scientific reports*, vol. 7, p. 45672, 2017.

[140] D. Fierens, H. Blockeel, M. Bruynooghe, and J. Ramon, "Logical bayesian networks and their relation to other probabilistic logical models," in *International Conference on Inductive Logic Programming*, pp. 121–135, Springer, 2005.

[141] T. Sztyler, G. Civitarese, and H. Stuckenschmidt, "Modeling and reasoning with problog: an application in recognizing complex activities," in *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pp. 259–264, IEEE, 2018.

[142] A. Kimmig, S. Bach, M. Broecheler, B. Huang, and L. Getoor, "A short introduction to probabilistic soft logic," in *Proceedings of the NIPS Workshop on Probabilistic Programming: Foundations and Applications*, pp. 1–4, 2012.

[143] S. H. Bach, M. Broecheler, B. Huang, and L. Getoor, "Hinge-loss markov random fields and probabilistic soft logic," *arXiv preprint arXiv:1505.04406*, 2015.

[144] S. Kundu and J. Chen, "Fuzzy logic or lukasiewicz logic: A clarification," *Fuzzy Sets and Systems*, vol. 95, no. 3, pp. 369–379, 1998.

[145] G. Farnadi, S. H. Bach, M.-F. Moens, L. Getoor, and M. De Cock, "Soft quantification in statistical relational learning," *Machine Learning*, vol. 106, no. 12, pp. 1971–1991, 2017.

[146] D. A. Schum, G. Tecuci, D. Marcu, and M. Boicu, "Toward cognitive assistants for complex decision making under uncertainty," *Intelligent Decision Technologies*, vol. 8, no. 3, pp. 231–250, 2014.

[147] M. A. K. Siddike and Y. Kohda, "Trust in cognitive assistants: a theoretical framework," *International Journal of Applied Industrial Engineering (IJAIE)*, vol. 6, no. 1, pp. 60–71, 2019.

[148] A. Viros Martin and D. Selva, "From design assistants to design peers: Turning daphne into an ai companion for mission designers," in *AIAA Scitech 2019 Forum*, p. 0402, 2019.

[149] P. Williams, "A monte carlo dispersion analysis of the x-33 simulation software," in *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, p. 4067, 2001.

[150] E. Baumann, C. Bahm, B. Strovers, R. Beck, and M. Richard, "The x-43a six degree of freedom monte carlo analysis," in *46th AIAA Aerospace Sciences Meeting and Exhibit*, p. 203, 2008.

[151] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM computing surveys (CSUR)*, vol. 51, no. 5, p. 93, 2018.

[152] G. Pilania, P. Balachandran, J. E. Gubernatis, and T. Lookman, "Classification of abo3 perovskite solids: a machine learning study," *Acta Crystallographica Section B: Structural Science, Crystal Engineering and Materials*, vol. 71, no. 5, pp. 507–513, 2015.

[153] P. D. McNicholas, T. B. Murphy, and M. O'Regan, "Standardising the lift of an association rule," *Computational Statistics & Data Analysis*, vol. 52, no. 10, pp. 4712–4721, 2008.

[154] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," *Data mining and knowledge discovery*, vol. 8, no. 1, pp. 53–87, 2004.

[155] D. W. Way, J. L. Davis, and J. D. Shidner, "Assessment of the Mars Science Laboratory entry, descent, and landing simulation," in *Advances in the Astronautical Sciences*, vol. 148, pp. 563–581, 2013.

[156] E. B. Myklebust, E. Jimenez-Ruiz, J. Chen, R. Wolf, and K. E. Tollefsen, "Knowledge graph embedding for ecotoxicological effect prediction," in *International Semantic Web Conference*, pp. 490–506, Springer, 2019.

[157] J. Stebbing, A. Phelan, I. Griffin, C. Tucker, O. Oechsle, D. Smith, and P. Richardson, "Covid-19: combining antiviral and anti-inflammatory treatments," *The Lancet Infectious Diseases*, vol. 20, no. 4, pp. 400–402, 2020.

[158] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, "A review of relational machine learning for knowledge graphs," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 11–33, 2015.

[159] R. Nemani, "Nasa earth exchange: Next generation earth science collaborative," *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 3820, pp. 17–17, 2011.

[160] J. Lee and Y. Wang, "On the semantic relationship between probabilistic soft logic and markov logic," *arXiv preprint arXiv:1606.08896*, 2016.

[161] P. Brugarolas, "Guidance, navigation and control for the entry, descent, and landing of the mars 2020 mission," 2017.

[162] N. Hussein, A. Alashqur, and B. Sowan, "Using the interestingness measure lift to generate association rules," *Journal of Advanced Computer Science & Technology*, vol. 4, no. 1, p. 156, 2015.