RYERSON UNIVERSITY
FACULTY OF ENGINEERING, ARCHITECTURE AND SCEINCE
DEPARTMENT OF AEROSPACE ENGINEERING

Development of an AI Ecosystem for AR Applications within Aerospace

Vishaal Venkatesh

AER 870 Aerospace Engineering Thesis – Final Report

Faculty Advisor: Dr. Joon Chung

Date: April 13, 2020

Acknowledgements

I would like to thank my advisor, Dr. Joon Chung, for his support and encouragement throughout this research study providing guidance in the process and evolution of the idea. I would also like to thank the Mixed-Reality Immersive Motion Simulation (MIMS) Laboratory Research Group members, including Pratik Pradhan, Jafer Kamoonpuri, Abhinav Sundar and Vamshi Chittaluri for providing resources, support and guidance for conducting my research.

Abstract

The purpose of this paper is to develop an Artificial Intelligence (AI) ecosystem, to effectively introduce a dynamic approach for Augmented Reality (AR) based instruction modules through predictive machine learning algorithms. The main applications explored are the manufacturing and maintenance sectors within the Aerospace industry. This paper explores the possibility of implementing an innovative process of integrating efficient data handling methodologies and machine learning techniques through a micro-level coordinate system approach. ML algorithms which are expected to have high prediction accuracies have been analyzed and a proof of concept prototype has been developed with a selected algorithm. There are various applications within AR based manufacturing/maintenance instructions in the Aerospace industry, the use cases span from aircraft part manufacturing & maintenance, CAD modelling, part assembly and routine maintenance within the International Space Station (ISS). The research study in this thesis looks to optimize the current AR model-based animation techniques by incorporating AI methodologies throughout the overall process. The current techniques include: Using 3D image information from calibration markers, Teleconference setting for improved communication and Mobile viewing & prototyping unknown surfaces using the HoloLens. This paper proposes an innovative methodology that current techniques can adapt to improve efficiencies and capabilities.

Table of Contents

List of Figures

List of Tables

Nomenclature

AI: Artificial Intelligence

AR: Augmented Reality

CAD: Computer Aided Design

CG: Centre of Gravity

CNN: Convolutional Neural Network

FP: False Positive

FN: False Negative

HMD: Head Mounted Display

iDMU: industrial Digital Mock-Up

IDE: Integrated Development Environment

ISS: International Space Station

KDD: Knowledge Discovery from Data

MBI: Model Based Instructions

ML: Machine Learning

SVM: Support Vector Machine

TP: True Positive

TN: True Negative

RF: Random Forest

RL: Reinforcement Learning

WI: Worker Instructions

1. Introduction

1.1 General

The primary objective of this thesis is to design a methodology to develop a potential Artificial Intelligence (AI) ecosystem that can efficiently provide instructions through predictive Machine Learning (ML) algorithms for various Augmented Reality (AR) based maintenance applications within the aerospace industry. AR technology has played a key role in aiding users to visualize and simulation their surroundings effectively, the major application explored in this project is AR technology used for virtual instructions for maintenance procedures. This method provides instructions to users that make their daily tasks more intuitive, minimize time spent on instruction manuals and optimize the overall maintenance performance [1]. For this thesis project, existing virtual instruction methods using AR were studied. Then, human factors impact were analyzed for the potential integration of ML algorithms with current AR technology and lastly, classification-based ML algorithms were explored through custom hyperparameters and performance comparison. Based on the fundamental concepts of ML and virtual instructions, an experimental application using the flight simulator's inverter from the Mixed-Reality Immersive Motion Simulation (MIMS) Laboratory was designed. The application explores three different ML algorithms namely, Support Vector Machine (SVM), logistic regression and Random Forest (RF). Based on the dataset developed, the algorithms were customized using built-in hyperparameters and optimization methods, the algorithm's prediction performances were compared using traditional classification metrics such as, precision, recall and F1-score. Developing a complete AI ecosystem that is integrated with existing AR modules can require months of testing from a large work force, along with appropriate industrial support and laboratory resources. Therefore, given the fact that this study was only initiated 3 months ago with limited resources and the inconvenience caused due to COVID-19, does not provide sufficient time to complete the intended vision of this project. Therefore, this thesis demonstrates an idea with a design proof of concept based on the knowledge in ML algorithms and python programming.

1.2 Human Factors Consideration

"Human factors" is a critical aspect within AR based virtual maintenance as two separate features are incorporated with each other, the real maintenance is merged with the virtual information for the maintenance personnel to perform tasks. Within AR, the maintenance personnel have access to tracking devices such as head-mounted display (HMD), which displays detailed technical manual in 3D graphics through computation. Virtual information is then mapped to the real world which empowers the personnel with increased amount of information. The fusion has a potential issue of cognitive overload, where the personnel is overloaded with data, graphics and information that makes it difficult for them to complete their desired tasks [2]. Interactive through this information overload can be a challenging task, various data inputs such as gesture, voice and eye movements need to be monitored to provide a holistic real-time experience for the user.

For an overall maintenance system, the personnel might have a simpler interaction, but the details increase while performing micro-level maintenance tasks which require increased 3D graphics, data processing, efficient management between the user & virtual world and lastly, accurate target tracking for real-time perception by the user [2]. With these major factors considered, the study focuses on reducing the load on maintenance personnel during micro-level tasks; the increased amount of 3D graphics and data processing occurs due to the intricate details that exists within these tasks.

For micro-level tasks, the HMD devices are pre-loaded with all the process steps that exists within the infrastructure, the virtual process flow interact with the user by receiving inputs throughout the process which then modify the virtual instructions provided. This project focuses on designing effective input parameters that alleviate information load, the data will then be processed based on the specific user inputs making the overall process user defined where the virtual information reacts to the inputs provided by the real world's information. This process benefits the user with restricted data flow, efficient interaction between the virtual world as the amount of accessible information is restricted and lastly, the accuracy of target tracking increases with the decreased size of input metadata. These factors improve the overall user experience and enable the maintenance personnel to focus on the intricate tasks without clutter and cognitive overload.

1.3 Machine Learning Introduction

Machine Learning (ML) which is a branch of Artificial Intelligence (AI) has been an evolving field of study that aims to enable machines with the essential skillsets to perform tasks using intelligent software. ML algorithms are primarily based on statistical learning methods that use data to learn, these algorithms utilize concepts from computer science and statistics to develop machine intelligence. Figure 1 illustrates the various disciplines within machine learning such as statistics, pattern recognition, databased, neurocomputing, data mining and Knowledge Discovery from Data (KDD) [3].

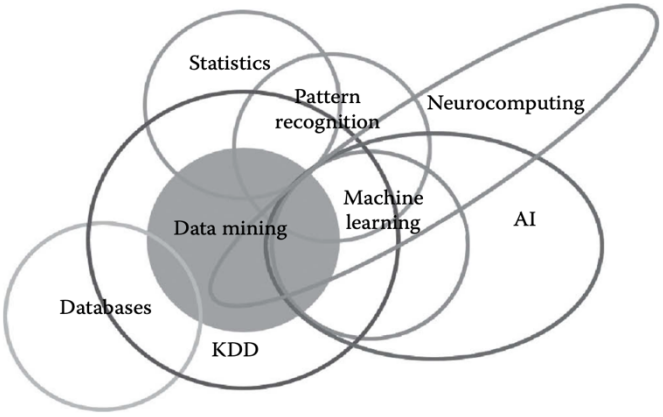


Figure 1: Different Disciplines of knowledge and the disciple of machine learning [3]

Within ML ecosystem, there are various learning techniques that are used to achieve the desired output. The major techniques are supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning. Each of these learning methods have a set of rules and data input requirements for functioning, figure 2 shows the various machine learning techniques and their required data.

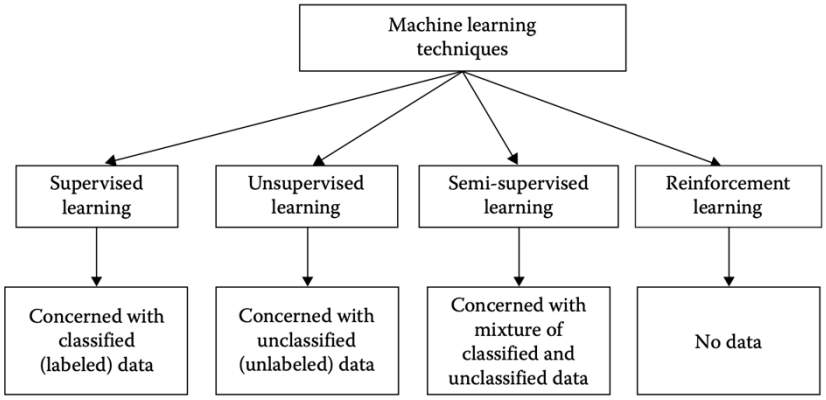


Figure 2: Machine Learning Techniques and their Required Data [3]

As seen in figure 2, the first learning technique is supervised learning where the target is to identify a function and develop predictions based on the training data that is labeled. The training dataset consists of input and output vectors defined as X and Y respectively, the output vector Y consists of the labels for each training example which is used by the learning algorithm to predict outcomes for new input vectors X [3]. This learning technique can be further split into two categories namely regression and classification, regression algorithms predict the target numeric values and classification algorithms predict the classes of the input vectors.

The next learning technique is unsupervised learning where the training data is not labelled, the target is to recognize the pattern from the hidden structure. Mot unsupervised learning cases pertain to big data as these are larger unstructured datasets that require further analysis to understand the basic structure. These algorithms are used for three major purposes, data visualization, feature extraction and anomaly detection [3].

The semi-supervised learning technique as the name suggests is a combination of supervised and unsupervised learning techniques, the dataset have a mixed characteristic of both labeled and unlabeled data [3]. These algorithms are able to classify the future data with a higher accuracy when compared to the labeled algorithms, this type of learning is effective for large datasets that are unlabeled while there are instances within those data frames that are labeled.

The last learning technique is reinforcement learning which doesn't require any data input, but these algorithms learn using observations gathered from the interaction with the environment. The learning "agent" constantly looks to maximize the reward and minimize the risk, the reinforcement learning goes through a specific process flow. Figure 3 illustrates the process flow of a typical reinforcement leaning algorithm.

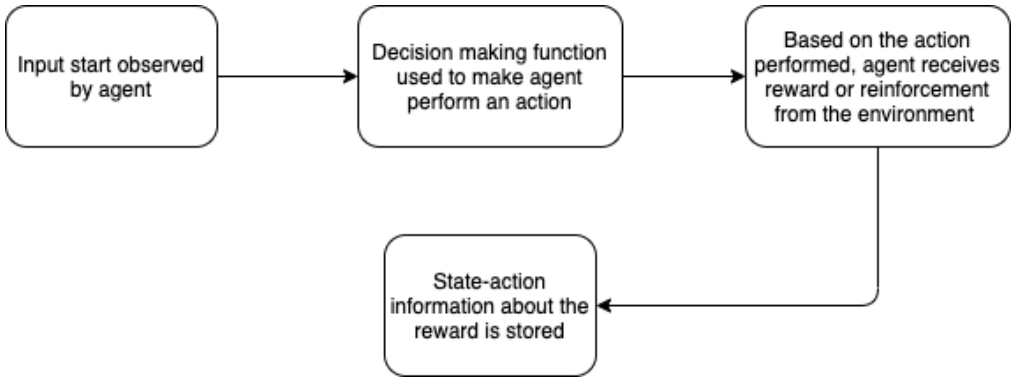


Figure 3: Reinforcement Learning Process Flow

1.4 Mission Objectives

The purpose of this research study is to explore how AI can be integrated into existing AR based instruction modules. The study is currently limited to enhancing maintenance procedures for maintenance personnel, there is an increased potential with predicting maintenance procedures along with the integration of computer vision algorithms for object recognition and Reinforcement Learning (RL) algorithms to optimize the maintenance procedures further. With the completion of this thesis study and future considerations, the study hopes to accomplish the following objectives: -

→ **OBJ-AI ECO-001:** To explore dynamic techniques within existing AR instruction modules using ML algorithms

→ **OBJ-AI ECO-002:** To analyze the performance of various ML algorithms used for process predictions

→ **OBJ-AI ECO-003:** To integrate the design with computer vision algorithms that utilize Convolutional Neural Networks (CNNs)

→ **OBJ-AI ECO-004:** To develop an RL technique that has the ability to validate and optimize maintenance procedures

1.5 Scope of Research

Within the aerospace sector, regular maintenance procedures play a large role in ensuring the aerospace systems are consistently to the best of its ability. Within the aircraft industry, most maintenance procedures are scheduled and require the use of long maintenance manuals, AR based instruction modules have enabled maintenance personnel with ease of access and reduced workload [4]. Within the space industry, there are routine maintenance within the International Space Station (ISS) which require micro-level planning where virtual instructions provides astronauts with accurate procedural directions and instructions. The results of this thesis intends to improve upon the maintenance process for such micro-level components, integration of AI systems can further empower maintenance personnel to perform tasks with higher accuracy and decreased times.

2. Background

The design process developed in this thesis are derived from various existing technologies and researches. Most of the existing designs for model-based instruction include the use various AR techniques, some have the option of hand-held tablets while others utilize the HMD. The major designs reviewed are Airbus's AR use-case used in their shop floor, computer vision validation system for AR based maintenance procedures and Boeing's Mixed Reality (MR) use-case.

Using AR in A400M shop floor by AIRBUS [5]

Airbus has developed an AR application to generate assembly instructions, the project initiated was call "Project MOON (asseMbly Oriented authOring augemeNted reality)" which initially developed by AIRBUS Military in 2010. The vision of this project was to create Work Instructions (WI) using 3D data from industrial Digital Mock-Up (iDMU) which generates assembly instruction using AR, this application was tested on the electrical harness routing in the frame 36 of the AIRBUS A400M.

The system architecture for this project was to use a tablet PC equipped with a webcam, images of the work area were captured in real time with webcam and, the CAD virtual elements extracted from the iDMU were superimposed on top of the real time image. Figure 4 illustrates a working example of the test performed.

Figure 4: Moon Project Workshop Testing [5]

The AR system receives inputs from two inputs to generate the work instruction as an output: -

1) Real time world information in the form of images, as well as artificial markers used for calibration are captured using the webcam
2) Virtual information defined in the iDMU

The AR system then utilizes three subsystems: 3D processing, positioning and information integration, which use the inputs from the real and virtual information. Based on the tests conducted, it was concluded that using AR technologies improve the use of iDMU to visually represent the process information. The results showed, model interpretation was easier for personnel and the possibility of errors were minimized during the execution process.

Mixed Reality Training by The Boeing Company [6]

Through this study, Boeing was looking to fuse various datapoints into a single coherent profile of any trainee that can improve future training accuracy and retention, with these datapoints the efficiency of training can be increased with the use of fewer resources. The study performed was to analyze AR as a delivery for work instructions in a manufacturing workspace.

The design methodology for this project was to evaluate three different methods of presenting work instructions, desktop Model-Based Instructions (MBI), tablet MBI and tablet AR. The tablet AR method which is the tablet MBI but represents the work information to the trainee in AR, figure 5 illustrates the tablet AR used for this study.

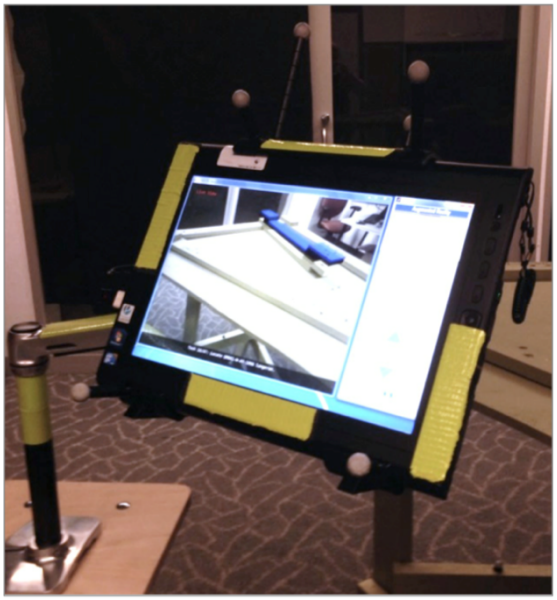


Figure 5: Tablet AR Method [6]

The study was performed to mimic a traditional work cell for manufacturing and the selected testing area was the wing location, the full task consisting of 46 different steps which were completed twice by the trainees. The performances of the trainees were compared based on the traditional MBI with AR instructions, the three major areas of interest were: -

1) First time quality
2) Fastest time
3) Worker efficiency

Based on the results assessed on these parameters, it was evident that the use of AR for work instruction delivery can increase first time quality while reducing time on task. There was also indication that AR allowed trainees to focus more during the tasks and lastly, specific tasks also strongly benefitted from the use of AR.

Computer vision validation system for AR based maintenance procedures [7]

This study looks to introduce an automatic validation system that validates the users' actions for correctness of work, this design also looks to validate the completeness of each step within the maintenance procedure. The technology introduced in this design was to utilize computer vision algorithms that would evaluate each step of the of a maintenance procedure, the validation was completed by comparing an image of the final status of the machinery and a virtual 3D representation of the expected final status. Figure 6 outlines the methodology followed in this study to achieve an automated validation.

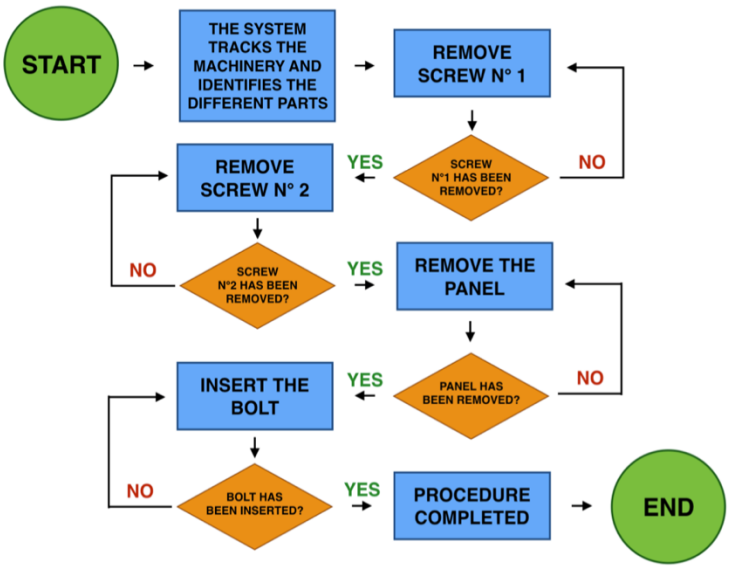


Figure 6: State Diagram Describing the Steps of a Maintenance Procedure [7]

The solution architecture utilizes OpenCV for the use of computer vision algorithms, the algorithm has the capability of evaluating each step of the maintenance procedure and assess the accurate completeness. The study also proposes various mitigation strategies as it identifies motion in the scene and discriminate between motion in the scene or a change of position with the camera. To conclude, the proposed method solves an open ended within AR based instruction methods which can ensures maintenance personnel are completing the tasks with accuracy.

3. Research Materials

3.1 Apparatus

The main apparatus used for this research study is the inverter of the MaxFlight Full Motion Simulator from the Mixed-Reality Immersive Motion Simulation (MIMS) Laboratory, this test subject was used as it contains the required complexity to conduct micro-level maintenance analysis. A local coordinate system was established to ensure precise measurements were taken on the test subject which were used for CAD modelling and, developing the training dataset for the ML algorithms. Figure 4 shows the inverter used for this research, as seen in the image there are various intricate parts within the inverter, the process of disassembling the inverter was specifically analyzed for this thesis.

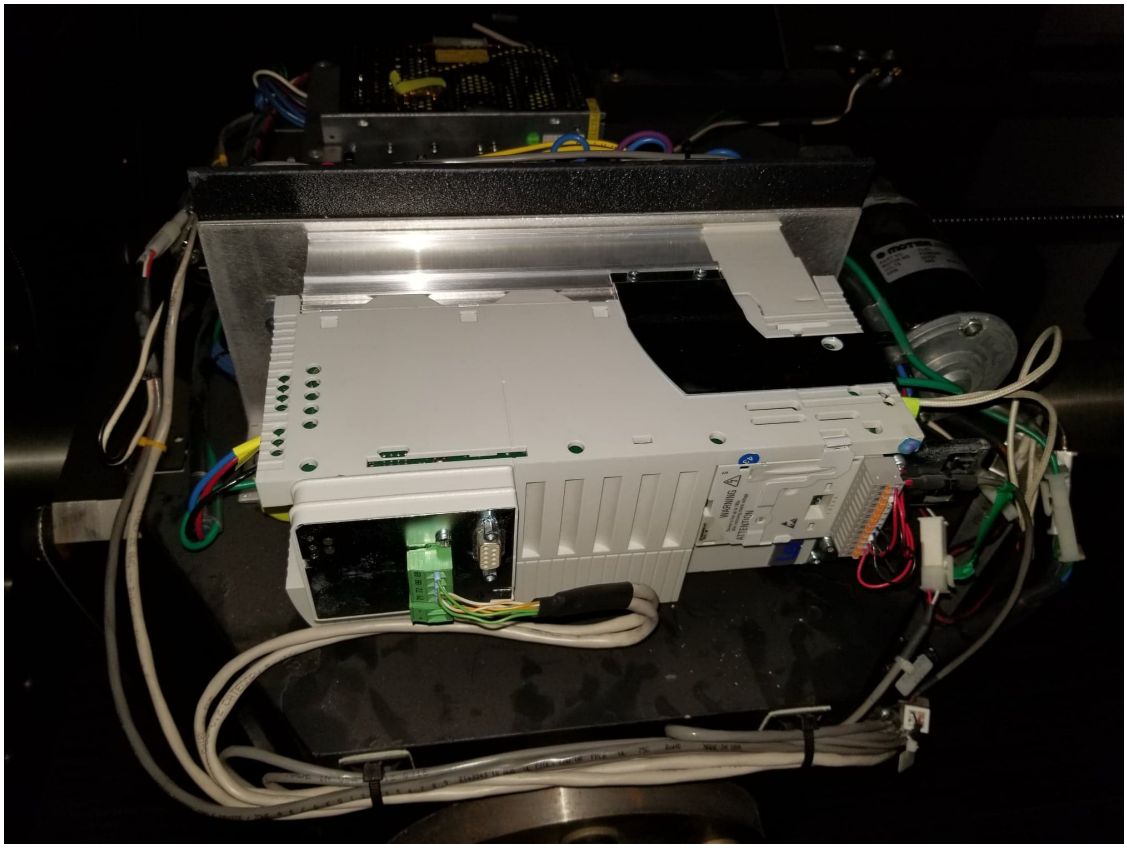


Figure 7: Inverter of the FlightMax Fill Motion Simulator

3.1.1 Test Subject Coordinate System

As mentioned in section 3.1, the measurement process was initialized by establishing the local coordinate system; this process was very crucial as the coordinate determined the position of each component within the inverter which was then correlated to a specific step within the infrastructure of the inverter. There were 13 unique steps defined within the process of disassembling the inverter, each step has precise X, Y and Z axis points which are the input features for the training dataset. Figure 5 exhibits the established local coordinate system and the initialized point which is considered to be the Centre of Gravity (CG).

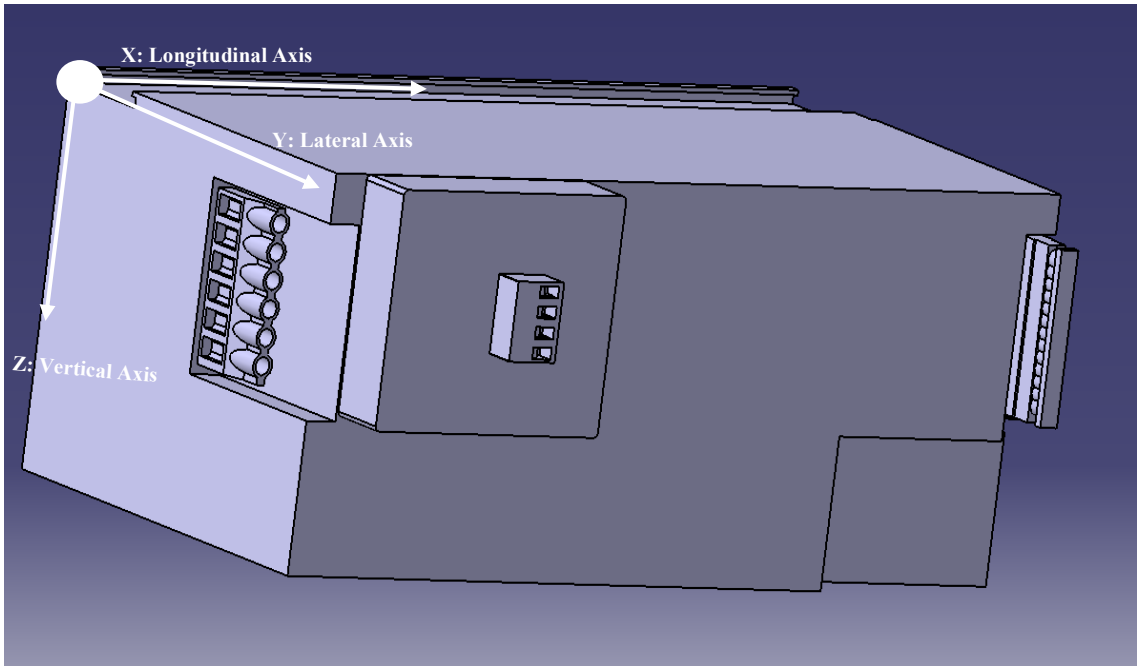


Figure 8: 3D CATIA Model of the Inverter

The local coordinate system follows the same system as that of an aircraft with all measurements taken from the CG position; the X axis referred to as the longitudinal axis which runs along the length of the inverter, the Y axis referred to as the lateral axis runs along the width of the inverter and lastly, the Z axis referred to as the vertical axis runs along the height of the inverter. The measurements of each component were taken using the combination of an electronic calliper and an inch tape.

3.2 Hardware and Software

The major hardware components use for design and analysis are the following: -

1) Desktop Computer with Windows 10 Professional
2) Mac Book Pro 2015

Among these devices, the desktop computer was used to develop 3D CATIA models of the inverter as seen in figure 5 and the Mac Book was used to perform analysis on the ML algorithms.

The software component used for this study is: -

1) Python 3.0 on Jupyter Notebook Integrated Development Environment (IDE)

Python 3.0 was used to run the selected ML algorithms and develop the framework of a prototype; the software was run on the Jupyter Notebook platform IDE.

4. Design Concept

4.1 Design Objective and Requirements

The main reason conducting this study was to implement an innovative approach for virtual instructions by utilizing a specific class of ML algorithms, that can enable assembly or maintenance personnel to work in a non-linear manner while performing tasks. Development of this application provided a foundation for an AI ecosystem that can be later integrated into MS HoloLens for AR applications within aerospace manufacturing. This project was completed within the span of roughly about 2.5 months.

Currently, micro level virtual instructions within AR have a linear process flow where the user is only allowed to follow a specific path of steps, the user can have multiple instructions at a specific step within their overall process, but they need to follow the pre-determined path. The visual demonstrations and instructions are then synchronized to this pre-determined set of procedures enabling the user to complete the required tasks. Although this technique is very effective as it allows the user to go through all the required steps within a process flow, the major issues with this approach are: 1. The user is restricted to a specific process flow due to the linearized approach, which leads to a lack in agile decision making and 2. Large datasets that are constantly being processed in real-time, this creates data latency issues leading to inaccurate visual instructions.

Based on the afore-mentioned issues, the major requirement is to identify the right step that corresponds to users' input within the overall process flow in the most efficient manner possible. The proposed technique explores a more dynamic method that takes user preferences and their interaction with the overall process infrastructure into consideration. The major modification made, would be to enable the user to randomly select a specific object within the overall infrastructure and determine which particular step that object belongs to using the theory of probability. While there are various ways this problem can be approached, a data science and ML method amalgamates the use of foundational concepts of probability and statistics such as the permutation, statistical measures and the use of effective ML algorithms that optimize probability theories. This technique solves the linearity issue, but there are large amount of possibilities within the process flow infrastructure that could exponentially increase based on its size and complexity. By using the statistical concepts solely, complex and time-consuming stochastic optimization techniques will have to be utilized to optimize the probabilities, with the addition of ML methods

to the optimization process is accelerated as most algorithms have the framework to optimize which increases the accuracy of prediction.

One of the major steps with applying probabilistic theories to identify permutation, which finds the number of ordered sequences of the elements within a set of elements, the following equations best describes permutation, mean and standard deviations of various elements in a set S.

$$n! = n \times (n-1) \times (n-2) \times \ldots \times 2 \times 1 \quad (1)$$

$$\mu = \frac{\sum_{i=1}^{n} x_i}{n} \quad (2)$$

$$\sigma = \sqrt{\frac{\sum (x-\mu)^2}{n}} \quad (3)$$

Equation 1 provides the number of permutations of n different elements; this concept is a result of the multiplication rule. The permutation is developed by placing an element in the first position of the sequence from n elements, then selecting an element from the remaining $n-1$ to be placed in the second position. This process is followed until there are no remaining elements within the set S [8]. Due to the stochastic (random) nature of the use case, additional statistical measures such as understanding mean (2) and standard deviation (3) provide a better understanding of the dataset and act as key measures that provide statistical approximations and help identify the right optimization algorithms. Stochastic optimization algorithms are the most viable for this particular problem, as the data is random and statistical analysis will be complete before applying these optimization techniques. Multistage stochastic algorithms such as stochastic gradient descent which aim to find a sequence of decisions that minimize the expected function is a possibility.

ML algorithms have the ability to learn effectively from training data, most algorithm have the capability to understand patterns from stochastic datasets, apply stochastic optimization techniques and provide accurate predictions. For this particular use-case supervised learning is selected, this learning technique can be further divided into classification and regression. Classification algorithms are selected for this research study, these algorithms classify the data into categories based on their labels [8]. As seen in the description, ML algorithms can handle the complex stochastic nature of the use case and optimize their respective cost functions which are the foundational working mechanics of any ML algorithm.

Based on the problem statement stated above, table 1 provides an outline of the requirements this project intends to achieve with the development of the AI ecosystem. These requirements act as guidelines for the application design, meeting these requirements would ensure a viable solution that solves the current issues within model-based instruction techniques.

Table 1: Design Requirements Outlined

REQ – AI ECO - 001	A dynamic approach for virtual instructions shall be explored. → This is to provide additional agility to the user during maintenance tasks. → This also has the upside of decreasing the processing speed and improving the accuracy of the visual instructions.
REQ – AI ECO - 002	An AI ecosystem application shall be prototyped as a proof of concept. → This is to ensure a viable proof of concept is developed and the ecosystem can be integrated with the visual instructions.
REQ – AI ECO - 003	Accurate predictions based on random user input shall be provided. → This is a good measure of robustness for the ML algorithms. → This also provides a good understanding of the application's scalability.
REQ – AI ECO - 004	Performance measures shall be analyzed. → This ensures the best ML algorithm is used to satisfy the requirements. → This also provides a better understanding of the impact created by the proposed application.

With the requirements defined, a process flow for the design phase is presented in the following flowchart (figure 9). The process outlines the major tasks to be achieved within the overall project and focuses on the core attributes that enable dynamic workflows for its users.

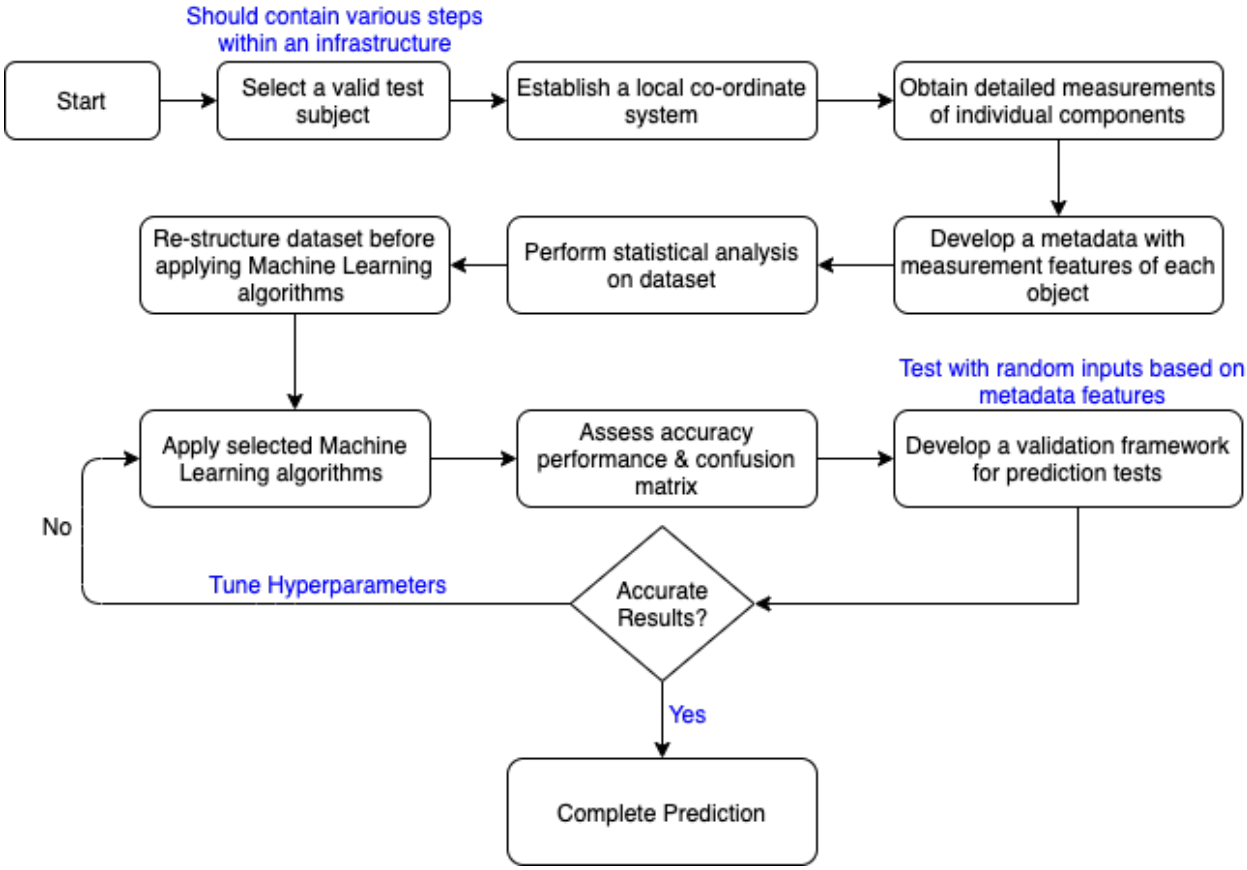


Figure 9: Process Flow for the Proposed Machine Learning Approach

4.2 Methodology

The design methodology of this thesis involves designing an AI ecosystem through classification-based algorithms, to further assess the foundational theory of this approach, concepts of classification-based algorithms were analyzed and the optimization theories within the four selected algorithms were further examined. Classification is a supervised learning process of predicting the class from a given dataset, there are various algorithms within classification, and they can be split into two major categories; lazy learners and eager learners [8]. Eager learner algorithms have been used for this thesis, the algorithms within this category develop a specific hypothesis that have the potential to cover the entire instance space, with a given training dataset these algorithms construct classification models prior to seeing the test dataset which is considered to be the new data [9].

The major eager learners selected for this solution architecture are: 1. Support Vector Machine (SVM), 2. Logistic Regression and 3. Random Forests (RF). All these algorithms have unique learning technique and prediction methodologies. Each algorithm was segregated into four major categories core concepts, process flow, optimization methods and hyperparameters, this provided a good understanding of their mechanics and their impact on the final prediction. Lastly, the training dataset developed for this project incorporate 3 input features and 13 different classes of labels; the input features are the X, Y and Z coordinates of each object and their labels correspond to the respective step they belong to within the infrastructure.

4.2.1 Support Vector Machine

Support Vector Machine (SVM) is a classifier that uses a certain bias which is primarily defined by a hyperplane separation, the objective of this algorithm is to identify the hyperplane in the N-dimensional space that classifies the data points into specific labels. The major concepts within this algorithm are separating hyperplane, maximum margin hyperplane, soft margin and kernel function [10]. Figure 10 below illustrates the key parameters pertaining to an SVM algorithm.

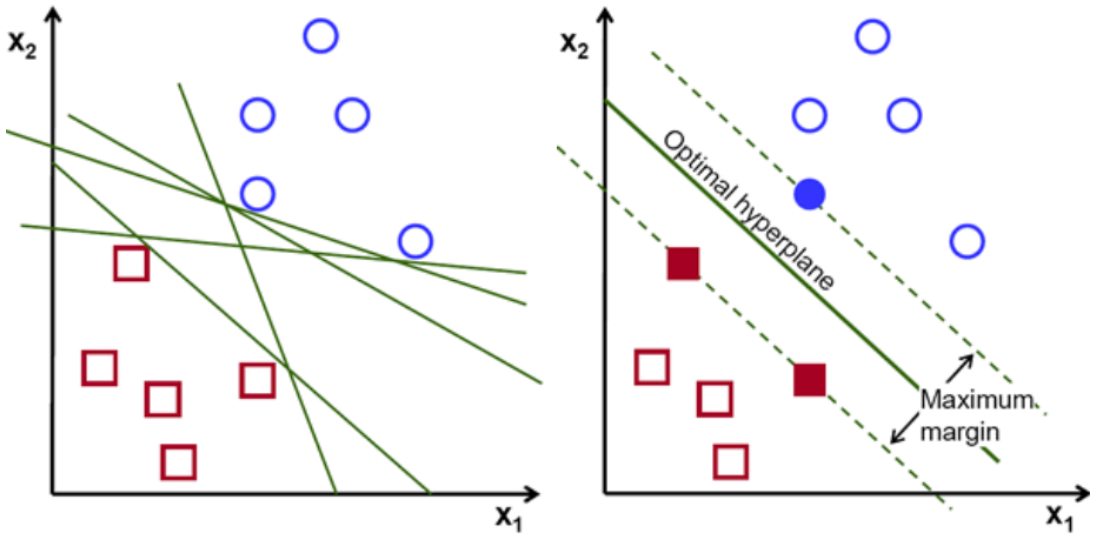


Figure 10: SVM Hyperplane Working Mechanism [10]

The separating hyperplane, which is used to separate between the classes the separation can have various possible solutions. The dimension of the hypermeter is an important factor in the separation

method which mainly depends on the number of features, equation 4 shows the relationship between the total number of input features and the hyperplane [10].

$$N_{Hyperplane} = Number\ of\ Features - 1 \tag{4}$$

Based on equation 4, if the number of input features is 2 then the hyperplane becomes a line and if the number of input features is 3, then the hyperplane becomes a two-dimensional plane.

The main objective of this algorithm is to identify a plane that maximizes the margin between the training instances, the hyperplane that yields in a maximum margin is considered to be the optimized. The margin is established as the distance from the separating hyperplane to the nearest expression vector, SVM ensures to select the maximum-margin hyperplane which maximized the accuracy of the predictions [11]. Soft margin is another principle within SVM, while the hyperplane has the potential to split the classes effectively there are still errors that can occur within the dataset [11]. SVM's way of dealing with these errors is to add a soft margin allowing certain data points to be misclassified without affecting the final prediction results, this flexible model's objective is to strike a balance between hyperparameter plane violations that occur due to misclassifications and the size of the margin [8]. The last concept within SVM is the kernel function, this aspect of SVM becomes significant when there isn't a single point that can separate the two classes [11]. The kernel function increases the dimensionality of the data from the original one, there are various types of kernels that can be used within SVM, but the gaussian Radial Basis Function kernel was used as the hyperparameter for this particular use-case.

Since there are 3 input features, the non-linear SVM hypothesis and cost function was considered. Equation 5 and 7 outline the hypothesis and cost function (hinge loss) for this algorithm variant [12].

$$h_\theta(x) = f(x) = \begin{cases} 1, & if\ \theta^T f >= 0 \\ 0, & otherwise \end{cases} \tag{5}$$

$$\theta^T x = \theta_o x_o + \theta_1 x_1 + \cdots + \theta_j x_j \tag{6}$$

$$J(\theta) = C\left[\sum_{i=1}^{m} y^{(i)} Cost_1(\theta^T(f^{(i)}) + (1 - y^{(i)}) Cost_0(\theta^T(f^{(i)})\right] \tag{7}$$

$$Cost_1 = max\,(0, 1 - \theta^T x) \tag{8}$$

$$Cost_0 = max\,(0, 1 + \theta^T x) \tag{9}$$

$$K(a, b) = exp\,(-\gamma(a - b)^2) \tag{10}$$

$$\gamma = \frac{1}{number\ of\ features \times \sigma^2} \tag{11}$$

Based on the cost function J(θ) in equation 6, there are various deciding parameters within the SVM algorithm; the θ^Tvalue is feature weight vector which is used to calculate the decision function $\theta^T x$, if the decision function's results is positive the prediction class is positive (1) if not it belongs to the negative class (0) and with a smaller weight vector, the margin hyperplane increases [12]. The next feature is, the C value which is known as the regularization term which controls the complexity of the model, it impacts the width of the margin hyperplane and is critical when dealing with non-linear dataset that have increased possibilities for misclassification and margin violation [8]; the C value selected for this project was 10 as it provided the most optimized solution without overfitting to the training data. Lastly, the f value represents the kernel function, there are various kernel functions within the SVM algorithm; the gaussian RBF kernel was used in this project, they are calculated with Euclidean distances between two vectors and the γ parameter based on equation 10 [8] describes the smoothness of the function and the influence on a single training example.

To conclude, SVM's cost function is convex and a Sequential Minimal Optimization (SMO) was used, as training an SVM increases the dimensionality of the solution to a very large quadratic programming optimization problem. SMO disintegrates the large quadratic programming problem into series of smallest possible problems that are solved analytically. This process avoids heavy matrix computation and increases the computational time as it scales the data efficiently, making it much faster than other chunking algorithms [13].

Based on the concepts of SVM, the selected hyperparameters and optimization techniques; the python script modelled for this project is presented below: -

```
from sklearn import svm
clf = svm.SVC(kernel='rbf',gamma = 'scale', C=10)
clf.fit(X_train, y_train)
y_pred_SVM = clf.predict(X_test)
```

4.2.2 Logistic Regression

The logistic regression algorithms is one of the most common classification, it estimates the probability of occurrence of an event by fitting to a logistic curve [14]. Logistic regression is inherently a binary classifier, there are various strategies by which binary classifiers can be used as a multiclass classifier [8]. Since the training dataset used has 13 unique classes, logistic regression was transformed using the softmax regression strategy which computes the score at any

instance for each class, then estimates the probability of each class by applying the softmax function (12) [8].

$$p_k = \frac{exp\ (s_k(x))}{\sum_{j=1}^{K} exp\ (s_j(x))} \tag{12}$$

Based on the softmax function (12), the K value represents the number of classes, s(x) is a vector that contains the scores of each class for the instance x and p_k is the estimated probability that instance x belongs to, in class k based on the score of each class for that instance. Since this algorithm is based on the concept of probabilities, the values are expected to remain with the range of 0 and 1 which are not an ideal scenario for extreme values. The method logistic regression uses to constraint the range of any input values is by utilizing a sigmoid function whose range always remains within the range of 0 and 1 [15]. The logistic function is a type of sigmoid function that constraints the output to remain within the specified range regardless of the extreme input values. Equation 13 [15] outlines the logistic or hypothesis function and the figure 11 illustrates the mechanics of the logistic function.

$$h_\theta(x) = \frac{1}{1+exp\ (-\theta^T x)} \tag{13}$$

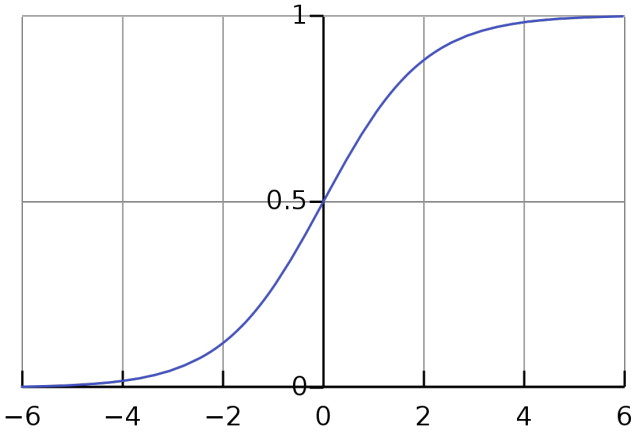


Figure 11: Logistic Function Representation [15]

Based on the theory established, the logistic regression algorithm pivots on the concept of probability and the objective for optimization is to adjust the θ parameter that yields the highest probabilities for positive instances and lower for negative instances that corresponds to minimizing the cost function in that process [16]. The cost function for a single instance (14) uses a logarithmic

function, this function ensures the output to be an estimated probability of 1 for a positive instance and, 0 for a negative instance which results in minimizing the overall cost function [15].

$$Cost(h_\theta(x), y) = f(x) = \begin{cases} -log\,(h_\theta(x)), & if\ y = 1 \\ -log\,(1 - h_\theta(x)), & if\ y = 0 \end{cases} \quad (14)$$

A loss function is introduced to increase the penalty on the cost function when the model predicts 1 while it actually 0 and vice-versa, the loss function used for logistic regression is logistic loss. The overall cost function of the model (15) is the averaged cost over all training instances [15].

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\, log\left(h_\theta\left(x^{(i)}\right)\right) + (1 - y^{(i)}) log\,\left(1 - h_\theta\left(x^{(i)}\right)\right)\right] \quad (15)$$

The loss function for this algorithm results in a convex function which uses the gradient descent optimization technique to minimize the overall cost function. The optimization solver used for this project is stochastic average gradient descent that uses an aggregated gradient approach and random sample data. The last hyperparameter used is the regularization term, which is denoted with a C; the value selected was 10 which is similar to SVM for consistency purposes. The C value is inverse regularization strength, there are two major regularization types and L2 is used for this algorithm; L2 penalizes the large coefficients which is a constraint that prevents overfitting, the L2 equation (16) has another parameter λ that controls the weight of the constraint that balances regularization and prevents underfitting [15]. Lastly, m represents the number of samples and n represents the number of features.

$$L2: J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right) + \frac{\lambda}{2m}\sum_{j=1}^{n} {\theta_j}^2 \quad (16)$$

Based on the concepts of the logistic regression algorithm and optimization hyperparameters; the python script modelled for this project is presented below: -

```
from sklearn.linear_model import LogisticRegression
LogisticReg = LogisticRegression(multi_class = 'multinomial',solver = 'sag', C=10)
LogisticReg.fit(X_train, y_train)
y_pred_logReg = LogisticReg.predict(X_test)
```

4.2.3 Random Forests

The random forest is an ensemble learning algorithm that uses a group of predictors, this algorithm specifically uses an ensemble of decision trees which has been considered to be one of the most powerful ML algorithms. The overall process is to receive recommendations from

individual decision and based on the recommendation, the final prediction is made through a voting procedure. Since random forest uses the averaging process, decision trees are an ideal candidate due to its ability to have low bias and high variance which increases the accuracy when multiple decision trees aggregated [17]. An important concept within random forest is bagging method, which uses the same training algorithm (in this case decision trees) for every predictor but trains them on a replaced random subset of the original training set.

Once the predictors have been trained, the ensemble makes the prediction for a new instance by aggregating all the predictions made initially and for classification, the aggregation function is the statistical mode. The random forest algorithm adds an additional layer of randomness to the bagging method, it examines the best feature amongst the random subset of features which results in a greater tree diversity. The overall concept of this algorithm is illustrated in figure 12 [18].

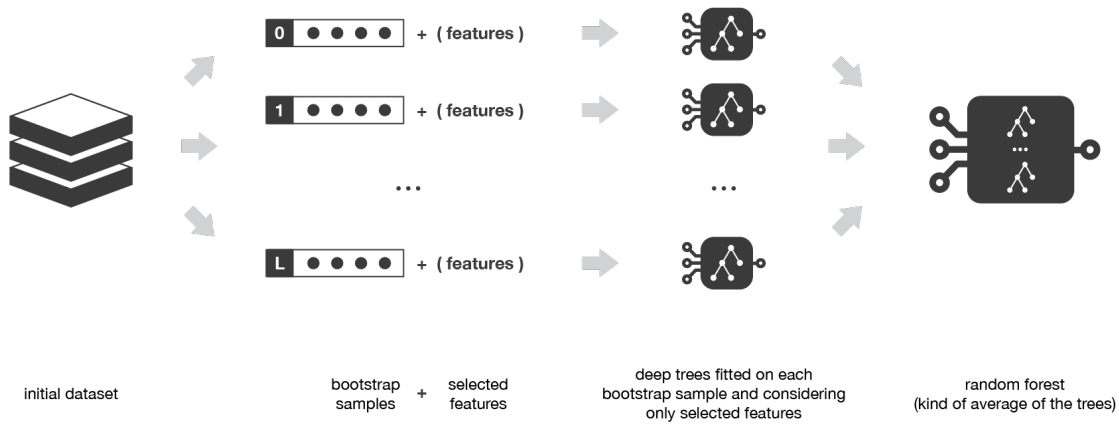


Figure 12: Random Forests Working Mechanics [18]

Although the random forest algorithm is black box model, the major theories used within this algorithm pertain to decision trees. The two aspects within decision trees are, impurity measure and training algorithm; the Gini impurity was used for this project and the Classification and Regression Tree (CART) algorithm was used to train the individual trees [8]. The Gini impurity measure applies to each node within the tree, a node is considered to be pure if the class of all the training instances it applies to is the same; the Gini measure for such a node is 0. The Gini impurity equation (17) [8] outlines the key parameter which is the ratio of the class of instances within the training instances in a specific node.

$$G_i = 1 - \sum_{k=1}^{n} {p_{i,k}}^2 \quad (17)$$

This impurity measure plays a critical role within the cost function of the CART algorithm (18), which is a greedy algorithm which looks to find the optimum split at the top level but does not verify the splits at lower levels. The algorithm initially splits the training set into two subsets using the single feature k and a threshold t_k; these values are a pair (k, t_k) that produce the purest subsets [8]. The same logic is recursively applied until the maximum depth hyperparameter is reached.

$$J(k, t_k) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right} \quad (18)$$

Where $G_{left/right}$ is the measure of impurity of the left/right subset and $m_{left/right}$ is the number of instances in the left/right subset, similar to other cost functions in the ML ecosystem the algorithm looks to minimize the cost function with hyperparameters that optimize the process [19].

Lastly, the hyperparameters used to constraint the algorithm are number of estimators, maximum depth, random state and maximum features. The number of estimators is the measure for the total number of trees in the forest, a value of 100 was selected as the random forest algorithm performs better with deeper trees. Maximum depth is the depth of each tree, since the CART algorithm was used this parameter plays a critical role as the stopping criterion. Through an iterative process defined by the bias and variance trade-off, the final value was determined to be 4. Random state is a parameter that controls the random number generator used, to ensure the algorithm is deterministic the value was set to 42. Lastly, maximum features is the parameter that the random forest algorithm considers when looking for the best split, the value was based on the maximum features equation (19) [8] which yields to be approximately 1.73≈2.

$$maximum\ features = \sqrt{Total\ number\ of\ features} \quad (19)$$

Based on the concepts of the random forest algorithm and optimization hyperparameters; the python script modelled for this project is presented below: -

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators = 100, max_depth=4, random_state = 42)
rfc.fit(X_train, y_train)
y_pred_rfc = rfc.predict(X_test)
```

4.2.4 Design Summary and Process Flow

Based on the ML design architecture built using the algorithms mentioned in section 4.2, the summary table outlined in table 2 contains the core features of each algorithm. The features are namely; cost function, hyperparameters used and built in optimization method.

Table 2: Algorithm Design Summary

Algorithms	Cost Function	Hyperparameters	Optimization
Support Vector Machine	$J(\theta) = C\left[\sum_{i=1}^{m} y^{(i)} Cost_1(\theta^T(f^{(i)}) + (1 - y^{(i)}) Cost_0(\theta^T(f^{(i)})\right]$	- Kernel **(RBF)** - Gamma **(Scale)** - C **(10)**	Sequential Minimal Optimization
Logistic Regression	$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log\left(h_\theta(x^{(i)})\right) + (1 - y^{(i)}) \log\left(1 - h_\theta(x^{(i)})\right)\right]$	- Solver **(SAG)** - C **(10)**	Stochastic Average Gradient Descent
Random Forest	$(k, t_k) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right}$	- No. of estimators **(100)** - Maximum depth **(4)** - Random state **(42)** - Maximum features **(2)**	Randomized Feature Selection

The summary table above, outlines the core variables used for this project within each algorithm and the final values used for each hyperparameter have also been highlighted. The ML design approach eliminates the use of additional optimization techniques, all the algorithms selected for this design have a built-in technique that minimizes the cost function and provides predictions efficiently.

Each algorithm has a micro-level process that enables it to provide predictions, they follow similar preliminary steps, but they operate based on their respective cost functions and the designated hyperparameters. Figure 13 presents the process flowchart for the ML ecosystem developed with each algorithm.

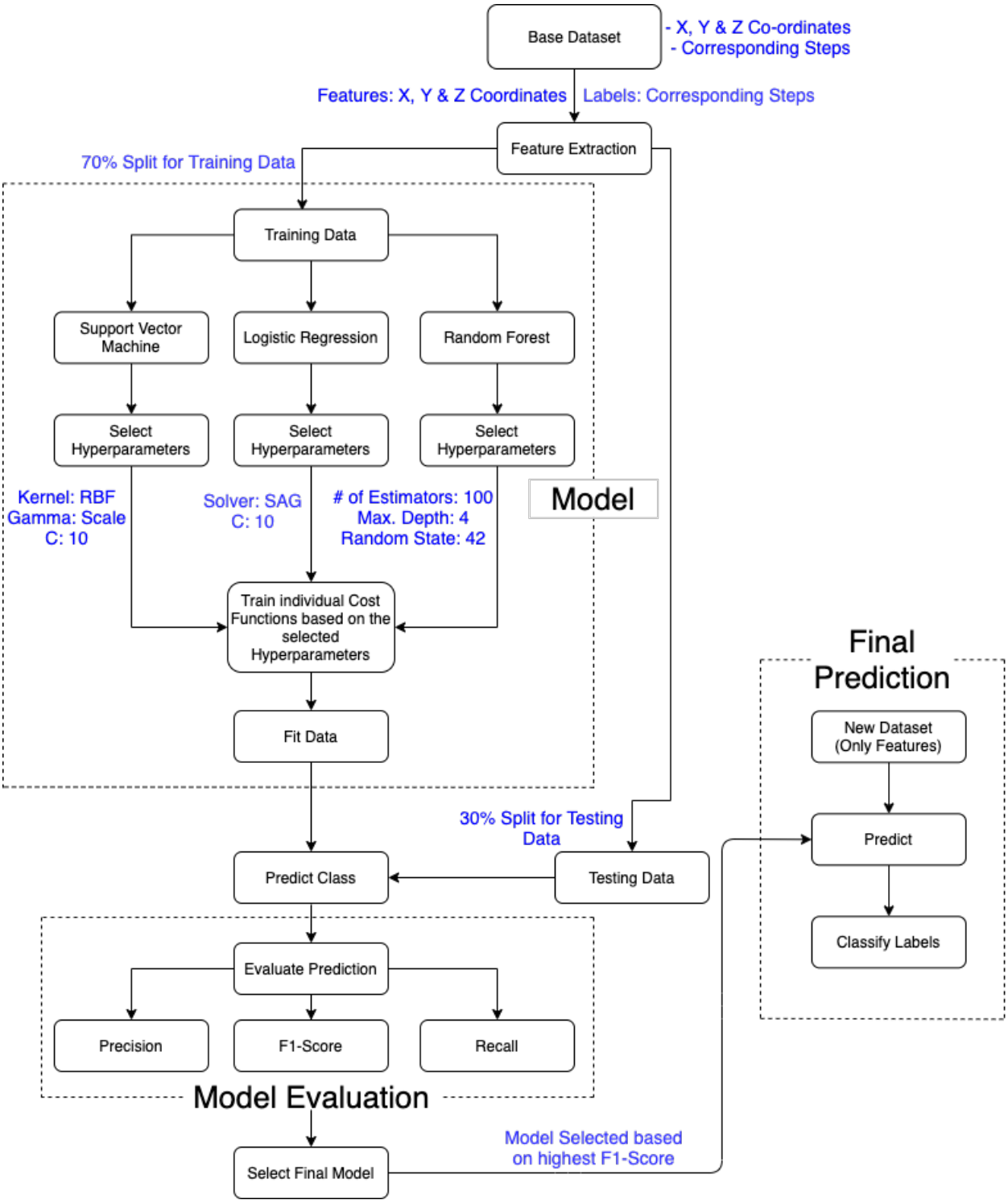


Figure 13: Overall Machine Learning Solution Architecture

5. Results

Classification algorithms have three major performance measures precision, recall and F1-score. Each of the algorithm in section 4.2 were analyzed separately on each of performance measure, these metrics indicate the accuracy of the model's prediction and the algorithm with the best measure was selected for the final prediction test. To further understand the importance of each performance measure, the measures were mathematically and theoretically analyzed.

The four major variables these measures rely on are True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN) [8]. Precision is the accuracy measure of positive predictions; recall is the ratio of positive instances that are correctly detected by the classifier and F1-score is the harmonic mean combination of precision and recall, where harmonic mean gives more weight to low values [20]. Since the project deals with a multiclass classification and each class has varying number of instances, the performance measure values for each class were calculated and the weighted averages for those values were taken as the final measuring value.

The purpose of this experiment was to determine which algorithm would yield in better performance measures; this test was performed using the python 3.0 environment which has various built-in packages that provided the results shown in table 3.

Table 3: Algorithms' Performance Metrics Comparison

Algorithm	**Python Results**	**Weighted Precision**	**Weighted Recall**	**Weighted F1-Score**
Support Vector Machine	precision recall f1-score support 1 1.00 1.00 1.00 5 2 0.50 1.00 0.67 5 3 0.00 0.00 0.00 7 4 0.60 0.60 0.60 5 5 0.60 1.00 0.75 3 6 1.00 1.00 1.00 4 7 1.00 1.00 1.00 38 8 1.00 1.00 1.00 79 9 1.00 1.00 1.00 79 10 0.64 1.00 0.78 9 11 0.00 0.00 0.00 10 12 0.58 1.00 0.74 7 13 1.00 1.00 1.00 7	89%	93%	90.2%

Logistic Regression	precision recall f1-score support 1 0.71 1.00 0.83 5 2 0.38 0.60 0.46 5 3 1.00 0.14 0.25 7 4 0.80 0.80 0.80 5 5 0.75 1.00 0.86 3 6 1.00 1.00 1.00 4 7 1.00 1.00 1.00 38 8 1.00 1.00 1.00 79 9 1.00 1.00 1.00 79 10 0.75 1.00 0.86 9 11 1.00 0.50 0.67 10 12 0.75 0.86 0.80 7 13 0.88 1.00 0.93 7	96%	94%	93.5%
Random Forest	precision recall f1-score support 1 0.83 1.00 0.91 5 2 0.50 0.80 0.62 5 3 0.00 0.00 0.00 7 4 0.62 1.00 0.77 5 5 1.00 1.00 1.00 3 6 1.00 1.00 1.00 4 7 1.00 1.00 1.00 38 8 1.00 1.00 1.00 79 9 1.00 1.00 1.00 79 10 1.00 0.78 0.88 9 11 0.83 1.00 0.91 10 12 1.00 1.00 1.00 7 13 1.00 1.00 1.00 7	95%	96%	95%

As seen in table 3, the precision and recall values are inversely proportional to each other in terms of their values. The precision value is higher, the recall value decreases and vice-versa. This oscillation of values makes it difficult for selecting the right algorithm, as mentioned earlier, the F1-score is a measure that takes the values from both precision and recall and computes the final value. Hence, the algorithm with the highest F1-score was the selected algorithm which was random forest that had an F1-score of 95%. This solution can be related to the theoretical hypothesis of the random forest algorithm, since this an ensemble learning method the cost function is optimized through the aggregation of randomized feature selection the accuracy tends to improve through this process while training various trees in parallel within the algorithm. Lastly, the hyperparameters also played a critical role; with a tree depth of 4 and 100 trees within the algorithm, the precision and recall trade-off has been achieved.

5.1 Design Analysis

There final design was analyzed using two methods, one through visualizing the confusion matrix which evaluates the overall performance of the classifier for each class and the second is to validate the hypothesis by testing the accuracy of the random forest algorithm with new data points. The confusion matrix was analyzed for the random forest algorithm, with an F1-score of 95% it was expected to have a perfect matrix with almost all values across the diagonal. Figure 14 shows the confusion matrix of the random forest algorithm, by analyzing each class specifically it is evident that the algorithm was able to classify all classes accurately with the exception of class 3. Since the overall performance of the algorithm was meeting standards, this issue was neglected as there will have to be further data analysis performed to understand the irregularities within this class.

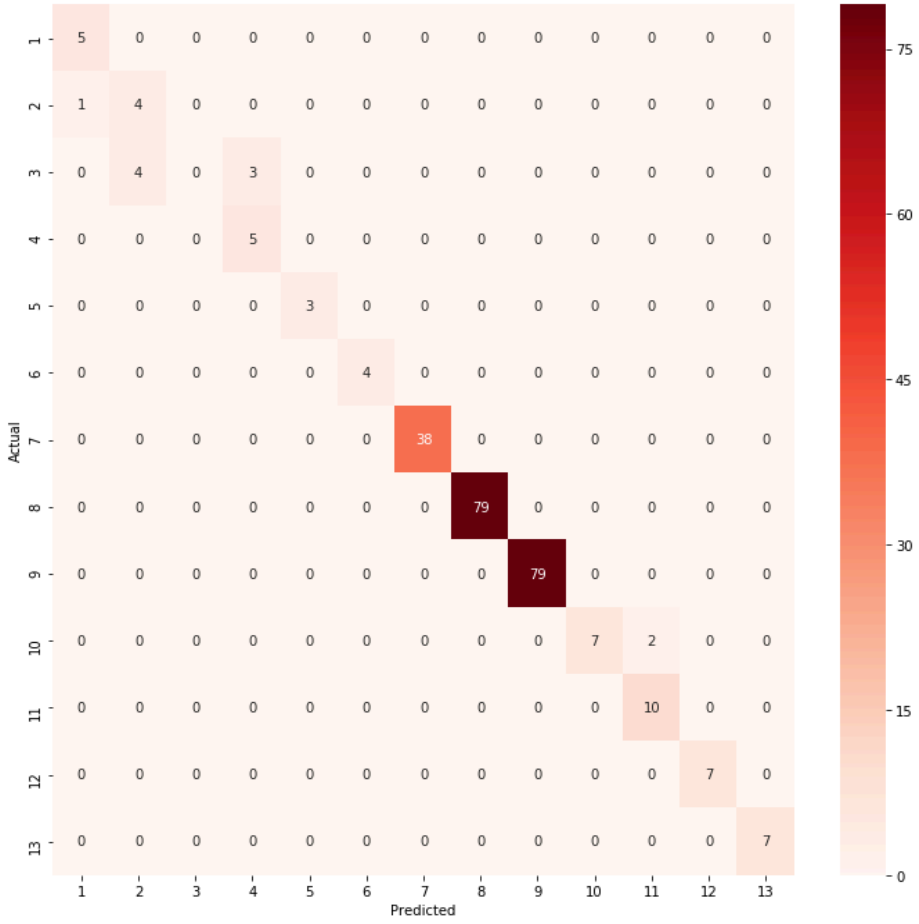


Figure 14: Confusion Matrix of the Random Forests Algorithm

The second method of analysis was to input random data that correlated to features selected while training the algorithm. A proof of concept was developed to mimic random user inputs and provide label predictions, the prediction were then compared with the original labels to validate if the predictions are accurate on this data. Table 4 shows the results from a series of 5 tests performed, the first 3 tests were random datasets selected from the original dataset and the last 2 tests were random approximated values.

Table 4: Prototype Test Results on Random Data Inputs

Test	Inputs	Actual Label	Predicted Label	Match
Test Run 1	X:9.375, Y:3.0625, Z:1.51	5	5	Yes
Test Run 2	X:6.995, Y:5.125, Z:0.3875	8	8	Yes
Test Run 3	X:0, Y:3.0625, Z:1.93	13	13	Yes
Test Run 4	X:9.4, Y:3, Z:1.8	6	6	Yes
Test Run 5	X:9.4, Y:3, Z:1.3	4	4	Yes

It can be seen that the ML model was able to predict with high accuracy for all the test runs, the data inputs had been varied through a large spectrum, but the random forest algorithm was able to learn the patterns and classify the labels successfully. The python script modelled to validate this proof of concept is presented below: -

```
from sklearn.externals import joblib
joblib.dump(rfc, 'Classifier.joblib')

# Load the model trained previously
rfc_1 = joblib.load('Classifier.joblib')

step_to_classify= [[9.375,3.0625,1.51],[6.995,5.125,0.3875],[0,3.0625,1.93],[9.4,3,1.8],[9.4,3,1.3]]

steps_to_classify = [step_to_classify]
predicted_step_values = rfc_1.predict(step_to_classify)
for predicted_value in predicted_step_values:
    print("These Co-Ordinates are in step:",predicted_value)
```

```
These Co-Ordinates are in step: 5
These Co-Ordinates are in step: 8
These Co-Ordinates are in step: 13
These Co-Ordinates are in step: 6
These Co-Ordinates are in step: 4
```

5.2 Degree of Meeting Requirements

Based on the performance of the algorithm and design analysis, table 5 summarizes the requirements conditions and the relevant comments.

Table 5: Degree of Meeting Requirements

REQ #	Requirement	Compliant	Comments
AI ECO-001	A dynamic approach for virtual instructions shall be explored.	Yes	With the AI ecosystem developed, users have the agility to complete any maintenance task of choice without having to begin from the first step of the overall process.
AI ECO-002	An AI ecosystem application shall be prototyped as a proof of concept.	Yes	A proof of concept prototype has been developed with the selected algorithm (Random Forest), that predicts the label of an object with minimum input parameters.
AI ECO-003	Accurate predictions based on random user input shall be provided.	Yes	A series of tests were performed on random data, the predicted labels were accurately matching the actual labels for all test runs.
AI ECO-004	Performance measures shall be analyzed.	Yes	The performance measures (Precision, Recall and F1-Score) have been analyzed for each class, and a confusion matrix was developed to further analyze the algorithm's performance on each class.

6. Conclusion

The intent of this study was to explore ways in which AI systems can be integrated into existing AR instruction modules that are used for maintenance and manufacturing procedures within the aerospace industry. These methods were developed for micro-level maintenance tasks that provides a dynamic approach for users during maintenance tasks, the dynamic method takes various human factors into consideration and utilize statistical ML algorithms that predict classes based on given random input features within the defined infrastructure. Three ML algorithms were analyzed namely, support vector machine, logistic regression and random forests; each algorithms' cost function and working principles were analyzed and custom hyperparameters were selected that provided optimum prediction results.

A detailed ML framework was developed with the combination of the selected algorithms, the training dataset created for the test subject was used as input for the ML algorithms. The input dataset was split into two with a split ratio of 70% and 30 % namely called training and testing set respectively, the training set was used to train the algorithms while the testing set was used to predict the classes for each algorithm. Various performance metrics were used to assess the performance of each algorithm, the major metrics considered were precision, recall and F1-score. The weighted average value of each score was taken as the final score for each algorithm, the F1-score was used as the main comparison parameter as it identifies a balance between the precision and recall values. The algorithm with the highest F1-score was selected to be the algorithm of choice for the prototype, random forests algorithm had the highest F1-score of 95%.

Based on the selected algorithm, a proof of concept prototype was developed that takes random input features and predicts the class it belongs to. A series of tests were performed with random data inputs and the predictions were compared to the actual classes of the input variables; the results were very favourable with no errors observed on any of the test runs. However, the random forests algorithms had an issue with predicting one class out of the 13 classes, the future work will consider further data analysis to understand the reasoning behind the error. Lastly, the design developed in this thesis provides the users with the agility to select any component within the overall infrastructure without having to go through the entire established process flow.

7. Future Work Considerations

This thesis was the starting point for the AI ecosystem intended to be integrated with current AR instruction modules, the overall mission objectives is to have a complete suite of AI algorithms that can elevate the overall performance of virtual instructions for maintenance and manufacturing purposes. There are three major areas of focus for future work, developing a data analysis framework, integrating object recognition algorithms using CNNs and reinforcement algorithms.

Developing a data analysis is critical for ML algorithms as mapping the performance of each class is challenging without having an established framework. As seen in this research study, the prediction for one of the classes was inaccurate and without a mapped backtracking feature, the anomalies are difficult to reason. The next aspect is the integration of object recognition algorithms. The current ML framework utilizes input features developed for an infrastructure. To further improve this process, algorithms using CNN have the potential to recognize objects directly, which can be used as inputs for the ML algorithms to perform prediction. Reinforcement learning is an exciting technique that almost simulates human interactions and continues to learn from them. With the addition of this algorithm, further optimizations can be performed, and maintenance tasks can be validated based on the performances of the users which potentially decreases human errors. With the addition of these AI systems, the ecosystem will have comprehensive tools to tackle a diverse range of dynamic maintenance/manufacturing tasks.

References

[1] R. Oliveira, T. Farinha, H. Raposo and N. Pires, "Augmented Reality and the Future of Maintenance," Centre for Mechanical Engineering of the University of Coimbra – CEMUC, Coimbra, Portugal.

[2] Z. h. a. L. Wenhua, "Architecture and Key Techniques of Augmented Reality Maintenance Guiding System for Civil Aircrafts," Journal of Physics: Conf. Series 787, Shanghai , China.

[3] M. B. K. a. E. B. M. B. Mohssen Mohammed, Machine Learning : Algorithms and Applications, Boca Raton: CRC Press LLC, 2016.

[4] M. A. F. a. E. C. C. d. Silva, "Augmented Reality in Aerospace Manufacturing: A Review," Journal of Industrial and Intelligent Information Vol. 4, No. 2, Brazil, 2016.

[5] F. M. J. M. J. R. J. Serván, "Using Augmented Reality in AIRBUS A400M Shop Floor Assembly Work Instructions," AIP, Madrid, 2012.

[6] S. B. G. J. H. F. T. A. M. R. R. E. W. P. D. a. S. T. Trevor Richardson, "Fusing Self-Reported and Sensor Data from Mixed-Reality Training," Interservice/Industry Training, Simulation, and Education Conference , St. Louis and Ames, 2014.

[7] A. P. a. A. S. Federico Manuri, "A State Validation System for Augmented Reality Based Maintenance Procedures," MDPI, Torino, 2019.

[8] A. Géron, Hands-On Machine Learning with Scikit-Learn and TensorFlow, Sebastopol: O'Reilly Media, Inc, 2017.

[9] K. K. a. M. W. I. Maglogiannis, Emerging Artificial Intelligence Applications in Computer Engineering, Amsterdam: IOS Press, 2007.

[10] R. Gandhi, "Towards Data Science," Medium, 30 June 2018. [Online]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47. [Accessed 30 March 2020].

[11] W. S. Noble, "What is a support vector machine?," Nature Publishing group, Berlin, 2006.

[12] S. Luo, "Towards Data Science," Medium, 15 October 2018. [Online]. Available: https://towardsdatascience.com/optimization-loss-function-under-the-hood-part-iii-5dff33fa015d. [Accessed 30 March 2020].

[13] L. H. Hamel, Knowledge Discovery with Support Vector Machines, Hoboken: John Wiley & Sons, Incorporated, 2009.

[14] S. Swaminathan, "Towards Data Science," Medium, 15 March 2018. [Online]. Available: https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc. [Accessed 30 March 2020].

[15] S. Luo, "Towards Data Science," Medium, 13 October 2018. [Online]. Available: https://towardsdatascience.com/optimization-loss-function-under-the-hood-part-ii-d20a239cde11. [Accessed 30 March 2020].

[16] H.-A. Park, " An Introduction to Logistic Regression: From Basic Concepts to Interpretation with Particular Attention to Nursing Domain," Journal of Korean Academy of Nursing, Seoul, 2013.

[17] G. Louppe, "Understanding Random Forests: From Theory to Practice," Cornell University, Ithaca, 2015.

[18] J. Rocca, "Towards Data Science," Medium, 22 April 2019. [Online]. Available: https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205. [Accessed 31 March 2020].

[19] T. Yiu, "Towards Data Science," Medium, 12 June 2019. [Online]. Available: https://towardsdatascience.com/understanding-random-forest-58381e0602d2. [Accessed 31 March 2020].

[20] A. Forbes, "Classification-algorithm evaluation: Five performance measures based onconfusion matrices," Journal of Clinical Monitoring, Palo Alto, 1995.

Appendix

```
import pandas as pd import
numpy as np import
matplotlib

import matplotlib.pyplot as plt from
matplotlib import cm import seaborn
as sns

import statsmodels.api as sm
%matplotlib inline

from sklearn.model_selection import train_test_split
```

```
df = pd.read_csv('Thesis_data.csv',header=0,encoding ='unicode_escape') df
```

	X	Y	Z	Step
0	9.375	3.0625	0.50	1
1	9.375	3.0625	0.51	1
2	9.375	3.0625	0.52	1
3	9.375	3.0625	0.53	1
4	9.375	3.0625	0.54	1
..	…	… …	…	
855	0.000	3.0625	1.89	13
856	0.000	3.0625	1.90	13
857	0.000	3.0625	1.91	13
858	0.000	3.0625	1.92	13
859	0.000	3.0625	1.93	13

[860 rows x 4 columns]

[5]:

```
df.describe()
```

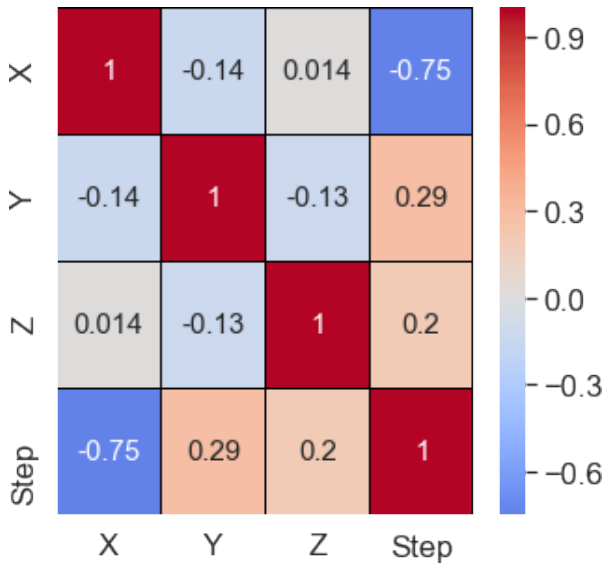
[5] :

		X	Y	Z	Step
	count	860.000000	860.000000	860.000000	860.000000
	mean	5.587116	4.845605	1.197465	7.756977
	std	3.719067	1.142329	0.522844	2.407837
	min	0.000000	3.062500	0.000000	1.000000
%	25□	1.562500	3.062500	0.783800	7.000000
%	50□	7.770000	5.125000	1.220000	8.000000
%	75□	8.575000	5.845000	1.616975	9.000000
	max	9.375000	5.845000	2.350000	13.000000

```
pearsoncorr = df.corr(method='pearson') pearsoncorr


#Correlation Matrix Heatmap
plt.figure(figsize=(5,5))
sns.set(font_scale=1.5)

sns.heatmap(pearsoncorr, xticklabels=pearsoncorr.columns,␣
 ↪yticklabels=pearsoncorr.columns, cmap='coolwarm', annot=True,
                    linewidth = 0.3, linecolor='black',robust = True,annot_kws={"size":
```



[6]:
```
X = df.drop(columns=['Step'])
y = df.Step


#split dataset into train and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=109)



from sklearn import metrics
from sklearn.metrics import classification_report, confusion_matrix, f1_score

from sklearn import svm
clf = svm.SVC(kernel='rbf',gamma = 'scale', C=10)
clf.fit(X_train, y_train)
```

	precision	recall	f1-score	support
1	1.00	1.00	1.00	5
2	0.50	1.00	0.67	5
3	0.00	0.00	0.00	7
4	0.60	0.60	0.60	5
5	0.60	1.00	0.75	3
6	1.00	1.00	1.00	4
7	1.00	1.00	1.00	38
8	1.00	1.00	1.00	79
9	1.00	1.00	1.00	79
10	0.64	1.00	0.78	9
11	0.00	0.00	0.00	10
12	0.58	1.00	0.74	7
13	1.00	1.00	1.00	7
accuracy			0.93	258
macro avg	0.69	0.82	0.73	258
weighted avg	0.89	0.93	0.90	258

[6]: 0.9022663036087016

[7]:
```
from sklearn.linear_model import LogisticRegression

LogisticReg = LogisticRegression(multi_class = 'multinomial',solver = 'sag',␣
↪C=10)

LogisticReg.fit(X_train, y_train) y_pred_logReg =
LogisticReg.predict(X_test)
```

```
print(classification_report(y_test, y_pred_logReg))

score2 = f1_score(y_test, y_pred_logReg,average='weighted') score2
```

	precision	recall	f1-score	support
1	0.71	1.00	0.83	5
2	0.38	0.60	0.46	5
3	1.00	0.14	0.25	7
4	0.80	0.80	0.80	5
5	0.75	1.00	0.86	3
6	1.00	1.00	1.00	4
7	1.00	1.00	1.00	38
8	1.00	1.00	1.00	79
9	1.00	1.00	1.00	79
10	0.75	1.00	0.86	9
11	1.00	0.50	0.67	10
12	0.75	0.86	0.80	7
13	0.88	1.00	0.93	7
accuracy			0.94	258
macro avg	0.85	0.84	0.80	258
weighted avg	0.96	0.94	0.94	258

[7]: 0.935310361473152

[8]:
```
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n_estimators = 100, max_depth=4, random_state =42) rfc.fit(X_train, y_train)

y_pred_rfc = rfc.predict(X_test)
print(classification_report(y_test, y_pred_rfc))

score3 = f1_score(y_test, y_pred_rfc,average='weighted') score3
```

	precision	recall	f1-score	support
1	0.83	1.00	0.91	5
2	0.50	0.80	0.62	5
3	0.00	0.00	0.00	7
4	0.62	1.00	0.77	5
5	1.00	1.00	1.00	3
6	1.00	1.00	1.00	4
7	1.00	1.00	1.00	38
8	1.00	1.00	1.00	79
9	1.00	1.00	1.00	79
10	1.00	0.78	0.88	9
11	0.83	1.00	0.91	10

12	1.00	1.00	1.00	7
13	1.00	1.00	1.00	7
accuracy			0.96	258
macro avg	0.83	0.89	0.85	258
weighted avg	0.95	0.96	0.95	258

[8]: 0.9512962812381417

[9]:
```
from sklearn.tree importDecisionTreeClassifier dtc =
DecisionTreeClassifier(max_depth=8) dtc.fit(X_train,
y_train)

y_pred_dtc = dtc.predict(X_test)
print(classification_report(y_test, y_pred_dtc))

score4 = f1_score(y_test, y_pred_dtc,average='weighted') score4
```

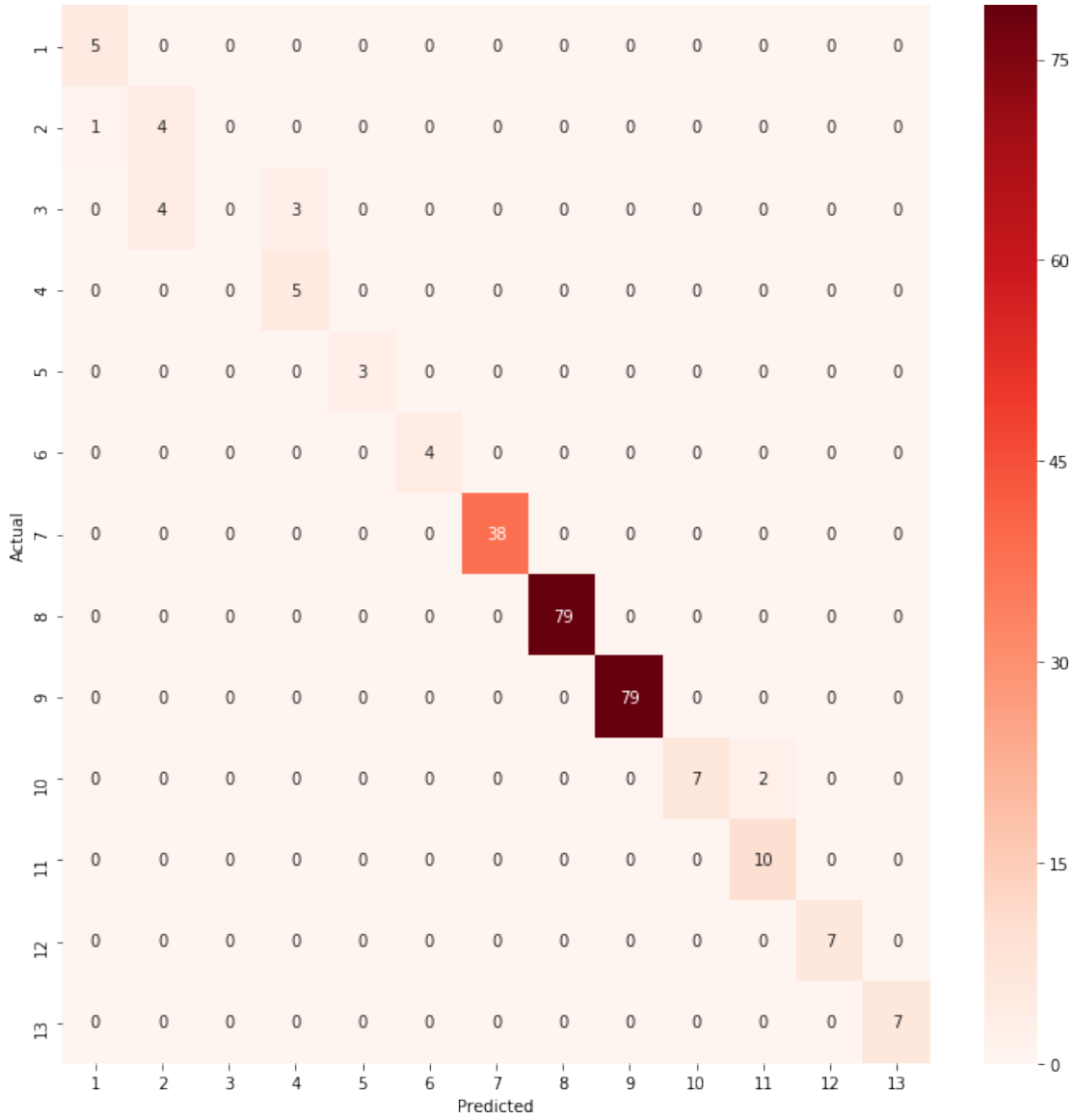
	precision	recall	f1-score	support
1	1.00	1.00	1.00	5
2	1.00	1.00	1.00	5
3	0.44	1.00	0.61	7
4	0.33	1.00	0.50	5
5	1.00	1.00	1.00	3
6	1.00	1.00	1.00	4
7	1.00	1.00	1.00	38
8	1.00	1.00	1.00	79
9	1.00	1.00	1.00	79
10	0.00	0.00	0.00	9
11	0.00	0.00	0.00	10
12	1.00	1.00	1.00	7
13	1.00	1.00	1.00	7
accuracy			0.93	258
macro avg	0.75	0.85	0.78	258
weighted avg	0.90	0.93	0.91	258

[9]: 0.9060498820357262

[10]:
```
conf_mat3 = confusion_matrix(y_test, y_pred_rfc)


fig, ax3 = plt.subplots(figsize=(12,12)) sns.heatmap(conf_mat3,
annot=True, fmt='d',
                    xticklabels=np.unique(df.Step), yticklabels=np.unique(df.Step),␣
↪cmap = 'Reds',linecolor='black')
```

```
plt.ylabel('Actual')
plt.xlabel('Predicted') plt.show()
```



[24]:
```
f1_score_array = [score1, score2, score3, score4] #score1-SVM,score2-Logistic␣
↪regression, score3-Random Forests, score4-Decision Tree
Max_f1_score= max(f1_score_array) Max_f1_score
```

[24]: 0.9512962812381417

```
from sklearn.externals import joblib joblib.dump(rfc,
'Classifier.joblib')


# Load the model trained previously

rfc_1 = joblib.load('Classifier.joblib')

step_to_classify= [[9.375,3.0625,1.51],[6.995,5.125,0.3875],[0,3.0625,1.93],[9.4,3,1.8],[9.4,3,1.3]]

steps_to_classify = [step_to_classify] predicted_step_values =
rfc_1.predict(step_to_classify) for predicted_value in
predicted_step_values:

    print("These Co-Ordinates are in step:",predicted_value)
```

```
These Co-Ordinates are in step: 5 These Co-
Ordinates are in step: 8 These Co-Ordinates are
in step: 13 These Co-Ordinates are in step: 6
These Co-Ordinates are in step: 4
```