PennState

NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data

Alec Dady: Major in CS
Daniel Daniel Donley: Major in SE
James Fennelly: Major in SE

Course Instructor: Naseem Ibrahim
Faculty Advisor: Wen-Li Wang

Industry Sponsor: NASA Psyche Mission - Arizona State University
Project Mentor: Cassie Bowman

A capstone project report submitted to the faculty of
The Computer Science and Software Engineering Department
Penn State Erie, The Behrend College

April 2020
(Version 5.0)

Technical Report Series: PSU-BD-CSSE-Class2020-Sec-001-Team-015

Table of Contents

1. Abstract

The Psyche mission is a space mission to observe a special asteroid thought to possibly be the core of an early planet orbiting the Sun between Mars and Jupiter [1]. The project is organized by Arizona State University in collaboration with the National Aeronautics and Space Administration (NASA). Within the orbiter, Hall thrusters are used as a form of electric propulsion to eliminate the need for chemical fuel. The current problem with Hall thrusters are sensitivities to thruster output, performance, and plume properties as a result of changing background pressures and facility design parameters of HET vacuum test facilities [2]. As researchers, the team will carefully analyze the facility effects data to uncover previously undiscovered correlations and devise a method to predict and possibly correct for the sensitivity if time permits. Various techniques from single variable correlations to multivariate correlation analysis through machine learning will be used. As a whole, this project will contribute to knowledge on hall thrusters and inconsistencies in thruster performance.

To achieve the objective, the team implemented a variety of data preprocessing algorithms, a Kalman filter, and a Deep Belief Network (DBN). The data preprocessing algorithms effectively clean collected SPT data records as well as perform Pearson correlation analysis, Principal Component Analysis (PCA), and other useful tools. Furthermore, the data records are processed through an implementation of the Extended Kalman filter to generate sample data points. These sample data points were utilized to train a multitude of Deep Belief Network machine learning models, which are used for predicting the Hall thruster performance and operation parameters.

With the use of this system, future developers can continue researching the effects of facility parameters on SPT-140 and SPT-100 Hall thruster. The developers will be able to customize their research approach, i.e. the parameters of the data set, the training parameters, etc. This feature allows researchers to ask more varied and detailed questions. Additionally, future developers may have access to more and higher quality data, allowing them to take the conceptual ideas presented here and expand upon them, helping to discover more avenues of research and help answer questions.

In this report we document the output of our requirements engineering process, displaying all user requirements, system requirements, and various diagrams to help describe the theoretical system to be built over the course of the next few months. The sections are ordered as follows. Sec. 2 discusses the changes made throughout the iterations of the report. Following, Sec. 3 will go over the problem statement, providing background information about the business domain. Sec. 4 will go over the user and system requirements the system must conform to, followed by an explanation of exploratory studies performed in Sec. 5. Next, Sec. 6 discusses the design of the system, Sec. 7 details the implementation of the system, while Sec. 8 discusses the test suites and cases used to validate and verify the software. Sec. 9 describes the challenges faced while developing the system. The report concludes with the system manual in Sec. 10, the conclusion in Sec. 11, and the references in Sec. 12.

2. Report Revision History

The following section discusses in detail the changes made throughout the various versions of this report, starting with the changes made from version 1.0 to version 1.5.

2.1. Changes in Version 1.5

Overall, minor grammar and structure mistakes were corrected throughout the report. Likewise, references to sources were added throughout the document where appropriate, then added to the references at the end of the report.

Sec. 3 was altered to better match the description of the project based on the Psyche website and provided resources. The business background was changed to better reflect the project background, objectives, and needs. Likewise, the broader impacts in Sec. 5.3 were also updated.

Continuing, Sec. 4 had text added to Secs. 4.1, 4.1.3, 4.1.4, 4.2, and 4.2.2 discussing the differences between user and system requirements, as well as functional and non-functional requirements. Also, user requirement UP-01 and system requirement SP-01 were modified to the rate at which the data must be evaluated. Their corresponding paragraphs about UP-01 and SP-01 in Secs. 4.2.2.2 and 4.1.4.2 were likewise updated to reflect this change.

2.2. Changes in Version 2.0

With the updates for report 2.0, several sections and subsections were added throughout the document. These sections include the entirety of 8, which covers the testing of the system, and Secs. 6.2, 6.3, 6.4, 6.5, 7.4, and 9.2. Secs. 6.2 through 6.5 cover the structural designs, behavioral designs, and rationale for each design choice, while Sec. 7.4 talks about the implementation rationale. The last major addition, Sec. 9.2, talks about the challenges the team has faced while implementing the system thus far.

The abstract, as well as exploratory studies, were elaborated on to better match the project descriptions and information provided by the Psyche team at Arizona State University. Sec. 3.3, the objectives were also changed slightly to better describe the impacts on individuals. The exploratory studies section was expanded to include the HET performance parameter research. Additionally, more references were added to verify the information provided in the abstract and exploratory studies.

In Sec. 4, the system and user requirements were updated. Notably, the derived system requirements from UF-A have been reorganized into 6 system requirements that better explain what the system is calculating. The dependability requirement in Sec. 4.2.2.2 and Sec. 4.1.4.2 was removed since the industry mentor simply needs the results, and the actual system itself has no time-constraints it needs to abide by. Instead, the requirement was replaced with a security requirement. This requirement ensures the security of the generated models so they cannot be

modified or deleted by anybody but the developers. Lastly, the use cases were updated and the 'Correct Sensitivity' use case was added.

2.3. Changes in Version 2.5

Overall, references were added to increase the validity of the content of the report. Additionally, grammatical errors were corrected throughout the document and some sentences rewritten for clarity. Fig. 5 previously contained a dependency link between the DataInputSystem and DataNormalizer modules. This link has been changed to an associative link since the DataInputSystem will need a DataNormalizer module.

In Sec. 6.4, the behavioral design of the system. All of the corresponding diagrams from figures 11 through 16 have been modified to contain the actual class objects at the top, as well as better reflect the data flow between the objects. Sec. 8.3.1's second paragraph was altered to explain the unit testing report in more accuracy, as the previous explanation did not reference the right image. Lastly, Sec. 9.2 was altered to clarify the challenges faced during development.

2.4. Changes in Version 3.0

Within the requirements, the only revised user requirements include UP-01, being removed since only the current development team will have access to the models. The team will only submit a report describing the effectiveness of the research efforts and methodology over the course year. For the system requirements, SF-A-02 was removed to fit a revised understanding of our outputs for the system. System requirement SF-A-06 was also changed from Deep Q Learning to utilizing a Kalman filter to estimate output parameters. SF-B-01 was grammatically updated to improve clarity for the given requirement. SF-C-01 to SF-C-08 were also revised/added to reflect how the data input system will be built. SF-D-01 to SF-D-05 were revised/added to reflect the current state for machine learning algorithms and data mining techniques.

Within the use cases, UC-006 and UC-005 were merged into one single use case contained into UC-005 because automatic report generation is too costly in terms of time and man-hours, so only the calculations themselves are saved along with any generated figures. UC-007 was removed because the correct sensitivity use case can be considered a subset of the model evaluation. UC-006, which was the report generation use case, was removed to match the project boundary.

For each of the test suites, TS-001 (Data Input System) was converted into Data Cleaning with more test cases added to represent our current system. TS-002 was revised to better fit each function within the DBN Learner, including an integration test case also being added. TS-003 was added for unit testing the single variate analysis functions. TS-004 was added to include unit testing for each Kalman filter function and integration testing for the entire module.

In Sec. 5.3, the relevant techniques section includes the data science technique of generating data using the Kalman Filter, as well as the FilterPy package being added in Sec. 5.1.

In Sec. 6.2, the class diagram of the scraper was edited in Fig. 4 to reflect the current iteration of the scraper's functionality. The isGraph AI was removed due to the feature being unnecessary and not an efficient use of the team research time. The body paragraphs of Sec. 6.2 were changed to reflect this change. Also within Sec. 6.2, the envelope pattern for the NoSQL database was removed in an effort to simplify the data input process. Since the data could be easily contained within a CSV file hosted on Google documents, a NoSQL database is not useful with so few data points and such a small user base utilizing the system. In Sec. 6.5, the design alternatives and rationale were revised to meet project changes to remove the NoSQL database with a CSV formatted spreadsheet. Also in Sec. 6.5, the decision rationale behind the Kalman Filter and the alternatives such as extended Kalman Filter and FIR Filter have been added.

In Sec. 6.3, the graphical user interface previously located within Fig. 10 was replaced with a console-based user interface to improve simplicity for the project, since only the team will be actively using the user interface, so a graphical interface is considered a luxury feature. The industry sponsors are only interested in a final report at the end for the results from the given system, so any features implemented will go to waste unless used by the team for research purposes.

In Sec. 7.4, the Kalman Filter system implementation description was added. The comparisons between the Kalman Filter and the extended Kalman Filter as well as the Kalman Filter and the FIR Filter were added.

Sec. 8 was updated to include changes to the test suites, test cases, and test execution reports. The test suites were updated and restructured to better organize how tests were being carried out in Sec. 8.2.1. Several new test cases and test execution reports were added to Sec. 8.2.2 and Sec. 8.3.1 for testing the Kalman filter and data cleaning functionality. Additionally, integration test cases were added in Sec. 8.2.3, and the corresponding test execution reports in Sec. 8.3.2.

In Sec. 9.2, a few new challenges the team faced were added. The first involved the implementation of the Kalman filter, the second about setting proper project boundaries to effectively organize work.

2.5. Changes in Version 3.5

Requirements SF-C-07 & SF-C-08 were modified to be more concise on normalization versus standardization. Requirement SP-02-01 was updated to clarify that the SPT-140 is the thruster being used on the Psyche spacecraft. The execution report of TC-016 now includes execution steps.

A transition sentence was added in Sec. 5.3 to explain what the Psyche spacecraft will use to reach the asteroid. Sec 9.1.3 was edited to explain the challenge faced using clearer terminology. Sec. 2.4 was edited to better explain the changes made in report 3.0. Statements were added to Sec. 8.2.2 and Sec. 8.2.3 for clarification.

2.6. Changes in Version 4.0

The first major change added was the addition of two paragraphs in the abstract. The first goes over the main functionalities of the system, while the other discusses the broader impacts of the system on the scientific community.

Continuing, Sec 6.4, the behavioral design of the system, was refactored to match system operation as it currently stands. The corresponding figures were likewise modified as detailed throughout Sec 6.4. Sec 7.5 was added to explain key implementations of preprocessing and the configuration of the Extended Kalman filter.

In addition to this, Sec. 8 was thoroughly updated to include new test suites, test cases, and execution reports. Additionally, execution steps were added to Sec. 8.1.2 to explain how other test cases operate. The test suites were refactored and updated in Sec. 8.2.1, the unit test cases added to Sec. 8.2.2. Furthermore, new sections were added to reflect the addition of system test cases, and acceptance test cases. The system test cases were added in Sec 8.2.4, while the acceptance test cases in Sec 8.2.5. The execution reports were likewise updated to reflect the new additions. Sec 8.3.3 was added for the system testing execution reports, while Sec 8.3.4 covers the acceptance testing execution reports.

A section was added regarding the open issues within the system. This is Sec 9.3 and contains information on current problems and potential solutions.

The final major change came in the additions of Sec. 10.2.1, the platform requirements, as well as Sec. 10.2, the system deployment, and the resulting subsections. These subsections include Sec 10.2, the platform requirements for the system, as well as Sec. 10.2.2, system installation instructions. Finally, Sec 10.3 was added to give instruction to end users, the future developers.

2.7. Changes in Version 5.0

In Sec 6.4, the description of Fig. 11 and Fig. 15 were updated to better describe their respective sequence diagrams. Additionally, the time-complexity of the preprocessing analysis described in Sec. 7.5 was corrected to the proper big-O notation. Further clarification was added to Sec. 8.3.4 to explain the difference in the system and acceptance test cases and execution reports.

Sec. 11 was added to provide a conclusion to the technical report. This section adds Sec. 11.1, the achievements from the project, Sec. 11.2, the lessons learned by the developers, as well as Sec. 11.3, a brief acknowledgment of the factors that led to the success of the project. Finally, an additional appendix was added to cover the analysis of the recorded results of using the system. This appendix is called “Appendix RA” and is located immediately after Sec. 12.

3. Problem Statement

The following section provides a series of information regarding the problem statement. The problem statement includes the business background, project needs, and objectives throughout the duration of the capstone design project.

3.1. Business Background

The Psyche mission is a space mission to a metal asteroid orbiting the Sun between Mars and Jupiter [3]. Psyche is both the name of an asteroid orbiting the Sun between Mar and Jupiter, as well as the name of this mission to visit the asteroid. What's interesting is that Psyche appears to be the exposed nickel-iron core of an early planet, one of the building blocks of our solar system. The Psyche spacecraft is targeted to launch in summer 2022 and travel to the asteroid using solar-electric (low-thrust) propulsion, arriving in 2026, following a Mars flyby and gravity-assist in 2023. After arrival, the mission plan calls for 21 months spent at the asteroid, mapping it and studying its properties [1].

Throughout the mission, the orbiter will also attempt to determine: if the asteroid is a core or just unmelted material, the relative ages of regions of asteroid Psyche's surface, compare the various features to that of Earth's core, and characterize Psyche's topography [4]. To study the asteroid, the orbiter will contain a Multispectral Imager, Gamma Ray and Neutron Spectrometer, and a Magnetometer.

To reach this asteroid, NASA will rely on the use of Hall effect thrusters (HET), a form of solar-electric propulsion. The Hall thrusters will allow for cheap and efficient travel over long distances, making this NASA Psyche mission possible without the necessity of large fuselages primarily for storing chemical fuel. The Psyche mission will also test a new form of communication called Deep Space Optical Communication (DSOC), which utilizes high-rate communication by sending lasers between the spacecraft and ground stations on Earth [3]. In theory, Deep Space Optical Communication should allow for more information to be sent and received within a smaller amount of time as compared to normal radio transmission.

3.2. Needs

The popularity of HET has led to an increase in the number of HET vacuum test facilities. Unfortunately, these HET test facilities are unable to perfectly recreate the pressures experienced in space. Since Hall thrusters are very sensitive to these pressures, they can experience multiple different effects while being tested in these HET vacuum test facilities, impacting their performance [2].

HET operation, performance, and plume properties are dependent on test facility configuration, justly named "facility effects" [2]. These facilities can vary in geometry, materials, pumping capacities, placement, and other parameters, causing significant changes in HET operation.

Despite current physics-based models, no universal explanation has yet to explain all of these observed effects. To correctly understand how the HET operates in these facilities, some model or solution must be developed.

3.3. Objectives

Throughout the course of the project, various statistical approaches will be taken to analyze publicly released research data on the Hall Effect Thruster Facility Effects to determine the cause of sensitivity in controlling the thrust and other performance parameters. The team will apply machine learning and data mining techniques to uncover previously undiscovered correlations in data sets and devise a method to predict and possibly correct for this sensitivity. The team will also keep documentation on any statistical or machine learning model developed, analysis of the machine learning models, and conclusions for the research.

Development will be split into two main phases, with the first phase focusing on collecting the initial set of data as well as designing the tools to train and deploy different types of machine learning algorithms. The team will focus on gathering data from the plethora of research material available, while simultaneously developing the proposed system to train and deploy models, as well as generate reports consisting of the training and evaluation of the models. To help facilitate data collection, the team will develop a side tool, called a scraper, to gather abstracts from websites to pull data from. The second phase will have the team test different machine learning models and data mining techniques to discover correlations within the facility effects data, as well as devise a method to predict the changes in the HET performance, output, and plume properties.

4. Requirements

The following section details the requirements that need to be implemented. These requirements are the main features and functional constraints the system must conform to function properly. The first portion, Sec. 3.1, will describe the user functional and non-functional requirements, including a glossary of important domain terminology and the key user groups. Secs. 3.2 will discuss the elicited functional and non-functional system requirements.

4.1. User Requirement

This sub-section covers the user requirements; the user requirements are statements written in natural language to be later refined into system requirements in Sec. 4.2.

4.1.1. Glossary of Relevant Domain Terminology

16 Psyche: The asteroid that the spacecraft will be orbiting. Psyche is one of the largest asteroids in the asteroid belt and is believed to be the iron-nickel core of a protoplanet.

Deep Space Optical Communication (DSOC): A new and sophisticated form of communication, Deep Space Optical Communication encodes data in photons as compared to radio waves in normal radio transmission, allowing for higher data rates [3].

Electric Propulsion: A form of propulsion that uses electricity, rather than chemical components, to accelerate a propellant.

Hall Effect Thrusters (HET): A type of rocket thruster that uses electric propulsion to operate at high thrust efficiency and density

HET Facility Effects: The observed impacts experienced by the Hall effect thrusters at HET vacuum test facilities.

HET Vacuum Test Facilities: Commonly called HET Facilities, a site with a vacuum-sealed area for simulating the effects of space and testing the performance of Hall effect thrusters.

National Aeronautics and Space Administration (NASA): The federal agency responsible for aeronautics, research into aerospace, and other important space programs.

Stationary Plasma Thruster (SPT): A type of Hall effects thruster that creates a stream of charged particles in the form of plasma to provide the electric propulsion [5].

4.1.2. User Groups

The primary group of users utilizing the system are the developers. The developers will need to upload facility effects data and clean the data in order to evaluate the dynamics of the system. The developer will also be able to run a Kalman Filter, or some other filter in order to generate more data. As such, the developers will input data, train the model, and deploy the machine learning model. The developers merely need to discover effective models to hand off to the researchers, who will then take the results to conduct more research. Since the developers will further expand on the system as a whole, with new machine learning models and/or other statistical analysis techniques, this makes them the primary users of the system.

In terms of experience, the developers are novices in the business domain and journeyman in the technological domain. The developers have no experience in electric propulsion and have most likely never heard of Hall thrusters before this project. They have an understanding of some physics but must learn more about the project domain such as how SPT-140 thrusters operate. Their technological domain knowledge is at the journeyman level. The developers all have experience and a good understanding of machine learning tactics but must conduct more research on the advantages and disadvantages of different model types to implement the most effective network.

4.1.3. Functional Requirements

Sec. 4.1.3. will discuss the user functional requirements associated with this project. Functional requirements are statements of services the system should provide, how the system reacts to certain inputs, as well as how the system needs to behave in particular situations.

4.1.3.1. Project Scope

Fig. 1 below demonstrates the interactions the developer can have with the system. The developers can begin by launching the ‘Clean the Data Set’ case, where data is uploaded and formatted into an acceptable standard for further processing. After this formatting, the system will perform correlation analysis on the data set by calling the ‘Correlate the Data Set’ case. The developer can then launch either the ‘Run Kalman Filter Script’ case or the ‘Evaluate DBN Model’ case to process the data and extract useful information.

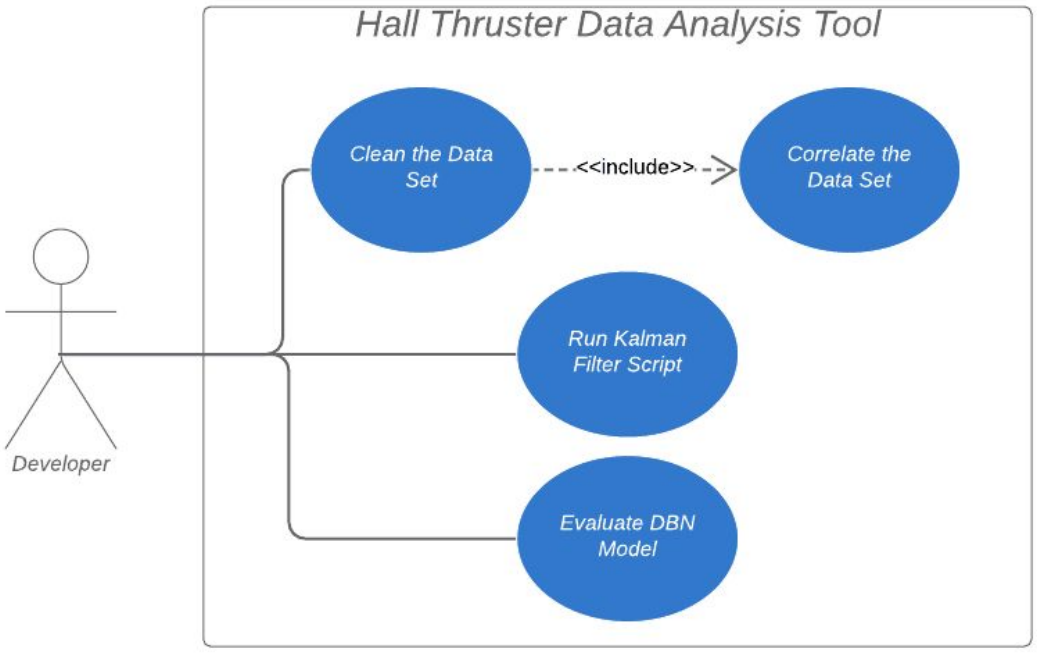

Fig. 1. Use Case diagram of main user-system interactions.

4.1.3.2. User Scenarios

Appendix U gives an overview of the user scenarios, as well as tables detailing their descriptions. Table 4.1 shows all five user scenarios, consisting of one summary use case, 3 primary tasks, and 1 subfunctions. The actor featured in this use case is a developer, since the developers have access to all user scenarios found in the system.

Starting, Table 4.2 begins with the developer wishing to utilize the system to process data and extract information using various algorithms, UC-001. The developer interacts with the system through launching the preprocessing script. Afterward, the system will begin UC-002, UC-003, or UC-004. The developer only needs to process the data once in UC-002 before beginning any other user scenario.

Moving on to the primary task, Table 4.3 explains the process for cleaning, transforming, and reducing the data set During the process, the system loads in the various CSV files containing the SPT data. If the data is not present, the developer will receive a notification. After providing valid data, the system will then clean, transform, reduce, then save the data into the system by removing unnecessary columns, outliers, and fill in missing values with appropriate ones.

As for the next primary task, Table 4.4 covers UC-003, running the cleaned data set through the correlation analysis algorithms. After data is successfully uploaded, the system will call the necessary functions needed to calculate the various correlation artifacts. This includes scatter plots, Pearson correlations, Principal Component Analysis correlations, and Lasso regression correlations. The system then records the results into local storage.

The final primary task is covered in Table 4.5, and it involves running the Kalman filter script to process the data and extract estimates of the output as well as potential data points. The system will begin by calculating the initial state the Kalman filter should be in, followed by obtaining the lengths of those state vectors. After creating the Kalman filter object, the system then calculates the necessary covariance and uncertainty matrices to be used by the filter. When all the required values are given to the filter, the next state is estimated, and data points are extracted from this state. The state and data points are finally saved onto the local storage.

While the system is evaluating the model, a few steps will be taken to assess the capabilities of the trained machine learning model as well as constructing the Deep-Belief Network (DBN) model. Table 4.6 goes over these steps in the first subfunction user scenario. The system begins this subfunction by constructing and compiling the model to create a DBN model. Afterward, the system begins training the model by fitting the data to the model. This process can take hours depending on the dimensional complexity and number of records in the data set. The system will then evaluate the accuracy of the model, then serialize the model, weights, and results in local storage.

4.1.3.3. User Functional Requirements

Appendix R goes into detail regarding the different functional user requirements in the Hall Thruster Data Analysis Tool. The first major requirement is in Table 4.7, requirement UF-A. The desired outcome of this project is to be able to predict and/or correct a sensitivity experienced by the Hall thruster that cannot be predicted using current scientific models. The user requirement mandates the system to be able to perform this action. However, it is possible that no machine learning techniques can accurately model the facility effects data well enough to be effective in predicting and minimizing the sensitivity.

Continuing, Table 4.8 details the requirement UF-B, where a report of the machine learning training, evaluation, and deployment must be generated. This report will detail the model's accuracy and predictive capability, allowing users to determine if the trained model is appropriate to solve a problem. The report will need such information as the training data utilized, the analysis generated by the machine learner, and its accuracy. This report is of the highest priority since the model cannot be determined to be effective unless the results are compiled into a readable report and examined by a human.

The next major user functional requirement describes what kind of data must be used by the machine learning model to produce the desired output. Table 4.9, or requirement UF-C, states that information and data should be obtained from HET facilities that utilize Hall thrusters in their simulations. This data will be the basis of training and evaluating machine learners, and will hopefully have hidden correlations that can be used to explain the scientific phenomenon.

Table 4.10 and requirement UF-D goes over how the client wishes to solve the sensitivity issue. Essentially, the system will make use of machine learning models to find undiscovered correlations in published data sets. Current physics models are unable to calculate this experienced sensitivity, so a machine learner will perform this tedious investigation. Alternatively, the machine learner may be unable to find any correlations or scientific findings, so data mining techniques can also be performed on the dataset in the event few machine learning options are remaining. Data mining techniques are capable of performing operations on large data sets, allowing them to discover previously unknown correlations and patterns hidden within data sets.

4.1.4. Non-functional Requirements

The following subsection discusses the non-functional requirements incorporated into the system. Unlike functional requirements, non-functional requirements are constraints and expectations on the services and functions provided by the system. These constraints can include restrictions on the timing of system operations, how the system must be developed, etc.

4.1.4.1. Product: Usability Requirements

Starting off the non-functional requirements, the client wishes that the system's data set the first focus of the Hall thruster type being used in the Psyche space-craft. This constraint on the data set is outlined in requirement UP-02, located in Table 4.11 of the appendix. The thruster type that the client desires to focus on first is the SPT-140. If possible, we may also include the SPT-100 thruster type in the initial data set as well. If the system is successful in correcting or predicting the sensitivity, the data set should be expanded to other Hall thruster types to determine if similar results can be produced.

4.1.4.2. Organizational: Operational Requirements

Continuing, Table 4.12 describes the operational user requirement associated with the system. To be effective in every HET vacuum facility, the system must be able to function properly on computers so long as they correctly install the software. The requirements on what computer specifications need to be considered are detailed in Sec. 3.2.2.3.

4.1.4.3. External: Legislative Requirements on Safety/Security

There is only one legislative requirement the system must follow to properly obey the law. As per Public Law 112-10, Section 1340(a) and 112-55, Section 536, participants cannot be citizens

of the People's Republic of China (PRC). This requirement is documented in Table 4.13. This means that the main developers of the Hall Thruster Data Analysis Tool cannot have citizenship in the People's Republic of China. The reasoning for the law is unknown but must be followed to ensure software engineering ethical guidelines are being followed. Anybody developing the system with citizenship from the PRC must be ejected from the development process.

4.2. System Requirements

Sec. 4.2 covers the system requirements that are implemented into the project. System requirements are structured documents that give detailed descriptions of the system's operations and constraints. In essence, the system requirements define what should be implemented into the system.

4.2.1. Functional Requirements

The system functional requirements can be split into four sections. System requirements from user requirements UF-A, UF-B, and UF-C will be discussed further in Sec. 3.2.1.1 as those are the main features the system will implement to accomplish its goal. System requirements from user requirement UF-D will be discussed further in Sec. 3.2.1.2 as this requirement provides restrictions on the data set and data to be utilized.

4.2.1.1. System Functional Requirements

Starting off the system functional requirements, Table 4.14, Table 4.15, and Table 4.16 detail requirements SF-A-01, SF-A-03, and SF-A-04, the output parameters expected to be produced by the machine learning model. The outputs include the thrust of the HET, the specific impulse of the thruster, and the efficiency of the thruster. Only these three parameters will be calculated for now as they are the ones best understood by the team.

Table 4.17 and Table 4.18 requires the system to implement two techniques believed to be beneficial in predicting the output parameters. The first requirement in Table 4.17 has the system implement a Deep Belief Network (DBN) to extract deep hierarchical representations of the data to potentially produce a model with high predictive capability [6]. The other requirement, from Table 4.18 uses a Kalman Filter in order to provide an estimate of what the next state, or array of values, will be next [8].

Continuing, the next grouping of system requirements all stem from UF-B, the data that needs to be gathered to create the final report for ASU. The refined system requirements all detail the different portions of the report that need to be collected and compiled together. Table 4.19 describes the first main requirement. After running either a data mining technique or machine learning model, the results of the model will be recorded and stored in local storage for the team to examine and include in the report. The information recorded for the report is stated in Table 4.20, Table 4.21, Table 4.22, and Table 4.23. Table 4.20 requires the recording of the training

data the model is trained with. Table 4.21 requires that the model type and resulting evaluation be recorded, while Table 4.22 requires the model's accuracy in predictions as well as the predictive capability to be measured. Lastly, Table 4.23 requires the system to record any correlation data extracted from the data set. This correlation data can be useful to examine the usefulness of the data, as well as point to possible areas of research. The combination of all of these system requirements allows reports detailing the effectiveness of each model trained or technique used to be examined and compared against other results. This, in turn, meets all required functionality for UF-B.

Lastly, the final grouping of system requirements are all derived from UF-D. UF-D states that the system must utilize machine learning algorithms or data mining techniques to achieve the correct output. Starting, Table 4.31 speaks to requirement SF-D-01. A basic way to discover correlations is to use Pearson correlation analysis, which measures the linear relationship between two variables [9]. Requirement SF-D-02, shown in Table 4.32, is another correlation analysis technique. In this technique, the correlations of all of the variable combinations are displayed on a heatmap to examine which variables have distinguishing patterns. Table 4.33 shows the next requirement, which is using Principal Component Analysis (PCA) in order to determine feature correlations and measure the dimensionality of the data set [12]. Table 4.34 and Table 4.35 also cover requirements for generating correlation data. Table 4.34 calculates the Lasso Correlations within the data set, while Table 4.35 has the system generate scatter plots to examine the type of relationship between the variables. The combination of all of these requirements completes UF-D, allowing the developers to discover any correlations within the data set.

4.2.1.2. Data Requirements

The only functional data requirements come from user requirement UF-C. From UF-C, several system requirements were derived to handle the raw data and transform it into a usable form by the various techniques and machine learning models. Requirement SF-C-01 in Table 4.24 requires that the data that is read into the system via a CSV file. The reason for this is explained later in Sec. 6.

The next requirement in Table 4.25 has the system to remove data columns with more than 60% of the values missing. Since analyzing data is the main focus of this project, it's vital not to throw out any data collected regarding the subject matter. However, if any of the values are missing, it becomes hard to extract any useful information from that variable. In this case, it's important to remove these data columns from the data set to be used so no wrong conclusions are drawn. The same can be said for Table 4.26's requirement where the system can remove nominal data columns, nominal being value with no ordinance or sense of scale, or reference point. Data like this cannot be useful for calculating correlations or several other techniques but can be useful for techniques like clustering.

The next two data requirements are covered in Tables 4.27 and 4.28. Requirement SF-C-05 in Table 4.27 has the system replace missing values where most of the values are known with estimations based on the value's type. For example, since nominal data has no ordinance, reference, or scale, missing values can only be replaced with the mode or most common data value. Interval and ratio values can be replaced with either the mean or median of that column. Table 4.28's requirement is similar to the previous table's requirement, except it replaces outliers identified with PCA with the appropriate mathematical function instead.

For the final two requirements described in Table 4.29 and 4.30, the data set must be transformed in order to account for the different scales between the variables. This can be done either using normalization techniques, as specified in SF-C-07 of Table 4.29, or standardization techniques, as specified in SF-C-08 in Table 4.30. These two requirements will ensure that the data will be in an acceptable format for processing.

4.2.2. Non-functional Requirements

As stated previously, non-functional requirements are constraints placed on the operation and development of the system. The following subsection goes over the system non-functional requirements, including product, organizational, and external requirements for the system.

4.2.2.1. Product: Usability Requirements

There are two non-functional usability requirements for system operation that place constraints on the data to be used inside the data set. Table 4.36 speaks to requirement SP-02-01, where the Hall thruster type used in the data set must be SPT-140 or SPT-100, where the SPT-140 type thruster is the type being used in the Psyche spacecraft. The other usability requirement is seen in Table 4.37, requirement SP-02-02. This requirement allows the system to utilize data from other thruster types, under the assumption that the rest of the system is fully operational. If the problem is solved with the thruster types SPT-140 and SPT-100, then the team is advised to see if a similar solution can be expanded to these thruster types.

4.2.2.2. Organizational: Operational Requirements

Continuing with how the system must operate, three operational requirements can be derived from user requirement UO-01. Going in order, Table 4.38, Table 4.39, and Table 4.40 require the system to run on the operating systems Linux, Windows 10, and macOS, respectively. The system must be capable of running on any system. Considering that the Python packages are supported on any platform, the only difference in a system that must be considered is the support for different operating systems. Since ASU and the HET vacuum test facility researchers may use any of these platforms, it's necessary to require each operating system to ensure the system can operate on them.

4.2.2.3. External: Legislative Requirements on Safety/Security

As previously stated in Sec. 3.1.4.4., user requirement UE-01 states citizens of the PRC must not be allowed to develop the Hall Thruster Data Analysis Tool. As such, the only system requirement that can be derived from this user requirement is seen in table 4.41, system requirement SE-01-01. This system requirement simply restricts the development of the system to those who do not have citizenship in the PRC per the law.

4.3. Requirements Trace Table

The final table in Appendix R in Table 4.42, which gives a mapping table for all of the user & system requirements talked about in the previous sections. The user requirements are displayed on the left with descriptions to match. On the right half of the mapping table are the corresponding system requirements that were derived from the user requirements.

Additionally, it's important to note a small graphical error produced in Appendix R's Table 4.42. Upon inspection of the table heading, the indentation of the line starts at the left side of the page as opposed to on the margin of the page. This is most probably due to a report generation error inside CapStone.

5. Exploratory Studies

The following section contains significant research related to system architectural design, setting up the environment, and broader impacts of the project. The section will start by discussing relevant techniques needed to create the system, then continue to the relevant packages needed by the system. The section then concludes by discussing the broader impacts of the Hall Thruster Data Analysis Tool.

5.1. Relevant Techniques

Some relevant techniques that will be applied throughout this project are mainly Machine Learning techniques, such as supervised learning, unsupervised learning, and reinforcement learning. This would involve potentially implementing a Q-Learning algorithm to train the model to make predictions based on a reward that the team will have to account for the model to learn and improve.

The team will also have to look at which Neural Network makes the most sense to be applied in the system, this could range from a Multilayered Perception (Classic Neural Network) to any Recurrent Neural Network, and could potentially lead the team to design a Hybrid Neural Network. The team will consider all of these avenues when attempting to solve the problem. The starting point the team believes would help solve the problem would be to implement a Deep Belief Network (DBN) that can extract deep hierarchical representations of training data sets [6]. These DBN networks can reconstruct its inputs when trained without supervision, then trained for classification with supervision. However, if Time Series Analysis is critical we might have to incorporate a Recurrent Neural Network. This would indicate an LSTM model that would keep track of previous states in memory in pursuit of its goal.

To better understand what the machine learning model needs to calculate to solve the issue described in Sec. 3.2, the objectives of the project. To accomplish this, several abstracts, such as Frieman's dissertation on neutral flow ingestion, will be studied to better understand what facility effects and HET operation parameters are necessary to produce an accurate model [2].

5.2. Relevant Packages/Products

Some relevant packages that the system uses are Keras, Tensorflow, Numpy, Matplotlib, FilterPy Seaborn, and Pandas. Numpy is a mathematical library that supplies functionality for efficient scientific computations in Python [7]. Numpy contains several useful features, such as three different types of correlation techniques, as well as a multitude of other tools needed to perform data analysis on data sets. Pandas is essential to this project as it is a library that allows for reading and writing to and from datasets [10]. Without this package, the team would not be able to begin attempting to solve the problem. The data would be unable to be processed and used in a machine learning model. The FilterPy package allows the team to use multiple data filters such as the Kalman Filter, or extended Kalman Filter easily by creating them as objects [11]. This cuts

down on lines of code along with man-hours due to the fact that the functions for predicting the future state and updating the matrices are already built-in with the package.

Matplotlib and Seaborn are both data visualization libraries that may be used to help the team visualize the correlations inside the dataset [13][14]. Within Matplotlib and Seaborn, they are capable of performing analysis on datasets using correlation scatter plots, scatter matrices, heat maps and many more. Keras and Tensorflow are the Deep Learning libraries that allow for the creation of Neural Networks and other Deep Learning techniques [15][16]. This will allow easy implementation of commonly used machine learning models.

Some products that will assist the team in the development process are the Anaconda Navigator and the Anaconda Cloud. The Anaconda Navigator allows for the creation of multiple different Python Environments with different packages installed [17]. It also allows for easy package installation and easy activation of the Python Environment itself. On the other hand, the Anaconda Cloud will be used to make sure that the team works within the same Python Environment. The Anaconda Cloud allows for Python Environments to be uploaded and shared with the members of the development team so that everyone works in the same environment with the same packages installed. This minimizes error within the development process as some packages conflict with one another which could lead to an entire system breaking and having to uninstall python and all packages associated and reinstalling.

5.3. Broader Impacts

Because we cannot measure Earth's core directly, Psyche offers a unique perspective into the violent history of collisions and accretion that created terrestrial planets [1]. Since Psyche may be the core of a protoplanet, data collected during orbit will provide information on the building blocks of the solar system. The mission will help society and organizations around the globe delve into a new field of research by exploring a new type of world made of metal.

In order to reach the asteroid, the Psyche spacecraft will make use of Hall effect thrusters (HET) to travel with low fuel consumption. These thrusters are tested in vacuum facilities designed to simulate deep space. However, the performance of these thrusters does not match what is expected due to unknown phenomena. In the absence of a physics-based explanation for these facility effects, data-driven modeling might provide a method for analyzing the wealth of existing facility effects data and producing a model that can be used to accurately predict these effects. This, in turn, can lead organizations to eventually discover a physics-based solution to these phenomena, opening new avenues of research.

On the individual level, this tool will provide an analysis of the facility effects data utilizing SPT-140 thrusters, providing evidence for researchers to conduct different types of research. Although it doesn’t impact many people, the tool is still useful for individual researchers to explore new avenues in attempting to understand the impacts of the facility parameters on Hall thruster performance.

6. System Design

The following section will cover the entire system design as of the current sprint. At the moment, the current relevant design consists of only the Architectural Design.

6.1. Architectural Design

The architectural design that the system uses is the Data-Model-Learner (DML), that splits up the system into three different components, the Data, the Model, and the Learner [18]. In addition to the DML pattern, a view module is included to visualize the data as well as display the generated reports. Fig. 2 below shows this design pattern in action. The Data component named the Data Management subsystem takes charge of managing the dataset that the system utilizes. This is where the input for the Learner is generated for training. The next component in the design is the Model component, named Data modeling. The Model is the Machine Learning model that takes an input and predicts an output. The model will handle functions for the specific model and for preprocessing data. The model itself is a static entity. The final part of the pattern, the Learner, is a class that runs learning tasks on the model given data. The Learner, in this case, is named the Data Analysis subsystem. The Learner component is where all of the Machine Learning happens within the system. Additionally, it takes the data from the Data component and produces a Model component that will be used to make a prediction. The View model is not included in the design since all data-related matters are to be handled inside the Data module itself, including visualization and utilization of the data. The Data module is more accurately referred to as Data Management, since it must offer a convenient way to interface with the data, such as a graphical user interface [18].

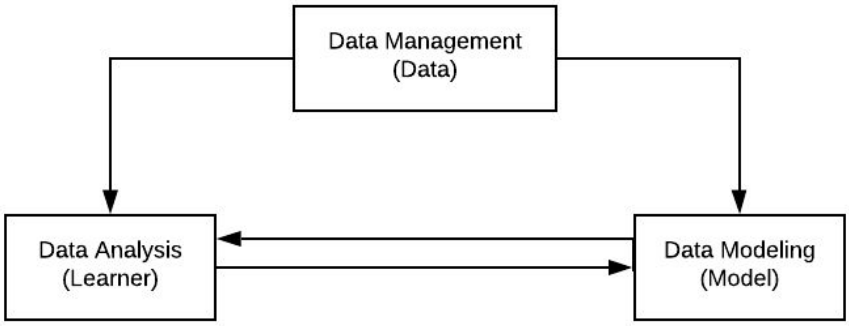

Fig. 2. The Data Model Learner (DML) design pattern for the system architecture.

Utilizing the architectural design pattern shown in Fig. 2, the component design for how the major subsystems will operate with each other is shown below in Fig. 3. Within the System Architectural Design, it can be seen that the published datasets will first be normalized into a

single format to allow for easier data handling. Next, single variables will be correlated against thrust within the Correlation Handler System, routing the output to the Python Data Graphing/Exporting System to help visualize the output datasets. Running Separately from the Correlation Handler System, normalized datasets will also be distributed to each model and learner system to follow the MDL design pattern. The Correlation Analysis Learner system will have higher priority over the Sensitivity Adjustment Learner System, as the adjustment system will take the output of the Correlation Analysis System as its input.

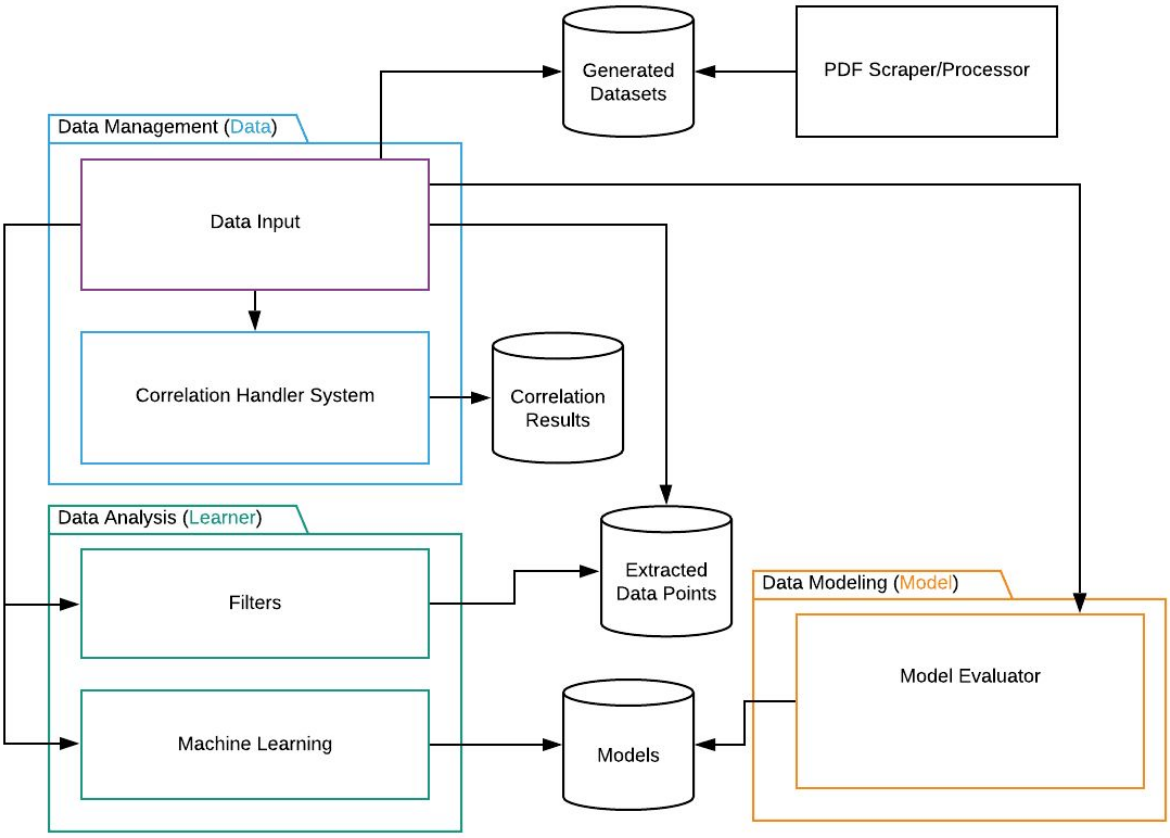

Fig. 3. The component view of module interaction within the system.

6.2. Structural Design

Within this section, the team will explore each of the components previously shown within Fig. 3 that are relevant to our current progress. The team's main focus for implementation was to focus on the Data and Learner modules, specifically the Data input, Filters, and Machine Learning systems. To facilitate data collection, the team also chose to implement a data scraper, with the structural design for the scraper provided below in Fig. 4. This section will additionally cover the

data model used to store the information from the initial dataset that will be fed into a machine learning model, as well as how the models themselves will be stored.

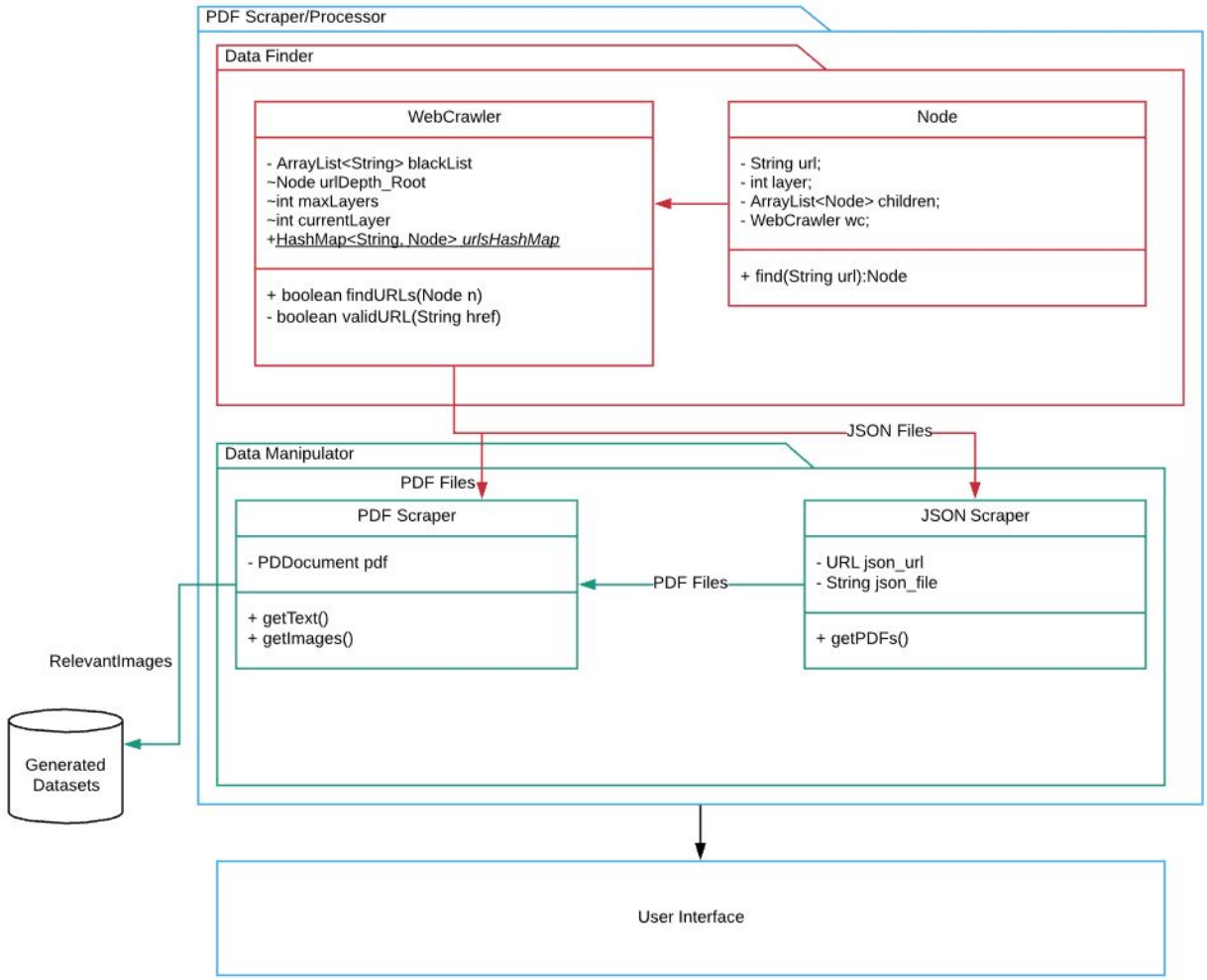

Fig. 4. The structural design of PDF Scraper/Processor component.

Within the PDF Scraper/Processor component as shown in Fig. 4, the scraper works by utilizing the library JSoup-1.12.1 to make connections to any given URL and analyze the HTML page [19]. In order to effectively crawl the page, a generalized tree structure is used to load all URLs found on the page into child nodes for each URL node. Utilizing the Breadth-First Traversal technique, the tree is dynamically built to a specified max depth to prevent infinite web crawling. To prevent loops from occurring within the Tree, a HashMap was utilized to ensure the uniqueness of each link. After debugging, the team noticed that it was typically getting off track, visiting websites that had virtually no possibility of a valid PDF. To fix this time complexity issue, a blacklist has been implemented to avoid checking any link with forbidden domain names. With each URL being visited, as PDFs were found, they are automatically sent to the PDF Scraper to immediately save the PDF, relevant text, and images to a predefined directory

structure. Similarly, with JSON files, any link relating to a PDF would then be routed to the PDF Scraper class. For each PDF file, the images were routed through a relevant image discriminator implemented within the User Interface, helping to quickly identify images for data processing. All graphs are saved in a subdirectory relative to the PDF's original domain name.

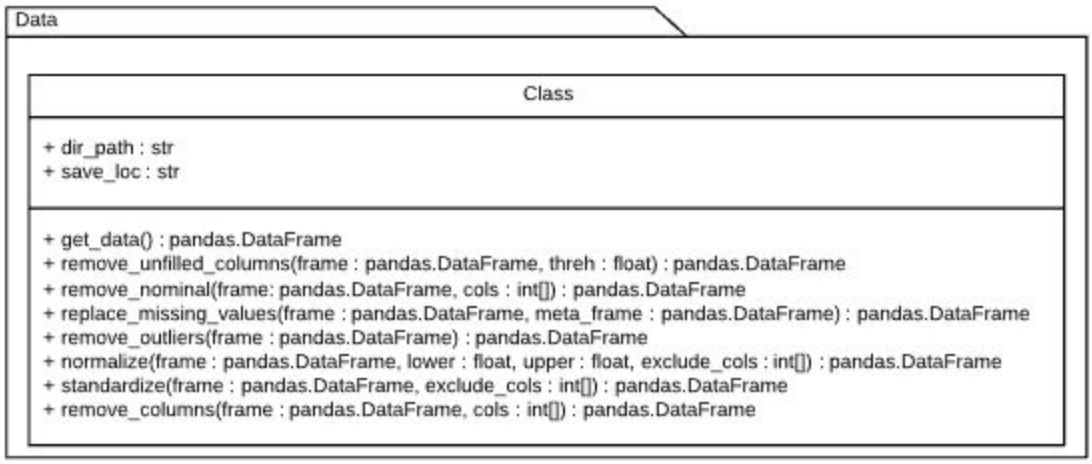

Fig. 5. Data Input package structural design.

Fig. 5 above shows the design for the data input class within the Data package. According to the DML pattern design guidelines, all data preprocessing and interfacing should be done within the Data package [5]. The class diagram will need functionality to interface with and process data. The data_input class will use the Visitor design pattern to provide the functionality for processing the various forms of data as seen in Fig. 3, as well as being able to access and navigate their folder structures.

The package shown in Fig. 6 is the Filters package. The team plans to implement multiple different filters, such as the Kalman filter, Extended Kalman filter, and FIR filter. To make things easier, the Filters package will utilize the Factory Method design pattern, which will allow for easy implementation of more filter types if time permits.

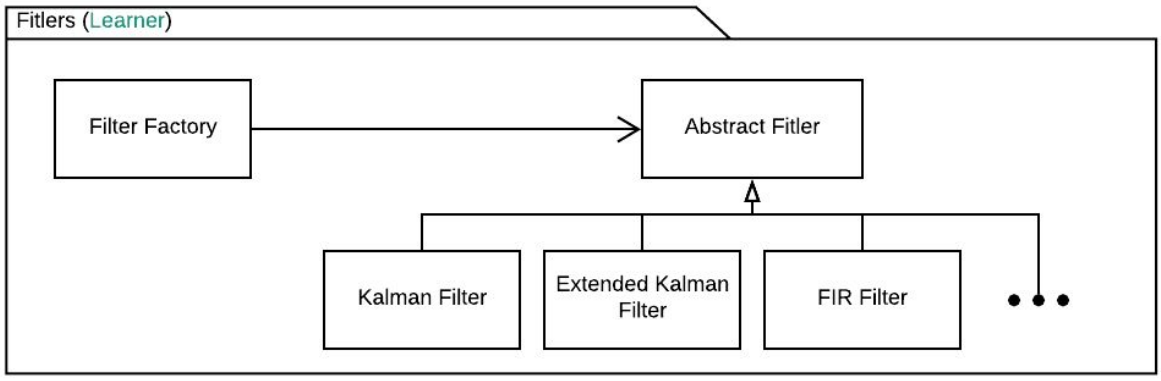

Fig. 6. Fitlers package structure design

Within the Machine Learning package shown below in Fig. 7, the team decided to make use of the factory design pattern similar to the Filters package. With many different models potentially being implemented, it's wise to set the system up in a way that can handle easy expansion without adding too much additional implementation. With this pattern, the team can achieve just that.

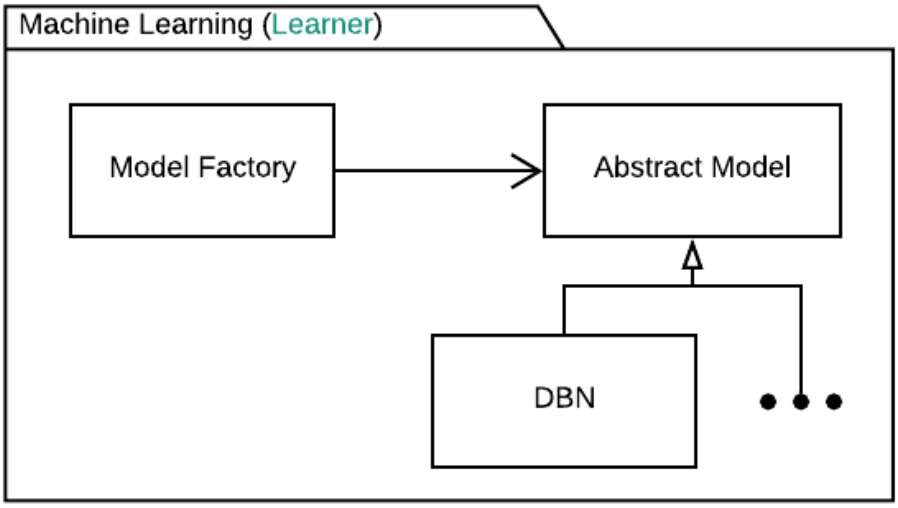

Fig. 7. The Machine Learning package structural design.

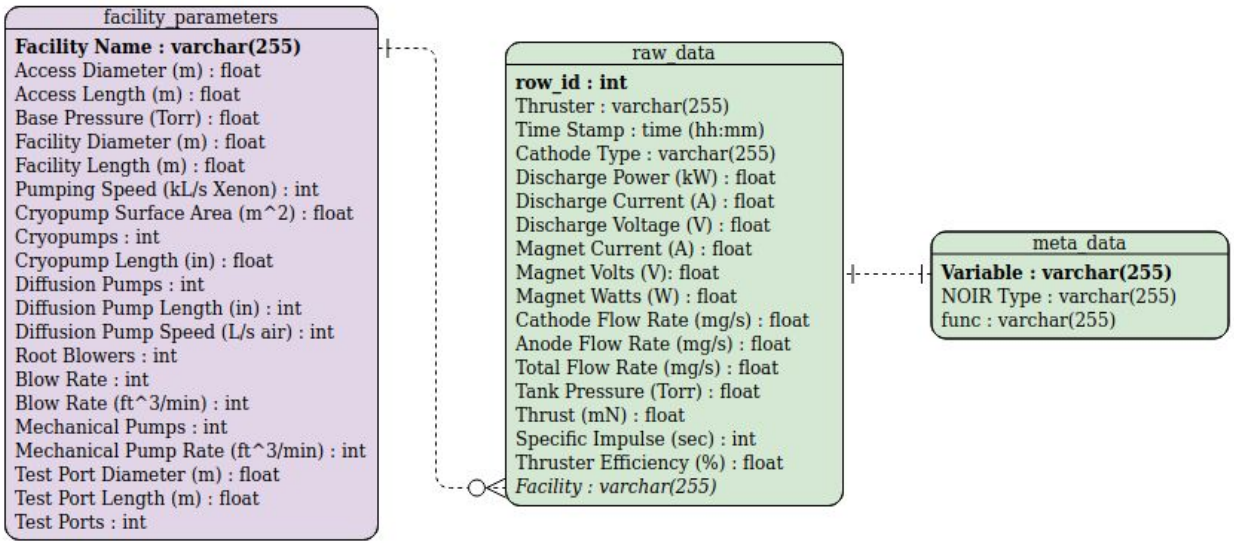

Fig. 8. The Machine Learning package structural design.

For the data model, CSV files were used to store the raw SPT data, facility parameters, and metadata found within the collected abstracts. Fig. 8 above shows the Entity-Relationship (ER) model used to relate the CSV files with each other. While the raw data and facility parameters tables, the metadata table will hold information pertaining to if the variable is nominal, ordinal, interval, or ratio data.

6.3. User Interface Design

Within this section, the overall User Interface (UI) design of the system is explored. Two interfaces are presented in this section. The first UI discussed is for the web scraper currently under development. The user interface allows for easy implementation of the WebCrawler as shown in Fig. 9 for any researchers to be able to quickly parse PDFs and sort it into a predefined directory format. The second interface discussed is a prototype for the Hall Thruster Data Analysis Tool to help the developers and researchers evaluate new data.

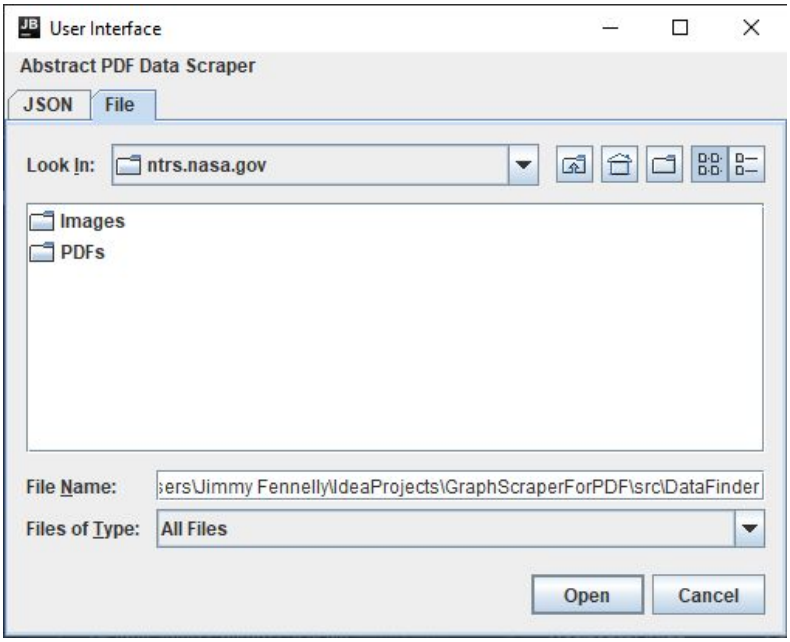

Fig. 9. The file Chooser for the PDF Scraper/Processor.

Within the user interface for the PDF Scraper and Processor, the File tab is used to easily upload local files to pass through the parsing algorithm, as shown below in Fig. 10. The user will simply go to the directory containing a PDF file and upload one to many PDF files for parsing. The program will parse each PDF and add relevant resources to their respective locations within the data collection directory.

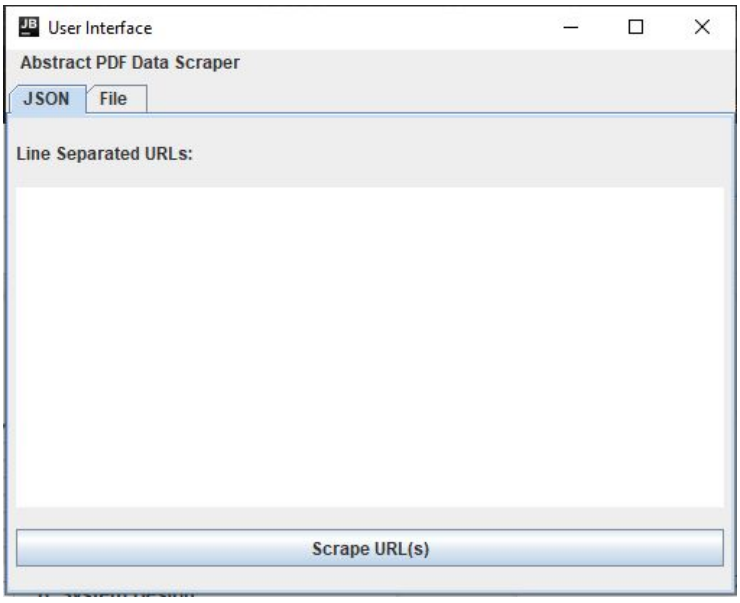

Fig. 10. The line Separated URL Interface.

The JSON tab is for the utilization of web-based URLs to index websites for PDF and JSON files as shown in Fig. 10. The user will simply copy and paste URLs, separated by newlines, to crawl each URL before sending any PDF or JSON file found through the parsing process. All results are shown within the console as of the third sprint.

As mentioned previously, ASU would like the team to write a report detailing the results of the research. Since they do not need the system themselves, only the results from the system, making a GUI is a luxury feature. For all intents and purposes, reading in necessary inputs from the console will be sufficient enough. These inputs will be for changing the various filters, models, and correlation analysis parameters and will be simple prompts. The developers are the only users of the system, so not much context will be needed to explain the input.

6.4. Behavioral Design

The following sequence diagram in Fig. 11 shows the interaction between the Developer and the Hall Thruster Data Analysis Tool. The system is controlled through running the various python scripts used to preprocess, analyze, and make use of the data records. First, the user launches the preprocessing script in order to convert the raw SPT data into a suitable format for future steps. Afterward, the system will return cleaned data, to which the developer launches the various correlation scripts that contain the logic for generating scatter plots, a correlation heatmap, Lasso Regressions, Ridge Regressions, and performing Principal Component Analysis.

The Kalman filter processing is handled in the run_filter script, while the machine learning training is handled in the run_model script. The run_filter script will accept the DataFrame produced by the preprocessing script, and generates data points by processing the frame through a Kalman filter configured to take pairs of records and generate new records. These data records are then used to train a DBN model by running the run_model script.

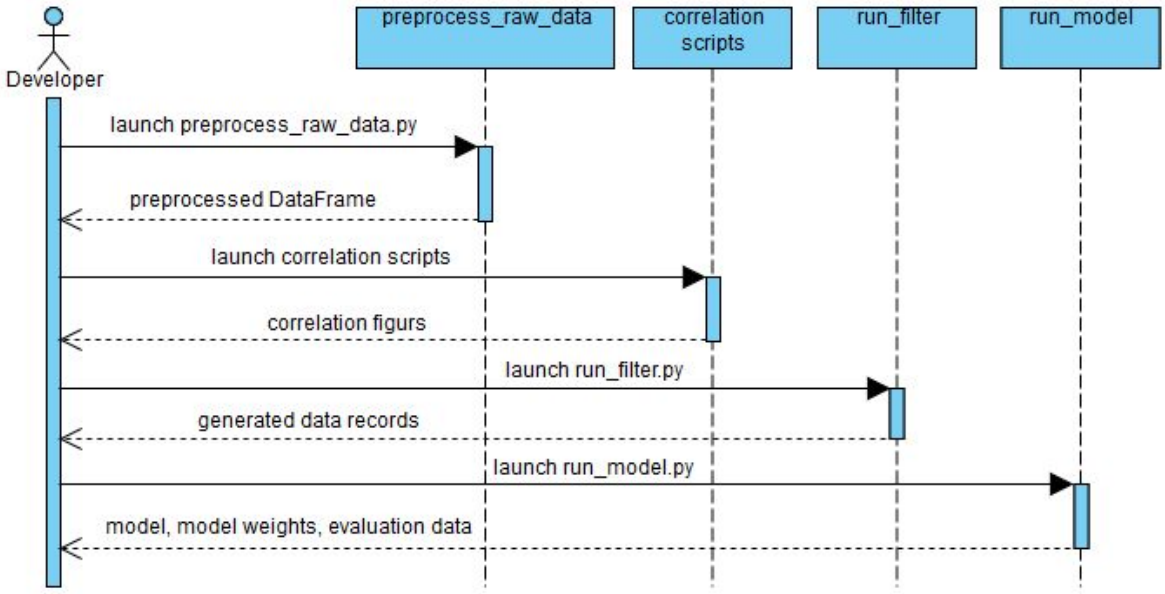

Fig. 11. Hall Thruster Data Analysis Tool Sequence Diagram

Next, Fig. 12 shows the interaction of objects when the data set is read into the system for preprocessing. Preprocessing involves cleaning the data set in order to maximize the useful information with the least amount of points. The system first uses the read_csv function to obtain the SPT data, facility parameter data, and metadata. The data_input module will then proceed to clean, transform, and reduce the data into an acceptable format. It begins by removing columns with a number of values greater than the specified threshold, 60% in this case. The data_input module will then remove nominal data columns, handle outliers and missing values within the set, then finish off by producing normalized and standardized versions of the data sets, which are saved to local storage for models and techniques to process.

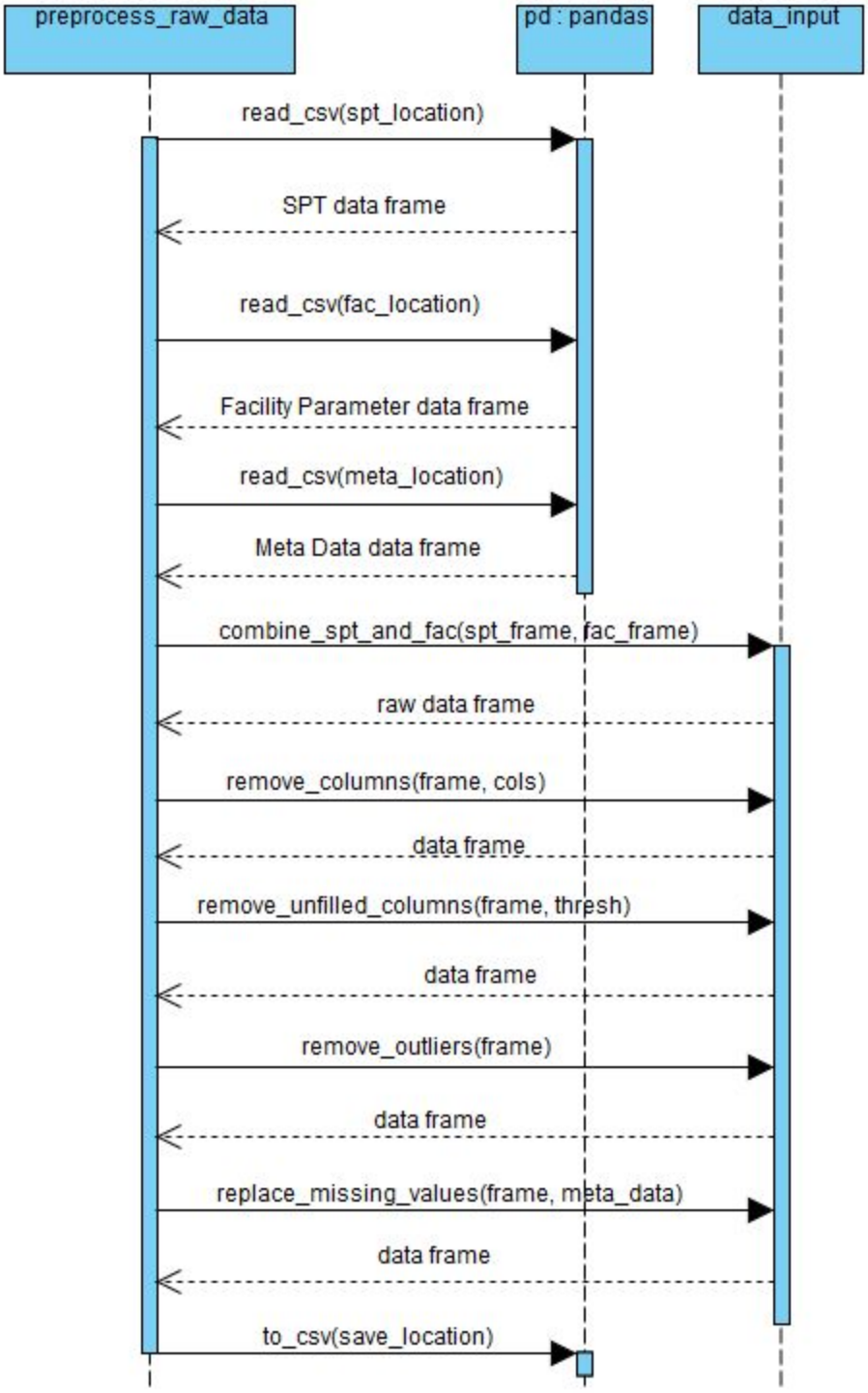

Fig. 12. Clean the Data Set Sequence Diagrams

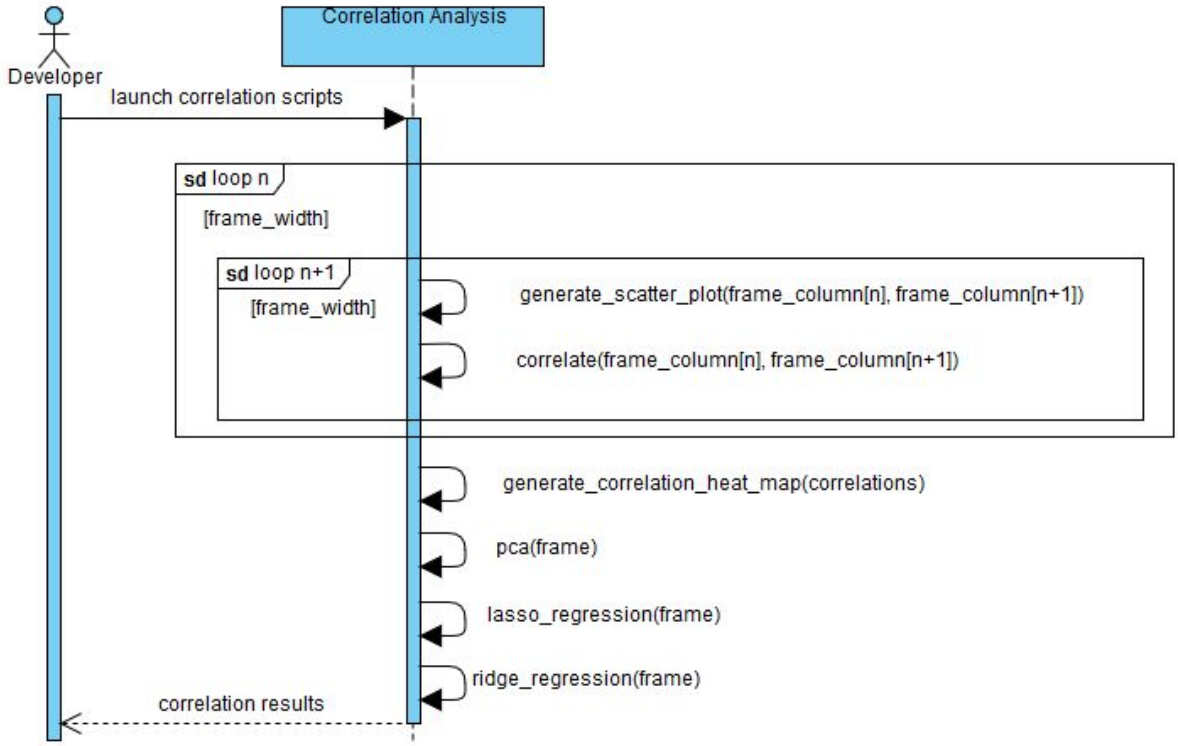

Fig. 13. Correlate the Data Set Sequence Diagram

The next sequence diagram, depicted in Fig. 13, describes the interaction of objects necessary to perform correlation analysis on the data set. When the developer launches the correlation scripts, the system will begin by calculating correlations and scatter plots of each variable against every other variable. After generating the correlations and saving the scatter plots, the system will generate a correlation heat map using the correlations. Afterward, principal component analysis (PCA) is utilized to calculate more correlations within the data set, then Lasso and Ridge regression are used to calculate even more correlations. The results of all these correlation analysis methods are then returned to the data_input module.

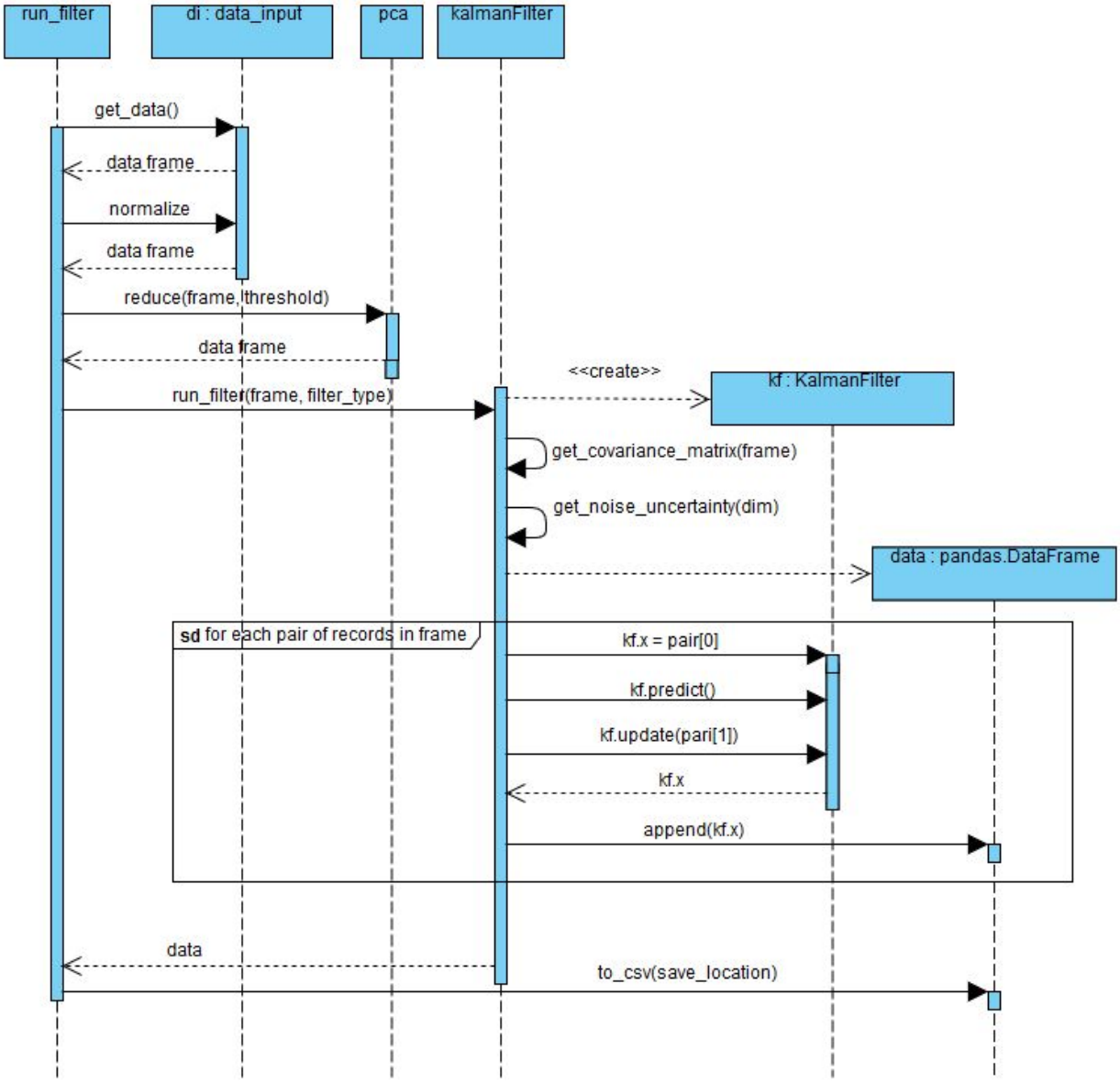

Fig. 14. Run Kalman Filter Sequence Diagram

Fig. 14 shows the sequence diagram for running a Kalman filter to evaluate data. The diagram essentially consists of a series of calculations needed to get the values to run the system. Starting, the system will first retrieve the cleaned data from preprocessing. After, the data is put through PCA reduction to tune the data as well as be normalized so it's in a proper form and the filter can process it. The data is then passed along to the Kalman filter script. The Kalman filter then creates the filter object, as well as the necessary matrices needed to operate the filter. After filling in the necessary matrices, the filter is started and is run once for each pair of points in the record. One point is put into the current state, while the other is put into the measurement to compare against, the produced point from performing the predict and update functions, kf.x, is

then appended to the newly created DataFrame. Once all pairs of points are run through, the data is returned to the run_filter scripts and saved to local storage.

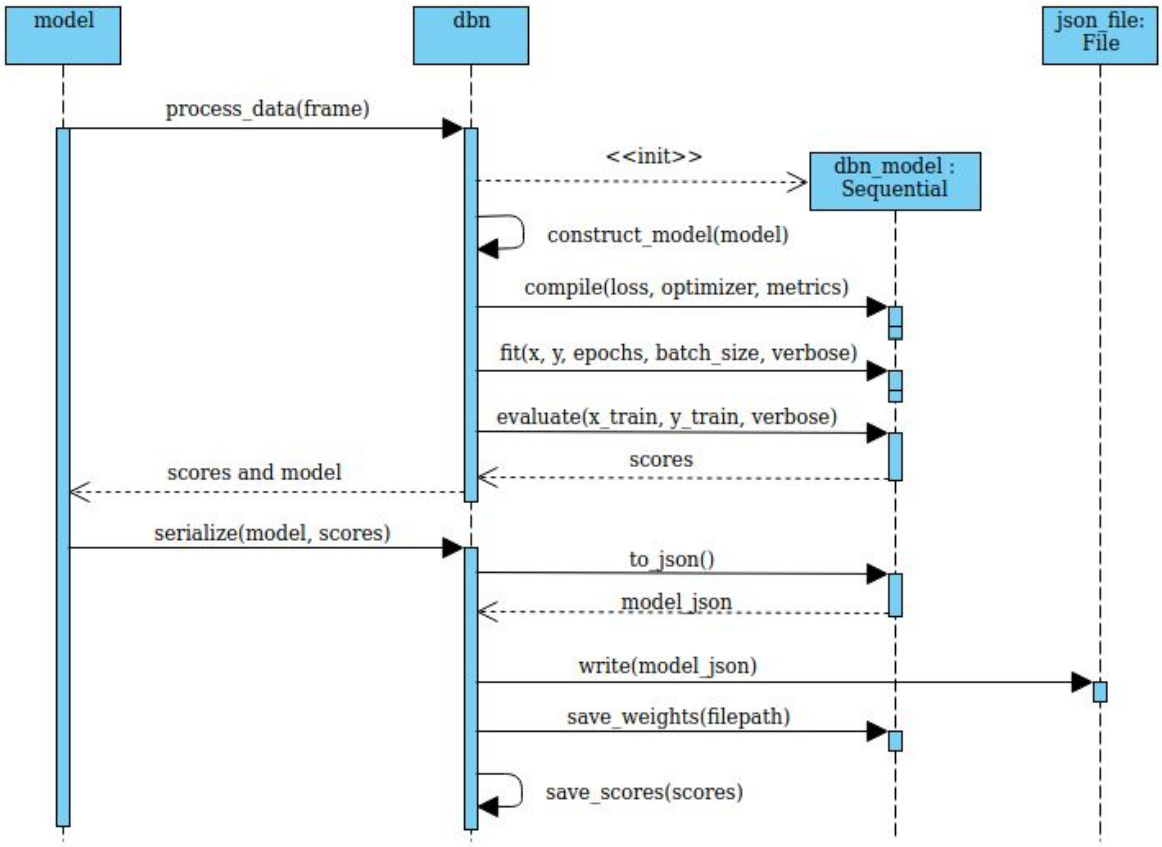

Fig. 15. Evaluate DBN Model Sequence Diagram

The final sequence diagram, Generate Report, describes the interaction of objects necessary to train a Deep-Belief Network (DBN) model using the data set, as seen in Fig. 15. The system begins by initializing a Sequential model object. More processing is required, however. The system will then construct the structure of the model, then compile the model. Afterward, the system will begin fitting the data to the model to train it, which can take quite a while with a lot of data. Following the training, the model is then evaluated to determine its accuracy and predictive capability, with any scores returning. The model and scores are then returned to be recorded into local storage. To do this, the DBN module first converts the Sequential model object to a JSON structure, then opens a File object to write the model to. The weights are saved afterward. Lastly, the scores are recorded into the local folder structure.

6.5. Design Alternatives & Decision Rationale

Another useful design pattern that could be used for the system architecture is the Repository design pattern, which allows for all of the components of the system to access the data at a single

point. This would be beneficial since all of the machine learning data, as well as the results, will be stored and accessible at a single location. However, this pattern creates a single point of failure, where if the data is incorrectly handled going to or from the repository, or if the system fails, the entire system will be unable to function. Additionally, the repository design pattern does not facilitate the scalability needed to implement multiple machine learning algorithms. As such, the Data-Model-Learner (DML) was chosen as the main system architecture since it allows for this scalability of machine learning models.

For data management, NoSQL was a possibility of being used instead of a traditional database. With a lot of data coming from different sources, there is no predictable format that the team can force the data to effectively conform to without having a large number of tables. Utilizing the envelope pattern, the data can conform to any structure while picking out relevant data into the envelope. This was eventually chosen to be omitted in favor of a spreadsheet in Comma Separated Value (CSV) format to streamline data input processing into a proper data frame for the learning system and Kalman filter. Since this project only has 3 tables for holding data, utilizing a traditional database or NoSQL database is more costly to implement as well as manage as opposed to just using CSV files. This way, the implementation time is cut while achieving the same goal.

The team chose to create a scraper to obtain more data to speed up data collection, increasing the efficiency of collecting data. As opposed to manual data collection, digging through websites for the PDFs will no longer be a time-consuming process. The system will also enforce a structured schema for data to make pre-processing of data significantly easier. Without this scraper, manual data collection would take a significant portion of the team's time and would be more prone to human error when obtaining and saving data.

In regards to the statistical analysis approach of using a Kalman Filter to generate a large sum of data for the team, there is another filter that the team looked at that can still be a viable option to explore. This filter is called the FIR Filter. The FIR Filter is almost the exact same thing as the Kalman filter, however, it does not take into account the noise in the data frame, which eliminates the noise matrix and the measured noise parameters that would normally be seen in a Kalman Filter [21]. The team decided to explore the Kalman Filter first as it would account for all of the effects within the dataset, however, an FIR Filter can still be explored as FIR Filters actually maintain a lower total error metric than the Kalman Filters.

7. System Implementation

The following section covers all information on the system implementation. The section begins by discussing the programming languages and tools utilized to develop the system, followed up by the coding conventions used. The section concludes with the version control utilized in the project.

7.1. Programming Languages & Tools

The system will utilize the Python 3.7 Programming language, the Spyder 3.6 IDE, and the Tensorflow and Keras packages. Python is used for data processing and is especially useful for implementing and training machine learning algorithms. The process of implementing these complicated algorithms can be shortened with packages like Tensorflow and Keras, which provide easy methods for creating neural networks.

To facilitate data collection, a web scraper was developed that's capable of extracting useful abstracts pertaining to HET operation and performance in different vacuum test facilities. This tool will speed up the collection of the data, allowing the team to spend more time on other parts of the project. This tool was developed on the side and in Java using the IntelliJ IDE.

7.2. Coding Conventions

The system follows the Python Enhancement Proposal Style Guide, also known as PEP-8, written by Guido van Rossum, Barry Warsaw, and Nick Coghlan [20]. The guide will use such rules as restricting the indentation of lines to improve readability. For instance, arguments are not allowed in the first line of the definition of the function, unless the arguments in each line are aligned by the opening delimiter.

7.3. Code Version Control

The system utilized Git Version Control 2.23.0 through the Spyder IDE, and the repository is being hosted on GitHub. Using the built-in Git version control will allow for easy synchronization with project development through the IDE, simplifying the process by eliminating the need for third-party version control software.

7.4. Implementation Alternatives & Decision Rationale

The following section will highlight the implementation decisions and the rationale behind why those decisions were made. The system's development environment is set up through Anaconda Navigator. Anaconda Navigator offers the ease of installing multiple Python Packages, along with creating Python Environments. An alternative would be to utilize the anaconda prompt, which requires using commands in order to manage environments and launch different software. This approach is more prone to mistakes since it is command-based. For implementation purposes and environment management, Anaconda Navigator will be used.

The Anaconda Navigator also allows the installation and launching of the Spyder 3.6 IDE on any created Python environment. Spyder 3.6 was chosen because the group members are all familiar with this IDE, and it also has a lot of nice features that are presented in a more transparent way than other IDEs, such as PyCharm. One of the nice features is the variable explorer. The variable explorer allows developers to watch every single variable that is used in their software. This

makes tracing the program much more simple for the developer. This also helps reduce the cost of debugging as users have a window that displays each variable and what they contain, whether it be values in an array or an object with a specific class type. This variable explorer is presented separately from the kernel/console window entirely and is part of the default Spyder layout. In PyCharm the variables, types, and instances are displayed in something similar to the variable explorer, however, it is located next to the kernel/console that will display the program output. This can become confusing as a developer coming from Spyder to PyCharm might not be able to effectively debug the program as they will be searching for the variable explorer. Another great feature that Spyder has is the help window and file explorer. These two windows are conjoined with the variable explorer, in the fact that the developer only has to click a tab to switch between the views. The help window allows the developer to search for documentation on imported classes without having to go to the Internet. The file explorer displays the working directory of the project and displays the files of the system. PyCharm has these features however they are not as transparent. Since the group is familiar with Spyder and contains the required testing and development packages, Spyder is a practical choice for developing this system.

As for the libraries and packages that the system's implementation uses, it was decided to use Keras, and Tensorflow in order to create, train, and evaluate the Machine Learning model [15][16]. Keras allows developers to create Neural Networks with ease as it provides a class Sequential(). The Sequential() class is an object that has a defined Neural Network structure, and the behaviors of the class allow for building a model, training the model, evaluating a model, and then deploying that model in order to make predictions on a dataset. In regards to this system, the dataset the model predicts is the SPT-140 Hall Thruster Effects dataset. The packages Pandas and Numpy were also used for data importation and normalization. The data gets read into a Pandas DataFrame. The DataFrame allows the developer to have a variable that contains labels, samples and is conveniently formatted like a Microsoft Excel spreadsheet. The DataFrame object can then have its values stripped from it, using Numpy, in order to create an array of the values for normalization. Normalization is then performed using a Tensorflow function, normalize(). The function takes in the Numpy array for the input, and performs a normalization algorithm depending on the normalization order chosen, it outputs the normalized Numpy array. With these four packages, it makes the development and implementation of the system much simpler. One could argue the use of PyTorch, SciKit-Learn, ML Kit or CUDA over Keras, however, the members of the group were familiar with the Keras package.

For the machine learning algorithm, the team decided to use a Neural Network. The Neural Network would be able to learn and discover the correlations within the dataset. There are various types of Neural Network categories that are more equipped for certain tasks. For example, RNNs (Recurrent Neural Networks) contain recursive nodes that allow for time series analysis, or CNNs (Convolutional Neural Networks) contain convolutional nodes that allow for image classification. For the purposes of the system being able to output predicted data values based on an input sample, the team decision based on the data is numerical, that a regular ANN (Artificial Neural Network) would be equipped to handle the task. Within the three different Neural Network categories are various structures. For example, within the ANN category, there are eighteen different structures that all have different advantages and disadvantages. Through

research, the team discovered that the Deep Belief Network structure would best suit this project as Deep Belief Networks are exceptional at finding correlations and patterns within a dataset and then generated a new dataset based on the input data. Also during the research endeavor, it was discovered that a similar project had been completed using a Markov Chain structure which led the team to believe that multiple structures could be used [22]. Within the development time, the team will implement multiple different Neural Network structures to be tested and evaluated by the system.

For our data generation, a Kalman Filter has been implemented in order to obtain a sufficient amount of data in order to train the machine learning model. A Kalman Filter is an algorithm that defines a Gaussian distribution between the filter state estimate and the covariance matrix [8]. The algorithm takes into account the noise in the dataset, the measured noise, the measurement function, a state transition matrix, and a control transition matrix. This allows the team to generate new data samples that are accurate to the data gathered from the abstracts. An alternative to this is to implement an extended Kalman Filter or an FIR Filter. The team is currently processing the Kalman Filter which is able to handle linear correlations. In the future development time, an extended Kalman Filter will be implemented in order to handle non-linear correlations [23]. The Kalman Filter was decided over the FIR Filter due to the fact that the Kalman Filter accounts for the noise in the dataset, which the team decided was important to account for when generating new datasets. The extended Kalman Filter will be implemented as a necessity in the future, and the FIR Filter can be explored as well.

7.5. Analysis of Key Algorithms

The following section discusses key algorithms implemented within the system. Sec 7.5.1 goes over key data preprocessing algorithms needed in order to clean, transform, and reduce the data. Sec 7.5.2, on the other hand, explains the configuration used for the Kalman filter, specifically the model configuration and the calculation for the state transition matrix.

7.5.1. Data Preprocessing

To properly store the raw data files, they are separated into three CSV files as explained in Fig. 8 shown in Sec. 6.1. These data files are separated into the SPT Hall effect thruster data, and the vacuum facility data. The link between the two data files link, as shown in Fig. 8, is the facility name used to collect that record. The following pseudocode and analysis explains the combination process.

1) spt_columns = spt_data_frame.columns
2) facility_columns = facility_data_frame.columns
3) new_columns = spt_columns + facility_columns
4) spt_values = spt_data_frame.values
5) facility_values = facility_data_frame.values
6) new_values = *empty*
7) **for** *row* **in** *spt_values:*
    a) to_add = *empty*

    b) **for** *record* **in** *facility_values:*
        i) **if** *record.facility_name* **equals** *row.facility_name:*
            (1) to_add = record
            (2) break
    c) new_values.add(concatenate(row, to_add))
8) combined_data = data_frame(new_values, new_columns)

The pseudocode assumes that the variables containing the SPT data and facility data are already loaded into the environment. The algorithms begin by extracting the column names from the SPT, and facility datasets, then combining them. The same thing is done for the values, except the new matrix holding the new values is left empty for now.

The goal at this point is to take the facility name located in the spt_values matrix, and find the corresponding facility record located in the facility_values matrix. Since the correct facility name must be found manually, this requires a nested loop, meaning this algorithm has a worst case of O(n*m), where n is the number of SPT records, and m is the number of vacuum facilities. The to_add array is initially left blank. This is to ensure a default value is available in the case that the facility record is not present within the facility_values. The inner loop will check every facility record, and if the record is found, the algorithm will mark this record and break out of the inner loop.

The outer loop will then concatenate the row from the spt_values with the record from the facility_values to create one long row associating the SPT parameters as their affiliated facility. Once every row in the spt_values matrix has a corresponding facility record appended to it, the resulting matrix is combined with the columns derived earlier to form the combined dataset. Since the complexity of this algorithm does not stem deeper than two loops, the time-complexity of the algorithm is O(n*m).

7.5.2. Implementation of the Kalman Filter

In order to make the Hall Thruster and Data Analysis tool viable, an implementation of a Kalman Filter was the proposed solution. The Kalman Filter is originally a tracking algorithm, however the team created a modified Kalman Filter that is used for data generation. In order for the Kalman Filter to generate accurate data, the team implemented a pairwise approach. This approach continuously sets the state and measurements based off of the pairs. A single point can be taken and this single point represents a generated sample estimate of the data. The following pseudocode will explain the process of implementing the Extended Kalman filter to generate estimated data records:

1) **function** get_state_transition_matrix(dataframe):
    a) correlations = dataframe.correlations
    b) **for** *row* **in** *correlations*:
        i) sum = summation(row)
        ii) **for** *value* **in** *row*:
            (1) value = value / sum

```
        c) return correlations
2) values = dataset.values
3) x = values[0]
4) z = values[1]
5) dimension_x = x.length
6) dimension_z = z.length
7) kalman_filter = ExtendedKalmanFilter(dimension_x, dimension_z)
8) HJac = lambda x, hx_args=None : H
9) Hx = lambda x, hx_args=None : x
10) kalman_filter.x = x
11) kalman_filter.P = dataset.covariance_matrix
12) kalman_filter.Q = random_noise(dimension_x)
13) H = identity_matrix(dimension_z)
14) kalman_filter.F = normalize(get_state_transition_matrix(values))
15) R = random_noise(dimension_z)
16) generated_data = empty
17) for (i=0; i < values.length; i++):
    a) for (j=i+1, j < values.length; j++):
        i)   kalman_filter.x = values[i]
        ii)  z = values[j]
        iii) kalman_filter.predict()
        iv)  kalman_filter.update(z, Hjac, Hx, R)
        v)   generated_data.append(kalman_filter.x)
```

The above pseudocode begins by extracting the values that the Extended Kalman filter will utilize from the dataset. Afterward, the initial x & z state values are pulled from the values matrix in order to perform the necessary calculations to form the filter, as seen when calculating dimension_x and dimension_z in the following steps. After this, the Extended Kalman filter will be initialized. Within the project, the filter was initialized using the ExtendedKalmanFilter class found within the sklearn library.

Now that the filter is initialized, the algorithm then begins populating the necessary values: HJac, Hx, x, P, Q, H, F, and R. Each of these matrices is configured and can be changed to best match the experiment, but this pseudocode only utilizes default values and random noise, with the only exception being in step 14 where the function “get_state_transition_matrix” is called. Within this function, a matrix of size dimension_x by dimension_x must be returned.

To simulate this matrix, the correlation matrix was extracted using the numpy library. This matrix is size dimension_x by dimension_x and each cell within this matrix is the correlation between the two variables. For each row in this matrix, the sum of the row will be obtained, which will be all of the correlations summed together for that row. For each value in that row then, the value will be divided by the sum of the row. The end result will have each value in the

row be a percentage of the total correlation of that row, thus emulating the state transition matrix through percentages based on the correlation matrix.

After filling in the missing matrices, the filter can now be run. As mentioned before, a pairwise approach is used where the Extended Kalman filter is run once for every pair of records within the values matrix. This causes the time-complexity of the algorithm to already be at O(n^2). Within the nested loops, the x & z variables are assigned, the filter's predict function is called, then the update function is called. After that, the data point is generated in the filter, so it's extracted and appended to the genrated_data frame. To run the Extended Kalman filter once has a time-complexity of O(n^3), so running the algorithm has a total time-complexity of O(n^5) [24].

8. Testing

The following section goes over the test suites and cases used to validate and verify the software under development. The section begins with a look at the test automation framework being utilized in Sec. 8.1, then continues with the design of the test cases and their execution reports in Secs. 8.2 and 8.3, respectively.

8.1. Test Automation Framework

The testing framework used to automate unit tests is the spyder-unittest framework. This framework can also be used to automate integration tests. Additionally, spyder-unittest allows for multiple test cases to be executed from a built-in panel added to the Spyder IDE [25].

8.1.1. Steps for Installing Test Framework

The spyder-unittest package must first be installed onto the anaconda package directory using the conda command before adding it to an environment. First, open the terminal if on Linux or macOS, or the command prompt if on a Windows machine. Then, run the following line in the terminal to install spyder-unittest onto Anaconda Navigator [25]:

- Using Anaconda: conda install -c spyder-ide spyder-unittest
- Using pip: pip install spyder-unittest

The spyder-unittest testing framework can then be quickly and easily installed via the Anaconda Navigator program used to keep environments uniform across all working machines. On the home page, select the “Environments” tab on the left panel to bring up the environments currently available on the system. After selecting the working environment, change the filter box to the left of the “Change” button to say “All” instead of “Installed.” The “Installed” option only shows the packages already installed in the environment, while the “All” option will show all available packages.

Afterward, search for the “spyder-unittest” selection within the list of packages, then simply click the checkbox to add the package to the environment. If spyder-unittest does not appear, check to make sure it was installed onto the system using the above commands, then try installing through Anaconda Navigator again.

8.1.2. Steps for Running Test Cases

To create a test file for running most test cases, first ensure that the spyder-unittest package is installed. If the package is not installed, refer to Sec. 8.1.1. for instructions on how to install the spyder-unittest package. If the package is installed, launch Spyder. Then, click the View menu and expand the Panes option. At the very bottom, there should be a checkbox labeled Unit

Testing. Clicking this box will provide a new pane on the right-hand side of the IDE called “Unit testing.”

Clicking on this new pane will bring up a small window that will display all of the current test cases. If the box is empty, simply hit the green Run Tests button, select unittest as the test framework, then select the folder containing the test files.

To create a test file for running test cases, certain standards must be met. The first is that the name of the test file must be named “test_[FILE NAME].py,” where [FILE NAME] is the name of the python script being tested. Next, the script must create a class with the same file name and must inherit the unittest.TestCase class to be recognized as a test case by the spyder-unittest package.

Lastly, each procedure being tested must be named similar to the file, starting with “test_” and ending with the name of the procedure being tested. Each test procedure function acts as a separate test case since each procedure must be tested prior to being put together. From there, it’s just a matter of inserting preconditions and postconditions utilizing the self.assert procedures provided by the unittest.TestCase class. After creating the test file, hit the green Run Tests button to run the test case.

Some test cases needed to be tested manually and could not be tested using the spyder-unittest testing framework. These were primarily the system and acceptance tests, where checking the output required running scripts that did not contain functions, and therefore could not be tested by using spyder-unittest.

In one instance, it was more beneficial to create a custom testing script for the sole purpose of performing acceptance testing on a non-functional system requirement instead of using the spyder-unittest testing framework. The requirement, SP-02-01, requires the data records be from either SPT-140 or SPT-100 HETs. This test is represented in test case 34 of Table 8.2.40. This custom test script iterated through the raw data set, and failed if any data record had something other than ‘SPT-140’ or ‘SPT-100’ as the thruster type. The team decided that writing additional code to check for this instance may not be beneficial for future research by other developers.

8.2. Test Case Design

This next section goes over the design of each test case given in appendix T. For starters, each test case has a corresponding collection of similar test cases, known as a test suite. These test suites are described in detail in Sec. 8.2.1. A test case can be categorized as unit testing, integration testing, system testing, or acceptance testing. Each test case will also include other information, such as what is being tested, and the corresponding requirements and use cases that the test case covers.

Although needed for this report version, no integration testing has been performed as of this moment. If found necessary, it will be included in the report version 2.5.

8.2.1 Test Suites

For the test suites, there are six suites the test cases are organized into. The first has to do with the data input and preprocessing needed to provide data for the machine learning model, simply called the Data Cleaning test suite, which is shown in Table 8.2.1. This test suite is covered in test cases TC-001, TC-002, TC-009, TC-010, TC-011, TC-012, TC-013, and TC-017.

For the second test suite, called Deep Belief Network (DBN), the test cases are shown in table 8.2.2 and cover everything needed to test the functionality of training machine learning models. In total, there are seven test cases that cover the different functionalities required in order to form, train, and evaluate a machine learning model. An additional test case was added, TC-039, to test the functionality of the model_evaluator script. This test case is not included in the integration test case which confirms the model can be trained.

For the third test suite, called Correlation Analysis, the test cases are shown in Table 8.2.3 and cover all of the functional unit testing for the single variate analysis code. In total, there are six test cases that cover the different functionalities required to analyze the single variate correlations between data points obtained from the data collection process. These include procedures for Principal Component Analysis, Pearson correlation analysis, Lasso Regression, and Ridge Regression. All test cases and execution reports follow the spyder-unittest testing instructions laid out in Sec 8.1.2. As many of the test cases act independently of one another, an integration test case was not necessary.

For the fourth test suite, called Kalman Filter, the test cases are shown in Table 8.2.4 and cover all of the functional unit testing for the Kalman filter code. There are a total of four unit test cases and one integration test case. Each unit test case tests the functionality of each function within the Kalman filter, and the integration test ties the entire module together to test the final output.

The fifth test suite covers the main execution that covers the system testing and acceptance testing of the functional requirements of the system. This suite can be seen in Table 8.2.5. There are a total of eight test cases in the suite, with four being system testing and the other four being acceptance testing. The sixth and final test suite covers the acceptance testing for the non-functional requirements. Since there were very few, only five test cases were needed to cover every non-functional requirement. This test suite is located in Table 8.2.6.

8.2.2. Unit Test Cases

Starting with the seven test cases in the first test suite, Data Cleaning, TC-001 shown in Table 8.2.7 tests for the incoming data to be in CSV format, and the resulting data parsed into the system is in a matrix object, has more than 0 rows and columns, and the resulting matrix contains the same data from the file. The test input used was a dummy matrix containing values ranging from negative infinity to infinity.

The second test case for the first test suite, TC-002, is shown in Table 8.2.8. This test case covers the normalization necessary to put the data into an acceptable format for machine learning processing. The test input for the data is the output from TC-001, which is a DataFrame object containing the dummy input data. The expected output is a matrix in a DataFrame object, with all of the values being a positive, non-zero value. The third through seventh test cases (TC-009, TC-010, TC-011, TC-012, TC-013, respectively) are functional tests to transform the dataset when required to effectively clean and remove unnecessary data points. The inputs of each test case include the data frame, and the outputs include the transformed data frame. The third through seventh test cases are located in Table 8.2.15, Table 8.2.16, Table 8.2.17, Table 8.2.18, and Table 8.2.19, respectively.

The next six test cases are all covered under the second test suite, and cover the functionality of building and training a neural network. Table 8.2.9 covers test case TC-003, which tests whether or not a Deep-Belief Network (DBN) machine learning model can be constructed. After construction, this model is fed into the next test case in Table 8.2.10, where the model is then compiled using the Keras library. If the test passes, the model will have the structure of a DBN. Following compilation, the model is then the input for the next test case, TC-005, represented in Table 8.2.11. The test input, in this case, is uploaded, normalized data, as well as the model that was recently compiled. The system then attempts to train the model by fitting the model to the data to obtain values for different weights. The test case passes when the model is successfully trained. Since any accuracy is required for completing the test case, no threshold is provided.

The next two test cases in this suite, TC-006 & TC-007, deal with evaluating the accuracy of the model and finally serializing the model for later deployment. TC-006 is described in Table 8.2.12 and takes the trained model from TC-005 as test input. If the model is successfully evaluated, then accuracy in the range of 0 to 1 will be returned. Table 8.2.13 covers TC-007. In this test case, it again takes the trained model from TC-005, but now serializes the model into a JSON format for storage. The expected output for this case is a JSON file containing the information for the model. The final unit test case is TC-039, testing the model evaluator to see if a model can be evaluated against a data set. This test case is located in Table 8.2.28.

The Correlation Analysis test suite contains six unit test cases. The first test case in this suite, TC-008, tests the reduce function in the pca file. This function’s sole purpose is to take a data set and perform PCA reduction on it based on the threshold provided. The returned result is a data set with fewer columns, called a reduced data set. The details of this case can be seen in Table 8.2.14. The next text case is TC-014 in Table 8.2.20. This test determines if the inverse_reduce function works. This function's purpose is to perform the inverse operation on the reduced data set to produce the original set.

The next two test cases, TC-015 and TC-022, deal with the use of Pearson correlation analysis to generate scatter plots of the data set and a correlation heatmap of the data set. TC-015 is in Table 8.2.21 and the other in Table 8.2.25. The test will check the number of graphs in the directory to ensure the correct numbers were generated. Unfortunately, a manual check of the graphs is

necessary to ensure the file fits the correct format, has the right labels, and looks appropriate. The same standards apply to generating the heatmap.

The fourth test suite includes three unit test cases for the Kalman filter. These cases--TC-019, TC-020, and TC-021--all contain the same input of a data frame containing more than 0 rows and columns and no missing values. TC-019 can be shown in Table 8.2.22, with an output of a data frame of size 1 by the number of original columns containing the average of each column in the original data frame. TC-020 can be shown in Table 8.2.23, with an output of a symmetrical matrix where each element is the covariance of one column against another. TC-021 can be shown in Table 8.2.24, with an output of a square matrix of the size of the number of columns where all values are random noise from a gaussian distribution.

8.2.3. Integration Test Cases

In the first Test Suite TS-001, shown in Table 8.2.1, there is one integration test case TC-017. The integration test case TC-017 combines all of the unit test cases listed TC-001~TC-013 in order to convert the manually gathered data from the scientific abstracts provided by NASA and the team's explorations to DataFrames, an object from the Pandas Python package. This test case can be found in Table 8.2.29.

In the second Test Suite TS-002, shown in Table 8.2.2,there is a second integration test case TC-018, which can be seen in Table 8.2.30. The integration test case combines all of the unit test cases, TC-003-TC-007, TC-018, in order to test that the DBN neural network can accept the DataFrame generated from the previous integration test case TC-017. This test case also tested that the DataFrame can be accepted and used to construct, train, and serialize a machine learning model.

The final integration test case, TC-025, is located in Test Suite TS-004, shown in Table 8.2.4. The test case is in Table 8.2.31. Integration test case TC-025, combines all of the unit test cases, TC-019-TC-021. The combination of the unit test cases test that the Kalman filter can accept a DataFrame object, and output newly generated data in another DataFrame object.

8.2.4. System Test Cases

Only one suite has system test cases, TS-005, which is the test suite housing all of the test scripts for evaluating the main execution of the system’s main functionalities. The system test cases ensure that the correlation analysis , preprocessing, Kalman filter, and DBN scripts all function and fulfill the system needs.

TC-026 is covered in Table 8.2.32 and deals with ensuring the correlation analysis scripts all generate their respective plots. TC-027 ensures that the preprocessing script can successfully generate a proper data set. This test case is in Table 8.2.33. The next test case, TC-028, covers the script which controls the training for the Deep Belief Network model. This test case will ensure that the different parameters can all generate a model which can then be evaluated against data. It’s located in Table 8.2.34. The final system test case, TC-029, deals with ensuring the

Kalman filter script can generate points, then save those points to the file directory for training models. Table 8.2.35 has this test case.

8.2.5. Acceptance Test Cases

There are a total of nine acceptance test cases, four of which are in test suite TS-005, while the remaining are in TS-006. The first four cover the acceptance testing of the first four scripts, this ensures that the created functionality matches the requirements. The remaining four go over the acceptance testing for the non-functional requirements.

The first four cases follow the same pattern as the four system test cases. There is one for each set of script executions. TC-035 covers the preprocessing script in Table 8.2.41, the correlation results in Table 8.2.42, the Kalman filter execution in Table 8.2.43, and finally the DBN script in Table 8.2.44.

The last five test cases in suite TS-006 all have to do with non-functional requirements. The first case, TC-030, tests the citizenship of each developer and is described in Table 8.2.36. Table 8.2.37, Table 8.2.38 and Table 8.2.39 all test to see if the system can launch on Windows, Linux, and macOS, respectively. The last acceptance test case is in Table 8.2.40 This case checks the 'ModelType' column inside the data set and tests if each record is either 'SPT-140' or 'SPT-100'.

8.3. Test Case Execution Report

The following subsection discusses the execution reports of each of the test cases created thus far, as given in appendix TE. Each test case has a matching test execution report, which records the results of running the test case.

8.3.1. Unit Testing Report

Each execution has a similar execution setup, and will only be described once since the main difference between each execution is the folder containing the correct test scripts. To run a test case using a spyder-unit test, the user must navigate to the unit testing panel found on the right-hand side of the Spyder IDE. If the panel isn't present, select the View tab on the menu, then select "Panes," then at the bottom check the box that states "Unit Testing." This will activate the unit testing pane. On the pane, select the gear in the top-right corner of the pane, then select "Configure." On this pop-up window, select the unit test for the testing framework, then select the folder containing the test scripts. Afterward, press the "Run Tests" button above the pane to execute all available test cases.

Some execution reports include different instructions, since some non-functional requirements either did not require the use of a test script, or they were manually tested as the spyter-unittest framework could not cover it.

The first two execution reports deal with the module to upload data into the system, then the data normalization to format the data. For the first execution report as seen in Table 8.3.1 of Appendix TE, the first execution resulted in failure since the resulting matrix needed did not yet exist. After implementing the required parts of the function to accept data, the test passed. The second report, detailed in Table 8.3.2, had a similar problem as in the previous report but had an added problem. The Information Normalization System shown in Fig. 5 contains the DataInputSystem class and DataNormalization class. The DataInputSystem class was needed in order to make the DataNormalization class perform correctly. An error occurred where the file path for the DataInputSystem.py file could not be found. After correcting the file locations and implementing the code, the test passed. The next five execution reports deal with the data cleaning process, effectively transforming the dataset to be useful for further analysis. Table 8.3.9 through Table 8.3.13 displays the test execution report for TC-009, TC-010, TC-010, TC-011, TC-012, and TC-013, respectively. For each execution report, the tests failed once during the initialization of the test cases but worked on the first implementation afterward. Multiple more records were run to ensure that the test cases still passed after numerous additions.

The next set of execution reports in the second suite covered the functionality of the building and training a machine learning model. For the reports in Table 8.3.3 through Table 8.3.7, running the test cases using the spyder-unittest package caused Spyder to crash unexpectedly. The solution to this was to remove the class structure used to organize the model building, training, and evaluation. After this, TC-004 in Table 8.3.4 passed and successfully constructed the model. The execution reports in Table 8.3.4, Table 8.3.5, Table 8.3.6, and Table 8.3.7 all had another issue involving the use of the Keras and TensorFlow modules inside class structures. Removing the class structure solved the issue. The last report is for TC-039 and goes over the model evaluation, located in Table 8.3.22.

The next set of execution reports in the third suite covered the functionality of the correlation analysis. Table 8.3.8 shows the execution report for the reduce function in the PCA script. A few errors occurred in trying to get the appropriate index, but the data was reduced after a few attempts. For the reports in Table 8.3.14 show the report for testing the inversion operation for PCA reduction. Table 8.3.15 shows the report execution of generating graphs from the PCA components. The execution report for generating the scatter plots and the correlation heatmap are in Table 8.3.19 and Table 8.3.20, respectively. Finally ,the Ridge and Lasso correlation graphs are covered in the next report located in Table 8.3.21.

Finally, the last set of execution reports contains three test cases, TC-019 through TC-021, mapped to Table 8.3.16 through Table 8.3.18. The first test execution initially failed due to incorrect mapping of values from string to float but successfully constructed the state estimate after correction. The next test execution initially failed due to a misalignment in the size of the covariance matrix. After re-evaluating the mathematics for constructing the correct covariance matrix, the test cases successfully passed. Moving to the generation of the noise uncertainty matrix generation in TC-021, the execution initially failed due to not being initialized, before being correctly instantiated.

8.3.2. Integration Testing Report

For the integration test execution reports, there are 3. TC-017's report is described in 8.3.23, while TC-018's report is described in 8.3.24, and TC-025's report is described in 8.3.25. For TC-017, a logical error occurred where the normalized and standardized data sets were successfully calculated and within range, but did not come out to be the expected values. The error was discovered to be an improper calculation, and was corrected thereafter. After that, the data_input script can successfully format data to a standardized and normalized format.

For TC-018, the DBN functionality was integrated together to successfully train and evaluate the predictiveness of the trained model. The test case was successful after only two runs. For TC-025, it tested the integration of the Kalman filter. Since thorough research and planning went into the implementation, only two records were needed for the first pass of the test case. With this, the Kalman filter is now able to accept a data frame containing SPT data and extract predicted states as well as data points that can be used for further analysis.

8.3.3. System Testing Report

All four system testing reports are under the same test suite, as all system test cases cover the basic execution of the main functionality of the system. Each of these cases are manually tested since the generated graphs need to be generated manually and the only thing that can be automated is the number of graphs generated. For Table 8.3.26 which has TC-026, the execution report requires the tester to open the scripts, then run them to generate the plots. A total of three plots are expected, and were analyzed to ensure they have proper labels and a title.

The report for TC-027 is found in Table 8.3.27 and evaluates the preprocessing script used to clean, transform, and reduce the data. Furthermore, the reports for running the model and the Kalman filter are found in Table 8.3.28 and Table 8.3.29, respectively. Since these functionality has gone through diligent unit testing, little testing was needed to ensure accurate results for these reports.

8.3.4. Acceptance Testing Report

There are a total of nine acceptance tests to confirm the functional and nonfunctional requirements of the system. The first four are in test suite TS-005 and deal with the functional requirements, while the latter five deal with the non-functional requirements. Starting with TC-035 in Table 8.3.35, the report tests to see if the requirement of preprocessing the data in the required format is achieved. Similar tests were performed for the correlation analysis scripts, the Kalman filter, and the DBN model script found in Table 8.3.26, Table 8.3.27, and Table 8.3.28, respectively. These acceptances are similar to the system tests of the same vein, except they test if the functionality achieves the requirements. This testing is "black-box" testing, where the testers are unaware of how the inner-workings of the product function, and instead focus on the higher-level concepts. The main difference between the system testing reports and acceptance testing reports is that the system testing reports cover the end-to-end functionality of the system,

while the acceptance testing report instead ensures the system conforms to the user and system requirements.

The latter five reports as mentioned before all test the non-functional requirements, and can be found Table 8.3.30 through Table 8.3.34. The first report, which covers TC-030, located in Table 8.3.30, checks the citizenship status of each developer. Since all the developers don't have a Chinese citizenship, the test passes. The next three reports all test the environment on Windows, Linux, and macOS, respectively. The final report tests if the data records are all from SPT-140 or SPT-100 thrusters.

9. Challenges & Open Issues

This section will present the team's challenges and issues faced in the requirements engineering process with some insight on the availability of the industry mentor and understanding the problem domain.

9.1. Challenges Faced in Requirements Engineering

Overall, the sponsor was not very descriptive with their problem at first, and it was difficult to determine what the output of the program should be. It seems as if our sponsor is a middle man to the technical people which makes it a little difficult as the person we meet with bi-weekly does not have all of the requirements, specifications, or technical knowledge of the problem we are working on and has to ask the technical people for us to get our desired answers.

In the future, this problem could be eliminated by asking them to be as specific as they can be with what they want.

9.1.1. Availability of Industry Mentor

One of the biggest problems faced in development was maintaining proper contact with the Industry Mentor, Arizona State University's Cassie Bowman. Taking into account her schedule and the developer's schedule, there was only 1 small block of time on Tuesdays when each party could attend a meeting. Additionally, while Cassie is in charge of managing the project to keep it on track, the developers were instructed to contact other members of the Psyche team at ASU with more technical questions, meaning to completely elicit all requirements, the development team needs to adhere to the schedules of several other people.

The meetings are set up to be bi-weekly, with informative optional meetings providing background context once or twice a month. These optional meetings also caused issues, being difficult to attend due to tight time constraints. No member has missed a meeting to date. Additionally, since Arizona was not affected by daylight savings time, both the meeting with the industry sponsor, as well as the faculty advisor, needed to be rescheduled.

9.1.2. Understanding the Problem Domain

The problem domain primarily involves knowledge of physics at the atomic level. The Hall thrusters utilize charged particles to propel the spacecraft. Since none of the developers have a physics background, several hours of studying will be required throughout the course of the project to build a solid foundation of understanding. The material is complicated and bountiful as well, with over 20 papers published on HET facility effects on Hall thrusters. If an understanding is not built properly, then the team will be unable to devise a technique to predict or correct the

sensitivity to build this complicated understanding, the team will need to spend most of the initial development time conducting research and gathering information on the type of data to analyze.

9.1.3. Correctly Setting the Project Boundary

One recurring challenge the team has faced is defining the project boundary well enough to limit the amount of feature creep from unnecessary components in the system. Previously, the team considered implementing a report generator to speed up the data analysis portion of the project. However, this was overstepping the boundary of what was needed since the cost to implement this feature was high and contributed little to nothing to the system.

ASU does not require a report generator or some fancy system for processing data, only a final report detailing the results of our analysis and any other information we deem important. As such, the project scope at the beginning encapsulated too many ‘nice-to-have’ features and did not focus on the entire goal of the project: research and analysis.

Through constantly refining our goals, documents, and understanding of the domain, the team has overcome this issue through diligent work. The system today is more well-defined and is more oriented on analyzing the SPT data set for any useful information rather than attempting to streamline the process through automation. ASU does not require this automation, so any feature that does so is feature creep.

9.2. Challenges Faced in System Development

Some issues arose during the development of the system. These issues occurred during the creation of the data input system for accepting the facility effects data, as well as the prototyping for creating the neural networks. The first issue involving the data input system had to do with a limitation with Python 3.7. The first issue involved the files being unable to import classes from other files. The solution to this was to change the default working directory to the local repository’s folder path.

The next problem had to do with a limitation to the Keras library for implementing different machine learning algorithms. A class structure was used in the Python scripts to organize the files associated with developing the system. Due to the class structure, Keras’ functions could not be called to initialize or train the model. The solution here is to either figure out how to correctly inherit the necessary procedures to ensure the model can be initialized in the class structure, or simply remove the machine learning from the class structure approach to simplify the code. The latter approach will be implemented for ease of use.

The next challenge the team faced was involving the implementation of the Kalman Filter. In order to have a fully functional and accurate Kalman Filter, there are many parameters that must be accounted for and accounted for accurately. Some of the parameters that were causing issues were the noise matrix, the state transition matrix, the measurement function, and the control

transition matrix. This is where the team had to really come together in order to have a discussion on what these parameters were, how they affect the Kalman Filter mathematically, and how the parameters are acquired. After gathering a solid concrete understanding of the Kalman Filter and the associated parameters, it was relatively simple to implement into the system itself.

9.3. Open Issues & Ideas for Solutions

As it stands, there are two primary issues with the current Hall Thruster Data Analysis Tool. The first problem is the lack of a scalable solution for implementing more algorithms for analysis for both Kalman filters and machine learning. This occurred because of the lack of time needed to fully explore each option. The team instead performed a detailed analysis of one approach taken. In the future, more developers could work on scalable solutions to implement more types of filters and machine learning algorithms.

The machine learning algorithm implemented, the Deep Belief Network (DBN), was also not optimized to attain the highest accuracy possible. More can be done to optimize the performance and accuracy. Another such improvement would be to collect more data. For more information on the experiment and findings, refer to Appendix RA, the results analysis.

10. System Manuals

This section highlights the Instructions on how the system will be developed and deployed. Instructions for setting up the development environment are provided in Sec. 8.1.

10.1. Instructions for System Development

The following subsection details the needed steps for setting up the development environment and any further extensions needed to develop the system.

10.1.1. How to set up development environment

There are four main steps needed to set up the development section. The first is to install the language being used, Python 3.7, and the corresponding package manager, Anaconda. After setting up those tools, the steps proceed for setting up the programming environment, installing the packages needed for development, then concluding with the installation of Spyder 3.

10.1.1.1. Installing Python 3.7 and Anaconda

Download and Install the Anaconda Navigator from Anaconda Python Distribution using the instructions from the following URL:

https://www.anaconda.com/distribution/#download-section

To run the development environment, Python 3.7 and any dependents must be installed to run the system's main feature: machine learning algorithms. This is because Python contains several useful packages, mentioned in Sec. 8.1.1.3, that will be used for easy implementation of data analysis techniques and machine learning algorithms. Fig. 17 below shows the download page for Python 3.7. Python 3.7 and the Anaconda Navigator can be downloaded at the following link:

Fig. 17. Download page for Python 3.7

After the Anaconda Navigator and Python 3.7 have been installed, it is time to set up the Python Programming Environment

10.1.1.2. Setting up the Programming Environment

First, open the anaconda prompt to begin initializing the Nasa Environment as shown below in Fig. 18. This prompt will be launched by Anaconda, and will begin setting up the programming environment.

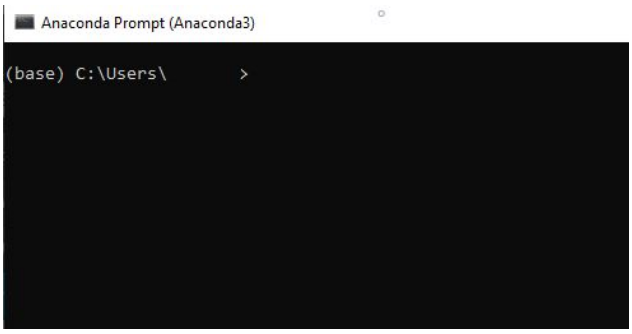

Fig. 18. Anaconda Prompt begins initializing the NASA development environment.

The prompt should look exactly like a regular command prompt or terminal, however it will be named Anaconda Prompt. Once this prompt is open you can go ahead and run these commands to create the Python Environment and then activate the environment:

1. conda create -n NASAEnvironment python=3.7 anaconda
2. source activate NASAEnvironment

10.1.1.3. Install Necessary Packages

There are several packages used inside the development environment. Such packages like Keras, TensorFlow, Matplotlib, Numpy, Pandas, and seaborn will need to be installed through the Anaconda terminal. To install the above-mentioned packages, follow the following instructions:

1. conda install -n NASAEnvironment keras==2.2.4
2. conda install -n NASAEnvironment tensorflow==1.14.0
3. conda install -n NASAEnvironment matplotlib==3.1.1
4. conda install -n NASAEnvironment numpy==1.16.5
5. conda install -n NASAEnvironment pandas==0.25.1
6. conda install -n NASAEnvironment seaborn==0.9.0

10.1.1.4. Installing Spyder 3

To install Spyder 3, the IDE, go to the Anaconda Navigator Environments, make sure that NASAEnvironment is selected and then go back to Home and Install Spyder, if specific version is needed click the settings gear and it will show a dropdown to install the specific version of 3.3.6. Fig. 19 below depicts the process. Clicking on the black gear as indicated by the arrow will allow one to switch the version. Then, by pressing install, Spyder 3 will install on the machine.

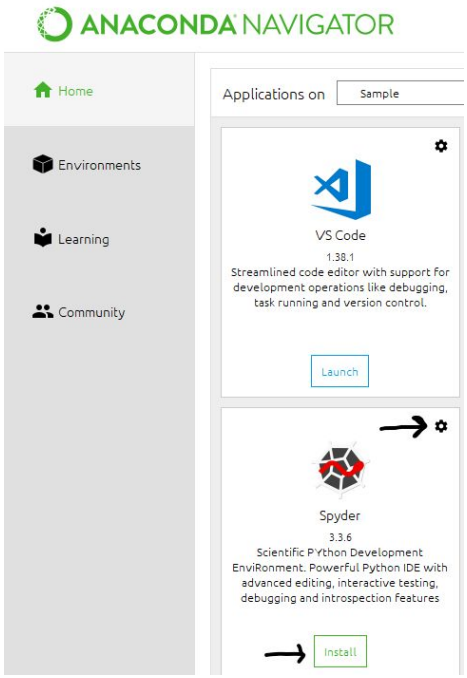

Fig. 19. Installation of Spyder 3 through Anaconda Navigator.

10.2. Instructions for System Deployment

Within this section, the overall instructions will be explored for system deployment. All platform requirements and system installation instructions are included below for anyone wishing to utilize the models described throughout the technical document.

10.2.1. Platform Requirements

The Hall Thruster Data Analysis tool was implemented in Python and is cross-platform between Windows, macOS, and Linux. Simply follow the installation guide detailed in Sec. 10.2 if you wish to deploy the system, or Sec 10.1 if you wish to develop for the system. All that is required is to set up the appropriate environment to execute any required scripts.

10.2.2. System Installation

The Hall Thruster and Data Analysis tool can be installed with ease. The codebase can be downloaded from the GitHub Repository. From there the scripts can either be run through a Python IDE, or through the command prompt. Fig. 20 below shows an image of the preprocessing script being run within the Spyder IDE. The steps for installing the necessary packages, dependencies, and IDE are the same as covered in Sec. 10.1.1.

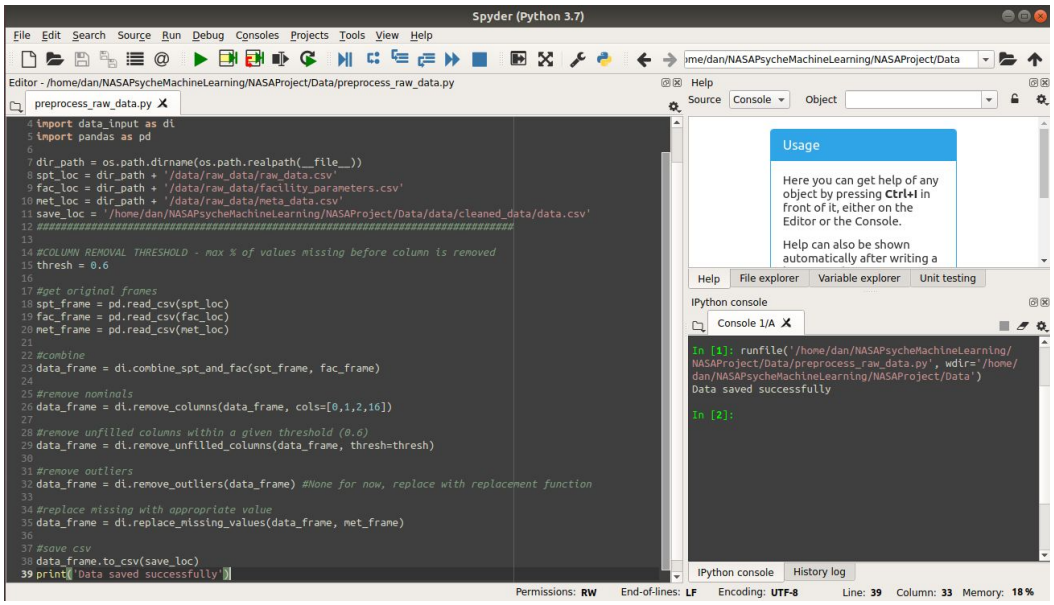

Fig. 20. Running the preprocessing script through the Spyder IDE.

All of the scripts are functional through the IDE. If the IDE is not available, all scripts are still executable as long as python 3.7 is installed as specified in previous sections. The scripts can be run by clicking the green arrow button shown at the top bar below 'Debug' in Fig. 20. Fig. 21 shows how the preprocessing script can be executed through the command prompt.

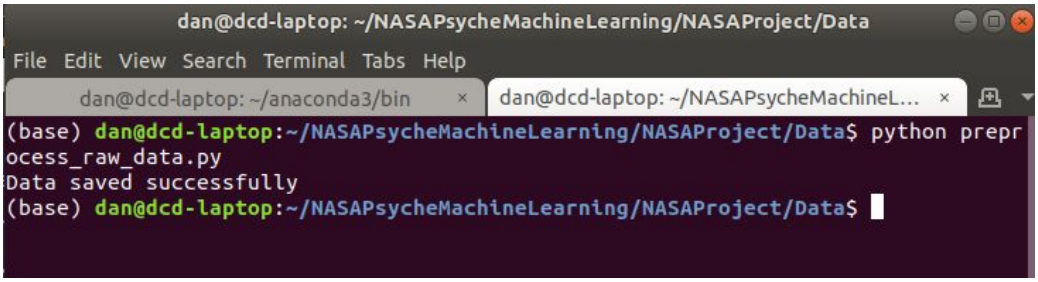

Fig. 21. Running the preprocessing script through the terminal.

The instructions for running the script outside the IDE are the same for all platforms, as long as the terminal on Linux/macOS, or the command prompt on Windows is available, and the correct python version is installed. For running the script as shown in Fig. 21, first navigate to the folder containing the script, then run the following command:

python [SCRIPT_NAME].py

If any errors occur, ensure that the command prompt or terminal is being run as an administrator, and ensure that the script is executable from the command line. On linux, you can determine if a file is executable by a user by using the command:

ls -l

10.3. Instructions for System End Users

End users in this case are considered to be the future developers looking to gain information from the system. For these individuals, following the above steps in Sec 10.1.1 will allow the future developers to continue research utilizing the implemented techniques and begin their own research.

On the other hand, if the developer merely wishes to utilize the scripts on new data sets without modifying the system, then following the steps in Sec 10.2 will set up the system for immediate execution. If the developers wish to utilize the system as-is, then they should ensure that their personal dataset matches the format of the raw_data csv file to ensure that the system works as configured. Any changes to the raw dataset's format should follow instructions laid out in Sec 10.1.1 to ensure the developer is correctly setting up their experiment.

11. Conclusion

The following section discusses the conclusion of this CapStone project and the technical report. Sec. 11.1 discusses the achievements of the project, Sec. 11.2 covers the lessons learned throughout the project by the developers, while Sec 11.3 acknowledges those who supported the team throughout their college and CapStone experience.

11.1. Achievement

The goal of creating the Hall Thruster Data Analysis Tool was to examine relationships between the SPT performance variables and vacuum facility parameters to discover new correlations, as well as try to predict the sensitivities experienced by the thrusters within these vacuum facilities. Overall, the team was successful in implementing a deep learning structure called the Deep Belief Network (DBN), and through experimentation, the team was able to produce a model with an accuracy of 98% when tested against the original data set.

The team was also able to overcome the problem of not having enough data records to adequately train models. This was accomplished by implementing a Kalman filter to generate estimated data records. Through this, machine learning models could be trained using the plethora of points generated. Additionally, the team generated numerous scatterplots, a heatmap, several regression graphs, and other artifacts to examine the relationships between the variables. These results are further discussed in Appendix RA, the analysis of the results.

11.2. Lessons Learned

Throughout the project, the team overcame numerous challenges, as detailed in Sec. 9. The team learned early on about the sunk cost fallacy, where continuing to invest resources to avoid loss and maintain the status quo can be more harmful than admitting errors and starting in a new direction [26]. This occurred while the team was struggling to redefine its project boundary. Instead of working on a system with multiple faults, the team instead chose to rework a large portion of the system's structural design to better match the functional requirements. Although it was more work, the result is superior to using a flawed system.

The team additionally learned about being personally responsible for the planning, execution, and confirmation of activities through the CapStone system. The team did not have much experience in project and workflow management before this project, so managing the work without much assistance from faculty was a challenge. Like any job, however, simply putting forth an effort to learn and to try allowed the team to learn these management skills and complete the project.

11.3. Acknowledgment

This CapStone project is sponsored by Arizona State University in collaboration with NASA. This project is managed by Cassie Bowman, an associate research professor in the School of Earth and Space Exploration at Arizona State University and co-investigator on the Psyche Asteroid mission. The principal investigator of the project is Dr. Jason Frieman, an electric propulsion systems researcher at the NASA Glenn Research Center.

Each member experiences their journey during their time at Penn State Behrend. For Daniel Donley, much of his financial support came out of his pocket by working multiple part-time jobs throughout his college career, with some support from his parents, Patrick and Tina Donley. To pay for his college tuition, Dan took on part-time jobs on and off-campus to make up the deficit each semester and continue his education. Often, he would work 20 to 30 hours per week on top of schoolwork and maintain an active social life. At some points, he even had up to 3 part-time jobs at once just to make ends meet. Regardless, much of Daniel's academic success comes from the support of the excellent faculty found in the software and computer engineering department at Behrend. Through the assistance of industry mentor Dr. Wen-li Wang, Daniel and the rest of the team learned valuable skills and received critical feedback to improve themselves. Additionally, Dr. Abdallah Abdallah encouraged Daniel to explore areas of interest outside his comfort zone, solve problems in unique ways, and gain confidence in presenting his work to others for critical review.

James Fennelly experienced a multitude of new perspectives within his years in Penn State. Starting in Penn State Harrisburg, James was financially supported through his own personal loans and scholarships, as well as some financial support from his parents, Jim and Betty Fennelly. James worked throughout his years at school as a Chegg tutor, a food service worker, and student researcher under Dr. Abdallah Abdallah. Always having a passion for computers and programming, James has been intrigued with how computers worked since about the age of four, beginning to program at the age of ten. With support by friends and colleagues, James continuously learned new material to refine his current success within primary school and college. James found a plethora of invaluable knowledge working under Dr. Abdallah Abdallah, learning how to effectively research and document coherently, opening new ways of approaching problems to find unique solutions to unsolved problems. James would also like to thank Dr. Wen-li Wang for constantly challenging his problem solving abilities over the past two years, providing a stronger foundation to becoming an effective software engineer. Finally, James would like to thank Dr. Naseem Ibrahim, for exposing James to the world of web services and instructing our class in senior design. Everyone listed above made a massive impact on James Fennelly's progress and perspective on algorithmic design, engineering, and research.

The experience of Alec's journey during his time at Penn State Behrend was great at times and not so great at other times. Most of Alec's financial support came in the form of student loans and out of the pockets of family members such as his mother Christy Szolis, and his now late great grandmother, Nancy Fischer/Nunzia Mineo. To pay for necessary goods to live such as

rent, food, utility bills, etc., Alec took up part-time jobs off-campus to pay for these necessities without impacting his family's financial situation whatsoever. Alec would typically work 20 hours a week or less to cover these necessities while also trying to maintain good academic standing. There were some points in Alec's college career where Alec thought he wasn't good enough at what he was studying or planning on becoming, however with the help of his friends, colleagues and some faculty at Penn State Behrend, he was reassured that he was good enough to complete his academic studies. Alec always tried to surround himself with peers that he considered were better than him, as he would try to interpret their extensive knowledge to add it to his own. Some of these peers include the very group members that worked on this project, Daniel Donley and James Fennelly. He contributes parts of his academic success to them as he could always go to them for help if he encountered trouble in something he was working on, whether it was this project itself or a project for another class. Another colleague that affected Alec's work ethic was Daniel Kovelavich. Alec had many classes with this student and became friends with this student within the classroom. If you know who this student is then you might know he has a job offer waiting for him at Microsoft, a massive software company that almost anyone would want to work for. With Daniel's accomplishment, he set the bar for how great Alec must become at programming and software engineering to accomplish his own goals. Lastly, Alec contributes to the rest of his academic success to the faculty at Penn State Behrend. More specifically he contributes his academic success to his faculty advisor Dr. Wen-Li Wang, along with Dr. Matthew White, Dr. Richard Zhao, and Dr. Naseem Ibrahim. All of these professors proved to be very helpful for Alec and Alec was able to contact them whenever he needed assistance. Alec thinks that these professors make up the heart and soul of the software engineering department at Penn State Behrend and hope that if new professors come, that they emulate the same ethics that these professors do, as these professors show that they truly care about a students academic learning progress and are always willing to help any student.

12. References

[1] Arizona State University, *Psyche Mission - A Mission to a Metal World,* https://psyche.asu.edu/
[2] Frieman, J.D., Characterization of Background Neutral Flows in Vacuum Test Facilities and Impacts on Hall Effect Thruster Operation, 2017, Chapter 1.
[3] Oh, D. Y., Collins, S., Goebel, D., Hart, B., Lantoine, G., Snyder, S., ... & Rotlisburger, L. (2017). *Development of the Psyche Mission for NASA's Discovery Program.* In 35th International Electric Propulsion Conference, Atlanta, Georgia.
[4] Jet Propulsion Laboratory at the California Institute of Technology, *Missions | Psyche*, https://www.jpl.nasa.gov/missions/psyche/
[5] Delgado, J. J., Lord, P., & Rotlisberger, L. C. (2016). *Adaptability of the SSL SPT-140 Subsystem for use on a NASA Discovery Class Missions: Psyche.* 52nd AIAA/SAE/ASEE Joint Propulsion Conference, p. 4542.
[6] Lisa lab, *Deep Belief Networks*. Date Accessed 10/9/19, http://deeplearning.net/tutorial/DBN.html
[7] NumPy developers, *NumPy*. Accessed 10/26/19, https://numpy.org/
[8] tbabb, *How a Kalman filter works, in pictures.* Accessed 1/27/20, https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/
[9] Statistics Solutions 2019, *Correlation (Pearson, Kendall, Spearman).* Accessed 2/17/20, https://www.statisticssolutions.com/correlation-pearson-kendall-spearman/
[10] McKinney, W., *The pandas project*. Date Accessed 10/26/19, https://pandas.pydata.org/about.html#community
[11] Labbe, R., *FilterPy*. Accessed 2/6/20, https://filterpy.readthedocs.io/en/latest/index.html
[12] Praynay, D., *Not 1, not 2…but 5 ways to Correlate*. Accessed 2/17/20, https://towardsdatascience.com/not-1-not-2-but-5-ways-to-correlate-6ac92cf42f0f
[13] Hunter, J., Dale, D., Firing, E., Droettboom, M., *Matplotlib.* Date Accessed 10/26/19, https://matplotlib.org/#
[14] Waskom, M., *seaborn: statistical data visualization*. Date Accessed 10/26/19, https://seaborn.pydata.org/
[15] Chollet, F., *Keras: The Python Deep Learning library.* Date Accessed 10/26/19, https://keras.io/
[16] Google Brain. *Why TensorFlow.* Date Accessed 10/26/19, https://www.tensorflow.org/about
[17] Anaconda Inc, *Anaconda Distribution.* Accessed 10/7/19, https://www.anaconda.com/distribution/#download-section
[18] Feng, J., *The MVC for Machine Learning: Data-Model-Learner (DML) — Part 1*. Date Accessed 10/7/19, https://hackernoon.com/the-mvc-for-machine-learning-data-model-learner-dml-8127d793f930
[19] Hedley, J., *jsoup: Java HTML Parser.* Date Accessed 12/13/19, https://jsoup.org/
[20] Rossum, G., Warsaw, B., & Coghlan, N. (2001, July 5). *PEP 8 -- Style Guide for Python Code.* Accessed 10/7/19, https://www.python.org/dev/peps/pep-0008/#introduction.
[21] Shmaliy, Y. S., Zhao, S., & Ahn, C. K. I. (2017). Unbiased Finite Impulse Response Filtering: An Iterative Alternative to Kalman Filtering Ignoring Noise and Initial Conditions. *IEEE Control Systems*, *37*(5), 70–89. doi: 10.1109/mcs.2017.2718830

[22] Byrne, M.P. and Jorns, B.A., "Data-driven Models for the Effects of Background Pressure on the Operation of Hall Thrusters," IEPC Paper 2019-630
[23] Chadha, H. S. (2019, November 8). *Extended Kalman Filter: Why do we need an Extended Version?*. Accessed February 14, 2020, https://towardsdatascience.com/extended-kalman-filter-43e52b16757d
[24] Samsuri, S. B., Zamzuri, H., et. al. (2015, September) *COMPUTATIONAL COST ANALYSIS OF EXTENDED KALMAN FILTER IN SIMULTANEOUS LOCALIZATION & MAPPING (EKF-SLAM) PROBLEM FOR AUTONOMOUS VEHICLE.* Accessed 4/19/20, http://www.arpnjournals.com/jeas/research_papers/rp_2015/jeas_0915_2637.pdf
[25] Spyder Project Contributors, *Spyder-Unittest*. Accessed 11/19/19, https://github.com/spyder-ide/spyder-unittest
[26] Arkes, H. R., & Blumer, C. (1985), The psychology of sunk costs. *Organizational Behavior and Human Decision Processes, 35,* 124-140.

Introduction

The following appendix discusses the results obtained from utilizing the Hall Thruster Data Analysis Tool. First, the experimental setup will be discussed on how the machine learning models were trained and evaluated. Then, an analysis of the various correlation figures and results obtained from running the Kalman filter and creating the Deep Belief Network (DBN) models.

Experimental Setup

The DBN models were trained under varying conditions. These conditions include the number of attributes within the data set, the threshold for the Principal Component Analysis (PCA), the number of original records, the number of generated records from the Kalman filter, what type of Kalman filter was being used to generate the data records, the neuron count, the number of epochs, the batch size, the optimizer, and the loss function.

The first property that can be varied when training models is to vary the number of attributes within the data set. Initially, there are a total of 28 attributes within the data set and they are as followed:

- Discharge Power, kW
- Discharge Current, A
- Cathode Flow Rate, mg/s
- Total Flow Rate, mg/s
- Thrust, mN
- Thruster Efficiency, %
- Access Length (m)
- Diameter (m)
- Pumping Speed (kL/s Xenon)
- Diffusion Pump Count
- Diffusion Pump Speed (l/s air)
- Blow Rate (ft^3/min)
- Mechanical Pump Rate (ft^3/min)
- Test Port Length (m)
- Discharge Voltage, V
- Magnet Current, A
- Anode Flow Rate, mg/s
- Tank Pressure, Torr Xe
- Specific Impulse, sec
- Access Diameter (m)
- Base (No Load) Pressure, Torr
- Length (m)
- Cryopump Surface Area (m^2)
- Diffusion Pump Length (in)
- Root Blowers Count
- Mechanical Pump Count
- Test Port Diameter (m)
- Test Port Count

This data set, which will be referred to as DS1 (data set 1), was further reduced to 25 attributes to examine the impact of removing different attributes to the system. DS2 removes the Test Port Diameter (m), Test Port Length (m), and Test Port Count. The data set was reduced further to 18 attributes and is known as DS3. The attributes that were further removed in DS3 were the Diffusion Pump Count, the Diffusion Pump Length (m), and the Diffusion Pump Speed (l/s air). This was later reduced to the final set, DS4, with 16 attributes. The large bulk of the experiments utilized DS4, as many of the original 28 attributes were determined to pose little to no impact on the system or had little variation in their distributions. The final set of attributes used in the experiments were as follows:

- Discharge Power, kW
- Discharge Current, A
- Cathode Flow Rate, mg/s
- Total Flow Rate, mg/s
- Thrust, mN
- Thruster Efficiency, %
- Pumping Speed (kL/s Xenon)
- Diameter (m)
- Discharge Voltage, V
- Magnet Current, A
- Anode Flow Rate, mg/s
- Tank Pressure, Torr Xe
- Specific Impulse, sec
- Cryopump Surface Area (m^2)
- Base (No Load) Pressure, Torr
- Length (m)

Principal Component Analysis (PCA) is used within the system to reduce the dimensional complexity of the data set being used. For example, DS4 can be reduced from 16 attributes to 10 utilizing PCA. This can reduce the noise within the data set, as well as reduce the time needed to generate the data records and train machine learning models. This is controlled through the PCA threshold parameter, which specifies the sum percentage of the principal components of the PCA reduced data set. For example, a PCA threshold of 0.9 will produce a PCA model with several components, where the sum of each component percentage contribution to the model will equal the PCA threshold. Utilizing this parameter, the machine learning models produced can be further optimized.

During the data collection phase of the project, only approximately 400 data records were collected from various abstracts involving SPT-100 and SPT-140 thrusters. This metric represents the original record count used in the Kalman filter. This filter is then used to generate data records, known as the generated data records. Using these 400 original records and the attributes from one of the four provided data sets, 81,000 new records were generated using the Kalman filter. It's important to note that the number of generated data records is independent of the number of attributes; the number of generated records is only dependent on the original record count.

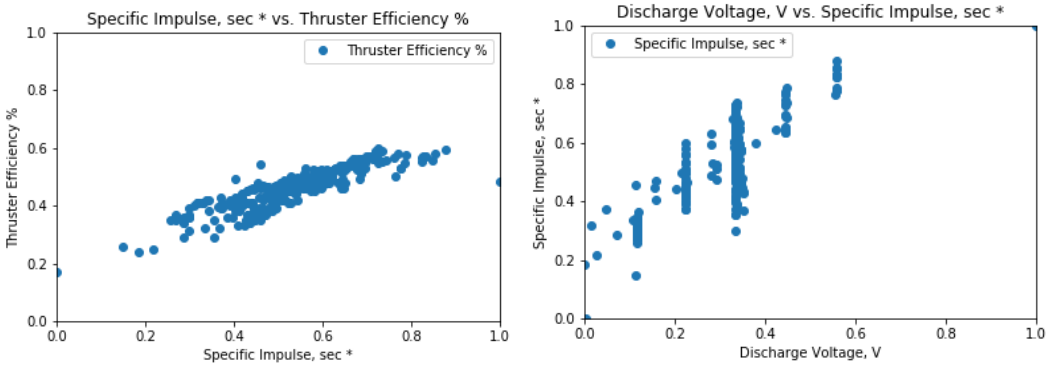

Fig. RA-1. Examples of linear distributions found within the attributes.

For the Kalman filter, two were implemented within the Hall Thruster Data Analysis Tool. The first is the regular Kalman filter, while the other was the Extended Kalman filter. A normal Kalman filter can handle linearly related attributes. Some examples of these relationships are shown in Fig RA-1. The Extended Kalman filter was implemented due to the presence of

nonlinear attributes within the data set, as seen in Fig. RA-2. The standard Kalman filter does not handle nonlinear relationships well, so the Extended Kalman filter was implemented to handle those nonlinearly related attributes.

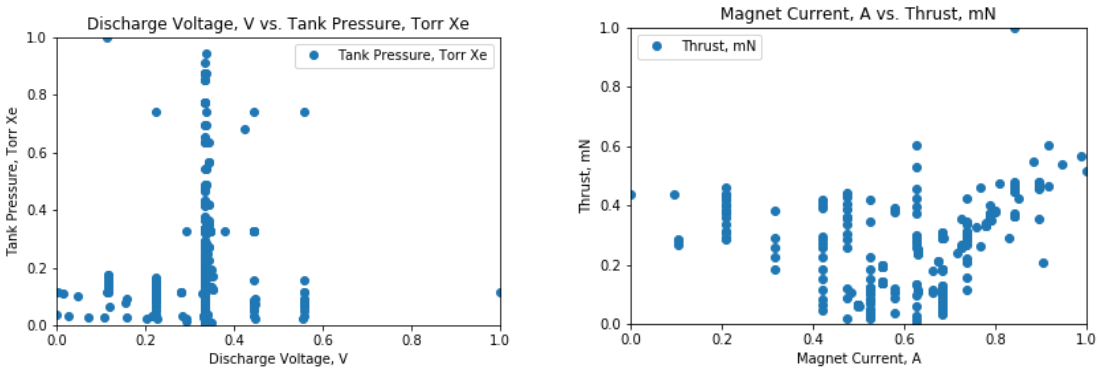

Fig. RA-2. Examples of nonlinear distributions found within the attributes.

Lastly, there are a few parameters that are changed during the training of the machine learning models. The first is the number of epochs that the model is trained for. An epoch is one complete presentation of the data set to the machine learning model. The second is the neuron count, which defines the structure of the model. Next is the batch size, or how many data records are being shown to the model at one time in the current epoch. The last two are the optimizer technique and the loss function used by the model. The optimizer determines how the model parameters will change after each epoch, while the loss function is used to calculate the performance of the machine learning model. These parameters are necessary not only to train accurate models quickly and effectively but to ensure that the model is not over or under-trained.

The experiments performed to vary the following parameters to produce different machine learning models of varying accuracies. Of these parameters, many remain the same, such as the original record count, generated record count, and loss metric do not vary. This is since to change the original record count would mean finding more records or create subsets of the original raw data set. The loss function was not changed to focus on varying other parameters instead.

Results Analysis

Throughout the CapStone project, a total of 26 experiments were conducted to produce the most accurate model possible. Table RA-1 below showcases the results of these 26 experiments. Of these 26 DBN models produced, 9 were able to score an evaluation accuracy of over 90%, meaning the model was over 90% accurate in evaluating the original data set containing the 400 records. Most models, however, scored in the 97-99% range when evaluating against the test set derived from the 81,000 generated data records.

The experiments can be divided into two categories, separated by the black line shown in Table RA-1. These top experiments all have their PCA threshold as 1 to indicate that no PCA was used

to create the model. This was because the PCA features were not fully implemented nor needed at that time. While the PCA was being implemented, the team studied the differences between using the Kalman filter versus the Extended Kalman filter, as well as configuring the neuron counts to optimize the evaluation accuracy. Of these models, the highest-scoring model earned 90%, while most models scored around 50% or below.

ModelNo	FilterType	ReducedAttributeCount	PCA Threshold	AttributesAfterPCA	HiddenLayer1NeuronCount	HiddenLayer2NeuronCount	Epochs	BatchSize	Optimizer	Loss	TrainingAcc	EvalAcc
1	Kalman	25	1	25	18	9	100	32	sgd	mse	0.753	
2	Kalman	25	1	25	18	9	100	32	sgd	mse	0.997	0.2
3	Kalman	21	1	21	18	9	10	128	adam	mse	0.86	0.45
4	Kalman	18	1	18	18	200	100	128	adam	mse	0.92	0.32
5	Kalman	18	1	18	324	81	100	128	adam	mse	0.97	0.32
6	E Kalman	18	1	18	324	81	100	128	adam	mse	0.96	0.52
7	E Kalman	18	1	18	324	81	100	128	adam	mse	0.96	0.5
8	E Kalman	16	1	16	144	36	100	128	adam	mse	0.98	0.79
9	E Kalman	16	1	16	324	81	100	128	adam	mse	0.98	0.9
10	E Kalman	16	1	16	324	81	100	128	adam	mse	0.96	0.85
11	E Kalman	28	0.99	11	324	81	100	128	adam	mse	0.98	0.87
12	E Kalman	16	0.99	11	324	81	100	128	adam	mse	0.97	0.86
13	E Kalman	16	0.9	6	324	81	100	128	adam	mse	0.97	0.95
14	E Kalman	16	0.8	4	324	81	100	128	adam	mse	0.99	0.89
15	E Kalman	16	0.85	5	324	81	100	128	adam	mse	0.99	0.96
16	E Kalman	16	0.95	8	324	81	100	128	adam	mse	0.98	0.88
17	E Kalman	16	0.875	6	324	81	100	128	adam	mse	0.99	0.95
18	E Kalman	16	0.925	7	324	81	100	128	adam	mse	0.98	0.95
19	E Kalman	16	0.9	6	36	9	100	128	adam	mse	0.99	0.5
20	E Kalman	16	0.9	6	512	256	100	128	adam	mse	0.99	0.91
21	E Kalman	16	0.9	6	256	128	100	128	adam	mse	0.99	0.71
22	E Kalman	16	0.9	6	324	128	100	128	adam	mse	0.99	0.88
23	E Kalman	16	0.85	5	512	256	100	128	adam	mse	0.99	0.88
24	E Kalman	16	0.9	6	1024	512	100	128	adam	mse	0.99	0.94
25	E Kalman	16	0.9	5	2048	1024	100	128	adam	mse	0.99	0.94
26	E Kalman	16	0.99	10	2048	1024	100	128	adam	mse	0.99	0.97

Table RA-1. The results table from training Deep Belief Network Models

From this point, the team decided that the Extended Kalman filter would be better suited for more experiments to optimize accuracy. It's important to note, however, that experiment 1 does not have an evaluation accuracy. This is because the model produced was a test to determine if the system was functional enough to begin experimenting. The evaluation accuracy was recorded starting with experiment 2.

In the bottom portion of the experiments, the team kept the filter type the same, while examining the impact of using Principal Component Analysis on the system. To control how the PCA reduced the data, the threshold parameter was included to track the results. By changing this value, the PCA model will reduce the data based on the threshold, and the result will be a data set with several columns equal to the 'AttributesAfterPCA' column in Table RA-1. Experiments 11 through 18 were conducted to determine the impact of the threshold. Utilizing DS4 primarily, the team found that utilizing a PCA threshold of around 0.9 produced decently accurate models.

After determining approximately what value was appropriate for the PCA threshold, the neuron layer counts were varied to determine optimal values as well. The team found that with a lower neuron count, as seen in experiments 19 & 21, the evaluation accuracy dropped to lower percentages as opposed to the higher neuron counts seen in other experiments like experiment 20. Overall, at least 512 neurons for the first kind of layer and 256 neurons for the second kind of layer proved effective in fitting the 81,000 records to the model during training. Due to time

constraints, the optimizer and loss functions remained mostly the same. The only exceptions were in experiments 1 and 2, which used the Stochastic Gradient Descent (SGD) over the over experiments, which used the Adam Optimizer.

To help examine the results within the DBN results table, several types of figures and graphs were created to facilitate the analysis. These plots include scatter plots, a correlation heatmap between the variables, Lasso Regression analysis on the system, Ridge Regression analysis on the system, and graphs depicting the principal components from PCA.

To examine the correlations within the system, a correlation heatmap displaying the Pearson correlation values between all variables was created. This heatmap is displayed below in Fig. RA-3.

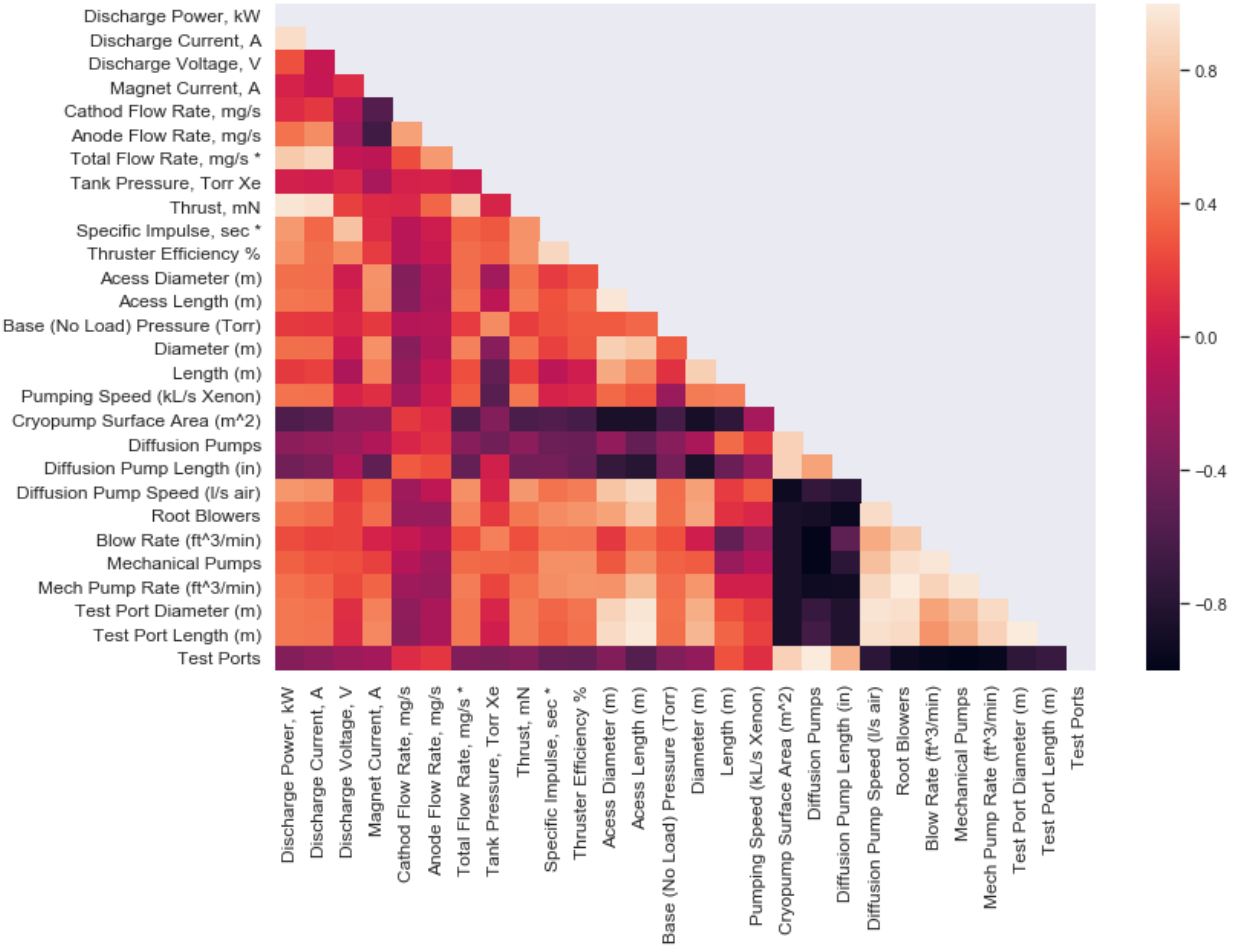

Fig. RA-3. A Heatmap of the Pearson correlations between all attributes in the system

Looking at the correlation heatmap shown in Fig. RA-3, there are several attributes with high correlations, specifically amongst the facility parameters. As discussed previously, these facility

parameters were removed due to these high correlations, since many of these parameters had little varied data, which is not very useful for creating accurate models. These variables could be included if more values are obtained. Regardless, the heatmap shown in Fig. RA-3 played a vital role in reducing the original 28 attributes down to the 16 attributes seen in DS4.

Lasso regression and Ridge regression were also performed to produce more correlations for examinations. Utilizing Ridge regression and Lasso regression, the beta coefficients were calculated for DS4, as seen in Fig. RA-4. Within this figure, many of the beta values are greater than 1, indicating that the system is complex and is at risk for overfitting. To ensure that no overfitting is occurring, more experimentation is necessary for utilizing more new data records to form new data sets. In the Ridge regression graph, the tank pressure has a very high beta value when compared to the thruster efficiency.

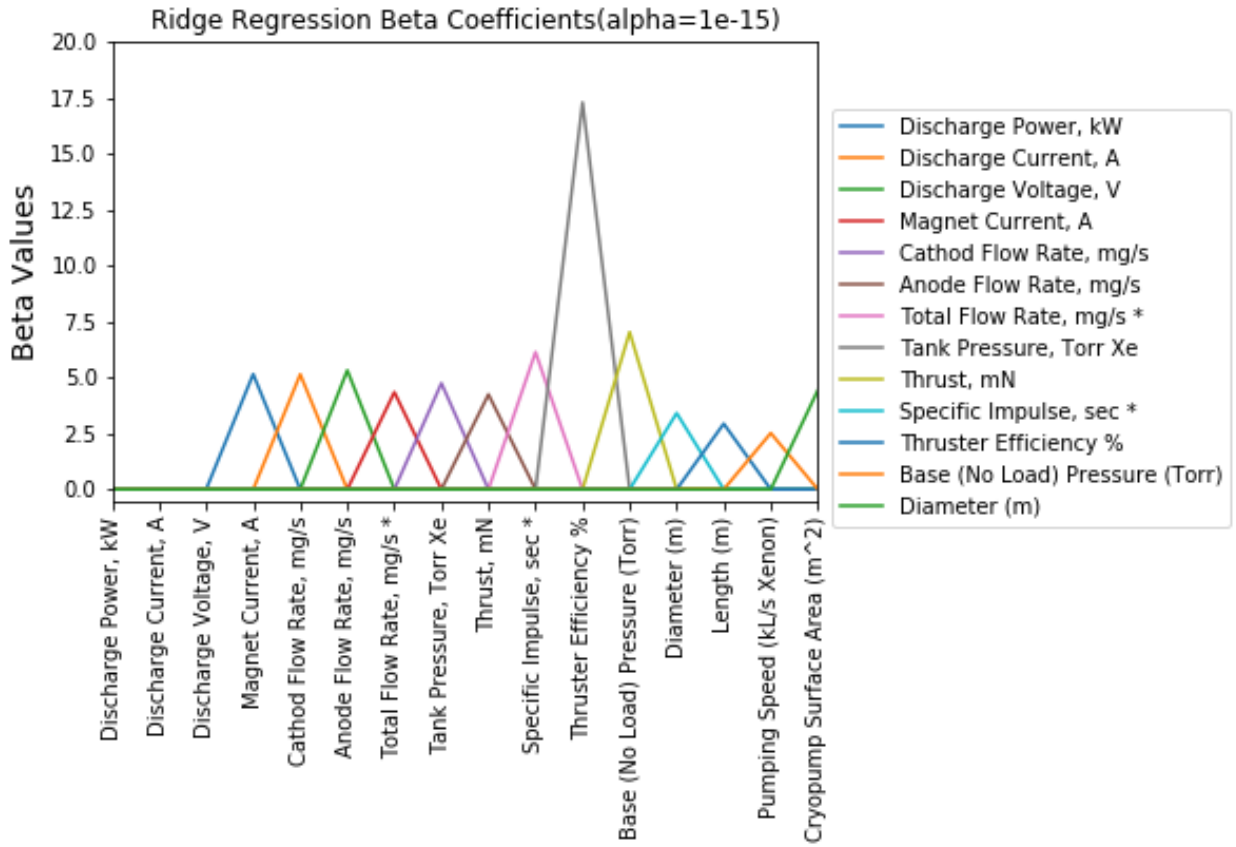

Fig RA-4. Ridge regression beta values with an alpha coefficient of $1*10^{-15}$.

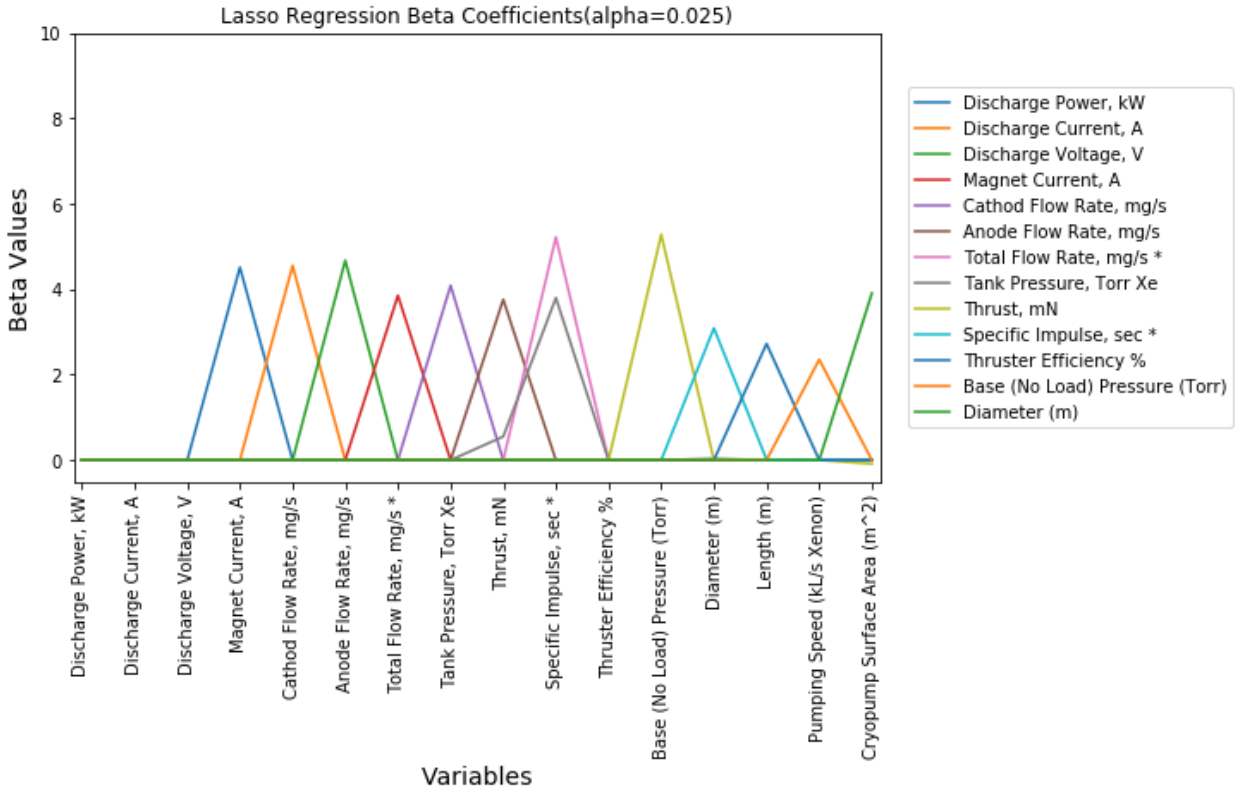

Fig RA-5. Lasso regression beta values with an alpha coefficient of 0.025.

The Lasso regression produced slightly different results, as seen above in Fig. RA-5. The beta value for the thruster efficiency is gone and instead is on the specific impulse. Similarly, the beta values are all above 1, indicating the system is complex and at risk of overfitting.

Moving onto the graphs created by the Principal Component Analysis (PCA), they go over the feature influence scores among the different principal components. Looking at Fig RA-6, this principal component accounts for 62.58% of the PCA model used to perform the reduction, making it the most influential component. Within Fig RA-6, it's clear that the pumping speed, specific impulse, and thrust played the biggest roles in the reduction. A possible improvement to the model then could be to see the impact of removing the pumping speed to see how it plays a role in training machine learning models.

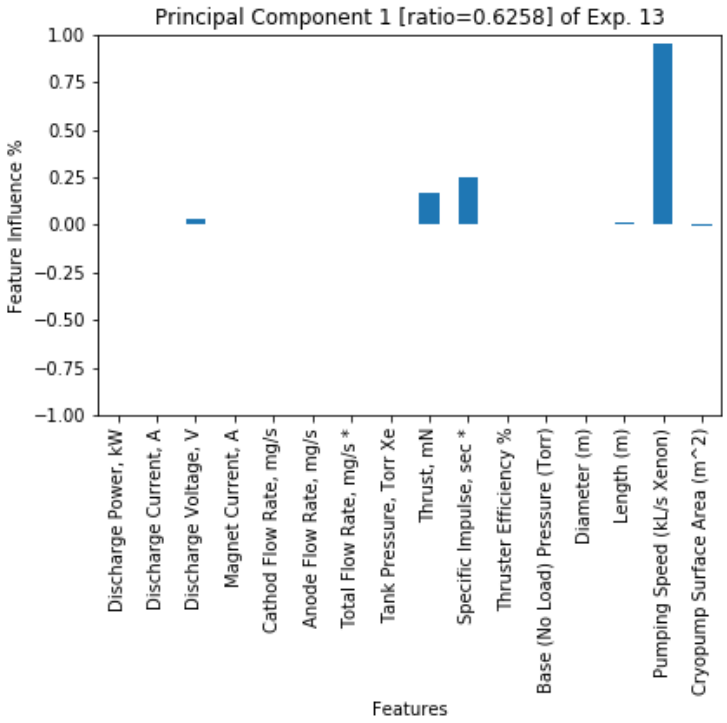

Fig RA-6. Feature Influence Scores of Principal Component 1.

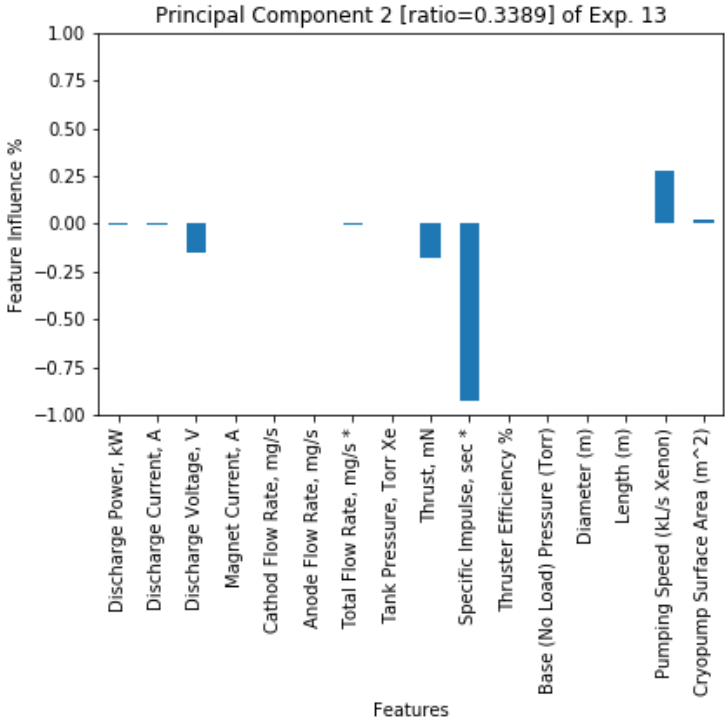

Fig RA-7. Feature Influence Scores of Principal Component 2.

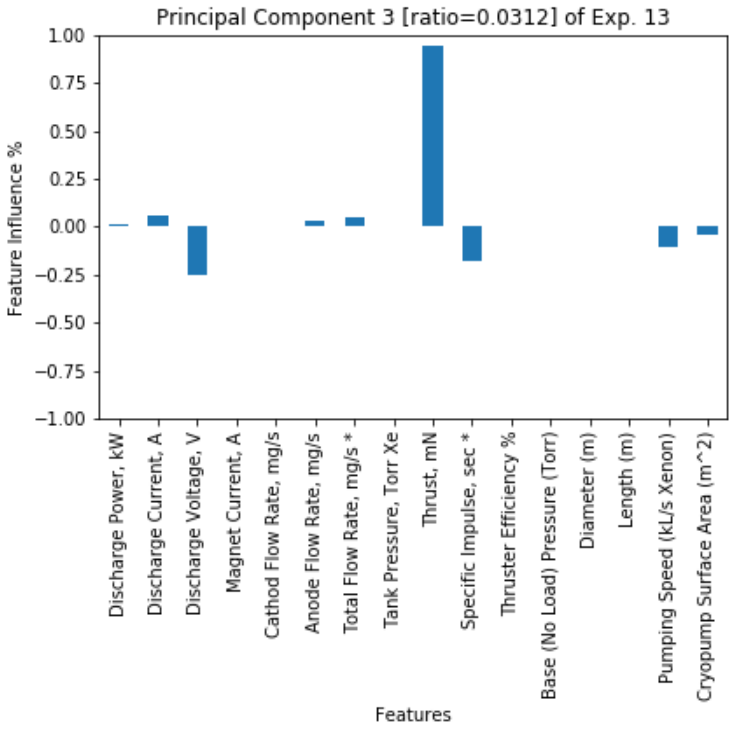

Fig RA-8. Feature Influence Scores of Principal Component 3.

Fig RA-7 shows principal component 2, which accounts for 33.89% of the PCA model. Similar to the first component, there is a positive influence score from the pumping speed, while there is a large negative score from the specific impulse, with smaller negative scores for the thrust and discharge voltage.

The last PCA graph comes from principal component 3, which accounts for 3.12% of the model. Similar to the other PCA graphs, the main attributes are the discharge voltage, thrust, specific impulse, and pumping speed. However, the thrust has the highest score and is positive, as opposed to the other graphs. Looking at these three PCA graphs, it's clear that more research into the pumping speed, thrust, specific impulse, and discharge voltage is necessary to increase the accuracy of future models.

Overall, there are several more aspects of the model that can be improved based on the results from Table RA-1, and the various figures shown throughout this analysis. For example, backward attribute selection can be used to examine how the specific impulse impacts the system and training models. Additionally, more graphs can be generated utilizing the Hall Thruster Data Analysis Tool for further analysis.

Conclusion

Throughout this project, several experiments were conducted to optimize the accuracy in training DBN models, many of which obtained an evaluation accuracy against the original data set of around 90% or more.

The PCA graphs, as well as the Lasso and Ridge graphs, leave questions about how certain attributes impact the system. For example, the tank pressure needs to be further researched from the Ridge regression graph from Fig RA-4, and based on the PCA graphs, the pumping speed, thrust, specific impulse, and discharge voltage all play key roles in forming the PCA models used to reduce the complexity of the data set. Researching the impact of these attributes may be useful.

There are several aspects to the experiments that can be improved to possibly improve the accuracy of the machine learning models. For instance, backward attribute selection may be utilized to further determine which attributes can be removed from the system. One attribute can be removed at a time to observe the impact on the system. Right now, the data set with the lowest number of attributes is DS4 with 16. This data set can be reduced further to examine the impact on the system. Likewise, different loss functions and optimizers can be implemented to examine their impacts as well. Looking back at Fig. RA-3, the correlation heatmap, some variables can also be removed based on the correlations. For instance, the discharge current has a high correlation to the discharge power, meaning that one of those variables can be removed to potentially improve the system. This is just one possible improvement among others.

Additionally, different filters can be used besides the Kalman filter and Extended Kalman filter to generate the data records, or if enough data records are provided initially, then the filter will not be needed to generate records at all. It’s also possible that the solutions provided do not scale to larger data sets that are independent of the system. For instance, a model with a 95% accuracy rating against the initial raw data set may score significantly lower when evaluated against these different, larger data sets. The only way to determine if this is true is to evaluate the models against more data sets to confirm their results.

Different types of machine learning models such as those in the reinforcement learning category could also be implemented to attempt to minimize the sensitivities. Other algorithms could additionally be implemented in substitute of the DBN. Based on the approximately 400 data records collected, the team was able to produce a model with a 97% accuracy rate. Although this percentage can be improved, this is a significant step in helping to understand the impact of these facility parameters on SPT-100 and SPT-140 thrusters.

Table 4.7. User Functional Requirements: UF-A

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**UF-A**			**Type**	**Functional** / **Non-Functional**
Creation:				**User**	☒ / ☐
Modification:				**System**	☐ / ☐
Description:	The team will devise a method to predict and/or correct for the sensitivity of the thruster output parameters (i.e. HET performance, operation, and plume parameters).				
Priority:	Highest	✔ **High**	Medium	Low	Lowest
This Req. is Refined Into:	**SF-A-01, SF-A-03, SF-A-04, SF-A-05, SF-A-06**				
Justify why UF-A can be completely covered by SF-A-01, SF-A-03, SF-A-04, SF-A-05, SF-A-06	The system has various performance and operation parameters that need to be modeled. The first few requirements discuss the main parameters the system needs to predict. The last requirements discuss the various methods used to make these predictions.				
Traceability:	**Use cases cf.**	**UC-001, UC-004, UC-005**			
	Test cases cf.	**TC-003, TC-004, TC-005, TC-006, TC-007, TC-018, TC-019, TC-020, TC-021, TC-025, TC-028, TC-029, TC-037, TC-038, TC-039**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Table 4.8. User Functional Requirements: UF-B

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**UF-B**			**Type**	**Functional** / **Non-Functional**
Creation:				**User**	☒ / ☐
Modification:				**System**	☐ / ☐
Description:	The team must include a report detailing the results of the machine learning analysis and data mining techniques including utilized data sets, any models/analyses generated by the analysis and techniques, and an evaluation of the model accuracy and predictive capability.				
Priority:	✔ **Highest**	High	Medium	Low	Lowest
This Req. is Refined Into:	**SF-B-01, SF-B-02, SF-B-03, SF-B-04, SF-B-05**				
Justify why UF-B can be completely covered by SF-B-01, SF-B-02, SF-B-03, SF-B-04, SF-B-05	ASU requires a report to be generated with the training data, models and analysis generated by the system's training, and the predictive capability and accuracy from each model trained. These requirements discuss the various pieces of information that will be included in the final report to ASU.				
Traceability:	**Use cases cf.**	**UC-001, UC-003, UC-004, UC-005**			
	Test cases cf.	**TC-003, TC-005, TC-006, TC-007, TC-014, TC-015, TC-018, TC-022, TC-023, TC-024, TC-025, TC-026, TC-036, TC-037, TC-038**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Table 4.9. User Functional Requirements: UF-C

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**UF-C**			**Type** / **Functional** / **Non-Functional**	
Creation:				**User** ☒	☐
Modification:				**System** ☐	☐
Description:	The team should gather information and data from HET vacuum test facilities that tested Hall thrusters in order to build the data set for their system.				
Priority:	✔ **Highest**	High	Medium	Low	Lowest
This Req. is Refined Into:	**SF-C-01, SF-C-03, SF-C-04, SF-C-05, SF-C-06, SF-C-07, SF-C-08**				
Justify why UF-C can be completely covered by SF-C-01, SF-C-03, SF-C-04, SF-C-05, SF-C-06, SF-C-07, SF-C-08	To successfully gather the data, it must be extracted from the source(s), then altered into a consistent format so it can be properly analyzed to extract information. This involves removing unnecessary data like outliers and missing values, as well as normalizing or standardizing the data so useful information can be extracted.				
Traceability:	**Use cases cf.**	**UC-001, UC-002**			
	Test cases cf.	**TC-001, TC-002, TC-008, TC-009, TC-010, TC-011, TC-012, TC-013, TC-017, TC-027, TC-035**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Table 4.10. User Functional Requirements: UF-D

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**UF-D**			**Type** / **Functional** / **Non-Functional**	
Creation:				**User** ☒	☐
Modification:				**System** ☐	☐
Description:	The team should utilize data mining techniques to discover previously undiscovered correlations within public data sets.				
Priority:	✔ **Highest**	High	Medium	Low	Lowest
This Req. is Refined Into:	**SF-D-01, SF-D-02, SF-D-03, SF-D-04, SF-D-05**				
Justify why UF-D can be completely covered by SF-D-01, SF-D-02, SF-D-03, SF-D-04, SF-D-05	There are various techniques available to discovering correlations within the data set being used. These techniques include Pearson correlation analysis, correlation matrix, PCA, and Lasso Regression. These various techniques will provide correlations between the variables within the system.				
Traceability:	**Use cases cf.**	**UC-001, UC-003**			
	Test cases cf.	**TC-014, TC-015, TC-022, TC-023, TC-024, TC-026, TC-036**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Table 4.11. User NonFunctional Requirements: UP-02

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**UP-02**			**Type** / **Functional** / **Non-Functional**	
Creation:				**User** ☐	☒
Modification:				**System** ☐	☐
Description:	The client wants the system to utilize published data sets from HET Facilities that used SPT-140 and/or SPT-100 Hall thrusters and If time permits, the teams should try to incorporate data from other thruster models.			**Product (sub-type below)**	
				Usability Requirements	
Priority:	✔ **Highest**	High	Medium	Low	Lowest
This Req. is Refined Into:	**SP-02-01, SP-02-02**				
Justify why UP-02 can be completely covered by SP-02-01, SP-02-02	The data set must first be initially restricted to SPT-100 & SPT-140 to find an optimal solution for the thruster types being used, which is requirement UP-02-01. UP-02-02 will then extend the data set restriction if time is available. These 2 thus fit the user requirement.				
Traceability:	**Use cases cf.**	**N/A**			
	Test cases cf.	**TC-001, TC-034**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Table 4.12. User NonFunctional Requirements: UO-01

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**UO-01**			**Type** / **Functional** / **Non-Functional**	
Creation:				**User** ☐	☒
Modification:				**System** ☐	☐
Description:	The user needs the machine learning model to be able to run on the computers in HET facilities so the software can be accessible by anybody who needs it.			**Organizational (sub-type below)**	
				Operational Requirements	
Priority:	Highest	✔ **High**	Medium	Low	Lowest
This Req. is Refined Into:	**SO-01-01, SO-01-02, SO-01-03**				
Justify why UO-01 can be completely covered by SO-01-01, SO-01-02, SO-01-03	The system needs to be able to run on the computers in the HET facilities, and since the implementation will be using Python, all that needs to be considered are the OS types the software will run on. The three system requirements cover the three primary OS types used: Linux, Windows 10, MacOS.				
Traceability:	**Use cases cf.**	**N/A**			
	Test cases cf.	**TC-031, TC-032, TC-033**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Table 4.13. User NonFunctional Requirements: UE-01

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**UE-01**			**Type** / **Functional** / **Non-Functional**	
Creation:				**User** ☐	☒
Modification:				**System** ☐	☐
Description:	Participants may not be citizens of the People's Republic of China (PRC), per Public Laws 112-10, Section 1340(a) and 112-55, Section 536.			**External (sub-type below)**	
				Legislative Requirements on Safety/Security	
Priority:	✔ **Highest**	High	Medium	Low	Lowest
This Req. is Refined Into:	**SE-01-01**				
Justify why UE-01 can be completely covered by SE-01-01	The law simply states that the application cannot be developed by somebody from the PRC, thus this requirement won't allow the system to be worked on by those individuals.				
Traceability:	**Use cases cf.**	**N/A**			
	Test cases cf.	**TC-030**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Table 4.14. System Functional Requirements: SF-A-01

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SF-A-01**			**Type** / **Functional** / **Non-Functional**	
Creation:				**User** ☐	☐
Modification:				**System** ☒	☐
Description:	The system shall predict the thrust in units of newtons the HET is outputting.				
Priority:	Highest	✔ **High**	Medium	Low	Lowest
This Req. is Engineered From:	**UF-A**				
Justify why meeting SF-A-01 can contribute to the fulfilment of UF-A	One of the parameters that is changing as a function of the background pressure and facility parameters is the thrust of the HET. To correctly predict the sensitivity of the thruster output parameters, thrust must be predicted.				
Traceability:	**Use cases cf.**	**UC-001, UC-005**			
	Test cases cf.	**TC-006, TC-018, TC-025, TC-028, TC-038**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Table 4.15. System Functional Requirements: SF-A-03

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SF-A-03**			**Type** / **Functional** / **Non-Functional**	
Creation:				**User** ☐	☐
Modification:				**System** ☒	☐
Description:	The system shall predict the thruster efficiency as a percentage.				
Priority:	Highest	✔ **High**	Medium	Low	Lowest
This Req. is Engineered From:	**UF-A**				
Justify why meeting SF-A-03 can contribute to the fulfilment of UF-A	One of the parameters that changes as a function of facility parameters and background pressure is the discharge current oscillation, which is largely disputed how it changes. Hence, the model should be able to predict the levels.				
Traceability:	**Use cases cf.**	**UC-001, UC-005**			
	Test cases cf.	**TC-006, TC-018, TC-025, TC-028, TC-038**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Table 4.16. System Functional Requirements: SF-A-04

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SF-A-04**			**Type** / **Functional** / **Non-Functional**	
Creation:				**User** ☐	☐
Modification:				**System** ☒	☐
Description:	The system shall predict the specific impulse in seconds the thruster is using.				
Priority:	Highest	✔ **High**	Medium	Low	Lowest
This Req. is Engineered From:	**UF-A**				
Justify why meeting SF-A-04 can contribute to the fulfilment of UF-A	One of the parameters that varies as a result of changing background pressures and facility parameters is the specific impulse that allows the thruster to operate with low fuel usage. Part of predicting the operation and performance of the thruster is knowing the specific impulse, hence it must be calculated.				
Traceability:	**Use cases cf.**	**UC-001, UC-005**			
	Test cases cf.	**TC-006, TC-018, TC-025, TC-028, TC-038**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Table 4.17. System Functional Requirements: SF-A-05

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SF-A-05**				
Creation:					
Modification:					
Description:	The system shall implement a deep belief network (DBN) to predict different thruster output parameters.				
Priority:	Highest	High	Medium	✔ **Low**	Lowest
This Req. is Engineered From:	**UF-A**				
Justify why meeting SF-A-05 can contribute to the fulfilment of UF-A	The goal is to devise a method to accurately predict the output parameters. The model will allow to make predictions of what the values would be, so training a DBN model may prove beneficial to achieving this goal.				
Traceability:	**Use cases cf.**	**UC-001, UC-005**			
	Test cases cf.	**TC-003, TC-004, TC-005, TC-006, TC-007, TC-018, TC-028, TC-038, TC-039**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Type	**Functional**	**Non-Functional**
User	☐	☐
System	☒	☐

Table 4.18. System Functional Requirements: SF-A-06

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SF-A-06**				
Creation:					
Modification:					
Description:	The system shall utilize a Kalman filter to estimate the values of the output parameters.				
Priority:	Highest	✔ **High**	Medium	Low	Lowest
This Req. is Engineered From:	**UF-A**				
Justify why meeting SF-A-06 can contribute to the fulfilment of UF-A	An efficient method of analyzing the data is to use a Kalman filter, which is capable of accurate predictions when dealing with systems with high amounts of uncertainty. Using the filter, we will be able to obtain more data points, make estimates of the predictions, and even potentially train machine learning models.				
Traceability:	**Use cases cf.**	**UC-001, UC-004**			
	Test cases cf.	**TC-019, TC-020, TC-021, TC-025, TC-028, TC-029, TC-037**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Type	**Functional**	**Non-Functional**
User	☐	☐
System	☒	☐

Table 4.19. System Functional Requirements: SF-B-01

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SF-B-01**			**Type** / **Functional** / **Non-Functional**	
Creation:				**User** ☐	☐
Modification:				**System** ☒	☐
Description:	The system shall record the results for running the machine learning model or data mining technique to local storage.				
Priority:	✔ **Highest**	High	Medium	Low	Lowest
This Req. is Engineered From:	**UF-B**				
Justify why meeting SF-B-01 can contribute to the fulfilment of UF-B	The overall goal of the requirement is to provide ASU with a report explaining how effective the machine learning models were. A report generator would be able to quickly compile and print out the information required to be handed over.				
Traceability:	**Use cases cf.**	**UC-001, UC-004, UC-005**			
	Test cases cf.	**TC-006, TC-018, TC-023, TC-024, TC-025, TC-026, TC-036, TC-038, TC-039**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Table 4.20. System Functional Requirements: SF-B-02

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SF-B-02**			**Type** / **Functional** / **Non-Functional**	
Creation:				**User** ☐	☐
Modification:				**System** ☒	☐
Description:	The system shall record the training data used to train a model for the report.				
Priority:	✔ **Highest**	High	Medium	Low	Lowest
This Req. is Engineered From:	**UF-B**				
Justify why meeting SF-B-02 can contribute to the fulfilment of UF-B	The training data is needed help evaluate the model and generate any analysis.				
Traceability:	**Use cases cf.**	**UC-001, UC-004**			
	Test cases cf.	**TC-005, TC-018, TC-037, TC-038**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Table 4.21. System Functional Requirements: SF-B-03

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SF-B-03**				
Creation:					
Modification:					
Description:	The system shall record the machine learning model type being used and the resulting analysis from the model for the report.				
Priority:	✔ **Highest**	High	Medium	Low	Lowest
This Req. is Engineered From:	**UF-B**				
Justify why meeting SF-B-03 can contribute to the fulfilment of UF-B	Part of evaluating the model's effectiveness is knowing what type of model was used, and the results of the trained model. Thus, these pieces of information need to be recorded to be inserted into the report.				
Traceability:	**Use cases cf.**	**UC-001, UC-005**			
	Test cases cf.	**TC-003, TC-007, TC-018, TC-038, TC-039**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Type	**Functional**	**Non-Functional**
User	☐	☐
System	☒	☐

Table 4.22. System Functional Requirements: SF-B-04

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SF-B-04**				
Creation:					
Modification:					
Description:	The system shall calculate the predictive capability of the model/technique to include in the report.				
Priority:	✔ **Highest**	High	Medium	Low	Lowest
This Req. is Engineered From:	**UF-B**				
Justify why meeting SF-B-04 can contribute to the fulfilment of UF-B	Part of the report is knowing how effective each machine learning model/data mining technique is, so the best technique is chosen for the job of predicting the sensitivity. Hence, the accuracy and capability need to be calculated and included in the report.				
Traceability:	**Use cases cf.**	**UC-001, UC-005**			
	Test cases cf.	**TC-006, TC-018, TC-025, TC-038**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Type	**Functional**	**Non-Functional**
User	☐	☐
System	☒	☐

Table 4.23. System Functional Requirements: SF-B-05

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SF-B-05**			**Type**	**Functional** / **Non-Functional**
Creation:				**User**	☐ / ☐
Modification:				**System**	☒ / ☐
Description:	The system shall record the correlation results to be included in the report.				
Priority:	✔ **Highest**	High	Medium	Low	Lowest
This Req. is Engineered From:	**UF-B**				
Justify why meeting SF-B-05 can contribute to the fulfilment of UF-B	Besides results from machine learning models, ASU would like us to include anything we deem relevant in this problem's domain. This includes the results of the correlation analysis, which will give us information on how different system parameters will relate to each other.				
Traceability:	**Use cases cf.**	**UC-001, UC-003**			
	Test cases cf.	**TC-014, TC-015, TC-023, TC-024, TC-026, TC-036**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Table 4.24. System Functional Requirements: SF-C-01

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SF-C-01**			**Type**	**Functional** / **Non-Functional**
Creation:				**User**	☐ / ☐
Modification:				**System**	☒ / ☐
Description:	The system shall read in the data to be used from a .CSV file.				
Priority:	✔ **Highest**	High	Medium	Low	Lowest
This Req. is Engineered From:	**UF-C**				
Justify why meeting SF-C-01 can contribute to the fulfilment of UF-C	The first step to process the data for machine learning algorithms/data mining techniques is to read the data into the system for cleaning and processing. This will be done by accepting .CSV files containing all of the different parameters needed.				
Traceability:	**Use cases cf.**	**UC-001, UC-002**			
	Test cases cf.	**TC-001, TC-017, TC-027, TC-035**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Table 4.25. System Functional Requirements: SF-C-03

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SF-C-03**				
Creation:					
Modification:					
Description:	The system shall remove a column of data if more than 60% of the column's values are missing.				
Priority:	Highest	✔ **High**	Medium	Low	Lowest
This Req. is Engineered From:	**UF-C**				
Justify why meeting SF-C-03 can contribute to the fulfilment of UF-C	While we don't want to throw away any data we pull from any abstracts, it might turn out a variable we're tracking has many values missing. In this case, we should preserve the raw data, but exclude that variable for a separate analysis.				
Traceability:	**Use cases cf.**	**UC-001, UC-002**			
	Test cases cf.	**TC-009, TC-010, TC-017, TC-027, TC-035**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Type	**Functional**	**Non-Functional**
User	☐	☐
System	☒	☐

Table 4.26. System Functional Requirements: SF-C-04

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SF-C-04**				
Creation:					
Modification:					
Description:	The system may remove the nominal data columns.				
Priority:	Highest	High	✔ **Medium**	Low	Lowest
This Req. is Engineered From:	**UF-C**				
Justify why meeting SF-C-04 can contribute to the fulfilment of UF-C	If the system is not looking at nominal data groups (things like labels with no ordinance), then they should be removed unless clustering is being performed. This action is necessary to clean, transform, and reduce the data into an acceptable form.				
Traceability:	**Use cases cf.**	**UC-001, UC-002**			
	Test cases cf.	**TC-010, TC-017, TC-027, TC-035**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Type	**Functional**	**Non-Functional**
User	☐	☐
System	☒	☐

Table 4.27. System Functional Requirements: SF-C-05

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SF-C-05**			**Type** / **Functional** / **Non-Functional**	
Creation:				**User** ☐	☐
Modification:				**System** ☒	☐
Description:	The system shall replace missing values with a result from an appropriate mathematical equation.				
Priority:	Highest	✔ **High**	Medium	Low	Lowest
This Req. is Engineered From:	**UF-C**				
Justify why meeting SF-C-05 can contribute to the fulfilment of UF-C	If a column is missing one or two values, it wouldn't hurt to replace those values with a mean, median, or mode depending on the type of data that column is. Since nominal data cannot be compared against each other, missing values will be replaced with the mode. Ordinal data will be replaced with the median. Interval and ratio data will be replaced with either the median or mean.				
Traceability:	**Use cases cf.**	**UC-001, UC-002**			
	Test cases cf.	**TC-011, TC-017, TC-027, TC-035**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Table 4.28. System Functional Requirements: SF-C-06

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SF-C-06**			**Type** / **Functional** / **Non-Functional**	
Creation:				**User** ☐	☐
Modification:				**System** ☒	☐
Description:	The system shall remove any outliers within the data set.				
Priority:	Highest	✔ **High**	Medium	Low	Lowest
This Req. is Engineered From:	**UF-C**				
Justify why meeting SF-C-06 can contribute to the fulfilment of UF-C	Outliers have the potential to skew results when utilizing machine learning algorithms or data mining techniques. To eliminate this unwanted noise, PCA and quartile ranges can be implemented to identify points that are unlikely to be produced, the outliers. Handling these values will improve the quality of information within the data set for analysis.				
Traceability:	**Use cases cf.**	**UC-001, UC-002**			
	Test cases cf.	**TC-012, TC-017, TC-027, TC-035**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Table 4.29. System Functional Requirements: SF-C-07

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SF-C-07**				
Creation:					
Modification:					
Description:	The system shall normalize each column in the data set so all values are between 0 and 1.				
Priority:	Highest	High	✔ **Medium**	Low	Lowest
This Req. is Engineered From:	**UF-C**				
Justify why meeting SF-C-07 can contribute to the fulfilment of UF-C	If two variables are on different scales, like 0 to 10 and 1000 to 3000, many machine learning algorithms and data mining techniques will struggle to identify differences in the smaller scales. To solve this, we can normalize the data to values between 0 and 1. This way, all the variables will be on the same scale so any analysis will be able to find differences easier.				
Traceability:	**Use cases cf.**	**UC-001, UC-002**			
	Test cases cf.	**TC-002, TC-017, TC-027, TC-035**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Type	**Functional**	**Non-Functional**
User	☐	☐
System	☒	☐

Table 4.30. System Functional Requirements: SF-C-08

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SF-C-08**				
Creation:					
Modification:					
Description:	The system shall standardize each column in the data set so all variables have a mean of 0 and standard deviation of 1.				
Priority:	Highest	✔ **High**	Medium	Low	Lowest
This Req. is Engineered From:	**UF-C**				
Justify why meeting SF-C-08 can contribute to the fulfilment of UF-C	If two variables are on different scales, like 0 to 10 and 1000 to 3000, many machine learning algorithms and data mining techniques will struggle to identify differences in the smaller scales. To solve this, we can standardize the data so all columns have a mean of 0 and standard deviation of 1. This will allow us to easily identify outliers, as well as look at the variations between the variables for more beneficial analysis and solve the scaling issue. This is the preferred method over normalization, but both will be implemented for differing reasons.				
Traceability:	**Use cases cf.**	**UC-001, UC-002**			
	Test cases cf.	**TC-013, TC-017, TC-027, TC-035**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Type	**Functional**	**Non-Functional**
User	☐	☐
System	☒	☐

Table 4.31. System Functional Requirements: SF-D-01

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SF-D-01**			**Type**	**Functional** / **Non-Functional**
Creation:				**User**	☐ / ☐
Modification:				**System**	☒ / ☐
Description:	The system shall perform Pearson correlation analysis on the data set.				
Priority:	✔ **Highest**	High	Medium	Low	Lowest
This Req. is Engineered From:	**UF-D**				
Justify why meeting SF-D-01 can contribute to the fulfilment of UF-D	A way to extract correlations from data is to use Pearson correlation analysis, which measures the strength of the linear relationship between two variables. This will show a basic analysis of the correlations within the data.				
Traceability:	**Use cases cf.**	**UC-001, UC-003**			
	Test cases cf.	**TC-022, TC-023, TC-026, TC-036**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Table 4.32. System Functional Requirements: SF-D-02

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SF-D-02**			**Type**	**Functional** / **Non-Functional**
Creation:				**User**	☐ / ☐
Modification:				**System**	☒ / ☐
Description:	The system shall generate a correlation matrix to show the correlations between all of the variables.				
Priority:	✔ **Highest**	High	Medium	Low	Lowest
This Req. is Engineered From:	**UF-D**				
Justify why meeting SF-D-02 can contribute to the fulfilment of UF-D	A correlation matrix can be used when multiple variables are in a system to show all of the correlations against each other in the form of a heat map. This can be useful to examine the patterns between the correlations and potentially eliminate redundant data and extract useful information.				
Traceability:	**Use cases cf.**	**UC-001, UC-003**			
	Test cases cf.	**TC-002, TC-023, TC-026, TC-036**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Table 4.33. System Functional Requirements: SF-D-03

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SF-D-03**				
Creation:					
Modification:					
Description:	The system shall use Principal Component Analysis (PCA) to identify correlations within the data.				
Priority:	✔ **Highest**	High	Medium	Low	Lowest
This Req. is Engineered From:	**UF-D**				
Justify why meeting SF-D-03 can contribute to the fulfilment of UF-D	Using PCA, the system will be identify various correlations by measuring their dimensionality, then determining the level of influence the feature has on the data set. Negative influences will correlate together while positive influences will also correlate together.				
Traceability:	**Use cases cf.**	**UC-001, UC-003**			
	Test cases cf.	**TC-008, TC-014, TC-015, TC-026, TC-036**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Type	**Functional**	**Non-Functional**
User	☐	☐
System	☒	☐

Table 4.34. System Functional Requirements: SF-D-04

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SF-D-04**				
Creation:					
Modification:					
Description:	The system shall calculate Lasso Regression correlations within the data set.				
Priority:	Highest	High	✔ **Medium**	Low	Lowest
This Req. is Engineered From:	**UF-D**				
Justify why meeting SF-D-04 can contribute to the fulfilment of UF-D	Lasso Regression is a variant of ordinary least square (OLS) to calculate correlations. Using this method may show the team some new correlations that other techniques did not show.				
Traceability:	**Use cases cf.**	**UC-001, UC-003**			
	Test cases cf.	**TC-024, TC-026, TC-036**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Type	**Functional**	**Non-Functional**
User	☐	☐
System	☒	☐

Table 4.35. System Functional Requirements: SF-D-05

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SF-D-05**			**Type**	**Functional** / **Non-Functional**
Creation:				**User**	☐ / ☐
Modification:				**System**	☒ / ☐
Description:	The system shall generate scatter plots displaying each variable against every other variable.				
Priority:	✔ **Highest**	High	Medium	Low	Lowest
This Req. is Engineered From:	**UF-D**				
Justify why meeting SF-D-05 can contribute to the fulfilment of UF-D	One way to analyze the relationship between the variables is to graph scatter plots comparing each variable against another variable. This will show linear and nonlinear relationships within the data for further research.				
Traceability:	**Use cases cf.**	**UC-001, UC-003**			
	Test cases cf.	**TC-022, TC-026, TC-036**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Table 4.36. System NonFunctional Requirements: SP-02-01

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SP-02-01**			**Type**	**Functional** / **Non-Functional**
Creation:				**User**	☐ / ☐
Modification:				**System**	☐ / ☒
Description:	The system shall utilize data from HET Facilities that utilized SPT-140 and/or SPT-100 Hall thrusters.			**Product (sub-type below)**	
				Usability Requirements	
Priority:	✔ **Highest**	High	Medium	Low	Lowest
This Req. is Engineered From:	**UP-02**				
Justify why meeting SP-02-01 can contribute to the fulfilment of UP-02	The data set needs to use data from SPT-140 and SPT-100 Hall. The reason for this is because the SPT-140 is the thruster being used on the Psyche space craft, while the SPT-100 is the precursor to the SPT-140 thruster. Thus, this requirement is a constraint that restricts our data set to these thruster types.				
Traceability:	**Use cases cf.**	**N/A**			
	Test cases cf.	**TC-001, TC-034**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Table 4.37. System NonFunctional Requirements: SP-02-02

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SP-02-02**			**Type** / **Functional** / **Non-Functional**	
Creation:				**User** ☐ ☐	
Modification:				**System** ☐ ☒	
Description:	The system may include data pertaining to other Hall thruster types.			**Product (sub-type below)**	
				Usability Requirements	
Priority:	Highest	High	Medium	Low	✔ **Lowest**
This Req. is Engineered From:	**UP-02**				
Justify why meeting SP-02-02 can contribute to the fulfilment of UP-02	We must first focus on the two Hall thruster types being used, then if time is permitted after accomplishing the other requirements, we should expand to other thruster types to see if similar results can be obtained. Since it's not necessary to have these other types, this requirement is of the lowest priority.				
Traceability:	**Use cases cf.**	**N/A**			
	Test cases cf.	**TC-001**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Table 4.38. System NonFunctional Requirements: SO-01-01

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SO-01-01**			**Type** / **Functional** / **Non-Functional**	
Creation:				**User** ☐ ☐	
Modification:				**System** ☐ ☒	
Description:	The system shall run on Linux distributions.			**Organizational (sub-type below)**	
				Operational Requirements	
Priority:	Highest	✔ **High**	Medium	Low	Lowest
This Req. is Engineered From:	**UO-01**				
Justify why meeting SO-01-01 can contribute to the fulfilment of UO-01	Since the system needs to be run on any computer, the main OS types that need to be considered are Linux, Windows, and MacOS. This requirement deals with Linux.				
Traceability:	**Use cases cf.**	**N/A**			
	Test cases cf.	**TC-032**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Table 4.39. System NonFunctional Requirements: SO-01-02

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SO-01-02**				
Creation:					
Modification:					
Description:	The system shall run on Windows 10 operating systems.				
Priority:	Highest	✔ **High**	Medium	Low	Lowest
This Req. is Engineered From:	**UO-01**				
Justify why meeting SO-01-02 can contribute to the fulfilment of UO-01	Since the system needs to be run on any computer, the main OS types that need to be considered are Linux, Windows, and MacOS. This requirement deals with Windows.				
Traceability:	**Use cases cf.**	**N/A**			
	Test cases cf.	**TC-031**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Type	**Functional**	**Non-Functional**
User	☐	☐
System	☐	☒

Organizational (sub-type below)

Operational Requirements

Table 4.40. System NonFunctional Requirements: SO-01-03

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SO-01-03**				
Creation:					
Modification:					
Description:	The system shall run on computers with MacOS.				
Priority:	Highest	✔ **High**	Medium	Low	Lowest
This Req. is Engineered From:	**UO-01**				
Justify why meeting SO-01-03 can contribute to the fulfilment of UO-01	Since the system needs to be run on any computer, the main OS types that need to be considered are Linux, Windows, and MacOS. This requirement deals with MacOS.				
Traceability:	**Use cases cf.**	**N/A**			
	Test cases cf.	**TC-033**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Type	**Functional**	**Non-Functional**
User	☐	☐
System	☐	☒

Organizational (sub-type below)

Operational Requirements

Table 4.41. System NonFunctional Requirements: SE-01-01

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**				
Requirement #:	**SE-01-01**			**Type** / **Functional** / **Non-Functional**	
Creation:				**User**	☐	☐
Modification:				**System**	☐	☒
Description:	The system shall not be developed by a citizen of the People's Republic of China (PRC) as per Public Laws 112-10, Section 1340(a) and 112-55, Section 536.			**External (sub-type below)**	
				Legislative Requirements on Safety/Security	
Priority:	✔ **Highest**	High	Medium	Low	Lowest
This Req. is Engineered From:	**UE-01**				
Justify why meeting SE-01-01 can contribute to the fulfilment of UE-01	The user requirement is that the workers cannot be citizens of the PRC. This requirement restricts the system from being developed by those individuals, meeting the user requirement.				
Traceability:	**Use cases cf.**	**N/A**			
	Test cases cf.	**TC-030**			
Acknowledgment	***Generated from the CapStone Process Management System ©2015***				

Table 4.42. Mapping from user requirements to system requirements

Project Name: NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data			
User Requirements		**System Requirements**	
Req ID	**Description**	**Req ID**	**Description**
UE-01	Participants may not be citizens of the People's Republic of China (PRC), per Public Laws 112-10, Section 1340(a) and 112-55, Section 536.	SE-01-01	The system shall not be developed by a citizen of the People's Republic of China (PRC) as per Public Laws 112-10, Section 1340(a) and 112-55, Section 536.
UF-A	The team will devise a method to predict and/or correct for the sensitivity of the thruster output parameters (i.e. HET performance, operation, and plume parameters).	SF-A-01	The system shall predict the thrust in units of newtons the HET is outputting.
		SF-A-03	The system shall predict the thruster efficiency as a percentage.
		SF-A-04	The system shall predict the specific impulse in seconds the thruster is using.
		SF-A-05	The system shall implement a deep belief network (DBN) to predict different thruster output parameters.
		SF-A-06	The system shall utilize a Kalman filter to estimate the values of the output parameters.
UF-B	The team must include a report detailing the results of the machine learning analysis and data mining techniques including utilized data sets, any models/analyses generated by the analysis and techniques, and an evaluation of the model accuracy and predictive capability.	SF-B-01	The system shall record the results for running the machine learning model or data mining technique to local storage.
		SF-B-02	The system shall record the training data used to train a model for the report.
		SF-B-03	The system shall record the machine learning model type being used and the resulting analysis from the model for the report.
		SF-B-04	The system shall calculate the predictive capability of the model/technique to include in the report.
		SF-B-05	The system shall record the correlation results to be included in the report.
UF-C	The team should gather information and data from HET vacuum test facilities that tested Hall thrusters in order to build the data set for their system.	SF-C-01	The system shall read in the data to be used from a .CSV file.
		SF-C-03	The system shall remove a column of data if more than 60% of the column's values are missing.
		SF-C-04	The system may remove the nominal data columns.
		SF-C-05	The system shall replace missing values with a result from an appropriate mathematical equation.
		SF-C-06	The system shall remove any outliers within the data set.
		SF-C-07	The system shall normalize each column in the data set so all values are between 0 and 1.
			The system shall standardize each column in

		SF-C-08	the data set so all variables have a mean of 0 and standard deviation of 1.
UF-D	The team should utilize data mining techniques to discover previously undiscovered correlations within public data sets.	SF-D-01	The system shall perform Pearson correlation analysis on the data set.
		SF-D-02	The system shall generate a correlation matrix to show the correlations between all of the variables.
		SF-D-03	The system shall use Principal Component Analysis (PCA) to identify correlations within the data.
		SF-D-04	The system shall calculate Lasso Regression correlations within the data set.
		SF-D-05	The system shall generate scatter plots displaying each variable against every other variable.
UO-01	The user needs the machine learning model to be able to run on the computers in HET facilities so the software can be accessible by anybody who needs it.	SO-01-01	The system shall run on Linux distributions.
		SO-01-02	The system shall run on Windows 10 operating systems.
		SO-01-03	The system shall run on computers with MacOS.
UP-02	The client wants the system to utilize published data sets from HET Facilities that used SPT-140 and/or SPT-100 Hall thrusters and If time permits, the teams should try to incorporate data from other thruster models.	SP-02-01	The system shall utilize data from HET Facilities that utilized SPT-140 and/or SPT-100 Hall thrusters.
		SP-02-02	The system may include data pertaining to other Hall thruster types.

Table 4.1. Use Case Index Table

Project Name: NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data				
Use Case ID	**Use Case Name**	**Level**	**Author**	**Version**
UC-001	Hall Thruster Data Analysis Tool	Summary	Daniel Daniel Donley	1.8
UC-002	Clean the Data Set	Primary task	Daniel Daniel Donley	0.7
UC-003	Correlate the Data Set	Primary task	Daniel Daniel Donley	1.4
UC-004	Run Filter Script	Primary task	Daniel Daniel Donley	0.3
UC-005	Evaluate DBN Model	Subfunction	Daniel Daniel Donley	0.6
Acknowledgment: Generated from the CapStone process management system ©2015				

Table 4.2. Use Case UC-001

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Use Case ID:	UC-001
Use Case Name:	Hall Thruster Data Analysis Tool
User Goal:	The system generates useful information from the correlations and techniques.
Scope:	Hall Thruster Data Analysis Tool
Level:	Summary
Relevant User Reqs:	UF-A,UF-B,UF-C,UF-D
Relevant System Reqs:	SF-A-01,SF-A-03,SF-A-04,SF-A-05,SF-A-06,SF-B-01,SF-B-02,SF-B-03,SF-B-04,SF-B-05,SF-C-01,SF-C-03,SF-C-04,SF-C-05,SF-C-06,SF-C-07,SF-C-08,SF-D-01,SF-D-02,SF-D-03,SF-D-04,SF-D-05
Primary Actor:	Developer
Precondition:	System is executed or finishes a loop.
Minimal Guarantee:	The console application fails to launch.
Success Guarantee:	The console application executes and the developer is at the main menu.
Trigger:	The developer launches the system, or returns from one of the other use cases..
Success Scenario:	**Step** **Actions** **1** The developer launches the script to pre-process and analyze the raw data. **2** The system cleans the data and saves the cleaned set to the file directory. **3** The developer launches the script to generate correlation data. **4** The system takes the cleaned data and calculates requested correlation figures. **5** The developer launches the script to generate data from a filter. **6** The system records the generated data to the file directory. **7** The developer launches a model script. **8** The system trains amodel, then reocrds the model, weights, training data, and accuracy.
Extensions:	**Branching Scenarios**
1A	**Condition:** The system is unable to locate the data sets.
	Step **Actions** **1** The system informs the developer why the data sets couldn't be obtained. **2** The system shuts down.
3A	**Condition:** The cleaned data file cannot be found.
	Step **Actions** **1** The system informs the developer the file is not found. **2** The system shuts down.
5A	**Condition:** The cleaned data file cannot be found.

	Step **Actions** **1** The system infoms the developer the file is not found. **2** The system shuts down.
7A	**Condition:** The system cannot find the generated filter data.
	Step **Actions** **1** The system infoms the user that the filter data has not been generated yet. **2** The system shuts down.

Table 4.3. Use Case UC-002

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Use Case ID:	UC-002
Use Case Name:	Clean the Data Set
User Goal:	Remove unnecessary and unuseful data to improve the overall quality of the data set.
Scope:	Data
Level:	Primary task
Relevant User Reqs:	UF-C
Relevant System Reqs:	SF-C-01,SF-C-03,SF-C-04,SF-C-05,SF-C-06,SF-C-07,SF-C-08
Primary Actor:	Developer
Precondition:	The system launches.
Minimal Guarantee:	The data cannot be cleaned or is not fully cleaned.
Success Guarantee:	The data is clean and can be used for processing.
Trigger:	Hall Thruster Data Analysis Tool
Success Scenario:	**Step** **Actions** **1** The system retrieves the SPT csv file from the directory. **2** The system retrieves the facility csv file from the directory. **3** The system retrieves the meta data csv file from the directory. **4** The system joins the SPT and facility file data based on the meta data. **5** The system removes any nominal columns from the data set. **6** The system handles any outliers within the data. **7** The system handles any remaining missing values within the data. **8** The system saves the new cleaned data set to the file directory.
Extensions:	**Branching Scenarios**
1A	**Condition:** The SPT data file is not present.
	Step **Actions** **1** The system informs the developer the file is missing. **2** The system shuts down.
1B	**Condition:** The meta data file is not present.
	Step **Actions** **1** The system informs the developer the file is missing. **2** The system shuts down.
2A	**Condition:** The facility data file is not present.
	Step **Actions** **1** The system informs the developer the file is missing.

	2 The system shuts down.

Table 4.4. Use Case UC-003

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Use Case ID:	UC-003
Use Case Name:	Correlate the Data Set
User Goal:	Output information about the correlations in the data set.
Scope:	Data
Level:	Primary task
Relevant User Reqs:	UF-B,UF-D
Relevant System Reqs:	SF-B-05,SF-D-01,SF-D-02,SF-D-03,SF-D-04,SF-D-05
Primary Actor:	System
Precondition:	The system successfully obtains standardized and normalized data.
Minimal Guarantee:	No correlations are generated within the data.
Success Guarantee:	A correlation matrix is generated, scatter plots between variables are created, and correlations are recorded to a file.
Trigger:	Hall Thruster Data Analysis Tool
Success Scenario:	**Step** / **Actions** **1** The system creates scatter plots of each variable against every other variable. **2** The system calculates the Pearson correlations between each pair of variables. **3** The system creates a correlation matrix displaying each correlation that was calculated. **4** The system creates a plot of the feature influences from PCA. **5** The system calculates the Lasso and Ridge Regression correlations. **6** The system outputs all correlation data to the folder directory.
Extensions:	**Branching Scenarios**
Acknowledgment: Generated from the CapStone process management system ©2015	

Table 4.5. Use Case UC-004

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Use Case ID:	UC-004
Use Case Name:	Run Filter Script
User Goal:	Save data points from estimations and display current state.
Scope:	Learner
Level:	Primary task
Relevant User Reqs:	UF-A,UF-B
Relevant System Reqs:	SF-A-06,SF-B-01,SF-B-02
Primary Actor:	System
Precondition:	The developer launches a kalman filter script.
Minimal Guarantee:	The Kalman filter could not be set up.
Success Guarantee:	Data points from the estimations are saved and the developer can see values of the current state.
Trigger:	Hall Thruster Data Analysis Tool
Success Scenario:	**Step** / **Actions** **1** The system gets the cleaned data from storage. **2** The system normalizes the data. **3** The system performs PCA reduction on the data. **4** The system runs the data through the designated Kalman filter. **5** The system initializes the designated Kalman filter object. **6** The system calculates the covariance matrix. **7** The system The system calculates a noise uncertainty matrix. **8** The system creates a new DataFrame to hold the newly generated data records. **9** The system predicts the state estimate with the Kalman filter object. **10** The system updates the Kalman filter object. **11** The system appends the state estimate from the filter to the DataFrame. **12** The system returns the data frame to the developer.
Extensions:	**Branching Scenarios**
3A	**Condition:** The threshold is invalid.
	Step / **Actions** **1** The system wanrs the developer the threshold is invalid and PCA cannot be performed. **2** The system exits.
Acknowledgment: Generated from the CapStone process management system ©2015	

Table 4.6. Use Case UC-005

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Use Case ID:	UC-005
Use Case Name:	Evaluate DBN Model
User Goal:	Calculate the accuracy and predictive capability of the machine learning model
Scope:	Learner
Level:	Subfunction
Relevant User Reqs:	UF-A,UF-B
Relevant System Reqs:	SF-A-01,SF-A-03,SF-A-04,SF-A-05,SF-B-01,SF-B-03,SF-B-04
Primary Actor:	System
Precondition:	The system successfully trains the model
Minimal Guarantee:	The model is not evaluated and no report is generated.
Success Guarantee:	The model is evaluated and the report is generated.
Trigger:	The system evaluates the machine learning model.
Success Scenario:	**Step** **Actions** **1** The system constructs a DBN model. **2** The system compiles the model. **3** The system trains the model. **4** The system evaluates the model with sample training data and reocrds its accuracy. **5** The system evaluates the model with the orginial data set and reocrds its accuracy. **6** The system serializes the model and weights.
Extensions:	**Branching Scenarios**
3A	**Condition:** No data is available for training.
	Step **Actions** **1** The system informs the user that training data is needed from the filter scripts. **2** The system shuts down.
4A	**Condition:** The accuracy cannot be calculated.
	Step **Actions** **1** The system will record the accuracy as a null value.
Acknowledgment: Generated from the CapStone process management system ©2015	

Table 8.2.1. Test Suite TS-001: Data Cleaning

Test Case ID	Test Stage	Test Case Description	Tested
TC-001	Unit	The get_data function in data_input.py.	Yes
TC-002	Unit	The normalize function in data_input.py	Yes
TC-009	Unit	The remove_unfilled_columns function in data_input.py	Yes
TC-010	Unit	The remove_columns function in data_input.py	Yes
TC-011	Unit	The replace_missing_values function in data_input.py	Yes
TC-012	Unit	The remove_outliers function in data_input.py	Yes
TC-013	Unit	The standardize function in data_input.py	Yes
TC-017	Integration	Combine functions in data_input.py to convert CSV files to pre-processed DataFrames.	Yes

Table 8.2.2. Test Suite TS-002: Deep Belief Network (DBN)

Test Case ID	Test Stage	Test Case Description	Tested
TC-003	Unit	The construct_model function in dbn.py	Yes
TC-004	Unit	The compile_model function in dbn.py	Yes
TC-005	Unit	The fit_model function in dbn.py.	Yes
TC-006	Unit	The evaluate_model function in dbn.py.	Yes
TC-007	Unit	The serialize_model function in dbn.py.	Yes
TC-018	Integration	The DBN can accept a DataFrame and output a trained model for evaluation.	Yes
TC-039	Unit	Test model_evaluator.py	Yes

Table 8.2.3. Test Suite TS-003: Correlation Analysis

Test Case ID	Test Stage	Test Case Description	Tested
TC-008	Unit	Test the reduce function in pca.py	Yes
TC-014	Unit	Test the inverse_reduce function in pca.py	Yes
TC-015	Unit	Test generate_plots function of pca.py	Yes
TC-022	Unit	Test generate_scatter_plots of single_variate.py	Yes
TC-023	Unit	Test the generate_heatmap of the single_variate.py script.	Yes
TC-024	Unit	Test the lasso_ridge_regression.py script.	Yes

Table 8.2.4. Test Suite TS-004: Kalman Filter

Test Case ID	Test Stage	Test Case Description	Tested
TC-019	Unit	Test the get_state_transition_mtarix function from the filter.py script.	Yes
TC-020	Unit	The get_covariance_matrix function in kalmanFitler.py	Yes
TC-021	Unit	The get_noise_uncertainty function in kalmanFilter.py.	Yes
TC-025	Integration	Give the Kalman filter a DataFrame and have it save points and display values for the current state.	Yes

Table 8.2.5. Test Suite TS-005: Main Execution

Test Case ID	Test Stage	Test Case Description	Tested
TC-026	System	Test execution of correlation analysis scripts	Yes
TC-027	System	Test execution of preprocess_raw_data.py	Yes
TC-028	System	Text execution of the run_model script.	Yes
TC-029	System	Test the run_filter.py script	Yes
TC-035	Acceptance	Confirm data is preprocessed from running preprocess_raw_data.py.	Yes
TC-036	Acceptance	Test that correlation results are being generated.	Yes
TC-037	Acceptance	Test if Kalman fitler fully implemented.	Yes
TC-038	Acceptance	Test if a Neural Network is trained, evaluated, then saved.	Yes

Table 8.2.6. Test Suite TS-006: Nonfunctional Testing

Test Case ID	Test Stage	Test Case Description	Tested
TC-030	Acceptance	Test if developers are citizens of China.	Yes
TC-031	Acceptance	The system launches on a Windows platform.	Yes
TC-032	Acceptance	The system runs on a Linux platform.	Yes
TC-033	Acceptance	Test if the system can launch on macOS.	Yes
TC-034	Acceptance	Test if all data records in the raw SPT data are either SPT-140 or SPT-100 data records.	Yes

Table 8.2.7. Test Case TC-001

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-001: Data Cleaning	
Test Case ID	TC-001 (Unit Test)	
What To Test	The get_data function in data_input.py.	
Test Data Input	A csv file containing input data.	
Expected Result	A matrix with more than 0 rows and columns containing the data of the csv file.	
Traceability	**Relevant User Req.(s)**	UF-C,UP-02
	Relevant System Req.(s)	SF-C-01,SP-02-01,SP-02-02
	Relevant Use Case(s)	UC-001,UC-002
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.8. Test Case TC-002

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-001: Data Cleaning	
Test Case ID	TC-002 (Unit Test)	
What To Test	The normalize function in data_input.py	
Test Data Input	The DataFrame object containing a matrix of data from the input file.	
Expected Result	A DataFrame with every value being a positive value between 0 and 1.	
Traceability	**Relevant User Req.(s)**	UF-C
	Relevant System Req.(s)	SF-C-07,SF-D-02
	Relevant Use Case(s)	UC-001,UC-002
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.9. Test Case TC-003

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-002: Deep Belief Network (DBN)	
Test Case ID	TC-003 (Unit Test)	
What To Test	The construct_model function in dbn.py	
Test Data Input	A file containing sample Hall thruster data, uploaded and normalized.	
Expected Result	A machine learning model that can then be analyzed, deployed, or trained further.	
Traceability	**Relevant User Req.(s)**	UF-A,UF-B
	Relevant System Req.(s)	SF-A-05,SF-B-03
	Relevant Use Case(s)	UC-001,UC-005
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.10. Test Case TC-004

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-002: Deep Belief Network (DBN)	
Test Case ID	TC-004 (Unit Test)	
What To Test	The compile_model function in dbn.py	
Test Data Input	The constructed model from the construct_DBN procedure.	
Expected Result	A successfully compiled DBN without an error or exception occurring.	
Traceability	**Relevant User Req.(s)**	UF-A
	Relevant System Req.(s)	SF-A-05
	Relevant Use Case(s)	UC-001,UC-005
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.11. Test Case TC-005

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-002: Deep Belief Network (DBN)	
Test Case ID	TC-005 (Unit Test)	
What To Test	The fit_model function in dbn.py.	
Test Data Input	The constructed and compiled model, and data that can be used to train the model.	
Expected Result	A trained model that ran without throwing an exception.	
Traceability	**Relevant User Req.(s)**	UF-A,UF-B
	Relevant System Req.(s)	SF-A-05,SF-B-02
	Relevant Use Case(s)	UC-001,UC-005
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.12. Test Case TC-006

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-002: Deep Belief Network (DBN)	
Test Case ID	TC-006 (Unit Test)	
What To Test	The evaluate_model function in dbn.py.	
Test Data Input	The model that was trained as well as the training data.	
Expected Result	The accuracy of the model.	
Traceability	**Relevant User Req.(s)**	UF-A,UF-B
	Relevant System Req.(s)	SF-A-01,SF-A-03,SF-A-04,SF-A-05,SF-B-01,SF-B-04
	Relevant Use Case(s)	UC-001,UC-005
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.13. Test Case TC-007

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-002: Deep Belief Network (DBN)	
Test Case ID	TC-007 (Unit Test)	
What To Test	The serialize_model function in dbn.py.	
Test Data Input	The trained DBN model.	
Expected Result	A JSON file containing the values for the weights of the model.	
Traceability	**Relevant User Req.(s)**	UF-A,UF-B
	Relevant System Req.(s)	SF-A-05,SF-B-03
	Relevant Use Case(s)	UC-001,UC-005
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.14. Test Case TC-008

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-003: Correlation Analysis	
Test Case ID	TC-008 (Unit Test)	
What To Test	Test the reduce function in pca.py	
Test Data Input	A DataFrame containing standardized data, and a precision between 0 and 1.	
Expected Result	A DataFrame with no more columns than the original DataFrame.	
Traceability	**Relevant User Req.(s)**	UF-C
	Relevant System Req.(s)	SF-D-03
	Relevant Use Case(s)	UC-002
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.15. Test Case TC-009

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-001: Data Cleaning	
Test Case ID	TC-009 (Unit Test)	
What To Test	The remove_unfilled_columns function in data_input.py	
Test Data Input	A pandas DataFrame containing data.	
Expected Result	A DataFrame with all columns with more than 60% of the values missing removed.	
Traceability	**Relevant User Req.(s)**	UF-C
	Relevant System Req.(s)	SF-C-03
	Relevant Use Case(s)	UC-001,UC-002
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.16. Test Case TC-010

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-001: Data Cleaning	
Test Case ID	TC-010 (Unit Test)	
What To Test	The remove_columns function in data_input.py	
Test Data Input	A pandas DataFrame containing more than 0 rows and column and the meta data of that DataFrame and an array of numbers where each number is an index of the column to remove.	
Expected Result	A DataFrame where every specified column is removed.	
Traceability	**Relevant User Req.(s)**	UF-C
	Relevant System Req.(s)	SF-C-03,SF-C-04
	Relevant Use Case(s)	UC-001,UC-002
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.17. Test Case TC-011

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-001: Data Cleaning	
Test Case ID	TC-011 (Unit Test)	
What To Test	The replace_missing_values function in data_input.py	
Test Data Input	A DataFrame containing more than 0 rows and columns and missing values.	
Expected Result	A DataFrame that contains no missing values.	
Traceability	**Relevant User Req.(s)**	UF-C
	Relevant System Req.(s)	SF-C-05
	Relevant Use Case(s)	UC-001,UC-002
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.18. Test Case TC-012

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-001: Data Cleaning	
Test Case ID	TC-012 (Unit Test)	
What To Test	The remove_outliers function in data_input.py	
Test Data Input	A DataFrame containing more than 0 rows and columns and no missing values.	
Expected Result	A DataFrame where all values considered an outlier are handled.	
Traceability	**Relevant User Req.(s)**	UF-C
	Relevant System Req.(s)	SF-C-06
	Relevant Use Case(s)	UC-001,UC-002
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.19. Test Case TC-013

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-001: Data Cleaning	
Test Case ID	TC-013 (Unit Test)	
What To Test	The standardize function in data_input.py	
Test Data Input	A pandas DataFrame with no missing values and more than 0 rows and columns.	
Expected Result	A pandas DataFrame where each column has a mean of 0 and a standard deviation of 1.	
Traceability	**Relevant User Req.(s)**	UF-C
	Relevant System Req.(s)	SF-C-08
	Relevant Use Case(s)	UC-001,UC-002
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.20. Test Case TC-014

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-003: Correlation Analysis	
Test Case ID	TC-014 (Unit Test)	
What To Test	Test the inverse_reduce function in pca.py	
Test Data Input	A DataFrame containing test SPT data that's already been through PCA.	
Expected Result	A DataFrame containing the orginal data before PCA.	
Traceability	**Relevant User Req.(s)**	UF-B,UF-D
	Relevant System Req.(s)	SF-B-05,SF-D-03
	Relevant Use Case(s)	UC-003
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.21. Test Case TC-015

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-003: Correlation Analysis	
Test Case ID	TC-015 (Unit Test)	
What To Test	Test generate_plots function of pca.py	
Test Data Input	None	
Expected Result	Bar graphs totalling the number of components in the PCA model, each with the same number of points equal to the number of columns of the components.	
Traceability	**Relevant User Req.(s)**	UF-B,UF-D
	Relevant System Req.(s)	SF-B-05,SF-D-03
	Relevant Use Case(s)	UC-001,UC-003
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.22. Test Case TC-019

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-004: Kalman Filter	
Test Case ID	TC-019 (Unit Test)	
What To Test	Test the get_state_transition_mtarix function from the filter.py script.	
Test Data Input	A DataFrame containing the initial cleaned SPT data.	
Expected Result	A symmetric matrix where each cell is a percentage of the sum of the correlations of that row of the matrix.	
Traceability	**Relevant User Req.(s)**	UF-A
	Relevant System Req.(s)	SF-A-06
	Relevant Use Case(s)	UC-004
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.23. Test Case TC-020

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-004: Kalman Filter	
Test Case ID	TC-020 (Unit Test)	
What To Test	The get_covariance_matrix function in kalmanFitler.py	
Test Data Input	A DataFrame containing more than 0 rows and columns and no missing values.	
Expected Result	A symmetrical matrix where each element is the covariance of one column against another.	
Traceability	**Relevant User Req.(s)**	UF-A
	Relevant System Req.(s)	SF-A-06
	Relevant Use Case(s)	UC-001,UC-004
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.24. Test Case TC-021

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-004: Kalman Filter	
Test Case ID	TC-021 (Unit Test)	
What To Test	The get_noise_uncertainty function in kalmanFilter.py.	
Test Data Input	A DataFrame containing more than 0 rows and columns and no missing values.	
Expected Result	A square matrix of the size of of the number of columns where all values are random noise from a gaussian distribution.	
Traceability	**Relevant User Req.(s)**	UF-A
	Relevant System Req.(s)	SF-A-06
	Relevant Use Case(s)	UC-001,UC-004
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.25. Test Case TC-022

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-003: Correlation Analysis	
Test Case ID	TC-022 (Unit Test)	
What To Test	Test generate_scatter_plots of single_variate.py	
Test Data Input	A DataFrame containing test SPT data.	
Expected Result	A number of scatter plots equal to n^2/2 where n is the number of variables all saved to the directory, each showing a scatter plot between the variables.	
Traceability	**Relevant User Req.(s)**	UF-B,UF-D
	Relevant System Req.(s)	SF-D-01,SF-D-05
	Relevant Use Case(s)	UC-001,UC-003
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.26. Test Case TC-023

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-003: Correlation Analysis	
Test Case ID	TC-023 (Unit Test)	
What To Test	Test the generate_heatmap of the single_variate.py script.	
Test Data Input	DataFrame containing test SPT data.	
Expected Result	A heatmap with the variables on the x and y-axes, where each cell is the correlation between those two variables.	
Traceability	**Relevant User Req.(s)**	UF-B,UF-D
	Relevant System Req.(s)	SF-B-01,SF-B-05,SF-D-01,SF-D-02
	Relevant Use Case(s)	UC-001,UC-003
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.27. Test Case TC-024

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-003: Correlation Analysis	
Test Case ID	TC-024 (Unit Test)	
What To Test	Test the lasso_ridge_regression.py script.	
Test Data Input	A DataFrame containg test SPT data.	
Expected Result	Two plots, one of the Lasso regression correlations, another of the Ridge regression correlations.	
Traceability	**Relevant User Req.(s)**	UF-B,UF-D
	Relevant System Req.(s)	SF-B-01,SF-B-05,SF-D-04
	Relevant Use Case(s)	UC-001,UC-003
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.28. Test Case TC-039

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-002: Deep Belief Network (DBN)	
Test Case ID	TC-039 (Unit Test)	
What To Test	Test model_evaluator.py	
Test Data Input	A machine learning model json file, weights, and a sample DataFrame	
Expected Result	An evaluation with the same dimensional complexity as the DataFrame from the model.	
Traceability	**Relevant User Req.(s)**	UF-A
	Relevant System Req.(s)	SF-A-05,SF-B-01,SF-B-03
	Relevant Use Case(s)	UC-001
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.29. Test Case TC-017

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-001: Data Cleaning	
Test Case ID	TC-017 (Integration Test)	
What To Test	Combine functions in data_input.py to convert CSV files to pre-processed DataFrames.	
Test Data Input	CSV files containing the SPT data.	
Expected Result	A standardized DataFrame where each column has a mean of 0 and standard deviation of 1 and a normalized DataFrame where all values are between 0 and 1.	
Traceability	**Relevant User Req.(s)**	UF-C
	Relevant System Req.(s)	SF-C-01,SF-C-03,SF-C-04,SF-C-05,SF-C-06,SF-C-07,SF-C-08
	Relevant Use Case(s)	UC-001,UC-002
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.30. Test Case TC-018

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-002: Deep Belief Network (DBN)	
Test Case ID	TC-018 (Integration Test)	
What To Test	The DBN can accept a DataFrame and output a trained model for evaluation.	
Test Data Input	A pandas DataFrame containing more than 0 rows and columns and no missing values.	
Expected Result	A JSON file containing the model parameters and weights.	
Traceability	**Relevant User Req.(s)**	UF-A,UF-B
	Relevant System Req.(s)	SF-A-01,SF-A-03,SF-A-04,SF-A-05,SF-B-01,SF-B-02,SF-B-03,SF-B-04
	Relevant Use Case(s)	UC-001,UC-005
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.31. Test Case TC-025

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-004: Kalman Filter	
Test Case ID	TC-025 (Integration Test)	
What To Test	Give the Kalman filter a DataFrame and have it save points and display values for the current state.	
Test Data Input	Standardized and normalized DataFrames of the SPT data.	
Expected Result	Estimated data points are saved in a spreadsheet, and the current state values are displayed.	
Traceability	**Relevant User Req.(s)**	UF-A,UF-B
	Relevant System Req.(s)	SF-A-01,SF-A-03,SF-A-04,SF-A-06,SF-B-01,SF-B-04
	Relevant Use Case(s)	UC-001,UC-004
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.32. Test Case TC-026

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-005: Main Execution	
Test Case ID	TC-026 (System Test)	
What To Test	Test execution of correlation analysis scripts	
Test Data Input	DataFrame containing test SPT data.	
Expected Result	Outputs all associated correlation results (scatter plot, heatmap, Ridge, Lasso, PCA)	
Traceability	**Relevant User Req.(s)**	UF-B,UF-D
	Relevant System Req.(s)	SF-B-01,SF-B-05,SF-D-01,SF-D-02,SF-D-03,SF-D-04,SF-D-05
	Relevant Use Case(s)	UC-001,UC-003
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.33. Test Case TC-027

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-005: Main Execution	
Test Case ID	TC-027 (System Test)	
What To Test	Test execution of preprocess_raw_data.py	
Test Data Input	A DataFrame containing unclean SPT data.	
Expected Result	A DataFrame with no outliers, the same number of columns or less (but greater than 0), standardized, and normalized, then saved to the file directory.	
Traceability	**Relevant User Req.(s)**	UF-C
	Relevant System Req.(s)	SF-C-01,SF-C-03,SF-C-04,SF-C-05,SF-C-06,SF-C-07,SF-C-08
	Relevant Use Case(s)	UC-001,UC-002
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.34. Test Case TC-028

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-005: Main Execution	
Test Case ID	TC-028 (System Test)	
What To Test	Text execution of the run_model script.	
Test Data Input	A DataFrame containing SPT data.	
Expected Result	A DBN model file, weights file, and training data all saved to the file directory.	
Traceability	**Relevant User Req.(s)**	UF-A
	Relevant System Req.(s)	SF-A-01,SF-A-03,SF-A-04,SF-A-05,SF-A-06
	Relevant Use Case(s)	UC-001,UC-005
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.35. Test Case TC-029

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-005: Main Execution	
Test Case ID	TC-029 (System Test)	
What To Test	Test the run_filter.py script	
Test Data Input	A DataFrame containing SPT data.	
Expected Result	A DataFrame containing generated SPT data that totals n^2/2 records.	
Traceability	**Relevant User Req.(s)**	UF-A
	Relevant System Req.(s)	SF-A-06
	Relevant Use Case(s)	UC-001,UC-004
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.36. Test Case TC-030

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-006: Nonfunctional Testing	
Test Case ID	TC-030 (Acceptance Test)	
What To Test	Test if developers are citizens of China.	
Test Data Input	The citizenship of each developer.	
Expected Result	All developers are permitted to work on the project.	
Traceability	**Relevant User Req.(s)**	UE-01
	Relevant System Req.(s)	SE-01-01
	Relevant Use Case(s)	UC-001
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.37. Test Case TC-031

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-006: Nonfunctional Testing	
Test Case ID	TC-031 (Acceptance Test)	
What To Test	The system launches on a Windows platform.	
Test Data Input	None, the system launches.	
Expected Result	The scripts run without issue.	
Traceability	**Relevant User Req.(s)**	UO-01
	Relevant System Req.(s)	SO-01-02
	Relevant Use Case(s)	UC-001
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.38. Test Case TC-032

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-006: Nonfunctional Testing	
Test Case ID	TC-032 (Acceptance Test)	
What To Test	The system runs on a Linux platform.	
Test Data Input	None	
Expected Result	All scripts can be run.	
Traceability	**Relevant User Req.(s)**	UO-01
	Relevant System Req.(s)	SO-01-01
	Relevant Use Case(s)	UC-001
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.39. Test Case TC-033

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-006: Nonfunctional Testing	
Test Case ID	TC-033 (Acceptance Test)	
What To Test	Test if the system can launch on macOS.	
Test Data Input	None	
Expected Result	All scripts can run	
Traceability	**Relevant User Req.(s)**	UO-01
	Relevant System Req.(s)	SO-01-03
	Relevant Use Case(s)	UC-001
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.40. Test Case TC-034

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-006: Nonfunctional Testing	
Test Case ID	TC-034 (Acceptance Test)	
What To Test	Test if all data records in the raw SPT data are either SPT-140 or SPT-100 data records.	
Test Data Input	A DataFrame containing the raw SPT data.	
Expected Result	For all records, the filed 'ModelType' has either the value 'SPT-140' or 'SPT-100'	
Traceability	**Relevant User Req.(s)**	UP-02
	Relevant System Req.(s)	SP-02-01
	Relevant Use Case(s)	
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.41. Test Case TC-035

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-005: Main Execution	
Test Case ID	TC-035 (Acceptance Test)	
What To Test	Confirm data is preprocessed from running preprocess_raw_data.py.	
Test Data Input	DataFrames containing raw facility and SPT data.	
Expected Result	A DataFrame containing cleaned data and saved to the file directory.	
Traceability	**Relevant User Req.(s)**	UF-C
	Relevant System Req.(s)	SF-C-01,SF-C-03,SF-C-04,SF-C-05,SF-C-06,SF-C-07,SF-C-08
	Relevant Use Case(s)	UC-001,UC-002
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.42. Test Case TC-036

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-005: Main Execution	
Test Case ID	TC-036 (Acceptance Test)	
What To Test	Test that correlation results are being generated.	
Test Data Input	A DataFrame containing SPT data	
Expected Result	Graphs containing scatter plots, a heatmap, Lasso correlations, Ridge correlations, and PCA correlations	
Traceability	**Relevant User Req.(s)**	UF-B,UF-D
	Relevant System Req.(s)	SF-B-01,SF-B-05,SF-D-01,SF-D-02,SF-D-03,SF-D-04,SF-D-05
	Relevant Use Case(s)	UC-001,UC-003
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.43. Test Case TC-037

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-005: Main Execution	
Test Case ID	TC-037 (Acceptance Test)	
What To Test	Test if Kalman fitler fully implemented.	
Test Data Input	A DataFrame containing SPT data	
Expected Result	A DataFrame with n^2/2 records, where n is the number of records in the original DataFrame.	
Traceability	**Relevant User Req.(s)**	UF-A,UF-B
	Relevant System Req.(s)	SF-A-06,SF-B-02
	Relevant Use Case(s)	UC-001,UC-004
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.44. Test Case TC-038

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Suite	TS-005: Main Execution	
Test Case ID	TC-038 (Acceptance Test)	
What To Test	Test if a Neural Network is trained, evaluated, then saved.	
Test Data Input	A DataFrame of SPT data.	
Expected Result	A Neural Network model file, weights file, training data, and evaluation data all saved to the directory.	
Traceability	**Relevant User Req.(s)**	UF-A,UF-B
	Relevant System Req.(s)	SF-A-01,SF-A-03,SF-A-04,SF-A-05,SF-B-01,SF-B-02,SF-B-03,SF-B-04
	Relevant Use Case(s)	UC-001,UC-005
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.3.1. Execution Report of Test Case TC-001

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-001
Testing Tools Used:	spyder-unittest
Testing Type:	Agile (automated) testing
Execution Steps:	1 Navigate to the unit test pane
	2 Click the gear in the top-right and select 'Configure'
	3 Select unittest for the testing framework
	4 Select the 'Data' folder under Data
	5 Click OK
	6 Click 'Run Tests'

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	11/06/2019	Exception during execution	Fail	KeyError: 0	11/06/2019 by Dan Donley
2	Dan Donley	11/06/2019	Data successfully opened	Pass		
3	Dan Donley	01/21/2020	Data successfully opened	Pass		
4	Dan Donley	02/17/2020	Data successfully opened	Pass		
5	Dan Donley	03/05/2020	Data successfully opened	Pass		
6	Dan Donley	04/05/2020	Data successfully opened	Pass		
7	Dan Donley	04/06/2020	Data file not opened	Fail	Error: Can only compare identically-labeled DataFrames	04/06/2020 by Dan Donley
8	Dan Donley	04/06/2020	Data file not opened	Fail	ValueError: The truth value of the DataFrame is ambiguous	04/06/2020 by Dan Donley
9	Dan Donley	04/06/2020	Data file matches and can be opened	Pass		

Execution Summary:	The function works properly. Data can now be accepted into the system for processing.

Table 8.3.2. Execution Report of Test Case TC-002

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-002
Testing Tools Used:	spyder-unittest
Testing Type:	Agile (automated) testing
Execution Steps:	1 Navigate to the unit testing pane in spyder
	2 Click the gear in the top-right and click 'Configure'
	3 Select unit test as the test framework
	4 Select the 'Data' folder.
	5 Click OK
	6 Click 'Run Tests'

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	11/06/2019	Failed during execution	Fail	Path to "data_Input_System.py" could not be found.	11/08/2019 by Dan Donley
2	Dan Donley	11/08/2019	Failed during execution	Fail	Key Error: 0	11/09/2019 by Dan Donley
3	Dan Donley	11/09/2019	The data was normalized properly.	Pass		
4	Dan Donley	01/21/2020	Failed during execution	Fail	TypeError: loop of ufunc does not support argument 1053 of type str which has no callable conjugate	01/26/2020 by Dan Donley
5	Dan Donley	01/26/2020	Failed during execution	Fail	AttributeError: 'float' object has no attribute 'sqrt'	02/12/2020 by Dan Donley
6	Dan Donley	02/12/2020	All values normalized to between zero and one.	Pass		
7	Dan Donley	02/22/2020	All values normalized to between zero and one.	Pass		
8	Dan Donley	03/05/2020	All values normalized to between zero and one.	Pass		
9	Dan Donley	04/05/2020	All values normalized to between zero and one.	Pass		
10	Dan Donley	04/06/2020	Frame not normalized	Fail	AssertionError: value not within range of -1 to 1	04/06/2020 by Dan Donley

11	Dan Donley	04/06/2020	All values normalized to between zero and one	Pass		

Execution Summary: The function now normalizes all values by column to values between 0 and 1.

Table 8.3.3. Execution Report of Test Case TC-003

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-003
Testing Tools Used:	spyder-unittest
Testing Type:	Agile (automated) testing
Execution Steps:	1 Navigate to the unit testing pane in Spyder
	2 Click the gear in the top-right and click 'Configure'
	3 Select unittest for the testing framework
	4 Choose the 'Correlation Learner' folder under the 'Learner' directory
	5 Click OK
	6 Click 'Run Tests'

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	11/13/2019	Model not constructed	Fail	Spyder crashed	11/13/2019 by Dan Donley
2	Dan Donley	11/13/2019	Model successfully constructed	Pass		
3	Alec Dady	02/22/2020	Model successfully constructed	Pass		
4	Dan Donley	03/05/2020	Model successfully constructed	Pass		
5	Alec Dady	03/09/2020	Model successfully constructed	Pass		
6	Alec Dady	03/12/2020	Model successfully constructed	Pass		
7	Alec Dady	03/17/2020	Model successfully constructed	Pass		
8	Alec Dady	04/05/2020	Model successfully constructed	Pass		

Execution Summary:	The function works and the model can construct.

Table 8.3.4. Execution Report of Test Case TC-004

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-004
Testing Tools Used:	spyder-unittest
Testing Type:	Agile (automated) testing

Execution Steps:

1	Navigate to the unit testing pane in Spyder
2	Click the gear in the top-right and select 'Configure'
3	Choose unittest for the testing framework
4	Choose the 'Correlation Learner' folder under the 'Learner' directory
5	Click OK
6	Click 'Run Tests'

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	11/13/2019	Model not compiled	Fail	Spyder crashed	11/13/2019 by Dan Donley
2	Dan Donley	11/13/2019	Model not compiled	Fail	AttributeError: module 'tensorflow' has no attribute 'get_default_graph'	11/14/2019 by Alec Dady
3	Alec Dady	11/14/2019	Model compiled	Pass		
4	Alec Dady	02/22/2020	Model compiled	Pass		
5	Dan Donley	03/05/2020	Model compiled	Pass		
6	Alec Dady	03/09/2020	Model compiled	Pass		
7	Alec Dady	03/12/2020	Model compiled	Pass		
8	Alec Dady	03/17/2020	Model compiled	Pass		
9	Alec Dady	04/05/2020	Model compiled	Pass		

Execution Summary: The function works properly and the model compiles

Table 8.3.5. Execution Report of Test Case TC-005

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-005
Testing Tools Used:	spyder-unittest
Testing Type:	Agile (automated) testing

Execution Steps:

1	Navigate to the unit testing pane in Spyder
2	Click the gear in the top-right and click 'Configure'
3	Select unittest for the testing framework
4	Choose the 'Correlation Learner' folder under the 'Learner' directory
5	Click OK
6	Click 'Run Tests'

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	11/13/2019	Model not trained	Fail	Spyder crashed	11/13/2019 by Dan Donley
2	Dan Donley	11/13/2019	Model not trained	Fail	AttributeError: module 'tensorflow' has no attribute 'get_default_graph'	11/14/2019 by Alec Dady
3	Alec Dady	11/14/2019	Model successfully trained	Pass		
4	Alec Dady	02/22/2020	Model successfully trained	Pass		
5	Dan Donley	03/05/2020	Model successfully trained	Pass		
6	Alec Dady	03/09/2020	Model successfully trained	Pass		
7	Alec Dady	03/12/2020	Model successfully trained	Pass		
8	Alec Dady	03/17/2020	Model successfully trained	Pass		
9	Alec Dady	04/05/2020	Model successfully trained	Pass		

Execution Summary:	The function works properly and can train models, given there is data

Table 8.3.6. Execution Report of Test Case TC-006

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-006
Testing Tools Used:	spyder-unittest
Testing Type:	Agile (automated) testing

Execution Steps:

1	Navigate to the 'Unit testing' pane in Spyder
2	Click the gear in the top-right corner of the pane and select 'Configure'
3	Select unittest for 'Testing Framework'
4	Select the 'Learner' folder with the script and corresponding test script
5	Click OK
6	Click 'Run Tests'

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	11/13/2019	Model's accuracy not evaluated	Fail	Spyder crashed	11/13/2019 by Dan Donley
2	Dan Donley	11/13/2019	Model's accuracy not evaluated	Fail	AttributeError: module 'tensorflow' has no attribute 'get_default_graph'	11/14/2019 by Alec Dady
3	Alec Dady	11/14/2019	Model's accuracy evaluated and recorded	Pass		
4	Alec Dady	02/22/2020	Model's accuracy evaluated and recorded	Pass		
5	Dan Donley	03/05/2020	Model's accuracy evaluated and recorded	Pass		
6	Alec Dady	03/09/2020	Model's accuracy evaluated and recorded	Pass		
7	Alec Dady	03/12/2020	Model's accuracy evaluated and recorded	Pass		
8	Alec Dady	03/17/2020	Model's accuracy evaluated and recorded	Pass		
9	Alec Dady	04/05/2020	Model's accuracy evaluated and recorded	Pass		

Execution Summary:	The function works as intended and can record the accuracy of a model.

Table 8.3.7. Execution Report of Test Case TC-007

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-007
Testing Tools Used:	spyder-unittest
Testing Type:	Agile (automated) testing

Execution Steps:		
	1	Nagivate to the unit test pane in Spyder
	2	Click the gear in the top-right and select 'Configure'
	3	Enter 'unittest' for the testing framework
	4	Select the 'Learner' folder to load the correct script and test script
	5	Click OK
	6	Click 'Run Tests'

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	11/13/2019	JSON file not created	Fail	Spyder crashed	11/13/2019 by Dan Donley
2	Dan Donley	11/13/2019	JSON file not created	Fail	AttributeError: module 'tensorflow' has no attribute 'get_default_graph'	11/14/2019 by Alec Dady
3	Alec Dady	11/14/2019	JSON file created in local storage	Pass		
4	Alec Dady	02/22/2020	JSON file created in local storage	Pass		
5	Dan Donley	03/05/2020	JSON file created in local storage	Pass		
6	Alec Dady	03/09/2020	JSON file created in local storage	Pass		
7	Alec Dady	03/12/2020	JSON file created in local storage	Pass		
8	Alec Dady	03/17/2020	JSON file created in local storage	Pass		
9	Alec Dady	04/05/2020	JSON file created in local storage	Pass		

Execution Summary:	The function can serialize the model and weights for use later.

Table 8.3.8. Execution Report of Test Case TC-008

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-008
Testing Tools Used:	spyder-unittest
Testing Type:	Agile (automated) testing

Execution Steps:

1	Select 'Unit testing'
2	Select the gear and choose 'Configure'
3	Choose 'unittest' for the test framework
4	Choose the folder with the test scripts
5	Click 'OK'
6	Click 'Run tests'

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	03/15/2020	No PCA data generated	Fail	Not Initialized	03/15/2020 by Dan Donley
2	Dan Donley	03/15/2020	No PCA data generated	Fail	ValueError: shapes (6,5) and (1,1) not aligned: 5 (dim 1) != 1 (dim 0)	03/15/2020 by Dan Donley
3	Dan Donley	03/15/2020	No PCA data generated	Fail	IndexError: index 1 is out of bounds for axis 0 with size 1	03/15/2020 by Dan Donley
4	Dan Donley	03/15/2020	No PCA data generated	Fail	IndexError: index 1 is out of bounds for axis 0 with size 1	03/15/2020 by Dan Donley
5	Dan Donley	03/15/2020	PCA reduced successfully	Pass		
6	Dan Donley	04/05/2020	PCA reduced successfully	Pass		

Execution Summary: PCA reduction can now be applied to the data set.

Table 8.3.9. Execution Report of Test Case TC-009

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-009
Testing Tools Used:	spyder-unittest
Testing Type:	Agile (automated) testing
Execution Steps:	1 Navigate to the Unit Testing pane on the right side. 2 Click the gear and select 'Configure' 3 For the framework, select unittest 4 For the directory, select the 'Data' folder 5 Click OK 6 Click 'Run tests'

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	02/17/2020	Tests threw an error	Fail	Not Initialized	02/17/2020 by Dan Donley
2	Dan Donley	02/17/2020	Threw error during execution	Fail	Key Error: 0	02/17/2020 by Dan Donley
3	Dan Donley	02/17/2020	Columns with 60% of data missing successfully removed.	Pass		
4	Dan Donley	02/22/2020	Columns with 60% of data missing successfully removed.	Pass		
5	Dan Donley	03/05/2020	Columns with 60% of data missing successfully removed.	Pass		
6	Dan Donley	04/05/2020	Columns with 60% of data missing successfully removed.	Pass		
7	Dan Donley	04/06/2020	Columns with 60% of data missing successfully removed.	Pass		

Execution Summary:	Columns with more than 60% of the data missing are removed from the data set.

Table 8.3.10. Execution Report of Test Case TC-010

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-010
Testing Tools Used:	spyder-unittest
Testing Type:	Agile (automated) testing
Execution Steps:	1 Navigate to the Unit Testing pane on the right side.
	2 Click the gear and select 'Configure'
	3 For the framework, select unittest
	4 For the directory, select the 'Data' folder
	5 Click OK
	6 Click 'Run tests'

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	02/17/2020	Tests failed to remove nominal data	Fail	Not Initialized	02/17/2020 by Dan Donley
2	Dan Donley	02/17/2020	Nominal data columns successfully removed from data set	Pass		
3	Dan Donley	02/22/2020	Nominal data columns successfully removed from data set	Pass		
4	Dan Donley	03/05/2020	Nominal data columns successfully removed from data set	Pass		
5	Dan Donley	04/05/2020	Nominal data columns successfully removed from data set	Pass		
6	Dan Donley	04/06/2020	Nominal data columns successfully removed from data set	Pass		

Execution Summary:	The data set does not contain any nominal data columns.

Table 8.3.11. Execution Report of Test Case TC-011

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-011
Testing Tools Used:	spyder-unittest
Testing Type:	Agile (automated) testing

Execution Steps:

1	Navigate to the Unit Testing pane on the right side.
2	Click the gear and select 'Configure'
3	For the framework, select unittest
4	For the directory, select the 'Data' folder
5	Click OK
6	Click 'Run tests'

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	02/17/2020	Values not replaced	Fail	Not Initialized	02/17/2020 by Dan Donley
2	Dan Donley	02/17/2020	Values not replaced	Fail	Cannot convert string to float	02/17/2020 by Dan Donley
3	Dan Donley	02/17/2020	All missing values replaced with appropriate number	Pass		
4	Dan Donley	02/22/2020	All missing values replaced with appropriate number	Pass		
5	Dan Donley	03/05/2020	All missing values replaced with appropriate number	Pass		
6	Dan Donley	04/05/2020	All missing values replaced with appropriate number	Pass		
7	Dan Donley	04/06/2020	All missing values replaced with appropriate number	Pass		

Execution Summary:	The values within the data set that were missing are all replaced with appropriate values based on the data type.

Table 8.3.12. Execution Report of Test Case TC-012

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-012
Testing Tools Used:	spyder-unittest
Testing Type:	Agile (automated) testing

Execution Steps:

1	Navigate to the Unit Testing pane on the right side.
2	Click the gear and select 'Configure'
3	For the framework, select unittest
4	For the directory, select the 'Data' folder
5	Click OK
6	Click 'Run tests'

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	02/17/2020	No outliers handled	Fail	Not initialized	02/17/2020 by Dan Donley
2	Dan Donley	02/17/2020	No outliers handled	Fail	Cannot calculate distribution of string	02/17/2020 by Dan Donley
3	Dan Donley	02/17/2020	3 outliers handled in test data set	Pass		
4	Dan Donley	02/22/2020	Outliers were sucessfully handled in data set	Pass		
5	Dan Donley	03/05/2020	Outliers were sucessfully handled in data set	Pass		
6	Dan Donley	04/05/2020	Outliers were sucessfully handled in data set	Pass		
7	Dan Donley	04/06/2020	Outliers were sucessfully handled in data set	Pass		

Execution Summary:	Outliers are no longer present within the data set and are replaced with appropriate values.

Table 8.3.13. Execution Report of Test Case TC-013

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-013
Testing Tools Used:	spyder-unittest
Testing Type:	Agile (automated) testing

Execution Steps:

1	Navigate to the Unit Testing pane on the right side.
2	Click the gear and select 'Configure'
3	For the framework, select unittest
4	For the directory, select the 'Data' folder
5	Click OK
6	Click 'Run tests'

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	02/17/2020	Values unable to be standardized	Fail	Not initialized	02/17/2020 by Dan Donley
2	Dan Donley	02/17/2020	Values not sucessfully standardized	Fail	Values do not match expected output	02/17/2020 by Dan Donley
3	Dan Donley	02/19/2020	Values standardized to a mean of 0 and standard deviation of 1	Pass		
4	Dan Donley	02/22/2020	Values standardized to a mean of 0 and standard deviation of 1	Pass		
5	Dan Donley	03/05/2020	Values standardized to a mean of 0 and standard deviation of 1	Pass		
6	Dan Donley	04/05/2020	Values standardized to a mean of 0 and standard deviation of 1	Pass		
7	Dan Donley	04/06/2020	Error on execution	Fail	AttirbuteError: 'numpy.ndarray' object has no attribute 'values'	04/06/2020 by Dan Donley
8	Dan Donley	04/06/2020	Values standardized to a mean of 0 and standard deviation of 1	Pass		

Execution Summary:	The data set can now be standardized so each column has a mean of 0 and standard deviation of 1 to elminate different scales.

Table 8.3.14. Execution Report of Test Case TC-014

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-014
Testing Tools Used:	spyder-unittest
Testing Type:	Agile (automated) testing

Execution Steps:

1	Select 'Unit testing' on the right side
2	Select the gear and choose 'Configure'
3	Select 'unittest' for testing framework
4	Choose the folder with the test scripts
5	Choose 'OK'
6	Click 'Run tests'

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	03/15/2020	Test data not reversed	Fail	Not initialized	03/15/2020 by Dan Donley
2	Dan Donley	03/15/2020	Test data not reversed	Fail	AttributeError: 'PCA' object has no attribute 'components_'	03/15/2020 by Dan Donley
3	Dan Donley	03/15/2020	Assertion thrown	Fail	AssertionError: Inversion does not match expected result	03/15/2020 by Dan Donley
4	Dan Donley	03/15/2020	System can reverse PCA given a model and a frame	Pass		
5	Dan Donley	03/15/2020	System can reverse PCA given a model and a frame	Pass		

Execution Summary:	PCA inversion now operational

Table 8.3.15. Execution Report of Test Case TC-015

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-015
Testing Tools Used:	spyder-unittest
Testing Type:	Agile (automated) testing
Execution Steps:	1 Select 'Unite testing' on the right
	2 Select the gear and choose 'Configure'
	3 Select 'unittest' for the testing framework
	4 Select the folder with the testing scripts
	5 Click 'OK'
	6 Click 'Run tests'

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	03/15/2020	No plots created	Fail	Not initialized	04/09/2020 by Dan Donley
2	Dan Donley	04/09/2020	Assertion thrown	Fail	AssertionError: Number of graphs in folder incorrect	04/09/2020 by Dan Donley
3	Dan Donley	04/09/2020	Graphs generated successfully	Pass		

Execution Summary:	Graphs can be created from PCA components

Table 8.3.16. Execution Report of Test Case TC-019

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-019
Testing Tools Used:	spyder-unittest
Testing Type:	Agile (automated) testing

Execution Steps:

1	Select 'Unit testing' on the right
2	Select the gear and click 'Configure'
3	Choose 'unittest' for the testing framework
4	Choose the folder with the testing scripts
5	Click 'OK'
6	Click 'Run tests'

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	02/24/2020	Transition matrix not created	Fail	Not initialized	02/24/2020 by Dan Donley
2	Dan Donley	02/24/2020	Failed during execution	Fail	Cannot perform operation on value of type str	02/24/2020 by Dan Donley
3	Dan Donley	02/24/2020	Failed during execution	Fail	ValueError: The truth value of an array with more than one elemtn is ambiguous.	02/24/2020 by Dan Donley
4	Dan Donley	02/24/2020	Assertion thrown	Fail	AssertionError: state transition matrix not correctly calculated	02/24/2020 by Dan Donley
5	Dan Donley	02/24/2020	Matrix calculated	Pass		
6	Dan Donley	03/05/2020	Matrix calculated	Pass		
7	Dan Donley	04/05/2020	Matrix calculated	Pass		

Execution Summary:	State transition matrix algorithm accurate and correct

Table 8.3.17. Execution Report of Test Case TC-020

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-020
Testing Tools Used:	spyder-unittest
Testing Type:	Agile (automated) testing
Execution Steps:	1 Navigate to the Unit Testing pane on the right side. 2 Click the gear and select 'Configure' 3 For the framework, select unittest 4 For the directory, select the 'Learner' folder 5 Click OK 6 Click 'Run tests'

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	02/09/2020	Covariance matrix not calculated	Fail	Not initialized	02/10/2020 by Dan Donley
2	Dan Donley	02/10/2020	Threw assertion during test case	Fail	Size of covariance matrix does not align with expected size	02/10/2020 by Dan Donley
3	Dan Donley	02/10/2020	Threw assertion during test case	Fail	Covariance matrix does not match with expected values	02/11/2020 by Dan Donley
4	Dan Donley	02/11/2020	Covariance matrix of data set sucessfully calculated and is the right size	Pass		
5	Dan Donley	02/22/2020	Covariance matrix of data set sucessfully calculated and is the right size	Pass		
6	Dan Donley	02/24/2020	Error during execution	Fail	ValueError: The truth value of an array with more than one element is ambiguous.	02/24/2020 by Dan Donley
7	Dan Donley	02/24/2020	Covariance matrix of data set sucessfully calculated and is the right size	Pass		
8	Dan Donley	03/05/2020	Covariance matrix of data set sucessfully calculated and is the right size	Pass		
9	Dan Donley	04/05/2020	Covariance matrix of data set sucessfully calculated and is the right size	Pass		

Execution Summary:	The covariance matrix of a given data set is sucessfully calculated and is a symmetrical matrix of the right size based on the data set.

Table 8.3.18. Execution Report of Test Case TC-021

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-021
Testing Tools Used:	spyder-unittest
Testing Type:	Agile (automated) testing

Execution Steps:

1	Navigate to the Unit Testing pane on the right side.
2	Click the gear and select 'Configure'
3	For the framework, select unittest
4	For the directory, select the 'Learner' folder
5	Click OK
6	Click 'Run tests'

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	02/09/2020	Noise uncertinaty not obtained	Fail	Not initialized	02/09/2020 by Dan Donley
2	Dan Donley	02/09/2020	Noice uncertainty calculated without throwing error	Pass		
3	Dan Donley	02/22/2020	Noice uncertainty calculated without throwing error	Pass		
4	Dan Donley	02/24/2020	Noice uncertainty calculated without throwing error	Pass		
5	Dan Donley	03/05/2020	Noice uncertainty calculated without throwing error	Pass		
6	Dan Donley	04/05/2020	Noice uncertainty calculated without throwing error	Pass		

Execution Summary:	The noise uncertainty is sucessfully derived from pulling points from a gaussian distribution with a mean zero and standard deviation of one.

Table 8.3.19. Execution Report of Test Case TC-022

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-022
Testing Tools Used:	spyder-unittest
Testing Type:	Agile (automated) testing

Execution Steps:

1	Click 'Unit testing' on the right side
2	Click the gear and click 'Configure'
3	Select 'unittest' for the testing framework
4	Select the folder with the test scripts
5	Click 'OK'
6	Click 'Run tests'

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	James Fennelly	03/08/2020	No scatter plots created	Fail	Not initialized	03/08/2020 by James Fennelly
2	James Fennelly	03/08/2020	Scatter plots created, axis labels are wrong	Fail	Graphs in folder have extra characters in axis labels	03/08/2020 by James Fennelly
3	James Fennelly	03/08/2020	Plots generate	Pass		
4	Dan Donley	04/05/2020	Plots generate	Pass		

Execution Summary:	Scatter plots of variables generate and right number is generated

Table 8.3.20. Execution Report of Test Case TC-023

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-023
Testing Tools Used:	spyder-unittest
Testing Type:	Agile (automated) testing

Execution Steps:

1	Click 'Unit testing' on the right side
2	Click the gear and click 'Configure'
3	Select 'unittest' as the testing framework
4	Select the folder with the testing scripts
5	Click 'OK'
6	Click 'Run tests'

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	James Fennelly	03/08/2020	No heatmap mde	Fail	Not initialized	03/09/2020 by James Fennelly
2	James Fennelly	03/09/2020	Heatmap made and is correct	Pass		
3	Dan Donley	04/05/2020	Heatmap made and is correct	Pass		

Execution Summary:	Heatmap is present and is correct4

Table 8.3.21. Execution Report of Test Case TC-024

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Case ID:	TC-024	
Testing Tools Used:	Manual testing	
Testing Type:	Function coverage	
Execution Steps:	1	Load the lasso_ridge_regression.py script
	2	Run the script
	3	Examine results in folder structure, there should be 1 ridge correlation graph and 1 lasso correlation graph

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	James Fennelly	03/16/2020	No graphs generated	Fail	Not initialized	03/16/2020 by James Fennelly
2	James Fennelly	03/16/2020	Assertion thrown	Fail	AssertionError: Number of graphs in folder incorrect	03/21/2020 by Dan Donley
3	James Fennelly	03/21/2020	Graphs generated and correct	Pass		

Execution Summary:	Ridge and lasso correlation graphs now generated

Table 8.3.22. Execution Report of Test Case TC-039

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-039
Testing Tools Used:	spyder-unittest
Testing Type:	Agile (automated) testing

Execution Steps:

1	Select 'Unit testing' on the right side of the IDE
2	Select the gear and click 'Configure'
3	Choose 'unittest' for the testing framework
4	Select the folder with the test scripts
5	Click 'OK'
6	Click 'Run tests'

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Alec Dady	03/19/2020	Points not evaluated with model	Fail	Not initialized	03/19/2020 by Alec Dady
2	Alec Dady	03/19/2020	Crashed on execution	Fail	Error: 'model.h5' not found	03/20/2020 by Alec Dady
3	Alec Dady	04/17/2020	Model can be loaded, not evaluated	Fail	Error: evaluation not implemented	04/19/2020 by Alec Dady
4	Alec Dady	04/19/2020	Model can be loaded & evaluated	Pass		

Execution Summary:	Model can be loaded & evaluated

Table 8.3.23. Execution Report of Test Case TC-017

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-017
Testing Tools Used:	Manual Testing Script
Testing Type:	Component interface testing
Execution Steps:	1 Open preprocess_raw_data.py in spyder IDE 2 Click Run 3 Examine results in file directory

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	02/22/2020	Data set not sucessfully normalized and standardized	Fail	Not initialied	02/22/2020 by Dan Donley
2	Dan Donley	02/22/2020	Data set normalized and standardized, but not to expected value	Fail	Data set does not match expected value	02/22/2020 by Dan Donley
3	Dan Donley	02/22/2020	Data set normalized and standardized	Pass		
4	Dan Donley	03/05/2020	Data set normalized and standardized	Pass		
5	Dan Donley	04/05/2020	Data set normalized and standardized	Pass		
6	Dan Donley	04/06/2020	Values not correctly preprocessed (failure of another case)	Fail	AttributeError: 'numpy.ndarray' has no attribute 'values'	04/06/2020 by Dan Donley
7	Dan Donley	04/06/2020	Data set normalized and standardized	Pass		

Execution Summary:	The data set can now be extracted from CSV files and transformed into a matrix for processing through various algorithms.

Table 8.3.24. Execution Report of Test Case TC-018

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-018
Testing Tools Used:	spyder-unittest
Testing Type:	Component interface testing

Execution Steps:

1	Navigate to the Unit Testing pane on the right side.
2	Click the gear and select 'Configure'
3	For the framework, select unittest
4	For the directory, select the 'Learner' folder
5	Click OK
6	Click 'Run tests'

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Alec Dady	02/22/2020	Model couldn't be trained and saved	Fail	Test code not yet implemented	02/22/2020 by Alec Dady
2	Alec Dady	02/22/2020	Model is trained and saved	Pass		
3	Dan Donley	03/05/2020	Model is trained and saved	Pass		
4	Alec Dady	03/09/2020	Model is trained and saved	Pass		
5	Alec Dady	03/12/2020	Model is trained and saved	Pass		
6	Alec Dady	03/17/2020	Model is trained and saved	Pass		
7	Alec Dady	04/05/2020	Model is trained and saved	Pass		

Execution Summary:	The DBN model can be trained using a data set and saved to storage.

Table 8.3.25. Execution Report of Test Case TC-025

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-025
Testing Tools Used:	spyder-unittest
Testing Type:	Component interface testing
Execution Steps:	1 Navigate to the Unit Testing pane on the right side. 2 Click the gear and select 'Configure' 3 For the framework, select unittest 4 For the directory, select the '' folder 5 Click OK 6 Click 'Run tests'

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	02/22/2020	State not shown and points not saved	Fail	Not initialized	02/22/2020 by Dan Donley
2	Dan Donley	02/22/2020	State is shown, points saved	Fail	Data points not correct value	02/24/2020 by Dan Donley
3	Dan Donley	02/24/2020	Points are saved and returned in new DataFrame	Pass		
4	Dan Donley	03/05/2020	Points are saved and returned in new DataFrame	Pass		
5	Dan Donley	04/05/2020	Points are saved and returned in new DataFrame	Pass		

Execution Summary: Data points are saved and stored

Table 8.3.26. Execution Report of Test Case TC-026

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-026
Testing Tools Used:	Manual testing
Testing Type:	Functional testing
Execution Steps:	1 Open pca.py
	2 Open lasso_ridge_regression.py
	3 Open singleVariateAnalysis.py
	4 Ensure preprocessed data is in correct location
	5 Run the scripts and observe the generated graphs

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	04/05/2020	No correlation graphs generated	Fail	Not initialized	04/05/2020 by Dan Donley
2	Dan Donley	04/05/2020	Graphs generated	Pass		

Execution Summary:	Correlation scripts completed and ready for acceptance testing

Table 8.3.27. Execution Report of Test Case TC-027

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Case ID:	TC-027	
Testing Tools Used:	Manual testing	
Testing Type:	Functional testing	
Execution Steps:	1	Launch the preprocess_raw_data.py script
	2	Ensure the raw data is in the correct location
	3	Run the script and check if the cleaned data is in the correct location

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	04/05/2020	Data not preprocessed and cleaned	Fail	Not initialized	04/05/2020 by Dan Donley
2	Dan Donley	04/05/2020	Data preprocessed and stored in correct location	Pass		

Execution Summary:	Data Input of Data module complete and ready for requirements acceptance testing

Table 8.3.28. Execution Report of Test Case TC-028

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Case ID:	TC-028	
Testing Tools Used:	Manual testing	
Testing Type:	Functional testing	
Execution Steps:	1	Launch the run_model.py script
	2	Ensure the generated data from the filter is in the correct location
	3	Run the script

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Alec Dady	04/05/2020	Model wan't trained	Fail	Not initialized	04/05/2020 by Alec Dady
2	Alec Dady	04/05/2020	Model trained and produces accuracy result	Pass		

Execution Summary:	Model can be trained given inputs, ready for acceptance testing

Table 8.3.29. Execution Report of Test Case TC-029

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-029
Testing Tools Used:	Manual testing
Testing Type:	Functional testing
Execution Steps:	1 Launch the run_filter.py script 2 Ensure the cleaned data is in the right location 3 Run the script and ensure the generated data is in the correct location

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	03/17/2020	No data generated	Fail	Not initialized	03/19/2020 by Dan Donley
2	Dan Donley	03/19/2020	No data generated	Fail	ValueError: shape of (5, 6) does not match expected shape of (5, 1)	03/19/2020 by Dan Donley
3	Dan Donley	03/19/2020	Data generated and stored	Pass		
4	Dan Donley	03/25/2020	Data generated and stored	Pass		
5	Dan Donley	04/05/2020	Data generated and stored	Pass		

Execution Summary:	Filter can now be run, ready for acceptance testing

Table 8.3.30. Execution Report of Test Case TC-030

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-030
Testing Tools Used:	None
Testing Type:	Non-Functional testing
Execution Steps:	1 Examine citizenship status of each developer

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	09/01/2019	Not citizen of China	Pass		
2	Alec Dady	09/01/2019	Not citizen of China	Pass		
3	James Fennelly	09/01/2019	Not citizen of China	Pass		

Execution Summary:	All developers do not have citizenship to China

Table 8.3.31. Execution Report of Test Case TC-031

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-031
Testing Tools Used:	None
Testing Type:	Operational readiness testing
Execution Steps:	1 Setup environment on Windows machine
	2 Launch preprocess_raw_data.py
	3 Launch generate_correlation_data.py
	4 Launch run_filter.py
	5 Launch run_model.py

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	04/05/2020	Failed during execution	Fail	Not initialized	04/08/2020 by Dan Donley
2	Dan Donley	04/08/2020	System can launch all scripts	Pass		

Execution Summary:	System operational on Windows

Table 8.3.32. Execution Report of Test Case TC-032

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-032
Testing Tools Used:	None
Testing Type:	Operational readiness testing
Execution Steps:	1 Setup environment on Linux machine
	2 Launch preprocess_raw_data.py
	3 Launch generate_correlation_data.py
	4 Launch run_filter.py
	5 Launch run_model.py

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	04/05/2020	Failed during execution	Fail	Not initialized	04/08/2020 by Dan Donley
2	Dan Donley	04/08/2020	System scripts launch	Pass		

Execution Summary:	System operational on Linux

Table 8.3.33. Execution Report of Test Case TC-033

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-033
Testing Tools Used:	None
Testing Type:	Operational readiness testing
Execution Steps:	1 Setup environment on macOS machine
	2 Launch preprocess_raw_data.py
	3 Launch generate_correlation_data.py
	4 Launch run_filter.py
	5 Launch run_model.py

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Alec Dady	04/05/2020	Failed during execution of script	Fail	Not initialized yet	04/10/2020 by Alec Dady
2	Alec Dady	04/10/2020	Scripts all ran	Pass		

Execution Summary:	System works on macOS

Table 8.3.34. Execution Report of Test Case TC-034

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-034
Testing Tools Used:	Manual Testing
Testing Type:	Non-Functional testing
Execution Steps:	1 Launch test_tc034.py
	2 Examine results

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	03/15/2020	Failed during execution	Fail	Not initialized	03/15/2020 by Dan Donley
2	Dan Donley	03/15/2020	All records are either SPT-100 or SPT-140 points	Pass		

Execution Summary:	All points in SPT data are either SPT-100 or SPT-140 records

Table 8.3.35. Execution Report of Test Case TC-035

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-035
Testing Tools Used:	Manual testing
Testing Type:	Functional testing
Execution Steps:	1 Launch preprocess_raw_data.py
	2 Check data stored in correct place

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	04/11/2020	Data not generated	Fail	Not initialized	04/11/2020 by Dan Donley
2	Dan Donley	04/11/2020	Data preprocessed successfully	Pass		

Execution Summary:	Associated requirements fulfilled

Table 8.3.36. Execution Report of Test Case TC-036

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**	
Test Case ID:	TC-036	
Testing Tools Used:	Manual testing	
Testing Type:	Functional testing	
Execution Steps:	1	Run correlation scripts
	2	Examine results created in folder

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	04/11/2020	No resutls created	Fail	Not initialized	04/11/2020 by Dan Donley
2	Dan Donley	04/11/2020	Results created	Pass		

Execution Summary:	Associated requirements fulfilled

Table 8.3.37. Execution Report of Test Case TC-037

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-037
Testing Tools Used:	Manual testing
Testing Type:	Functional testing
Execution Steps:	1 Run run_filter.py
	2 Check if results sent to right location

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	04/11/2020	Nothing happened	Fail	Not initialized	04/11/2020 by Dan Donley
2	Dan Donley	04/11/2020	Results sent to right location, filter can run and generates data	Pass		

Execution Summary:	Associated requirements fulfilled

Table 8.3.38. Execution Report of Test Case TC-038

Project Name:	**NASA Psyche Mission: Machine Learning Analysis of Hall Thruster Facility Effects Data**
Test Case ID:	TC-038
Testing Tools Used:	Manual testing
Testing Type:	Functional testing
Execution Steps:	1 Run run_model.py
	2 Examine produced results in folder structure

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Dan Donley	04/11/2020	Model not created, saved, and evaluated	Fail	Not initialized	04/12/2020 by Alec Dady
2	Alec Dady	04/12/2020	Model created, saved, and evaluated	Pass		

Execution Summary:	Associated requirements fulfilled