

2019

Comparison Study of Supervised Machine Learning Algorithms for Real Time Prediction of UAV Behaviors

Brian Baity

Follow this and additional works at: <https://digital.library.ncat.edu/theses>



Part of the [Electrical and Computer Engineering Commons](#)

Comparison Study of Supervised Machine Learning Algorithms for Real Time Prediction of
UAV Behaviors

Brian Baity

North Carolina A&T State University

A thesis submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Department: Electrical and Computer Engineering

Major: Electrical Engineering

Major Professor: Dr. Abdollah Homaifar

Greensboro, North Carolina

2019

The Graduate College
North Carolina Agricultural and Technical State University
This is to certify that the Master's Thesis of

Brian Baity

has met the thesis requirements of
North Carolina Agricultural and Technical State University

Greensboro, North Carolina
2019

Approved by:

Dr. Abdollah Homaifar
Major Professor

Dr. Berat A. Erol
Committee Member

Dr. Daniel Limbrick
Committee Member

Dr. Ali Karimoddini
Committee Member

Dr. Abdullah Eroglu
Department Chair

Dr. Clay S. Gloster, Jr.
Interim Dean, The Graduate
College

© Copyright by

Brian Baity

2019

Biographical Sketch

Brian Baity was born in Atlanta, Georgia. After completing his secondary education at Southwest DeKalb High School in Decatur, Georgia, Brian enrolled at North Carolina Agricultural and Technical State University in Greensboro, North Carolina. He received a Bachelor of Science in Electrical Engineering in May of 2017. Brian then elected to continue his education and pursue a Master of Science in Electrical Engineering as well. In August of 2017, he entered the Electrical and Computer Engineering graduate program at North Carolina Agricultural and Technical State University.

Dedication

I would like to dedicate this thesis to the many individuals who have encouraged me throughout my education: my parents, who instilled in me the necessary lessons and tools to be successful in this life; my family, for supporting my dreams and every endeavor throughout my journey; and my love, who stood unwavering by my side during the late nights and early mornings.

I would also like to dedicate this thesis to my friends, who have demonstrated a relentless commitment to helping me complete this process. A special dedication is due to Phillip “Phil” Michael Thomas Jackson, whose wisdom, compassion, and spirit will never be forgotten. Long Live Phil! Lastly, I dedicate this thesis to my line brothers, when met with adversity we stood tall as one.

North Carolina Agricultural and Technical State University, Aggie Pride.

Acknowledgments

I would like to acknowledge the support from Air Force Research Laboratory and OSD for sponsoring this research under agreement number FA8750-15-2-0116 through the TECHLAV DOD Center of Excellence in Autonomy at North Carolina A&T State University.

Table of Contents

List of Figures	ix
List of Tables	x
ABSTRACT.....	1
CHAPTER 1 Introduction.....	2
1.1 Robots	2
1.1.1 Quadcopters	7
1.1.1.1 Quadcopter Societal Impact	7
1.1.1.2 Quadcopter Intelligence	8
1.2 Motivation.....	8
1.3 Problem Description	9
1.4 Thesis Organization	10
CHAPTER 2 Literature Review	12
2.1 Artificial Intelligence	12
2.1.1 Type 1	13
2.1.2 Type 2	14
2.1.3 Natural Processing Language (NLP)	15
2.1.4 Vision.....	16
2.1.5 Robotics	17
2.1.6 Autonomous Vehicles	17
2.2 Machine Learning	19
2.2.1 Machine Learning Types	20
2.2.1.1 Reinforcement Learning	20
2.2.1.2 Unsupervised Learning	21
2.2.1.3 Supervised Learning	22
2.3 Supervised Machine Learning Algorithms	24
2.3.1 k-Nearest Neighbors (kNN)	25
2.3.2 Naïve Bayes (NB)	26
2.3.3 Support Vector Machine (SVM).....	27
2.3.4 Decision Trees	28
2.3.5 Artificial Neural Networks (ANN)	30
2.4 Machine Learning Comparison and Evaluation.....	31
2.5 Machine Learning and Quadcopters	36
CHAPTER 3 Unamanned Aerial Vehicles (UAVs)	38
3.1 Introduction.....	38
3.2 Structures/Types	38

3.3 State of the Art	39
3.4 Mathematical Representation.....	39
CHAPTER 4 Case study, Results, Discussion.....	41
4.1 Problem Statement	41
4.2 Experimental Setup	41
4.3 Preliminary Results	44
4.4 System Representation and Modification	49
4.5 Results.....	51
4.6 Discussion/Limitations	58
CHAPTER 5 Conclusion	65
References.....	67
Appendix.....	71

List of Figures

Figure 1- 1 Classification of industrial robots by mechanical structure (Industrial robots,2016). ..	4
Figure 2-1 Artificial Intelligence Breakdown (Types of Artificial Intelligence, n.d.)	12
Figure 2-2 Different Paths of AI (Kumar GN, 2018).	15
Figure 2-3 Standard reinforcement learning model (Kaelbling, Littman, & Moore, 1996).	21
Figure 2-4 Supervised machine learning left (a) classification right (b) regression (Korbut, 2017).	22
Figure 2-5 Real-world application of supervised machine learning (Sathya & Abraham, 2013). ..	23
Figure 2-6 kNN Pseudocode (Meneses, Chavez, & Rodriguez, 2019).	25
Figure 2-7 Graphical Representation of kNN approach (Guo, Wang, Bell, & Bi, 2004).	26
Figure 2-8 Graphical Representation of Linearly (a) and Nonlinearly (b) Separable classes (Raschka, 2014).	28
Figure 2-9 A taxonomy of feed-forward and recurrent network architectures (Raza, 2016).	31
Figure 3-1 UAV classification (Norouzi Ghazbi, Aghli, Alimohammadi, & Akbari, 2016).	38
Figure 3-2 State space model of quadcopter (he & Zhao, 2014).	40
Figure 4-1 Scenario visual representation.....	42
Figure 4- 2 State Transition Flow Chart.	43
Figure 4- 3 Drones used for experiment.	43
Figure 4-4 Decision Tree performance.	45
Figure 4- 5 Naïve Bayes Performance.	46
Figure 4-6 k-Nearest Neighbors Performance.	47
Figure 4-7 Artificial Neural Network Performance.	48
Figure 4-8 Support Vector Machine Performance.	49
Figure 4-9 Adding noise to simulation test signal velocity in y directions.	50
Figure 4-10 Adding noise to simulation train signal altitude.	50
Figure 4-11 Decision Tree Performance 1-std.	51
Figure 4-12 Decision Tree Performance 2-std.	52
Figure 4-13 Decision Tree Performance 3-std.	52
Figure 4-14 Naïve Bayes Performances 1-std.	53
Figure 4-15 Naïve Bayes Performance 2-std.	53
Figure 4-16 Naïve Bayes Performance 3-std.	54
Figure 4-17 k-Nearest Neighbors Performance 1-std.	54
Figure 4-18 k-Nearest Neighbors Performance 2-std.	55
Figure 4-19 k-Nearest Neighbors Performance 3-std.	55
Figure 4-20 Artificial Neural Networks Performance 1-std.	56
Figure 4-21 Artificial Neural Networks Performance 2-std.	56
Figure 4-22 Artificial Neural Networks Performance 3-std.	57
Figure 4-23 Support Vector Machine Performance 1-std.	57
Figure 4-24 Support Vector Machine Performance 2-std.	58
Figure 4-25 Support Vector Machine Performance 3-std.	58
Figure 4-26 Highest Accuracy Performing Algorithm (Noiseless).	61
Figure 4-27 Highest Accuracy Performing Algorithm (90db).	62
Figure 4-28 Highest Accuracy Performing Algorithm (70db).	62
Figure 4-29 Highest Accuracy Performing Algorithm (50db).	63

List of Tables

Table 1-1 Classification of personal service robots by application areas and types of robots (Service Robots, 2016).....	5
Table 1-2 Classification of professional service robots by application areas and types of robots (Service Robots, 2016).....	6
Table 2-2 Comparison of kNN, SVM, ANN algorithms. (Prusty, Chakraborty, Jayanthi, & Velusamy, 2014).	33
Table 2-3 Model Performance Comparison. (Stevens & Diesing, 2014).	35
Table 4-1 Algorithm Accuracy with addition of SNR.....	60
Table 4-2 Percentage distribution table representing the accuracy of each classification method regarding the noise level.	64

ABSTRACT

An American scientific writer/critic of the mid-20th century, Joseph Wood Krutch, stated, “Technology made large populations possible: large populations now make technology indispensable.” Unmanned aerial vehicles (UAVs) have become a vital part of society, evolving from solely military use to commercial and even personal day-to-day use. UAV application has grown for mobility on demand, increasing the number of UAVs in the sky. Consequently, more UAVs increases the possibility of aerial collisions, challenging the safety of passengers flying and bystanders on the ground. Understanding the behavior of UAVs and unmanned aerial systems (UAS) in general, therefore, will decrease the possibility of aerial collisions. This thesis focuses on perception of UAVs; comparing how well machine-learning (ML) algorithms can analyze and predict their current states through supervised learning approaches. In this thesis, a data-driven software tool was developed to create a fundamental approach for UAV performance inspection. The developed algorithm was used for UAV behavioral analysis and the results provided better accuracy for predicting UAVs current state. This procedure includes using a multi-class classification for a single testing scenario, running the simulation environment with one Ar.Drones quadcopter for data gathering, hardware implementation for real-world implementation, and using Robot Operating System (ROS) as a middleware. All software development and implementation were conducted in Python programming language due to its high compatibility and robustness within a ROS development environment. The scenario stores the velocity readings on x, y, z directions, the altitude, and the corresponding state labels in matrix form. The procedure breaks the samples into training and testing for application of proposed supervised learning algorithms to predict output states of the system. Furthermore, these predictions are evaluated and analyzed, where results were compared within different ML approaches.

CHAPTER 1

Introduction

1.1 Robots

A robot, as Oxford's English Dictionary defines it, is a machine – especially one programmable by a computer – capable of carrying out a complex series of actions automatically. The word comes from the Czech *robota*, a word literally meaning forced labor, but which is also used figuratively to mean drudgery, hard work (Wilton, 2013). A term coined by Czech playwright, novelist and journalist named Karel Čapek, who introduced it in his 1920 hit play, *R.U.R.*, or *Rossum's Universal Robots*. (Science Friday, n.d.). There were many ideas and influence of robots such as Leonardo da Vinci sketched plans for a human look-alike robot that dates back as early as 1495. During the era between the 17 and 1900s, there were numerous life size automatons created. The nature of robots has evolved tremendously since then.

Robots started as simple task machines. Jacques de Vaucanson, a French inventor, developed three automata. (Cutting Tool Engineering, 2019). The first two were used for musical application, one specifically played flute and the other was capable of playing tambourine, drum, and flute. His last and most notable work was a duck capable of mimicking the life of an actual duck, including eating, wing flapping, and quacking. Another impactful advancement was George Devol's invention of the industrial manipulator Unimate which specific functions were transporting die-castings and welding them into automobiles (Cutting Tool Engineering, 2019). This implementation is considered one of the most important milestones in the history of robots, creating a process of taking and using industrial robots to replace unskilled workers. These examples presented along with other advancements during this era laid the foundations of Robotics and sparked heavy interest in the innovation of robots.

In general, robots are categorized by several classes, some by their application domains and others by their purposes such as locomotion and kinematics. The international federation of robotics (IFR), however, has two main classes, service and industrial robots (Robotics, 2016).

Industrial robots are automatically controlled, reprogrammable, multipurpose manipulator programmable in three or more axes, which can be either fixed in place or mobile for use in industrial automation applications (Industrial robots, 2016). These robots are classified by their mechanical structure and consist of:

- Linear robots (including Cartesian and gantry robots) – axes are corresponding to a Cartesian coordinate system and arm has three prismatic joints.
- SCARA robots – two parallel rotary joints to provide compliance in a plane.
- Articulated robots – arm has at least three rotary joints.
- Parallel robots (delta) – arms have rotary or concurrent prismatic joints.
- Cylindrical robots – axes form a cylindrical coordinate system.
- Not classified.

Figure 1.1 illustrates the configuration of a few of the above mechanical structures.


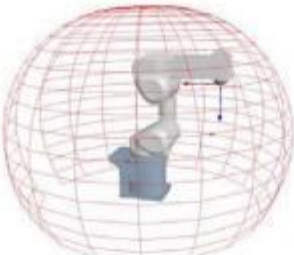

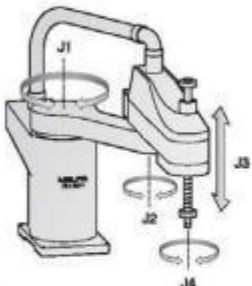
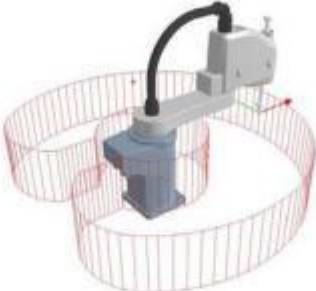

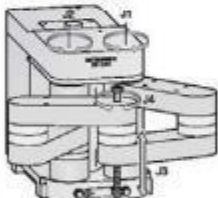
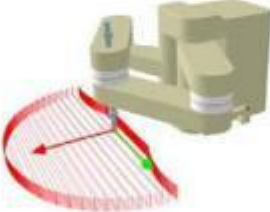







Principle	Kinematic Structure	Photo
Articulated Robot 		
SCARA Robot 		
SCARA Robot 		
Cartesian Robot 		
Parallel Robot 		

Figure 1-1 Classification of industrial robots by mechanical structure (Industrial robots,2016).

A service robot is a robot that performs useful tasks for humans or equipment excluding industrial automation application (Service Robots, 2016). This class breaks further into two subclasses based on the type of service it is performing.

- Personal service robots – applied for non-commercial task, usually by non-professional person. Represented in Table 1.1
- Professional service robots – applied for commercial task, operated by professionally trained personnel. Represented in Table 1.2.

Table 1-1 Classification of personal service robots by application areas and types of robots (Service Robots, 2016).

Section I	Types of robots: Service robots for personal/domestic use
1-6	Robots for domestic tasks
1	Robot companions/assistants/humanoids
2	Vacuuming, floor cleaning
3	Lawn-mowing
4	Pool cleaning
5	Window cleaning
6	Others
7-10	Entertainment robots
7	Toy/hobby robots
8	Multimedia/remote presence
9	Education and research
10	Others
11-13	Elderly and handicap assistance
11	Robotized wheelchairs
12	Personal aids and assistive devices
13	Other assistance functions
14	Personal transportation (AGV for persons)
15	Home security & surveillance
16	Other Personal / domestic robots

Table 1-2 Classification of professional service robots by application areas and types of robots (Service Robots, 2016).

Section II	Types of robots: Service robots for professional use
17-23	Field robotics
17	Agriculture
18	Milking robots
19	Other robots for livestock farming
20	Forestry and silviculture
21	Mining robots
22	Space robots
23	Other field robotics
24-28	Professional cleaning
24	Floor cleaning
25	Window and wall cleaning (incl. wall climbing robots)
26	Tank, tube and pipe cleaning
27	Hull cleaning (aircraft vehicles etc.)
28	Other cleaning tasks
29-31	Inspection and maintenance systems
29	Facilities, plants
30	Tank, tubes, pipes and sewers
31	Other inspection and maintenance systems
32-35	Construction and demolition
32	Nuclear demolition & dismantling
33	Building construction
34	Robots for heavy/civil construction
35	Other construction and demolition systems
36-39	Logistic systems
36	Automated guided (AGV) vehicles manufacturing environments
37	AGVs non-manufacturing environments (indoor)
38	Cargo handling, outdoor logistics
39	Other logistic systems
40-43	Medical robotics
40	Diagnostic systems
41	Robot assisted surgery or therapy
42	Rehabilitation systems
43	Other medical robots
44-46	Rescue & security applications
44	Fire and disaster fighting robots
45	Surveillance / security robots
46	Other rescue and security robots
47-50	Defense applications
47	Demining robots
48	Unmanned aerial vehicles
49	Unmanned ground based vehicles
50	Unmanned underwater vehicles
51	Other defense applications
52	Underwater systems (civil / general use)
53	Powered Human Exoskeletons
54	Unmanned aerial vehicles (general use)
55	Mobile Platforms in general use
56-60	Underwater systems (civil / general use)
56	Hotel & restaurant robots
57	Mobile guidance, information robots
58	Robots in marketing
59	Robot joy rides
60	Others (i.e. library robots)
61	Other professional service robots not specified above

These categories provided by the IFR can be broken into several other sub-categories. However, the focus of this work is Unmanned Aerial Vehicles (UAVs), mainly quadcopters.

1.1.1 Quadcopters

UAVs are a class of aerial vehicles that can take flight with the absent of an on-board human operator. Quadcopters classify as a type of UAV, they are agile vehicles that maneuver based on the rotational speed of four rotors. They were among the first vertical take-off and landing vehicles (VTOLs) (Quadcopter Arena, 2018). Being that helicopters use tail rotors, to counter the overall torque of the main rotor causing inefficiencies and limitations with flight (Quadcopter Arena, 2018), Quadcopters were designed in order to combat the problems helicopters faced when making vertical flights.

1.1.1.1 Quadcopter Societal Impact

The limitless potential applications of quadcopters have been reshaping many industries. The development of UAVs is primarily rooted in military research (Rao, Gopi, & Maione, 2016). Evolving from simply weaponized missions in hostile environments to applications of, but not limited to, data collection and surveying, surveillance, transportation, entertainment, emergency response, etc. While most intentions of innovating drones are designed for positive use, some of society feels reluctant to the push for drone practice. According to (Rao, Gopi, & Maione, 2016), the way UAV technology is currently used has an impact on society's conception of safety and security, privacy and ownership, individual and commercial liability, and the effectiveness and process of governmental regulation.

1.1.1.2 Quadcopter Intelligence

The main reason behind the numerous innovations on quadcopter technology and the new application domains to use them was the advancement of artificial intelligence. As known, these unmanned air systems work in a complex environment and these environments mostly apply to industrial domains, engineering problems, as well as the open source community, which generate vast scenarios, harsh environments, and incorporate different sensors, to evaluate how well the machines can adapt and perform. Due to the many sensors and cameras used in different scenarios, drones are creating large amounts of data, sometimes more than humans can handle. To date, almost every company that deals with data processing, analytics or ‘autonomous’ flight control and claims the use of artificial intelligence, machine or deep learning (Schroth, 2018).

1.2 Motivation

The Federal Aviation Administration (FAA) estimates the number of drones in the U.S. to reach seven million by 2020 (Hedlund, 2018). Although drones have had a significant positive impact on society, this increase in drone usage has unfortunately increased the risk of drone collision related events. There have been numerous reports of drone crashes and “close calls” within the past few years:

- August 2013 - drone crashed into grandstand during Virginia’s Great Bull Run
- December 2014 - drone crashed into customer at New York restaurant and in the same month drone nearly hits Airbus A320 during approach to Heathrow Airport
- January 2015 - drone crashed on White House lawn,
- September 2015 - American Airline pilots forced evasive action due to drone

- November 2015 - drone nearly collides with helicopter leaving St. Louis Children's Hospital

These are a few examples on an alarmingly increasing list of incidents.

Understanding that these incidents can be reduced with implementing rules and regulations, the Federal Aviation Administrations (FAA) released a list of regulations on June 21, 2016 (Dorr & Duquette, 2016). While the rules were put in place to control drone operation, many incidents have and will continue to occur. The FAA should apply a second level of screening for drone registration, where significant testing should be executed. These tests would incorporate different commands and trajectories for the UAV to perform based off its specifications and model dynamics. Evaluators will be able to recognize the structures and behaviors of standard flight where this type of second screening can help determine which drones should and should not be airborne.

1.3 Problem Description

UAV performance is a significant topic due to the many incidents that have occurred over the past years. The FAA has certified individuals to pilot UAVs and have included a mandatory drone registration. According to (FAA seal, n.d.), the pilot and drone registration process consists of:

1. Register your drone when flying under Part 107.
2. Label your drone (PDF) with your registration number. Registration costs \$5 per aircraft and is valid for 3 years.

In order to register, an individual would need to provide the following information:

- Email address

- Credit or debit card
- Physical address and mailing address (if different from physical address)
- Make and model of your unmanned aircraft
- Register an unmanned aircraft online.
- Register an unmanned aircraft by paper.

Despite these terms, the FAA still needs to consider incorporating different test and evaluation processes to ensure that drones are up to par with performance. For instance, a car registration is a personal proof that an individual rightly owns and pays taxes on that car. The purpose of registration is to declare a connection between a car and an owner. The owner must also have proof the particular car they drive passed the required standard inspection (which varies from state to state). The registration is important, but not the sole requirement for operations. The same should be taken into consideration when operating a UAV manually and unquestionably when an autonomous system is implemented with a UAV.

This thesis focuses on generating a test and evaluation process that can be fundamentally applied and developed for UAV inspection regarding the FAA for future screening, standards, and regulations. For achieving this, a data-driven approach is developed, where five supervised learning algorithms, are compared regarding how well they predict the user-defined states of a quadcopter for a single scenario in both simulation and real-world environments.

1.4 Thesis Organization

Chapter 2 presents an overview of Artificial Intelligence and its fundamental structures, applications, comparisons of those applications, and the metrics. Chapter 3 breaks down the classification of UAVs including the different design structures, the system dynamics and mathematical representation of a common quadcopter. Chapter 4 introduces the case study

regarding test and evaluation of a UAV's performance. This chapter includes the necessary results, modifications and limitations as well. Lastly, Chapter 5 ties the previous chapters together while reiterating the significance and contribution of this work.

CHAPTER 2

Literature Review

2.1 Artificial Intelligence

In the 21st century, artificial intelligence (AI) has become an important area of research in virtually all fields: engineering, science, education, medicine, business, accounting, finance, marketing, economics, stock market and law, among others (Oke, 2008). AI has applications for multiple aspects of human life – however, there continues to be ambiguity on both the definition of AI as well as on its constituent elements (Anand, 2018). Generally, the term “AI” is used when a machine simulates functions that human’s associate with other human minds such as learning and problem solving. (Gupta, 2017). Changes in the definition of artificial intelligence, however, are based upon the goals that are trying to be achieved with an AI system. (Marr, 2018). AI is difficult to define, but its definition has some consistencies. AI is not natural; as shown in human beings and some animal species, it is synthetic. Nevertheless, AI can make decisions and reason with respect to several factors, mimicking how the human brain would function. Although there are many structures and definitions associated with AI, they all classify into two specific types as shown in Figure 2.1.

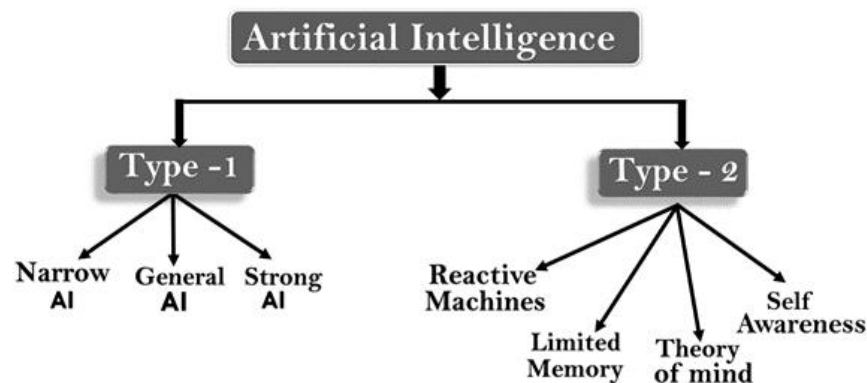


Figure 2-1 Artificial Intelligence Breakdown (Types of Artificial Intelligence, n.d.)

2.1.1 Type 1

Type one AIs consist of three levels: weak (narrow), strong (general), and super intelligence. Narrow AI is the most fundamental of type one. Narrow is used for performing a single task, whether that be playing chess, weather prediction, or ad suggestions. Even self-driving cars are a collection of narrow AI structures. Narrow AI is very limited to specific task it is programmed to complete. For example, one could not expect the chess playing AI to respond to a task of reporting the latest news. That is another AI's task to perform.

General AI is a machine that can think, reason, and perform tasks as a human would. They are programmed to handle situations in which they may be required to problem solve without having a person intervene (Frankenfield, 2019). This level of AI has not been reached, although many marketers will say otherwise to consumers. According to (Yao, 2017), developing a true artificial intelligence and establishing it as such is an ongoing challenge that continues to be hindered by difficulties devising definitions, metrics and tests. The notion behind the general AI is to create a system that could be smarter and think like a human on its own.

Lastly, Super AI is defined by the Oxford's AI expert as the level in which AI becomes much smarter than the best human brains in practically every field, including scientific creativity, general wisdom and social skills. Currently this level of AI is further ambiguous than that of General AI. Some scientists like Google's Demis Hassabis, optimistically believe that full AI development will help humans in areas such as space exploration, disease fighting, and environmental preservation. Other scientists, like Stephen Hawkins, believes the progression of AI will ultimately be the demise of humankind.

2.1.2 Type 2

Type 2's basis is functionality and contains four sub categories: (1) reactive machines, (2) limited memory, (3) theory of mind, and (4) self-awareness. Reactive machines are the basic types of AI. Reactive machines do not store memory or past information for current and future actions. An example of this AI is the IBM chess program that beat Garry Kasparov in the 1990s (Kumar GN, 2018). Their emphasis is on the present scenarios and carry out the best likely action.

Limited memory machines can store and use memory from previous experiences for a short time period. Some of the decision-making functions in self-driving cars have been designed this way (Kumar GN, 2018). These cars can keep the current speed of nearby cars, the distance of other cars, speed, and additional observations to navigate the road. Being that these systems' memory is limited, the observations are not permanently stored.

Theory of mind machines should be able to understand human emotions, different beliefs and views of people, and to have the capacity to interact socially. Researchers are making efforts to develop these types of machines. Even though many improvements exist, this kind of AI is not yet complete (Kumar GN, 2018).

Self-awareness machines are the final versions of AI. These machines will be super intelligent, possessing their individual consciousness and sentiments. Self-awareness machines will also be smarter than humans. They can be described as, "In simple words a complete human being" (Kumar GN, 2018). Although, self-awareness AI does not exist and is a hypothetical concept, once achieved it will be a momentous accomplishment in the AI field.

There are various paths towards building intelligent machines. Furthermore, the machines path depends on what task(s) or function(s) need(s) to be performed and what level of thought

capacity it takes to perform them. This thesis mentions five important paths of AI: (1) Natural Language Processing, (2) Vision, (3) Robotics, (4) Autonomous Vehicles, and (5) Machine Learning.

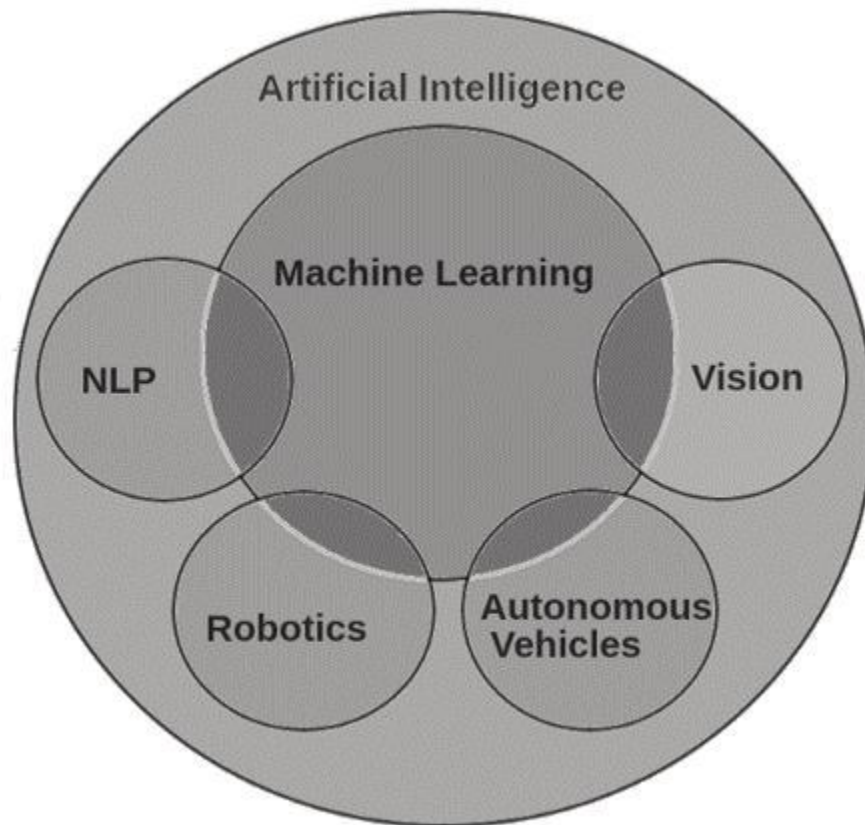


Figure 2-2 Different Paths of AI (Kumar GN, 2018).

2.1.3 Natural Processing Language (NLP)

Natural Language Processing (NLP) is an interdisciplinary field of computer science, artificial intelligence, and linguistics that explores how computers can be used to understand and manipulate natural language text or speech (Collobert & Weston, 2008). According to (Liu, Li, & Thomas, 2017), Natural language (NL) refers to any written or spoken human language that has naturally evolved for human communication. Two NL actions comprise human and computer interaction, generation and understanding. NL generation is the computer system

producing understandable human language texts and NL understanding is the computational process of transforming human language to a format the machine can understand. Different tasks of NLP include:

- Part-of-Speech Tagging (POS) – labeling each word by syntactic role indication.
- Chunking – also referred to as shadow parsing, labeling segments with syntactic role i.e. noun phrase (NP) or verb phrase (VP).
- Named Entity Recognition (NER) – labeling elements of a sentence into categories such as a place, location, animal, or company.
- Language Models – Uses statistical probability to estimate what the next word would be in a sequence.
- Semantic Role Labeling (SER) – assigns labels to phrases or words that specify their semantic role in a sentence.
- Semantic Related Words – predicting if words are related semantically i.e. holonyms, synonyms, and hypernyms.

2.1.4 Vision

Vision is a field comprised of machine vision and computer vision. This field allows machines to “see” as good as and better than humans do. Machine vision captures and analyzes visual information using a camera, analog-to-digital conversion, and digital signal processing (Kumar GN, 2018). As stated by (Crouch, 2019) computer vision and machine vision systems share most of the same components and requirements:

- An imaging device containing an image sensor and a lens.

- An image capture board or frame grabber may be used (in some digital cameras that use a modern interface, a frame grabber is not required).
- Application-appropriate lighting.
- Software that processes the images via a computer or an internal system, as in many “smart” cameras.

Computer vision is the automation of capturing and processing images, highlighting image analysis. Computer vision’s goal is to see, process, and provide meaningful results with respect to the observation. Machine vision is the use of computer vision in industrial environments, positioning it a subdivision of computer vision.

2.1.5 Robotics

The field of robotics is devoted to designing, manufacturing, and operation of robots. Robots are built to perform repetitive tasks that are either too dangerous or too difficult for humans to perform on a consistent basis at an efficient level. Examples include industrial assembly lines, massive production, nuclear power plants, military missions and law enforcement tasks, surgical operations in hospitals, service and hospitality tasks, and patrolling farm areas. There are even new directions to develop new humanoid service robots to assist police officers (Kumar GN, 2018).

2.1.6 Autonomous Vehicles

The development of autonomy and vehicles date back as far as the early 20th century, in Wisconsin, where a full-sized vehicle was controlled by radio waves. Further, interests sparked from the 2004 “DARPA Grand Challenge” creation; a contest that required a driverless car to complete a 150-mile course for a one-million-dollar prize. In its first year, no car was able to

complete the course; but in 2005, five vehicles were successful and finished it. Autonomous vehicles have exponentially increased their success since then, spreading to other platforms beyond road vehicles.

The U.S. DOT's National Highway Traffic Safety Administration policy on automated vehicles established five distinct levels of automation in vehicles (Levinson, Boies, Cao, & Fan, 2016). Defining levels of automation provides clarity for discussing automation among different states, product developers, and other stakeholders (Levinson, Boies, Cao, & Fan, 2016). These levels range from zero to four and represent the following:

- Level 0 stands for “No Automation” – The operator is in complete control of all primary vehicle controls at all times.
- Level 1 stands for “Function Specific Automation” – The vehicle is designated by the operator to control one or more specific functions and can seize control at any time.
- Level 2 stands for “Combined Function Automation” – The vehicle automates at least two primary-correlated functions of the vehicle, ultimately to relieve the operator from those specified primary functions by the operator. Although the two or more specified are automated, the operator must be cognizant to regain control from the vehicle.
- Level 3 stands for “Limited Self-Driving Automation” – All safety-critical functions are transferred from the operator to the vehicle. The vehicle is responsible for notifying the operator to intervene in situations where the operator's assistance is a necessity.
- Level 4 stands for “Full Self-Driving Automation” – The vehicle is to perform and control all safety critical functions while monitoring the surrounding environment for adaptation. The operator is only obligated to give a destination for the vehicle.

2.2 Machine Learning

Machine learning is an interdisciplinary field of research with influences and concepts from mathematics, computer science, artificial intelligence, statistics, biology, psychology, economics, control theory, and philosophy. In general terms, machine learning concerns computer programs that improve their performance at some task through experience (Lavesson, 2006). It is a method where the target (goal) is defined and the steps to reach that target is learned by the machine itself by training (e.g., gaining experience) (Kumar GN, 2018).

There are generally four questions used to address the design and implementation of learning programs.

- What is the input?
- What feedback is available?
- How should the solution be represented?
- What metrics are needed to evaluate performance?

The input is data or observations used for processing. The input has a set of attributes that describe the specifics of an instance/sample; it could be weight, height, speed, color, and so on. These instances can be represented as real (-3.14, 1.2, 0.5, 35.69) and integer numbers (-2, 4, 17, -9, -8, 5), or as a Boolean output (True, False). For instance, the attributes length, width, height, color, top speed, and horse power could be used to describe cars (Lavesson, 2006). The feedback depends on the style of learning applications used. These styles consist of Supervised, Unsupervised, and Reinforcement Learning, concepts discussed in detail in the next section.

The type of feedback implemented is also contingent upon the problem constraints and requirements. The solution representation of the learning type is the structure of the model

created once the machine has learned the pattern of the instances from the correlating attributes. The last constraint is the evaluation metrics used to determine how well the machine is learning i.e., accuracy, confusion matrix, and F1-score. Addressing these four concerns generates an increased level of understanding on how machines learn in different environments.

2.2.1 Machine Learning Types

As mentioned previously, the three types of learning are supervised, unsupervised, and reinforcement learning. They are distinguished by their different levels of access to the data provided and what type of scenarios are applicably just. This thesis focuses on supervised learning but does not exclude the descriptions of the other types.

2.2.1.1 Reinforcement Learning

Reinforcement learning is the problem faced by an agent that learns behavior through trial-and-error interactions with a dynamic environment (Kaelbling, Littman, & Moore, 1996). There are two main strategies of reinforcement learning:

- Search the behaviors space to find one that is performing well in the environment (i.e. Novel Search Genetic Programming, and Genetic Algorithms).
- Estimate the value of the state actions in the environment using statistical techniques and dynamic programming.

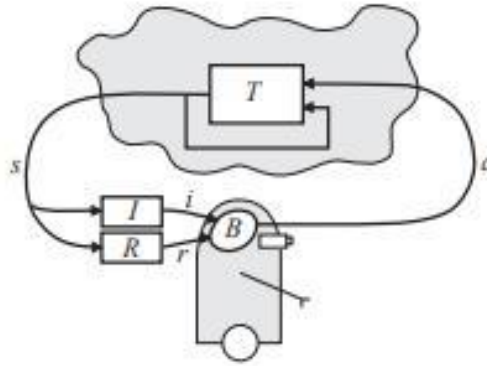


Figure 2-3 Standard reinforcement learning model (Kaelbling, Littman, & Moore, 1996).

Figure 2.3 shows the basic structure of reinforcement learning. In each step of the iteration, the machine/agent receives an input: i , and the current state of the environment: s . Then, the agent chooses an action: a , to produce an output. The action changes the state of the environment and this state transition is relayed to the agent through a scalar reinforcement signal, r . The agent's behavior: B , chooses actions to increase the long-run sum of values of reinforcement signal. The I in the figure represents an input function for the agent to determine how it views the environment state.

2.2.1.2 Unsupervised Learning

In unsupervised learning, only response variables are known (Amruthnath & Gupta, 2018). This type of learning searches for patterns where an input-output relationship is not given. It does not call for human annotation; it is fully automated (Beaula, Marikkannu, Sungheetha, & Sahana, 2016). A large subclass of unsupervised tasks is clustering, a method that observes and groups together the members that show similarity. These clusters are either user-defined or defined based on a constructed model with respect to each member's distance, density, or characteristics from other members. In unsupervised classification, previous information is not

necessary. The algorithm spots out the huddles in data and also analyst labels huddles (Beaula, Marikkannu, Sungheetha, & Sahana, 2016).

2.2.1.3 Supervised Learning

Supervised learning is based on training a data sample from a specific data source with the correct classification already assigned (Sathya & Abraham, 2013). Supervised learning tasks can be classified into two subgroups, regression and classification predictive modeling. Classification is the task of predicting a discrete class labels and regression is the task of predicting a continuous quantity shown in figures 2.4(a) and (b).

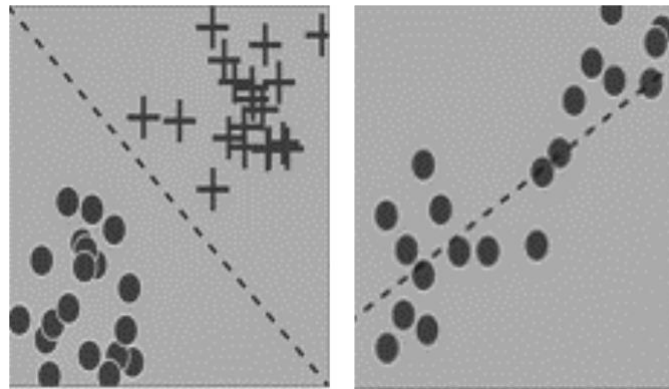


Figure 2-4 Supervised machine learning left (a) classification right (b) regression (Korbut, 2017).

These examples of correct input-output pairs can be shown to the machine during a training phase. The machine generates a learned model based off the input-output relationships provided during training. That model is then tested by trying to predict the output of a new set of examples, instances it has not previously seen. Lastly, the model is evaluated on how well it predicted the new instances. This process is formally shown in Figure 2.5.

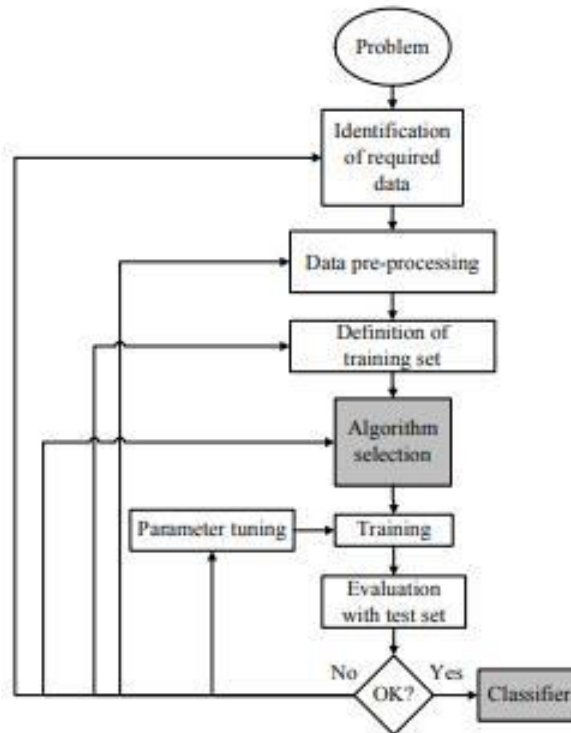


Figure 2-5 Real-world application of supervised machine learning (Sathya & Abraham, 2013).

1. Collect data. An available expert determines which attributes which attributes are of importance to the problem.
2. Data preprocessing. Consists of cleaning the dataset with respect to noise and missing values.
3. Definition of training set. There are many methods to split the data into training and testing samples. In this research, the data is randomly split into 70% training 30% testing.
4. Algorithm selection. This step is problem dependent; in this work, five algorithms are selected.
5. Training. The algorithms are applied to the training set learning and creating a model from the relationships between the input and outputs based on the attribute values.

6. Evaluation with test set. The model generated from training is applied to the test dataset and evaluated on user-defined metrics.
7. Based off how well the algorithm performs on the defined metrics of evaluation determines if the algorithm can be used as a classifier or needs its parameters tuned to increase performance (i.e. training speed, accuracy, and so on)

2.3 Supervised Machine Learning Algorithms

Trend forecasting and price prediction in stock trading, retail commerce, and sales are the most common areas that use supervised algorithms. In this work; however, supervised algorithms are used for predictive modeling regarding how to infer the behavior of a UAV. In this case, the algorithms use data to calculate possible outcomes. Below is a list of widely used supervised machine learning algorithms from the literature (Ayodele, 2010):

- Nearest Neighbor
- Gradient Boosted Trees
- Decision Trees
- Logistical Regression
- Naive Bayes
- Support Vector Machines (SVM)
- Random Forest
- Linear Regression
- Neural Networks

From these algorithms, five of them were chosen for model comparison and evaluation, k-Nearest Neighbors, Naïve Bayes, Support Vector Machines, Decision Trees, and Artificial Neural Networks.

2.3.1 k-Nearest Neighbors (kNN)

The k-Nearest-Neighbors (kNN) is one of the more simple and basic learning algorithms. The k-Nearest-Neighbours (kNN) is a non-parametric classification method that is simple but effective in many cases of classification (Guo, Wang, Bell, & Bi, 2004). This algorithm finds a group of “k” objects in the training set that is closest to the test instance. The instance learning process is based on solutions for similarly known problems, establishing the name “nearest neighbor learning”. The following pseudocode and parameters illustrate the implementation of this type of learning:

Algorithm 1: The k -nearest neighbors (KNN) algorithm.

Data: $D = \{(x_i, c_i), \text{ for } i = 1 \text{ to } m\}$, where $x_i = (v_1^i, v_2^i, \dots, v_n^i)$ is an observation that belongs to class c_i
Data: $x = (v_1, v_2, \dots, v_n)$ data to be classified
Result: class to which x belongs

$distances \leftarrow \emptyset;$
for y_i **in** D **do**
 $d_i \leftarrow d(y_i, x);$
 $distances \leftarrow distances \cup \{d_i\};$
end
Sort $distances = \{d_i, \text{ for } i = 1 \text{ to } m\}$ in ascending order;
Get the first K cases closer to x , D_x^K ;
 $class \leftarrow$ most frequent class in D_x^K

Figure 2-6 kNN Pseudocode (Meneses, Chavez, & Rodriguez, 2019).

Parameters:

1. Distance function, to determine closest neighbors between training instances and new instances (i.e. Euclidean, Manhattan, Minkowski).
2. The value k , which determines the number of neighbors considered when addressing the new instance.

3. A weighting function to evaluate the contribution of each neighbor of the new instance.
4. Evaluation method, to testing how well the found neighbors classify the new instance.

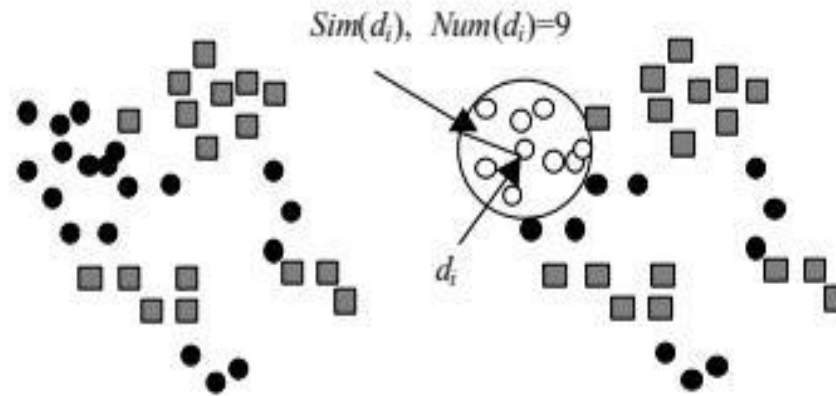


Figure 2-7 Graphical Representation of kNN approach (Guo, Wang, Bell, & Bi, 2004).

Figure 2.7 represents the kNN implementation by using Euclidean distance as the distance measure. An ideal local region can be represented by the central data point – d_i , the number of data instances inside the local region – $Num(d_i)$, and the similarity of the further point inside the local region – $Sim(d_i)$.

kNN is robust to noisy data and is effective when it is applied to large sets of training data. This is due to its lazy learning method; where there is no computation performed on the data before the new instance is given to the system.

2.3.2 Naïve Bayes (NB)

Naïve Bayes is a collection of learning algorithms that depend on probability theory and Bayes' Theorem for prediction. Naïve stems from the algorithm assuming that each feature is independent from the other features. Bayes refers to Thomas Bayes, a philosopher and statistician famously known for his Bayes Theorem that finds the probability of a hypothesis

given some prior knowledge (Hulden, 2014). Thus, the Naïve Bayes algorithm was created; an algorithm that is based on independence between input features and Bayes Theorem.

Bayes Theorem is given by the following formula (Sybba, et al., 2017):

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- $P(A|B)$ – the probability of event A occurring, given the event B has occurred (posterior probability).
- $P(B|A)$ – the probability of event B occurring, given event A has occurred (likelihood).
- $P(A)$ – the probability of event A (class prior probability).
- $P(B)$ – the probability of event B (predictor class probability).

Notice that both events A and B are independent, meaning the outcome probability of event A does not depend on event B's outcome. This concept of probability is used to classify new entries. NB can use several different model references such as normal, lognormal, gamma and Poisson density functions. Although NB may be simplistic, it can perform well with inputs of high dimensions.

2.3.3 Support Vector Machine (SVM)

Support Vector Machines performs well among some of the most recognized algorithms do to its consistent ability to be robust and accurate. SVM does not require an abundant amount of data for training and can handle many dimensions.

SVM works on the principle of margin calculation (Dey, 2016). It uses labeled training data to output an optimal hyperplane to categorize new inputs. The fundamental technique classifies the data by creating a function to split the data into relating labels on two conditions,

the data points with the largest likely margin and the least likely amount of inaccuracies. The margins are drawn in such a fashion that the distance between the margin and the classes is maximum hence, minimizing the classification error (Dey, 2016). Which is illustrated in figure 2.9:

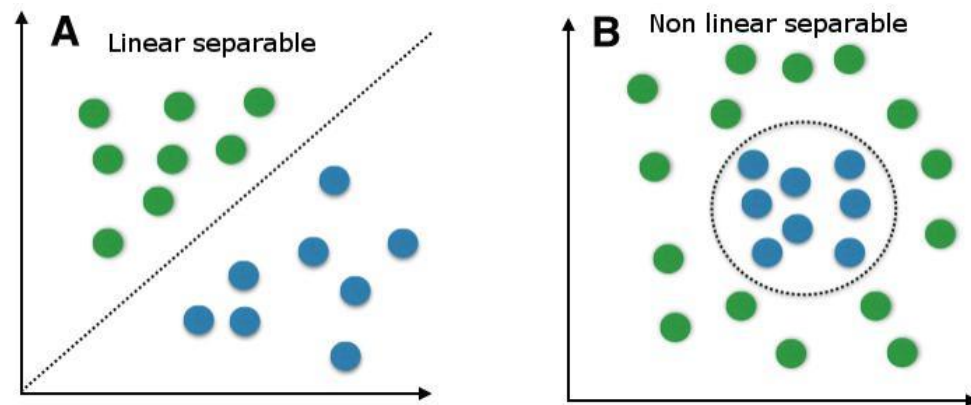


Figure 2-8 Graphical Representation of Linearly (a) and Nonlinearly (b) Separable classes (Raschka, 2014).

Support vector machines (SVM) were originally designed for binary classification (Chih-Wei & Lin, 2002). In this work, the SVM extends to handle multiclass classification. There are different ways to extend an SVM for multi class classification.

1. The SVM can be used as a fundamental classifier and can decompose the K-class problem into many binary class problems where K is the number of classes.
2. Directly considering all data in one optimization formulation (Chih-Wei & Lin, 2002).

2.3.4 Decision Trees

Decision tree is one of the most widely used techniques used in data mining (Sharma & Kumar, 2016). A tree model is represented with a set of if then rules for human interpretation. The name, tree model, comes from its tree like structure. It starts with an initial/root node that

then grows different branches and other nodes by making decisions until it reaches a leaf node. A leaf node corresponds to a tree starting from the roots and grows branches all the way to leaves.

There many different types of trees such as Iterative Dichotomiser 3 (ID3), C4.5, and Chi-square Automatic Interaction Detection (CHID). Although many variations of decisions trees were developed, the method of tree learning generally applies to problems with similar characteristics. The problems usually have a set of features/inputs (real or attribute values), discrete classes/outputs, errors in the training data or missing data, and disjunctive expressions. Despite similarities in composition, each tree grows in different ways. The following list provides descriptions on how some of the trees are grown (Al-Sagheer, Alharan, & Al-Haboobi, 2017):

- ID3 (Iterative Dichotomiser 3) is an easy way of decision tree algorithm. The evaluation used to build the tree is information gained for splitting criteria. The growth of tree stops when all samples have the same class or information gain is not greater than zero. It fails with numeric attributes or missing values.
- CHID (Chi-square–Automatic–Interaction–Detection): is an essential decision tree learning algorithm to only handle nominal attributes. It is a supplementation of the automatic interaction detector and theta automatic interaction detector procedures.
- C4.5 is the ID3 improvement or extension that presented by the same author. It is a mixture of C4.5, C4.5-no-pruning, and C4.5-rules. It uses gain ratio as splitting criteria. It is an optimal choice with numeric attributes or missing values. There are fundamental points that mark the two algorithms shown in the table below

These methods are some of the most popular algorithms, based upon inference, and successfully applies to a variety of tasks. Furthermore, decision trees' most important feature is

the capacity to break and represent very complex problems with a collection of simpler decisions.

Decision trees contribute a concrete technique for conceptual learning. They search the training set and grow top-down greedily choosing the next best feature for each new decision branch. The trees grow recursively from the root node down to the leaf node creating a fully interpretable tree.

2.3.5 Artificial Neural Networks (ANN)

The general function of a neural network is to produce an output pattern when given a particular input pattern and is loosely related to the way the brain operates (Dencelin & Ramkumar, 2016). ANNs have the ability of distributed information storage, parallel processing, reasoning, and self-organization (Kamruzzaman & Jehad Sarkar, 2011).

Based on the connection pattern (architecture), ANNs can be grouped into two categories (Jain & Mao, 1996):

- Feed-forward networks, in which graphs have no loops
- Recurrent (or feedback) networks, in which loops, and occur because of feedback connections.

Recurrent networks have the capacity to reprocess data in a feedback manner, acquiring information from earlier stages in the learning process, giving it the ability to adapt. On the other hand, feedforward networks do not feedback information from previous states, the data flows from the inputs through the hidden layer(s) to the output layer, as shown in figure 2.10. The output is obtained by applying all input values to a standard function for the network nodes (i.e. sigmoid).

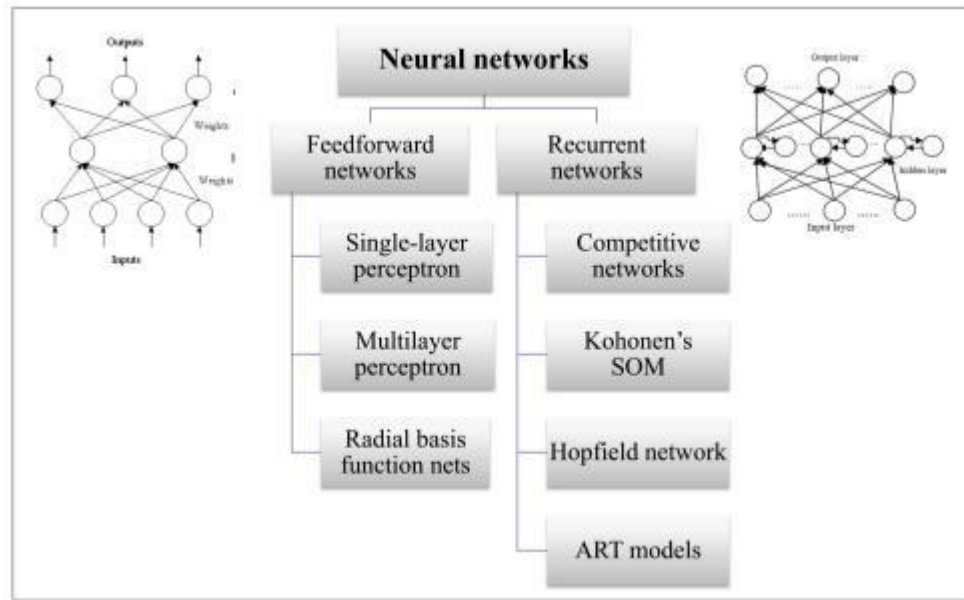


Figure 2-9 A taxonomy of feed-forward and recurrent network architectures (Raza, 2016).

In supervised learning, ANNs are trained using backpropagation. The theory of the BP algorithm is based on the error-correction learning rule which uses the error function in order to modify the connection weights to gradually reduce the error (Suliman & Zhang, 2015). The function based used for error correcting is defined as:

$$W = l * \varepsilon + m * W_p$$

Where W is the weight change, l is the learning rate, ε is the minimal error, m is the momentum, and W_p is the previous weight change.

2.4 Machine Learning Comparison and Evaluation

Supervised machine learning is the mission of conceiving a meaning from labelled training data that has a set of training examples (Praveena & Jaiganesh, 2017). The procedure consists of taking a dataset with known features/input and labels/outputs, generating a model based on the mapping from the inputs to the outputs, applying the generated model to unforeseen/test data and lastly evaluating the performance by different metrics. Many supervised

algorithms deal with classification, such as Artificial Neural Networks (ANNs), Support Vector Machines (SVMs), Decision Trees (DTs), Random Forest (RF), and Bayesian Networks. However, out of all the different algorithms applicable to classification, they all relate back to one question: Which one has the best performance compared to others?

To answer this question, one should accept that there is not an algorithm that better performs on every metric of evaluation. Learning algorithms are constantly tweaked and pinned against one another for performance evaluation. With that said, is it possible to objectively compare learning algorithms? This question cannot be generalized, as most algorithms' performance success is situational. It is important to evaluate learning algorithms on a variety of performance metrics because different learning algorithms are designed to optimize different criteria (e.g. SVMs and boosting optimize accuracy while neural nets typically optimize squared error). It is not uncommon for an algorithm to have optimal performance on one performance metric and be suboptimal on another (Caruana & Niculescu-Mizil, 2006). It is also important to take into account the parameters that would give the proposed algorithms an optimal performance when applied to the different problems. According to (Hossin & Sulaiman, 2015) accuracy or error rate is one of the common metrics in practice used by many researchers to evaluate the generalization ability of classifiers. Accuracy is a classifier's ability to correctly predict labels of unforeseen data based on a model generated from training data.

Although accuracy is an important metric, there have been studies of incorporating other metrics for performance evaluation. In (Prusty, Chakraborty, Jayanthi, & Velusamy, 2014) study, the authors state that k-nearest neighbors (kNNs), SVMs, and ANNs are the most widely used supervised multiclass algorithms. In their study, the authors perform transient classification

and compare the algorithms on the metrics of accuracy, training speed, computational cost and root mean square error, taking into consideration that accuracy is the major metric. The dataset consisted of five classes with two attributes/features. Training size was 746 and the testing size was 32. Table 2.1 is the comparison table of the algorithms' extensions and performance.

Table 2-1 Comparison of kNN, SVM, ANN algorithms. (Prusty, Chakraborty, Jayanthi, & Velusamy, 2014).

Algorithm	Best Prediction Accuracy (%)	Respective Parameter	Respective average prediction accuracy after 10-times 10-fold cross validation (%)
kNN	75.20	k = 13	75.96
SVM	93.75	rbf kernel	93.17
GDM-ANN	61.60	5 neurons	24.1
GDA-ANN	94.60	8 neurons	90.66
GDMA-ANN	99.10	8 neurons	89.5
RB-ANN	97.30	13 neurons	95.29
CGB-ANN	96.40	15 neurons	81.67
CGF-ANN	85.70	15 neurons	88.57
CGP-ANN	98.20	8 neurons	75.6
SCG-ANN	97.30	5 neurons	91.38
QN-ANN	94.60	15 neurons	90.62
LM-ANN	95.50	8 neurons	93.84
OSS-ANN	91.10	15 neurons	90.25
BR-ANN	99.10	5 neurons	95.68

The table shows that there are different extensions of ANN applied. There are six various categories of backpropagation (Prusty, Chakraborty, Jayanthi, & Velusamy, 2014).

Backpropagation is the optimization process to train a model in the least number of epochs, smallest possible error rate, and the fastest training time.

1. Additive Momentum:
 - a. Gradient Descent with momentum backpropagation (GDM).
2. Self-adaptive learning rate:
 - a. Gradient Descent with adaptive learning rate backpropagation (GDA).
 - b. Gradient Descent with momentum and adaptive learning rate (GDMA).
3. Resilient Backpropagation (RB):
4. Conjugate Gradient Backpropagation.
 - a. Scaled conjugate gradient back propagation (SCG).
 - b. Conjugate back propagation with Powell-Beale restarts (CGB).
5. Quasi-Newton:
 - a. Levenberg-Marquardt backpropagation (LM).
 - b. BFGS quasi-Newton backpropagation (QN).
6. Bayesian Regularization (BR).

Table 2.1 also shows that the Bayesian Regularization ANN produces the highest validation accuracy. GDM-ANN had a raw accuracy of 61.60 %, but when incorporating a k-fold validation metric the accuracy drops to 24.1% portraying a significant difference in accuracy. This outcome further proves that other metrics are necessary to effectively compare algorithms.

In a different study, six supervised algorithms' performances were compared on predicting the substrate type (substrate being the type of substance an organism lives on) from multibeam echo sounder data. The application techniques were SVMs, Classification Trees, ANNs, Naïve Bayes (NB), kNN, and Random Forest (RF). The metrics used for performance

evaluation were accuracy, balanced error rate (BER), and the Cohen's Kappa coefficient. BER is the average of the proportion of misclassifications in each class. Kappa measures the percentage of data values in the main diagonal and adjust the amount of agreement due to chance. According to (Stevens & Diesing, 2014) Kappa provides a more robust measure of agreement than accuracy. Table 2.2 shows the performance of each algorithm on the different metrics.

Table 2-2 Model Performance Comparison. (Stevens & Diesing, 2014).

Model	BER	Accuracy	Kappa
NB2	0.37	0.80	0.50
RF2	0.40	0.81	0.45
RF1	0.41	0.80	0.45
CT1	0.41	0.80	0.48
RF3	0.43	0.78	0.36
NB3	0.48	0.78	0.38
CT3	0.43	0.69	0.21
CT2	0.48	0.69	0.27
NN1	0.49	0.80	0.45
SVM1	0.53	0.78	0.39
1-NN	0.54	0.77	0.33
k-NN2	0.61	0.72	0.19
NB1	0.64	0.75	0.34
SVM2	0.67	0.78	0.27
NN3	0.69	0.78	0.21
k-NN3	0.69	0.78	0.22
SVM3	0.70	0.77	0.20
NN2	0.77	0.73	-0.07

As shown in the table 2-2, there are different versions of each algorithm. The number signifies the type of feature extraction was used in pre-processing; one denotes only primary features, two used the subset of features, and three used all features. Table 2-2 also shows RF2 has the highest accuracy, but it does not have the lowest error rate. This means that RF2 has the ability to correctly classify true positives, but it also had a significantly high number of misclassifications at 40%. On the other hand, NB2 appears to be the best performing classifier. Though NB2 did not have the top accuracy, it was second best. It had promising results in the other metrics having the lowest error rate and the highest kappa value signifying that it is the best for that problem.

These studies and more studies show that additional metrics are needed to perform a fair and unbiased comparison of supervised algorithms. In (Caruana & Niculescu-Mizil, 2006) study, the authors compare ten supervised learning algorithms on nine criteria of performance. Given this insight, for answering the previous general question presented we can conclude that there is not one algorithm that outperforms every evaluation metric. Fortunately, there are ways to measure an algorithm's performance with respect to how well it performs across multiple metrics, taking into consideration the problem's ranking importance of certain metrics.

2.5 Machine Learning and Quadcopters

Machine learning has played a pivotal role in flight navigation and control of UAVs such as using reinforcement learning for autonomous navigation, (Pham, La, Feil-Seifer, & Nguyen, 2018) and providing a framework for applying a RL algorithm to enable a UAV to operate in unknown environments. Another study by (Liakos, Busat, Moshou, Pearson, & Bochtis, 2018), used machine learning and UAVs to survey and monitor different agriculture parts, i.e., weed

detection, livestock production, crop quality, and water management. These and many other applications use machine learning to increase the functionality of the UAV, whether that be better object detection, increased efficiency of power consumption, path planning and route optimization, and so on. This research shifts the machine learning focus of UAV application to testing and evaluating the UAV's behavior, presenting a fundamental approach to standardizing how a UAV should behave in certain conditions.

CHAPTER 3

Unmanned Aerial Vehicles (UAVs)

3.1 Introduction

UAV innovation has become a very popular research area due to the improvement of different tools and applications involving their technology. In the previous chapters, I explained in-depth techniques used to build different levels of artificially intelligent machines. In the remaining chapters, I focus more on the applications of AI in UAVs. This includes modeling, simulation, hardware and software design, testing in simulation, and testing in a real-world environment. Specifically, I focus on using machine learning techniques to analyze and compare behaviors of UAVs in simulation and real-world environment.

3.2 Structures/Types

The types of UAV can be broken into categories distinguishable by their capabilities, model, and performance. Figure 3.1 shows the basic structures of UAVs.

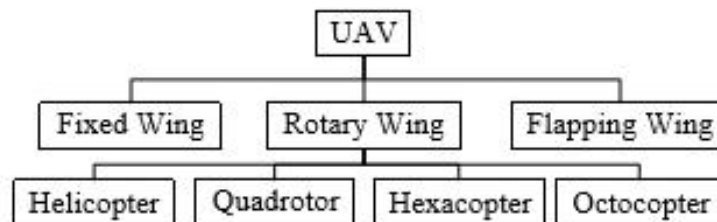


Figure 3-1 UAV classification (Norouzi Ghazbi, Aghli, Alimohammadi, & Akbari, 2016).

Fixed wings follow the same architecture as the standard commercial aircraft/airplane. Flapping wings mimics the motion of a bird or insect flapping to achieve flight. Rotary wings, most known for their ability to vertically take off and land, uses blades that revolve around a fixed mast for flight. These different types of wings coincide with how many rotors they have i.e. helicopter (1), hexacopter (6), and so on.

3.3 State of the Art

UAVs are increasingly adapted as remote sensing platforms (Aasen, 2017). Surveying and monitoring systems have become increasingly powerful due to specialized sensors. Collectively, this enables the aircraft to be significantly smaller than manned systems and thus capable of greater maneuverability (Jordan, et al., 2017). These improvements in UAV technology allows flights to be piloted in difficult access areas. The utility of UAVs is expanding dramatically, which increases the need of functional evaluation for drone inspection.

3.4 Mathematical Representation

There are many articles in the literature for mathematical and dynamic modeling of quadcopters. The primary focus of this thesis is on the state space model. To develop the dynamic mathematical model of a quadcopter, Newton-Euler and Euler-Lagrange equations are used and complex aerodynamic properties are considered. Being that quadcopter has only six degrees of freedom it is considered an under actuated robotic system. The mathematical model is linearized using these equations. The state space model adopted by the control system is $\dot{X} = (X, U)$, where X is the state vector and U is the control input vector. The state vector is chosen as $X = [x \ x' \ y \ y' \ z \ z' \ \theta \ \theta' \ \phi \ \phi' \ \psi \ \psi']$. In the design of controller, the state variables are chosen as $x_1 = x, x_2 = x', x_3 = y, x_4 = y', x_5 = z, x_6 = z', x_7 = \phi, x_8 = \phi', x_9 = \theta, x_{10} = \theta', x_{11} = \psi$, and $x_{12} = \psi'$ (he & Zhao, 2014).

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{y} \\ \ddot{y} \\ \dot{z} \\ \ddot{z} \\ \dot{\phi} \\ \ddot{\phi} \\ \dot{\theta} \\ \ddot{\theta} \\ \dot{\psi} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} x_2 \\ (\cos \psi \sin \theta + \sin \psi \sin \phi) \frac{U_1}{m} \\ x_4 \\ (\sin \psi \sin \theta \cos \phi - \cos \psi \sin \phi) \frac{U_1}{m} \\ x_6 \\ (\cos \theta \cos \phi) \frac{U_1}{m} - g \\ x_8 \\ \frac{1}{I_x} [(I_y - I_z) \dot{\psi} \dot{\theta} - J_r \dot{\theta} \Omega + U_2 l] \\ x_{10} \\ \frac{1}{I_y} [(I_z - I_x) \dot{\psi} \dot{\theta} + J_r \dot{\phi} \Omega + U_3 l] \\ x_{12} \\ \frac{1}{I_z} [(I_x - I_y) \dot{\theta} \dot{\phi} + U_4] \end{bmatrix}$$

Figure 3-2 State space model of quadcopter (he & Zhao, 2014).

In the above figure, m is the mass of the quadcopter, g is the gravitational acceleration, I_x , I_y , and I_z are inertia along each axis, Ω is the relative angular velocity of the rotor, J_r is the propeller inertia and l are the axis length of the quadcopter.

CHAPTER 4

Case Study, Results, and Discussion

4.1 Problem Statement

This thesis uses data driven supervised machine learning algorithms to predict and analyze the behaviors of a UAV given a predefined scenario. This scenario is carried out through simulation and real-world implementation in a controlled lab environment. The quadcopter generates data after the flight scenario is completed. Each proposed algorithm is applied to the data, creating a prediction model from the training portion of the data. The models produced are then evaluated and compared on the applicable performance metrics.

4.2 Experimental Setup

The experiment breaks into two parts simulation and real environment implementation, each of which follow the same scenario structure. That structure is a rectangular area-based structure in which the UAV searches through with the goal to observe the complete ground area below. The UAV begins from a home location, which has been predefined. Next, it vertically takes off, hovers shortly at a constant altitude, and then searches the area using a lawnmower pattern incorporating the waypoint navigation algorithm. Once it completes its search, it returns to the home location, hovers for a moment, and lands.

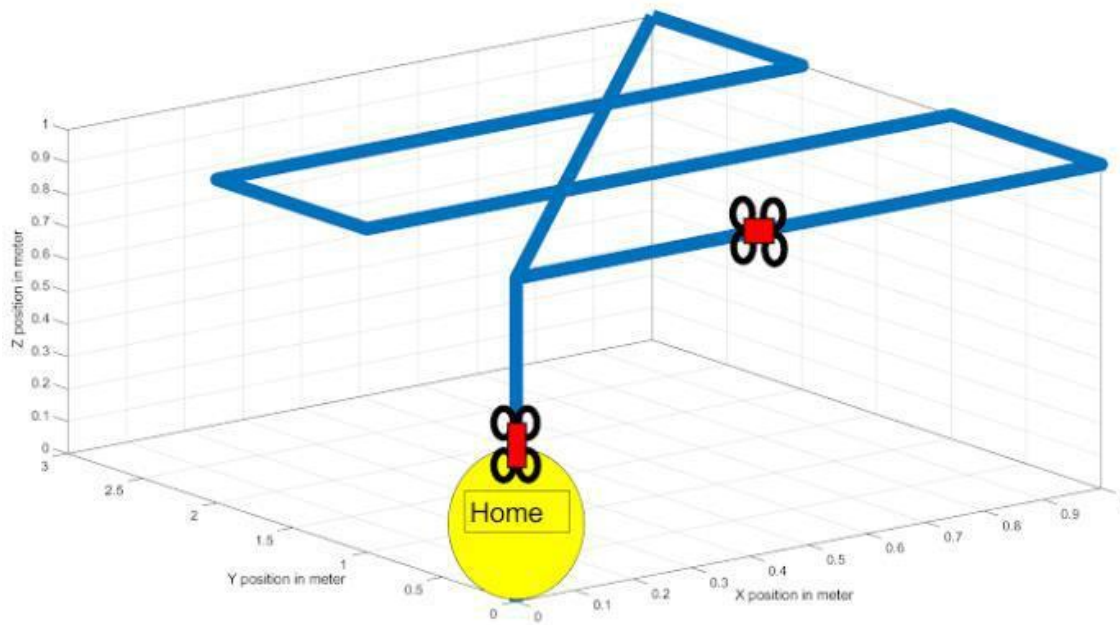


Figure 4-1 Scenario visual representation.

In this scenario, there are five high-level predefined states based four state variables.

1. Hold, $h = 0$, $V_x = 0$, $V_y = 0$, and $V_z = 0$.
2. Take-off, $h \neq 0$, $V_x = 0$, $V_y = 0$, and $V_z > 0$.
3. Hover, $h = \text{constant} \ \& \ h \neq 0$, $V_x = 0$, $V_y = 0$, and $V_z = 0$.
4. Search, $h = \text{constant} \ \& \ h \neq 0$, $V_x \neq 0$, $V_y \neq 0$, and $V_z = 0$.
5. Land, $h \neq 0$, $V_x = 0$, $V_y = 0$, and $V_z < 0$.

Where,

h = altitude, V_x = velocity in x direction, V_y = velocity in y direction, V_z = velocity in z direction. If we discretize the scenario, we observe that it is a combination of hold-takeoff-hover

search-hover-land-hold. Figure 4-2 illustrates the transition of the states traversed by the quadcopter throughout the scenario.

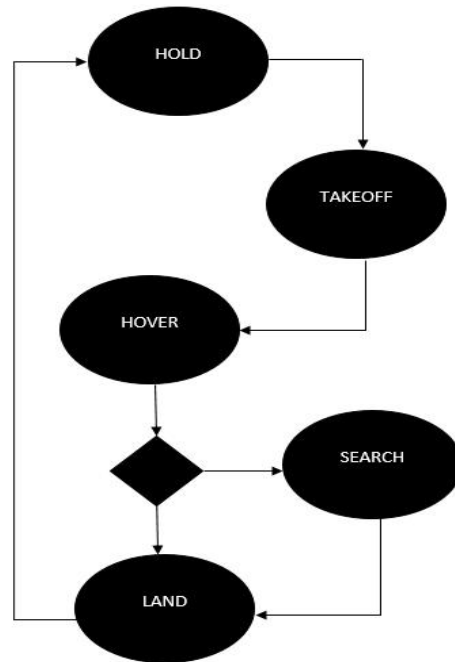


Figure 4-2 State Transition Flow Chart.

For the simulation portion, the scenario was generated in a ROS Gazebo environment. In both environments the sensor data was logged and save in a comma-separated value (.csv) formatted file based on the five user defined states x velocity, y velocity, z velocity, and altitude. All twelve state positions were saved position x, position y, position z, roll angle ϕ , pitch angle θ , yaw angle ψ , linear velocity in X direction \dot{x} , linear velocity in Y direction \dot{y} , linear velocity in Z direction \dot{z} , roll speed $\dot{\phi}$, pitch speed $\dot{\theta}$, and yaw speed $\dot{\psi}$. Figure 4-3 shows the drones used for simulation 3DR SOLO (left) and real implementation the Parrot AR.Drone 2.0.



Figure 4-3 Drones used for experiment.

4.3 Preliminary Results

The results are represented using three metrics:

- Accuracy – the number of correctly predicted labels divided by the number of labels in that state.
- Confusion Matrix – evaluation of the quality of the classifier outputs. The diagonal represents the correctly predicted labels regarding the true labels. The off diagonal represents the incorrectly predicted labels.
- Classification Report – it is a summary report of the precision, recall, f1-score, and support with respect to the true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). Also including how each class is weighted.
 - Precision - the ability of a classifier not to label an instance positive that is actually negative, $TP / (TP + FP)$.
 - Recall - the ability of a classifier to find all positive instances, $TP / (TP + FN)$.
 - F1-score - a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. $2 * (Recall * Precision) / (Recall + Precision)$.
 - Micro average – average with respect to the total true positives, false positives, and false negatives.
 - Macro average – unweighted mean per label.
 - Weighted average – support-weighted mean per label.

For the following figures, the left half of the figures represents the algorithms performance in the simulated environment and the right half represents the performance in real world environment.

Accuracy Noiseless is 0.9560300958446506

confusion Matrix Noiseless

```
[[9776  2  0  0 12]
 [ 2 1929 32  8  2]
 [  0  8 9588 557  6]
 [  0 22 723 7761  7]
 [ 15  4 17  9 1551]]
```

Report Noiseless

	precision	recall	f1-score	support
1	1.00	1.00	1.00	9790
2	0.98	0.98	0.98	1973
3	0.93	0.94	0.93	10159
4	0.93	0.91	0.92	8513
5	0.98	0.97	0.98	1596
micro avg	0.96	0.96	0.96	32031
macro avg	0.96	0.96	0.96	32031
weighted avg	0.96	0.96	0.96	32031

Accuracy Noiseless is 0.9087858970713676

confusion Matrix Noiseless

```
[[1377  0  0  0  6]
 [ 1 1276 23 12  3]
 [  0 14 6003 497  8]
 [  0 15 657 3570  4]
 [  3  2 22  6 569]]
```

Report Noiseless

	precision	recall	f1-score	support
1	1.00	1.00	1.00	1383
2	0.98	0.97	0.97	1315
3	0.90	0.92	0.91	6522
4	0.87	0.84	0.86	4246
5	0.96	0.95	0.95	602
micro avg	0.91	0.91	0.91	14068
macro avg	0.94	0.93	0.94	14068
weighted avg	0.91	0.91	0.91	14068

Figure 4-4 Decision Tree performance.

Accuracy Noiseless is 0.7092597795885236

confusion Matrix Noiseless

```
[[9531  62  86  24  87]
 [ 225 336 1225 109  78]
 [   0  28 9365 715  51]
 [   0  10 5774 2687  42]
 [ 278  41  497  65 715]]
```

Report Noiseless

	precision	recall	f1-score	support
1	0.95	0.97	0.96	9790
2	0.70	0.17	0.27	1973
3	0.55	0.92	0.69	10159
4	0.75	0.32	0.44	8513
5	0.73	0.45	0.56	1596
micro avg	0.71	0.71	0.71	32031
macro avg	0.74	0.57	0.59	32031
weighted avg	0.74	0.71	0.68	32031

Accuracy Noiseless is 0.6375959624680125

confusion Matrix Noiseless

```
[[1334  34   0   0  15]
 [  12 1053  178   1  71]
 [   0  42 6118  353   9]
 [   0  14 3763  464   5]
 [  99  339  110   1  53]]
```

Report Noiseless

	precision	recall	f1-score	support
1	0.92	0.96	0.94	1383
2	0.71	0.80	0.75	1315
3	0.60	0.94	0.73	6522
4	0.57	0.11	0.18	4246
5	0.35	0.09	0.14	602
micro avg	0.64	0.64	0.64	14068
macro avg	0.63	0.58	0.55	14068
weighted avg	0.62	0.64	0.56	14068

Figure 4-5 Naïve Bayes Performance.

Accuracy Noiseless is 0.9588804595548064

confusion Matrix Noiseless

```
[[9688   4    0    0   13]
 [   5 1928   21    7    1]
 [   0   27 9664  591   18]
 [   0   13   614 7885    5]
 [   9    5   13   15 1505]]
```

Report Noiseless

	precision	recall	f1-score	support
1	1.00	1.00	1.00	9705
2	0.98	0.98	0.98	1962
3	0.94	0.94	0.94	10300
4	0.93	0.93	0.93	8517
5	0.98	0.97	0.97	1547
micro avg	0.96	0.96	0.96	32031
macro avg	0.96	0.96	0.96	32031
weighted avg	0.96	0.96	0.96	32031

Accuracy Noiseless is 0.9116789877736707

confusion Matrix Noiseless

```
[[1363   0    0    0   20]
 [   2 1272   22   12    7]
 [   0   16 5938  557   11]
 [   0   12  535 3685   14]
 [   4    3   18    1  576]]
```

Report Noiseless

	precision	recall	f1-score	support
1	1.00	0.99	0.99	1383
2	0.98	0.97	0.97	1315
3	0.91	0.91	0.91	6522
4	0.87	0.87	0.87	4246
5	0.92	0.96	0.94	602
micro avg	0.91	0.91	0.91	14068
macro avg	0.93	0.94	0.94	14068
weighted avg	0.91	0.91	0.91	14068

Figure 4-6 *k*-Nearest Neighbors Performance.

Accuracy Noiseless is 0.8453810371202897

confusion Matrix Noiseless

```
[[9838  1  0  0  3]
 [ 33 1787 38 113  2]
 [  0  4 7863 2369 10]
 [  0  4 2238 6206  4]
 [ 139 320  43  95 921]]
```

Report Noiseless

	precision	recall	f1-score	support
1	0.98	1.00	0.99	9842
2	0.84	0.91	0.87	1973
3	0.77	0.77	0.77	10246
4	0.71	0.73	0.72	8452
5	0.98	0.61	0.75	1518
micro avg	0.83	0.83	0.83	32031
macro avg	0.86	0.80	0.82	32031
weighted avg	0.83	0.83	0.83	32031

Accuracy Noiseless is 0.7005046914984361

confusion Matrix Noiseless

```
[[1406  2  0  0  1]
 [ 22 1137 81 37  0]
 [  0  0 5764 660  3]
 [  0  1 2714 1609  0]
 [ 144 342  70  27 48]]
```

Report Noiseless

	precision	recall	f1-score	support
1	0.89	1.00	0.94	1409
2	0.77	0.89	0.82	1277
3	0.67	0.90	0.77	6427
4	0.69	0.37	0.48	4324
5	0.92	0.08	0.14	631
micro avg	0.71	0.71	0.71	14068
macro avg	0.79	0.65	0.63	14068
weighted avg	0.72	0.71	0.67	14068

Figure 4-7 Artificial Neural Network Performance.

Accuracy Noiseless is 0.7970840748025351

confusion Matrix Noiseless

```
[[ 9781    0    0    0    40]
 [  150 1404   305   52   29]
 [    0    6 10135   59    0]
 [    0   12 5221  3304    0]
 [  253  268  163    0  849]]
```

Report Noiseless

	precision	recall	f1-score	support
1	0.96	1.00	0.98	9821
2	0.83	0.72	0.77	1940
3	0.64	0.99	0.78	10200
4	0.97	0.39	0.55	8537
5	0.92	0.55	0.69	1533
micro avg	0.80	0.80	0.80	32031
macro avg	0.86	0.73	0.76	32031
weighted avg	0.85	0.80	0.78	32031

Accuracy Noiseless is 0.653483082172306

confusion Matrix Noiseless

```
[[1407    0    0    0    0]
 [  79 1046  180    0   44]
 [    0    0 6396    2    0]
 [    0    1 4008  308    1]
 [ 152  266  123    1  54]]
```

Report Noiseless

	precision	recall	f1-score	support
1	0.86	1.00	0.92	1407
2	0.80	0.78	0.79	1349
3	0.60	1.00	0.75	6398
4	0.99	0.07	0.13	4318
5	0.55	0.09	0.16	596
micro avg	0.65	0.65	0.65	14068
macro avg	0.76	0.59	0.55	14068
weighted avg	0.76	0.65	0.56	14068

Figure 4-8 Support Vector Machine Performance.

4.4 System Representation and Modification

In produced the aforementioned results, I attempted to see how far the algorithms could be tested with respect to adding white gaussian noise. The process for adding noise begins with separating the input data by the corresponding features, then finding the mean and

standard deviation of each input feature and randomly adding it to different instances in that specific feature. This process is done for 1, 2, and 3 standard deviations and is shown below.

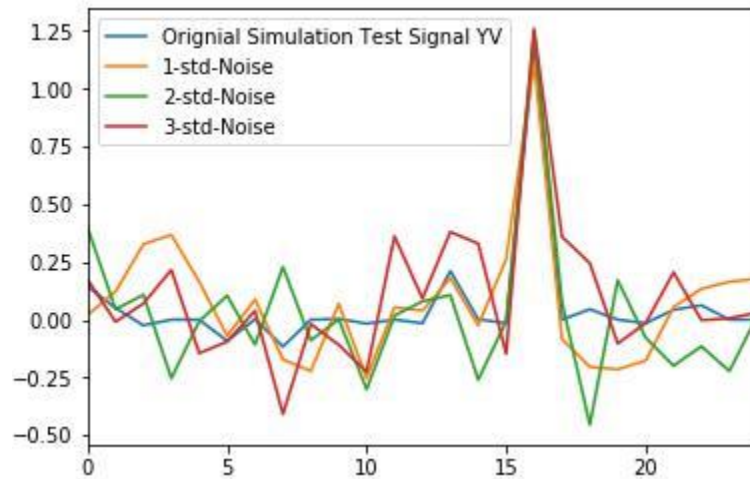


Figure 4-9 Adding noise to simulation test signal velocity in y directions.

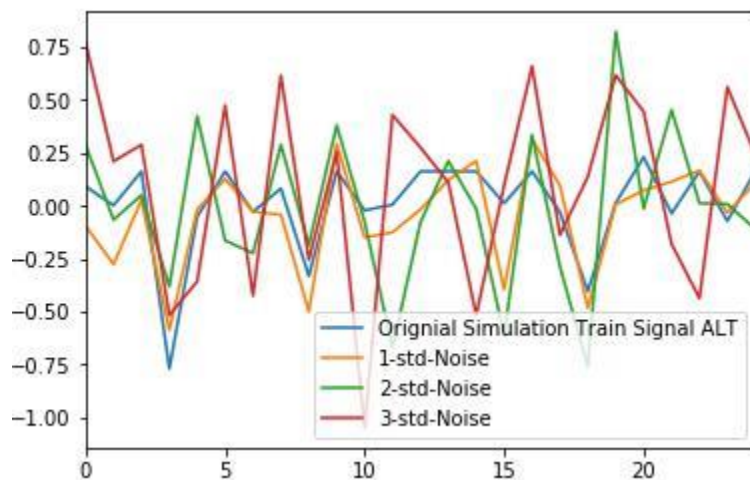


Figure 4-10 Adding noise to simulation train signal altitude.

The above graphs represent only two implementation of noise addition to give a visual representation of how adding one to three standard deviations of noise impacts the signal. This

method was done sixteen times, breaking down into four attributes for both real-world and simulation done for testing and training datasets.

4.5 Results

The results are organized in the same structure as the preliminary results; the simulation on the left and real world on the right. Each algorithm will have figures showing the progression from one to three standard deviations.

Accuracy 1std noise is 0.29604133495676066					Accuracy 1std noise is 0.24660932613022463				
Confusion Matrix 1std noise					Confusion Matrix 1std noise				
[[2684 1616 251 929 4310]					[[3 623 1 5 751]				
[166 569 52 744 442]					[1 809 43 208 254]				
[370 2100 528 5872 1289]					[0 2582 428 3185 327]				
[265 1488 479 5361 920]					[0 1734 245 2019 248]				
[161 285 120 305 725]]					[0 313 16 92 181]]				
Report 1std noise					Report 1std noise				
	precision	recall	f1-score	support		precision	recall	f1-score	support
1	0.74	0.27	0.40	9790	1	0.75	0.00	0.00	1383
2	0.09	0.29	0.14	1973	2	0.13	0.62	0.22	1315
3	0.37	0.05	0.09	10159	3	0.58	0.07	0.12	6522
4	0.41	0.63	0.49	8513	4	0.37	0.48	0.41	4246
5	0.09	0.45	0.16	1596	5	0.10	0.30	0.15	602
micro avg	0.31	0.31	0.31	32031	micro avg	0.24	0.24	0.24	14068
macro avg	0.34	0.34	0.26	32031	macro avg	0.39	0.29	0.18	14068
weighted avg	0.46	0.31	0.30	32031	weighted avg	0.47	0.24	0.21	14068

Figure 4-11 Decision Tree Performance 1-std.

Accuracy 2std noise is 0.2306359464268989					Accuracy 2std noise is 0.21610036963321008				
Confusion Matrix 2std noise					Confusion Matrix 2std noise				
[[1406 1624 398 2211 4151]					[[1 504 15 110 753]				
[161 361 65 803 583]					[0 581 14 340 380]				
[613 1297 630 5215 2404]					[5 2240 144 3306 827]				
[432 1022 660 4498 1901]					[2 1473 96 2105 570]				
[121 194 130 445 706]]					[0 223 6 160 213]]				
Report 2std noise					Report 2std noise				
	precision	recall	f1-score	support		precision	recall	f1-score	support
1	0.51	0.14	0.22	9790	1	0.12	0.00	0.00	1383
2	0.08	0.18	0.11	1973	2	0.12	0.44	0.18	1315
3	0.33	0.06	0.10	10159	3	0.52	0.02	0.04	6522
4	0.34	0.53	0.41	8513	4	0.35	0.50	0.41	4246
5	0.07	0.44	0.12	1596	5	0.08	0.35	0.13	602
micro avg	0.24	0.24	0.24	32031	micro avg	0.22	0.22	0.22	14068
macro avg	0.27	0.27	0.20	32031	macro avg	0.24	0.26	0.15	14068
weighted avg	0.36	0.24	0.23	32031	weighted avg	0.37	0.22	0.17	14068

Figure 4-12 Decision Tree Performance 2-std.

Accuracy 3std noise is 0.20696200555711655					Accuracy 3std noise is 0.2048123400625533				
Confusion Matrix 3std noise					Confusion Matrix 3std noise				
[[888 1473 559 2777 4093]					[[1 351 10 227 794]				
[129 301 89 852 602]					[1 381 19 430 484]				
[560 1133 731 4855 2880]					[2 1647 99 3313 1461]				
[491 915 763 4153 2191]					[2 1023 66 2173 982]				
[109 176 148 493 670]]					[3 155 7 187 250]]				
Report 3std noise					Report 3std noise				
	precision	recall	f1-score	support		precision	recall	f1-score	support
1	0.41	0.09	0.15	9790	1	0.11	0.00	0.00	1383
2	0.08	0.15	0.10	1973	2	0.11	0.29	0.16	1315
3	0.32	0.07	0.12	10159	3	0.49	0.02	0.03	6522
4	0.32	0.49	0.38	8513	4	0.34	0.51	0.41	4246
5	0.06	0.42	0.11	1596	5	0.06	0.42	0.11	602
micro avg	0.21	0.21	0.21	32031	micro avg	0.21	0.21	0.21	14068
macro avg	0.24	0.24	0.17	32031	macro avg	0.22	0.25	0.14	14068
weighted avg	0.32	0.21	0.20	32031	weighted avg	0.36	0.21	0.16	14068

Figure 4-13 Decision Tree Performance 3-std.

Accuracy 1std noise is 0.3176859916955449

Confusion Matrix 1std noise

```
[[ 898  583  838 3972 3499]
 [  44  444  287  877  321]
 [ 140 1059 2217 6166  577]
 [  99  652 1439 5896  427]
 [  40  129  149  506  772]]
```

Report 1std noise

	precision	recall	f1-score	support
1	0.74	0.09	0.16	9790
2	0.15	0.23	0.18	1973
3	0.45	0.22	0.29	10159
4	0.34	0.69	0.45	8513
5	0.14	0.48	0.21	1596
micro avg	0.32	0.32	0.32	32031
macro avg	0.36	0.34	0.26	32031
weighted avg	0.47	0.32	0.29	32031

Accuracy 1std noise is 0.14821580892806369

Confusion Matrix 1std noise

```
[[  0  657  0  5  721]
 [  0  618  3 121  573]
 [  0 2681 84 1350 2407]
 [  0 1651 38 1079 1478]
 [  0  265  1  36  300]]
```

Report 1std noise

	precision	recall	f1-score	support
1	0.00	0.00	0.00	1383
2	0.11	0.47	0.17	1315
3	0.67	0.01	0.03	6522
4	0.42	0.25	0.32	4246
5	0.05	0.50	0.10	602
micro avg	0.15	0.15	0.15	14068
macro avg	0.25	0.25	0.12	14068
weighted avg	0.45	0.15	0.13	14068

Figure 4-14 Naïve Bayes Performances 1-std.

Accuracy 2std noise is 0.2392994286784677

Confusion Matrix 2std noise

```
[[ 104  450  252 5757 3227]
 [  12  243  63 1117  538]
 [  57  855  489 6962 1796]
 [  51  584  310 6205 1363]
 [  8  71  42  761  714]]
```

Report 2std noise

	precision	recall	f1-score	support
1	0.45	0.01	0.02	9790
2	0.11	0.12	0.12	1973
3	0.42	0.05	0.09	10159
4	0.30	0.73	0.42	8513
5	0.09	0.45	0.15	1596
micro avg	0.24	0.24	0.24	32031
macro avg	0.27	0.27	0.16	32031
weighted avg	0.36	0.24	0.16	32031

Accuracy 2std noise is 0.12177992607335797

Confusion Matrix 2std noise

```
[[  0  542  0  85  756]
 [  0  492  0 146  677]
 [  0 2254  5 1113 3150]
 [  0 1392  2  883 1969]
 [  0  215  0  67  320]]
```

Report 2std noise

	precision	recall	f1-score	support
1	0.00	0.00	0.00	1383
2	0.10	0.37	0.16	1315
3	0.71	0.00	0.00	6522
4	0.38	0.21	0.27	4246
5	0.05	0.53	0.09	602
micro avg	0.12	0.12	0.12	14068
macro avg	0.25	0.22	0.10	14068
weighted avg	0.46	0.12	0.10	14068

Figure 4-15 Naïve Bayes Performance 2-std.

Accuracy 3std noise is 0.21738003808810208

Confusion Matrix 3std noise

```
[[ 31 317  80 6067 3295]
 [  4 119  16 1246  588]
 [ 24 541 112 7030 2452]
 [ 17 381 118 6111 1886]
 [  2  68  19  865  642]]
```

Report 3std noise

	precision	recall	f1-score	support
1	0.40	0.00	0.01	9790
2	0.08	0.06	0.07	1973
3	0.32	0.01	0.02	10159
4	0.29	0.72	0.41	8513
5	0.07	0.40	0.12	1596
micro avg	0.22	0.22	0.22	32031
macro avg	0.23	0.24	0.13	32031
weighted avg	0.31	0.22	0.13	32031

Accuracy 3std noise is 0.11366221211259597

Confusion Matrix 3std noise

```
[[  0 484  0 149 750]
 [  0 430  0 189 696]
 [  0 2065  0 1087 3370]
 [  0 1340  0 774 2132]
 [  0 186  0 71 345]]
```

Report 3std noise

	precision	recall	f1-score	support
1	0.00	0.00	0.00	1383
2	0.10	0.33	0.15	1315
3	0.00	0.00	0.00	6522
4	0.34	0.18	0.24	4246
5	0.05	0.57	0.09	602
micro avg	0.11	0.11	0.11	14068
macro avg	0.10	0.22	0.09	14068
weighted avg	0.11	0.11	0.09	14068

Figure 4-16 Naïve Bayes Performance 3-std.

Accuracy 1std noise is 0.3193000530735849

Confusion Matrix 1std noise

```
[[2042 1373 470 1330 4490]
 [ 172  657 101  716  316]
 [ 437 1796 739 6372  956]
 [ 278 1180 516 5958  585]
 [ 115  258  64  309  801]]
```

Report 1std noise

	precision	recall	f1-score	support
1	0.67	0.21	0.32	9705
2	0.12	0.33	0.18	1962
3	0.39	0.07	0.12	10300
4	0.41	0.70	0.51	8517
5	0.11	0.52	0.18	1547
micro avg	0.32	0.32	0.32	32031
macro avg	0.34	0.37	0.26	32031
weighted avg	0.45	0.32	0.29	32031

Accuracy 1std noise is 0.3346815467728177

Confusion Matrix 1std noise

```
[[ 485 529  2  5 362]
 [ 107 735 101 189 183]
 [  16 1893 1453 2716 444]
 [  7 1200  893 1841 305]
 [  66 244  48  90 154]]
```

Report 1std noise

	precision	recall	f1-score	support
1	0.71	0.35	0.47	1383
2	0.16	0.56	0.25	1315
3	0.58	0.22	0.32	6522
4	0.38	0.43	0.41	4246
5	0.11	0.26	0.15	602
micro avg	0.33	0.33	0.33	14068
macro avg	0.39	0.36	0.32	14068
weighted avg	0.47	0.33	0.35	14068

Figure 4-17 k-Nearest Neighbors Performance 1-std.

Accuracy 2std noise is 0.24375448783990508

Confusion Matrix 2std noise

```
[[ 764 1404 211 2978 4348]
 [ 140 476 43 883 420]
 [ 502 1429 219 6251 1899]
 [ 367 968 177 5495 1510]
 [ 75 202 26 540 704]]
```

Report 2std noise

	precision	recall	f1-score	support
1	0.41	0.08	0.13	9705
2	0.11	0.24	0.15	1962
3	0.32	0.02	0.04	10300
4	0.34	0.65	0.45	8517
5	0.08	0.46	0.14	1547
micro avg	0.24	0.24	0.24	32031
macro avg	0.25	0.29	0.18	32031
weighted avg	0.33	0.24	0.19	32031

Accuracy 2std noise is 0.27043645152118284

Confusion Matrix 2std noise

```
[[ 329 563 32 138 321]
 [ 149 537 91 332 206]
 [ 245 2003 707 2940 627]
 [ 135 1256 471 1965 419]
 [ 72 208 36 135 151]]
```

Report 2std noise

	precision	recall	f1-score	support
1	0.35	0.24	0.28	1383
2	0.12	0.41	0.18	1315
3	0.53	0.11	0.18	6522
4	0.36	0.46	0.40	4246
5	0.09	0.25	0.13	602
micro avg	0.26	0.26	0.26	14068
macro avg	0.29	0.29	0.24	14068
weighted avg	0.40	0.26	0.26	14068

Figure 4-18 k-Nearest Neighbors Performance 2-std.

Accuracy 3std noise is 0.2168961318722488

Confusion Matrix 3std noise

```
[[ 405 1399 87 3558 4256]
 [ 75 391 12 951 533]
 [ 352 1327 90 6001 2530]
 [ 272 958 69 5133 2085]
 [ 47 181 7 640 672]]
```

Report 3std noise

	precision	recall	f1-score	support
1	0.35	0.04	0.07	9705
2	0.09	0.20	0.13	1962
3	0.34	0.01	0.02	10300
4	0.32	0.60	0.41	8517
5	0.07	0.43	0.12	1547
micro avg	0.21	0.21	0.21	32031
macro avg	0.23	0.26	0.15	32031
weighted avg	0.31	0.21	0.15	32031

Accuracy 3std noise is 0.23851293716235428

Confusion Matrix 3std noise

```
[[ 240 504 41 248 350]
 [ 146 513 43 386 227]
 [ 350 1825 401 2980 966]
 [ 258 1169 261 1959 599]
 [ 75 164 25 181 157]]
```

Report 3std noise

	precision	recall	f1-score	support
1	0.22	0.17	0.20	1383
2	0.12	0.39	0.19	1315
3	0.52	0.06	0.11	6522
4	0.34	0.46	0.39	4246
5	0.07	0.26	0.11	602
micro avg	0.23	0.23	0.23	14068
macro avg	0.26	0.27	0.20	14068
weighted avg	0.38	0.23	0.21	14068

Figure 4-19 k-Nearest Neighbors Performance 3-std.

Accuracy 1std noise is 0.3449252286847117

Confusion Matrix 1std noise

```
[[3721 1119 236 1080 3686]
 [ 343 547 67 649 367]
 [ 633 1668 425 6041 1479]
 [ 440 1164 305 5510 1033]
 [ 269 197 41 293 718]]
```

Report 1std noise

	precision	recall	f1-score	support
1	0.69	0.38	0.49	9842
2	0.12	0.28	0.16	1973
3	0.40	0.04	0.08	10246
4	0.41	0.65	0.50	8452
5	0.10	0.47	0.16	1518
micro avg	0.34	0.34	0.34	32031
macro avg	0.34	0.36	0.28	32031
weighted avg	0.46	0.34	0.32	32031

Accuracy 1std noise is 0.2734646005117998

Confusion Matrix 1std noise

```
[[ 20 624 2 1 762]
 [ 9 686 34 200 348]
 [ 0 1700 487 3120 1120]
 [ 0 1216 318 2074 716]
 [ 3 264 19 90 255]]
```

Report 1std noise

	precision	recall	f1-score	support
1	0.62	0.01	0.03	1409
2	0.15	0.54	0.24	1277
3	0.57	0.08	0.13	6427
4	0.38	0.48	0.42	4324
5	0.08	0.40	0.13	631
micro avg	0.25	0.25	0.25	14068
macro avg	0.36	0.30	0.19	14068
weighted avg	0.45	0.25	0.22	14068

Figure 4-20 Artificial Neural Networks Performance 1-std.

Accuracy 2std noise is 0.2810839499235116

Confusion Matrix 2std noise

```
[[2668 950 172 2424 3628]
 [ 421 364 39 737 412]
 [1382 1230 213 5557 1864]
 [1039 959 164 4851 1439]
 [ 268 155 30 480 585]]
```

Report 2std noise

	precision	recall	f1-score	support
1	0.46	0.27	0.34	9842
2	0.10	0.18	0.13	1973
3	0.34	0.02	0.04	10246
4	0.35	0.57	0.43	8452
5	0.07	0.39	0.12	1518
micro avg	0.27	0.27	0.27	32031
macro avg	0.27	0.29	0.21	32031
weighted avg	0.35	0.27	0.25	32031

Accuracy 2std noise is 0.23531418822860398

Confusion Matrix 2std noise

```
[[ 59 431 12 117 790]
 [ 37 429 24 342 445]
 [ 50 1684 147 3255 1291]
 [ 52 1125 66 2226 855]
 [ 24 182 6 153 266]]
```

Report 2std noise

	precision	recall	f1-score	support
1	0.27	0.04	0.07	1409
2	0.11	0.34	0.17	1277
3	0.58	0.02	0.04	6427
4	0.37	0.51	0.43	4324
5	0.07	0.42	0.12	631
micro avg	0.22	0.22	0.22	14068
macro avg	0.28	0.27	0.17	14068
weighted avg	0.42	0.22	0.18	14068

Figure 4-21 Artificial Neural Networks Performance 2-std.

Accuracy 3std noise is 0.25573975211513844

Confusion Matrix 3std noise

```
[[2366 879 112 3133 3352]
 [ 417 254 34 819 449]
 [1626 960 149 5425 2086]
 [1279 782 113 4577 1701]
 [ 276 113 22 555 552]]
```

Report 3std noise

	precision	recall	f1-score	support
1	0.40	0.24	0.30	9842
2	0.09	0.13	0.10	1973
3	0.35	0.01	0.03	10246
4	0.32	0.54	0.40	8452
5	0.07	0.36	0.11	1518
micro avg	0.25	0.25	0.25	32031
macro avg	0.24	0.26	0.19	32031
weighted avg	0.32	0.25	0.22	32031

Accuracy 3std noise is 0.21908586863804377

Confusion Matrix 3std noise

```
[[ 112 326 9 253 709]
 [ 58 294 8 458 459]
 [ 206 1366 58 3283 1514]
 [ 142 945 38 2207 992]
 [ 28 150 3 210 240]]
```

Report 3std noise

	precision	recall	f1-score	support
1	0.21	0.08	0.11	1409
2	0.10	0.23	0.13	1277
3	0.50	0.01	0.02	6427
4	0.34	0.51	0.41	4324
5	0.06	0.38	0.11	631
micro avg	0.21	0.21	0.21	14068
macro avg	0.24	0.24	0.16	14068
weighted avg	0.37	0.21	0.16	14068

Figure 4-22 Artificial Neural Networks Performance 3-std.

Accuracy 1std noise is 0.3154163154444133

Confusion Matrix 1std noise

```
[[2391 1237 353 934 4906]
 [ 238 562 98 587 455]
 [ 538 1819 878 5683 1282]
 [ 333 1243 554 5310 1097]
 [ 181 232 37 252 831]]
```

Report 1std noise

	precision	recall	f1-score	support
1	0.65	0.24	0.35	9821
2	0.11	0.29	0.16	1940
3	0.46	0.09	0.14	10200
4	0.42	0.62	0.50	8537
5	0.10	0.54	0.16	1533
micro avg	0.31	0.31	0.31	32031
macro avg	0.35	0.36	0.26	32031
weighted avg	0.47	0.31	0.31	32031

Accuracy 1std noise is 0.37003838498720504

Confusion Matrix 1std noise

```
[[ 797 472 3 2 133]
 [ 181 828 216 60 64]
 [ 22 2528 2805 966 77]
 [ 10 1666 1739 791 112]
 [ 130 283 85 25 73]]
```

Report 1std noise

	precision	recall	f1-score	support
1	0.70	0.57	0.63	1407
2	0.14	0.61	0.23	1349
3	0.58	0.44	0.50	6398
4	0.43	0.18	0.26	4318
5	0.16	0.12	0.14	596
micro avg	0.38	0.38	0.38	14068
macro avg	0.40	0.38	0.35	14068
weighted avg	0.49	0.38	0.40	14068

Figure 4-23 Support Vector Machine Performance 1-std.

Accuracy 2std noise is 0.20895070400549468					Accuracy 2std noise is 0.2779286323571225				
Confusion Matrix 2std noise					Confusion Matrix 2std noise				
[[673 1004 166 2315 5663]					[[720 364 68 72 183]				
[71 299 18 609 943]					[366 454 127 257 145]				
[417 1253 179 4569 3782]					[491 1774 974 2450 709]				
[268 896 131 4041 3201]					[354 1269 572 1622 501]				
[72 160 19 375 907]]					[192 165 47 102 90]]				
Report 2std noise					Report 2std noise				
	precision	recall	f1-score	support		precision	recall	f1-score	support
1	0.45	0.07	0.12	9821	1	0.34	0.51	0.41	1407
2	0.08	0.15	0.11	1940	2	0.11	0.34	0.17	1349
3	0.35	0.02	0.03	10200	3	0.54	0.15	0.24	6398
4	0.34	0.47	0.40	8537	4	0.36	0.38	0.37	4318
5	0.06	0.59	0.11	1533	5	0.06	0.15	0.08	596
micro avg	0.19	0.19	0.19	32031	micro avg	0.27	0.27	0.27	14068
macro avg	0.26	0.26	0.15	32031	macro avg	0.28	0.31	0.25	14068
weighted avg	0.35	0.19	0.16	32031	weighted avg	0.41	0.27	0.28	14068

Figure 4-24 Support Vector Machine Performance 2-std.

Accuracy 3std noise is 0.15530891948424963					Accuracy 3std noise is 0.2277011657662781				
Confusion Matrix 3std noise					Confusion Matrix 3std noise				
[[204 670 38 2044 6865]					[[658 239 56 186 268]				
[39 168 8 450 1275]					[442 282 66 340 219]				
[158 703 46 2945 6348]					[1133 1131 436 2402 1296]				
[126 563 36 2471 5341]					[717 738 301 1629 933]				
[22 79 4 369 1059]]					[186 110 28 135 137]]				
Report 3std noise					Report 3std noise				
	precision	recall	f1-score	support		precision	recall	f1-score	support
1	0.37	0.02	0.04	9821	1	0.21	0.47	0.29	1407
2	0.08	0.09	0.08	1940	2	0.11	0.21	0.15	1349
3	0.35	0.00	0.01	10200	3	0.49	0.07	0.12	6398
4	0.30	0.29	0.29	8537	4	0.35	0.38	0.36	4318
5	0.05	0.69	0.09	1533	5	0.05	0.23	0.08	596
micro avg	0.12	0.12	0.12	32031	micro avg	0.22	0.22	0.22	14068
macro avg	0.23	0.22	0.10	32031	macro avg	0.24	0.27	0.20	14068
weighted avg	0.31	0.12	0.10	32031	weighted avg	0.36	0.22	0.21	14068

Figure 4-25 Support Vector Machine Performance 3-std.

4.6 Discussion/Limitations

This research is done on two levels of comparison. The first comparison being the algorithms with respect to how well they perform against themselves in simulation versus real world performance evaluation (for example; How well does the ANN perform in the real world

versus how ANN performs in simulation?) and the second being how well the algorithms perform against each other in different environments (i.e. how well an algorithm performs in a simulation environment based off F1-score).

These preliminary results show that each individual algorithm performs better on the simulated data versus the real-world experiment. Which stems from the fact that simulation is ideal in the manner of operating perfectly as opposed to real world implementation, where different constraints can cause the scenario not to be executed flawlessly. These results also provide that the k-Nearest Neighbors outperforms all the algorithms in both simulation and real-world environments. Further proving that even though it is simplistic in construct, it can perform as well or better than its counterparts.

Many inferences can be made from the inclusion of noise, trivially the addition of noise significantly decreases the performance of each algorithm in both simulation and real-world implementation. ANN had the highest accuracy in simulation across all three stds of noise addition. However, SVM was the least affected in both experiments by the initial addition of noise. SVM also had the highest accuracy in real world implementation with 1 and 2-std of noise and kNN produce the highest accuracy in real world implementation with 3-std of noise. The kNN algorithm performance on the simulations experiment was outperformed across all three stds by the real-world implementation. Lastly, the performance level drop was consistently greater in the simulation experiment demonstrating that though both experiments are very sensitive to noise addition with respect standard deviation, the real-world implementation is less sensitive.

The incorporation of noise provides additional insight to better improve algorithms, making them more robust. However, the level in which noise is added can be improved, seeing as though 1-std was significantly harsh on the input signal. Re-evaluating the algorithms performance led to

introducing a second method of noise addition, signal to noise ratio (SNR). The SNR is defined as signal strength divided by noise strength and is measured in decibels (db). A noiseless signal has a decibel value closer infinity. In this study I calculated the SNR for 1std of the signals in the system and all of which were less than 4db. Hence, my signals were completely distorted causing the algorithms to perform poorly.

Table 4-1 Algorithm Accuracy with addition of SNR.

Name	Algorithm Accuracy				
	Noiseless	1 STD Noise	SNR 90db	SNR 70db	SNR 50db
Simulation DT	95.60%	29.60%	88.49%	85.12%	79.04%
Experimental DT	90.87%	24.66%	<u>75.85%</u>	68.54%	56.37%
Simulation NB	70.92%	31.76%	<u>70.78%</u>	70.74%	70.87%
Experimental NB	63.75%	14.82%	63.31%	63.63%	54.68%
Simulation kNN	95.88%	31.93%	87.88%	85.50%	80.19%
Experimental kNN	<u>91.16%</u>	33.46%	73.97%	<u>71.36%</u>	61.79%
Simulation ANN	84.53%	34.49%	83.99%	83.91%	82.91%
Experimental ANN	70.50%	27.34%	70.06%	69.27%	<u>67.91%</u>
Simulation SVM	79.70%	31.54%	79.68%	79.65%	79.51%
Experimental SVM	65.34%	<u>37.00%</u>	65.40%	65.30%	65.28

Table 4.1 illustrates the algorithms performance with the addition of noise by means of SNR. The bold values signify the algorithms that had the highest accuracy in the simulation environment for each test and the underline is the highest accuracy in the real-world environment test. The kNN algorithm performance accuracy remains the highest in the simulation environment for the noise addition of 90db and 70db. However, it was the most impacted by the addition of noise, followed by DT. Although SVM was second to last in accuracy, was least impacted by noise. ANN performed decent in the accuracy metric as well as demonstrated high robustness. The use of SNR in this problem domain better validates how well the algorithms perform at different noise

levels rather than using standard deviation. The following figures, Figure 4-26 to Figure 4-29, depict a graphical representation of the algorithms that achieved the highest accuracy with respect to each level of noise, plotting the predicted values and true values of each state. Following the figures, Table 4-2 illustrates the percentage distribution of how accurate each method classified each state with respect to the noise level.

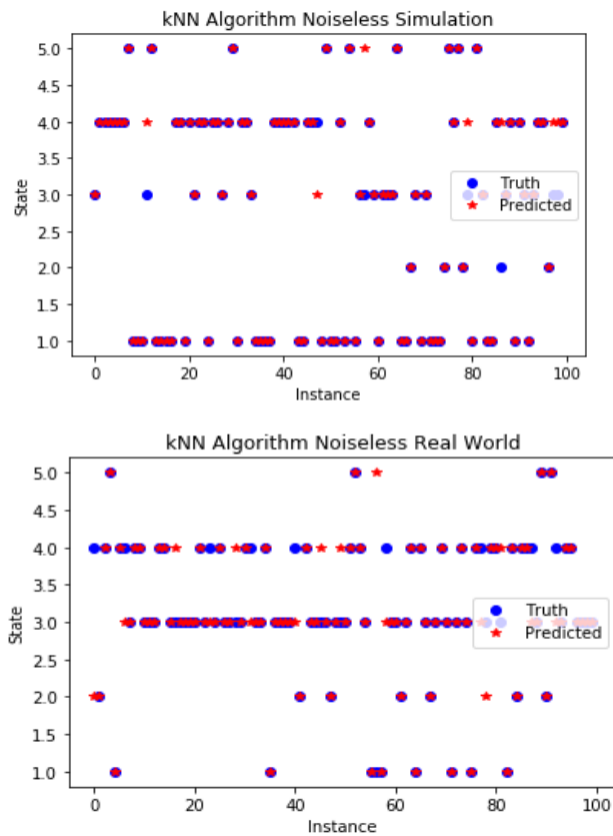


Figure 4-26 Highest Accuracy Performing Algorithm (Noiseless).

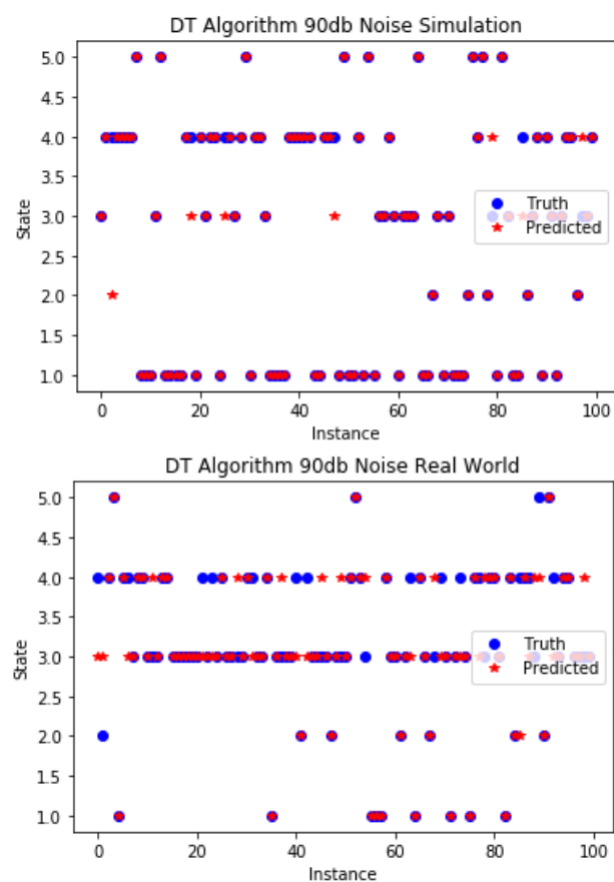


Figure 4-27 Highest Accuracy Performing Algorithm (90db).

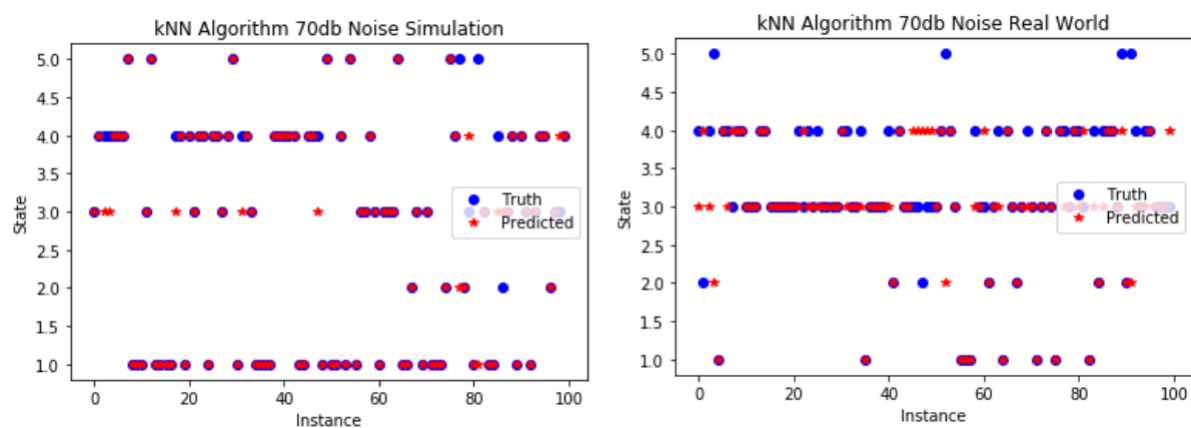


Figure 4-28 Highest Accuracy Performing Algorithm (70db).

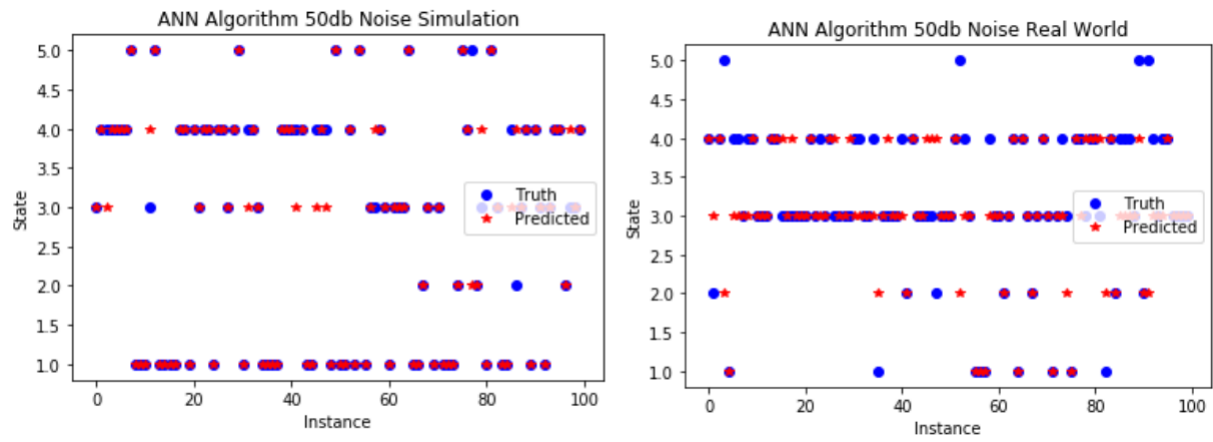


Figure 4-29 Highest Accuracy Performing Algorithm (50db).

Table 4-2 Percentage distribution table representing the accuracy of each classification method regarding the noise level.

Algorithms Noise Measurement	HOLD		TAKE OFF		HOVER		SEARCH		LAND	
	SIM	EXP	SIM	EXP	SIM	EXP	SIM	EXP	SIM	EXP
Decision Tree										
Noiseless	100%	95%	98%	96%	93%	87%	92%	75%	98%	72%
SNR 90db	100%	93%	93%	89%	83%	77%	84%	58%	93%	67%
SNR 70db	97%	94%	87%	84%	78%	75%	78%	52%	85%	59%
SNR 50db	92%	40%	66%	72%	77%	72%	74%	40%	62%	19%
1 STD	40%	0%	14%	22%	9%	12%	49%	41%	16%	15%
Naïve Bayes										
Noiseless	96%	93%	27%	81%	69%	77%	44%	27%	56%	17%
SNR 90db	98%	93%	32%	81%	64%	78%	46%	28%	69%	17%
SNR 70db	98%	93%	32%	81%	64%	77%	46%	27%	69%	17%
SNR 50db	98%	51%	32%	62%	64%	77%	46%	27%	69%	15%
1 STD	16%	0%	18%	17%	29%	3%	45%	32%	21%	10%
kNN										
Noiseless	99%	93%	98%	96%	87%	85%	88%	74%	93%	68%
SNR 90db	99%	95%	95%	84%	84%	82%	86%	64%	90%	26%
SNR 70db	98%	95%	86%	84%	79%	81%	80%	61%	74%	32%
SNR 50db	95%	64%	74%	78%	79%	73%	79%	51%	75%	11%
1 STD	32%	47%	18%	25%	12%	32%	51%	41%	18%	15%
ANN										
Noiseless	99%	95%	87%	85%	77%	82%	70%	55%	79%	16%
SNR 90db	99%	96%	86%	85%	78%	81%	75%	53%	79%	26%
SNR 70db	99%	94%	86%	85%	76%	79%	74%	54%	79%	18%
SNR 50db	99%	91%	78%	81%	75%	77%	75%	53%	79%	17%
1 STD	49%	3%	16%	24%	8%	13%	50%	42%	16%	13%
SVM										
Noiseless	99%	91%	87%	78%	74%	79%	57%	24%	81%	9%
SNR 90db	99%	91%	87%	79%	74%	79%	57%	26%	81%	9%
SNR 70db	99%	91%	87%	78%	74%	79%	57%	26%	81%	9%
SNR 50db	99%	91%	87%	79%	74%	79%	57%	26%	81%	9%
1 STD	35%	63%	16%	23%	14%	50%	50%	26%	16%	14%

CHAPTER 5

Conclusion

The main purpose of this thesis was to develop data-driven testing and evaluation of UAVs using supervised machine learning algorithms to address the increasing safety issues of UAVs. This work highlights the importance of these techniques in generating a fundamental approach for inspecting UAVs that may be used by certification companies and organizations such as the FAA to increase safety operations and regulations.

Chapter 1 explains the history behind robotics, first touching on the creation of “Unimate” and progressing to the development of very complex machines we have today. Chapter 1 points out that even though society has achieved great milestones in developing flying robotics like UAVs, it should apply more levels of safety and tests to ensure that the UAVs operating have met the necessary requirements to be operated.

Chapter 2 reviews artificial intelligence techniques and their importance in several problem domains. It briefly discusses various types of machine learning algorithms features and their applications, while also foreshadowing the uses of these techniques for data-driven UAV testing and evaluation.

In Chapter 3 we presented the mathematical modeling and state space representation of a general quadrotor. It centralizes the explanation of different types of UAVs and how they can fundamentally be represented in the simulation environment.

Chapter 4 presents our case study for machine learning algorithms modeling for overall testing and evaluation of UAVs' behavior. It provides scenarios for the simulation and real-world environment. It analyzes and compares five widely used supervised machine learning algorithms

for predicting the UAVs' behavior based on two levels of comparison, where the first comparison is based on their performance in simulation versus real world experiment and the second comparison is each algorithm's performance against one another, regarding the different environments. These algorithms were unbiasedly compared across a different selection of metrics, considering accuracy as the top metric, due to classification being the problem domain.

This work presents different classification methods that were developed for test and evaluation of unmanned quadcopters. In the post experiments, we introduce the noise to the system to test the robustness of each algorithm. Adding noise is a meaningful step to take to evaluate the experiment's robustness, since the nature of the real-world use cases are not in perfect condition. However, increasing from one standard deviation to three by using a step value of one has resulted in unstable test outcomes. Therefore, using SNR was a more feasible method while adding noise to the system. The proposed extension of this work is incorporating UGV-UAV collaboration in the scenario for multiple behavior predictions and analysis.

References

1. (n.d.). Retrieved from Science Friday: <https://www.sciencefriday.com/segments/the-origin-of-the-word-robot/>
2. (n.d.). Retrieved from All On Robots: <http://www.allonrobots.com/types-of-robots.html>
3. (n.d.). Retrieved from FAA seal: https://www.faa.gov/uas/getting_started/register_drone/
4. (2018, September 4). Retrieved from Quadcopter Arena: <https://quadcopterarena.com/the-history-of-drones-and-quadcopters/>
5. (2019, February 8). Retrieved from Cutting Tool Engineering: <https://www.ctemag.com/news/articles/evolution-of-robots>
6. Aasen, H. (2017). STATE-OF-THE-ART IN UAV REMOTE SENSING SURVEY – FIRST INSIGHTS INTO. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*.
7. Al-Sagheer, R., Alharan, A., & Al-Haboobi, A. (2017). *Popular Decision Tree Algorithms of Data Mining Techniques: A Review*. International Journal of Computer Science and Mobile Computing.
8. Amruthnath, N., & Gupta, T. (2018). A research study on unsupervised machine learning algorithms for early fault detection in predictive maintenance. In *2018 5th International Conference on Industrial Engineering and Applications (ICIEA)* (pp. 355-361).
9. Anand, S. (2018). *Artificial Intelligence - Literature Review*. Retrieved from The Centre for Internet and Society: <https://cis-india.org/internet-governance/files/artificial-intelligence-literature-review>
10. Arefin, A. S., Riveros, C., Berretta, R., & Moscato, P. (2012). *GPU-FS-kNN: a software tool for fast and scalable kNN computation using GPUs*.
11. Ayodele, T. (2010). Types of Machine Learning Algorithms. In *New Advances in Machine Learning*.
12. Beaula, A. R., Marikkannu, P., Sungheetha, A., & Sahana, C. (2016). Comparative study of distinctive image classification techniques. In *2016 10th International Conference on Intelligent Systems and Control (ISCO)* (pp. 1-8).
13. Caruana, R., & Niculescu-Mizil, A. (2006, June 29). An empirical comparison of supervised learning algorithms.
14. Chih-Wei, H., & Lin, C.-J. (2002). A Comparison of Methods for Multi-class Support Vector Machines. *IEEE Transactions on Neural Networks*.
15. Collobert, R., & Weston, J. (2008). A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. *Proceedings of the 25th international conference on Machine learning - ICML 08*.
16. Crouch, E. (2019, April 4). *Computer Vision vs. Machine Vision — What's the Difference?* Retrieved from appen: <https://appen.com/blog/computer-vision-vs-machine-vision/>

17. Dencelin, L., & Ramkumar, T. (2016). *Analysis of multilayer perceptron machine learning approach in classifying protein secondary structures*.
18. Dey, A. (2016). Machine Learning Algorithms: A Review . *International Journal of Computer Science and Information Technologies (IJCSIT)*, 1174-1179.
19. Dorr, L., & Duquette, A. (2016, June 21). *Fact Sheet – Small Unmanned Aircraft Regulations (Part 107)*. Retrieved from Federal Aviation Administration:
https://www.faa.gov/news/fact_sheets/news_story.cfm?newsId=20516
20. Frankenfield, J. (2019, June 13). *Artificial Intelligence (AI)*. Retrieved from Investopedia:
<https://www.investopedia.com/terms/a/artificial-intelligence-ai.asp>
21. Guo, G., Wang, H., Bell, D., & Bi, Y. (2004). *KNN Model-Based Approach in Classification*.
22. Gupta, N. (2017). A Literature Survey on Artificial Intelligence. *International Journal of Engineering Research & Technology (IJERT)*.
23. he, Z., & Zhao, L. (2014). A Simple Attitude Control of Quadrotor Helicopter Based on Ziegler-Nichols Rules for Tuning PD Parameters. *TheScientificWorldJournal*.
24. Hedlund, B. (2018, August 8). Drone Crashes. Retrieved from
<https://www.baumhedlundlaw.com/aviation-accident/drone-crashes/>
25. Hossin, M., & Sulaiman, M. (2015, March). A REVIEW ON EVALUATION METRICS FOR DATA CLASSIFICATION EVALUATIONS.
26. Hulden, V. (2014, Decemeber 17). *Supervised Classification: The Naive Bayesian Returns to the Old Bailey*. Retrieved from Programming Historian:
<https://programminghistorian.org/en/lessons/naive-bayesian>
27. *Industrial robots*. (2016). Retrieved from IFR:
https://ifr.org/img/office/Industrial_Robots_2016_Chapter_1_2.pdf
28. Jain, A. K., & Mao, J. (1996). Artificial neural networks: a tutorial. *Computer*.
29. Jordan, S., Moore, J., Hovet, S., Box, J., Perry, J., Kirsche, K., . . . Tse, Z. T. (2017). State-of-the-art technologies for UAV. *The institution of Engineering and Technology*.
30. Kaelbling, L. P., Littman, M. L., & Moore, A. P. (1996). Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 237-385.
31. Kamruzzaman, S. M., & Jehad Sarkar, A. M. (2011). A New Data Mining Scheme Using Artificial Neural Networks . *Sensors*.
32. Korbut, D. (2017, October 26). *Machine Learning Algorithms: Which One to Choose for Your Problem*. Retrieved from Stats and Bots: <https://blog.statsbot.co/machine-learning-algorithms-183cc73197c>
33. Kumar GN, C. (2018, August 31). *Artificial Intelligence: Definition, Types, Examples, Technologies*. Retrieved from Medium: <https://medium.com/@chethankumargn/artificial-intelligence-definition-types-examples-technologies-962ea75c7b9b>

34. Lavesson, N. (2006). *Evaluation and Analysis of Supervised Learning Algorithms and Classifiers*. Blekinge Institute of Technology.
35. Levinson, D., Boies, A., Cao, J., & Fan, Y. (2016). *The Transportation Futures Project: Planning for Technology Change*. Center for Transportation Studies, University of Minnesota. Retrieved from minnesotago.org:
https://minnesotago.org/application/files/3614/6222/6829/Autonomous_Vehicles.pdf
36. Liakos, K. G., Busat, P., Moshou, D., Pearson, S., & Bochtis, D. (2018). Machine Learning in Agriculture: A Review. *Sensors*.
37. Liu, D., Li, Y., & Thomas, M. A. (2017). A Roadmap for Natural Language Processing Research in Information Systems. *Proceedings of the 50th Hawaii International Conference on System Sciences (2017)*.
38. Marr, B. (2018, February 14). *The Key Definitions Of Artificial Intelligence (AI) That Explain Its Importance*. Retrieved from Forbes: <https://www.forbes.com/sites/bernardmarr/2018/02/14/the-key-definitions-of-artificial-intelligence-ai-that-explain-its-importance/#11516c6f4f5d>
39. Meneses, J. S., Chavez, Z. R., & Rodriguez, J. G. (2019). Compressed kNN: K-Nearest Neighbors with Data Compression. *Entropy*.
40. Norouzi Ghazbi, S., Aghli, Y., Alimohammadi, M., & Akbari, A. A. (2016). Quadrotors Unmanned Aerial Vehicles: A Review. *International Journal on Smart Sensing and Intelligent Systems*, 309-333.
41. Oke, S. A. (2008). A Literature Review on Artificial Intelligence. *International Journal of Information and Management Sciences*, 535-570.
42. Pham, H., La, H., Feil-Seifer, D., & Nguyen, L. (2018). *Autonomous UAV Navigation Using Reinforcement Learning*.
43. Praveena, M. S., & Jaiganesh, V. (2017, July). A Literature Review on Supervised Machine Learning Algorithms and Boosting Process.
44. Prusty, M., Chakraborty, J., Jayanthi, T., & Velusamy, K. (2014, December 18). Performance Comparison of Supervised Machine Learning Algorithms for Multiclass Transient Classification in a Nuclear Power Plant.
45. Rao, B., Gopi, A. G., & Maione, R. (2016). *The societal impact of commercial drones*. Technology in Society.
46. Raschka, S. (2014, October 4). *Naive Bayes and Text Classification*. Retrieved from Dr. Sebastian Raschka: https://sebastianraschka.com/Articles/2014_naive_bayes_1.html
47. Raza, K. (2016). Analysis of Microarray Data using Artificial Intelligence Based Techniques. In S. Dash, & B. Subudhi, *Handbook of Research on Computational Intelligence Applications in Bioinformatics* (pp. 865-888). IGI Global.
48. *Robotics*. (2016). Retrieved from IFR.

49. Sathya, R., & Abraham, A. (2013). Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification. *International Journal of Advanced Research in Artificial Intelligence*.
50. Schroth, L. (2018, August 8). Retrieved from Drone Industry Insights: <https://www.droneii.com/drones-and-artificial-intelligence>
51. *Service Robots*. (2016). Retrieved from IFR.
52. Sharma, H., & Kumar, S. (2016). *A Survey on Decision Tree Algorithms of Classification in Data Mining*. International Journal of Science and Research (IJSR).
53. Stevens, D., & Diesing, M. (2014, April 3). A Comparison of Supervised Classification Methods for the Prediction of Substrate Type Using Multibeam Acoustic and Legacy Grain-Size Data.
54. Suliman, A., & Zhang, Y. (2015). A Review on Back-Propagation Neural Networks in the Application of Remote Sensing Image Classification. *Journal of Earth Science and Engineering*, 52-65.
55. Sybba, Patel, P., Girghar, R., Mehta, R., Hora, R., Gupta, R., & Bardi, T. (2017). Bayes' Theorem and Operations Research. *Imperial Journal of Interdisciplinary Research (IJIR)*.
56. *Types of Artificial Intelligence*. (n.d.). Retrieved from Java T Point: <https://www.javatpoint.com/types-of-artificial-intelligence>
57. Wilton, D. (2013, November 29). *Robot*. Retrieved from Wordorigins.org: <http://www.wordorigins.org/index.php/site/comments/robot/>
58. Yao, M. (2017, September 4). *Despite AI hype, True General Intelligence Still Out of Reach*. Retrieved from Topbots: <https://www.topbots.com/despite-ai-hype-true-general-intelligence-still-reach/>

Appendix

The following code was executed for experiment and simulation. However, only experiment code was provided.

Experiment Decision Tree

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics

df = pd.read_csv('C:/Users/bbaity/Documents/Thesis//files/arDroneData.csv')
df.columns = ['xv','yv','zv','alt','label']
agg_acc = 0
for x in range(10):
    X_train, X_test, y_train, y_test = train_test_split(df[['xv','yv','zv','alt']], df['label'],
    test_size=0.3)
    dt_exp_mod = DecisionTreeClassifier(max_depth = None, criterion =
'gini').fit(X_train, y_train)
    y_pred = dt_exp_mod.predict(X_test)
    acc = metrics.accuracy_score(y_test, y_pred)
    conf_mat = metrics.confusion_matrix(y_test,y_pred)
    agg_acc += acc
mathews_coef = metrics.matthews_corrcoef(y_test,y_pred)
report = metrics.classification_report(y_test,y_pred)
print('The accuracy is',agg_acc/10)
print(conf_mat)
print('The mathew coefficient is',mathews_coef)
print(report)
```

Experiment k_nearest_Neighbors

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics

df = pd.read_csv('C:/Users/bbaity/Documents/Thesis//files/arDroneData.csv')
df.columns = ['xv','yv','zv','alt','label']
myList = list(range(1,50))
neighbors = list(filter(lambda x: x % 2 != 0, myList))
cv_scores = []
for k in neighbors:
```

```

knn = KNeighborsClassifier(n_neighbors=k)
X_train, X_test, y_train, y_test = train_test_split(df[['xv','yv','zv','alt']], df['label'],
test_size=0.3)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
scores = metrics.accuracy_score(y_test, y_pred)
cv_scores.append(scores)
MSE = [1 - x for x in cv_scores]
optimal_k = neighbors[MSE.index(min(MSE))]
plt.plot(neighbors, MSE)
plt.xlabel('Number of Neighbors K')
plt.ylabel('Misclassification Error')
plt.show()
agg_acc = 0
for x in range(10):
    optk = KNeighborsClassifier(n_neighbors=optimal_k, metric = 'manhattan')
    X_train, X_test, y_train, y_test = train_test_split(df[['xv','yv','zv','alt']], df['label'],
test_size=0.3)
    knn_exp_mod = optk.fit(X_train, y_train)
    ykopt_pred = knn_exp_mod.predict(X_test)
    acc = metrics.accuracy_score(y_test, ykopt_pred)
    conf_mat = metrics.confusion_matrix(y_test,ykopt_pred)
    report = metrics.classification_report(y_test,ykopt_pred)
    agg_acc += acc

```

Experiment Naïve Bayes

```

import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn import metrics

df = pd.read_csv('C:/Users/bbaity/Documents/Thesis//files/arDroneData.csv')
df.columns = ['xv','yv','zv','alt','label']
agg_acc = 0
for x in range(10):
    X_train, X_test, y_train, y_test = train_test_split(df[['xv','yv','zv','alt']], df['label'],
test_size=0.3)
    scaler = StandardScaler()
    scaler.fit(X_train)
    X_train = scaler.transform(X_train)
    X_test = scaler.transform(X_test)
    gnb_exp_mod = GaussianNB(priors=None).fit(X_train, y_train)
    y_pred = gnb_exp_mod.predict(X_test)
    acc = metrics.accuracy_score(y_test, y_pred)
    conf_mat = metrics.confusion_matrix(y_test,y_pred)

```

```

report = metrics.classification_report(y_test,y_pred)

agg_acc += acc
Experiment Artificial Neural Network
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import StandardScaler
from sklearn import metrics

df = pd.read_csv('C:/Users/bbaity/Documents/Thesis/files/arDroneData.csv')
df.columns = ['xv','yv','zv','alt','label']

agg_acc = 0

for x in range(10):
    X_train, X_test, y_train, y_test = train_test_split(df[['xv','yv','zv','alt']], df['label'],
test_size=0.3)
    scaler = StandardScaler()
    scaler.fit(X_train)

    X_train = scaler.transform(X_train)
    X_test = scaler.transform(X_test)

    mlp_exp_mod = MLPClassifier(activation = 'tanh', alpha = 0.0001,
                                learning_rate = 'constant', solver='adam',
                                hidden_layer_sizes = (50,100,50))
    mlp_exp_mod.fit(X_train,y_train)
    y_pred = mlp_exp_mod.predict(X_test)
    acc = metrics.accuracy_score(y_test, y_pred)
    conf_mat = metrics.confusion_matrix(y_test,y_pred)
    agg_acc += acc

Experiment Support Vector Machine
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn import metrics
from sklearn.externals import joblib

df = pd.read_csv('C:/Users/bbaity/Documents/Thesis//files/arDroneData.csv')
df.columns = ['xv','yv','zv','alt','label']
clf = svm.SVC(C = 1, gamma = 1, kernel = 'rbf')
agg_acc = 0
for x in range(10):
    X_train, X_test, y_train, y_test = train_test_split(df[['xv','yv','zv','alt']], df['label'],
test_size=0.3)

```

```

svm_exp_mod = clf.fit(X_train, y_train)
y_pred = svm_exp_mod.predict(X_test)
acc = metrics.accuracy_score(y_test, y_pred)
conf_mat = metrics.confusion_matrix(y_test, y_pred)
report = metrics.classification_report(y_test, y_pred)
agg_acc += acc

```

Clean Train Noisy Test knn, dt, nb

```

import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn import metrics

```

```

def add_1_std_noise(x):
    df=x
    std_xv = df.xv.std()
    noise_xv = np.random.normal(0, std_xv, df.xv.shape)
    noisy_xv = pd.DataFrame(df.xv + noise_xv)
    std_yv = df.yv.std()
    noise_yv = np.random.normal(0, std_yv, df.yv.shape)
    noisy_yv = pd.DataFrame(df.yv + noise_yv)
    std_zv = df.zv.std()
    noise_zv = np.random.normal(0, std_zv, df.zv.shape)
    noisy_zv = pd.DataFrame(df.zv + noise_zv)
    std_alt = df.alt.std()
    noise_alt = np.random.normal(0, std_alt, df.alt.shape)
    noisy_alt = pd.DataFrame(df.alt + noise_alt)
    sim_test_1std_data = pd.concat([noisy_xv, noisy_yv, noisy_zv, noisy_alt], axis=1,
    sort=False) return sim_test_1std_data

```

```

def add_2_std_noise(x):
    df=x
    std_xv = df.xv.std()
    noise_xv = np.random.normal(0, 2*std_xv, df.xv.shape)
    noisy_xv = pd.DataFrame(df.xv + noise_xv)
    std_yv = df.yv.std()
    noise_yv = np.random.normal(0, 2*std_yv, df.yv.shape)
    noisy_yv = pd.DataFrame(df.yv + noise_yv)
    std_zv = df.zv.std()
    noise_zv = np.random.normal(0, 2*std_zv, df.zv.shape)
    noisy_zv = pd.DataFrame(df.zv + noise_zv)

```

```

std_alt = df.alt.std()

noise_alt = np.random.normal(0, 2*std_alt, df.alt.shape)
noisy_alt = pd.DataFrame(df.alt + noise_alt)
sim_test_2std_data = pd.concat([noisy_xv, noisy_yv, noisy_zv, noisy_alt], axis=1,
sort=False) return sim_test_2std_data

def add_3_std_noise(x):
    df=x
    std_xv = df.xv.std()
    noise_xv = np.random.normal(0, 3*std_xv, df.xv.shape)
    noisy_xv = pd.DataFrame(df.xv + noise_xv)
    std_yv = df.yv.std()
    noise_yv = np.random.normal(0, 3*std_yv, df.yv.shape)
    noisy_yv = pd.DataFrame(df.yv + noise_yv)
    std_zv = df.zv.std()
    noise_zv = np.random.normal(0, 3*std_zv, df.zv.shape)
    noisy_zv = pd.DataFrame(df.zv + noise_zv)
    std_alt = df.alt.std()
    noise_alt = np.random.normal(0, 3*std_alt, df.alt.shape)
    noisy_alt = pd.DataFrame(df.alt + noise_alt)
    sim_test_3std_data = pd.concat([noisy_xv, noisy_yv, noisy_zv, noisy_alt], axis=1,
sort=False) return sim_test_3std_data

df = pd.read_csv('C:/Users/bbaity/Documents/Thesis//files/arDroneData.csv')
df.columns = ['xv','yv','zv','alt','label']

dt_agg_acc = 0
dt_agg_acc_1std = 0
dt_agg_acc_2std = 0
dt_agg_acc_3std = 0

nb_agg_acc = 0
nb_agg_acc_1std = 0
nb_agg_acc_2std = 0
nb_agg_acc_3std = 0

knn_agg_acc = 0
knn_agg_acc_1std = 0
knn_agg_acc_2std = 0
knn_agg_acc_3std = 0

for x in range(10):
    X_train, X_test, y_train, y_test = train_test_split(df[['xv','yv','zv','alt']], df['label'],
test_size=0.3)

```

```

scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

#####
#####
dt_exp_mod = DecisionTreeClassifier(max_depth = None, criterion =
'gini').fit(X_train, y_train)
dt_y_pred = dt_exp_mod.predict(X_test)
dt_acc = metrics.accuracy_score(y_test, dt_y_pred)
dt_conf_mat = metrics.confusion_matrix(y_test, dt_y_pred)
dt_agg_acc += dt_acc
dt_report = metrics.classification_report(y_test, dt_y_pred)

X_test_1std = add_1_std_noise(X_test)
dt_y_pred_1std = dt_exp_mod.predict(X_test_1std)
dt_acc_1std = metrics.accuracy_score(y_test, dt_y_pred_1std)
dt_conf_mat_1std = metrics.confusion_matrix(y_test, dt_y_pred_1std)
dt_agg_acc_1std += dt_acc_1std
dt_report_1std = metrics.classification_report(y_test, dt_y_pred_1std)

X_test_2std = add_2_std_noise(X_test)
dt_y_pred_2std = dt_exp_mod.predict(X_test_2std)
dt_acc_2std = metrics.accuracy_score(y_test, dt_y_pred_2std)
dt_conf_mat_2std = metrics.confusion_matrix(y_test, dt_y_pred_2std)
dt_agg_acc_2std += dt_acc_2std
dt_report_2std = metrics.classification_report(y_test, dt_y_pred_2std)

X_test_3std = add_3_std_noise(X_test)
dt_y_pred_3std = dt_exp_mod.predict(X_test_3std)
dt_acc_3std = metrics.accuracy_score(y_test, dt_y_pred_3std)
dt_conf_mat_3std = metrics.confusion_matrix(y_test, dt_y_pred_3std)
dt_agg_acc_3std += dt_acc_3std
dt_report_3std = metrics.classification_report(y_test, dt_y_pred_3std)
#####
#####

nb_exp_mod = GaussianNB(priors=None).fit(X_train, y_train)
nb_y_pred = nb_exp_mod.predict(X_test)
nb_acc = metrics.accuracy_score(y_test, nb_y_pred)
nb_conf_mat = metrics.confusion_matrix(y_test, nb_y_pred)
nb_agg_acc += nb_acc

```

```
nb_report = metrics.classification_report(y_test,nb_y_pred)
```

```
X_test_1std = add_1_std_noise(X_test)
nb_y_pred_1std = nb_exp_mod.predict(X_test_1std)
nb_acc_1std = metrics.accuracy_score(y_test, nb_y_pred_1std)
nb_conf_mat_1std = metrics.confusion_matrix(y_test,nb_y_pred_1std)
nb_agg_acc_1std += nb_acc_1std
nb_report_1std = metrics.classification_report(y_test,nb_y_pred_1std)
```

```
X_test_2std = add_2_std_noise(X_test)
nb_y_pred_2std = nb_exp_mod.predict(X_test_2std)
nb_acc_2std = metrics.accuracy_score(y_test, nb_y_pred_2std)
nb_conf_mat_2std = metrics.confusion_matrix(y_test,nb_y_pred_2std)
nb_agg_acc_2std += nb_acc_2std
nb_report_2std = metrics.classification_report(y_test,nb_y_pred_2std)
```

```
X_test_3std = add_3_std_noise(X_test)
nb_y_pred_3std = nb_exp_mod.predict(X_test_3std)
nb_acc_3std = metrics.accuracy_score(y_test, nb_y_pred_3std)
nb_conf_mat_3std = metrics.confusion_matrix(y_test,nb_y_pred_3std)
nb_agg_acc_3std += nb_acc_3std
nb_report_3std = metrics.classification_report(y_test,nb_y_pred_3std)
```

```
#####
#####
```

```
myList = list(range(1,50))
neighbors = list(filter(lambda x: x % 2 != 0, myList))
cv_scores = []
for k in neighbors:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    optk_y_pred = knn.predict(X_test)
    scores = metrics.accuracy_score(y_test, optk_y_pred)
    cv_scores.append(scores)
    MSE = [1 - x for x in cv_scores]
    optimal_k = neighbors[MSE.index(min(MSE))]
```

```
optk = KNeighborsClassifier(n_neighbors=optimal_k, metric = 'manhattan')
knn_exp_mod = optk.fit(X_train, y_train)
knn_y_pred = knn_exp_mod.predict(X_test)
knn_acc = metrics.accuracy_score(y_test, knn_y_pred)
knn_conf_mat = metrics.confusion_matrix(y_test,knn_y_pred)
knn_agg_acc += knn_acc
knn_report = metrics.classification_report(y_test,knn_y_pred)
```



```

X_test_1std = add_1_std_noise(X_test)
knn_y_pred_1std = knn_exp_mod.predict(X_test_1std)
knn_acc_1std = metrics.accuracy_score(y_test, knn_y_pred_1std)
knn_conf_mat_1std = metrics.confusion_matrix(y_test, knn_y_pred_1std)
knn_agg_acc_1std += knn_acc_1std
knn_report_1std = metrics.classification_report(y_test, knn_y_pred_1std)

```

```

X_test_2std = add_2_std_noise(X_test)
knn_y_pred_2std = knn_exp_mod.predict(X_test_2std)
knn_acc_2std = metrics.accuracy_score(y_test, knn_y_pred_2std)
knn_conf_mat_2std = metrics.confusion_matrix(y_test, knn_y_pred_2std)
knn_agg_acc_2std += knn_acc_2std
knn_report_2std = metrics.classification_report(y_test, knn_y_pred_2std)

```

```

X_test_3std = add_3_std_noise(X_test)
knn_y_pred_3std = knn_exp_mod.predict(X_test_3std)
knn_acc_3std = metrics.accuracy_score(y_test, knn_y_pred_3std)
knn_conf_mat_3std = metrics.confusion_matrix(y_test, knn_y_pred_3std)
knn_agg_acc_3std += knn_acc_3std
knn_report_3std = metrics.classification_report(y_test, knn_y_pred_3std)

```

Clean Train Noisy Test SVM

```

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn import metrics

def add_1_std_noise(x):
    df=x
    std_xv = df.xv.std()
    noise_xv = np.random.normal(0, std_xv, df.xv.shape)
    noisy_xv = pd.DataFrame(df.xv + noise_xv)
    std_yv = df.yv.std()
    noise_yv = np.random.normal(0, std_yv, df.yv.shape)
    noisy_yv = pd.DataFrame(df.yv + noise_yv)
    std_zv = df.zv.std()
    noise_zv = np.random.normal(0, std_zv, df.zv.shape)
    noisy_zv = pd.DataFrame(df.zv + noise_zv)
    std_alt = df.alt.std()
    noise_alt = np.random.normal(0, std_alt, df.alt.shape)
    noisy_alt = pd.DataFrame(df.alt + noise_alt)
    sim_test_1std_data = pd.concat([noisy_xv, noisy_yv, noisy_zv, noisy_alt], axis=1, sort=False)
    return sim_test_1std_data

def add_2_std_noise(x):

```

```

df=x
std_xv = df.xv.std()
noise_xv = np.random.normal(0, 2*std_xv, df.xv.shape)
noisy_xv = pd.DataFrame(df.xv + noise_xv)
std_yv = df.yv.std()
noise_yv = np.random.normal(0, 2*std_yv, df.yv.shape)
noisy_yv = pd.DataFrame(df.yv + noise_yv)
std_zv = df.zv.std()
noise_zv = np.random.normal(0, 2*std_zv, df.zv.shape)
noisy_zv = pd.DataFrame(df.zv + noise_zv)
std_alt = df.alt.std()
noise_alt = np.random.normal(0, 2*std_alt, df.alt.shape)
noisy_alt = pd.DataFrame(df.alt + noise_alt)
sim_test_2std_data = pd.concat([noisy_xv, noisy_yv, noisy_zv, noisy_alt], axis=1,
sort=False) return sim_test_2std_data

```

```

def add_3_std_noise(x):
    df=x
    std_xv = df.xv.std()
    noise_xv = np.random.normal(0, 3*std_xv, df.xv.shape)
    noisy_xv = pd.DataFrame(df.xv + noise_xv)
    std_yv = df.yv.std()
    noise_yv = np.random.normal(0, 3*std_yv, df.yv.shape)
    noisy_yv = pd.DataFrame(df.yv + noise_yv)
    std_zv = df.zv.std()
    noise_zv = np.random.normal(0, 3*std_zv, df.zv.shape)
    noisy_zv = pd.DataFrame(df.zv + noise_zv)
    std_alt = df.alt.std()
    noise_alt = np.random.normal(0, 3*std_alt, df.alt.shape)
    noisy_alt = pd.DataFrame(df.alt + noise_alt)
    sim_test_3std_data = pd.concat([noisy_xv, noisy_yv, noisy_zv, noisy_alt], axis=1,
sort=False) return sim_test_3std_data

```

```

svm_agg_acc = 0
svm_agg_acc_1std = 0
svm_agg_acc_2std = 0
svm_agg_acc_3std = 0

```

```

df = pd.read_csv('C:/Users/bbaity/Documents/Thesis//files/arDroneData.csv')
df.columns = ['xv','yv','zv','alt','label']

```

```

for x in range(10):
    X_train, X_test, y_train, y_test = train_test_split(df[['xv','yv','zv','alt']], df['label'],
test_size=0.3)
    X_train = scaler.transform(X_train)
    X_test = scaler.transform(X_test)

```

```

clf = svm.SVC(C = 10, gamma = 1, kernel = 'rbf')
svm_exp_mod = clf.fit(X_train, y_train)
svm_y_pred = svm_exp_mod.predict(X_test)
svm_acc = metrics.accuracy_score(y_test, svm_y_pred)
svm_conf_mat = metrics.confusion_matrix(y_test,svm_y_pred)
svm_agg_acc += svm_acc
svm_report = metrics.classification_report(y_test,svm_y_pred)

X_test_1std = add_1_std_noise(X_test)
svm_y_pred_1std = svm_exp_mod.predict(X_test_1std)
svm_acc_1std = metrics.accuracy_score(y_test, svm_y_pred_1std)
svm_conf_mat_1std = metrics.confusion_matrix(y_test,svm_y_pred_1std)
svm_agg_acc_1std += svm_acc_1std
svm_report_1std = metrics.classification_report(y_test,svm_y_pred_1std)

X_test_2std = add_2_std_noise(X_test)
svm_y_pred_2std = svm_exp_mod.predict(X_test_2std)
svm_acc_2std = metrics.accuracy_score(y_test, svm_y_pred_2std)
svm_conf_mat_2std = metrics.confusion_matrix(y_test,svm_y_pred_2std)
svm_agg_acc_2std += svm_acc_2std
svm_report_2std = metrics.classification_report(y_test,svm_y_pred_2std)

X_test_3std = add_3_std_noise(X_test)
svm_y_pred_3std = svm_exp_mod.predict(X_test_3std)
svm_acc_3std = metrics.accuracy_score(y_test, svm_y_pred_3std)
svm_conf_mat_3std = metrics.confusion_matrix(y_test,svm_y_pred_3std)
svm_agg_acc_3std += svm_acc_3std
svm_report_3std = metrics.classification_report(y_test,svm_y_pred_3std)

```

Clean Train Noisy Test ANN

```

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn import metrics

def add_1_std_noise(x):
    df=x
    std_xv = df.xv.std()
    noise_xv = np.random.normal(0, std_xv, df.xv.shape)
    noisy_xv = pd.DataFrame(df.xv + noise_xv)
    std_yv = df.yv.std()
    noise_yv = np.random.normal(0, std_yv, df.yv.shape)
    noisy_yv = pd.DataFrame(df.yv + noise_yv)
    std_zv = df.zv.std()

```

```

noise_zv = np.random.normal(0, std_zv, df.zv.shape)
noisy_zv = pd.DataFrame(df.zv + noise_zv)
std_alt = df.alt.std()
noise_alt = np.random.normal(0, std_alt, df.alt.shape)
noisy_alt = pd.DataFrame(df.alt + noise_alt)
sim_test_1std_data = pd.concat([noisy_xv, noisy_yv, noisy_zv, noisy_alt], axis=1,
sort=False) return sim_test_1std_data

```

```

def add_2_std_noise(x):
    df=x
    std_xv = df.xv.std()
    noise_xv = np.random.normal(0, 2*std_xv, df.xv.shape)
    noisy_xv = pd.DataFrame(df.xv + noise_xv)
    std_yv = df.yv.std()
    noise_yv = np.random.normal(0, 2*std_yv, df.yv.shape)
    noisy_yv = pd.DataFrame(df.yv + noise_yv)
    std_zv = df.zv.std()
    noise_zv = np.random.normal(0, 2*std_zv, df.zv.shape)
    noisy_zv = pd.DataFrame(df.zv + noise_zv)
    std_alt = df.alt.std()
    noise_alt = np.random.normal(0, 2*std_alt, df.alt.shape)
    noisy_alt = pd.DataFrame(df.alt + noise_alt)
    sim_test_2std_data = pd.concat([noisy_xv, noisy_yv, noisy_zv, noisy_alt], axis=1,
sort=False) return sim_test_2std_data

```

```

def add_3_std_noise(x):
    df=x
    std_xv = df.xv.std()
    noise_xv = np.random.normal(0, 3*std_xv, df.xv.shape)
    noisy_xv = pd.DataFrame(df.xv + noise_xv)
    std_yv = df.yv.std()
    noise_yv = np.random.normal(0, 3*std_yv, df.yv.shape)
    noisy_yv = pd.DataFrame(df.yv + noise_yv)
    std_zv = df.zv.std()
    noise_zv = np.random.normal(0, 3*std_zv, df.zv.shape)
    noisy_zv = pd.DataFrame(df.zv + noise_zv)
    std_alt = df.alt.std()
    noise_alt = np.random.normal(0, 3*std_alt, df.alt.shape)
    noisy_alt = pd.DataFrame(df.alt + noise_alt)
    sim_test_3std_data = pd.concat([noisy_xv, noisy_yv, noisy_zv, noisy_alt], axis=1,
sort=False) return sim_test_3std_data

```

```

ann_agg_acc = 0
ann_agg_acc_1std = 0
ann_agg_acc_2std = 0
ann_agg_acc_3std = 0

```

```

df = pd.read_csv('C:/Users/bbaity/Documents/Thesis//files/arDroneData.csv')
df.columns = ['xv','yv','zv','alt','label']

for x in range(10):
    X_train, X_test, y_train, y_test = train_test_split(df[['xv','yv','zv','alt']], df['label'],
test_size=0.3)
    X_train = scaler.transform(X_train)
    X_test = scaler.transform(X_test)

    ann_exp_mod = MLPClassifier(activation = 'tanh', alpha = 0.0001,
                                learning_rate = 'constant', solver='adam',
                                hidden_layer_sizes = (50,100,50))
    ann_exp_mod.fit(X_train,y_train)
    ann_y_pred = ann_exp_mod.predict(X_test)
    ann_acc = metrics.accuracy_score(y_test, ann_y_pred)
    ann_conf_mat = metrics.confusion_matrix(y_test,ann_y_pred)
    ann_agg_acc += ann_acc
    ann_report = metrics.classification_report(y_test,ann_y_pred)

    X_test_1std = add_1_std_noise(X_test)
    ann_y_pred_1std = ann_exp_mod.predict(X_test_1std)
    ann_acc_1std = metrics.accuracy_score(y_test, ann_y_pred_1std)
    ann_conf_mat_1std = metrics.confusion_matrix(y_test,ann_y_pred_1std)
    ann_agg_acc_1std += ann_acc_1std
    ann_report_1std = metrics.classification_report(y_test,ann_y_pred_1std)

    X_test_2std = add_2_std_noise(X_test)
    ann_y_pred_2std = ann_exp_mod.predict(X_test_2std)
    ann_acc_2std = metrics.accuracy_score(y_test, ann_y_pred_2std)
    ann_conf_mat_2std = metrics.confusion_matrix(y_test,ann_y_pred_2std)
    ann_agg_acc_2std += ann_acc_2std
    ann_report_2std = metrics.classification_report(y_test,ann_y_pred_2std)

    X_test_3std = add_3_std_noise(X_test)
    ann_y_pred_3std = ann_exp_mod.predict(X_test_3std)
    ann_acc_3std = metrics.accuracy_score(y_test, ann_y_pred_3std)
    ann_conf_mat_3std = metrics.confusion_matrix(y_test,ann_y_pred_3std)
    ann_agg_acc_3std += ann_acc_3std
    ann_report_3std = metrics.classification_report(y_test,ann_y_pred_3std)

```