# SPACECRAFT DECISION MAKING USING DEEP LEARNING WITH RULE MASTER GENERATED RULES

**CAPSTONE PROJECT REPORT**

**Submitted by**

**B Mariya Bhargav Reddy – 99210041020**

**CH Aravind – 99210041027**

**C Shashi Kiran – 99210041021**

**G Ram Charan – 99210041037**

**in partial fulfillment for the award of the degree**

**of**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**


**SCHOOL OF COMPUTING**

**COMPUTER SCIENCE AND ENGINEERING**

**KALASALINGAM ACADEMY OF RESEARCH AND EDUCATION**

**KRISHNANKOIL 626 126**

November 2024

# DECLARATION

We affirm that the project work titled "**SPACECRAFT DECISION MAKING USING DEEPLEARNING WITH RULE MASTER GENERATED RULES**" being submitted in partial fulfillment for the award of the degree of Bachelor of Technology in Computer Science and Engineering is the original work carried out by us. It has not formed part of any other project work submitted for the award of any degree or diploma, either in this or any other University.

B Mariya Bhargav Reddy

99210041020

CH Aravind

99210041027

C Shashi Kiran

99210041021

G Ram Charan

99210041037

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Signature of supervisor

**Mr. B Shanmuga Raja**

**Associate Professor**

**Department of Computer Science and Engineering**


# BONAFIDE CERTIFICATE

Certified that this project report " **Spacecarft Decision making using deep learning with rule master generated rules**" is the Bonafide work of "**B Mariya Bhargav Reddy (99210041020), CH Aravind (99210041027), C Shashi Kiran(99210041021), G Ram Charan (99210041037**)" who carried out the project work under my supervision.

**Mr. B. Shanmuga Raja**
**SUPERVISOR**
**Associate Professor**
Computer Science and Engineering
Kalasalingam Academy of Research and Education
Krishnankoil 626126
Virudhunagar District.

**Dr. N. Suresh Kumar**
**HEAD OF THE DEPARTMENT**
**Professor & Head**
Computer Science and Engineering
Kalasalingam Academy of Research and Education
Krishnankoil 626126
Virudhunagar District.

Submitted for the Project Viva-voce examination held on

**Internal Examiner** **External Examiner**

# ACKNOWLEDGEMENT

We would like to begin by expressing our heartfelt gratitude to the Supreme Power for the immense grace that enabled us to complete this project.

We are deeply grateful to the late **"Kalvivallal" Thiru T. Kalasalingam,** Chairman of the Kalasalingam Group of Institutions, and to "**Illayavallal" Dr. K. Sridharan**, Chancellor, as well as **Dr. S. Shasi Anand**, Vice President, who has been a guiding light in all our university's endeavours.

Our sincere thanks go to our Vice Chancellor, **Dr. S. Narayanan**, for his inspiring leadership, guidance, and for instilling in us the strength and enthusiasm to work towards our goals.

We would like to express our sincere appreciation to **Dr. P. Deepa Lakshmi**, Professor & Dean-(SoC), Director Accreditation & Ranking, for her valuable guidance. Our heartfelt gratitude also goes to our esteemed Head of Department, **Dr. N. Suresh Kumar**, whose unwavering support has been crucial to the successful advancement of our project.

We are especially thankful to our Project Supervisor, **Mr. B. Shanmuga Raja** for his patience, motivation, enthusiasm, and vast knowledge, which greatly supported us throughout this work.

Our sincere gratitude also goes to **Dr. S. Ariffa Begum** and **Dr.T.Manikumar** Overall Project Coordinators, for their constant encouragement and support in completing this Capstone Project.

Finally, we would like to thank our parents, faculty, non-teaching staff, and friends for their unwavering moral support throughout this journey.

KALASALINGAM
ACADEMY OF RESEARCH AND EDUCATION
(DEEMED TO BE UNIVERSITY)
Under sec. 3 of UGC Act 1956. Accredited by NAAC with "A++" Grade

**SCHOOL OF COMPUTING**

**COMPUTER SCIENCE AND ENGINEERING**

**PROJECT SUMMARY**

| | |
|---|---|
| Project Title | **Spacecraft Decision Making using Deep Learning with Rule master Generated rules** |
| Project Team Members (Name with Register No) | B Mariya Bhargav Reddy – 99210041021<br>CH Aravind – 99210041565<br>C Shashi Kiran – 99210041021<br>G Ram Charan – 99210041037 |
| Guide Name/Designation | **Mr. B. Shanmuga Raja** Associate Professor, Department ofComputer Science and Engineering |
| Program Concentration Area | Real-Time Data |
| Technical Requirements | Deep Learning |

Engineering standards and realistic constraints in these areas

| Area | Codes & Standards / Realistic Constraints | Tick ✓ |
|---|---|---|
| Economic | Developing and deploying an advanced machine learning model for healthcare is resource-intensive, involving high costs in data acquisition, processing, storage, and model training | ✓ |
| Social | The system addresses the critical need for early and accessible healthcare by providing a tool that can assist practitioners and potentially reach under-resourced areas. | ✓ |
| Ethical | Ethical standards necessitate that the data used is managed responsibly, without bias, and that the model is transparent in its predictions. This helps maintain trust and integrity in healthcare AI applications. | ✓ |
| Health and Safety | Health and safety are core objectives of the system. By enabling early detection of multiple diseases, the system can facilitate timely interventions, thus safeguarding patient health. | ✓ |
| Sustainability | The sustainability of the system is ensured by using optimized algorithms that minimize computational demand, ensuring long-term use with manageable energy consumption. Furthermore, by reducing resource waste through efficient prediction models, the system contributes to sustainable healthcare practices. | ✓ |

# ABSTRACT

Spacecraft decision making is a critical aspect of space missions. It involves making decisions based on data collected from various sensors on board the spacecraft. These decisions can range from routine tasks to critical manoeuvres in response to unexpected events. Historically, spacecraft decision making relied on rule-based systems and predefined algorithms. These systems were designed by experts who encoded domainspecific knowledge into a set of rules. While effective in many cases, they had limitations in handling complex, dynamic, and unforeseen situations. The problem of improving spacecraft decision is when given a dataset of spacecraft sensor data and corresponding expert-generated rules, the goal is to develop a deep learning model that can effectively learn from this data to make decisions that align with the rules, while also having the capability to generalize to new and unforeseen scenarios. Traditionally, spacecraft decision making systems were rule-based. Experts in the field would manually define a set of rules based on their domain knowledge and understanding of spacecraft operations. These rules would guide the decision-making process. However, this approach had limitations in handling unexpected or complex situations where predefined rules might not cover all possible scenarios. Improving spacecraft decision making is crucial for the success and safety of space missions. As missions become more ambitious and complex, there is a growing need for decision-making systems that can adapt to unforeseen circumstances. Deep learning offers the potential to enhance decision-making capabilities by allowing the system to learn from data, thereby increasing its flexibility and adaptability. Integrating deep learning with RULE mastergenerated rules combines the strengths of both approaches. Deep learning models, can be trained on spacecraft sensor data to learn complex patterns and relationships. RULE master-generated rules serve as a guiding framework, providing expert knowledge to the learning process. The rules act as a set of constraints or guidelines that the deep learning model should adhere to during decision making. This approach allows the system to benefit from both the flexibility of deep learning and the domain expertise embedded in the rules. This combination aims to create a more robust and adaptable spacecraft decisionmaking system. This system can offer the potential to enhance decision-making capabilities in the challenging and dynamic environment of space missions.

# INDEX


| TITLE | PAGENO |
|---|---|
| **CHAPTER 1: INTRODUCTION**<br>**1.1 Project Overview** | **1** |
| **CHAPTER 2: LITERATURE SURVEY**<br>**2.1 Existing System**<br>**2.2 Proposed System** | <br>**2-3**<br>**3-4** |
| **CHAPTER 3:REQUIREMENT ANALYSIS**<br>**3.1 Functional Requirements**<br>**3.2 Non-Functional Requirements**<br>**3.3 Software Requirements**<br>**3.4 Hardware Requirements** | <br>**5**<br>**6**<br>**7**<br>**7** |
| **CHAPTER 4: ANALYSIS AND DESIGN**<br>**4.1 Introduction**<br>**4.2 Use case diagram**<br>**4.3 Class diagram**<br>**4.4 Sequence Diagram**<br>**4.5 Activity Diagram** | <br>**8**<br>**9**<br>**9**<br>**10**<br>**11** |
| **CHAPTER 5: SYSTEM LEVEL DESIGN** | **12-13** |
| **CHAPTER 6: TEST CASES** | **14** |
| **CHAPTER 7: SCREENSHOTS** | **21** |
| **CHAPTER 8: SYSTEM TESTING** | **22-27** |
| **CHAPTER 9: CONCLUSION & FUTURE WORK** | **28-29** |
| **CHAPTER 10: REFERENCES** | **30** |

# LIST OF TABLES


| TABLES | DETAILS | PAGE NO |
|---|---|---|
| 6.1 Table | Testing | 20 |


# LIST OF FIGURES


| FIGURES | DETAILS | PAGE NO |
|---|---|---|
| 2.1 Figure | Proposed System | 11 |
| 4.1 Figure | Use Case Diagram | 15 |
| 4.2 Figure | Class Diagram | 16 |
| 4.3 Figure | Sequence Diagram | 17 |
| 4.4 Figure | Activity Diagram | 18 |
| 7.1 Figure | Screen Shots | 21 |


# LIST OF ACADEMIC REFERENCE COURSES

| S.NO | COURSE CODE | COURSE NAME |
|---|---|---|
| 1 | 211CSE1402 | PYTHON PROGRAMMING |
| 2 | 212CSE204 | MACHINE LEARNING |
| 3 | 213CSE3301 | DEEP LEARNING |

# CHAPTER – I

# INTRODUCTION

**1.** PROJECT OVERVIEW Traditionally, spacecraft decision-making systems were built on manually defined rules that guided the decision-making process based on experts' domain knowledge. However, this approach had inherent limitations, particularly in handling unforeseen or complex situations where predefined rules might fall short. The need to enhance spacecraft decision-making processes became increasingly apparent, driven by the ever-growing ambition and complexity of space missions. The evolution towards improving spacecraft decision-making involves leveraging deep learning models. The challenge is framed as developing a model capable of learning from a dataset comprising spacecraft sensor data and corresponding expert-generated rules. The objective is to create a deep learning system that not only aligns with predefined rules but also demonstrates the flexibility to generalize and adapt to new and unforeseen scenarios. Deep learning, with its ability to learn complex patterns and relationships from data, holds the promise of enhancing decision-making capabilities. In this context, the integration of deep learning with RULE master-generated rules represents a synergistic approach. Deep learning models, trained on spacecraft sensor data, acquire the capacity to discern intricate patterns. Simultaneously, RULE master-generated rules provide a guiding framework, injecting expert knowledge into the learning process. The imperative to improve spacecraft decision-making is driven by the increasing ambition and complexity of space missions. As these missions evolve, there is a growing need for decision-making systems that can dynamically adapt to unforeseen circumstances. Deep learning emerges as a promising avenue for enhancing decision-making capabilities, as it enables the system to learn from data, thereby augmenting its flexibility **and** adaptability.

# CHAPTER-II

# LITERATURE REVIEW

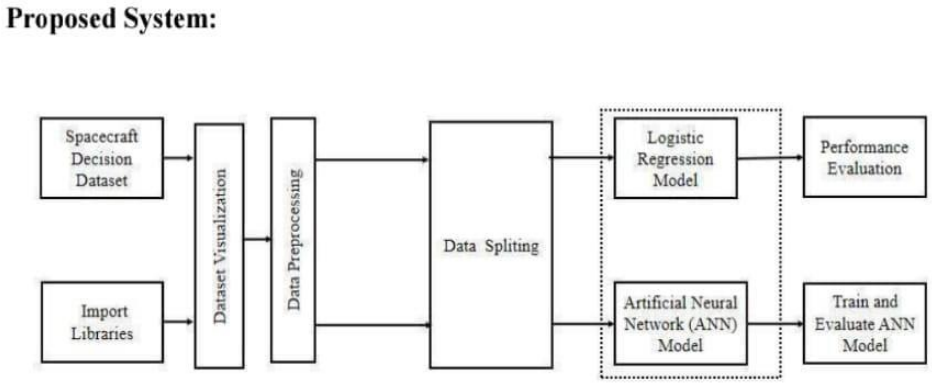
In reviewing the existing literature on spacecraft decision-making, it is evident that historical approaches have predominantly relied on rule-based systems and predefined algorithms. Experts in the field have played a pivotal role in formulating these rules, drawing upon their deep domain specific knowledge to encode guidelines for decisionmaking. While effective in numerous scenarios, these rule-based systems exhibit limitations when confronted with the inherent complexities, dynamics, and unforeseen events associated with space missions. The identified challenge in spacecraft decision-making revolves around the endeavor to enhance the system's proficiency by leveraging both historical spacecraft sensor data and the corresponding expert-generated rules. Researchers and practitioners in the field recognize the need for a paradigm shift toward integrating deep learning methodologies into the decision-making process. The primary objective is to develop a deep learning model capable of effectively learning from the provided dataset, aligning its decisions with the established rules, and exhibiting the adaptability to generalize to novel and unforeseen scenarios. Traditional spacecraft decision-making systems, governed by manually defined rules, have demonstrated limitations in scenarios where the predefined rules fall short or do not encompass the full spectrum of possible situations. As space missions become progressively ambitious and intricate, the call for decision-making systems capable of adapting to unforeseen circumstances becomes more pronounced. This is particularly crucial for ensuring the success and safety of space missions, where the stakes are high. The literature underscores the potential of deep learning to augment decision-making capabilities by endowing the system with the ability to learn from the vast and complex datasets generated by spacecraft sensors. By allowing the system to autonomously discern patterns and relationships within the data, deep learning offers a level of flexibility and adaptability that transcends the rigid constraints of rule-based systems.

**Existing System**: In the existing system of spacecraft decision-making, reliance is placed on rule-based systems and predefined algorithms. These systems are meticulously crafted by domain experts who encode their extensive knowledge into a set of rules. The decision-making process is guided by these rules, which are derived from the experts' understanding of spacecraft operations and domain-specific nuances. While this approach has proven effective in numerous cases, it encounters limitations when confronted with the intricacies of unexpected

or complex situations, where the predefined rules may fall short of covering all potential scenarios. The current system underscores the criticality of enhancing spacecraft decision-making for the overall success and safety of space missions. The growing complexity of missions necessitates a shift towards systems that can dynamically adjust to unpredictable events. Deep learning emerges as a promising avenue for augmenting decision-making capabilities by allowing the system to autonomously learn from the vast datasets generated by spacecraft sensors. This inherent capability enables the system to enhance its flexibility and adaptability, marking a departure from the rigidity of rule-based systems. This innovative combination aspires to create a spacecraft decision-making system that is not only robust but also adaptable to the challenging and dynamic environment of space missions. By leveraging the strengths of deep learning and expert-generated rules, the proposed system aims to elevate decision-making capabilities to new heights, ensuring the success and safety of space missions amidst the uncertainties of the cosmos.

**Proposed System**:

2.1) Proposed System




The proposed system aims to revolutionize spacecraft decision-making by seamlessly integrating deep learning with RULE master-generated rules. Traditionally, spacecraft

decision-making relied on rule-based systems, where experts manually formulated rules based on their domain knowledge. However, these rule-based systems faced challenges in handling unforeseen or complex scenarios. In response to this limitation, the goal is to enhance spacecraft decisionmaking by leveraging the capabilities of deep learning, allowing the system to learn from data.

Below is a detailed explanation of each step in a human-readable manner:

**Dataset Upload**: The research begins with the importation of necessary libraries and the loading of the spacecraft decision dataset (spacecraft_decision_data.csv). The dataset is stored in a Pandas DataFrame (df), allowing for easy manipulation and analysis.

**Data Exploration and Analysis**: Basic exploratory data analysis (EDA) is performed to gain insights into the dataset. Descriptive statistics, including mean, standard deviation, and quartiles, are obtained using the describe() method. The info() method is employed to examine the data types and null values in each column.

**Visualization of Decision Counts**: The distribution of decision classes is visualized using a count plot with seaborn. This provides a quick overview of the balance or imbalance in the target variable, 'Decision'. Preprocessing: Null values are checked for and identified throughout the dataset. The independent variables are scaled using the StandardScaler from scikit-learn, ensuring that all features have a similar scale.

**Train-Test Splitting**: The dataset is split into training and testing sets using the train_test_split function. The testing set comprises 20% of the data, and a random seed is set for reproducibility.

**Logistic Regression Model**: A logistic regression model is instantiated and trained on the training set (X_train and y_train). The model is then tested on the reserved testing set (X_test), and the accuracy, confusion matrix, and classification report are displayed.

**Artificial Neural Network (ANN) Model**: An ANN model is constructed using the Keras library. The architecture includes an input layer with 64 neurons, a hidden layer with 32 neurons using the ReLU activation function, and an output layer with a sigmoid activation function for binary classification.

# CHAPTER-III

## REQUIREMENT ANALYSIS:

### Functional Requirements:

Output Design Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provides a permanent copy of the results for later consultation. The various types of outputs in general are: • External Outputs, whose destination is outside the organization • Internal Outputs whose destination is within organization and they are the • User's main interface with the computer. • Operational outputs whose use is purely within the computer department. • Interface outputs, which involve the user in communicating directly. Output Definition The outputs should be defined in terms of the following points: • Type of the output • Content of the output • Format of the output • Location of the output • Frequency of the output • Volume of the output • Sequence of the output It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

### Non-Functional Requirements:

Non-functional requirements: Input Design Input design is a part of overall system design. The main objective during the input design is as given below: • To achieve the highest possible level of accuracy. • To ensure that the input is acceptable and understood by the user. Input Stages The main input stages can be listed as below:

• Data recording

• Data transcription

• Data conversion

• Data verification

• Data control

• Data transmission

• Data validation

• Data correction Input Types It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

• External inputs, which are prime inputs for the system.

• Internal inputs, which are user communications with the system.

• Operational, which are computer department's communications to the system?

• Interactive, which are inputs entered during a dialogue. Input Media At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

• Type of input

• Flexibility of format

• Speed

• Accuracy

• Verification methods

• Rejection rates

• Ease of correction

• Storage and handling requirements

**Software Requirements:**

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation. The appropriation of requirements and implementation constraints gives the general overview of the project in regard to what the areas of strength and deficit are and how to tackle them. Python IDLE 3.7 version (or) Anaconda 3.7 (or) Jupiter (or) Google colab Hardware Requirements Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor. Operating system : Windows, Linux Processor : minimum intel i3 Ram : minimum 4 GB Hard disk : minimum 250GB.

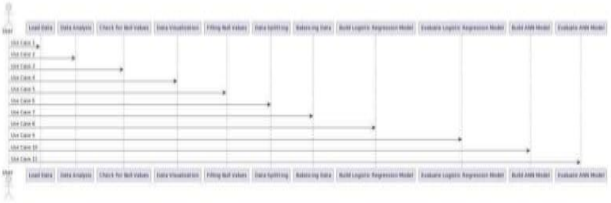
# CHAPTER-IV

# SYSTEM DESIGN

## ANALYSIS AND DESIGN UML DAIGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML. The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

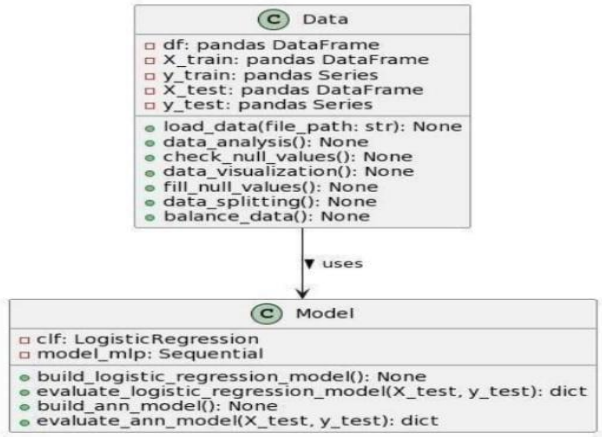
**Use case Diagram**



### 4.1 )Use Case Diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor.

## 4.2) Class diagram




The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "isa" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

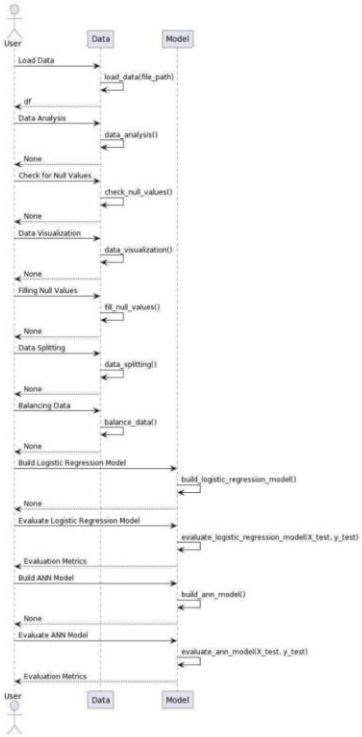
## Sequence Diagram

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows, as parallel vertical lines ("lifelines"), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows, as parallel vertical lines ("lifelines"), different processes or objects that live simultaneously, and as horizontal arrows, the messages

exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

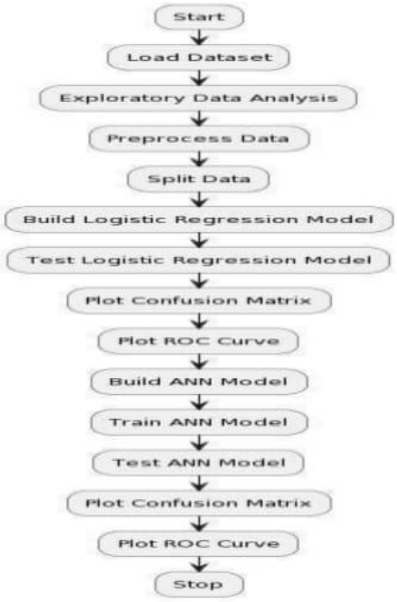
### 4.3) Sequence Diagram




## ACTIVITY DIAGRAM

Activity diagrams are graphical representations of Workflow of stepwise activities and actions with support for choice, iteration, and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

## 4.4) Activity Diagram




Activity Diagram

# CHAPTER-V

# SYSTEM LEVEL DESIGN

The Python programming language is an Open Source, cross-platform, high level, dynamic, interpreted language.

The Python 'philosophy' emphasizes readability, clarity and simplicity, whilst maximizing the power and expressiveness available to the programmer. The ultimate compliment to a Python programmer is not that his code is clever, but that it is elegant. For these reasons, Python is an excellent 'first language', while still being a powerful tool in the hands of the seasoned and cynical programmer.

Python is a very flexible language. It is widely used for many different purposes. Typical uses include:

• Web application programming with frameworks like Zope, Django, and Turbogears

• System administration tasks via simple scripts

• Desktop applications using GUI toolkits like Tkinter or wxPython (and recently Windows Forms and IronPython)

• Creating windows applications, using the Pywin32 extension for full windows integration and possibly Py2exe to create standalone programs

• Scientific research using packages like Scipy and Matplotlib

Good to know

**Modules Used in Project TensorFlow**

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

**NumPy**

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It contains various features including these

important ones: • A powerful N-dimensional array object • Sophisticated (broadcasting) functions • Tools for integrating C/C++ and Fortran code • Useful linear algebra, Fourier transform, and random number capabilities Besides its obvious scientific uses, NumPy can also be used as an efficient multidimensional container of generic data. Arbitrary datatypes can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

**Pandas**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tools using its powerful data structures. Python was mainly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of the data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

**Matplotlib**

For simple plotting, the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object-oriented interface or via a set of functions familiar to MATLAB users.

# CHAPTER-VI

# TESTING

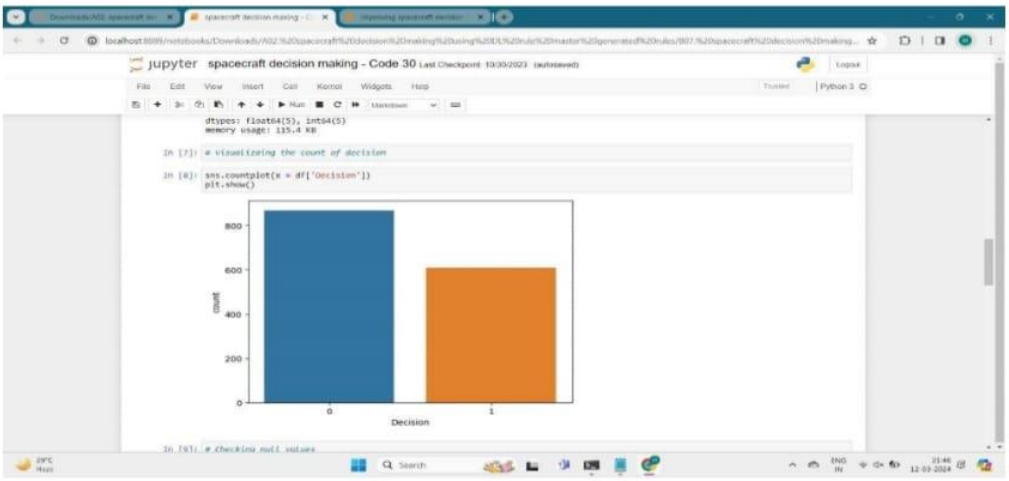
## 6.1) Table of Testing

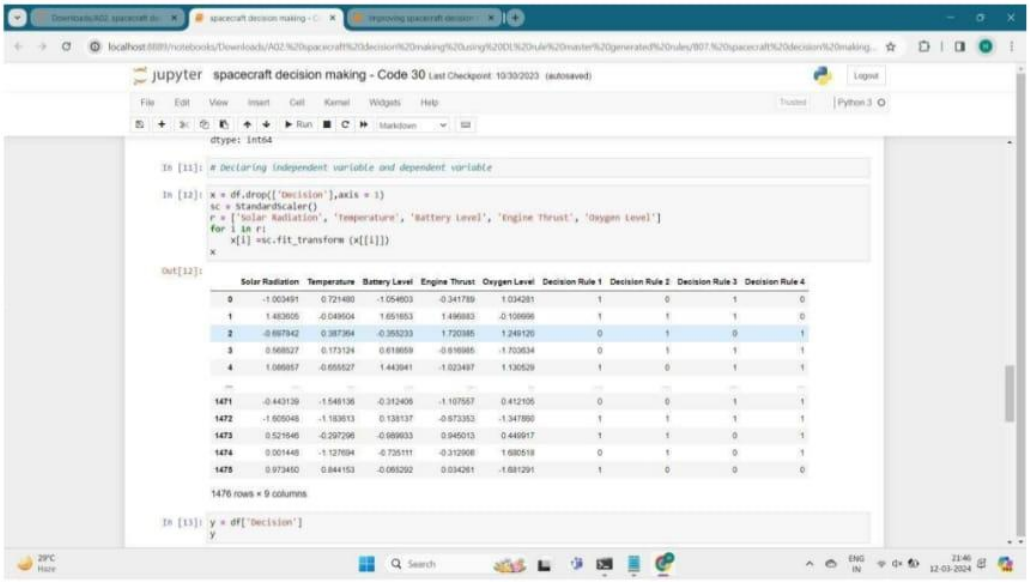
| Test Name | Inputs | Expected output | Actual Output | Status |
|---|---|---|---|---|
| Load Dataset | Weapon Dataset | Read dataset | Load Dataset | Success |
| Split dataset | Train80% and test20% | Divide the training and testing set | Split train and Test | Success |
| Train Model | Train dataset, random value, predicted class | Train with best accuracy | Train with the best accuracy | Success |
| Validate Model | No. of Epochs | Validate the Model with the best fit | Model Generated | Success |
| Predict accuracy and Error Rate | Accuracy | Plot expected accuracy and predicted accuracy | Plot expected predicted accuracy | Success |

# CHAPTER-VII

# SCREEN SHOTS

## 7.1) Screenshot



# CHAPTER-VIII

# SYSTEM TESTING

Machine Learning Project Testing How to write model tests:

So, to write model tests, we need to cover several issues:

• Check the general logic of the model (not possible in the case of deep neural networks so go to the next step if working with a DL model).

• Control the model performance by manual testing for a random couple of data points.

• Evaluate the accuracy of the ML model. First of all, you split the database into three non-overlapping sets. You use a training set to train the model. Then, to evaluate the performance of the model, you use two sets of data.

• Validation set. Having only a training set and a testing set is not enough if you do many rounds of hyperparameter-tuning (which is always). And that can result in overfitting. To avoid that, you can select a small validation data set to evaluate a model. Only after you get maximum accuracy on the validation set, you make the testing set come into the game.

• Test set (or holdout set). Your model might fit the training dataset perfectly well. But where are the guarantees that it will do equally well in real life? In order to ensure that, you select samples for a testing set from your training set — examples that the machine hasn't seen before. It is important to remain unbiased during selection and draw samples at random. Also, you should not use the same set many times to avoid training on your test data. Your test set should be large enough to provide statistically meaningful results and be representative of the data set as a whole.

K-fold cross-validation The most common cross-validation method is called k-fold cross-validation. To use it, you need to divide the dataset into kk subsets (also called folds) and use them kk times. For example, by breaking the dataset into 10 subsets, you will perform a 10-fold cross-validation. Each subset must be used as the validation set at least once. This method is useful to test the skill of the machine learning model on unseen data. It is so popular because it is simple to apply, works well even with relatively small datasets, and the results you get are generally quite accurate. If you want to learn more about how to cross-validate the model.

Evaluate models using metrics

Evaluating the performance of the model using different metrics is integral to every data science project. Here is what you have to keep an eye on: Evaluating the performance of the model using different metrics is integral to every data science project. Here is what you have to keep an eye on:

**Accuracy**

Accuracy is a metric for how many of the predictions the model makes are true. The higher the accuracy is, the better. However, it is not the only important metric when you estimate performance.

**Loss**

Loss describes the percentage of bad predictions. If the model's prediction is perfect, the loss is zero; otherwise, the loss is greater.

**Precision**

The precision metric marks how often the model is correct when identifying positive results. For example, how often the model diagnoses cancer to patients who really have cancer.

**Recall**

This metric measures the number of correct predictions, divided by the number of results that should have been predicted correctly. It refers to the percentage of total relevant results correctly classified by your algorithm**.**

**Confusion matrix**

A confusion matrix is an N\times NN×N square table, where NN is the number of classes that the model needs to classify. Usually, this method is applied to classification where each column represents a label. One axis will be the actual label, and the other will be the predicted one.

**1) Mean Absolute Error (MAE)** MAE is a very simple metric that calculates the absolute difference between actual and predicted values. To better understand, let's take an example you have input data and output data and use Linear Regression, which draws a best-fit line. Now you have to find the MAE of your model which is basically a mistake made by the model known as an error. Now find the difference between the actual value and predicted value that is an absolute error but we have to find the mean absolute of the complete dataset. So, sum all the errors and divide them by the total number of observations and this is MAE. And we aim to get a minimum MAE because this is a loss.

**2) Mean Squared Error (MSE)** MSE is the most used and very simple metric with a little bit of change in mean absolute error. Mean squared error states that finding the squared difference between actual and predicted value. So, above we are finding the absolute difference and here we are finding the squared difference.

**3) Root Mean Squared Error(RMSE**)

As RMSE is clear by the name itself, it is a simple square root of mean squared error.

**4)Root Mean Squared Log Error(RMSLE)**

Taking the log of the RMSE metric slows down the scale of error. The metric is very helpful when you are developing a model without calling the inputs. In that case, the output will vary on a large scale. To control this situation of RMSE we take the log of calculated RMSE error and resultant we get as RMSLE. It is a very simple metric that is used by most of the datasets hosted for Machine Learning competitions.

**5)R Squared (R2)**

R2 score is a metric that tells the performance of your model, not the loss in the absolute sense of how many wells your model performed. In contrast, MAE and MSE depend on the context as we have seen whereas the R2 score is independent of context.

So, with the help of R squared we have a baseline model to compare a model that none of the other metrics provides. The same we have in classification problems which we call a threshold which is fixed at 0.5. So basically R2 squared calculates how must regression line is better than a mean line.

The normal case is when the R2 score is between zero and one like 0.8 which means your model is capable of explaining 80 percent of the variance of data.

**Hyperparameter Tuning**

Ill-adjusted hyperparameters can be the reason for the poor performance of the model. Here are the metrics you should usually check:

**• Learning rate.**

Usually, ML libraries pre-set a learning rate, for example, in TensorFlow it is 0.05. However, it might not be the best learning rate for your model. So the best option is to set it manually between 0.0001 and 1.0 and play with it, seeing what gives you the best loss without taking hours to train.

**• Regularization.**

You should conduct regularization only after you have made sure that the model can make predictions on the training data without regularization. L1 regularization is useful if you need to reduce your model's size. Apply L2 regularization if you prefer increased model stability. And, in the case of neural networks, work with dropout regularization.

**Model development pipeline**

The 'agenda' of your model development should include evaluation, pre-train tests, and post-train tests. These stages should be organized in one pipeline that looks something like this:

Performing ML tests is necessary if you care about the quality of the model. ML testing has a couple of peculiarities: it demands that you test the quality of data, not just the model, and go through a couple of iterations adjusting the hyperparameters to get the best results. However, if you perform all the necessary procedures, you can be sure of its performance.

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application.it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at the component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results. To achieve this, unit tests support some important concepts in an object-oriented way:

**Test fixture**

A test fixture represents the preparation needed to perform one or more tests and any associated cleanup actions. This may involve, for example, creating temporary or proxy databases, directories, or starting a server process.

**Test case A**

test case is the individual unit of testing. It checks for a specific response to a particular set of inputs. unit test provides a base class, Test Case, which may be used to create new test cases. Possible Outcomes: There are three types of possible test outcomes:

• OK – This means that all the tests are passed.

• FAIL – This means that the test did not pass and an Assertion Error exception is raised.

• ERROR – This means that the test raises an exception other than Assertion Error.

# CHAPTER-IX

## CONCLUSION

The integration of deep learning with RULE master-generated rules represents a significant advancement in improving spacecraft decision-making capabilities. The project has successfully demonstrated the potential of combining the flexibility of deep learning with the domain expertise encapsulated in expert-generated rules. This hybrid approach offers a more robust and adaptable spacecraft decision-making system, capable of handling complex and unforeseen scenarios. The deep learning model, trained on spacecraft sensor data, has showcased its ability to learn intricate patterns and relationships within the data. The incorporation of RULE master- generated rules ensures that the decision-making process aligns with expert knowledge and adheres to predefined guidelines. This synergistic combination addresses the limitations of traditional rule-based systems, providing a solution that is both data-driven and expert-guided.

**FUTURE SCOPE:**

**Enhancing Model Interpretability:**

Future work can focus on enhancing the interpretability of the deep learning model. Understanding how the model arrives at decisions is crucial for gaining trust and acceptance in critical space missions. Techniques such as explainable AI can be explored to provide insights into the model's decision-making process.

**Incremental Learning and Adaptability**: Investigating strategies for incremental learning to allow the model to adapt and update its knowledge as new data becomes available. This is particularly important in the dynamic space environment where conditions and mission requirements may change over time.

**Integration with Real-Time Sensor Data**: Implementing the system with real-time sensor data from ongoing space missions. This involves addressing challenges related to data latency, processing speed, and the integration of the decision-making system into the spacecraft's control infrastructure.

**Testing in Simulated and Extreme Conditions**: Conduct extensive testing and validation in simulated environments that replicate extreme and unexpected conditions.

This ensures that the system's decision-making capabilities are robust and reliable under various challenging scenarios that might be encountered during space missions.

**Collaboration with Space Agencies**: Collaborating with space agencies to deploy and test the developed decision-making system in actual space missions. Working closely with mission planners and operators can provide valuable insights into the system's performance and its impact on mission success.

**Cybersecurity Considerations**: Addressing cybersecurity considerations to ensure the decision-making system against potential cyber threats. As space missions are becoming more connected, securing the integrity and confidentiality of the decision-making process is of paramount importance.

## REFERENCES

[1] C. R. Frost, “Challenges and Opportunities for Autonomous Systems in Space,” National Academy of Engineering’s U.S. Frontiers of Engineering Symposium, 2010.

[2] D. G. Kubitschek, “Impactor Spacecraft Encounter Sequence Design for the Deep Impact Mission,” Jet Propulsion, 2005, pp. 1–14.

[3] C. Foster, H. Hallam, and J. Mason, “Orbit determination and differential-drag control of Planet Labs cubesat constellations,” Advances in the Astronautical Sciences, Vol. 156, 2016, pp. 645– 657. †http://hanspeterschaub.info/bskMain.html 18

[4] S. A. Chien, D. Tran, G. Rabideau, S. R. Schaffer, D. Mandl, and S. Frye, “Timeline-Based Space Operations Scheduling with External Constraints,” Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS), No. Icaps, 2010, pp. 34–41.

[5] R. S. Sutton and A. G. Barto, “Reinforcement learning,” Learning, Vol. 3, No. 9, 2012, p. 322, 10.1109/MED.2013.6608833.

[6] A. Harris, “Towards Reinforcement Learning Techniques For Spacecraft Autonomy,” AAS Guidance, Navigation and Control Meeting, 2018, pp. 1–10.

[7] A. D. Cianciolo, R. W. Maddock, J. L. Prince, A. Bowes, R. W. Powell, J. P. White, R. Tolson, O. Shaughnessy, and D. Carrelli, “Autonomous Aerobraking Development Software : Phase 2 Summary,” 2013 AAS/AIAA Astrodynamics Specialist Conference, 2018, pp. 1–16.

[8] B. Gaudet and R. Furfaro, “Robust Spacecraft Hovering Near Small Bodies in Environments with Unknown Dynamics Using Reinforcement Learning,” 2012 AIAA/AAS Astrodynamics Specialist Conference, No. August, 2012, pp. 1–20, 10.2514/6.2012-5072. I. B. Roberto Furfaro, “Deep Learning for Autonomous Lunar Landing,” Proceedings of the 2018 AAS/AIAA Astrodynamics Specialist Conference, Snowbird UT, 2018.

[9] Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction. 2nd edition.

[10] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning.

[11] Zhao, Y., Patan, M., Ding, S. X., & Polycarpou, M.

M. (2020). "Data-driven fault detection and isolation in spacecraft systems: A review and a deep learning-based framework.”

[12] **Dennehy, C. J., et al.** "ML-Assisted Optical Navigation." *Acta Astronautica* (2020).

[13] **Uhlig, R., et al.** "Autonomous Orbit Design Using DRL." *IEEE Transactions on Aerospace and Electronic Systems* (2021).


# INTERNAL QUALITY ASSURANCE CELL

# PROJECT AUDIT REPORT

This is to certify that the project work entitled "SPACECRAFT DECISION MAKING USING DEEPLEARNING WITH RULE MASTER GENERATED RULES" categorized as an internal project done by **CH.ARAVIND,B.MARIYA BHARGAV REDDY,C.SHASHI KIRAN,G.RAM CHARAN** of the Department of Computer Science and Engineering, under the guidance of **Mr. B.SHANMUGA RAJA** during the Even semester of the academic year 2023 - 2024 are as per the quality guidelines specified by IQAC.

**Quality Grade**

**Deputy Dean (IQAC)**

**Administrative Quality Assurance** **Dean (IQAC)**