NAVAL
POSTGRADUATE
SCHOOL

MONTEREY, CALIFORNIA

THESIS

CLASSIFICATION OF BOLIDES AND METEORS IN DOPPLER RADAR WEATHER DATA USING UNSUPERVISED MACHINE LEARNING

by

Brendon P. Smeresky

December 2019

Thesis Advisor: Mark Karpenko
Co-Advisor: Paul Abell,
NASA Johnson Space Center
Second Reader: Lyn R. Whitaker

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** December 2019	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE CLASSIFICATION OF BOLIDES AND METEORS IN DOPPLER RADAR WEATHER DATA USING UNSUPERVISED MACHINE LEARNING	**5. FUNDING NUMBERS**
6. AUTHOR(S) Brendon P. Smeresky	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

This thesis presents a method for detecting outlier meteors and bolides within Doppler radar data using unsupervised machine learning. Principal Component Analysis (PCA), k-means Clustering, and t-Distributed Statistical Neighbor Embedding (t-SNE) algorithms are introduced as existing methods for outlier detection. A combined PCA and t-SNE method that uses a Nearest Neighbor Density Pruning method for dataset size reduction is also described. These methods are implemented to classify unlabeled radar data from four radar data sites from two bolide events: the KFWS radar for the Ash Creek bolide and the KDAX, KRGX, and KBBX radars for the Sutter's Mill bolide. The combined PCA + t-SNE method gives an accuracy rate of 99.7% and can classify the data in less than 8 minutes for a 121,000 return sized dataset. However, the classifier's recall and precision rates remained low due to difficulties in correctly classifying true positive bolides. Some ideas for improving algorithm accuracy, speed, and related follow-on applications are proposed. Overall, the algorithm presented in this research is a viable method to help NASA scientists with bolide detection and meteorite recovery.

14. SUBJECT TERMS asteroids, bolides, meteors, artificial intelligence, machine learning, unsupervised machine learning, principal component analysis, clustering, t-SNE, Doppler radar, nearest neighbors	**15. NUMBER OF PAGES** 119
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

CLASSIFICATION OF BOLIDES AND METEORS IN DOPPLER RADAR WEATHER DATA USING UNSUPERVISED MACHINE LEARNING

Brendon P. Smeresky
Lieutenant Commander, United States Navy
BS, U.S. Naval Academy, 2006
MS, Florida Institute of Technology, 2015

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ASTRONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2019

Approved by: Mark Karpenko
Advisor

Paul Abell
Co-Advisor

Lyn R. Whitaker
Second Reader

Garth V. Hobson
Chair, Department of Mechanical and Aerospace Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis presents a method for detecting outlier meteors and bolides within Doppler radar data using unsupervised machine learning. Principal Component Analysis (PCA), k-means Clustering, and t-Distributed Statistical Neighbor Embedding (t-SNE) algorithms are introduced as existing methods for outlier detection. A combined PCA and t-SNE method that uses a Nearest Neighbor Density Pruning method for dataset size reduction is also described. These methods are implemented to classify unlabeled radar data from four radar data sites from two bolide events: the KFWS radar for the Ash Creek bolide and the KDAX, KRGX, and KBBX radars for the Sutter's Mill bolide. The combined PCA + t-SNE method gives an accuracy rate of 99.7% and can classify the data in less than 8 minutes for a 121,000 return sized dataset. However, the classifier's recall and precision rates remained low due to difficulties in correctly classifying true positive bolides. Some ideas for improving algorithm accuracy, speed, and related follow-on applications are proposed. Overall, the algorithm presented in this research is a viable method to help NASA scientists with bolide detection and meteorite recovery.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AU	Astronomical Units
CSV	Comma Separated Value
FN	False Negative
FP	False Positive
KBBX	Beale Air Force Base, California, Doppler Radar Site
KDAX	Sacramento, California, Doppler Radar Site
KFWS	Ash Creek, Texas, Doppler Radar Site
KNN	K Nearest Neighbors
KRGX	Reno, Nevada Doppler Radar Site
NASA	National Aeronautics and Space Administration
NaN	Not a Number
netCDF	Network Common Data Form
NN	Nearest Neighbor
NNDP	Nearest Neighbor Density Pruning
NOAA	National Oceanic and Atmospheric Administration
PC	Principal Component
PCA	Principal Component Analysis
TN	True Negative
TP	True Positive
WCT	Weather and Climate Toolkit
WSR-88D	Weather Surveillance Radar–1988 Doppler

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would firstly like to thank my thesis advisors for their support and guidance throughout this research and my graduate program in general; I would not be where I am without the hours I spent talking to each of you. Dr. Mark Karpenko, you were always available to give guidance and talk about topics like engineering and optimization, future research into machine learning projects, or even the LEGO models we were building with our kids. I have enjoyed our talks more than you will ever know, and will look back fondly upon them. Dr. Lyn Whitaker, your lessons into machine learning were invaluable for my thesis and my development as a scientist. I appreciate you opening your door to a random student that wasn’t even in the OR program; I have taken a lot away from your lessons about school and life. Dr. Abell, you are the one that started me on this path, and I have learned a lot about NASA and planetary science because of you. I am thankful for the time you took out of your busy schedule for a graduate student, and truly hope that my research helps you and NASA in your endeavors.

Secondly, I would like to thank some of the people who supported my research and development. Although they were not my advisors, they provided valuable insight into various fields that I would not have without their inputs. Dr. I. Michael Ross, I count you as a pseudo advisor because of the number of times I came up to your office to go over thesis research, optimization topics, and academic planning. I am thankful for your help in my path as a student and researcher, and found your and Dr. Karpenko’s classes extremely rewarding. Dr. Paul Harasti, you provided insight into weather and radar systems and subject matter expertise that I could not have received anywhere else but the Fleet Numerical Meteorology and Oceanography Center. You enabled me to progress in my thesis early on when I was struggling to decipher my dataset. Mike Hankey, thank you for taking time away from your busy schedule to provide some of the expertise you have gained working on asteroid and bolide detection for the American Meteor Society. Your research was a significant starting point for my own, and I greatly appreciate the guidance you gave me early on in research directions to take.

Lastly, I have to thank my wife and children. NPS has been a difficult 2.5 years that has consumed far too much of my time. That time was taken from what I should have spent with you and I owe each of you so much for allowing me to study late every day of the week, supporting me, and not making me choose between school and family. Allison, you took care of the children while maintaining a job of your own, and I thank you for that, despite all the rough patches over the past few years. You also helped me by editing some of my papers and acting as a sounding board for some of my math and engineering calculations. To Ariana and Orion, I hope I have shown you how to endure through challenges. I know I have missed part of your childhood, but I aim to be a greater part of it now and in the future. I hope that as you progress in your schoolwork, you can look back at the time when I did homework and studied alongside you. I hope you always study and grow, despite when it is hard.

Finally, I need to reiterate to Allison that although I now have two graduate degrees, you still received yours before either of mine and are therefore far smarter than me.[1]

[1] Deppe, A. M., 2007, "Contact Stress in a Whole Joint Bioreactor: Intrinsic Levels & Augmentation by External Loading," PhD Thesis, UC San Diego.

I. INTRODUCTION

There are countless asteroids in orbit around the Sun, leftover from the creation of the solar system 4.6 billion years ago. An estimated 1.1 to 1.9 million asteroids greater than 1 km in size exist in the main asteroid belt between Mars and Jupiter alone, and as of 12 November 2019, a total of 21,379 asteroids have been found to have trajectories that take them across Earth's orbit. Of these, 901 of which have a diameter greater than 1 km [1]. Asteroids of varying sizes regularly come close enough to enter into the Earth's atmosphere, as seen in Figure 1; however, it is inevitable that another asteroid of a massive enough in size will penetrate all the way to the surface, impacting the ground and generating a crater like that near Winslow, Arizona, or explode in the atmosphere, generating a blast wave such as in the case of the Chelyabinsk, Russia event, denoted as the red circle in Russia in Figure 1 [2]. Therefore, the National Aeronautics and Space Administration (NASA) has been tasked with studying such objects in order to understand the potential threats that they pose to Earth's population. Even a relatively small asteroid impacting over a major population area would result in an extreme loss of life, and larger impactors have the potential to result in severe global catastrophic damage. In the extreme case, it is possible that Earth could experience another extinction level event such as the Chicxulub impact, which resulted in the extinction of the dinosaurs 65 million years ago [3].

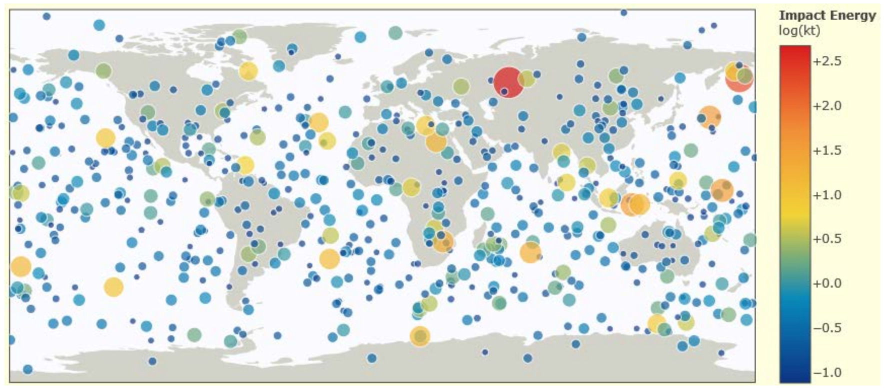


Figure 1. Bolides reported by U.S. government sensors 15 April 1988 to 05 November 2019. Source: [2].

A. MOTIVATION

The motivation behind NASA’s research is to increase the recovery rate of meteorite remains on Earth. In order to better defend our planet against asteroid strikes, as much information as possible needs to be understood about the composition and physical attributes of asteroids. Therefore, one of NASA’s goals is to better characterize the asteroid population through the recovery and subsequent analysis of these asteroid fragments (e.g., meteorites) that exist earthbound [4,5].

Unfortunately, the research longevity of meteorite fragments is relatively short. This is because the longer the amount of time has passed after a meteorite has reached the surface, the higher the likelihood that it has encountered inclement weather, native bacteria, or humans that have spoiled or acquired the sample, rendering the fragment’s research viability decreased [6]. Therefore, it is in NASA’s best interests to identify the fall locations of any meteorites as quickly as possible so they can rapidly dispatch a recovery team for recovery.

In order to recover a meteorite, the impact site must be found. This is completed through trajectory analysis via a dark flight model that predicts the debris field location from the position of the meteor in the sky, its trajectory, and the atmospheric condition at the time it was observed [7,8]. Previous research has already completed this task, but efforts still need to be completed to better determine the inputs to the dark flight model [6]. Therefore, all the potential meteor sighting reports must be analyzed and verified so that analysis with the dark flight model is not wasted on false sightings.

It is estimated that only 0.1% of recoverable meteorites are reported, which are the meteors that burn the brightest when they enter the atmosphere and often explode, gaining the name of “bolides” [9]. These 0.1% are often reported by civilian observers on websites like the American Meteor Society’s website, through social media, or in news reports [10,11]. However, even with this help, finding actual meteor events is difficult for humans. Therefore, machine-based methods are a potential source for identifying and verifying bolide fall locations among all the information that has been traditionally used to constrain meteorite fall sites [12].

B. PROBLEM STATEMENT

This thesis focuses on one non-human source of bolide information: Doppler radar data from the National Oceanic and Atmospheric Administration (NOAA). However, due to the volume of data collected by the weather radar sites, the number of potential bolide events that need to be verified, and the speed requirement for meteorite recovery, automation is essential. Therefore, the goal of this thesis is to determine whether it is feasible or not to detect and classify bolides in Doppler radar data using machine learning techniques to support automation. The method to do so is by building a detection and classification algorithm and employing it against known bolide events. At its core, the problem to be solved is an outlier detection problem, where the goal is to identify the outlier meteor events within all the terrestrial weather radar noise.

C. BACKGROUND

A brief introduction into relevant topics is helpful in understanding the scope of this thesis and the research completed by previous scientists. Therefore, the nomenclature of Near-Earth Objects, an overview of radar theory, and a summary of machine learning are introduced as they relate to the problem statement. The goal is not to provide a comprehensive background, but rather to establish a common foundation of knowledge for the rest of this thesis.

1. Nomenclature of Near-Earth Objects

In order to understand why bolide and meteor detection is important, it is first necessary to understand the difference between the terms used up to this point. Near-Earth Objects are defined as either asteroids or comets (composed of rock and metal or ice and rock respectively), that orbit within 1.3 astronomical units (AU) of the Sun [13]. They vary vastly in size and orbit trajectories, but are viewed as remnants from the formation of our solar system. This makes them a valuable and desirable source of information into a window from history 4.6 billion years ago [14]. Furthermore, they pose a risk to humanity, and according to Section 321 of the NASA Authorization Act of 2005, NASA’s objectives are to characterize the physical characteristics of Near-Earth Objects, among other tasks [13]. Through the study of asteroid interactions with the Earth’s atmosphere, and the subsequent meteorite fragments that survive to reach the surface, mitigation strategies to deflect, or neutralize an incoming asteroid can be created and refined.

The subset of Near-Earth Objects that are predominantly asteroids are known as Near-Earth Asteroids, of which there are 4 groups depending on their orbit. Atiras are asteroids that orbit inside the Earth's orbit around the sun. Amors are asteroids that orbit between the Earth and Mars. Lastly, Atens and Apollos are asteroids that have their semi-major axis smaller or greater than that of the Earth's semi-major axis, causing them to cross Earth's orbit [14]. The importance of all these types of asteroids is that their orbits put them in proximity with the Earth and due to gravitational perturbations, could pose a potential impact risk.

When an asteroid orbits close enough to Earth, it is pulled into Earth's gravity well. When it does, it enters the atmosphere and impacts the gasses in the atmosphere while traveling at speeds around 11 to 73 km/s [15,16]. This friction causes the asteroid to heat up and ablate, where the hot gasses enter the Earth's atmosphere and irradiate light. The emission of light makes the asteroid visible to observers and gains the asteroid an alternate name of a meteor (or meteoroid for small asteroids < 1 m in diameter). For the meteors that are excessively bright and detonate during orbital entry, they gain the additional nomenclature of bolides [15]. However, some meteors are large enough in size that their velocities decrease in the atmosphere before ablating completely. When this happens, they enter a dark flight phase, where the meteor remnants continue descending towards the ground without emitting light. Those remnants that reach the surface of the Earth are known as meteorites [15].

The ablative phase of the meteor's descent prior to dark flight is of importance to this research. This is because as the meteor ablates and fragments, it enters the atmosphere where both the meteor and the expanding cloud of debris can reflect radar energy [16]. This solid core of reflected energy surrounded by lower magnitude radar returns are therefore associated with the solid meteor and the ionized tail of ablative material, respectively, and provide a marker to look for when identifying meteors and bolides within Doppler radar returns [15].

2. Radar Theory

The theory of how radars function is based upon transmitted and received electromagnetic energy. Using Equation 1, a radar transmits energy at the speed of light, *c*, with a certain wave frequency, f, and wavelength, λ [17]. The wave propagates outwards from the transmitter is either absorbed, refracted, or reflected by all objects in its path. A small fraction of the radiated energy is directed back towards the radar's receiver, usually collocated with the transmitter, and measured [18]. The returned electromagnetic energy, E, has a signal amplitude A which is a function of the angle θ and phase ϕ, range r, frequency *f*, time t, and signal phase ψ, as seen in Equation 2. Using this information with Equations 1 and 2, the radar system calculates the range from the receiver to the pulse, and certain radars can determine other factors like speed or shape of the object that reflected the energy [17]. For this thesis, the radar is used as a near constant source of information regarding meteors within the atmosphere.

$$c = \lambda f = 3 \times 10^{8} \, m/s \tag{1}$$

$$E = \frac{A(\theta, \phi)}{r} e^{j2\pi f (t - \frac{r}{c}) + j\psi} \tag{2}$$

3. Machine Learning

Artificial Intelligence (AI) is the study of how to build rationally acting agents. The intent is to build an intelligent entity that behaves based on what it knows in order to accomplish a stated goal [19]. This is no easy feat, and multiple skills are needed to accomplish this intent. The agent must have knowledge representation to store information, apply that stored knowledge to determine courses of action, and a learning mechanism to improve and adjust based on the environment in which the agent operates [20]. The study of AI is multidisciplinary, drawing inspiration and perspective from fields such as philosophy, mathematics, economics, neuroscience, psychology, linguistic, computer science, and control theory. This multidisciplinary approach results in different fields approaching the problem of building artificial agents in different ways [19]. This thesis follows in that vein, applying elements of planetary science, radar theory, artificial intelligence, and machine learning together to solve the outlier detection problem of identifying bolides and meteors within Doppler radar data.

D. REVIEW OF LITERATURE

Science advances due to the incremental steps made by each successive generation of scientists. This section pays homage to the dedication and efforts of previous researchers in the fields of meteorology, planetary science, and meteor detection. Their methods paved the way for this manuscript and its multidisciplinary approach.

1. Doppler Radar

Research into Doppler radar extends back into World War II. However, from a weather perspective the 1979 seminal paper by Doviak et al. in [21] provided a comprehensive overview into the subject of identifying and using weather echoes, from which a significant body of work is derived. In the decades since their work was completed, both radar hardware and radar analysis software advanced enough that Schmidt et al. could identify individual rain drops from within radar observations [22]. In those observations, small particles could be identified using power spectrum analysis and those small particles were identified as composed of materials other than liquid water or ice. This was further affirmed by Kent et al. who used NASA’s Debris Radar system, a Doppler radar, to detect and characterize debris from the Space Shuttle during launch [23]. However, one of the larger uses of Doppler radar became the tracking of volcanic ash, which this research views as a conceptual proxy for meteor debris and is used to understand ranges of reflectivity values for rocky debris [24–26]. Unfortunately, the value as a proxy is limited because the magnitude of debris generated during a volcanic eruption was significantly larger than that generated during the re-entry and ablation of a meteor.

2. Meteor Analysis

Similar to radar theory, research into asteroids and meteors has a history that extends to the early 20th century and continued to develop throughout the century. For instance, the research of Ceplecha et al. in 1998 is an excellent primer on the subject [15]. This work provides a thorough analysis on several subjects of importance to this thesis, including the origins of asteroids within the solar system, phases of flight of meteors inside the atmosphere, and methods of detection. Then, in 2018, the work of Silber et al. added to the overall meteor knowledge base through a summary paper that focuses upon the interactions of meteors within the atmosphere, their ablation, and fragmentation [16]. Contrasting against these overview papers is the research conducted by Simon et al. [27] and Brown et al. [8] that focuses upon the Park Forest meteorite on 26 March 2003, and Popova et al. [28] who focused upon the Chelyabinsk airburst on 15 February 2013. Although their research provided information relative to the radar signatures of the individual meteorites, they furthered the background into how a single meteor enters the atmosphere and penetrates to the surface. Furthermore, all of the works in this section remain strictly within the field of planetary science.

3. Meteor Discovery within Doppler Radar

The fields of radar theory, planetary science, and computer science are pulled together by several individuals working together, in particular M. Fries et al. and Hankey et al., in numerous papers [4,5,29]. They pioneered the technique to search through Doppler weather radar data for meteor signatures, and Hankey provided a comprehensive guide in how to acquire and analyze the data from a computer and database perspective. This technique has been used on many different meteorite entry events by Fries et al., confirming its usefulness as a valid technique for meteor falls in Texas, Ontario, Wisconsin, Illinois, California, and various other locations [12,30–34]. Furthermore, some of their research incorporated data sources in addition to Doppler radar data, like seismic data and extracted information such as the fall mass from the radar signature [7,11,35,36]. Although this body of work is noteworthy and valid, the time spent analyzing the data for such events is significant, even with Hankey's techniques. This thesis attempts to reduce the time and workload spent discovering bolides through automation.

4. Artificial Intelligence and Machine Learning

The final field that needs to be introduced is that of AI and its subset, machine learning. The field itself goes back to the mid-20th century with a very rich history of diverse applications, especially as it grew in the 21st century. However, its application to the field of asteroid and meteor discovery is much smaller and more recent. Using machine learning for image classification has gained prevalence in recent years as supervised learning techniques have become more advanced; this has enabled their application towards astronomy classification problems. Djorgovski et al. used unsupervised clustering algorithms for machine assisted discovery of star clusters and galaxies within the Digital Palomar Observatory Sky Survey [37]. Similarly, Fayyad et al. used decision trees and other algorithms to classify faint sky images as stars and galaxies [38]. Lastly, Galindo et al. used a supervised learning, convolutional neural network to detect meteors within closed circuit television imagery [39]. However, these works focused on classifying imagery via pixel analysis, as compared to Misra et al. who used supervised learning with artificial neural networks to classify asteroids based upon their spectral class [40] or Smeresky who used neural networks and evolutionary techniques to classify asteroids from features derived from imagery [41]. Limited research existed on bolide detection using non-pixel related data until Rumpf et al. used a multi-stage filtering algorithm on time sequenced lightning data from the Geostationary Operational Environmental Satellite (GOES) 16 and 17 satellites [42]. Their research is innovative in both the dataset it uses and the method of analysis, but also inherently different from the technique and dataset in this research.

5. Uniqueness by Combining Fields of Study

Each of the previous fields have provided methods to address the meteor and bolide detection process. Furthermore, varying levels of progress and success have been reached in those fields, especially with the research completed by Fries and Hankey et al. The research in this thesis will attempt to follow on from their work in an attempt to address the speed limitation of the bolide detection problem. Therefore, by adding a machine learning perspective to the problem, a multi-disciplinary approach will enhance the already achieved results to use Doppler radar data to detect and identify bolides and meteors.

E. OUTLINE OF REMAINING CHAPTERS

The remaining chapters will provide a further background to the problem, and then go through the process of developing a method for the automated detection of bolides.

More specifically, Chapter II will provide a review of weather radar, including Doppler radar systems and introduce the dataset used in this thesis. Chapter III will provide a foundation of AI, how that differs from machine learning, and how to use machine learning for data analysis. Chapter IV will describe the process of preparing the dataset for the machine learning algorithms. Chapter V will introduce the machine learning algorithms used to detect bolides, and analyze their results and effectiveness. Lastly, Chapter VI will detail future works and the conclusion.

THIS PAGE INTENTIONALLY LEFT BLANK

II. WEATHER RADAR

The U.S. government is responsible for operating a continuous radar network across the continental United States, Hawaii, Alaska, and several overseas locations. The oversight for this network is shared by the Department of Commerce, the National Oceanic and Atmospheric Administration, the Department of Defense, and the Federal Aviation Administration as a subsidiary of the Department of Transportation. The NOAA in particular, within the Department of Commerce, is required to supply climate and meteorological information using systems owned and operated by the Department of Defense and Department of Transportation [43]. To accomplish the above goals, a network of over 160 Weather Surveillance Radar systems was established in the 1970s to replace an earlier system of radars [44]. The name of this radar system was the Weather Surveillance Radar–1988 Doppler (WSR-88D), and had primary functions of acquiring radar data, producing radar products, and displaying results [43].

This chapter provides a basic overview of radar theory and how the WSR-88D radar operates. Furthermore, it explains how the radar information is generated, archived, and downloaded for individual use. Lastly, it covers the data formats of the downloaded data and how they are initially used for visualizing the raw radar information as a precursor step to preprocessing the data.

A. RADAR BACKGROUND

The purpose of this section is not to delve into the depths of electromagnetism or radar theory, but to provide a brief overview that is helpful in understanding the genesis of radar systems and what information is actually produced by a radar system. With that in mind, the acronym of "radar" is attributed to the U.S. Navy in November 1940. It stands for ***ra***dio ***d***etecting ***a***nd ***r***anging, and the technology is a result of the experimental validation of James Clerk Maxwell's thesis on electromagnetism [17]. However, the first radar systems were developed by the British and Germans prior to WWII, with the first radar ranging demonstration completed in 11 December 1924 in London, UK [17].

Although there are multiple types of radar, the process by which radar works is by using an antenna to emit an electromagnetic pulse that spread outward from the transmitter at the speed of light. As the electromagnetic pulse expands, it encounters objects in the path of that pulse [17]. The objects could be particles in the air like dust or rain, objects on the ground like trees or mountains, or objects in the air like birds or aircraft. Upon hitting the surface of the object, the electromagnetic wave is reflected, refracted, or otherwise scattered in all directions.

A small portion of that energy is reflected back towards the origin source of the wave, where it is collected by a radar receiver that is typically co-located with the radar antenna [17,44]. The strength of the return is dependent upon the size, shape, and composition of the object's surface, as well as the distance between the radar's transmitter and the object. The loss due to distance is known as the free space path loss, and is seen mathematically in Equation 3, where P_t is the transmitted power, P_r is the received power, G_t is the gain of the transmitting antenna, R is the distance from radar to target, and A_r is the effective area of the receiving antenna. A_r is further defined in Equation 4 with G_r as the gain of the receiver and λ as the transmission wavelength. The fact that the distance is raised to the fourth power shows an exponential loss in power as the distance increases and this artifact has an effect on many radar applications [17].

$$P_r = \frac{P_t G_t A_r \sigma F^4}{(4\pi)^2 R^4} \tag{3}$$

$$A_r = \frac{G_r \lambda^2}{4\pi} \tag{4}$$

Once the return energy is received, the signal is analyzed using computers. By measuring the elapsed time between sending the pulse and receiving a return pulse, the distance of an object from the radar can be estimated, using $d = vt$ where d is distance, t is time, and v is the velocity of light in air. Similarly, by measuring the frequency of the transmitted pulse, f_t against that of the received frequency f_r, the velocity of the target in radial direction either towards or away from the radar can be determined [45]. This is defined in Equation 5 as the radial velocity, and is the basis behind Doppler Radar systems, such as those used in weather surveillance.

$$f_r = f_t\left(\frac{1+v/c}{1-v/c}\right) \tag{5}$$

1. Weather Doppler Systems

Doppler weather radar systems differ from other radar systems because they are tuned specifically for observing weather phenomenon, just as aircraft surveillance radar are tuned for detection of aircraft. This means that weather radars search for particles of small sizes like water, snow, or ice. Furthermore, because the size of those targets are on the order of 1–10 cm, weather radars use microwave length electromagnetic waves, which have a wavelength roughly one order of magnitude higher than the diameters of target water, snow, etc., so as to maintain the Rayleigh approximation in Equation 6 [17]. To further define the Rayleigh approximation equation, σ_b is the backscatter cross section, λ is the wavelength of the radar, K_m is the complex refraction index of water, and D is the target diameter. By maintaining this approximation, it is possible to calculate the amount of backscatter for the returns gathered by various weather phenomenon. However, if the particles are much smaller than compared to the size of the wavelength, then the radar energy is scattered isotopically due to Rayleigh scattering, as opposed to reflected back to the radar receiver; conversely, if much larger objects or those with a different K_m value are the target, then the amount of backscatter energy is no longer at a maximum [17]. Therefore, tuning the radar λ correctly for a certain sized target is very important.

$$\sigma_b = (\pi^5/\lambda^4)\left|K_m\right|^2 D^6 \tag{6}$$

Multiple pieces of information can be gathered by interpreting the radar returns. The easiest are the pulse information, such as the azimuth/radial angle, the elevation, and the timing of the pulse. However, it is the data retrieved from the reflected energy that provides the most interesting information. The magnitude of the backscattered energy can be used in Equation 7 to determine $\eta(r)$, the reflectivity value or the backscattering cross section per unit volume, where $N(D,\ r)$ is the particle size distribution based on the diameter D and range r, λ is the radar wavelength, K_w is a specific attenuation related to the refractive index of water. However, it is more useful for meteorologists to understand this reflectivity value in meteorological terms using Rayleigh approximations when observing small and spherical targets like rain and snow; therefore, $\eta(r)$ can be simplified into η [17]. To do this, Equation 7 uses the reflectivity factor defined as Z in Equation 8, which is often viewed on the logarithmic scale using Equation 9 and defined as dBZ, the standard unit used for measuring radar reflectivity returns [17].

$$\eta(r) = \int_0^\infty \sigma_b(D)N(D,r)dD \approx \frac{\pi^5}{\lambda^4}|K_w|^2 Z = \eta \tag{7}$$

$$Z = \frac{1}{\Delta V}\sum_i D_i^6 = \int_0^\infty N(D,r)D^6 dD \tag{8}$$

$$dBZ = 10\log_{10} Z \tag{9}$$

The next piece of information that can be gained is velocity spectrum width, normally truncated to spectrum width. This measure describes the density and velocity of particles within the returned radar pulse, and increases in value when the spectrum of return electromagnetic pulse's relative radial velocity broadens [17]. This can be better understood by imagining a volume of space filled with raindrops moving with random velocities and directions; a radar return for that volume will have a high velocity spectrum, indicating a region where the flow of air is chaotic. Conversely, a volume of space with rain falling in the same direction at the same speed will have a low spectrum width, indicating a region where the flow is smooth. Mathematically, Equation 10 shows that the velocity spectrum width is the sum of spectrum widths for shear σ_s, antenna motion σ_α, the speed of different sized particles σ_d, the orientation and vibration of the particles σ_0, and turbulence σ_t. Furthermore, variations in any of these parameters will increase the overall spectrum width. The benefit of determining the spectrum width for a volume of air is gaining a measure of the wind shear or turbulence in that volume, which yields an ability to determine how chaotic the environment is.

$$\sigma_v^2 = \sigma_s^2 + \sigma_\alpha^2 + \sigma_d^2 + \sigma_0^2 + \sigma_t^2 \tag{10}$$

In summary, for each radar sweep, the WSR-88D radar provides beam information and radar return information. The beam information includes temporal and spatial information like the time, azimuth, elevation angle, etc., and also moment information, where radar moments are defined as the product of distance and another parameter, similar to how torque is a force times a distance [46]. The three radar moments are reflectivity, radial velocity, and spectrum width. Each of these pieces of information is parsed out along range gates for easy interpretation of the data. However, there are specifics unique to the WSR-88D radar, which will be discussed next.

2. WSR-88D Radar

The WSR-88D is the primary radar used for weather observations. It transmits a 750,000 Watt pulse from a 28 Foot diameter antenna, and is optimized to detect rain, snow, and other weather phenomenon. At ranges within 37 km from the radar, ground clutter is observable, most levels of precipitation are observable within 148 km, and heavy precipitation is observable within 259 km [44]. The max range for sensing Doppler data and reflectivity data are 300 km and 460 km respectively [43].

The sky is scanned in one of two ways: clear air mode or precipitation mode. Clear air mode is used when no precipitation or limited precipitation is detected; 6 elevations between 0° and 5° are scanned every 10 minutes for returns. Precipitation mode is used during precipitation, including convective events; varying elevations are scanned between 0° and 19.5° every 4–6 minutes, depending on the specific precipitation mode selected [43]. Because of the differences in scan time and elevation, the fidelity of the outputs changes, and the probability of finding a specific return drastically changes between the modes utilized. Furthermore, the scan rates are tuned for tracking slowly progressing phenomenon like rainfall, clouds, etc., and not suddenly appearing targets.

The standard WSR-88D radar is a single polarization system. Because of atmospheric drag due to air resistance, raindrops are wider than they are tall; therefore, the radar uses electromagnetic signals that are horizontally polarized in order to maximize the reflectivity of target raindrops [17]. However, by the end of 2013, all 160 WSR-88D radar sites received a dual polarization upgrade. This upgrade provides the ability to send and receive both horizontally polarized signals and vertically polarized signals, which provided additional data regarding the shape of the particles in the radar return. Where the original WSR-88D radar could only determine that precipitation was in a radar return, dual polarization enabled the ability to identify snow, rain, birds, insects, etc., in return signals [47].

Identifying the types of returns within a signal was made possible by comparing the orthogonally polarized radar signals. The first additional analysis method is completed by analyzing the strength of the horizontally polarized return against that of the vertically polarized return. This comparison yields the overall size and shape of a target, where a ratio of 0 indicates a spherical target, negative values indicate a vertically oriented target, and positive values indicated a horizontally oriented target. This analysis is the known as differential reflectivity, and an estimate of this value is defined in Equation 11, where $\hat{S}_h$ is the estimated sample power from the horizontally polarized return signal, $\hat{S}_v$ is the estimated sample power from the vertically polarized return signal, and the result is on the logarithmic scale [17].

$$\hat{Z}_{DR} = 10\log\left(\frac{\hat{S}_h}{\hat{S}_v}\right) \tag{11}$$

The second additional analysis is the differential phase shift, and is a measure of the phase difference between the horizontal and vertical return signals. This phase difference is not due to movement like when measuring a Doppler shift, but rather through attenuation [48]. Therefore, phase shift can be used to measure precipitation levels, where a low ϕ_{DP} is associated with little to no attenuation due to precipitation and a high value indicates significant attenuation due to precipitation [17]. Differential phase shift is defined in Equation 12, where r is the range, k_h and k_v are the horizontal and vertical increments when hydrometers exist, and ϕ_{DS} is the differential phase after scattering [17].

$$\phi_{DP} = 2\int_{r}^{r_0} [k_h(r) - k_v(r)]dr + \phi_{DS} \tag{12}$$

The final analysis technique from dual polarization is the correlation coefficient between the horizontal and vertical at the zero lag point. The correlation coefficient $\rho_{hv}(T_s)$ is a measure of the consistency in the shapes of the return signals and is derived from reflectivity returns. The correlation coefficient can be estimated as per Equation 13, where $\hat{R}_a$ and $\hat{R}_b$ are the alternate polarization autocorrelation estimates. Values of 1 mean similar shapes, and values less than 1 mean dissimilar shapes [17].

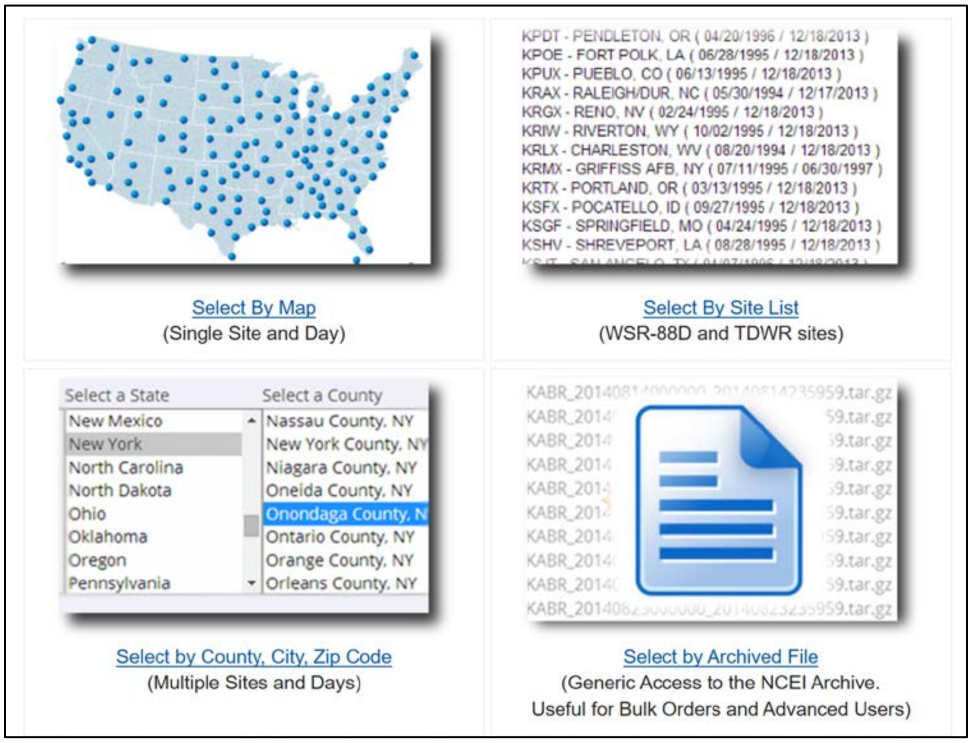
$$\hat{\rho}_{hv}(T_s) = \frac{\left|\hat{R}_a\right| + \left|\hat{R}_b\right|}{2\sqrt{\hat{S}_h \hat{S}_v}} \tag{13}$$

3. Archiving Data

The WSR-88D radar network regularly archives the radar data it produces. There are four levels of archived data: level I, II, III, and IV. Level I data is the lowest level radar data that is produced directly by the radar receiver; it is used for radar site diagnostics and tuning. Level II data is the digital output from the radar's signal processor. It is raw and unanalyzed, but can be a starting point and input for conducting analysis. Level III data is analyzed data, where various algorithms have corrected aspects of the data and generated products that are of interest to consumers. Lastly, level IV data is the output of the radar product generator, and is retained for training purposes, accident investigations, or similar reasons [43]. Both physical copies and electronic copies of the data, especially the level II and III products are retained online for easy access at NOAA's website [49].

B. ACQUIRING RADAR DATA

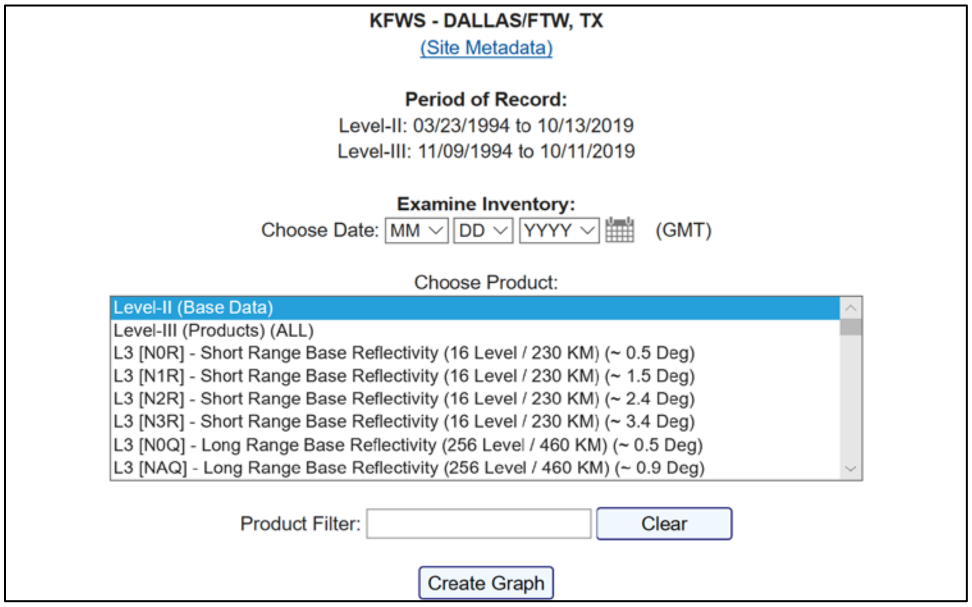
Weather data is acquired from the NOAA online website, found at https://www.ncdc.noaa.gov/data-access/radar-data. The data itself is housed on Amazon Web Services servers, and accessible for download via a selection process on NOAA's website [49]. The selection criteria involve the location, date, time, and type of data requested. The input can be selected via a map of the United States, inputting the radar identifier, or picking from a list of radar sites, as shown in Figure 2. The date criteria selection is made by inputting the date via text or clicking on a calendar, and the data type selection is made by clicking on level II base data or various level III products, depicted in Figure 3 [49]. Lastly, the time is selected by determining a start and stop time for the data period, from which the radar mode for that period of time can be observed. The data is then retrieved via an emailed link or direct download method, as seen in Figure 4 [49]. Once selected and downloaded, a zipped archive or individual .txt files can be retrieved for further analysis.


Select By Map
(Single Site and Day)
Select By Site List
(WSR-88D and TDWR sites)
Select a State
Select a County
Select by County, City, Zip Code
(Multiple Sites and Days)
Select by Archived File
(Generic Access to the NCEI Archive.
Useful for Bulk Orders and Advanced Users)

KPDT - PENDLETON, OR (04/20/1996 / 12/18/2013)
KPOE - FORT POLK, LA (06/28/1995 / 12/18/2013)
KPUX - PUEBLO, CO (06/13/1995 / 12/18/2013)
KRAX - RALEIGH/DUR, NC (05/30/1994 / 12/17/2013)
KRGX - RENO, NV (02/24/1995 / 12/18/2013)
KRIW - RIVERTON, WY (10/02/1995 / 12/18/2013)
KRLX - CHARLESTON, WV (08/20/1994 / 12/18/2013)
KRMX - GRIFFISS AFB, NY (07/11/1995 / 06/30/1997)
KRTX - PORTLAND, OR (03/13/1995 / 12/18/2013)
KSFX - POCATELLO, ID (09/27/1995 / 12/18/2013)
KSGF - SPRINGFIELD, MO (04/24/1995 / 12/18/2013)
KSHV - SHREVEPORT, LA (08/28/1995 / 12/18/2013)

Select a State	Select a County
New Mexico	Nassau County, NY
New York	New York County, NY
North Carolina	Niagara County, NY
North Dakota	Oneida County, NY
Ohio	Onondaga County, N
Oklahoma	Ontario County, NY
Oregon	Orange County, NY
Pennsylvania	Orleans County, NY

Figure 2. Options for using NOAA’s website for downloading radar data. Source: [49].


KFWS - DALLAS/FTW, TX
(Site Metadata)

Period of Record:
Level-II: 03/23/1994 to 10/13/2019
Level-III: 11/09/1994 to 10/11/2019

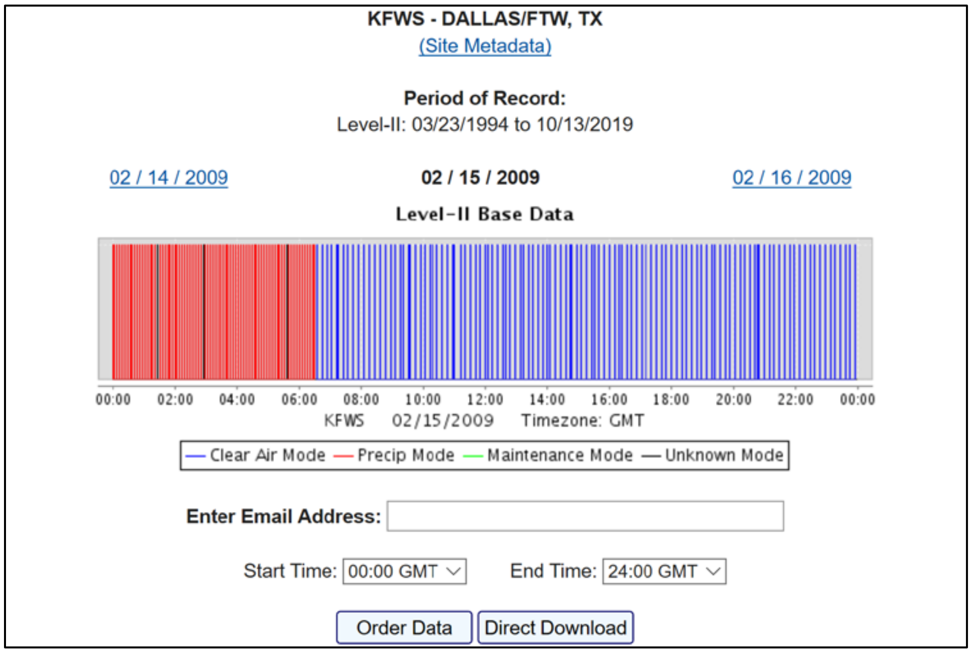
Examine Inventory:
Choose Date: MM DD YYYY (GMT)

Choose Product:
Level-II (Base Data)
Level-III (Products) (ALL)
L3 [N0R] - Short Range Base Reflectivity (16 Level / 230 KM) (~ 0.5 Deg)
L3 [N1R] - Short Range Base Reflectivity (16 Level / 230 KM) (~ 1.5 Deg)
L3 [N2R] - Short Range Base Reflectivity (16 Level / 230 KM) (~ 2.4 Deg)
L3 [N3R] - Short Range Base Reflectivity (16 Level / 230 KM) (~ 3.4 Deg)
L3 [N0Q] - Long Range Base Reflectivity (256 Level / 460 KM) (~ 0.5 Deg)
L3 [NAQ] - Long Range Base Reflectivity (256 Level / 460 KM) (~ 0.9 Deg)

Product Filter: Clear

Create Graph

Figure 3. Downloading level II radar data for the KFWS radar site. Source: [49].


KFWS - DALLAS/FTW, TX
(Site Metadata)

Period of Record:
Level-II: 03/23/1994 to 10/13/2019

02 / 14 / 2009 **02 / 15 / 2009** 02 / 16 / 2009

Level-II Base Data

Enter Email Address:

Start Time: 00:00 GMT End Time: 24:00 GMT

Order Data Direct Download

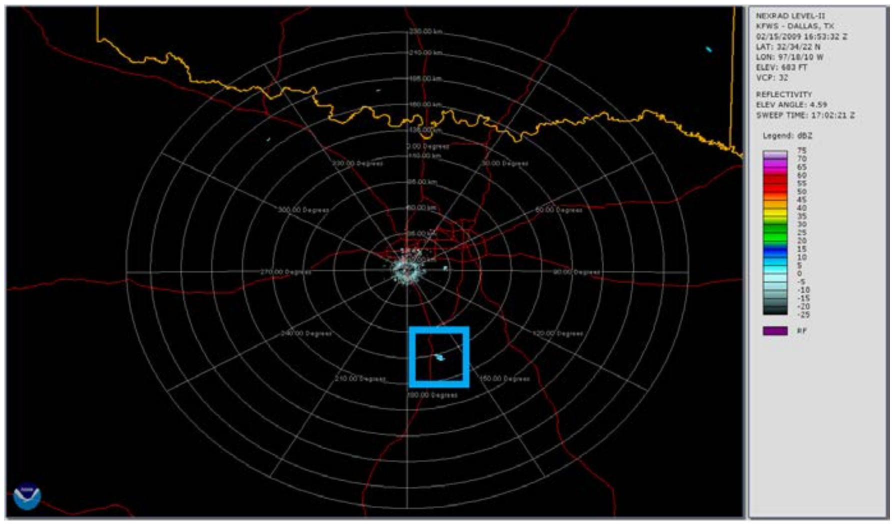
Figure 4. Downloading radar data for the KFWS radar site on 15 February 2009. Source: [49].

C. USING NOAA’S WCT PROGRAM TO VIEW RADAR FILES

The downloaded .txt files are labeled with the Doppler site, date, and time in the filename. They can be opened via notepad, or similar Word file readers; however, they are written in a coded format and therefore unintelligible using ASCII characters. In order to properly open and decipher the data within the file, the NOAA Weather and Climate Toolkit (WCT) is required.

NOAA’s WCT application is a comprehensive weather application for importing, viewing, analyzing weather files, and exporting data. Figure 5 depicts a level II radar reflectivity moment output for 15 February 2009, centered on the Dallas, Texas, radar site KFWS. Range and radial based location information is easily discerned from the application, and the legend on the right side of the image provides scaling data for determining the strength of the radar returns found within the plotted dataset. The far right column also provides radar site information, including date, time, location, and radar sweep angle and elevation information. Not depicted are the moment, elevation, and altitude parameters that can be selected for a specific dataset. Other functions exist within this application to view the data, but are beyond the scope needed for this research.

The last functionality that requires explanation is the ability to export the data. The WCT program can export data into 14 different file formats, including native network common data form (netCDF), comma separated value (CSV), and shapefile (polygon) file types. The process of exporting the data involves firstly selecting the output format and output directory. Secondly, various other optional parameters become enabled depending on the output format, such as radar moment, elevation, and location filters for radials, latitudes, and longitudes. Afterwards, the output file is created in the indicated directory. The applicable formats for this research are the Native netCDF format files with “filename.nc” names, and the CSV files with “filename.CSV” names. Both formats have their advantages and disadvantages; they will be described and compared in the following sections.



The National Oceanic and Atmospheric Administration's Weather and Climate Toolkit program output for the Ash Creek bolide at 16:53:32 on 15 February 2009. The bolide is visible in a blue square at approximately 162° and 189 km relative to the KFWS radar site.

Figure 5. NEXRAD level II Doppler radar reflectivity moment data from the Dallas, Texas, KFWS radar site.

D. THE netCDF UNIDATA FORMAT

The native netCDF format is as close to a representation of the raw data as possible, providing nearly unprocessed data from the radar sites. When selected as an export option within NOAA's WCT application, all available data for each radar moment and elevation is assembled into one output file. This information can then be accessed via Python's netCDF4 library, which imports the "filename.nc" files into a dataset file with a Python dictionary structure [50]. This structure enables the saving of a lot of information in a small, but organized package. The major components of the dataset are the directory of information contained in the dataset, the attributes and history of the dataset, and the variables contained within the dataset.

The dataset directory is the list of accessible methods and attributes that can be used with the dataset. It provides both the private and public Python functions that enable the accessing of data. Examples would be the "file_format" attribute that returns that the dataset is written in "NETCDF3_CLASSIC" format and that "dir()" is a private function that returns this list of methods and attributes. The end result is a beneficial list of ways to manipulate the dataset.

The attributes and history of the dataset are provided via the “ncattrs()” attribute. This attribute provides identifier information for the dataset itself. For example, the Doppler radar station’s identifier and name, the location in latitude and longitude coordinates, the coverage start time and stop time, the level (I, II, or III) of radar data provided, and a summary of the data collected, among other attributes.

The variables contained within the dataset are the most important piece of information, because they provide the access names to retrieve the raw data. Depending on the specific capabilities of the Doppler radar site, different variables will be present, resulting from both high fidelity and low fidelity scans. These in turn are each broken into two general categories: radar moments and spatial location information. The former provides reflectivity information in either decibels or DBZ, the latter provides spectral width and radial velocity information in knots, and all three are the radar returns that the radar sites read from various targets [43]. The latter are the azimuth, elevation, distance, time, radial, and distance gate information needed to determine where each radar return comes from. Additionally, the azimuth and radial values exist between 0 and 360°, the distance and distance gates are ranges from the radar’s center, the elevation is an angle relative to the ground, and the time is the timing of each ray. To summarize the data variables, the first provides the strength of the radar return, and the second provides its location in space and time.

An example of the netCDF dataset with variable dimensions are found in Table 1. The data is as found in the netCDF database, except for two modifications. The first is formatting, for readability. The second is that only the lower fidelity variables are returned. There is an equivalent set of variables appended with “_HI” on to each variable, denoting that there is a higher fidelity output for data return and dimensionality identifier. One important observation from the variables is that the lower and higher fidelity variables are discretized differently, resulting in different dataset sizes. Therefore, it is not possible to natively correlate the data from one dataset to another and one moment to another; a conversion or interpolation method is required in order to do so. This correlation process has several negative consequences that will be explained later.

Table 1. netCDF Dataset for KFWS in Dallas/Fort Worth, Texas, on 15 February 2009

Attribute	**Value**
Station	KFWS
StationName	Dallas/Fort Worth, TX, US
StationLatitude	32.57305556
StationLongitude	-97.30305556
time_coverage_start	2009-02-15T16:53:32Z
time_coverage_end	2009-02-15T17:03:04Z
Summary	Weather Surveillance Radar-1988 Doppler (WSR-88D) level II data are the three meteorological base data quantities: reflectivity, mean radial velocity, and spectrum width.

Variable	**Dimensions**	**Description**
reflectivity	(4, 720, 1832)	reflectivity
timeR	(4, 720)	time of each ray
elevationR	(4, 720)	elevation angle in degrees: 0 = parallel to pedestal base, 90 = perpendicular
azimuthR	(4, 720)	azimuth angle in degrees: 0 = true north, 90 = east
distanceR	(1832,)	radial distance to start of gate
numRadialsR	(4,)	number of valid radials in this scan
numGatesR	(4,)	number of valid gates in this scan
RadialVelocity	(2, 720, 1192)	Radial Velocity
timeV	(2, 720)	time of each ray
elevationV	(2, 720)	elevation angle in degrees: 0 = parallel to pedestal base, 90 = perpendicular
azimuthV	(2, 720)	azimuth angle in degrees: 0 = true north, 90 = east
distanceV	(1192,)	radial distance to start of gate
numRadialsV	(2,)	number of valid radials in this scan
numGatesV	(2,)	number of valid gates in this scan
SpectrumWidth	(2, 720, 1192)	Radial Spectrum

The raw data can be plotted for analysis, even if not correlated. However, before doing so, some processing is necessary: the data is assigned to a NumPy array, is aligned so that the smallest radial value from either the azimuthR or azimuthV, is correlated to 0° as North, and the decision to use azimuthR versus azimuthV is determined from the radar moment type being processed.

The raw radar data has a higher level of fidelity than the processed data. However, this fidelity comes at the cost of additional processing needed to manipulate and view the data. For example, because each radar site is unique and has a different starting azimuth, different radar sites are not aligned with one another and require subtle data modifications to align the azimuths between locations. Furthermore, even this process is not perfect. Comparing the location of the bolide within the netCDF provides Figure 6 and the truth data from the WCT program in Figure 5 yield an azimuthal difference that is unique for each radar site. The cause is unknown, and may be partially explained by magnetic variation across the different radar sites, but even this does not account for the full deviation compared against the truth data. The second processing cost comes from correlating the sizes of the variable arrays. Because the variables have arrays of different dimensions, they are not correlated to the other variables and they are not correlated to positions over the ground via latitudes and longitudes. Two methods to address this could be through either interpolation and sampling or pooling of data points; however, both methods would result in a decrease in accuracy and introduce error.

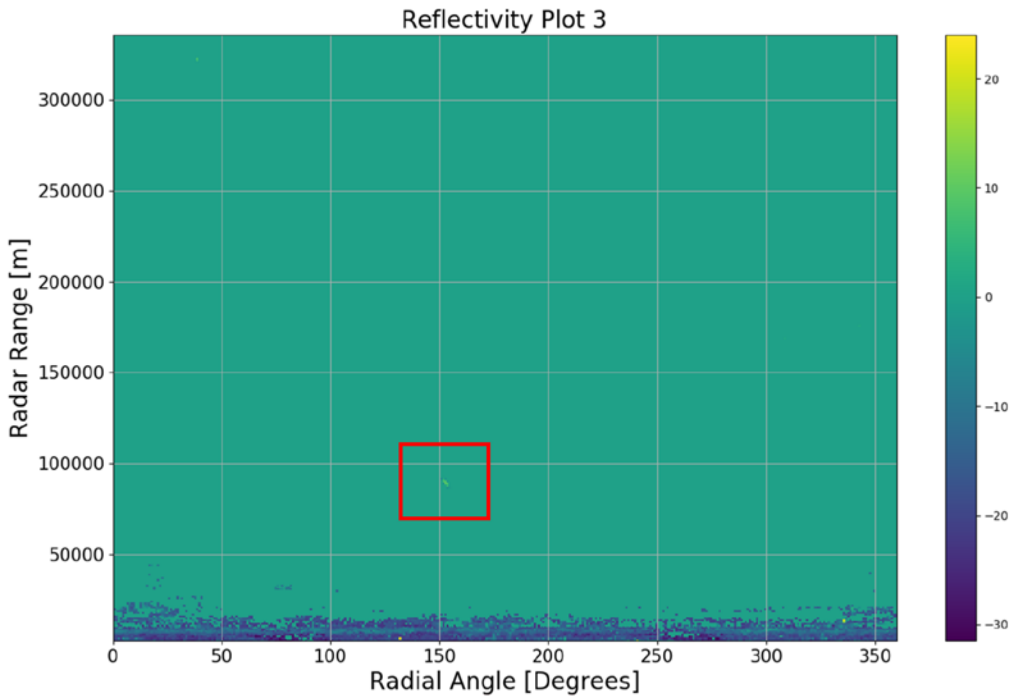


Figure 6. KFWS netCDF data from Table 1, and a bolide boxed in red

One last defining characteristic of the netCDF dataset is that each data point within the dataset represented through Figure 6 has an associated value. Each value is a floating point value ranging in the vicinity of -50 dB to +75 dB. However, depending on the radar moment being analyzed, some values may exist outside that range. The important factor is that the netCDF dataset exports all data from the raw data and provides a return value even if no radar moment return was observed, usually a value at the noise floor. In other words, no missing values exist within that dataset.

E. THE CSV FORMAT

Although the raw netCDF dataset provides a great deal of information within it, the CSV format provides a valuable alternative file format. Unlike the netCDF option, each elevation and radar moment needs to be specifically selected for export. Additional scope parameters are available to constrain the area of data selected for export, but are not enabled for this research because the full radar picture is needed for analysis. After exporting, a single CSV file is produced containing single elevation's moment data. This output is compared against the single output created by the netCDF file format export.

The CSV data file contains multiple rows and columns of data. One column exists for each of 10 features. Rows exists for azimuth and range combination, but unlike the netCDF format, only azimuth and range combinations where the radar moment is a non-zero value are retained, and can be observed by comparing the 0 valued returns in Figure 6 to the corresponding white space in Figure 7, where the white space denotes a complete lack of a return value within the dataset. This reduces the number of rows required to represent the dataset and reducing the overall file size. The dataset has another large advantage over the raw netCDF data: each azimuth and range combination is automatically correlated to its representative latitude and longitude location over the Earth.

The method of identifying each row's placement on the Earth is with a 4-point grid. Each of the grid points is defined by a latitude and longitude pair. Unfortunately, this requires special processing to interpret because the Pandas library within Python that imports the data into a dataframe or matrix of values does not process parentheses correctly. Furthermore, all four latitude and longitude pairs are within the same column of data. Because the column requires processing to begin with, the code necessary to separate them is a trivial addition. The end result yields 14 columns, identified as follows:

Table 2. CSV data format with attribute number and names.

Attribute Number	**Attribute Name**
1	Sweep Number
2	Sweep Time
3	Elevation Angle
4	Value
5	Radial Angle
6	Beginning Gate Range
7	End Gate Range
8	Relative Height
9	Height Above Sea Level
10	Geometry Point 1
11	Geometry Point 2
12	Geometry Point 3
13	Geometry Point 4
14	Geometry Point 5 (a repeat of Geometry Point 1)

With this information in Pandas, it is possible to create a plot similar to that of Figure 6, but created with processed information from the 5) Radial Angle, 6) Beginning Gate Range, and 4) Value variables. Therefore, Figure 7 is the generated heat map scatterplot, with only non-zero moments plotted. Furthermore, the CSV data has the advantage of being properly correlated over the ground and requires no data modification to rotate and align the data to 0° North.

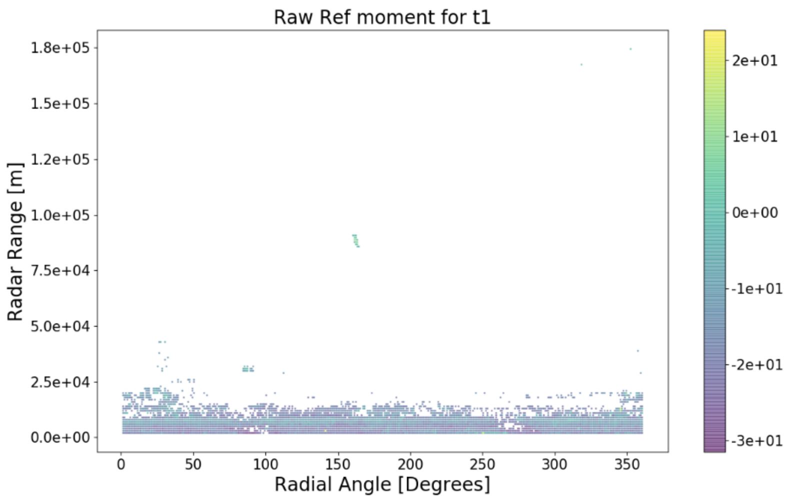


Figure 7. KFWS reflectivity plot using CSV data

Only two negatives exist with using the CSV data instead of the netCDF data. The first is the extra steps to export the data from within NOAA's WCT program. This is an inconvenience that can be rectified using scripting to automate the exporting process. The second is accuracy lost by processing the raw data into a CSV format. One example is the elevation angle for the data represented in Figure 7 is not exactly 4.59° over all 360 azimuth angles, but rather its mean elevation angle is 4.59°. This degradation is a result of an internal process within NOAA's WCT program, but is of minimal consequence. Therefore, the advantages of the CSV dataset outweigh that of the netCDF dataset, and this research will primarily use the CSV dataset.

F. SUMMARY

In this chapter, a basic introduction into the history and theory of radar was provided. In particular, the United States Doppler radar network was explained, and how radar data is collected, retained, and retrieved from NOAA repositories. The data may be downloaded into two formats, each of which were compared. The netCDF format provided a higher level of potential accuracy, but at a cost of alignment errors and additional processing steps compared to that of the CSV format. Therefore, the CSV format was determined to be functionally superior to the netCDF format for this thesis. The CSV datasets are chosen moving forward.

THIS PAGE INTENTIONALLY LEFT BLANK

III. ARTIFICIAL INTELLIGENCE

The field of AI has grown tremendously in the last decade, rising to public awareness in nearly all aspects of society. However, the field is much older and has a much richer history. This chapter provides a background framework from a historical and technical perspective that is built upon later in this thesis. More specifically, it briefly covers the history of AI and machine learning, machine learning methods, search methods, evaluating and comparing hypotheses, and lastly how to use algorithms for data analysis. This chapter is a prerequisite for the application of machine learning techniques used in the data analysis and bolide classification problem.

A. HISTORICAL PERSPECTIVE

The field of machine learning and more generally AI is accepted as starting in 1943 with Warren McCulloch and Walter Pitts [51]. Their work was inspired by biological neurons, based on their backgrounds in neuroscience, physiology, and computational logic. Through this, their research developed a model for artificial neurons and artificial neural networks, now known as Hebbian learning [19]. However, it was not until 1950 that the theory was demonstrated at Harvard University by Marvin Minsky and Dean Edmonds. The two undergraduate students built SNARC, a network of 40 neurons [19]. From this beginning, the field generated a lot of initial praise and excitement.

After the initial successes that defined the field, growth continued until the expectations exceeded the pace of development. This resulted in a pullback in both interest and more importantly funding for AI in the 1960s and 1970s. However, this slowdown did not stop all progress, and the initial development of the back propagation algorithm by Henry Kelley in 1960 helped lay the groundwork for implementing the theory into functional neural networks by Rumelhart, Hinton, and Williams in 1986 [52,53]. This temporary respite gave way to another lull in interest and funding until the 2000s, when two phenomena built upon each other to sustain the current development hype the world currently enjoys. The first was the rise of large datasets, coinciding with the spread of the internet and interconnected machines. As increasingly large amounts of data were collected and aggregated, an information rich diet was found to feed into the various algorithms that drove artificial intelligences. The second was the method to decrease the algorithm runtimes as the datasets grew larger: applying Graphical Processing Units to non-graphics based tasks. By increasing the hardware throughput, more calculations could be completed per second, enabling the use of larger and larger dataset sizes [19]. Due to the continued growth of both datasets and computational power, the current development era with regards to AI is likely to continue to develop and grow.

B. MACHINE LEARNING

The field of machine learning is a set of methods used by AI, providing an agent with the ability to modify its behavior based on environmental influences. The behavioral change is the result of taking in environmental information, performing an action based on an algorithm, comparing against another known action or actions, and updating the algorithm needed to perform future actions [20]. This can be thought of in how an agent is acting in an environment. The agent takes in information via sensors, calculates an action based on an algorithm in the performance element, and affects the environment. The learning occurs through the critic, learning element, and problem generator loop. The critic evaluates agent performance through a set of specific parameters. Then, the learning element uses that evaluation in conjunction with knowledge from the performance element on what external actions to take to update the system's learning goals. Next, the problem generator uses the updated learning goals to seek out new problems that will further enhance the agent's knowledge of its environment; it is the driver to continue searching the hypothesis space. Lastly, the problem generator's output is fed back to the performance element for updating the agent's algorithm [19].

Although all machine learning algorithms have this general methodology, the implementations vastly differ. Machine learning methods are typically partitioned into three types: supervised learning, unsupervised learning, and reinforcement learning. Each has different motivations and operates best over different problem sets, and some datasets are not suited to certain types of machine learning at all. The supervised and unsupervised learning methods are briefly discussed in this thesis.

1. Supervised Learning

The goal of supervised learning is to take a supervised dataset that contains input values, X, and corresponding true output values or targets, Y, and learn to predict the outputs via a trained model, $f(X)$, such that the predicted $\hat{Y}$ values are as close to Y as possible. By looking at the input values, determining an output through a model, and comparing that against the true output provides a mechanism for the machine learning algorithm to correct itself and improve. Another way to look at this implementation is through the lens of a student and teacher construct. As a student learns a subject through examples with known answers, the teacher will affirm correct answers and correct the wrong answers, guiding the student to learn the general logic or pattern to solve a specific problem type [20]. An example of this would be to have a student build the largest tower of blocks under the tutelage of a structural engineer. The student could try various methods, but be corrected through the guidance of the teacher to create a stable and tall tower. This form of learning can be very efficient, but requires a dataset with both input data and output targets. Supervised learning models with numeric outputs are often called regression models. When the output is categorical, such as in this thesis, these models are referred to as classification models.

2. Unsupervised Learning

Conversely, in unsupervised learning, there is no correct answer to accompany the initial data, nor is there a teacher [20]. In other words, unsupervised datasets contain input values X, but no true output values Y. The goal of unsupervised learning is to find relationships between pieces of data within the dataset. An example would be to measure how similar or dissimilar data points are within the dataset, and then using these measures to group data points into similar clusters of data. Using the previous block building example, this time the student has instructions to build the tallest tower out of various blocks, but no further guidance. The student may try various methods to build a strong and tall tower, but has no correct outcome to compare their achieved height against. They may be able to do so against other students that have the same task, or possibly against previous attempts by themselves if that information exists. However, in a worst case scenario, the student may not even know what a tower is or what "tallest" means, and resorts to just putting blocks together in various ways. In this extreme example, the student may have built 100 block structures and upon reviewing them all, can draw conclusions on what they look like, showing that some of the blocks are taller than others, or smaller, or maybe find different relationships like sorting by bock color. Because of the various ways the block structures can be built and clustered, unsupervised learning can be slower or harder to develop, but without guidance or bias, interesting patterns can be developed that would not otherwise be identified.

C. EVALUATING AND COMPARING MODELS

In order to find the best model for a given set of inputs, it becomes necessary to evaluate and compare them. This can be completed using statistical measures of accuracy and when it is available, comparing the true output value Y to the predicted $\hat{Y}$ value. This is an important aspect to machine learning because it enables some learning algorithms and helps prevent overfitting.

1. Overfitting

The process of learning is a spectrum, where an algorithm typically begins with a complete lack of knowledge. When an algorithm begins learning from a dataset, it will try to build a model or hypothesis that best represents the input to output mapping. As this model is refined through the addition and subtraction of terms or parameters that describe the model, it can become too specific [19,54]. This means that instead of learning the general patterns in the data, the algorithm has started to memorize the input to output patterns; this development signifies a shift from generalization to overfitting [55]. To use a curve fitting example with a linear relationship between X and Y as seen in Figure 8, where X is between 0 to 10 and observed outputs are noisy versions of $f(X)$, a generalized predictive model would find the general pattern of the data where $\hat{Y} = X$. This is shown with the order 1 polynomial. However, an overfit model, such as the order 18 polynomial has memorized the dataset and found a model with too many parameters that will not generalize well for any other dataset. In machine learning applications, an overfit model is too specialized for general use and will perform poorly on any dataset other than the one it is specifically trained on.

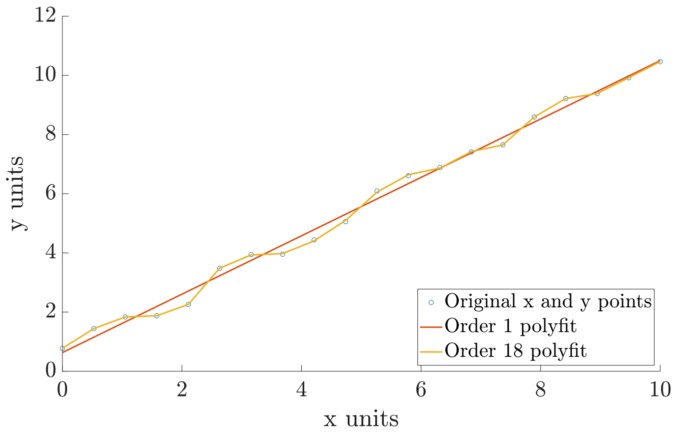


Figure 8. A supervised learning example with noise and overfitting.

2. Training Error and Prediction Error

The accuracy over the dataset used to fit the model versus the accuracy over a dataset of new observations highlights training error versus prediction error. The intent of machine learning is to form a model that performs such that not only the training error is low, but also the prediction error is as low as possible. Unfortunately, it is not possible to determine the prediction error from the training dataset, but it can be estimated by using "hold-out" subsets of the training dataset. These hold-out sets are called validation and test tests. The validation set is used to estimate the prediction error of models based on the training data sans the hold-out set. In this way, the model can be refined so that it achieves the lowest training error rate without overfitting. The test set is used to evaluate the performance of the final model fit. [54].

D. USING ALGORITHMS FOR DATA ANALYSIS

1. The Validity of Machine Learning for Data Analysis

Machine learning provides enhancements and abilities that supersede those of humanity in multiple aspects. Firstly, a machine or series of machines can analyze rows and columns within a dataset far faster and more diligently than homo-sapiens. Secondly, digital methods can hold vastly superior amounts of information within the memory of the computers, whereas humans have only a limited working memory. Thirdly, a machine operates with a level of mathematical precision that is not matched by normal human abilities. As verification, please note what a pocket calculator can do compared to a human. Lastly, automated methods can determine correlations beyond that of what humanity in 2D or 3D can ascertain. Therefore, if digital analysis tools can be applied to a problem, the benefits exist as a motivation to do so.

2. Roadmap for Applying Machine Learning

The process of applying digital methods to a problem set are far more challenging than finding the reasons to do so. The roadmap to build an algorithm is broken into five phases: problem identification, preprocessing, learning, evaluation, and prediction [56].

The first phase is problem identification. In other words, without first identifying what the challenge is, it cannot be solved. This provides an end state to work towards, and is no different than when applying non-machine based problem solving methods.

The next phase is data preprocessing, and this refers to an action performed on the dataset. Depending on the dataset, different actions need to be taken. Datasets can be large, small, sparse, redundant, filled with many variables or features, few features, can be stored in difficult or archaic languages or data structures, and may or may not have associated descriptors defining all variables [56]. For these reasons, preprocessing standardizes the data and turns it into something useable. Therefore, the input to data preprocessing is raw data, and the output is a dataset in the correct format, size, shape, and labels needed to send into the machine learning algorithm [56].

Learning is the phase dedicated to selecting and tuning the machine learning model. Depending on the dataset constructed via the preprocessing phase, supervised, unsupervised, or reinforcement learning algorithms may be selected for use. For example, if output data is not available, then supervised learning is not possible without processing the raw data in some way to generate a supervised dataset. Similarly, some problems are not well adapted for reinforcement learning, for example classification type problems.

Once the model is selected, it is tuned for optimal performance, then executed for validation, and finally metrics gathered for further analysis and optimization [56]. This cycle repeats until the overall performance reaches an acceptable threshold.

The fourth phase is the evaluation phase. Whereas the learning phase iterates through training data to select the best model to represent the data, a second dataset is used to independently test the model: the test dataset. Therefore, after the model is selected and finally tuned, it is given a final test to determine whether it can apply to a new and unique dataset. If it fails to achieve the metrics observed in the learning phase during the test, it could be too specialized or the wrong model entirely. At that point, significant changes may be required to be made to the model or even the type of machine learning algorithm chosen. Conversely, if it passes, the model is ready for operation as a predictor [56].

The final phase is the prediction phase. If the evaluation phase passes with acceptable performance metrics, then new data can be introduced into the machine learning algorithm and the outputs received [56]. The end product may need to be adjusted so that the output format is as needed by follow on processes, but the prediction process remains valid.

E. SUMMARY

This chapter provided a background into AI and machine learning. A brief history of the field is provided, followed by an introduction into machine learning where both supervised and unsupervised learning are explained. The theory behind how to evaluate and compare models is given, and how that relates to overfitting so that algorithmic decisions within this thesis can be understood. Lastly, a high level roadmap for how to apply the theory to an actual problem was provided.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. METHODS

This chapter provides an introduction to the experiments conducted for this thesis. It covers the programming language used, how the dataset was preprocessed, the experimental design, and the statistical background needed for analyzing and comparing machine learning approaches for identifying bolides from within radar returns.

A. CODING LANGUAGE

1. Python

The programming language used in this research is Python [57]. It is a higher level programming language that offers several attributes well suited for research. The language has relatively simple syntax, the language is free to implement, and has a large number of available libraries and modules.

Python's syntax is simplistic and easy to learn, and is well documented in online publications. It is a dynamic language, which makes it straight forward to both code and read. Furthermore, it enables prototyping, decreasing the time, effort, and often the number of lines of code needed for projects. However, the benefit of easier programing comes with a cost: Python has worse performance than other, lower level languages [58]. However, this decrease in performance is an acceptable negative, because once prototyped code is validated, it can be rewritten in a lower level language to increase execution speed if required.

Python is also a free language. The language's software can be downloaded and installed both easily and rapidly. Furthermore, certain operating systems, like Linux, typically come with Python already installed by default. Once the software is installed, it can be further modified under an open source license, and then exported to other users. This has resulted in the proliferation of the software across many users each of which modifies and adds to what Python as a whole can do. As a result, there exists a vast amount of libraries and modules that have been implemented within the language framework [58]. Multiple libraries were installed and used during this research; the primary ones were NumPy, Pandas, Scikit-learn, and matplotlib.

2. NumPy Library

NumPy is a science and engineering library in Python that provides multiple computational tools. This includes tools that enable linear algebraic manipulations, Fourier transforms, generating and using random numbers, and additional features. However, the primary benefit it provides is the capability to manipulate and modify *N*-dimensional arrays of data efficiently and quickly [59]. This library is limited to using and manipulating numbers, but there are other libraries that provide expanded non-numeric object handling.

3. Pandas Library

Where NumPy falls short, Pandas fills in with new capabilities: Pandas enables the handling of non-numerical objects in an object called a dataframe. It can handle numeric, string, time series, etc., forms of data in *N*-dimensional arrays. Pandas is designed to parse and handle data analysis tasks, so indexing, reshaping, and managing missing data are all within its limits [59]. Unfortunately, it is computationally slower than NumPy, but the enhanced data structures are extremely beneficial because real world datasets are rarely only numerical.

4. Scikit-learn Library

NumPy and Pandas provide methods to aggregate, sort, and manipulate data. However, they do not provide methods to analyze or learn from the data. Scikit-learn provides tools to perform those additional actions [59]. The library enables pre-built searching, clustering, and interpolation methods, along with additional classification and regression tools through its estimator interface, where models are instantiated and fit with training data via learning [60].

5. Matplotlib Library

Matplotlib is a plotting library for use with Python [59]. It was originally designed to mimic MATLAB’s features, but has grown since its inception. It includes all the basic plots expected within a GUI, and is not limited to typical line graphs, scatter plots, and heat map plots [61]. Its visualization functions enable easy data visualization.

B. DATA PREPROCESSING

The process of downloading the raw data files from NOAA's website and exporting them via NOAA WCT application as either netCDF or CSV files was described in Chapter II. Furthermore, of the two methods, the CSV format was determined to be superior for multiple reasons, and chosen as the most beneficial output format. Therefore, the CSV format is used as the baseline for further data preprocessing, where the dataset is transformed into something that can be used within machine learning algorithms for further analysis.

The exact steps and methods required to preprocess a dataset is unique to that problem and dataset, but a bolide detection algorithm includes raw dataset creation, dataset characterization, feature selection, processed dataset creation from selected features, and exporting the processed dataset. This research follows this process, and generates an output useable by machine learning algorithms. Furthermore, the first step beings with importing the CSV data into Python.

1. Importing CSV Data into Python

The import process relies upon the Pandas library and a few supporting functions. The required input is a directory path containing the WCT application's exported CSV files. With this path, Python creates a list of CSV files, opens the files, and uses Pandas to read the file contents into a dataframe. This sequential process yields a list of dataframes, one for each of the radar moments and time periods t_0, t_1, and t_2 associated with the original CSV files in the directory. Furthermore, each dataframe is created with radialAng and begGateRan set as the two-layer index; this convention is utilized for the remainder of this thesis. The variable names with their abbreviation are found in Table 2, and the visual output produced via Matplotlib is found in Figure 9. In Figure 9, raw CSV data for time t_1 was exported and plotted to depict the differing dimensionalities of the data. One note to highlight is that the reflectivity moment has 2 data points in the top right that are not present in the radial velocity and spectrum width moments, identified in the 2 yellow boxes.

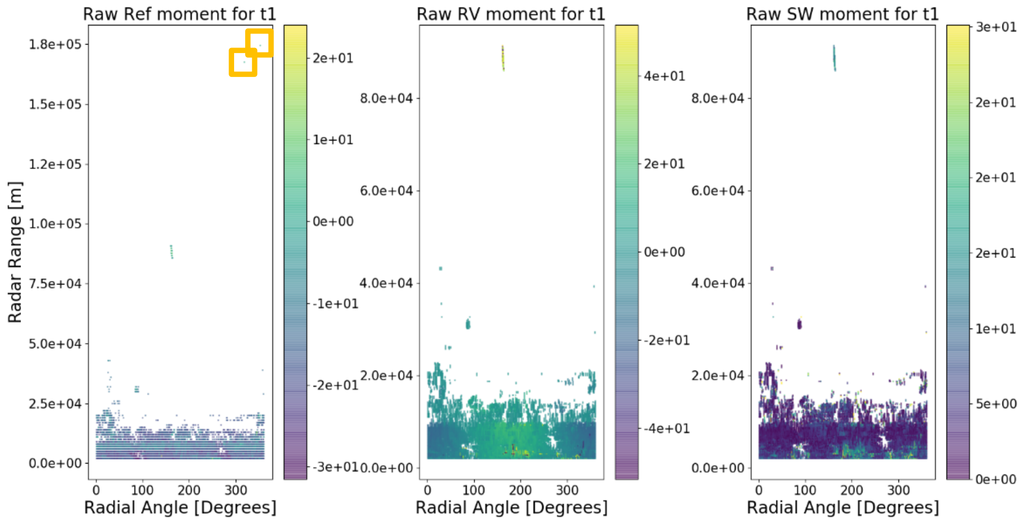


Figure 9. Raw CSV data for the Ash Creek bolide plotted to depict the differing dimensionalities of the data

2. Merging the Radar Moments Together

The process of data aggregation is completed through the database management tools inherent to Pandas. The merge function takes the list of radar moments for each time frame, and conducts an outer join with each data point's radialAng and begGateRan parameters as the enabled left and right index values. The result is a dataframe with a two-layer index and a column for each radar moment with the following columns in the dataframe: radialAng, begGateRan, Ref, RV, and SW.

3. Initial Problems with the CSV Dataset

The primary issue with the CSV dataset is that each radar moment is unique in its dimensionality. This is because after exporting the radar data from the WCT application, only non-zero returns are contained within the dataset. An example of this is depicted in Figure 9, where the radar reflectivity, radial velocity, and spectrum width moments are depicted from left to right respectively. All three plots contain multiple data points at approximately 89000 m and 160°, but only the reflectivity plot contains data points at both 167000 m and 319°, and 174000 m and 353°. Those two data points correspond to rows within the CSV dataset that do not exist within the radial velocity and spectrum width datasets, and therefore the reflectivity moment has a length at least 2 units longer than that of the radial velocity and spectrum width radar moments. The result is that data cannot be easily combined from different radar moments without the creation of missing values or Not a Number (NaN) values. Unfortunately, NaN values sometimes result in errors when a dataset containing them is analyzed with a machine learning algorithm. Therefore, they must be eliminated or otherwise resolved before proceeding.

A second and more minor issue that exists with the dataset is that sometimes the data is coded erroneously by the radar sites. The typical range of a dataset value is roughly -50 to +75, depending on the radar moment, as shown by the legend on the right hand side of Figure 10, the reflectivity returns for the KDAX radar on 22 April 2012. However, occasionally the radar outputs an erroneous return, such as in the purple RF region in Figure 10. The reason for the appearance of this artifact is unknown, but is clearly artificial. Furthermore, it is coded in the CSV dataset with a value of +800, significantly different from the nominal range described above. However, it is because the coding is so far outside the normal range that it is easily detected, and subsequently mitigated by removing all data points with radar moment values greater than 400. Unfortunately, this removal does result in additional missing values, which require further processing to address.

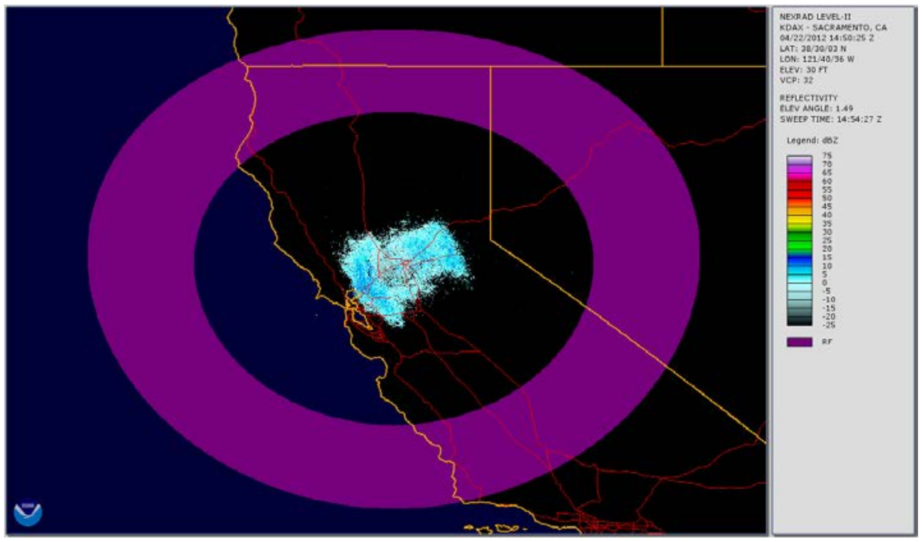


Figure 10. NEXRAD reflectivity plot within the WCT application, depicting an erroneous RF return

4. Replacing Missing/NaN Values

Three methods are attempted to resolve the appearance of missing values within the dataset of raw and merged radar moments. The first is to replace all of the existing data points with interpolated values, the second is to replace only the missing data points with interpolated values, and the third is to replace the missing values with radar moment minimum return values. Although all three methods are viable options, the first two have substantial flaws that inhibited their use.

The first method can be considered full interpolation. The input is the dataset of raw and merged radar moments and grid length L. The output is a dataset of interpolated values aligned to grid size L^2. The interpolation is applied to each radar moment and each time frame for which a raw CSV file exists. The method is through Python's "griddata" function using the linear interpolation parameter; two other interpolation parameter options were investigated, but not chosen due to results that skewed a higher amount of data points. Figure 11 depicts the interpolated grid data for each radar moment with the original raw data points overlaid in black for timeframe t_1 where $L = 200$.

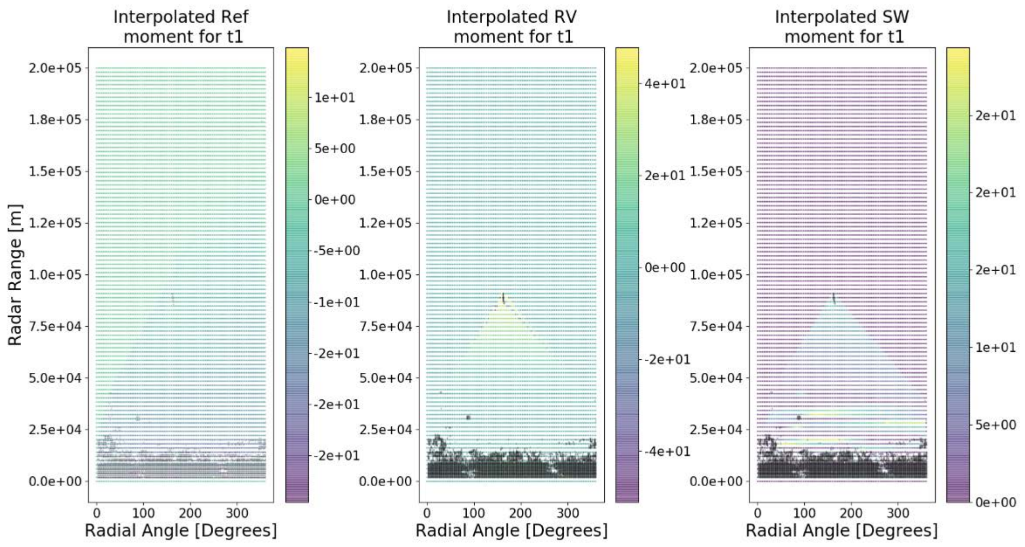


Figure 11. Fully interpolated data for the Ash Creek bolide plotted with raw data points denoted in black

This approach successfully creates an interpolated grid of points using the input data; however, two weaknesses become apparent. The first is that as the L^2 grid density increases, the temporal cost of performing the interpolation increases exponentially. This cascades into substantially longer computational times for follow on analysis, making the cost too high to make a large L feasible. The second weakness is a distortion in the data inherent from replacing the raw data points with interpolated readings. Figure 12 depicts the results of each radar moment plotted against one another. Because the interpolated dataset is interpolated linearly between raw data points, radar returns appear where there should be no valid returns and hold skewed values from what they should be. In Figure 12, these skewed returns can be identified easily as the blue-green, triangular artifact. The conclusion is that the linear interpolation in essence blurs the real data points too much to be useful, and is therefore discarded as a useful method to preprocess the data.

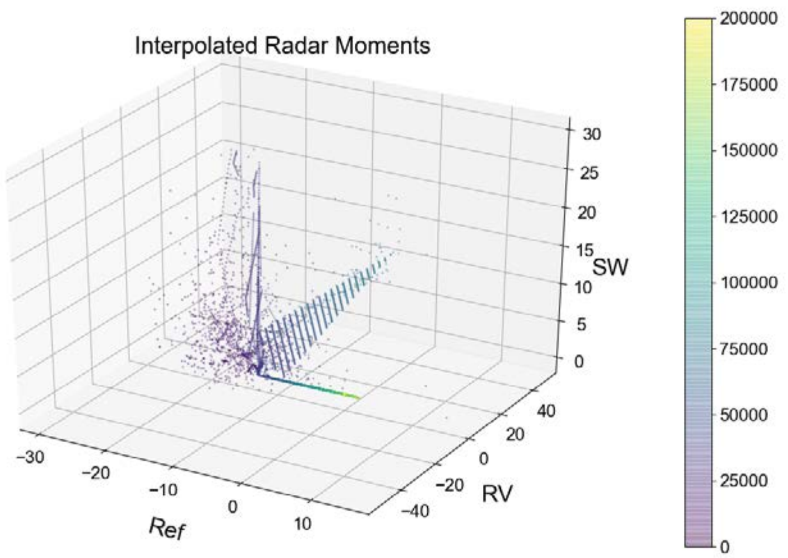


Figure 12. Fully interpolated data, plotted in 3D with radar moments as an axes for the Ash Creek bolide

The second method of resolving the existence of missing values is to only replace the missing values with interpolated values using a Nearest Neighbors (NN) approach, which finds the points closest to a point of interest using Euclidean distance [62]. Proportionally, this method results in a dataset of mainly of raw data points, with a small percentage of missing value points replaced with interpolated values. Three steps are required to execute this change. The first is to create an interpolated dataframe, identical in process to the full interpolation method. The second step is to create a K-D tree to categorize the data points by their radial angle and range. A K-D tree uses the median values of a set of values to iteratively split the data into successive halves and generate a binary tree that in this case partitions data points into groups of two. The third and final step is to use the K-D tree for a NN search using Euclidean distance to find the closest interpolated point within each partition of data points, where the Euclidean distance between any two given points p and q in n dimensional space is defined in Equation 14 [62,63]. This calculation is trivial because each partition only holds 2 data points, resulting in a quick search to identify the closest interpolated point and its associated radar moment data, which is then used to replace the missing value.

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{14}$$

This method of replacing missing values does come with flaws, which result in its dismissal as a viable method to resolve missing values. The K-D tree is a fast algorithm for lower dimensional data, such as this case with only 2 dimensions. However, the tree can sometimes miss the actual closest nearest neighbor because it only looks for the closest point within a single partition of data points. This inaccuracy has a negative effect on the outcome of this approach, but it is not the driving factor behind abandoning the approach. The issue still lies with the interpolated data, which is not accurate enough due to the blurring of the data points. Because the NN method still relies upon the interpolated data, the negatives previously discussed with the interpolated data still apply, although smaller in magnitude because far fewer interpolated points are used. The results of plotting the 3 radar moments against one another, with range information for coloring is depicted in

Figure 13. The triangular interpolation artifact has been removed, but the data points still have more dispersion in the 3D cloud of data point than is helpful in identifying a bolide.

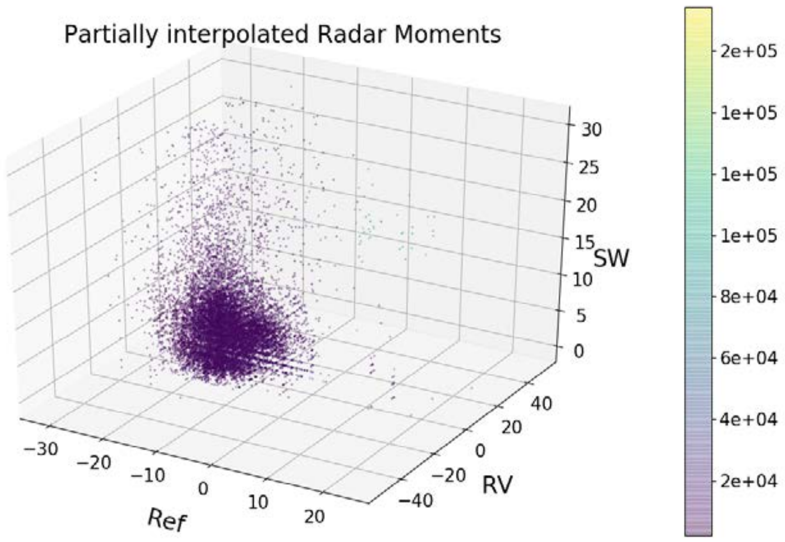


Figure 13. Partially interpolated data, plotted in 3D with radar moments as axes for the Ash Creek bolide

The two previous attempts to use interpolation to replace the missing values resulted in an increase in point cloud blurring and dispersion, necessitating a method that does not rely upon interpolation. Therefore, the third and most useful method to replace the NaNs with acceptable values is completed through a Pandas “fill” command. The rationale is that in most cases, a missing value exists because no return is observed in that region, with a small subset of missing values existing because they were coded as an erroneous radar moment value of +800 and subsequently removed earlier in preprocessing. Therefore, most of the missing values should hold a value equivalent to “no return,” but because the scale of the radar moments is the logarithmic decibels scale, a zero return is not the lowest value, but instead in the center of the distribution of values. Attempts to revert the dataset out of the logarithmic scale dispersed the data too far to be useful, and another method is required to process the missing values.

The alternate solution is instead to replace the missing values not with zero, but with the smallest radar return found elsewhere in that list of radar returns. This is accomplished using the Pandas “fillna()” command, which fills in NaN values with a user value, in this case the minimum value for a radar moment in a given time was used. The output when the three radar moments are plotted against one another is depicted in Figure 14. Comparing it against the fully interpolated dataframe shows a similar image, but looking at Figures 15 and 16 show that replacing interpolated values with minimum values drastically changes the distribution of the data points. In all three radar moments, the distribution is pulled to the left, where there are more minimum values due to the “fillna()” command, achieving the desired effect in the steps after preprocessing.

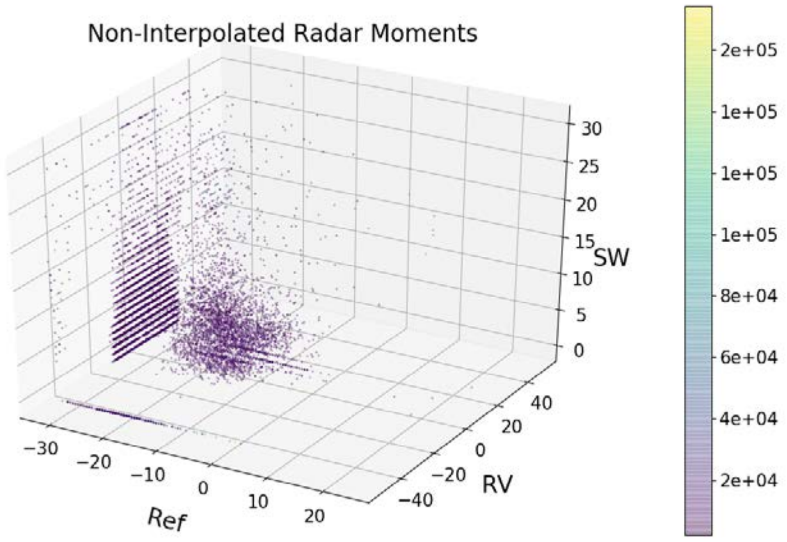


Figure 14. Missing values removed and plotted in 3D with radar moments as axes for the Ash Creek bolide

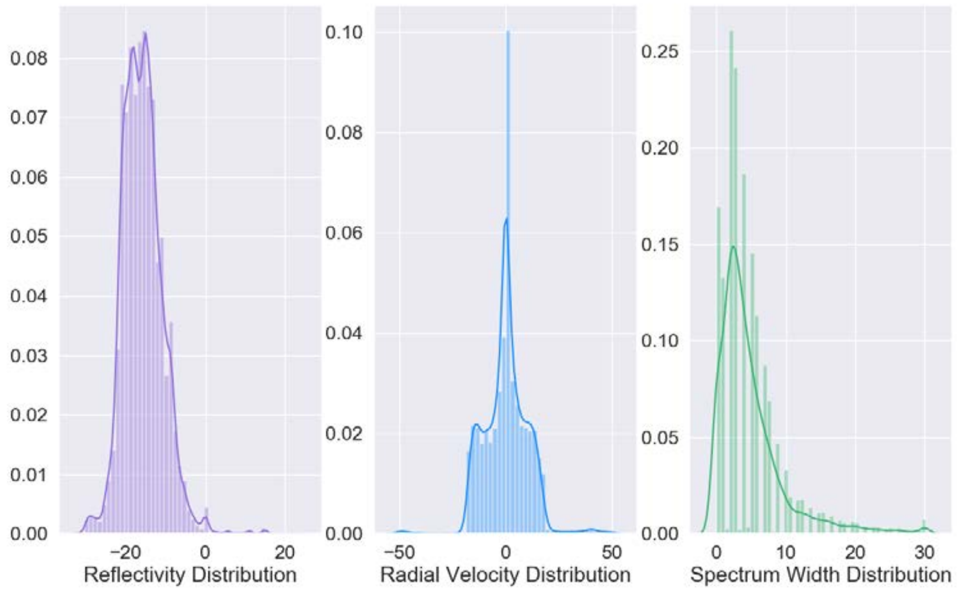


Figure 15. Distribution of the radar moments when interpolation is used to fill in missing values

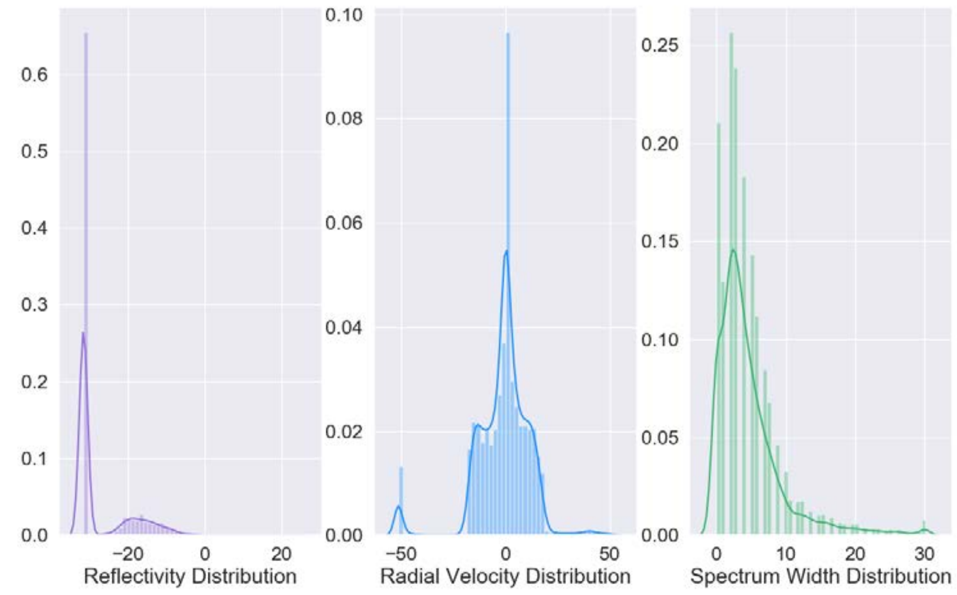


Figure 16. Distribution of radar moments when the missing values are replaced with minimum values

The result of this step of preprocessing is a dataset containing the three radar moment features from the original raw dataset, but without any missing values, which have been replaced with minimum values. The results are depicted in Figure 17, and are a basis for moving forward towards the next step in preprocessing the dataset.

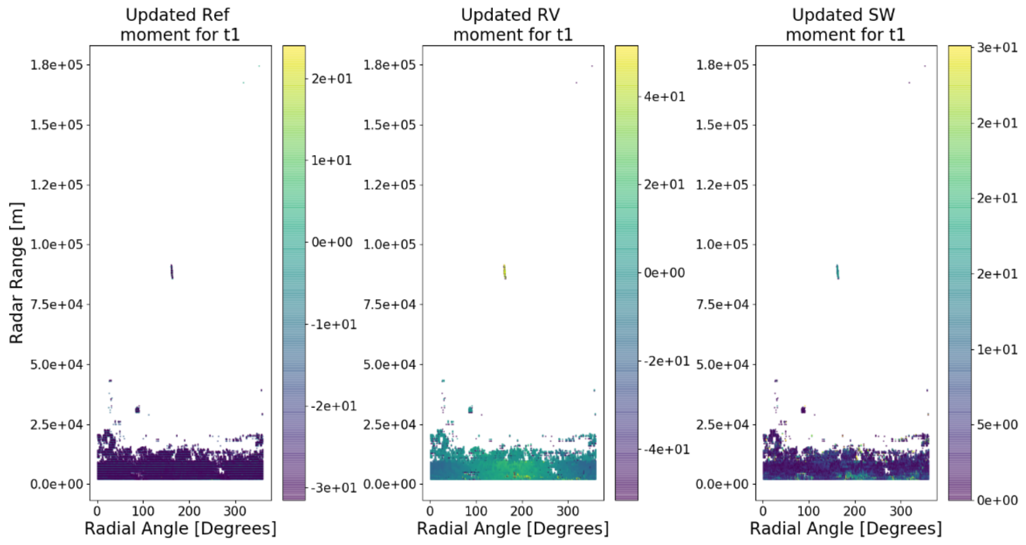


Figure 17. Radar returns for the Ash Creek bolide updated with missing values replaced by minimum values

5. Creating a Range Scaled Feature

The original dataset has three features: reflectivity, radial velocity, and spectrum width. However, because strength of each return is based upon the distance from the radar site, returns from small, but close objects can have the same magnitude return as those from larger, but further away objects. As a trivial example, a bird 10 meters away may have the same return magnitude as an aircraft 100 kilometers away. This is observed in Figure 17, where there are substantially more returns closer in to the radar than there are further away, where the smallest returns at far distances are too small be even be picked up by the radar's receiver.

In order to account for this phenomenon, three new features are created. They are equivalent to the original three features, except scaled by range using Equation 15 where *range* is the distance from the radar site to the radar return. The results are depicted in Figure 18, and show an image similar to that of Figure 17 except certain returns are more pronounced and the color bar scale covers a larger range. This is the desired effect in order to regularize the data based on distance and make more distant objects stand out against closer objects.

$$Value_{scaled} = Value \times Range \tag{15}$$

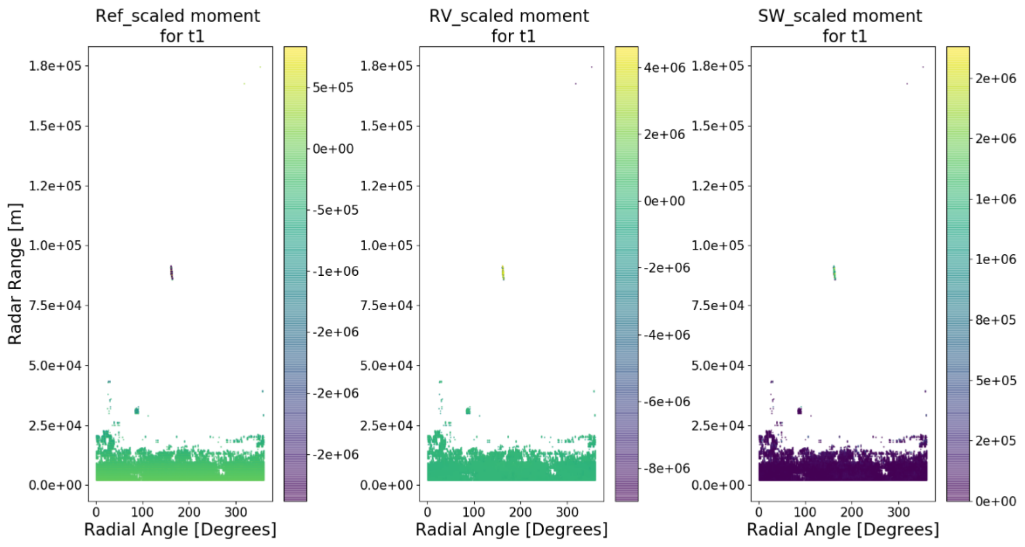


Figure 18. Radar Moment data for the Ash Creek bolide scaled by range and plotted

6. Creating a Change Detection Feature

The original three radar moment features and the range scaled features exist for each time slice: t_0, t_1, and t_2. However, each independent feature lacks any way to link the information from one time slice to another. This is contrary to how a human would identify the appearance of a bolide within a dataset; a human would look at a radar moment for each time slice and ask what changed from t_0 to t_1 and from t_1 to t_2. If an object covering multiple pixels with high reflectivity, radial velocity, and spectrum width values appeared in t_1 from a blank region in t_0, and then disappeared in the t_2 dataset, then that would register as something to investigate further as a potential bolide. Therefore, Equation 16 was developed as a method to simulate this approach of anomaly detection. In short, it identifies if a pixel has a large change from any one time slice to another, under the assumption that the pixel changes from weather are more slow and gradual than from other sources, like bolides. Originally, the difference between the maximum and minimum values was not squared, but after initial analysis, increasing the magnitude of the difference increased the sensitivity of this feature. Therefore, Equation 16 is applied to the t_0, t_1, and t_2 datasets of a single radar moment. The outputs of the process are three datasets with the maximum change for each radar moment. Each one is plotted in Figure 19, and can be analyzed for changes between t_0, t_1, and t_2 for each radar moment.

$$\Delta_{Pixel} = [\max(t_0, t_1, t_2) - \min(t_0, t_1, t_2)]^2 \tag{16}$$

Unfortunately, the issue of erroneously coded data described in Section IV.B.3 appears again, which makes information within Figure 19 less helpful than it could be. To better understand the issue, Figures 20 and 21 depict the WCT program's output for the KRGX NEXRAD site on 22 April 2022. In the reflectivity moment, return values at 230° and 90 m from the radar are depicted within the nominal range; however, in the radial velocity moment, returns at that same radial and distance are coded as RF. The RF return is coded at +800 in the CSV dataset and subsequently removed from the dataset by the preprocessor. A similar effect occurs in the spectrum width radar moment, where the returns are also coded as RF. The consequence is that only the reflectivity moment will register a change, rather than all three radar moments. The implication is that searching for bolides via pixel changes fails in two thirds of the plots because the data is not present, which confuses the overall process.

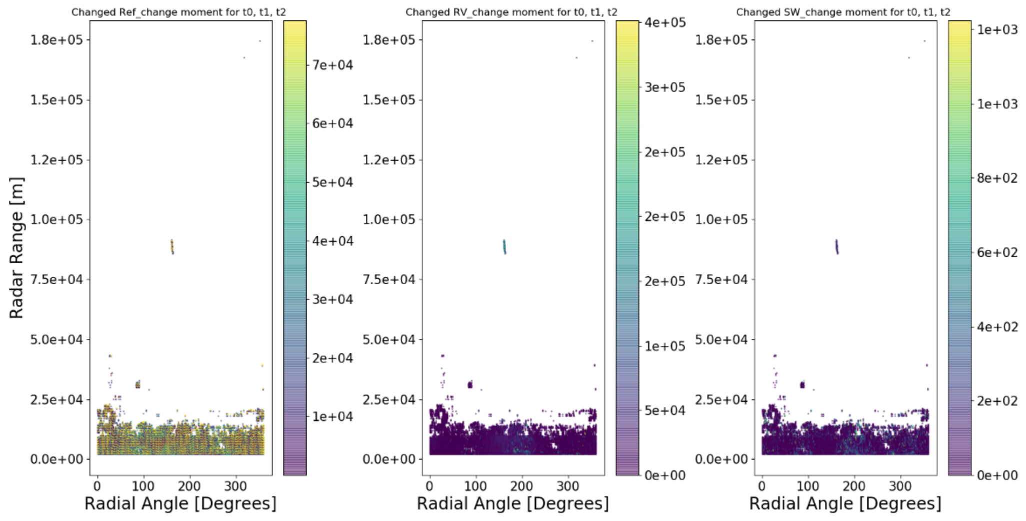
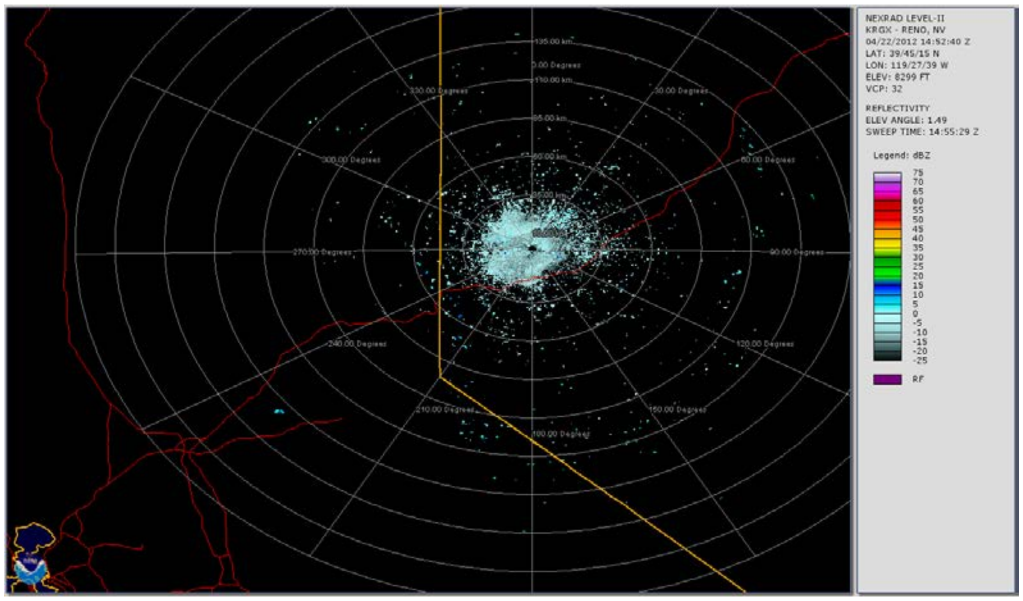


Figure 19. Change detection for the Ash Creek bolide between times t_0, t_1, and t_2


NEXRAD LEVEL-II
KRGX - RENO, NV
04/22/2012 14:52:40 Z
LAT: 39/45/15 N
LON: 119/27/39 W
ELEV: 8299 FT
VCP: 32

REFLECTIVITY
ELEV ANGLE: 1.49
SWEEP TIME: 14:55:29 Z

Figure 20. Reno, Nevada KRGX NEXRAD site on 22 April 2022, displaying reflectivity data with a valid return at 230° and 90 km

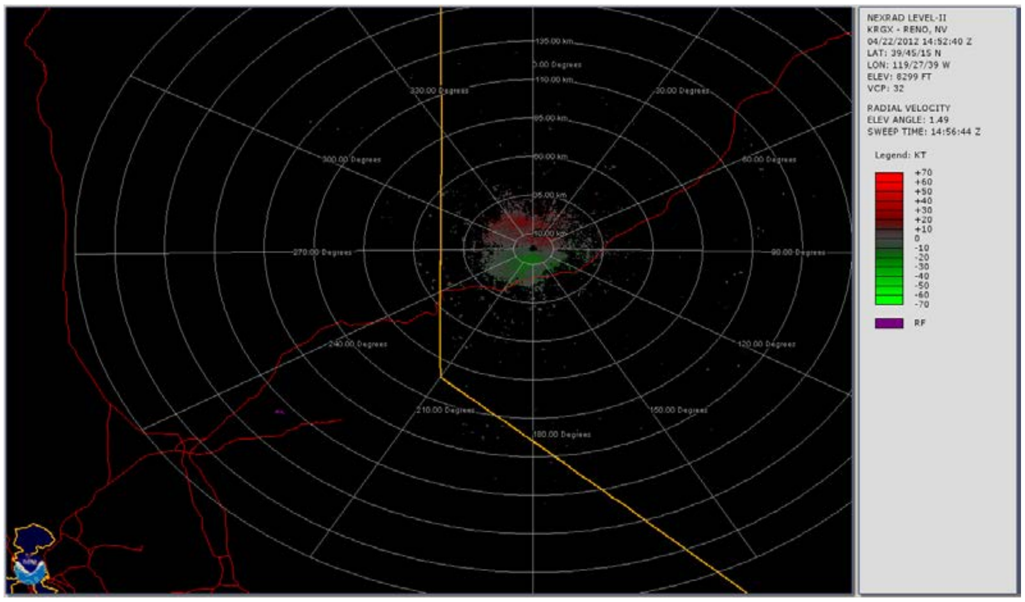


Figure 21. Reno, Nevada KRGX NEXRAD site on 22 April 2022, displaying radial velocity data with an invalid return at 230° and 166 km

A method to overcome the absence of pixel changes in each radar moment is to look for and retain a change in any moment. In this manner, the greatest magnitude pixel change is kept for further analysis in a single dataset, rather than individual datasets for each radar moment. Mathematically, this is described in Equation 17, and results in a single database to identify whether a pixel has changed in any moment and time slice. This provides the time varying change information needed to identify outliers in the dataset. Visualized, Figure 22 is an aggregation of the three subplots in Figure 19, and provide a single, comprehensive plot for detecting change in a dataset.

$$\begin{aligned}\Delta_{Pixel,max} = \max([&\max{}_{Ref}(t_0,t_1,t_2) - \min{}_{Ref}(t_0,t_1,t_2)],\\ &[\max{}_{RV}(t_0,t_1,t_2) - \min{}_{RV}(t_0,t_1,t_2)],\\ &[\max{}_{SW}(t_0,t_1,t_2) - \min{}_{SW}(t_0,t_1,t_2)])\end{aligned} \tag{17}$$

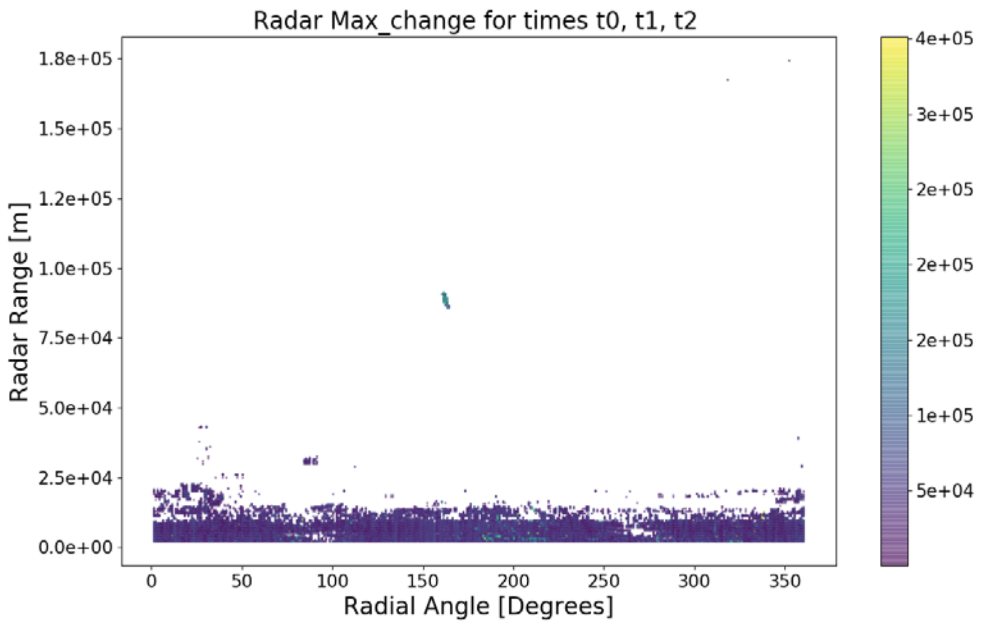


Figure 22. Max change for the Ash Creek Bolide, aggregated from all radar moments and times

7. Dual Polarization Features

The last three features added to the dataset are found on radars after receiving the dual polarization upgrade no later than the end of 2013 [47]. The three additional features are differential reflectivity, correlation coefficient, and differential phase, and are described in Chapter 2. They are not modified in the execution of post-processing code, and treated equivalently to the original three radar moments: reflectivity, radial velocity, and spectrum width. However, in writing the preprocessor's code, multiple checks need to be generated to check for and handle whether a specific dataset had the three additional features or not. This results in the dimensionality of the dataset increasing by potentially 3 features.

8. Dataset Feature Description

The developed preprocessor creates a dataset that contains 12 features. They have been described throughout this chapter, but summarized for convenience in Table 2. This dataset is the output of the preprocessor and contains all the information used in the subsequent analysis steps.

Table 3. Output dataset created by the developed preprocessor

Abbreviation	**Full Name**	**Purpose**
radialAng	Radial Angle	Index information for pixel location
begGateRan	Beginning Gate Range	Index information for pixel location
Ref	Reflectivity	Return value for a radar moment
RV	Radial Velocity	Return value for a radar moment
SW	Spectrum Width	Return value for a radar moment
Ref_scaled	Reflectivity scaled by range	Normalized radar return values
RV_scaled	Radial Velocity scaled by range	Normalized radar return values
SW_scaled	Spectrum Width scaled by range	Normalized radar return values
Ref_change	Reflectivity pixel change	Return change data for t_0, t_1, and t_2
RV_change	Radial Velocity pixel change	Return change data for t_0, t_1, and t_2
SW_change	Spectrum Width pixel change	Return change data for t_0, t_1, and t_2
Max_change	Max change for a pixel	Max change in Ref, RV, and SW for a pixel for t_0, t_1, and t_2
CC	Correlation Coefficient	Dual Polarization return value
DP	Differential Phase	Dual Polarization return value
DR	Differential Reflectivity	Dual Polarization return value

C. EXPERIMENTAL DESIGN AND LIMITATIONS

This research relied upon data from a small sample of bolide events, each of which was verified to contain a bolide as per published documentation [31,36]. In total, each of the 4 datasets is listed in Table 3 with the associated date of the bolide and the NEXRAD radar site. Each of the 4 events contains the requisite raw data that produces the information in Table 2 when inputted into the developed preprocessor.

Table 4. List of datasets with verified bolides

Bolide Location	**Date**	**Radar Site**	**Number of Data Points (after preprocessing)**
Ash Creek, Texas	15February2009	KFWS	15,262
Sutter’s Mill, California	22April2012	KDAX	66,388
Sutter’s Mill, California	22April2012	KRGX	137,332
Sutter’s Mill, California	22April2012	KBBX	121,782

After initial data exploration and data preprocessing, this research focused upon exploring machine learning algorithms to determine the best method to identify outlier bolides from nominal radar returns from weather phenomenon such as clouds, rain, etc. Multiple algorithms were tested for their ability to identify outliers, their parameters tuned for optimal performance, and the results analyzed and discussed.

The approaches tested were selected based on two limitations of the bolide detection problem. The first was the sparseness of the dataset. Not only were there a limited number of verified bolide events captured on radar, written about, and published, but the number of radar returns or pixels within a NEXRAD radar's dataset was extremely small. For instance, the Ash Creek, Texas, bolide was identified with 15 radar returns, while the raw dataset contained 15,263 or 0.098% of the data. Similarly, the Sutter's Mill KRGX NEXRAD site had 137,332 radar returns, with only 23 or 0.017% identifying the bolide. The second factor that made bolide detection difficult was the nature of the data itself. Because the raw data was unsupervised, many machine learning techniques and algorithms were unavailable, as they required truth data as an input. Generating a supervised dataset was possible, but again due to the sparseness of the dataset, unsupervised data was identified as the primary method to characterize and classify potential bolides.

D. PERFORMANCE STATISTICS

Minimal statistical analysis was required in this research. This was primarily due to the aforementioned limitations of the small number of bolide datasets analyzed. However, some statistical analysis was conducted to provide some rigor in the results. This included the computation of a confusion matrix, accuracy, precision, recall, and lastly a pruning rate.

A confusion matrix is a method used to represent the percentage of correctly and incorrectly classified objects. Table 4 depicts a confusion matrix, with the number of correctly classified objects in the green boxes and incorrectly classified objects in the red boxes. The goal is to maximize the occurrence of green, True Negative (TN) and True Positive (TP) values, while minimizing the occurrence of red, False Negative (FN) and False Positive (FP) values. The method to populate the confusion matrix is through summing the number of values classified in each of the categories, based on results from an algorithm. If an algorithm determined that 1 of 10 data points was classified as a bolide, and it was actually a bolide, then the TP value is 1. Similarly, if 5 of 10 data points were correctly classified as not a bolide, then the TN value is 5. However, if 1 of 10 data points was classified as a bolide, but not actually a bolide then the FP value would be 1, and if 3 data points were classified as not bolides, but were actually bolides, then they would be FN [56].

Table 5. Confusion Matrix example and definition

	Predicted Values		
Actual Values		Negative	Positive
	Negative	True Negative (TN)	False Positive (FP)
	Positive	False Negative (FN)	True Positive (TP)

Although the confusion matrix is helpful in visualizing data, it can be used to generate even more helpful statistics. The first performance metric that is helpful to analyze is accuracy (AC), and is defined in Equation 18. The second is recall or the True Positive Rate (TPR), and is defined in Equation 19. The third is the specificity or True Negative Rate (TNR), defined in Equation 20. Fourth is precision, defined in Equation 21. These statistics provide well known metrics within the machine learning community which can be used to compare methods and identify better and worse results [56].

$$AC = \frac{TN + TP}{TN + TP + FN + FP} \tag{18}$$

$$TPR = recall = \frac{TP}{TP + FN} \tag{19}$$

$$TNR = specificity = \frac{TN}{TN + FP} \tag{20}$$

$$precision = \frac{TP}{TP + FP} \tag{21}$$

The last statistic used for analysis is the pruning rate for both bolide and non-bolide data points. This metric was created for this thesis with the purpose of acting as an intermediary tool in order to analyze the performance of pruning a dataset. This measure was necessary due to the size of some of the datasets and the lengths of time needed to complete certain machine learning algorithms. The pruning rate is a simple metric. It is the ratio of the number of points pruned or removed from the dataset to the total number of points in the dataset.

E. SUMMARY

This chapter described the preparatory steps for transforming the dataset from an input CSV to the Pandas dataframe format needed by the machine learning algorithms. Furthermore, the software language and coding libraries used to complete this transformation were introduced. Finally, the experimental design and performance statistics used to conduct the experiments and measure its outcome were provided.

THIS PAGE INTENTIONALLY LEFT BLANK

V. APPLICATION OF UNSUPERVISED MACHINE LEARNING TO BOLIDE DETECTION

This chapter describes the process and results of using various unsupervised machine learning algorithms in an effort to detect bolides within Doppler radar data. The goal of this thesis is to take the output dataset from the preprocessor, denoted as the data in Table 2, and apply various machine learning techniques to analyze and identify outlier bolides, while rejecting the nominal weather returns and terrestrial ground returns. Because the data is unlabeled, meaning there is no truth data that identifies each radar return, multiple unsupervised learning algorithms are attempted to identify the bolides: principle component analysis, k-means clustering, t-Distributed Stochastic Neighbor Embedding (t-SNE), and lastly a combination of these algorithms. Each algorithm is described, applied against the problem datasets, and performance assessed.

A. COMPLETING A BASELINE ANALYSIS OF THE DATASET

The first step of applying the machine learning algorithms to the dataset is creating a baseline to compare against. Figure 23 shows the information from Table 2 for the Ash Creek bolide from 15 February 2009. Each plot depicts a 3D scatterplot of three features, with colors coding for the distance from the radar site to the radar return. The left plot has the three base radar moments, the middle plot shows the scaled version of the base radar moments, and the right plot depicts the change from t_0 to t_1 to t_2 for each base radar moment. Lastly, the green Xs represent the radar returns associated with the Ash Creek, Texas, bolide, found at 161° and 90,000 m ($9.0e+04$ m) from the radar site in Figure 17. This plot will be considered the baseline plot.

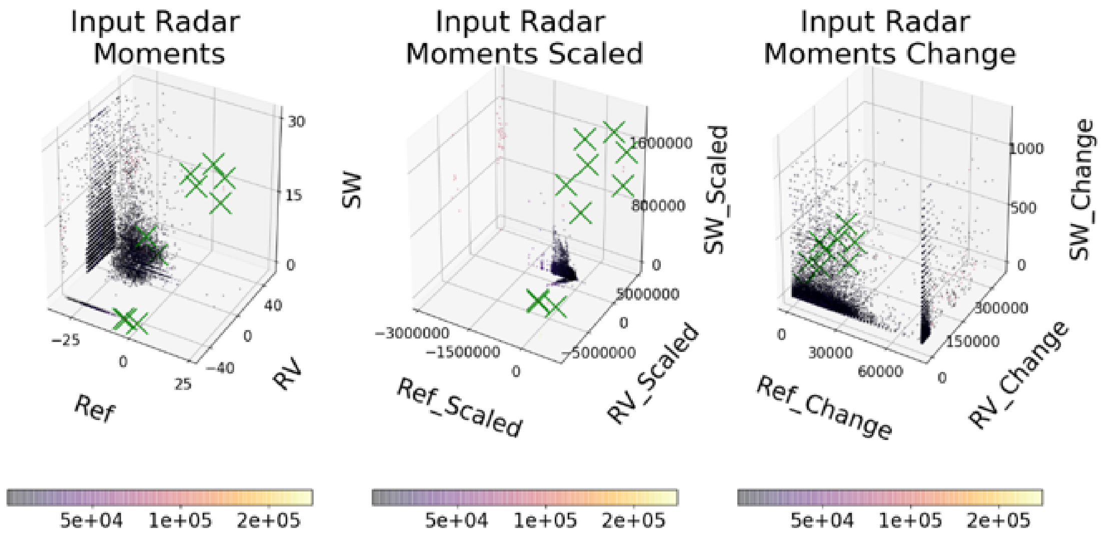


Figure 23. 3-D plots representing the data input to the analyzer; green Xs denote bolide returns

Visual analysis of the distribution and shape of the returns from Figure 23 shows no clustering of all the bolide returns in any of the three plots. The closest clustering comes from the change detection plot on the far right of the figure, but although the bolide returns are more closely clustered, they are not separated from all the non-bolide returns. The only other notable feature is seen in the left and middle plots, where there appears to be 2–3 distinct groups of bolide points. Therefore, upon visual inspection, the plots are not easily separable using linear methods, but this level of separation can be used as the standard to measure against.

B. PRINCIPAL COMPONENT ANALYSIS

1. Method Description

Principle Component Analysis (PCA) is the first method tried to analyze the radar data. This method has its origins with Pearson in 1901, but has been used in numerous ways since [55]. Although this multivariate analysis technique has many applications, two of the greatest benefits are through dimensionality reduction, feature selection, and data visualization [55,56]. The primary application for this research will be feature selection and subsequent visualization for outlier detection.

PCA gives an ortho-normal rotation of a multivariate dataset, $x_1,...,x_n$, of dimension q and size n. The orthogonal length-one vectors $v_1,...,v_q$, defined as the Principal Components (PCs), determine the rotation [55,56,64]. The first PC, v_1, gives the projection of the dataset with the largest (sample) variance. That is, among all length one vectors, v, the first PC maximizes the variance given by Equation 22 where $\overline{x} = \frac{1}{n}\sum_1^n v^t x_i$ is the vector of the dataset sample means.

$$\text{variance} = \frac{1}{n-1}\sum_1^n (v^t x_i - v^t \overline{x})^2 \tag{22}$$

The variance of a linear function of features may be expressed more compactly as $v^t \Sigma v$ where Σ is the sample variance-covariance matrix of the dataset (with sample variances on the diagonal and covariances on the off-diagonal of Σ). Subsequent PCs maximize the projected dataset variance among all length one vectors orthogonal to proceeding PCs. Thus, PCA gives a sequence of PCs with corresponding non-increasing variances $\lambda_1 \geq ... \geq \lambda_q \geq 0$ [55,56,64]. Often, the first several PCs are used to construct new features from the original dataset. For example, we use the first three PCs to display higher dimensional data in three dimensions. More generally, the projection of the dataset onto the first k PCs may be used in analysis in place of the original q variables. If the variances corresponding to the unused PCs are small, not much is lost with this dimension reduction.

The PCs and the corresponding variances are respectively the eigenvectors, v, and eigenvalues, λ, of Σ. An example, where $q = 2$ and $k = 2$, is depicted in Figure 24 reproduced from [65]. Here, the two variables are standardized by subtracting their respective sample means and dividing by their sample standard deviations prior to performing PCA. Thus, the plot on the right of Figure 24 is an ortho-normal rotation of the rescaled data. This rescaling puts all of the original variables on the same scale. Further details of PCA can be found in [55,56].

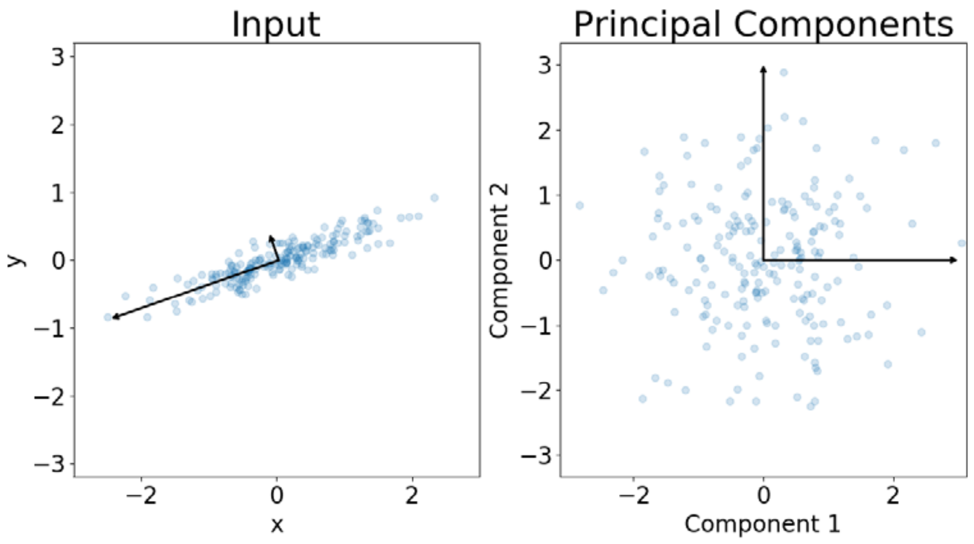


Figure 24. A PCA transformation using eigenvalues and eigenvectors.
Source: [65].

2. PCA implementation

Using the Scikit-learn library enables easy access to PCA methods [66]. The output of the preprocessor is used as the input to the PCA process, using the features from Table 2. A check was also executed to determine if the dual polarization features were available. If the dual polarization features were available within the dataset, a PCA model was created with 13 features; if they were not available, a PCA model was created with 10 features. In both cases, features are standardized prior to PCA to help in the model fitting process. After PCA, the outputted PC's (the eigenvectors), the corresponding variances (the eigenvalues), and the rotated data are saved.

3. Analysis

Plotting the information from Scikit-learn's implementation of the PCA method yields Figure 25. The left plot in Figure 25 can be used to determine the importance of each PC relative to the full dataset. The sum of the eigenvalues, equal to the sum of the original dataset variables' variances, is a measure of dataset variability. The left plot of Figure 25 gives ratio of each eigenvalue to the sum of the eigenvalues. For example, PC1 has a variance of 29%, showing that it represents 29% of the overall variance in the dataset. PC2 has a variance of 18% and PC3 has a variance of 13%. Collectively, they can represent approximately 60% of the dataset variability. The right plot of Figure 25 shows the “singular values,” which are proportional to the square-root of the eigenvalues. These are proportional to the standard-deviations of the dataset, projected along each PC. They represent a different way to look at how each PC contributes to the dispersion of the data, and in this case confirms how each PC affects the whole dataset. The interesting feature of this output is that there is no clear break where additional PCs no longer matter. All the PCs are important as far as representing the data is concerned. Furthermore, for a dataset with a small number of features like 10–13, it appears to be beneficial to keep all the features.

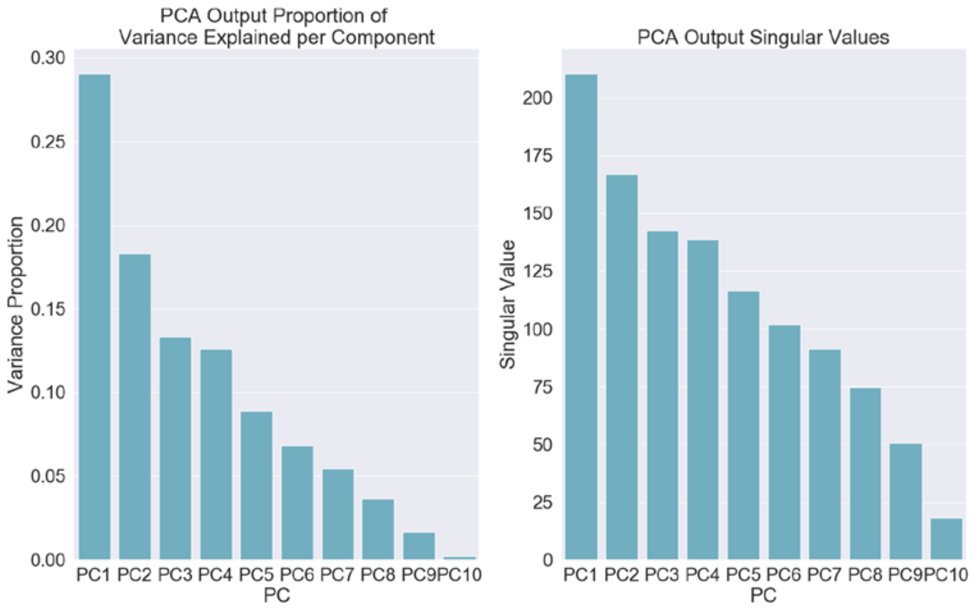


Figure 25. PCA process output: scaled PCA variance and singular values for the Ash Creek, Texas, bolide

The projected dataset from PCA can be plotted and compared against the baseline analysis plot in Figure 23. This is what is plotted in Figure 26: PCs 1, 2, and 3 are used in a 3D scatterplot, with PC 4 used to color the data points. In this manner 4 dimensions of data are available for analyzing where the non-bolide points are in relation to the large circles with Xs, denoting bolide locations. Looking at the plot, most of the data points are located in a cluster in the center, with most of the bolide points and some additional points visible as outliers. Furthermore, the bolide points are found in 3 mini-clusters, which prevents the results of this dataset from being easily separable. However, the PCA projection is an improvement upon the results of Figure 23. Lastly, using PC 4 yields no additional information; the colors of all bolide points are different. Therefore, even though a 4D tesseract plot cannot be seen directly, there is enough information to state that bolides are not separable in the 4th dimension.

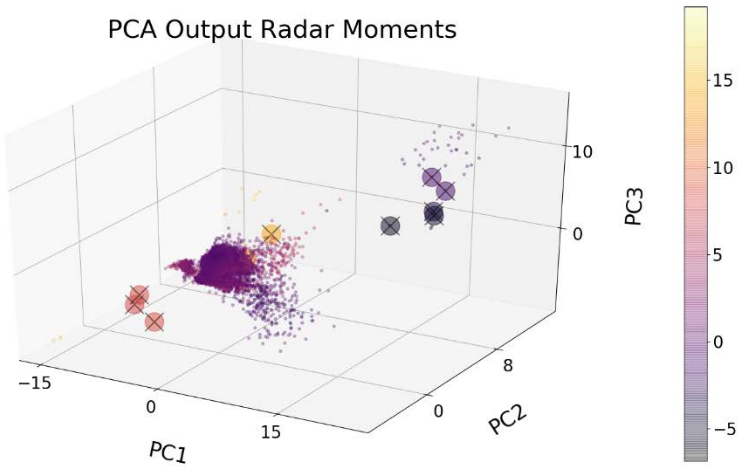


Figure 26. Scatterplot of PCA transformed data from the Ash Creek, Texas, bolide with PC4 coloring

4. Limitations

Figure 26 is a helpful plot in isolating where the bolide vs. non-bolide points are located. The non-bolide points are predominantly clustered in the center of the plot, while the bolides are the outliers and more distant from the center. However, this result is particular to the Ash Creek, Texas, dataset. Using the KDAX radar data for the Sutter's Mill, CA bolide yields the plot in Figure 27. There are substantially more data points overall comparing the number of data points from the Sutter's Mill bolide datasets in Table

3 against those of the Ash Creek bolide dataset, and the bolide points are no longer outliers when compared to the rest of the radar returns. The results are similar when comparing the datasets from the KBBX and KRGX radar sites. Secondly, because PCA only reduces variance at the expense of the local structure of the data, a lot of higher dimensional information is lost. Therefore, even though PCA was observed to be helpful in isolating bolides in some cases, PCA alone is not capable of doing so in the majority of cases tested. Further analysis using different techniques is required.

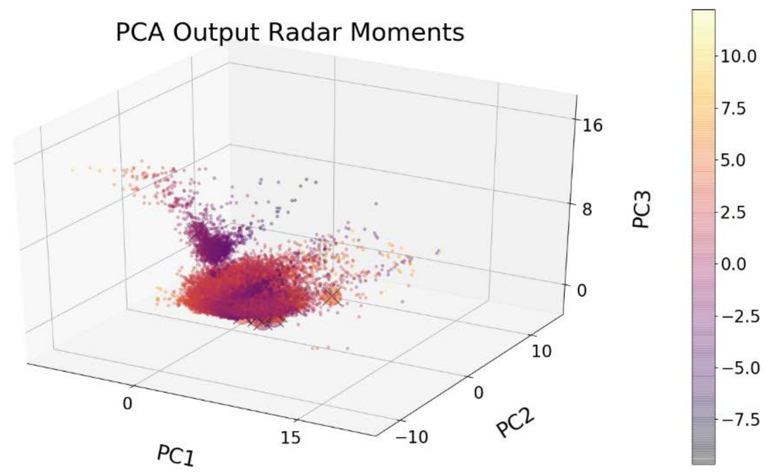


Figure 27. Scatterplot of PCA transformed data from the KDAX Sutter's Mill, California, bolide with PC4 coloring

C. K-MEANS CLUSTERING

1. Method description

The bolide data points in Figure 26 and Figure 27 are not easily separable from the non-bolide data points. However, looking at the two figures raises the question of whether a relationship exists between the centroid of all the data points and the distance to each of the bolide data points. Within unsupervised learning, k-means clustering is a method that could be used in order to characterize this relationship.

K-means clustering is a well-known and computationally efficient algorithm to separate data point into clusters [67]. It works on both high and low dimensional datasets and compares the similarity of each data point in order to assign similar data points into the same clusters. It is implemented algorithmically in [65].

The algorithm itself is completed using 4 steps. The user initializes the algorithm with *k* numbers of clusters, which firstly generates random initial locations for cluster centroids μ_j where $j = 1,...,k$. Secondly, each data point is assigned to the nearest of the k clusters using the Euclidean distance calculation defined in Chapter IV, Equation 14 [67]. Thirdly, the centroid locations for each cluster are recalculated using Equation 23, where μ_j is the mean location of the cluster data points, m_j is the number of data points in that cluster, and *x* the data points within cluster C_j. Lastly, the fourth step is repeating steps 2 and 3 until the cluster locations no longer move and convergence has occurred [56,68].

$$\mu_j = \frac{1}{m_j} \sum\nolimits_{x \in C_j} x \tag{23}$$

Convergence occurs when the inertia, also known as the within-cluster sum of squares criterion, is minimized and ceases to change on subsequent iterations of the algorithm. The inertia is defined in Equation 24 [68].

$$Inertia = \sum_{j=1}^{k} \sum_{x \in C_j} (\left\| x - \mu_j \right\|^2) \tag{24}$$

The k-means clustering algorithm [65,67] does have flaws. One is that the number of clusters must be assigned before running the algorithm, so dynamic expansion or elimination of clusters is not allowed. Secondly, an optimal solution will always be found, but will often be a locally optimal solution and not a globally best solution. Thirdly, the algorithm fails to properly cluster nonlinear boundaries [65]. Fourthly and lastly, the program runtime scales with the number of data points in the dataset, meaning it falls prey to the curse of dimensionality [65].

2. k-means Clustering Implementation

The Scikit-learn library has a k-means algorithm, which is used in this research. The documentation provides adequate information on how to call and modify the function calls. Therefore, the only decision required is to determine the number of clusters to use in the function call. However, this is not an easy number to determine. The ideal situation would be to have two clusters, where all the bolides would cluster in one group and all the non-bolide points would cluster in the other group. However, looking at the plotted output from the PCA process in Figure 26 and Figure 27, this ideal case is not likely due to the distribution of the data points throughout the graph of the first three principal components. Therefore, the algorithm is executed once for $k = 1$ clusters and once for $k = 2$ clusters to return the centroid of each cluster and the within-cluster sum of squares value for each case in an attempt to determine experimentally whether it is possible to cluster the data points into bolide and non-bolide clusters. Furthermore, the k-means algorithm is applied to the PCA projected data.

3. Analysis

The results of running the algorithm for the Ash Creek, Texas, dataset are depicted in Figure 28 for 2 clusters and Figure 29 for 1 cluster. The baseline plots are the same as those produced from the PCA process, but with large red Xs to denote the cluster centroids from the k-means clustering algorithm. Analyzing Figure 28 shows that after the algorithm finishes clustering the dataset, the data points are split roughly down the middle of the dataset. Therefore, the centroids are both towards the center of all the data points, which can be seen in Figure 28. This means that due to a nonlinear boundary separating bolides from non-bolides, the data could not be split into a bolide cluster and a non-bolide cluster, and another method of analysis was required.

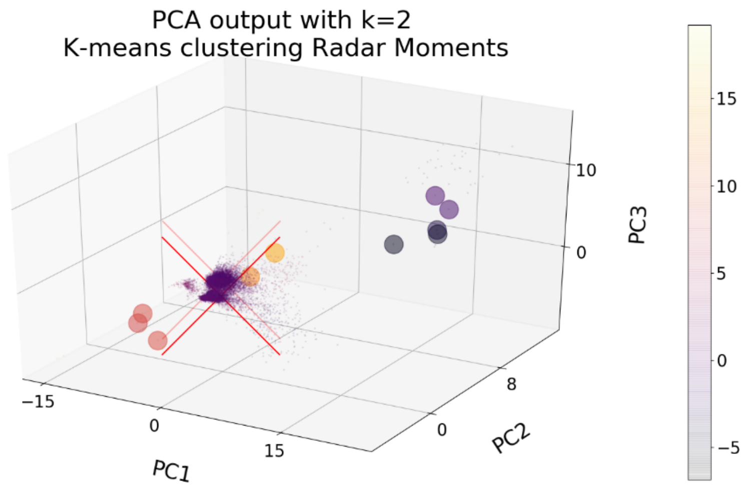


Figure 28. Scatterplot of PCA transformed data from the Ash Creek, Texas, bolide plus $k = 2$ k-means clusters denoted by two Xs.

Looking at the results from Figure 29 shows the centroid for the full dataset. The purpose behind determining the location of the centroid is to then calculate the distance from the centroid to each of the bolide returns. For the Ash Creek, Texas, bolide, the mean Euclidean distance from centroid to all the bolides is 12.7 units; however, for the Sutter's Mill, California, bolide, the mean Euclidian distance is 3.17 units. The full list of results is in Table 5, but comparing all the minimum, maximum, and mean values shows little consistency, even from the same Sutter's Mill bolide viewed from 3 different radar sites.

Table 6. Minimum, Maximum, and Mean Euclidean distances from the k-means process

Bolide Location	**Min Distance**	**Max Distance**	**Mean Distance**
Ash Creek, TX (KFWS)	6.31 units	19.1 units	12.7 units
Sutter's Mill, CA (KDAX)	1.28 units	8.09 units	3.17 units
Sutter's Mill, CA (KRGX)	7.26 units	8.82 units	7.83 units
Sutter's Mill, CA (KBBX)	2.35 units	6.51 units	3.70 units

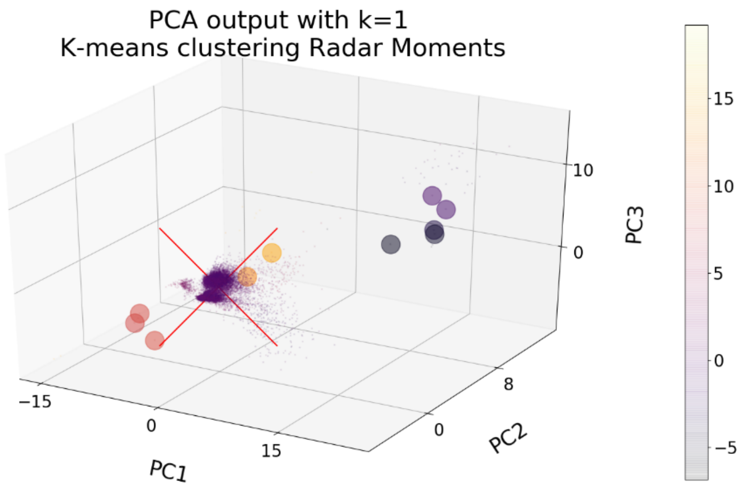


Figure 29. Scatterplot of PCA transformed data from the Ash Creek, Texas, bolide plus $k = 1$ k-means clusters denoted by a single X.

4. Limitations

The k-means algorithm correctly assigns all the data points into either one or two clusters, and identifies the centroid. However, this was found to be unhelpful for bolide detection due to the location of the clusters being too similar. Similarly, using a single cluster to determine the minimum, maximum, and mean squared distances from centroid to bolides is unhelpful on its own due to the variability among different bolide events and datasets. Therefore, using k-means clustering as a method for bolide detection is insufficient, and alternative techniques should be tried.

D. T-DISTRIBUTED STOCHASTIC NEIGHBOR EMBEDDING

One of the limitations of both PCA and k-means clustering is that both of the previously described methods only use data in three dimensions for plotting and up to a fourth for data point coloring. However, up to 10 additional dimensions worth of data exists from the dataset of 13 features and could be used for clustering. Unfortunately, it is not possible for humans to visualize information in 3 or greater dimensions. Therefore, a method of dimensionality reduction is needed, but something unlike the linear reduction methods used by PCA, which when visualized in lower dimensions tends to obscure clusters that are linearly separable in higher dimensions. Instead, manifold learning provides a mechanism to represent the nonlinear structure of data in lower dimensions [69]. In particular, this research uses the more recently developed t-Distributed Stochastic Neighbor Embedding (t-SNE) method of dimensionality reduction for visualizing data and cluster analysis.

1. Method Description

The t-SNE algorithm is an improvement to the original Stochastic Neighbor Embedding method from 2002, and was developed by van der Maaten and Hinton in 2008 [70]. t-SNE functions by measuring the distances between data points in higher dimensional space and using a cost function to drive convergence towards the best fit for that relationship in lower dimensional space [70]. This lower dimensional mapping can be used in either 2D or 3D as a way to visualize the dataset in ways that cannot otherwise be done.

t-SNE maps higher dimensional data points to those in lower dimensions through a series of steps. The first step is by converting the Euclidean distances between data points into affinities, which are a way to relate the various distances between data points. The original SNE relies upon a Gaussian distribution methodology in Equation 25 to create the conditional probability $p_{j|i}$ where σ_i^2 is the variance, and both x_i and x_j are data points. The equation effectively gives nearby data points relatively high values, and distant data points values approaching 0 [70]. Mapping the affinities to lower dimensional space is done using a Gaussian distribution to form the conditional probability $q_{j|i}$ found in Equation 26 where y_i and y_j are analogous to x_i and x_j, except in lower dimensional space. The minimization occurs using a gradient descent algorithm to minimize a cost function, where the cost function is the Kullback-Leibler divergences. However, this method results in what van der Maaten and Hinton deemed a “crowding problem” where optimizing the cost function becomes difficult and clusters of data points appear to congregate together in the same region without enough gaps between them [70].

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)} \tag{25}$$

$$q_{j|i} = \frac{\exp(-\|y_i - y_j\|^2)}{\sum_{k \neq i} \exp(-\|y_i - y_k\|^2)} \tag{26}$$

Fixing the optimization difficulties was accomplished by swapping out the Gaussian distribution for a Student's t-distribution with one degree of freedom [70], but only in the lower dimensional mapping. This is completed using Equation 27 when calculating the conditional probabilities in the lower dimensional space. In essence, the higher and lower dimensional spaces are no longer symmetric with regard to their distribution mappings, and this tends to create greater distances between clusters in the lower dimensional space while still retaining the closeness of near points in the clusters [70]. This can be seen by comparing the two distributions in Figure 30, where the distributions only overlap in two specific points and otherwise enable the asynchronous mapping used by t-SNE.

$$q_{j|i} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|y_i - y_l\|^2)^{-1}} \tag{27}$$

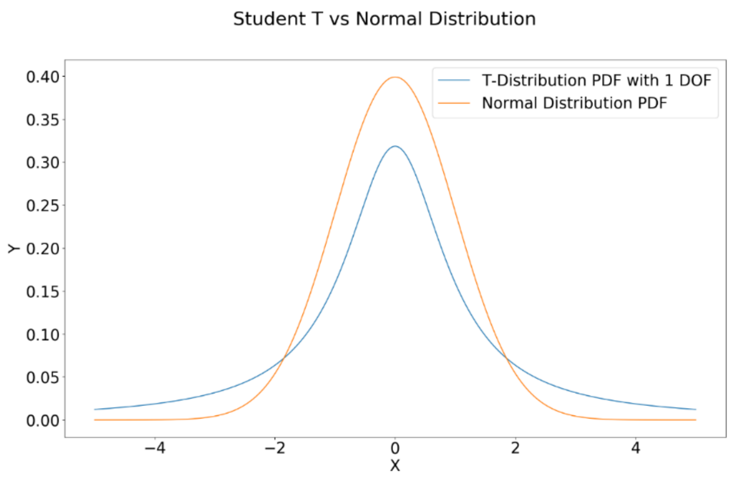


Figure 30. Comparison of a student T-Distribution vs. a Normal Distribution

The last piece of the algorithm that needs to be explained is the perplexity value. This value is the number of nearest neighbors used when calculating the conditional probabilities, and acts as a smoothing function for structure of the data points. It is defined in Equation 28, where P_i is the perplexity and $p_{j|i}$ is the conditional probability between data points [70]. The effect of the perplexity value is that larger values make the lower dimensional output less effected by the local structure because more data points are used in the mapping process, while a smaller perplexity enables closer adherence to the local structure. Typically, perplexity values between 5 and 50 are advised [69].

$$P_i = 2^{\sum_j p_{j|i} \log_2 p_{j|i}} \tag{28}$$

2. t-SNE Implementation

The t-SNE package within the Scikit-learn library was used for bolide detection in this research. The input to the algorithm was the PCA transformed dataset and multiple tuning parameters, and the outputs were 3D in nature, from which both 2D and 3D plots were created. The only input parameters that are modified from defaults were the initialization value of "pca" for additional global stabilization, a random state of 0 which sets the random number generator seed for reproducibility, a perplexity value of 50, and lastly a max number of iterations of 250 to speed up the algorithm.

3. Analysis

The results of the t-SNE algorithm are depicted in Figures 31–34. Both a 2D and 3D output are shown with the data created from the t-SNE output and the coloring done with the distance from radar site to radar return. The bolides are highlighted with Xs in the circles. For the Ash Creek, Texas, bolide, the separation is excellent in both 2D and 3D plots. Furthermore, all of the bolides are congregated together in one cluster, indicating that they are near each other in higher dimensional space. Lastly, note that the coloring is only based upon the distance from the radar site, and bears no meaning as one of the PCA transformed input values to the algorithm.

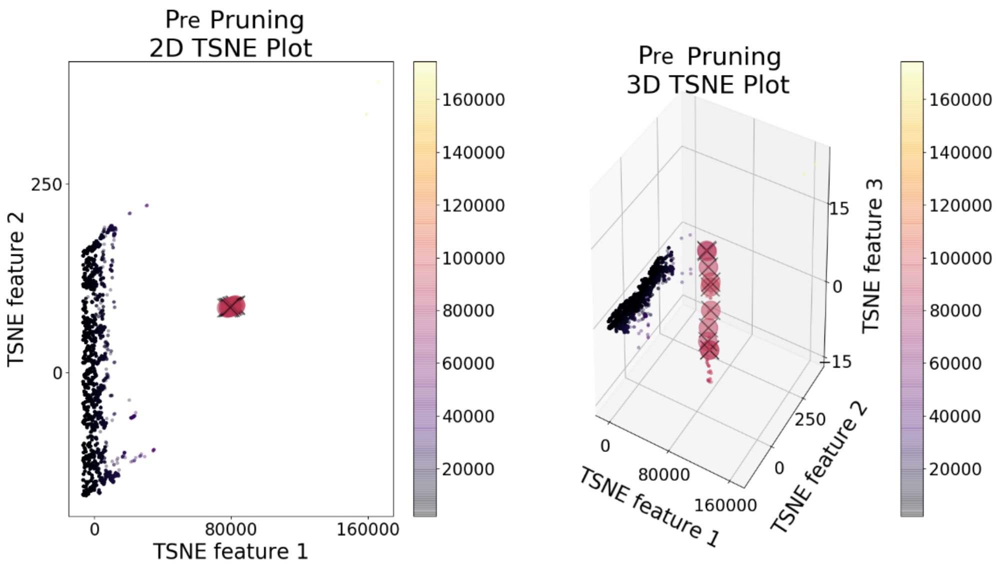


Figure 31. t-SNE output for the Ash Creek, Texas, bolide in 2D and 3D

Looking at the Sutter's Mill, California, results, there is less separation between bolide to non-bolide points than there is for the Ash Creek bolide. Each cluster of bolide points appears a little separated from the non-bolide points; however, without knowing the exact locations of the points, each of Figures 32–34 have multiple small clusters of data points that could be confused with the bolide points. Fortunately, these small clusters are distant from the bulk of the non-bolide points, so although it is not clearly linearly separable, it is a helpful outcome. This outcome is also present in all three results showing that it is not a one-off result.

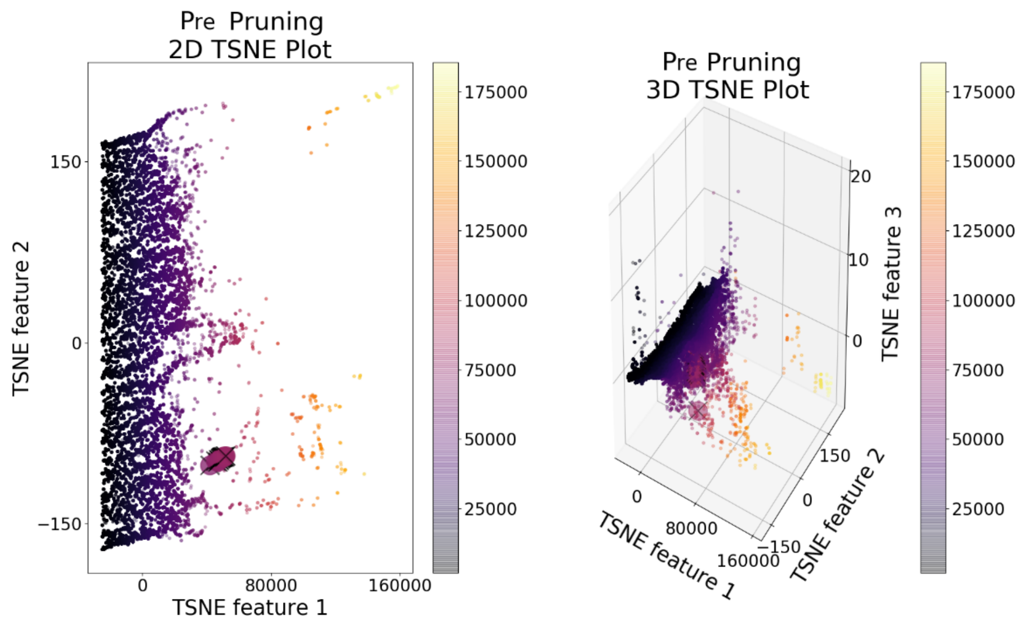


Figure 32. t-SNE output for the Sutter's Mill, California, bolide from the KDAX radar in 2D and 3D

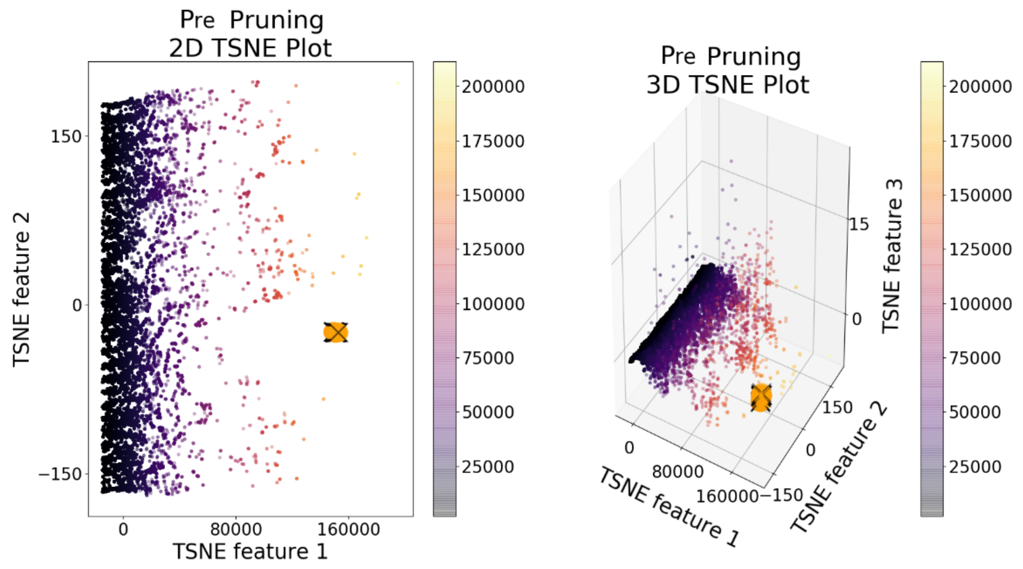


Figure 33. t-SNE output for the Sutter's Mill, California, bolide from the KRGX radar in 2D and 3D

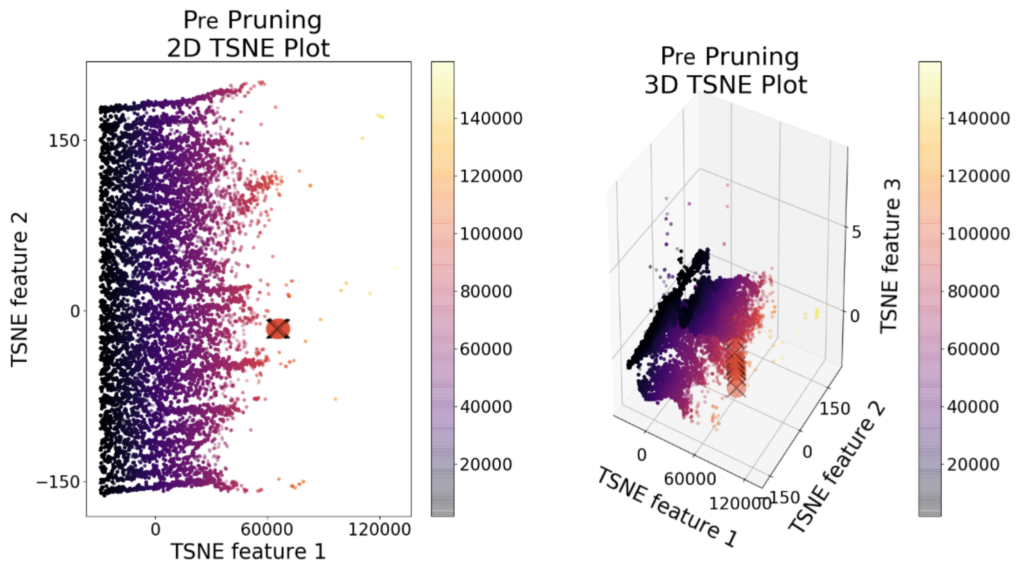


Figure 34. t-SNE output for the Sutter's Mill, California, bolide from the KBBX radar in 2D and 3D

4. Limitations

There are two limitations to the t-SNE algorithm: one minor and one major. The minor issue has already been identified, the fact that the bolide returns are only linearly separable in the best and most ideal of cases. Otherwise, the bolide returns could be confused with non-bolide returns. The major issue is the computational cost of executing the algorithm. The complexity of the algorithm is $O(n^2)$ in both memory and computational requirements, which is a substantial cost [70]. Furthermore, the creators of the algorithm warn against datasets with greater than 10,000 data points, which is the case in this research. In a practical sense, the four bolide datasets require substantial and impractical amounts of time to complete. The Ash Creek bolide requires 25 minutes to complete, but the Sutter's Mill bolide requires 3, 8, and 16 hours to complete for the KDAX, KRGX, and KBBX radars respectively. Because of the time sensitivity needed for recovery, this algorithm is not viable on its own.

E. COMBINING ALGORITHMS

The PCA, k-means, and t-SNE algorithms were each applied to multiple bolide algorithms; unfortunately, each one was individually not capable of identifying bolides. In particular, the PCA and t-SNE algorithms showed much promise in detecting outlier bolide radar returns among the non-bolide returns, but neither yielded a clear separation of the two classes and the computation cost for t-SNE was too high for regular, expeditious use. These limitations pointed towards a path forward: firstly, creating a method to lower the computational cost of t-SNE, and secondly, combining it with the existing PCA and t-SNE algorithms into a more comprehensive algorithm for bolide detection.

1. Method Description

The more complex of the two steps forward is reducing the computational cost by reducing the size of the dataset. The original dataset size varies between 15,000 to 121,000 data points as shown in Table 3, each with 10 to 13 features each. This is substantially more than the 10,000 data point maximum limit identified by van der Maaten and Hinton [70]. Therefore, a substantial reduction in data points is required to enable using t-SNE in a reasonable amount of time.

The method proposed in this research is a Nearest Neighbors Density Pruning (NNDP) method, which uses data point filtering based on the Nearest Neighbors algorithm. The NN algorithm itself can be used as a classifier that memorizes a training dataset to determine classification bounds that new data points can be compared against [56,62]. However, it is not the classification piece that helps in this case, but rather the aspect of the algorithm that calculates the Euclidean distance from a given data point, *p*, to the nearest m surrounding data points, q_j for $j = 1, ..., m$, that is important [63]. Therefore, building upon the Euclidean distance formula from Equation 14, the mean distance to the nearest m data points is calculated using Equation 29 where the sum of the distances from each data point *p* to every other point $q_1, ..., q_m$ is averaged by dividing by the number of points *m*. This output provides a way to determine how close, on average, a single point is compared to the surrounding data points.

$$\mu_m = \frac{\sum_{j=1}^{m} d(p, q_j)}{m} \tag{29}$$

The mean distance μ_m to the *m* closest points then provides a parameter that can be tuned using an upper and lower bound to separate out and remove non-bolide returns. For example, a single radar return data point that is distant from the 50 surrounding data points will have a very large μ_{50}; this data point is unlikely to be a bolide because bolides typically comprise multiple data returns clustered together. Conversely, a non-bolide data point that is closely surrounded by similar non-bolide data points would have a very small μ_{50} value. By iteratively tuning the μ_{Upper} and μ_{Lower} bounds, there exists a method to remove data points that are unlikely to be bolides.

The simpler of the two steps forward is finding a way to chain multiple algorithms together. This process is relatively straight forward and has already been completed by using the PCA transformed output as the input to the k-means and t-SNE algorithms. The method is simply by using the output from one algorithm as the input to another algorithm, and because the syntax has been standardized, the only necessary decisions are what single algorithms and what order should the chained algorithm execute. The PCA and t-SNE algorithms have already been introduced as options, but the NNDP algorithm can be used as well. Because the density pruning method uses percentile values, the algorithm can be executed multiple times with independent *m* values, upper bounds, and lower bounds. Firstly, this enables increased accuracy without using very precise upper and lower bound values that would be required for execution in a single step. Secondly, breaking the pruning into multiple steps enables removing non-bolide points based upon the global structure using a large μ_m value and then the local structure using a small μ_m value. Because the local and global structures of bolides differ based upon the m value, this method will highlight those differences.

2. Combined Algorithm Implementation

Using the above knowledge of the individual methods, the overall flow of the combined algorithm is: 1) Conduct PCA to increase the variance of the dataset, 2) Use the NNDP to reduce the overall dataset size, 3) Conduct t-SNE to reduce the dimensionality of the dataset and cluster the bolides in that lower dimensional space, 4) Use NNDP to remove non-bolides from the t-SNE dataset, and 5) Output the results to the original 2D radar plot with radar returns identified as probable bolides. The same Scikit-learn libraries previously used are again utilized and in the case of the NN library, retooled for use in this larger, combined algorithm. The final experiment was executed for all four bolide cases in Table 3, and the results analyzed for separability and accuracy in classifying bolides. Furthermore, the intermedia results from the combined algorithm are presented as well as the final results at the completion of the combined algorithm method.

3. Intermediate Results Analysis

The original PCA results for the Ash Creek, Texas, bolide are depicted in Figure 26, and this dataset is then input into the NNDP algorithm. The performance of the algorithm is depicted in Figure 35. Each data point for the scatter plot is μ_{50} for each radar return in 10-dimensional space. The horizontal lines are the upper and lower percentile bounds $\mu_{Upper} = 99.9$ and $\mu_{Lower} = 96$. The top plot in Figure 34 is the input dataset, and the bottom shows the post pruning dataset, where the returns outside the bounds are removed from the dataset. By excluding the data points outside the upper and lower bounds, 96.1% of the data is excluded, leaving only 3.9% for further analysis.

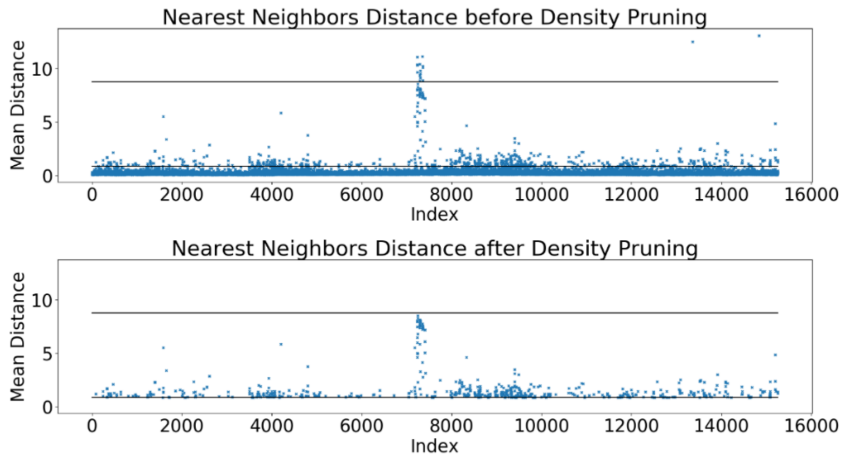


Figure 35. NN density pruning for the Ash Creek, Texas, bolide with horizontal line bounds at 96% and 99.9%

The results of the NNDP algorithm are shown in Figure 36. It is the same plot as Figure 26, except after exposure to a 96.1% pruning rate. The plot is visibly less cluttered after the data point reduction. However, some of the bolide radar returns have been removed in the process as well. This is not ideal, but some amount of false negatives are acceptable so that the algorithm's μ_{Upper} and μ_{Lower} bounds are not so overfit that the algorithm cannot be used for other datasets. Specifically, the original dataset had 10 identified bolide signatures, but after the first round of pruning, only 6 remained.

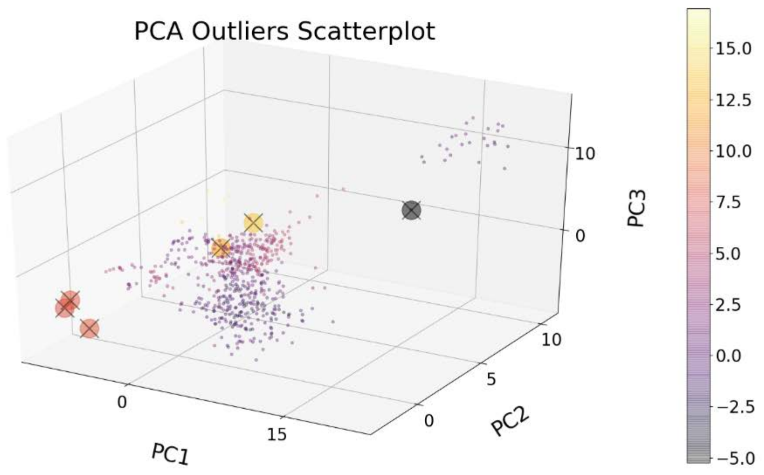


Figure 36. The first three principle components of the Ash Creek, Texas, bolide after NN Density Pruning

The results shown in Figure 36 are then input into the t-SNE algorithm. Since there are substantially fewer data points, the algorithm is capable of executing in a significantly reduced time. The results of the t-SNE algorithm are depicted in Figure 37, and the results are similar to those in Figure 31, but with fewer data points, which is the desired outcome. However, there are still too many data points in the output dataset, so the output from the t-SNE algorithm are then used as the input to a second round of the NNDP algorithm.

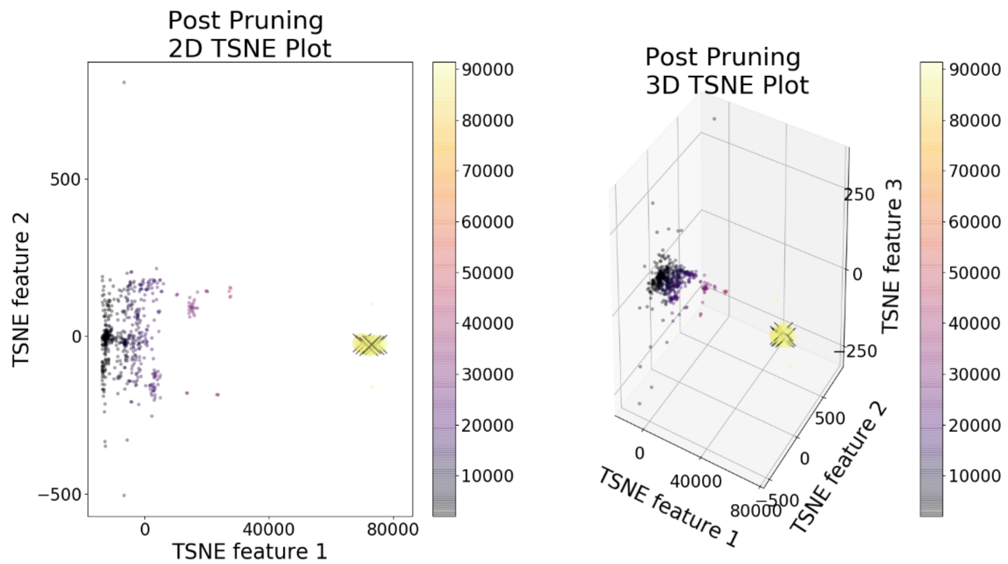


Figure 37. Post NN Density pruning followed by t-SNE output for the Ash Creek, Texas, bolide

Figure 38 shows the results of the second round of Density Pruning plotted with the first three principal components. The algorithm used a $\mu_{Upper} = 99$ and $\mu_{Lower} = 92$ percentile bounds and an m value of 20 that yields a μ_{20} which prunes based on the local cluster of the bolides instead of the previous global structure. Where the original dataset has 10 identified bolide signatures and the first round of pruning reduces it to 6 bolide signatures, this second round of pruning reduces it to 4 bolides. However, the pruning rate is 99.7%, yielding 42 data points to review. Four of them are bolides, with 3 clustered together, which is a beneficial result for classification.

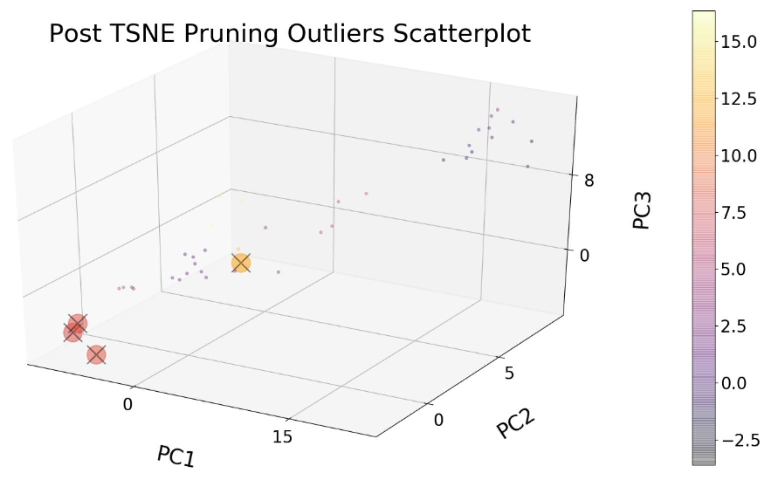


Figure 38. The first three PCs of the Ash Creek, Texas, bolide after density pruning 99.7% of the data points

4. Final Results Analysis

Although the second round of pruning depicted in Figure 38 is helpful in seeing the reduction in data points that should be investigated as possible bolides, it is not helpful for finding bolides within the original radar moment plots. Therefore, to resolve this, the information from Figure 38 is exported back to the original radar plots. Any of the original radar moment plots could have been chosen as the baseline plot, but the radar Reflectivity moment was chosen in Figures 39 through 42. Therefore, each of those figures shows the original radar reflectivity plot on the left, contrasted against the same plot on the right with green Xs denoting the locations of probable bolides. Each of these locations is the output of the combined algorithm method and represents 0.3% of the data points from the original dataset.

The Ash Creek, Texas, Bolide is the most trivial of all the cases sampled. There is limited radar return noise that could be confused with bolide returns. The results show this as well, where the bolide is highly covered by green Xs, and there are limited other locations to investigate. In total, there are 42 data points to review as an output from the chaining algorithm, of which there are 6 small clusters of less than five data points, 1 cluster of 6 data points, and the large cluster of 22 returns associated with the bolide. From Table 6, the Accuracy and Specificity are very high, but the overall results are skewed in favor of high percentages because of the large number of non-bolides within the dataset. A better way to interpret the results when two classes are imbalanced, such as this case with few bolides and many non-bolides, is through the recall and specificity rates. These measures paint a more accurate, but less beneficial picture. However, even though the numerical numbers are not as high as they could be, the bolide is clearly identified within Figure 39, and a human reviewer would select it for further analysis, so the algorithm has accomplished its goal of distilling down the large number of results to something more manageable for a human to look through. Lastly, the execution time is reduced from an original 8 minutes in the original t-SNE algorithm down to less than one minute. Although this reduction was not vital for this dataset, the combined algorithm shows promise for applications to lager datasets.

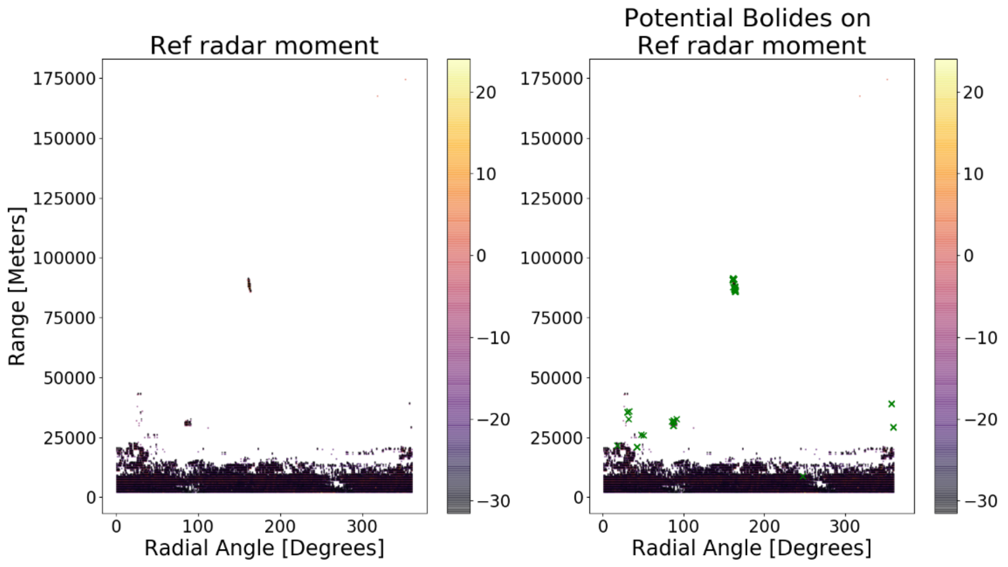


Figure 39. Final results for the Ash Creek, Texas, Bolide with possible bolides in green Xs on the right

Table 7. Confusion Matrix and associated algorithm execution results for the 4 bolide datasets

	KFWS Ash Creek, Texas	KDAX Sutter's Mill, California	KRGX Sutter's Mill, California	KBBX Sutter's Mill, California
Total data points	15,626	66,388	137,332	121,782
Bolide data points	53	69	22	21
Non-Bolide data points	15,573	66,319	137,310	121,761
Data points to review	42	182	375	332
True Positive Points	22	9	18	10
False Positive Points	20	173	357	322
True Negative Points	15,553	66,146	136,953	121,439
False Negative Points	31	60	4	11
Accuracy Rate	99.67%	99.65%	99.74%	99.73%
True Positive Rate (recall)	41.51%	13.04%	81.82%	47.62%
True Negative Rate (specificity)	99.87%	99.74%	99.74%	99.74%
Precision Rate	52.38%	4.95%	4.80%	3.01%
Execution Time [Minutes]	0.63083084	3.835250111	6.78067112	7.317465468

Reviewing the next three test cases, the Sutter's Mill bolide from the KDAX, KRGX, and KBBX radars yields a similar picture. The radar dataset is significantly more cluttered than that of the trivial case. There are a large number of individual data points to review as well as clusters of data points to review in each of Figures 40, 41, and 42. However, this makes sense because the dataset itself is larger, with 182, 375, and 332 data points to review as an output from the chaining algorithm. A human reviewer would be able to immediately remove some of the data points since they are solitary points, reducing the number to review relatively easily, but the overall number of data points to review is still an issue that needs resolving.

Reviewing the results in Table 6 for the Sutter's Mill Bolide, the Accuracy and Specificity values are still very high, just like that of the Ash Creek Bolide. However, the Precision rates are much lower, and the Recall values are not consistent between the three radar sites. Recall rates for KDAX are very low, moderate for KBBX, and high for KRGX. This shows that the same event can be seen vastly differently from three different radar locations. The execution time of the algorithm is significantly reduced down from the original 3, 8, and 16 hours to less than 4, 7, and 7 minutes respectively. Again, the data point reduction methods resultant from pruning enabled the use of t-SNE.

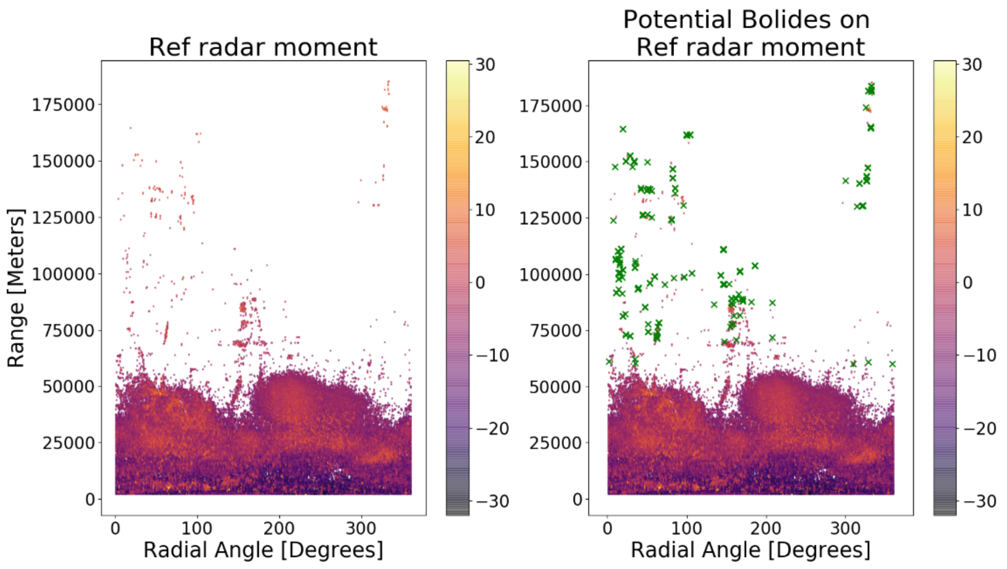


Figure 40. Final results for the KDAX Sutter's Mill Bolide with possible bolides in green Xs on the right

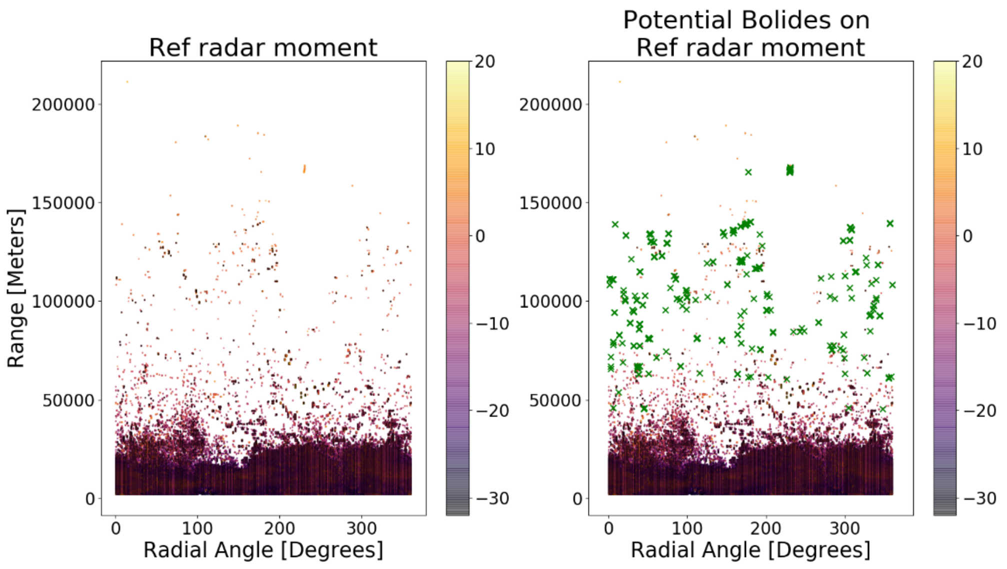


Figure 41. Final results for the KRGX Sutter's Mill Bolide with possible bolides in green Xs on the right

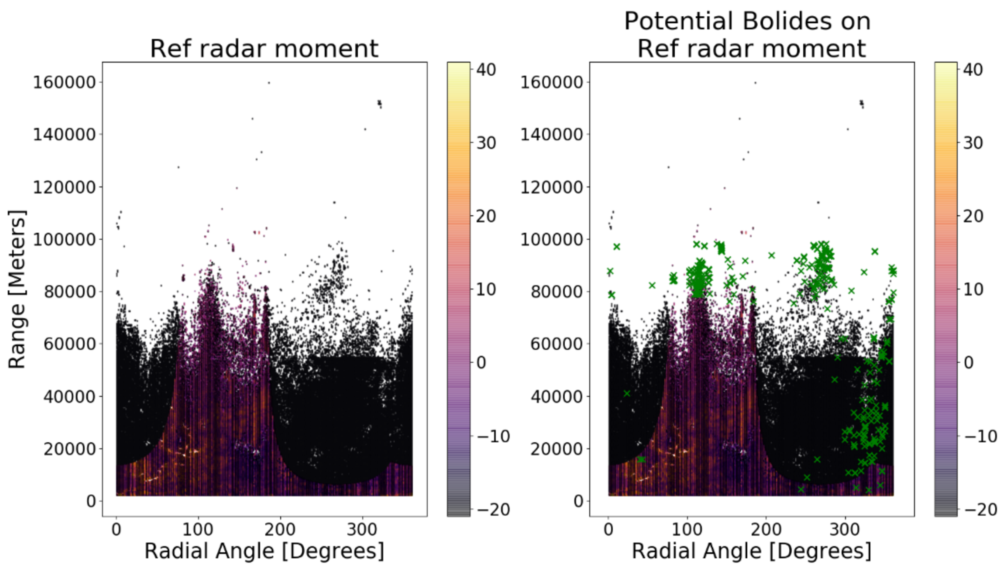


Figure 42. Final results for the KBBX Sutter's Mill Bolide with possible bolides in green Xs on the right

5. Limitations

Several shortcomings of the machine learning methods tried earlier in this thesis were improved by chaining the algorithms together and through reducing the dataset size. Despite the improvement, the process proposed in this research still has limitations. Firstly, although the execution time is orders of magnitude lower, it could still be improved by better streamlining the code and further dataset reductions. Secondly, the precision and recall rates are low, indicating that the number of true positives and false positives needs to be increased. Lastly and most importantly, the number of data points a human reviewer needs to verify as a bolide is still quite large. In each of the four cases, only 0.3% of the data remains for reviewing, but when the datasets themselves are upwards of 100,000 data points, even the distilled number of data points is significant. Therefore, although this algorithm is a good step in the correct direction, it can be iterated upon and improved to ease the burden on the human reviewer.

F. SUMMARY

This chapter applied multiple machine learning approaches to detect bolides within Doppler radar data. After creating a baseline from preprocessing the dataset, PCA was conducted and deemed helpful, but inadequate to detect bolide outliers. Next, k-means clustering was attempted and determined to be insufficient as well. Thirdly, t-SNE was found to be very helpful at taking higher dimensional data and displaying it in 2D or 3D, but came at such a computationally expensive cost that it was prohibitive to use. Therefore, a combined series of PCA and t-SNE, with density pruning through nearest neighbors analysis was created and found to be a viable method to identify bolide outliers.

VI. CONCLUSION AND FUTURE WORK

A. CONCLUSION

This thesis proposes a new approach for applying machine learning to the problem of meteor and bolide detection within Doppler radar data. This method supports the motivation of aiding the expedient recovery of meteorite samples in order to increase the knowledge of near-Earth asteroid physical properties and understand how these objects interact upon impacting Earth’s atmosphere. Such information will be invaluable for the advancement of planetary science and preparing future mitigation strategies to address the threat from potential Earth impacting objects. Doppler radar data is a comprehensive and ubiquitous source of meteor and bolide data, and this thesis developed a machine learning algorithm to filter through the large amounts of data to find the few outlier meteor and bolide signals.

Unsupervised methods were implemented as the means to search through the unlabeled radar data from four radar data sites for two bolide events: the KFWS radar for the Ash Creek bolide and the KDAX, KRGX and KBBX radars for the Sutter’s Mill bolide. PCA, k-means clustering, and t-SNE were investigated as specific algorithms to classify the outliers, but each on its own was determined to be inadequate. PCA failed to provide substantial separation between the bolide and non-bolide points. k-means clustering failed to categorize the radar return data points into bolide and non-bolide clusters. Lastly, t-SNE’s execution time was too long to fulfill the expedient nature of the project’s motivation and decreased the risk of acquiring suitable meteorite samples. However, a promising method was found by combining aspects of the PCA and t-SNE algorithms with a dataset reduction via NN Density Pruning. Through the PCA’s variance transformation method, the NNDP’s data reduction method, and the t-SNE’s dimensionality reduction method, outlier bolides could be filtered from non-bolide radar returns in reduced time.

The empirical results showed that 99.7% of the data returns were filtered in the machine learning algorithm, yielding an accuracy rate of 99.73% in a significantly reduced time of less than 8 minutes for a 121,553 return sized dataset (Sutter’s Mill, California bolide and the KBBX radar) and an accuracy of 99.67% in less than 1 minute for a 15,439 sized dataset (Ash Creek, Texas bolide and the KFWS radar). However, the algorithm still has room to improve, as the Recall and Precision rates remained low due to difficulties in correctly classifying True Positive bolides. Overall, the algorithm presented in this research is a viable method to help NASA scientists with bolide detection and meteorite recovery. Future work to refine the algorithm will enhance these efforts.

B. FUTURE WORK

There are several areas of future research that can be taken as a result of completing this research. They condense into three focus areas. The first is a series of ways to improve the accuracy of the algorithm, which would resolve the largest existing limitation from the current method's incarnation. The second is to improve the speed of the algorithm. The third and last is to use the algorithm's filtered bolides for additional and new avenues of research. Each has a great potential benefit that ties back to the original problem statement and motivation behind this research.

Improving the accuracy of the algorithm would significantly improve the results for NASA scientists. Using additional bolide events from different radars on different days would significantly help in this effort. Additional bolides would enable a characterization of the k-means distance that was relatively unhelpful due to only four data points, one for each of the four events analyzed in this research. It would also enable better estimation of calculating what the μ_{Upper} and μ_{Lower} bounds should be without overfitting the algorithm.

Next, additional features could be added to the dataset. This would increase the dimensionality of the dataset, but adding a few features should not change the execution time of the algorithm very much. However, it could potentially make bolides stand out through higher levels of variance. An example of a feature different from radar moment data would be seismic data, which has been proposed by Fries et al. [5].

Lastly, the accuracy could be improved by trying to re-order the execution of the algorithm in this research. The NN Density Pruning, PCA, and t-SNE algorithms are all proven helpful in this research, but investigating different execution sequences should be investigated. For instance, if the NN Density Pruning algorithm was first run on the radar moments plot, individual radar returns and large groups of returns from a cloud could be excluded from the dataset, immediately shrinking it down in size before it was transformed via PCA and pruned again in preparation for the t-SNE algorithm. Although not guaranteed to help, it is worthwhile to investigate how to increase the Precision and Recall rates.

The second focus area would be improving the speed of the algorithm. Density pruning helped tremendously in this regard, but additional speed gains would enable faster recovery in areas of adverse weather or large human populations. Reducing the amount of data points input into the t-SNE algorithm would help in this regard, but execution time is also wasted during the preprocessing phase. One method of speed reduction would be through an algorithm that better exported data from NOAA’s WCT program. For the current algorithm, it is manually exported, but this process could be scripted. Secondly, coordination with American Meteor Society and their Operations Manager, Mike Hankey, could enable using database management tools to more quickly query for radar return data as was already researched [29].

The final focus area would be on applying the output of this research to correlate bolide return strength to the original mass of the bolide. Some post recovery mass information is known from select few samples that have reached the surface of the Earth. Machine learning or statistical methods could be investigated to correlate the radar return strength to the mass of the final sample or samples like the work completed by Brown et al. [8]. This research would benefit from a source of verified bolides, so it is a natural follow on to the current research effort.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] Chodas, P., Khudiyan, S., and Chamberlin, A., 2019, "Center for Near Earth Object Studies: Discovery Statistics." Calif. Inst. Technol., Jet Propul. Lab. https://cneos.jpl.nasa.gov/stats/totals.html.

[2] Chodas, P., Khudiyan, S., and Chamberlin, A., 2019, "Center for Near Earth Object Studies: Fireballs." Calif. Inst. Technol., Jet Propul. Lab. https://cneos.jpl.nasa.gov/stats/totals.html.

[3] Durand-Manterola, H. J., and Cordero-Tercero, G., 2014, "Assessments of the Energy, Mass and Size of the Chicxulub Impactor."

[4] Fries, M., Matson, R., Schaefer, J., Fries, J., and Hankey, M., 2014, "Worldwide Meteorite Fall Recovery Using Weather Radars." Gold Schmidt 2014 Abstracts.

[5] Fries, M., Matson, R., Schaefer, J., Fries, J., Hankey, M., and Anderson, L., 2014, "Worldwide Weather Radar Imagery May Allow Substantial Increase in Meteorite Fall Recovery." Meteorit. Soc. Meet., pp. A124–A124.

[6] Walla, E., 2018, "Rapid Detection and Recovery: The Science of Hunting Meteorites." Univ. Ariz. News.

[7] Laird, C., Fries, M., and Matson, R., 2017, "A Method for Estimating Meteorite Fall Mass from Weather Radar Data." LPI, The Woodlands, Texas.

[8] Brown, P., Pack, D., Edwards, W. N., Revelle, D. O., Yoo, B. B., Spalding, R. E., and Tagliaferri, E., 2004, "The Orbit, Atmospheric Dynamics, and Initial Mass of the Park Forest Meteorite." Meteorit. Planet. Sci., 39(11), pp. 1781–1796.

[9] Daintith, J., and Gould, W., eds.: Bolide. Collins Dictionary of Astronomy.

[10] American Meteor Society. Home Page. https://www.amsmeteors.org/.

[11] Fries, M., Laird, C., Hankey, M., Fries, J., Maton, R., and Reddy, V., 2017, "Estimation of Meteorite Fall Mass and Other Properties from Weather Radar Data." LPI, Santa Fe, New Mexico, p. 6251.

[12] Fries, M., and Fries, J., 2010, "Doppler Weather Radar as a Meteorite Recovery Tool." Meteorit. Planet. Sci., 45(9), pp. 1476–1487.

[13] National Aeronautics and Space Administration, 2007, "Report to Congress: Near-Earth Object Survey and Deflection Analysis of Alternatives."

[14] Chodas, P., "NEO Basics. Calif. Inst. Technol., Jet Propul. Lab Center for Near Earth Object Studies." https://cneos.jpl.nasa.gov/about/basics.html.

[15] Ceplecha, Z., Borovička, J., Elford, W. G., ReVelle, D. O., Hawkes, R. L., Porubčan, V., and Šimek, M., 1998, “Meteor Phenomena and Bodies.” Space Science Review, 84(3/4), pp. 327–471.

[16] Silber, E. A., Boslough, M., Hocking, W. K., Gritsevich, M., and Whitaker, R. W., 2018, “Physics of Meteor Generated Shock Waves in the Earth’s Atmosphere – A Review.” Adv. Space Res., 62(3), pp. 489–532.

[17] Doviak, R. J., and Zrnić, D. S., 2006, Doppler Radar and Weather Observations. Dover Publications, Mineola, N.Y.

[18] Chrzanowski, E. J., 1990, Active Radar Electronic Countermeasures. Artech House, Norwood, MA.

[19] Russell, S. J., Norvig, P., and Davis, E., 2010, Artificial Intelligence: A Modern Approach. Prentice Hall, Upper Saddle River.

[20] Haykin, S. S., 1999, “Neural Networks: A Comprehensive Foundation. Prentice Hall, Upper Saddle River, N.J.

[21] Doviak, R. J., Zrnic, D. S., and Sirmans, D. S., 1979, “Doppler Weather Radar.” Proc. IEEE, 67(11), pp. 1522–1553.

[22] Schmidt, J. M., Flatau, P. J., Harasti, P. R., Yates, R. D., Littleton, R., Pritchard, M. S., Fischer, J. M., Fischer, E. J., Kohri, W. J., Vetter, J. R., Richman, S., Baranowski, D. B., Anderson, M. J., Fletcher, E., and Lando, D. W., 2012, “Radar Observations of Individual Rain Drops in the Free Atmosphere.” Proc. Natl. Acad. Sci., 109(24), pp. 9293–9298.

[23] Kent, B. M., Thomas, C., and Ryan, P., 2012, “The NASA Debris Radar for Characterizing Static and Dynamic Ascent Debris Events for Safety of Flight.” *Proceedings of the 2012 IEEE International Symposium on Antennas and Propagation*, IEEE, Chicago, IL, USA, pp. 1–2.

[24] Melnikov, V., Murnan, R., and Burgess, D., “Detecting and Tracking of Airborne Volcanic Ash with the WSR-88Ds, Task 8 of 2016 ROC MOU, NWS Radar Operations Center.”

[25] Marzano, F. S., Picciotti, E., Montopoli, M., and Vulpiani, G., 2013, “Inside Volcanic Clouds: Remote Sensing of Ash Plumes Using Microwave Weather Radars.” Bull. Am. Meteorol. Soc., 94(10), pp. 1567–1586.

[26] Marzano, F. S., Marchiotto, S., Textor, C., and Schneider, D. J., 2010, “Model-Based Weather Radar Remote Sensing of Explosive Volcanic Ash Eruption.” IEEE Trans. Geosci. Remote Sens., 48(10), pp. 3591–3607.

[27] Simon, S. B., Grossman, L., Clayton, R. N., Mayeda, T. K., Schwade, J. R., Sipiera, P. P., Wacker, J. F., and Wadhwa, M., 2004, “The Fall, Recovery, and Classification of the Park Forest Meteorite.” Meteorit. Planet. Sci., 39(4), pp. 625–634.

[28] Popova, O. P. and Coauthors, 2013, “Chelyabinsk Airburst, Damage Assessment, Meteorite Recovery, and Characterization.” Science, 342(6162), pp. 1069–1073.

[29] Hankey, M., Fries, M., Matson, R., and Fries, J., 2017, “AMSNEXRAD-Automated Detection of Meteorite Strewnfields in Doppler Weather Radar.” Planet. Space Sci., 143, pp. 199–202.

[30] Brown, P. and Coauthors, 2011, “The Fall of the Grimsby Meteorite-I, “Fireball Dynamics and Orbit from Radar, Video, and Infrasound Records, Grimsby Meteorite Fall.” Meteorit. Planet. Sci., 46(3), pp. 339–363.

[31] Fries, M., and Fries, J. “Partly Cloudy with a Chance of Chondrites - Studying Meteorite Falls Using Doppler Weather Radar.” 41st Lunar Planet. Sci. Conf. 2010.

[32] Fries, M., and Fries, J., 2010, “Doppler Weather Radar Observations of the 14 April 2010 Southwest Wisconsin Meteorite Fall.” 73rd Ann. Meteor. Soc. Meet.. 2010, 73.

[33] Fries, M., Fries, J., and Schaefer, J., 2011, “A Probable Unexplored Meteorite Fall Found in Archived Weather Radar Data.” 42nd Lunar Planet. Sci. Conf. 2011, 42.

[34] Fries, M., Fries, J., Hankey, M., and Matson, R., 2016, “Meteorite Falls Observed in U.S. Weather Radar Data in 2015 and 2016 (to Date).” 79th Ann. Meet. Meteor. Soc. 2016, 79.

[35] Fries, M., Matson, R., Fries, J., and Hankey, M., 2013, “Faster Recovery, Better Science: Meteorite Fall Events Detected with Weather Radars and Seismometers in 2012. 44th Lunar Planet. Sci. Conf. 2013, 44.

[36] Jenniskens, P. and Coauthors, 2012, “Radar-Enabled Recovery of the Sutter’s Mill Meteorite, a Carbonaceous Chondrite Regolith Breccia.” Science, 338(6114), pp. 1583–1587.

[37] Djorgovski, S. G., de Carvalho, R. R., Odewahn, S. C., Gal, R. R., Roden, J., Stolorz, P., and Gray, A., 1997, “Data Mining a Large Digital Sky Survey: From the Challenges to the Scientific Results.” A.G. Tescher, ed., San Diego, CA, pp. 98–109.

[38] Usama M. Fayyad, S. G. Djorgovski, and Nicholas Weir, 1996, “From Digitized Images to Online Catalogs Data Mining a Sky Survey.” AI Mag., 17(2).

[39] Galindo, Y., and Lorena, A. C., 2018, “Deep Transfer Learning for Meteor Detection. *Anais Do XV Encontro Nacional de Inteligência Artificial e Computacional (ENIAC 2018),* Sociedade Brasileira de Computação - SBC, São Paulo, pp. 528–537.

[40] Misra, A., and Bus, S. J., 2008, “Artificial Neural Network Classification of Asteroids in the Sloan Digital Sky Survey.”

[41] Smeresky, B. P., 2015, “Optimizing near Earth Object Classification Using Artificial Neural Networks and Evolutionary Methods.” Master’s Thesis, Florida Institute of Technology.

[42] Rumpf, C., Longenbaugh, R., Henze, C., Chavez, J., and Mathias, D., 2019, “An Algorithmic Approach for Detecting Bolides with the Geostationary Lightning Mapper.” Sensors, 19(5), p. 1008.

[43] U.S. Department of Commerce/NOAA, 2008, “Federal Meteorological Handbook No. 11: Doppler Radar Meteorological Observations, Part A: System Concepts, Responsibilities, and Procedures.”

[44] US Department of Commerce, NOAA, National Weather Service: About Our WSR 88-D Radar.” Accessed 21 August 2019. https://www.weather.gov/iwx/wsr_88d.

[45] Jaffe, B. M., 1973, “Forward Reflection of Light by a Moving Mirror.” Am. J. Phys., 41(4), pp. 577–578.

[46] American Meteorological Society, “Glossary of Meteorology.” Accessed: 09 October 2019. http://glossary.ametsoc.org/wiki/Moment.

[47] Binghamton, NY, Weather Forecast Office, and U.S. Department of Commerce, NOAA, National Weather Service, “WSR-88D Binghamton’s (KBGM) Upgrade to Dual Polarization. KBGM Upgrade Dual Polarization.” Accessed 14 August 2019. https://www.weather.gov/bgm/eventsDualPol.

[48] US Department of Commerce, NOAA, National Weather Service, J., Mississippi Weather Forecast Office, 2019, NWS Jackson Dual Polarization Upgrade Information, “What Is Dual-Pol Radar?” Accessed 21 August 2019. https://www.weather.gov/jan/dualpolupgrade-products.

[49] NOAA, “NEXRAD Data Inventory Search, National Centers for Environmental Information.” Accessed 13 June 2019. https://www.ncdc.noaa.gov/nexradinv/.

[50] Brett, M., 2019, “Python Software Foundation: NetCDF4 1.5.2. Python Softw. Found.” https://pypi.org/project/netCDF4/.

[51] McCulloch, W. S., and Pitts, W., 1943, “A Logical Calculus of the Ideas Immanent in Nervous Activity.” Bull. Math. Biophysics., 5(4), pp. 115–133.

[52] Kelley, H. J., 1960, “Gradient Theory of Optimal Flight Paths.” ARS J., 30(10), pp. 947–954.

[53] Rumelhart, D. E., Hinton, G. E., and Williams, R. J., 1986, “Learning Representations by Back-Propagating Errors.” Nature, 323, pp. 533–536.

[54] Mitchell, T. M., 1997: Machine Learning. McGraw-Hill, New York.

[55] Haykin, S. S., and Haykin, S. S., 2009, Neural Networks and Learning Machines. Prentice Hall, New York.

[56] Raschka, S., and Mirjalili, V., 2017, “Python Machine Learning: Machine Learning and Deep Learning with Python, Scikit-Learn, and TensorFlow. Packt Publishing, Birmingham Mumbai.

[57] van Rossum, G. Python Language Reference. Python Software Foundation, UK.

[58] Smith, M., 2017, “Python BeginnersGuide/Overview.” Accessed 12 Jun 2019. https://wiki.python.org/moin/BeginnersGuide/Overview.

[59] Python Softw. “Found: Python Package Index.” Available: https://pypi.org/.

[60] Buitinck, L. and Coauthors, 2013, “API Design for Machine Learning Software: Experiences from the Scikit-Learn Project. ArXiv: 1309.0238[cs].

[61] Hunter, J. D., 2007, “Matplotlib: A 2D Graphics Environment.” Comput. Sci. Eng., 9(3), pp. 90–95.

[62] Vanderplas, J., 2019, “Scikit-Learn: Nearest Neighbors Documentation.” Scikit-Learn. Accessed 14 August 2019. https://scikit-learn.org/stable/modules/neighbors.html.

[63] Anton, H., 1994, Elementary Linear Algebra. John Wiley, New York.

[64] Hastings, K. J., 1997, Probability and Statistics. Addison-Wesley, Reading, Mass.

[65] Vanderplas, J. T., 2016, “Python Data Science Handbook: Essential Tools for Working with Data.” O’Reilly Media, Inc, Sebastopol, CA.

[66] Scikit-learn developers, “Scikit Learn: Principal Component Analysis (PCA).” Scikit Learn. https://scikit-learn.org/stable/modules/decomposition.html#pca.

[67] Scikit-learn developers, “Scikit Learn Clustering and K-Means.” https://scikit-learn.org/stable/modules/clustering.html#k-means.

[68] Arthur, D., and Vassilvitskii, S., 2007, K-Means++: The Advantages of Careful Seeding. *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp. 1027–1035.

[69] Scikit-learn developers, "Scikit Learn Manifold Learning.' https://scikit-learn.org/ stable/modules/manifold.html.

[70] van der Maaten, L., and Hinton, G. E., 2008, "Visualizing Data Using T-SNE." J. Mach. Learn. Res. JMLR, 9(Nov).

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California