UNIVERSITY
OF OSLO

Master's thesis

A Novel Application of Machine Learning to Develop Pointing Models for Current and Future Radio/Sub-millimeter Telescopes

Bendik Nyheim

Computational Science: Physics
60 ECTS study points

Department of Physics
Faculty of Mathematics and Natural Sciences

Spring 2023

Bendik Nyheim

A Novel Application of Machine Learning to Develop Pointing Models for Current and Future Radio/Sub-millimeter Telescopes

Supervisors:
Signe Riemer Sørensen (Sintef)
Rodrigo Parrar (ESO)
Claudia Cicone (UiO)

Abstract

Radio/(sub)-mm telescopes are powerful instruments for studying the universe at radio, sub-millimeter, and millimeter wavelengths, but their accurate pointing is crucial for obtaining high-quality data. Pointing errors, which refer to the deviation of the telescope's orientation from its desired direction, can significantly impact the quality of the collected data. Radio/(sub)-mm telescopes use linear regression pointing models to correct for these errors, taking into account various factors such as weather conditions, telescope structure, and the target's position in the sky. However, residual pointing errors can still occur due to factors that are hard to model accurately, such as thermal and gravitational deformation and environmental conditions like humidity and wind. In this thesis, we investigated machine learning approaches to predict and correct residual pointing errors for the Atacama Pathfinder EXperiment (APEX) telescope, located in the high-altitude Atacama Desert in Chile. For our research, we used APEX telescope pointing data from 2022. We trained eXtreme Gradient Boosting (XGBoost) models that reduced azimuth and elevation (horizontal and vertical angle) root-mean-square errors by 7.0% and 9.4%, respectively, on an unseen test set. We also developed neural network pointing models from scratch that outperformed the commonly used linear regression model on a test case. Our study highlights the importance of larger datasets for accurate pointing models and the potential of machine learning to enhance the capabilities of current and future radio/(sub)-mm telescopes, such as the Atacama Large Aperture Submillimeter Telescope (AtLAST).

Acknowledgements

Thesis Structure

The thesis is structured into three parts, with the first chapter introducing the problem of pointing with radio telescopes. The first chapter also discusses the research questions, related works, and astronomy background.

Part I, Background & Theory, consists of two chapters. Chapter 2 provides a detailed description of the pointing model developed at APEX and the data we used for the research in this thesis. Chapter 3 presents the machine learning concepts we utilized.

Part II, Data Processing & Methods, also contains two chapters. Chapter 4 outlines the data processing and feature engineering techniques we used in the study. Chapter 5 describes two machine learning experiments that we aimed at answering the research questions posed in the introduction.

Finally, Part III, Results & Discussion, includes chapter 6 with the results and discussion of the two experiments, followed by the conclusion in chapter 7. In addition to the main body of the thesis, we provide an appendix, which includes other results from the research.

Contents

List of Figures

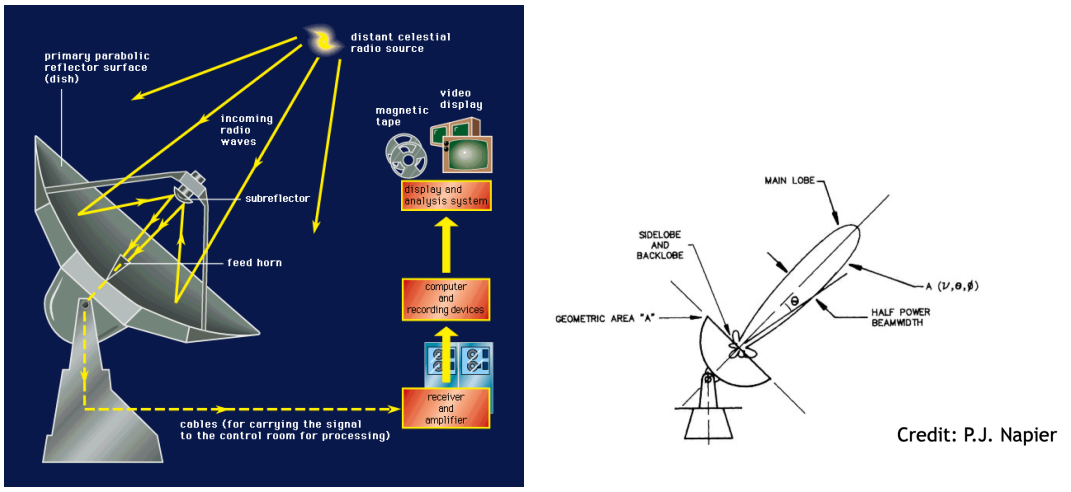
List of Tables

Chapter 1

Introduction

1.1 Introduction

Radio/(sub)-millimeter (mm) telescopes are powerful tools to study the Universe at radio, sub-millimeter, and millimeter wavelengths. These telescopes are designed to capture and detect electromagnetic radiation from space, which can provide valuable insights into a range of astronomical phenomena, from the formation of stars [13] and galaxies [3] to the behavior of black holes [5]. One of the key components of a radio telescope is its reflective surface, which collects and focuses the incoming radiation. Most radio/(sub)-mm telescopes have a large, parabolic dish-shaped primary mirror, reflecting incoming radiation onto a smaller, secondary mirror (see Figure 1.1a). The secondary mirror (or subreflector) then reflects the radiation onto a detector or receiver placed at the focal point, which records and processes the signals. In astronomy, the term ”pointing” refers to the process of positioning the telescope to observe an astronomical source. The goal is to fit the source at the center of the central resolution element or ”beam” of the telescope (see Figure 1.1b), which in the case of single-beam instruments corresponds to the field of view of the telescope.



(a) Radio telescope basics.

(b) Telescope beam sketch. Credit: P.J. Napier

Figure 1.1: **a)** Basics of how a radio telescope works and **b)** Sktech of a telescope's beam.

Radio/(sub)-mm telescopes detect and record photons over time, which are then processed to create a composite image or spectrum. However, this process requires highly accurate pointing, as even slight errors in the telescope's orientation can significantly affect the resulting data quality. Pointing errors, often referred to as pointing offsets, can be caused by various factors, including thermal deformation of the telescope components [10], gravitational deformation [27], and other environmental factors like humidity [6] and wind [11]. Most radio/(sub)-mm instruments do not have an imaging camera. Hence, the correct positioning of the source within the resolution element (beam) and at the center of the field of view cannot be checked directly. To achieve this accuracy, radio/(sub)-mm telescopes use pointing models [23], which take into account a range of factors that can contribute to the pointing error, including weather conditions, telescope structure, and the target's position in the sky.

The Atacama Pathfinder EXperiment (APEX) telescope, pictured in Figure 1.2[1], located in the high-altitude Atacama Desert in Chile, currently uses an effective analytical pointing model run in the background and recalibrated periodically through pointing measurement campaigns. However, there is still a residual pointing offset whose origin is not completely understood. This thesis aims to investigate two research questions. First, we want to explore using machine learning to increase the pointing accuracy of the current pointing model at APEX based on observational data such as weather patterns and telescope pointing. Secondly, we want to investigate a machine learning approach for developing a pointing model from scratch for a new radio/(sub)-mm single dish telescope. A machine learning approach would benefit larger radio/(sub)-mm telescopes like the future Atacama Large Aperture Submillimeter Telescope (AtLAST) [2]. AtLAST will have a 50-meter diameter primary mirror and 12-meter diameter subreflector. Hence, the subreflector will be as big as APEX's primary mirror. Due to the size and complexity of AtLAST, it will be harder to develop an analytical model. By developing a more advanced and reliable pointing model with machine learning, this research can enhance the capabilities of current and future radio/(sub)-mm telescopes to advance our understanding of the universe.

(a) Telescope structure. Credit: C. Cicone (2023)

(b) Subreflector. Credit: P. Gallardo (2023)

Figure 1.2: Pictures of the APEX telescope

[1]Link to APEX website http://www.apex-telescope.org/ns/

[2]Link to AtLAST website https://www.atlast.uio.no/

1.2 Related Works

The application of machine learning in astronomy has become increasingly popular in recent years, with various applications such as data analysis and prediction. For instance, Petrillo et al. [18] used two convolutional neural networks to detect gravitational lensing from images. George & Huerta [12] used a convolutional neural network to detect gravitational waves in real time at Laser Interferometer Gravitational-Wave Observatory(LIGO). Despite many use cases for machine learning in astronomy and the need for an accurate pointing model in radio telescopes, we have not found any studies that used machine learning to develop or maintain a pointing model for radio telescopes. Traditional methods for pointing models in radio telescopes involve modeling the pointing error as a function of various parameters, such as azimuth, elevation, temperature, and time. These models are often complex and require significant effort to develop and maintain. Moreover, they can be limited by the accuracy of the models used for atmospheric refraction, instrumental error, and other sources of noise. Several papers have described various approaches to improve the pointing accuracy of radio telescopes. For example, White et al. [28] developed a pointing model for the Green Bank Telescope using theoretical terms based on the telescope's structure and analysis on the thermal deformation of the telescope structure. Greve et al. [1] studied seasonal effects on the pointing. The use of machine learning for pointing models in radio telescopes poses several challenges and opportunities. One of the main challenges is the need for large datasets, which can be difficult to obtain in the context of radio telescopes. Large datasets require extensive observational "pointing" campaigns which take time and subtract it from astrophysical observations. Another challenge is to have access to such datasets (which are usually unpublished and used only internally by the staff at the observatories) and collect these datasets in a coherent way. Moreover, the accuracy of the pointing model depends on the accuracy of the data used for training, which can be affected by various sources of noise and error. Nonetheless, machine learning algorithms offer the potential for significant improvements in pointing accuracy, and can potentially reduce the complexity and maintenance requirements of traditional pointing models. Future research in this area could explore the development of machine learning algorithms that can handle the challenges unique to radio telescopes, and the integration of machine learning techniques into existing pointing models. Blakseth et al. [2] demonstrated a similar approach by combining physics-based modeling with data-driven techniques, such as deep neural networks, to learn a corrective source term in numerical experiments on one-dimensional heat diffusion. This approach could potentially be adapted for a pointing model, where a hybrid model that combines theoretical and empirically-based corrective terms commonly used in pointing models today with sensor data could result in a more accurate and robust pointing model. Such a model could require less maintenance and development time, which would benefit science operations.

1.3 Astronomy Background

In order to understand some of the parameters used in the pointing models, we need to introduce some basic concepts concerning astronomical coordinates

1.3.1 The Celestial Sphere

The celestial sphere is a fundamental concept in astronomy. It is an imaginary sphere with an arbitrary radius centered on Earth, and it allows us to represent the positions of celestial objects conveniently and intuitively. Any astronomical observation is a 2D projection onto the celestial sphere, a tool astronomers use to specify the position of a target as it appears in the sky without using its physical distance from Earth (which requires a deeper knowledge of the physical properties of the astronomical target, usually acquired after many different observations). We describe the position as two-dimensional angular coordinates on the sphere. While the celestial sphere is a universal concept, the coordinate system used to specify the location of a target can vary.

1.3.2 Altitude-Azimuth Coordinate System

Figure 1.3 depicts the altitude-azimuth coordinate system, which depends on the observatory's position on Earth and is commonly used when performing astronomical observations (but rarely used in scientific publications, which instead use a universal coordinate system). This system specifies the angular coordinates (e.g. in degrees or arcminutes, which are 1/60 of a degree, or arcseconds which are 1/60 of an arcmin) using an azimuth and an altitude (or elevation) angle. Azimuth is the angle around the axis perpendicular to the horizontal plane, with zero degrees corresponding to due north. At APEX, the convention is to increase the azimuth angle in a clockwise direction. The interval for azimuth angles is $[-270°, 270°]$ due to APEX's ability to rotate one and a half times around its axis in the horizontal plane. On the other hand, elevation is the angle perpendicular to the horizontal plane, with zero degrees corresponding to the telescope pointing at the horizon and $90°$ to the telescope pointing at the zenith directly above it. In this thesis, we used elevation instead of altitude to describe this coordinate.

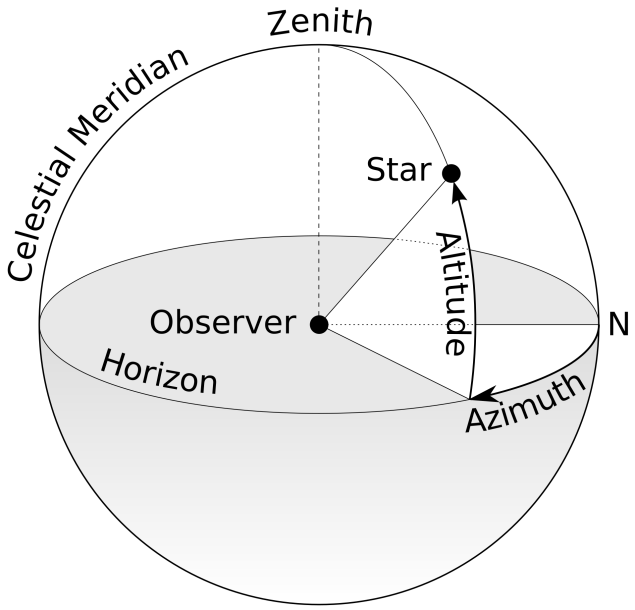


Figure 1.3: The altitude-azimuth coordinate system used at APEX. Source [25]

1.3.3 Radio/(Sub)-mm Telescope Basics

In this section, we present an overview of the fundamental components of a radio/(sub)-mm telescope. As illustrated in Figure 1.1a, the basic operation of a Nasmyth-Cassegrain telescope design involves collecting electromagnetic radiation with a primary mirror, which is then reflected onto a secondary mirror (subreflector). The photons are subsequently redirected toward the interior of the telescope, where they are converted into an electrical signal by an attached receiver (instrument). The electric signal is then processed and recorded for further analysis.

Additionally, Figure 1.1b provides a visual representation of the telescope's beam. The telescope's beam indicates the source of the strongest signal detected by the telescope. The beam power weakens towards the edges, and the goal is to place the source as the peak of the telescope's response, i.e., at the center of the beam The angular resolution of a telescope, or the solid angle it can observe, is determined by the width of its beam. Physical characteristics, including the primary dish diameter and the electromagnetic radiation frequency, determine the beam's width. The standard measure for beamwidth is the half-power beamwidth or the angle between two points on the beam where the radiation power is half its maximum.

Part I

Background and Theory

Chapter 2

APEX Pointing Model and Database

2.1 Pointing model

Since radio/(sub)-mm telescopes observe over an extended time, they need a pointing model to obtain sufficiently accurate pointings. The flux of the brightest radio sources is also weaker than the atmospheric emission, which means that the signal is often hidden in the noise and needs to be extracted using long integrations and modulation techniques. Therefore, astronomers must know that the pointing is accurate before initiating a long integration on the source.

The pointing model at APEX consists of two steps, an analytical model and additional pointing corrections performed at regular intervals based on recently observed pointing offsets. The analytical model consists of fitting multiple terms to the many measurements of the pointing offset (difference between input coordinates and the observed coordinates of the source). These terms can be geometric terms or terms related to, for example, metrology data. The fitted terms are used for 1-2 months and run in the background adjusting all input coordinates. The additional pointing corrections are performed by astronomers every 1-2 hours during the science observations and before observing a new target.

These equations explain the resulting pointing

$$Az = Az_{\text{input}} + \Delta Az_{\text{analytical model}} + \Delta Az_{\text{correction}} \tag{2.1}$$

$$El = El_{\text{input}} + \Delta El_{\text{analytical model}} + \Delta El_{\text{correction}} \tag{2.2}$$

Where the first terms, Az_{input} and El_{input}, are the input coordinates. The second terms $\Delta Az_{\text{analytical model}}$ and $\Delta El_{\text{analytical model}}$ are the adjustment made according to the analytical model. Furthermore, the last terms, $\Delta Az_{\text{correction}}$ and $\Delta El_{\text{correction}}$, are the corrections based on recently measured pointing offsets.

In the following section, we introduce and explain the adjustments from the analytical model and pointing offsets.

2.1.1 Analytical Model

Accurately measuring pointing offsets without a pointing model can be challenging as the error is typically larger than the beamwidth, causing the source to fall outside the beam. At APEX, astronomers use an optical receiver mounted in the primary mirror to make the initial observations, which allows them to observe the source in

real-time. During this process, which the astronomers perform periodically every 1-2 months, the telescope is pointed at various sources with known locations, yielding both input and observed coordinates.

The analytical pointing model at APEX considers various factors that affect pointing, including purely geometrical terms based on the imperfect mounting of telescope components and empirical terms. It uses the terms described below, all of which are dependent on the azimuth Az or elevation El, except for a couple of constant terms. The coefficients for all the terms are determined by the `TPOINT` [26] software, using a linear fit based on the observed offsets from a pointing campaign. The sum of all terms is the adjustment made by the model.

Most of the terms described in this section are fitted on data collected from the optical receiver mounted in the primary mirror. Then, the astronomers refine the terms using observations from different instruments to develop specialized pointing models for each, while most terms remain constant from the optical fit. The analytical model is crucial in accurately determining the telescope's pointing offsets, essential in obtaining high-quality observational data. Table 2.1 provides a list of the terms utilized in the analytical model, indicating whether each term is determined through optical data or recalibrated for individual radio receivers.

The following descriptions of the terms are taken directly from the `TPOINT` software manual [26].

Harmonic terms

The analytical model has multiple harmonic terms, some geometrical and some empirical. We introduce all the purely geometrical terms below, being AN, AW, CA, IA, IE, and NPAE, in addition to NRX, which is not purely geometrical but still affects pointing accuracy. The `TPOINT` software that the APEX staff uses to develop the analytical model suggests terms that improve the model's performance on the chosen dataset. The following terms are the empirical terms for azimuth.

$$\Delta Az = c_1 \cdot \sin Az + c_2 \cdot \frac{\cos 2Az}{\cos El} + c_3 \cdot \cos 3Az + c_4 \cdot \sin 2Az \tag{2.3}$$

$$+ c_5 \cdot \cos 2Az + c_6 \cdot \frac{\cos Az}{\cos El} + c_7 \cdot \frac{\cos 5Az}{\cos El}, \tag{2.4}$$

and the terms for elevation are

$$\Delta El = c_1 \cdot \sin El + c_2 \cdot \cos El + c_3 \cdot \cos 2Az + c_4 \cdot \sin 2Az \tag{2.5}$$

$$+ c_5 \cdot \cos 3Az + c_6 \cdot \sin 3Az + c_7 \cdot \sin 4Az + c_8 \cdot \sin 5Az \tag{2.6}$$

The `TPOINT` software denotes the harmonic terms in the format $Hrfci$. The list below explains the different terms.

- H: Stands for harmonics
- r: The resulting variable, either Az or El, denoting azimuth and elevation respectively. The resulting variable can also be S, which means the result is horizontal, or azimuth scaled by a factor $1/\cos El$.
- f: The harmonic function, either S or C denoting *sine* and *cosine*.
- c: The variable that the funciton f is dependent on, either Az or El.

- i: Integer value in the range 0-9, denoting the frequency of the harmonic.

For example, is $\Delta Az = \text{HACA3}\cos 3Az$ denoted as HACA3 in the `TPOINT` software.

Az/El non-perpendicularity (NPAE)

In an altazimuth mount, if the azimuth axis and elevation axis are not exactly at right angles, horizontal shifts proportional to $\sin El$ occur. This effect is zero when pointing at the horizon and increases with elevation proportional to $1/\cos El$

$$\Delta Az \simeq -\text{NPAE}\ \frac{\sin El}{\cos El} = -\text{NPAE}\ \tan El, \tag{2.7}$$

where NPAE is the horizontal displacement when pointing at Zenith.

Left-right collimation error (CA)

In an altazimuth mount, the collimation error is the non-perpendicularity between the nominated pointing direction and the elevation axis. It produces a horizontal image shift given by

$$\Delta Az \simeq -\text{CA}/\cos El \tag{2.8}$$

Azimuth and elevation index error (IA/IE)

Index errors are the errors when pointing at origo.

The azimuth index error is

$$\Delta Az = -\text{IA}, \tag{2.9}$$

and elevation index error is

$$\Delta El = \text{IE} \tag{2.10}$$

Azimuth axis misalignment (AN/AW)

In an altazimuth mount, misalignment of the azimuth axis north-south or east-west causes errors. The errors caused by misalignment in the north-south are given by

$$\Delta Az \simeq -\text{AN}\sin Az \cdot \tan El, \tag{2.11}$$

and

$$\Delta El \simeq -\text{AN}\cos Az, \tag{2.12}$$

where AN is the misalignment alignment in the north-south direction. The errors given by misalignment in east-west are given by

$$\Delta Az \simeq -\text{AW}\cos Az \tan El, \tag{2.13}$$

and

$$\Delta El \simeq \text{AW}\sin Az, \tag{2.14}$$

where AW is the misalignment alignment in the east-west direction.

Horizontal displacement of Nasmyth rotator (NRX)

In a Nasmyth altazimuth mount, a horizontal displacement between the elevation axis of the mount and the rotation axis of the Nasmyth instrument-rotator produces and image shift on the sky with a horizontal component

$$\Delta Az \simeq -\mathrm{NRX}, \tag{2.15}$$

and an elevation component

$$\Delta El \simeq -\mathrm{NRX} \sin El, \tag{2.16}$$

where NRX is the horizontal displacement.

Table 2.1: List of terms included in the analytical pointing model at APEX. All the terms are fitted using observations from the optical receiver, while the terms below "Radio receiver" are refitted specifically for each radio receiver.

Term	**Azimuth**	**Elevation**
Optical receiver		
NPAE	x	
HASA	x	
HECA2		x
HSCA2	x	
HACA3	x	
HASA2	x	
HACA2	x	
HSCA	x	
HESA2		x
HECA3		x
HSCA5	x	
HESA3		x
HESA4		x
HESA5		x
NRX	x	x
NRY	x	x
Radio receivers		
HESE		x
HECE		x
IA	x	
IE		x
CA	x	
AN	x	x
AW	x	x

2.1.2 Pointing Corrections

Although the analytical linear regression pointing model at APEX fits pointing campaign observations well, it cannot accurately model the changes in pointing over time, resulting in residual errors. Astronomers regularly correct these errors by observing a strong source with known coordinates, measuring the pointing offset,

and updating the model's terms for azimuth and elevation correction (CA and IE, respectively). The update procedure is as follows:

$$\mathrm{CA} = \mathrm{CA} + \delta_{\mathrm{Az}} \tag{2.17}$$

$$\mathrm{IE} = \mathrm{IE} - \delta_{\mathrm{El}}, \tag{2.18}$$

where δ_{Az} and δ_{El} are the recently observed pointing offsets in azimuth and elevation, respectively. The astronomers perform these pointing corrections every couple of hours to ensure the pointing is sufficient during science observations.

Note that we divide the term CA (2.8) by cosine elevation, which converts the observed horizontal offset to azimuth.

2.2 APEX Database

In this section, we introduce the different data sources utilized throughout the research in this thesis.

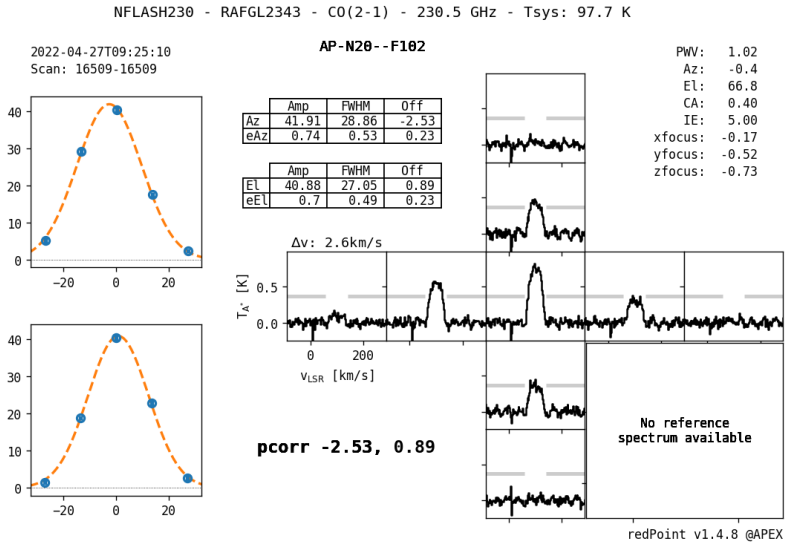
2.2.1 Pointing Scan Data

During a pointing scan, the telescope observes a source with a known location to obtain a pointing offset measured in arcseconds. The observers use this offset to correct the pointing model (using equations (2.17) and (2.18)). There are two types of pointing scans: Line-pointings and continuum scans.

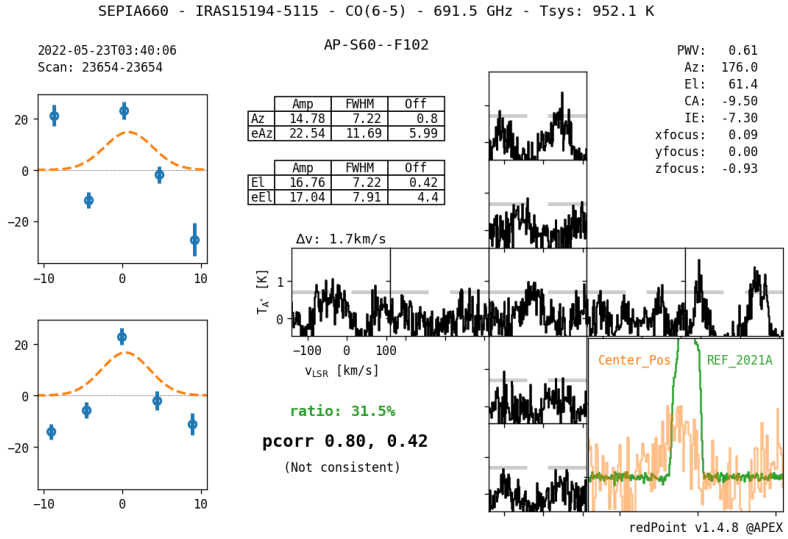
Line-pointing

A line-pointing involves pointing at an extended source. The telescope then makes ten scans, recording the flux intensity from the source, five vertically and five horizontally, around the center of the pointing, as shown in figure 2.1. The upper panel shows a high-quality pointing scan, and the lower panel shows a noisy, low-quality pointing scan. The cross-plot on the right side shows the line spectrum for each observation (center plus eight offset observations).

The integrals of the flux recorded from the source are plotted as blue dots on the left-hand side of the panel. A Gaussian is fitted to these points, and the table shows the resulting amplitude, full width at half maximum (FWHM), and offsets.



(a) Line-pointing with little noise and a good Gaussian fit.

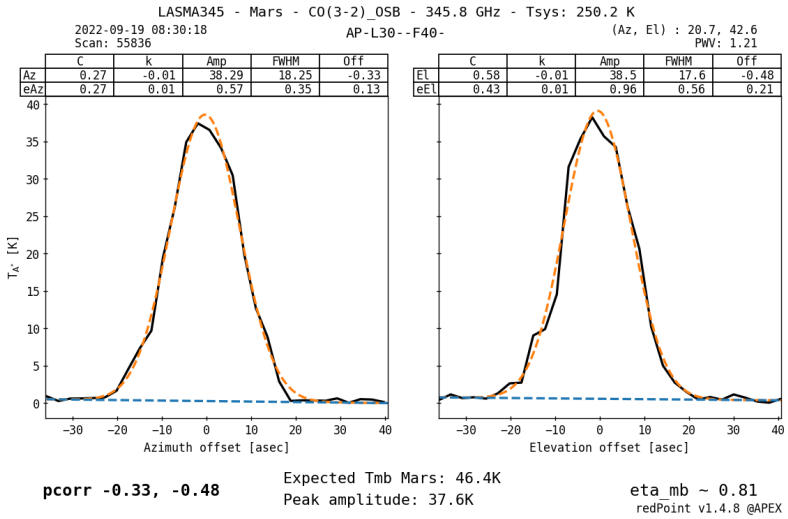


(b) Noisy line-pointing with bad Gaussian fit.

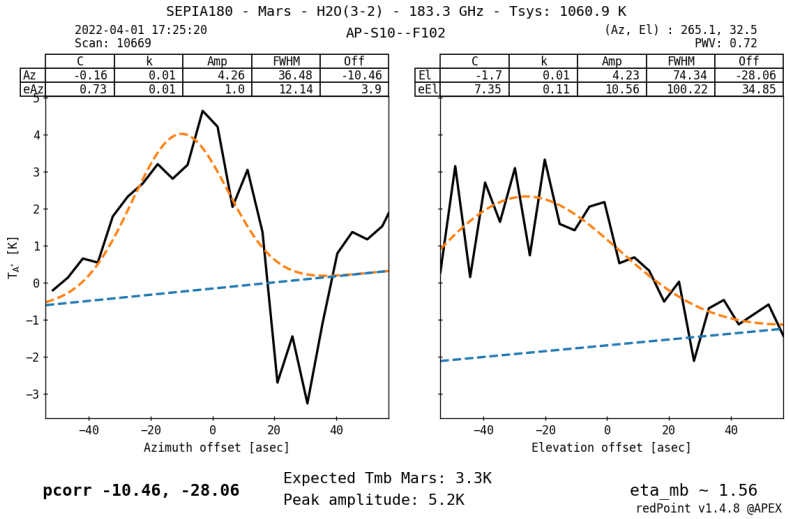
Figure 2.1: The two figures show line-pointing scans. a) is good and clean, and b) is noisy and unreliable. A Gaussian is fit both for the azimuth and elevation pointing. The table shows the amplitude, full width at half maximum (FWHM), offset, and the uncertainty of these measures, for azimuth and elevation. The figures also show the correction applied during the pointing (*ca* and *ie*), along with other metrics.

Continuum Scan

Not all sources have emission lines; for these sources, the telescope performs a continuum scan instead. In this case, a source is continuously scanned in azimuth and elevation while recording the flux intensity. A Gaussian curve is fitted to the recorded flux intensity to determine the offsets, amplitude, and full width at half maximum (FWHM). Figure 2.2 show examples of continuum scans and the corresponding Gaussian fits.



(a) Line-pointing with little noise and a good Gaussian fit.



(b) Noisy line-pointing with bad Gaussian fit.

Figure 2.2: The two panels show continuum pointing scans. a) is good and clean, and b) is noisy and unreliable. A Gaussian is fit both for the azimuth and elevation pointing. The amplitude, full width at half maximum, offset, and the uncertainty of these measures are shown for both of the fits.

Pointing scan timestamp

In the main database, each pointing scan has a timestamp in the format YYYY-MM-DD HH:MM:SS, with a one-second resolution. This timestamp does not reflect the actual start of a pointing scan. Also, no information in the database itself indicates whether the telescope is observing, but this information can be extracted from some dump files from the tiltmeter, which includes a flag indicating whether the telescope is idle, preparing to observe, or observing. Combining this flag with the timestamps, we can obtain the accurate start and end time of a pointing scan. However, these tiltmeter dump files are only available for some time periods.

Instruments

The observing instruments on the telescope operate at various frequencies. Table 2.2 provides information on the frequency range covered by each instrument, along with the number of scans performed using each instrument throughout the year 2022.

The broad range of frequencies at which astronomical phenomena emit electromagnetic radiation requires observation across a wide range of frequencies to study these phenomena comprehensively. APEX's website provides a complete list of instruments along with their descriptions.

Table 2.2: The number of times each instrument was used for a pointing scan in 2022. There are 8847 scans in total.

Instrument	Frequency band [GHz]	# of scans
NFLASH230	200-270	3197
LASMA345	268-375	1861
NFLASH460	385-500	1394
SEPIA660	578-738	856
SEPIA345	272-376	818
SEPIA180	159-211	359
HOLO	-	225
ZEUS2	-	103
CHAMP690	-	34

2.2.2 The Monitor Database

The monitor database is critical in this project, providing valuable sensory data from within and outside the telescope. In this section, we will explore the data contained within the monitor database and identify the most relevant variables to our purposes. We had a copy of the database containing data from 01.01.2022 to 17.09.2022.

The monitor database provides data with varying frequencies, as shown in Table 2.3, which lists the approximate number of data points per minute for each table used in this project. Figure 2.4 presents scatter plots of a selected subset of variables from the database during pointing scans.

Azimuth and Elevation

The database includes tables for the input azimuth and elevation, labeled COMMANDAZ and COMMANDEL. These tables contain the raw coordinates before the pointing model has adjusted the pointing.

The database also includes tables for the actual azimuth and elevation, labeled ACTUALAZ and ACTUALEL. These tables contain the coordinates obtained after applying the pointing model and automatic adjustments based on sensory data.

Finally, the database contains tables for the azimuth and elevation velocity, labeled ACTUALVELOCITYAZ and ACTUALVELOCITYEL. These tables provide information on the velocity of the telescope during observations.

The frequency of these measurements is 6 data points per minute. Figure 2.4a show these measurements for the duration of a pointing scan, along with additional data points before and after the scan.

Temperature Measurements

Multiple instruments located at different locations on the telescope measure the temperature and store the measurements in the database. The tables that contain these measurements are labeled TEMPERATURE, TEMP1 through TEMP6, TEMP26 through TEMP28, and TILT1T. Figure 2.3 indicates that many of these measurements are highly correlated. For example, TEMP1 through TEMP6 show a strong correlation ≥ 0.98. Similarly, TEMP26 through TEMP28 and TEMPERATURE are also highly correlated. The frequencies of some of these measurements are different, and they may all be found in Table 2.3. Figure 2.4c and 2.4d show the measurements of TEMP1 and TILT1T respectively for the duration of a pointing scan and additional data points before and after the scan.

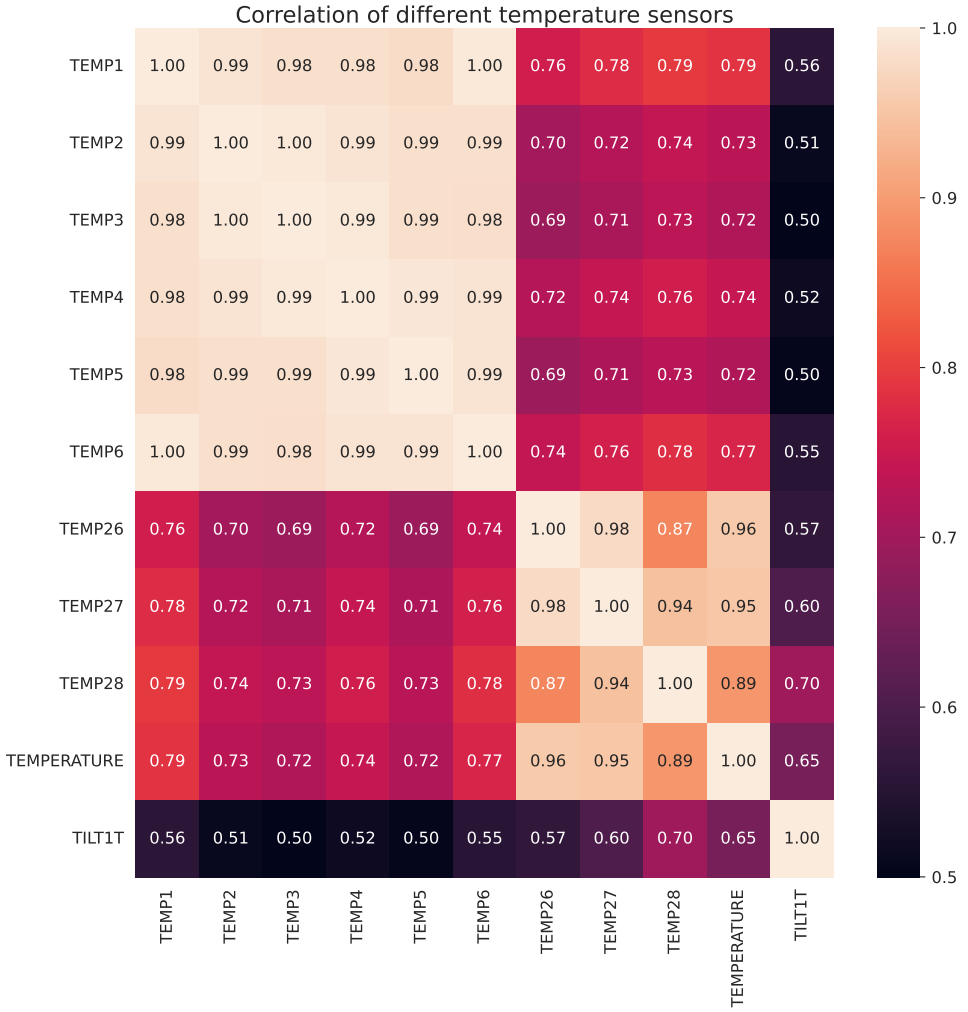


Figure 2.3: Linear correlation between temperature measurements at APEX. The values are sampled by the median value at each pointing scan.

Hexapod

The secondary mirror, also known as the subreflector, is supported by a hexapod. The hexapod moves in three dimensions and rotates around azimuth and elevation axes. There are five measures associated with the hexapod: POSITIONX, POSITIONY, POSITIONZ, ROTATIONX, and ROTATIONY. These measures are essential for

positioning the secondary mirror and ensuring accurate pointing. The frequency of these measurements is 6 data points per minute.

Tiltmeter

The telescope has two tiltmeters that measure its tilt or inclination to the vertical direction. One tiltmeter aligns with the telescope's pointing (TILT1X), while the other is orthogonal to the pointing (TILT1Y). The frequency of these measurements is 12 data points per minute.

Weather data

The weather station at the telescope provides measurements of various weather parameters, including dew point, humidity, pressure, wind speed, and wind direction. The instruments take measurements at a frequency of 5 data points per minute. The figures 2.4e and 2.4f show wind direction and speed measurements for the time period around a pointing scan.

Disp abs?

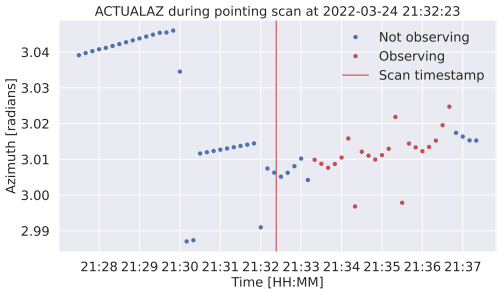
Frequency of 12 data points per minute.

Automatic adjustments

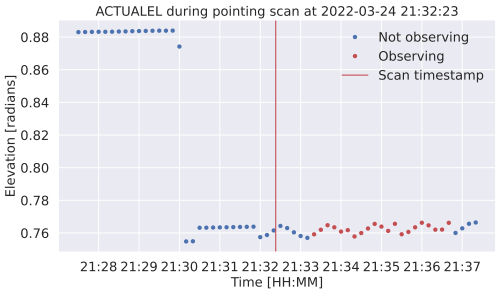
Automatic adjustments based on readings from various sensors ensure accurate and stable pointing of the telescope. These adjustments account for previously modeled systematic errors based on measurements from tiltmeters, temperature sensors installed at different locations, and other relevant data sources. Some system at the telescope automatically makes these adjustments, and the tables in the database that contain information about these adjustments start with DAZ or DEL, denoting adjustments in azimuth and elevation, respectively. The frequency of this data is 12 data points per minute. Table 2.3 shows a comprehensive list of these variables.

Table 2.3: The frequency in data points per minute of different variables in the monitor database.

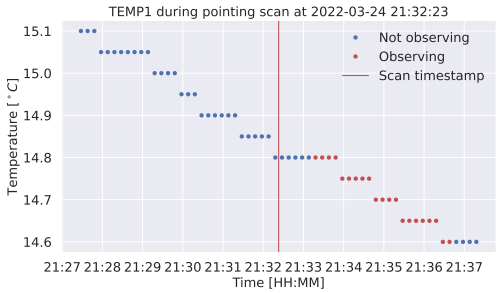
Table	Frequency [datapoints/minute]
ACTUALAZ	6
ACTUALEL	6
ACTUALVELOCITYAZ	6
ACTUALVELOCITYEL	6
COMMANDEL	6
COMMANDAZ	6
TILT1X	12
TILT2Y	12
TILT1T	12
TEMPERATURE	5
TEMP1	6
TEMP2	6
TEMP3	6
TEMP4	6
TEMP5	6
TEMP6	6
TEMP26	2
TEMP27	2
TEMP28	2
DAZ_TEMP	12
DAZ_TILT	12
DAZ_TILTTEMP	12
DAZ_SPEM	12
DAZ_DISP	12
DAZ_TOTAL	12
DEL_TEMP	12
DEL_TILT	12
DEL_TILTTEMP	12
DEL_SPEM	12
DEL_DISP	12
DEL_TOTAL	12
POSITIONX	6
POSITIONY	6
POSITIONZ	6
ROTATIONX	6
ROTATIONY	6
DISP_ABS1	12
DISP_ABS3	12
DISP_ABS2	12
DEWPOINT	5
PRESSURE	5
HUMIDITY	5
WINDSPEED	5
WINDDIRECTION	5



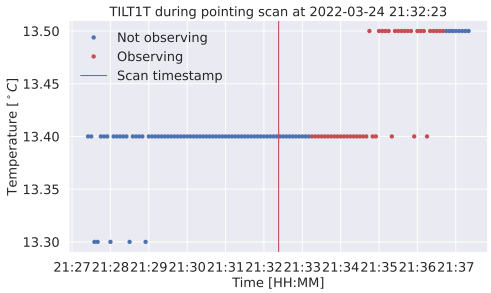
(a) Azimuth angle



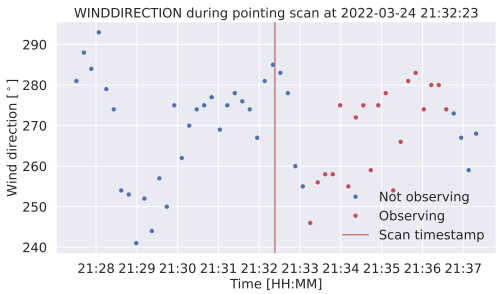
(b) Elevation angle.



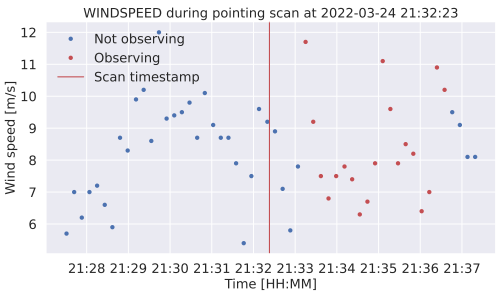
(c) Temperature measurements at temperature sensor 1.



(d) Temperature measurements at tiltmeter 1.



(e) Wind direction data from the weather station, measured in degrees from North, where clockwise is the positive angle direction.



(f) Wind speed data from the weather station.

Figure 2.4: Scatter plots that show different sensory data from before, during, and after a pointing scan. The red line denotes the timestamp for a scan in the pointing scan database. The red dots indicate when the telescope is observing, while the blue dots indicate when the telescope is idle or preparing to observe.

2.2.3 Raw Data

The raw data from the pointing scans using the NFLASH230 receiver provides input and actual coordinates. APEX staff has obtained the actual coordinates of the sources by combining the input coordinates with the adjustments made by the pointing model, automatic adjustments based on sensory data, and the observed offset. Then, they use this raw data to refine the model fit on data obtained from the optical receiver. Specifically, this data refines the model of the NFLASH230 model. Table 2.4 is included to provide an example of this data format.

Table 2.4: Extract of raw data obtained with NFLASH230. The data file also includes the source, which is irrelevant to this project.

	Input		Observed	
Date	Azimuth	Elevation	Azimuth	Elevation
2022-01-03 14:24:04	189.812879	41.0762	190.254779	40.883651
2022-01-03 18:59:40	50.842145	73.371647	51.269044	73.203243
2022-01-03 19:01:49	49.555916	73.752182	49.983112	73.583545
2022-01-03 19:16:10	39.378382	76.076236	39.781084	75.908956
2022-01-03 19:18:27	113.934309	39.345667	114.391232	39.170168
2022-01-22 13:54:31	94.04365	18.148405	94.492505	17.981161
2022-01-22 14:15:35	148.569964	89.044036	147.783271	88.852306
2022-01-22 14:18:15	215.664924	49.563821	216.104389	49.386438

2.2.4 Tiltmeter Dump Files

The tiltmeter dump files are a small part of the database and are only used to analyze when pointing scans start and end, using a ”scan flag”. In the pointing scan database, there is a timestamp linked to each pointing scan, but this timestamp does not mark the start of the scan, but rather some time right before it. The tiltmeter dump files contain a flag indicating whether the telescope is idle, preparing to observe, or observing. There are 280 of these files, and all have filenames in the format ”Tiltmeter_YYYY-MM-DD.dump,” which indicates the data's date. These files contain seven columns: datetime, azimuth, elevation, tilt1x, tilt1y, tilt1t, and the scan flag. For our purpose, only the datetime and scan flags provide useful information. Table 2.5 shows an extract of the datetime and scan flag columns from one of the tiltmeter dumps. Analysis of this data is presented in section 4.2.

Table 2.5: Extract from a tiltmeter dump file. It shows the timetamp, and a variable denoting if the telescope is a) idle, b) preparing to observer, or c) observing.

Datetime	Scan flag
2022-11-13T02:23:37	IDLE
2022-11-13T02:23:38	IDLE
2022-11-13T02:23:39	PREPARING
2022-11-13T02:23:40	PREPARING
⋮	⋮
2022-11-13T02:23:52	PREPARING
2022-11-13T02:23:53	PREPARING
2022-11-13T02:23:55	OBSERVING
2022-11-13T02:23:56	OBSERVING
2022-11-13T02:23:57	OBSERVING

Chapter 3

Machine Learning Theory

In this section, we present most of the machine learning concept utilized in our research. For more details on the theory in sections 3.1 through 3.6, see [15].

3.1 Supervised Learning

Supervised learning is a subfield of machine learning that refers to training a model to predict a specific target value based on input data. In this context, we refer to the input data as "features." The training is supervised when paired with the corresponding target value for prediction. There are two types of supervised learning, regression and classification. In regression, we predict a continuous variable, while in classification predict a binary value, true or false. The model architecture of the model can be the same regardless of predicting a true/false or continuous value. The difference is in the loss function, which is used to evaluate the model's performance during training. The last layer activation function for neural networks is different for regression and classification. In-depth explanations of this will come in the following sections.

3.2 Loss/Cost

In machine learning, the loss of a model refers to the discrepancy between the predicted and true values. It is calculated using a specific function designed to penalize incorrect predictions and measure the model's performance. The ultimate goal of any machine learning model is to minimize the loss and thereby reduce the difference between predicted and desired outputs. To achieve this, the model is trained by calculating the gradient of the loss function with respect to different components in the model. These gradients determine how the model is adjusted to minimize the loss through an iterative process. As a result, the model is optimized to make better predictions and achieve higher accuracy.

The most common loss function for regression is the mean squared error

$$\mathcal{L}(y, \tilde{y}) = \frac{1}{N} \sum_{i=1}^{N} (y - \tilde{y})^2, \tag{3.1}$$

where $\tilde{y}$ is the prediction, y the true value, and N the number of predictions.

3.2.1 Loss Functions

Loss functions are used to evaluate the performance of the machine learning model during training. We consider two different loss functions when predicting azimuth and elevation simultaneously with the same model, such as a neural network. One loss function considers the offset in azimuth and elevation separately, and one considers the total distance. Let $\tilde{y}_{Az}$ and $\tilde{y}_{El}$ denote the prediction for the offset in azimuth and elevation, respectively. y_{Az} and y_{El} are the true values. The first loss function is the mean squared error

$$\mathcal{L}_{\text{MSE}} = \frac{1}{2N} \sum_{i}^{N} \left((y_{Az,i} - \tilde{y}_{Az,i})^2 + (y_{El,i} - \tilde{y}_{El,i})^2 \right), \tag{3.2}$$

where N is the number of predictions.
For the second loss function, we use the mean squared distance

$$\mathcal{L}_{\text{MSD}} = \frac{1}{N} \sum_{i}^{N} \left[(y_{Az,i} - \tilde{y}_{Az,i})^2 + (y_{El,i} - \tilde{y}_{El,i})^2 \right], \tag{3.3}$$

It is difficult to predict the effects of these loss functions if any at all, but one difference could be that $\mathcal{L}_{\text{MSE}}$ is more sensitive to outliers, and $\mathcal{L}_{\text{MSD}}$ reduces the offsets more evenly.

For models with a single output azimuth or elevation, we use the regular mean squared error (3.1)

3.3 Datasplitting

Machine learning models can be highly complex and fit all the data points in a dataset. While this can result in perfect predictions on the training data, it often leads to poor performance on new data, a phenomenon known as overfitting. To counteract this, the data is typically split into two parts - a training set and a validation set. The model is trained on the training set, and the error on the validation set is used to evaluate the model's performance. By using a separate set of data for validation, we can better estimate the model's performance on new data and avoid overfitting.

When the error on the training data is low, the model has low bias. However, if the model is too complex, it may also have high variance, meaning that it is overly sensitive to the training data and unable to generalize well to new data. The key to building a good model is to balance bias and variance and to find the right level of complexity that will allow the model to generalize well. Proper selection of the train/test split ratio and other techniques, such as regularization, can help achieve this balance and improve the model's performance.

One usually picks the machine learning model with the best performance on the validation set, but this performance is not a reasonable estimate of the expected performance on future predictions. This is because multiple models are typically trained, and the model with the best performance on the validation set may have obtained its results by chance. Therefore, a third test set is used to get an unbiased estimate of the model's performance. The data in the test set is not used when training or validating and is only used to estimate the final model's performance.

3.4 Scaling

In machine learning, some models, such as neural networks, are highly sensitive to the scale of input data. The inputs to a model often contain different types of data with varying scales. Neural networks use weights to transform the input data, and each neuron in a fully connected network receives data from every input feature. If the input features have different scales, training the weights can be slow and unstable. Scaling the input data to have the same scale improves the speed and performance of the model. The scaling of the data range does not influence tree-based models, as they operate through binary tests rather than mathematical calculations. For our research, we scaled the data to have zero mean and a standard deviation of one. This is called standardizing and is achieved by subtracting the mean and dividing by the standard deviation. Mathematically, the standardization of a feature x is represented as:

$$x_{scaled} = \frac{x - \mu}{\sigma} \tag{3.4}$$

where μ is the mean of the feature values, and σ is the standard deviation of the feature values. The mean and standard deviation are computed using the following equations:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3.5}$$

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}, \tag{3.6}$$

where n is the number of observations of the feature. In addition to standardization, other scaling methods, such as min-max and robust scaling, are also used in specific cases. Overall, scaling is a crucial step in preprocessing data for machine learning, as it can significantly impact the performance of a model. However, for tree-based methods, scaling has no effect, as predictions are made based on conditions in the data, not mathematical operations.

3.5 Decision Trees

Decision trees are tree-like models that make decisions based on conditions. As shown in Figure 3.1, each circle represents a node with various types, including decision nodes that split into two other nodes and leaf/terminal nodes that do not. The root node is the topmost decision node. Given an observation, a single path to a leaf node represents the prediction made by the decision tree.

Trees are constructed greedily from the top, meaning that each split is made to minimize the loss function at the current step without considering future splits. More than a single decision tree is required for complex problems. Various methods exist to improve decision tree models, as Figure 3.2 demonstrates. The final step in the figure is XGBoost (Extreme Gradient Boost), a highly efficient and high-performing machine learning algorithm. This section will briefly cover the methods used to optimize decision trees for prediction.

3.5.1 Bagging

Bagging, also known as Bootstrap Aggregation, is a method for training an ensemble of models that contribute to the final prediction. Each model is trained using bootstrapped data (resampled from the original dataset with replacement), resulting in diverse decision trees. The final prediction is the average of all ensemble models.

3.5.2 Random Forest

Random forest is based on bagging, where each tree in the ensemble is made using only a randomly chosen subset of features. This often leads to better generalization and reduced overfitting.

3.5.3 Boosting

In boosting, an ensemble is created, but the trees are not made independently. They are trained one by one, considering the errors of the previous trees. A sample weight is assigned to each sample used to train a tree based on the current ensemble's accuracy. Samples with significant prediction errors are assigned larger weights, and those with accurate predictions are assigned lower weights. The final prediction is a weighted sum of all ensemble predictions, with weights based on each tree's accuracy.

3.5.4 Gradient Boosting

Like in regular boosting, an ensemble of trees is created iteratively by considering the errors made by previous trees. The process starts with a constant model that predicts the mean of all samples. The gradient of the loss function with respect to each sample is calculated, and a tree is made to predict these gradients. The new prediction is the constant plus a small step in the direction of the predicted gradients. Repeated iteration with small steps in the gradient direction helps reduce both bias and variance.

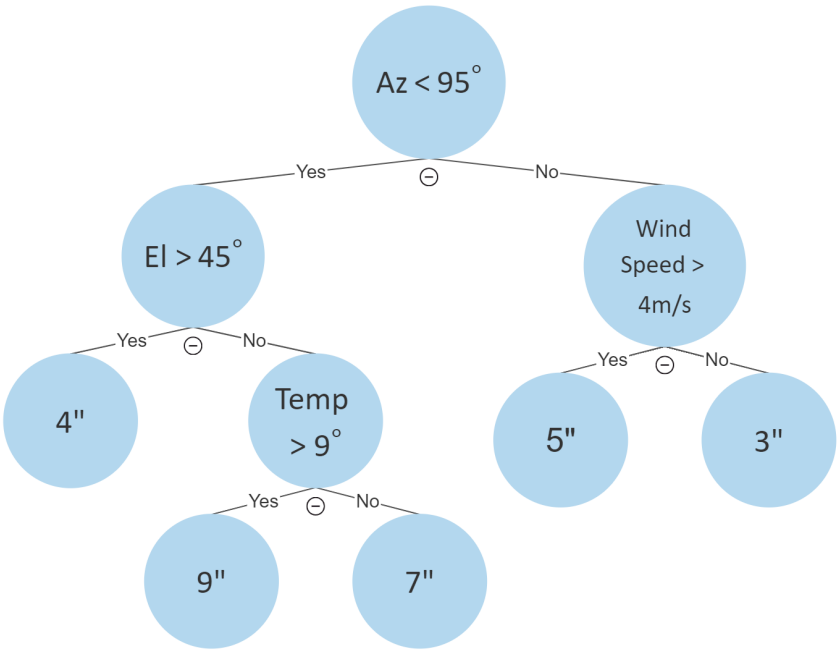


Figure 3.1: An example of a decision tree with three decision nodes and five leaf nodes.

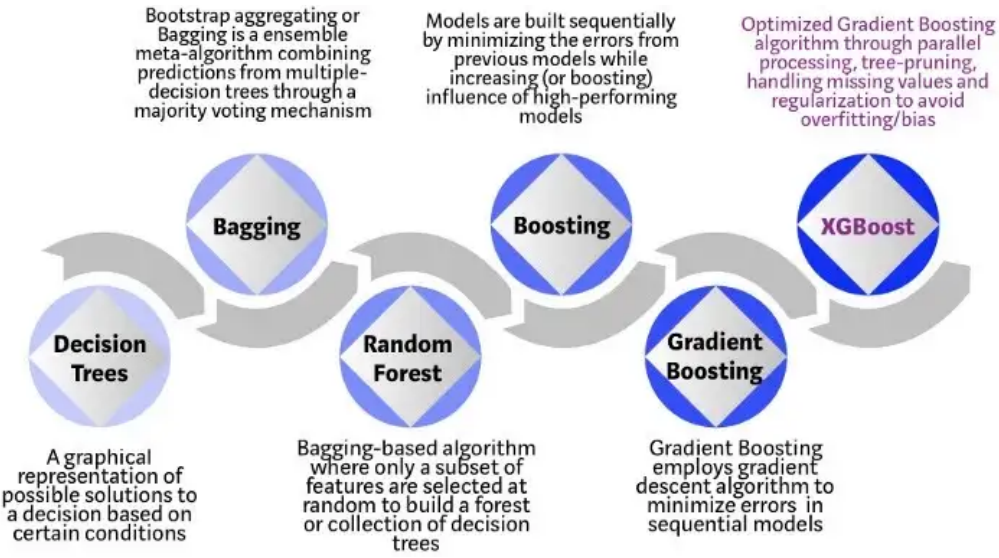


Figure 3.2: The evolution of XGBoost. Source: [16]

3.6 Neural Networks

A Neural Network (NN) is an Artificial Intelligence (AI) model composed of interconnected neurons inspired by biological neural networks in animal brains. These networks are arranged in layers, as shown in Figure 3.3, and consist of an input layer, one or more hidden layers, and an output layer. The size of the hidden layer(s) varies depending on the nature of the problem. A NN processes input to produce an output, ideally close to the true value.

Each connection in an NN has a trainable weight w_{jk}^l, representing the weight from the k^{th} neuron in layer $(l-1)$ to the j^{th} neuron in layer l. Each neuron also has its own bias b_j, added to its output to prevent the input to its activation function σ from being zero. The activation function σ applied to the neuron's output is the final transformation before passing data to the next layer. This nonlinear function is crucial in allowing NNs to learn nonlinear relationships in data [4].

The following is the mathematical explanation of how a neuron processes the outputs from the previous layer.

$$a_j^l = \sigma\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right) = \sigma(z_j^l) \tag{3.7}$$

The quantity

$$z_j^l = \sum_k w_{jk}^l a_k^{l-1} + b_j^l \tag{3.8}$$

will be helpful when explaining how to optimize a neural network and can be considered the weighted input for neuron j in layer l.

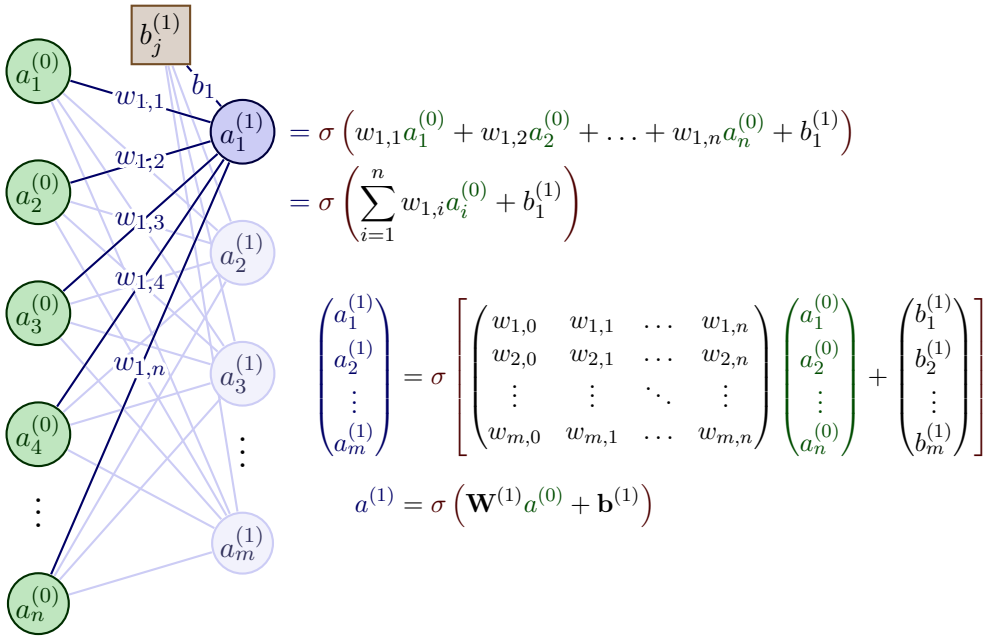


Figure 3.3: This is an illustration of how information is passed through and processed in a neural network. Adapted from work by Izaak Neutelings [17]. Generated using TikZ [24]

3.6.1 Backpropagation

Backpropagation[20] is a fundamental algorithm in training artificial neural networks. It calculates the gradient of the loss function with respect to all the weights and biases in the network, allowing for updating these parameters to reduce the loss. The algorithm is based on four key equations, which we describe in this section.

We define the error in the j^{th} neuron in the l^{th} layer by

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \frac{\partial C}{\partial a_j^l}\sigma'(z_j^l) \tag{3.9}$$

This can also be considered the partial derivative of the cost function with respect to the bias in neuron j in layer l, as

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \frac{\partial C}{\partial b_j^l}\frac{\partial b_j^l}{\partial z_j^l} = \frac{\partial C}{\partial b_j^l}, \tag{3.10}$$

where we have used the relation $\partial b_j^l / \partial z_j^l = 1$ from rearranging equation (3.8). The next equation relates the error in a neuron with the errors in the neurons in the subsequent layer.

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_k \frac{\partial C}{\partial z_k^{l+1}}\frac{\partial z_k^{l+1}}{\partial z_j^l} = \sum_k \delta_k^{l+1}\frac{\partial z_k^{l+1}}{\partial z_j^l} = \left(\sum_k \delta_k^{l+1} w_{kj}^{l+1}\right)\sigma'(z_j^l) \tag{3.11}$$

Note that the indices on the weight w are now swapped. We may think of this equation as an error propagating backward by multiplying the error in layer $l+1$ with the transpose of the weight connecting layer l with $l+1$ We derive the final equation from the partial derivative of the cost function with respect to the weight w_{jk}^l

$$\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{jk}^l} = \delta_j^l a_k^{l-1} \tag{3.12}$$

Equation (3.9) lets us calculate the error in the last layer, and using equation (3.11), we can propagate this error backward through the network, calculating the error for all the neurons. We then use equations (3.10) and (3.12) to calculate the gradient of the cost function with respect to the weights and biases.

3.6.2 Gradient Descent

Gradient Descent (GD) is an iterative optimization algorithm used in machine learning for minimizing a differentiable function. The goal of GD is to update the model's trainable parameters in such a way that the loss function is minimized. In mathematical terms, we aim to find the values of the parameters $\boldsymbol{\theta}$ that minimize the objective function $\mathcal{L}(\mathbf{x}, \boldsymbol{\theta})$, where $\mathbf{x}$ represents the input data. The loss function is typically defined as the mean squared error (3.1) for regression problems, so

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(\mathbf{x}_i, \boldsymbol{\theta}))^2, \tag{3.13}$$

where $f(\mathbf{x}_i, \boldsymbol{\theta})$ is the output of the model for input data $\mathbf{x}_i$, and y_i is the target value.

To achieve the goal, GD involves calculating the gradient of the loss function with respect to the model's trainable parameters and updating them iteratively by taking a small step in the negative direction of the gradient. The iterative update rule can be expressed as follows:

$$\mathbf{v}_t = \eta_t \nabla_\theta \mathcal{L}(\mathbf{x}, \boldsymbol{\theta}), \tag{3.14}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \mathbf{v}_t, \tag{3.15}$$

where η denotes the learning rate, and ∇_θ denotes the gradient with respect to $\boldsymbol{\theta}$. The learning rate determines the step size of the update, and it is important to choose a suitable value to ensure convergence of the optimization.

One major limitation of GD is that it can get stuck in local minima, yielding suboptimal results. The choice of initial parameter values $\boldsymbol{\theta}$ can also impact the final

optimized model. Moreover, computing the gradient using the entire dataset can be computationally expensive for large datasets. To address these limitations, various modifications of GD have been proposed, such as stochastic gradient descent (SGD) and mini-batch gradient descent (MBGD), which compute the gradient using only a subset of the data at each iteration. These modifications can help to accelerate the convergence and improve the scalability of GD.

3.6.3 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is a widely used optimization algorithm in machine learning that addresses some of the limitations of Gradient Descent (GD). Unlike GD, which computes the gradient using the entire dataset at each iteration, SGD computes the gradient using only a randomly sampled subset, called a mini-batch. This makes SGD more efficient and less computationally expensive than GD, particularly for large datasets. Furthermore, by randomly sampling mini-batches, SGD is more likely to escape local minima and converge to the global minimum. The update rule for SGD can be derived similarly to that for GD, with the only difference being the replacement of the full dataset with a mini-batch. By iteratively updating the model's parameters using mini-batches, SGD can converge faster and more robustly than GD.

However, SGD also has limitations to consider. If the learning rate is too large, the optimization may overshoot the minimum and fail to converge. On the other hand, if the learning rate is too small, the optimization may converge very slowly. In addition, if there are areas in the function space with small gradients, the optimization may stagnate and fail to converge. To address these limitations, various modifications of SGD have been proposed, such as adaptive learning rate methods like Adagrad and RMSprop, which adjusts the learning rate dynamically based on the history of the gradients. These modifications can improve the stability and convergence speed of SGD.

3.6.4 Momentum

In practice, SGD is mostly used with momentum. Momentum serves as a memory of previous momenta and can improve the convergence speed of SGD, particularly in areas of the function space with low gradients, such as local minima.

The update rule for momentum can be expressed as follows:

$$\mathbf{v}_t = \gamma \mathbf{v}_{t-1} + \eta_t \nabla_\theta \mathcal{L}(\mathbf{x}, \boldsymbol{\theta}) \tag{3.16}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \mathbf{v}_t, \tag{3.17}$$

where γ is the momentum parameter with $0 \leq \gamma \leq 1$. The momentum term considers the update of the previous step, in addition to the gradients at the current step. By incorporating previous momenta, momentum can smooth out variations in the optimization trajectory and accelerate convergence towards the minimum.

Momentum is particularly useful when the gradient direction is consistent across many iterations, as it allows the optimization to maintain a higher velocity in the same direction. In contrast, in areas of high variance or noisy gradients, momentum may cause overshooting and slow down convergence. To address this, adaptive momentum methods like Adam have been proposed, which adjust the momentum parameter dynamically based on the history of the gradients. These methods can improve the convergence speed and stability of momentum-based optimization algorithms.

3.6.5 Adam

Adam is an optimization algorithm that combines the benefits of both SGD with momentum and adaptive learning rate methods. It uses a running average of the first and second moments of the gradient to compute per-parameter adaptive learning rates. Adam updates the parameters iteratively as follows:

$$\mathbf{g}_t = \nabla_\theta \mathcal{L}(\mathbf{x}, \boldsymbol{\theta}) \tag{3.18}$$

$$\mathbf{m}_t = \beta_1 \mathbf{m} t - 1 + (1 - \beta_1)\mathbf{g}_t \tag{3.19}$$

$$\mathbf{s}_t = \beta_2 \mathbf{s} t - 1 + (1 - \beta_2)\mathbf{g}_t^2 \tag{3.20}$$

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1 - (\beta_1)^t} \tag{3.21}$$

$$\hat{\mathbf{s}}_t = \frac{\mathbf{s}_t}{1 - (\beta_2)^t} \tag{3.22}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{s}_t} + \epsilon}, \tag{3.23}$$

where $\mathbf{g}_t$ denotes the gradient at time step t, $\mathbf{m}_t$ and $\mathbf{s}_t$ are the first and second moment estimates, respectively. β_1 and β_2 control the decay rate of the first and second moments, respectively. η_t is the learning rate, and ϵ is a regularization constant to prevent division by zero.

Adam has several advantages over other optimization algorithms, including its ability to adaptively compute per-parameter learning rates and the robustness of its estimates to noise in the gradient. The adaptive learning rates can help speed up convergence and lead to better performance. Furthermore, the memory of previous first and second-order gradient estimates enables the algorithm to be more robust to noise and outliers in the data. As a result, Adam is widely used and has become the de facto standard optimization algorithm in deep learning.

3.6.6 Activation Functions

Activation functions play a crucial role in training a neural network by allowing it to learn non-linear relationships between inputs and outputs. Different activation functions have varying properties; we will discuss some of the most common ones in this section. Properties like non-linearity, differentiability, monotonicity, smoothness, and zero-centering are important for activation functions. Non-linearity enables the model to capture complex relationships, differentiability is necessary for calculating the derivative of the loss function with respect to the trainable weights, monotonicity helps ensure stability in activation outputs, smoothness stabilizes gradients during training, and zero-centering balances the activation distribution within the model.

- Non-linearity enables the model to capture complex relationships
- Differentiability is necessary for calculating the derivative of the loss function with respect to the trainable weights
- Monotonicity helps ensure stability in activation outputs, smoothness stabilizes gradients during training
- Smoothness: A smooth activation function helps stabilize the gradients and training.
- Zero-centering balances the activation distribution within the model.

Tanh

Tanh, the hyperbolic tangent function is given by

$$Tanh(x) = \frac{e^x + e^{-x}}{e^x - e^{-x}} \tag{3.24}$$

ReLU

The Rectified Linear Unit (ReLU) activation function pushes all negative values to zero while leaving positive values unchanged, which introduces non-linearity while solving the vanishing gradients problem by having a gradient of either 0 or 1 for negative and positive values, respectively.

$$ReLU(x) = \begin{cases} x & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.25}$$

GeLU

The Gaussian Error Linear Unit (GeLU) is a smooth approximation of the ReLU function, given by

$$GeLU(x) = x\Phi(x), \tag{3.26}$$

where $\Phi(x)$ is the standard Gaussian cumulative distribution function.

GeLU can be approximated with

$$GeLU(x) \approx 0.5x \left(1 + \tanh\left[\sqrt{2/\pi}(x + 0.044715x^3)\right]\right), \tag{3.27}$$

which is faster to compute than the original definition but can result in worse performance. For computational efficiency, we used this approximation.

3.7 Model Explainability

In the context of machine learning, SHAP [14] and SAGE [8] apply the same idea to determine the contribution of each feature to a prediction. SHAP provides a local explanation by computing the contribution of each feature to the prediction of a single data point. On the other hand, SAGE provides a global explanation by computing each feature's contribution to the model's overall prediction performance. These methods allow us to understand the relationship between the features and the prediction, which is particularly useful when the model is too complex to interpret. Additionally, they provide a way to validate the model's fairness and bias. By understanding which features contribute the most to a prediction, one can determine if the model is fair or biased and if the prediction is trustworthy.

Both SHAP and SAGE methods are based on Shapley values [22], a concept in game theory introduced by Lloyd Shapley in 1951. Shapley values determine each player's contribution to a group's surplus or overall value. The explanation below of Shapley values, SHAP, and SAGE is inspired by a blog post by Ian Covert [7], the author of [8].

The Shapley value for a player i in a cooperative game with d players is

$$\phi_i(w) = \frac{1}{d} \sum_{S \subseteq D \setminus \{i\}} \binom{d-1}{|S|}^{-1} [w(S \cup \{i\}) - w(S)] \tag{3.28}$$

where D is the set of all players, S is a coalition of players, $w(S)$ is the value of the coalition S, and $|S|$ is the number of players in the coalition. This formula satisfies four important conditions:

- Efficiency: The sum of all Shapley values is equal to the group's total value.
- Symmetry: If two players i and j have the same impact on all coalitions with $w(S \cup \{i\}) = w(S \cup \{j\})$ for all S, they should have the same Shapley value $\phi_i(w) = \phi_j(w)$.
- Dummy: A player i that makes no contribution to the group with $w(S \cup \{i\}) = w(S)$, should receive a value of zero, or $\phi_i(w) = 0$.
- Linearity: A player's value is proportional to their contribution to the group. If player i contributes twice as much as player j to the group's overall worth, then player i should have twice the Shapley value.

3.7.1 SHAP

Shapley values explain how each feature $(x^1, \ldots, x^d)$ in a model f contributes to the deviation from the mean prediction $\mathbb{E}[f(x)]$ of the dataset for a single prediction. It assigns a value $\phi_1, \ldots, \phi_d$ to each feature that quantifies the feature's influence on the prediction $f(x)$. SHAP (Shapley Additive Explanations) computes approximate Shapley values for machine learning models.

We define a cooperative game $v_{f,x}$ to represent a prediction given the features x^S, as

$$v_{f,x}(S) = \mathbb{E}\left[f(X)|X^S = x^S\right], \tag{3.29}$$

where x^S are known, and the remaining features are treated as random variable $X^{\bar{S}}$ (where $\bar{S} = D \backslash S$). This is the mean prediction $f(X)$ when the unknown values follow the conditional distribution $X^{\bar{S}}|X^S = x^S$.

Using a subset of features from the prediction while sampling the rest from the dataset reduces the chance of improbable samples. Given this convention for making predictions, we can apply the Shapley value to define each feature's contribution to the prediction $f(X)$ using Shapley values $\phi_i(v_{f,x})$. A Shapley value of $\phi_i(v_{f,x}) > 0$ indicates that feature i contributes to an increase in prediction $f(X)$. A negative Shapley value $\phi_i(v_{f,x}) < 0$ indicates the opposite, that the feature contributes to a decrease in $f(X)$. Uninformative features will have small values $\phi_i(v_{f,x}) \approx 0$.

3.7.2 SAGE

SAGE (Shapley Additive Global Importance) explains how every feature contributes to the model's overall performance, and it relates to SHAP in a simple way. For a given feature, the global feature importance is the average SHAP value (for that feature) across all samples in the dataset. This is, however, different from how it is calculated in practice. A paper by Ian Covert et al. [9] on global feature importance proposes an algorithm that aims directly at a global feature explanation, unlike the SHAP values, which makes it faster. This is the algorithm used for approximating the SAGE values for the features in the thesis.

3.8 Mutual Information

Mutual information is a fundamental measure of the statistical dependence between two random variables, providing a way to quantify the amount of information one variable conveys about the other [21]. For a pair of discrete random variables X and Y, we have

$$I(X;Y) = \sum_{y} \sum_{x} p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right), \tag{3.30}$$

where $p(x,y)$, $p(x)$, and $p(y)$ are the joint and marginal probabilities, respectively. The mutual information captures linear and nonlinear relationships between variables, unlike Pearson's correlation coefficient, which can only detect linear relationships. However, mutual information has limitations in that it relies on binning the data, which can introduce bias and limit the resolution of the information.

Furthermore, estimating mutual information for high-dimensional data sets can be computationally expensive. Despite these limitations, mutual information remains a popular tool in feature selection, data visualization, and machine learning.

Part II

Data Processing & Methods

Chapter 4

Data Processing

This chapter of the thesis outlines the methodology used for data analysis, cleaning, transformation, and feature engineering. Data-driven methods have the potential to learn all relations in the data, but the size of the dataset limits this. Therefore, cleaning the data and selecting features containing information relevant to the desired output is essential. With the system's complexity, identifying relevant features can be challenging. To address this, we employed data-driven modeling to help identify important features while using feature engineering to incorporate our understanding of the system and create informative features.

Cleaning data involves removing irrelevant data that could confuse the model, thereby ensuring that the model learns from the most relevant information. Feature engineering involves incorporating domain knowledge into the model to create features that provide additional information.

We performed data analysis to decide which features to train our models. This analysis involves understanding the relationships between the different variables in the dataset and identifying which variables could be helpful in predicting the target variable.

4.1 Cleaning Pointing Scan Data

When utilizing data-driven modeling for predictive purposes, ensuring that the dataset is clean and informative is crucial. In this project, various factors may impact the quality of the data, and therefore, we implemented measures to clean the data based on our knowledge of the telescope's operation. We employed a criteria-based approach and a machine learning classifier to remove pointing scans from the dataset. During the removal of pointing scans, it is important to strike a balance between removing noise and retaining relevant information. Outliers in the training data can introduce bias into machine learning models, as these data points may not accurately represent typical conditions. Consequently, having outliers in the training data can be more damaging than removing good pointing scans. Therefore, we have a strict approach when cleaning the data to ensure high-quality datasets for model training.

4.1.1 Cleaning Criteria

To eliminate unreliable or unusable scans, we applied criteria informed by the insights of astronomers at APEX. The following list outlines the criteria used to filter out such scans:

- Scans using the HOLO transmitter: These scans are aimed at a radio tower and are not realistic data for training an ML model.
- Scans using ZEUS2: These are highly experimental pointing scans and unreliable.
- Scans using CHAMP690: There are very few scans with this instrument.
- Scans in January and February of 2022: The weather is unreliable and there are few scans in this period.
- Scans that are tracking tests
- Scans after 17.09.2022 since we only have sensory data until this point

After this filtering, there were 5901 out of 8862 scans left.

4.1.2 Pointing Scan Classifier

Method

In addition to cleaning the data based on the criteria above, we had to remove the outright bad pointing scans (like 2.2b 2.1b, 2.1b). The scan quality is often obvious when inspecting the data visually, but it is hard to develop suitable measures to identify which scans are good or bad. Instead, we trained a classifier to predict whether a scan is of good or bad quality. We used an XGBoost classifier with 13 features as inputs, all of which are present in the pointing scan figures (2.1 and 2.2). The first 12 features are the amplitudes, FWHMs, pointing offsets, and these values' uncertainties. The last feature is the beamwidth of the telescope for the given observing frequency.

We had to label a dataset set by manually looking at pointing scans. The size of the dataset set was 369 samples with 270 good and 99 bad scans. Table 4.1 shows the hyperparameters and search ranges we used when optimizing this model, along with the resulting best parameter values. We also used *scale_pos_weight* to consider the unbalanced classes, for which the value is the ratio of negative to positive classes (number of bad scans divided by the number of good scans). We split the data into 80% for training and the rest for validating, corresponding to 295 and 74 samples for training and validation, respectively.

Table 4.1: This table presents a list of parameters we sampled during hyperparameter tuning for the pointing scan classifier. The table includes names, sampled distributions and corresponding ranges, and parameter values for the best model.

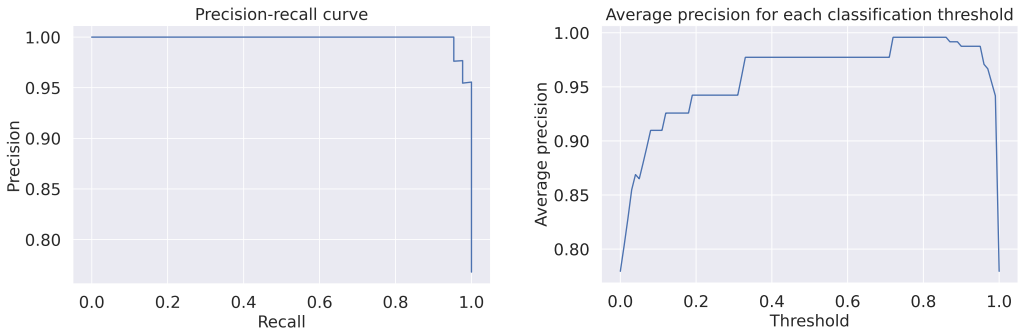
Parameter	Sample Distribution	Range	Best Parameter Value
max depth	Uniform	[1, 5]	2
n estimators	Uniform	[1, 80]	53

Results

The XGBoost classifier performed well with a 97% overall accuracy on the validation set. Figure 4.1 shows the precision-recall curve on the left and the average precision curve on the right. From the precision-recall curve, it is clear that we can achieve

close to 100% precision while still having a high recall. We select a large threshold such that the classifier removes most bad scans from the training data, because a bad pointing scan is potentially more harmful for the model than discarding a few good scans. The average precision curve shows an optimal threshold for maximizing the precision, which is about 80%.

We used the classifier to clean the dataset further, using prediction threshold 0.8, we remove another 575 scans, leaving us with 5326 scans for the rest of the analysis.



(a) Precision-recall curve on the test set.

(b) Average precision for different classification threshold.

Figure 4.1: Precision-recall and average precision curve for the XGBoost classifier when classifying good and bad pointing scans in the test set.

4.2 Scan Duration Analysis

As mentioned in the database section 2.2, the scans' timestamps are not the accurate start time of a scan. The tiltmeter dump files with the flag indicating whether the telescope is idle, preparing to observe, or observing, is the only accurate data we have when the telescope performs a pointing scan. Therefore, we need to combine the timestamp of the pointing scan with the flag in the dump files to analyze the duration of scans.

4.2.1 Analysis

First, we convert the different scan flags to numbers. *IDLE* and *PREPARING* is the set 0, and *OBSERVING* is set to 1. Then we can subtract the previous rows from all rows, resulting in the value 1 when the scan starts, and -1 when it ends. Table 4.2 shows an example of the result.

Time	Flag	Flag Integer	Δ
11:21:21	IDLE	0	0
11:21:22	PREPARING	0	0
11:21:23	OBSERVING	1	1
11:21:24	OBSERVING	1	0
11:21:25	OBSERVING	1	0
11:21:26	IDLE	0	-1

Table 4.2: This table shows the tiltmeter dump file containing the telescope state flag, and how we found the start ($\Delta = 1$) and end ($\Delta = -1$) of a scan.

4.2.2 Algorithm

With the scan timestamp and the observing flag from tiltmeter dumps, we used the following algorithm to obtain the start and end of pointing scans.

Algorithm 1 Algorithm that finds the start and end of pointing scan

Input:

- Pointing scan timestamps $D = \{D_1, \ldots, D_n\}$
- Timestamps $T = \{T_1, \ldots, T_m\}$ and scan flag $F = \{F_1, \ldots, F_m\}$

Output: Start and end of pointing scans $S = \{S_i, \ldots, S_n\}$ and $E = \{E_i, \ldots, E_n\}$

```
for i = 1, ..., m do
    if F_i = OBSERVING then
        F_i = 1
    else
        F_i = 0
    end if
end for

for i = 1, ..., n do
    T̂ = {T_j, if T_j > D_i}_j^m
    F̂ = {F_j, if T_j > D_i}_j^m
    for all t_i, f_i in T̂, F̂ do
        Δ = f_i − f_{i−1}
        if Δ = 1 then
            S_i = t_i
        end if
        if Δ = −1 then
            E_i = t_i
            Continue
        end if
    end for
end for
```

4.2.3 Results

By analyzing start and end timestamps for all the scans we had tiltmeter dumps for, we see that the first *OBSERVING* flag present after a scan is on average 53.9 seconds after the scan timestamp on average, with a standard deviation of 20.5 seconds.

Figure 4.2 shows boxplots of this time difference for each of the instruments, which strongly indicates that assuming the starting point of a scan is 53.9 seconds after the timestamp is reasonable. In the same plot, we also see that the starting time is fairly constant for the different instruments. The right plot of Figure 4.3 shows the time difference in seconds between the first observing flag after a scan timestamp throughout the year. From the plot, this stays constant over time.

Now that we have found the starting points of the pointing scans, we can look at their duration. The left plots in Figure 4.3 and 4.2 show the duration of the pointing scans for different instruments. From these figures, it is clear that the duration of a pointing scan varies a lot. A varying scan duration is problematic because we only have these tiltmeter dump files for $2875/8381 \approx 34\%$ of the pointing scans. To address this issue, we collected data for feature engineering over a shorter period of time. It is important to note that using data from after a pointing scan has ended can be inaccurate, as the telescope may start observing a different source. When examining the scatter plot of scan durations, we observed clusters of scans around 60-70 seconds, 120-130 seconds, and so on. To ensure accuracy, we used the mean scan duration grouped by instrument and shorter than 100 seconds as the cutoff for the duration of time from which we collected data. For the scans with an exact start and end time, we used this time period instead. Add list of values

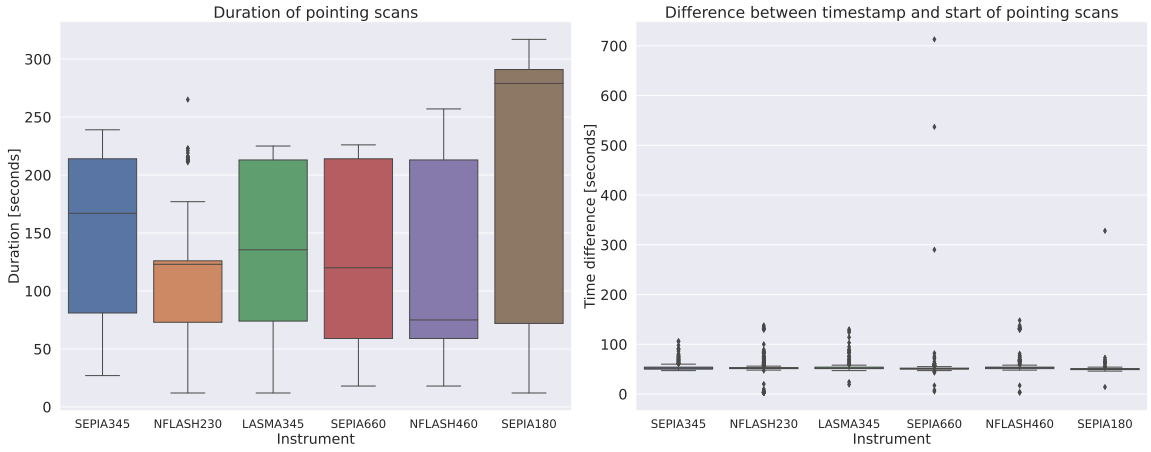


Figure 4.2: Box plot of the duration of scans, and the time difference between the timestamp of a scan and the actual start of it.

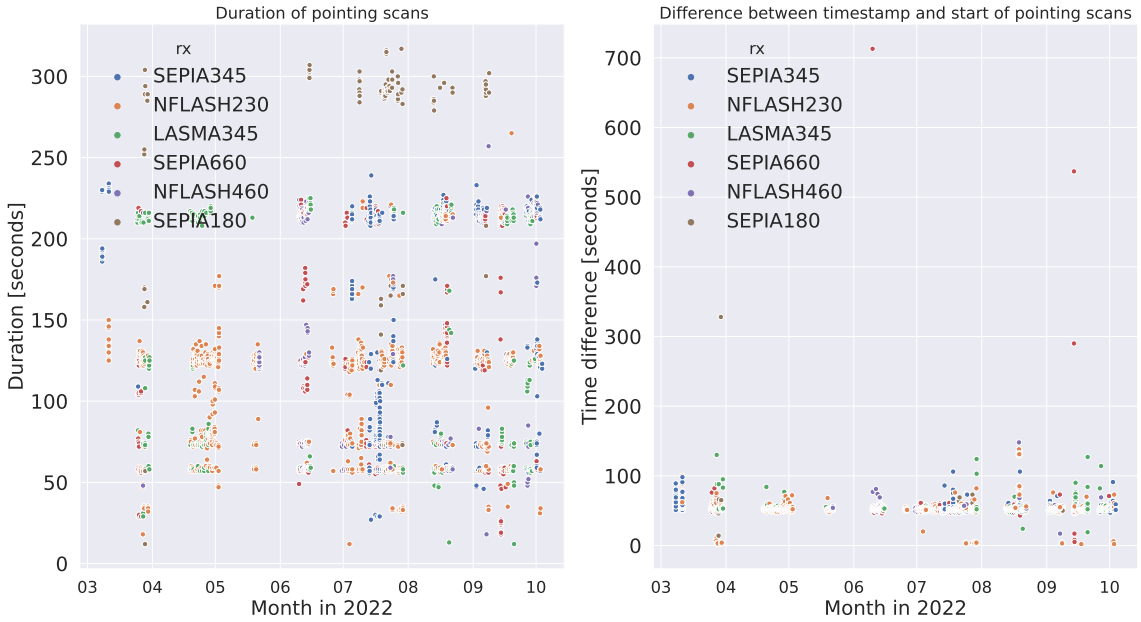


Figure 4.3: Scatter plot of the duration of scans, and the time difference between the timestamp of a scan and the actual start of it.

4.3 Feature Engineering

Feature engineering is important to ensure that the inputs for the machine learning models are as informative as possible. Informative features make it easier for the machine learning model to learn the relationship between features and target values. With limited training data, this is essential.

4.3.1 Median Values

From the scan duration analysis 4.2, we know that the duration of a scan varies. Therefore, we use the median value during a pointing scan as a feature.

4.3.2 Turbulence

Turbulence could affect the pointing, and we used the simple model

$$\tau_{wind} = \sigma^2_{wind}, \tag{4.1}$$

where the σ^2_{wind} variance in windspeed. We calculated the variance in a period of 5 minutes before the pointing scan.

4.3.3 Position of the Sun

Observers at the telescope report that the sun is affecting the pointing. It is most drastically affected when the sun sets or rises, likely due to rapid temperature change leading to deformation in the telescope structure. We also think the sun's position affects the pointing. For instance, if the sun is shining on the left side of the telescope, it will affect the pointing differently than if it is on the right side. Obtaining the Sun's position for the telescope's location is done using the Python module PyEphem [19].

Using the azimuth angle of the sun and the telescope, we can calculate the position of the sun with respect to the pointing with

$$\Delta Az_{\odot} = Az_t - Az_{\odot} \tag{4.2}$$

This will result in values outside the $[-180°, 180°]$. An example is if $Az_{\odot} = 179°$ and $Az_t = -179°$. The calculation in equation (4.2) yield $-179° - 179° = -358°$, which corresponds to the sun being 358° to the right of the telescope, while it ideally should be 2° to the left. Therefore, we adjust the values accordingly

$$\Delta Az_{\odot} = Az_{\odot} + 360°, \text{ for } \Delta Az_{\odot} < 180° \tag{4.3}$$

$$\Delta Az_{\odot} = Az_{\odot} - 360°, \text{ for } \Delta Az_{\odot} > 180° \tag{4.4}$$

Here, the interval of the difference in azimuth is fixed to the interval $(-180°, 180°)$, where 0° means the telescope is pointing towards the sun in the azimuth direction. $\Delta Az_{\odot} = 90°$ corresponds to the sun being direct to the left of the pointing direction.

We also calculated the change of $\Delta Az_{\odot}$ in the past five minutes. Let $\Delta Az_{\odot}(t)$ denote the difference between the telescope's and the Sun's azimuth given time t in minutes, then the change in the past five minutes are given by

$$\Delta_5 \Delta Az_{\odot} = \Delta Az_{\odot}(t) - \Delta Az_{\odot}(t-5) \tag{4.5}$$

4.3.4 Time based features

We also used time-based features to capture the time of day and year. There are seasonal changes that affect the pointing throughout the year, and there are environmental factors throughout the day that are correlated with the time of day. Therefore, we also used these features

- **Continous month**: The number of months after 2022-01-01 00:00:00
- **Integer month**: The number of integer months after 2022-01-01 00:00:00
- **Time of day**: The number of hours after 00:00:00 the given day

4.3.5 Additional features

There are additional features from the pointing scan data that we used as features. The correction terms CA (2.17) for azimuth and IE (2.18) for elevation are used as inputs when predicting pointing offsets. In addition, we used the type of receiver as input. The environmental factors that cause pointing offsets might affect the different receivers differently; therefore, the model must consider this possibility.

Chapter 5

Machine Learning Experiments

This chapter provides an overview of two machine learning experiments related to the two research questions in section 1. The first experiment aims to examine the effectiveness of an XGBoost model in predicting pointing scan offsets to enhance the pointing accuracy. The primary objective of this experiment is to assess whether the proposed model can outperform the current model in terms of pointing accuracy. The second experiment aims to investigate the effectiveness of neural networks in developing a pointing model that could replace the current linear model, which is created through linear regression. It explores the feasibility of a more sophisticated model that can account for environmental factors in addition to the theoretical and empirical terms used in the current pointing model.

5.1 Experiment 1: Pointing Correction Model

This experiment aims to improve the accuracy of the existing pointing model by training XGBoost models to predict offsets obtained from pointing scans. To accomplish this, we utilized two different datasets, which we processed using the cleaning outlined in section 4.1. The difference between these datasets is that one contains the scans from all instruments, while the other only contains the scans from NFLASH230. By training our models on these datasets, we aim to reduce the pointing offset and improve the accuracy of the pointing. In addition, we varied the way we split the datasets for training and testing. We considered two cases:

- **Case 1:** The dataset is sorted by date and split into six equal-sized folds. We consider each of the folds one by one. For each of these folds, we use the last 1/6th of the data as a test set and the remaining 5/6th as training and validation.
- **Case 2:** The dataset is sorted by date and split into six equal-sized folds. We used 5/6 of the data for training and validation and the remaining for testing. We repeated this process six times, using each fold for testing once.

Figure 5.1 illustrates the two cases. In both cases, we trained and validated the model on 5/6 of the data and tested on the last 1/6. The difference is the amount of data used for training, which can indicate whether models trained on shorter or longer periods perform better. Using longer period, and thus more data, can help the model find complex relations. However, a smaller period may be better for learning some relationships, as we expect less variation in a shorter period.

We also split the training and validation data such that scans from a given day only can be either in the training or validation set, not both. When splitting the data, we used 35% of the days for validation and 65% for training.

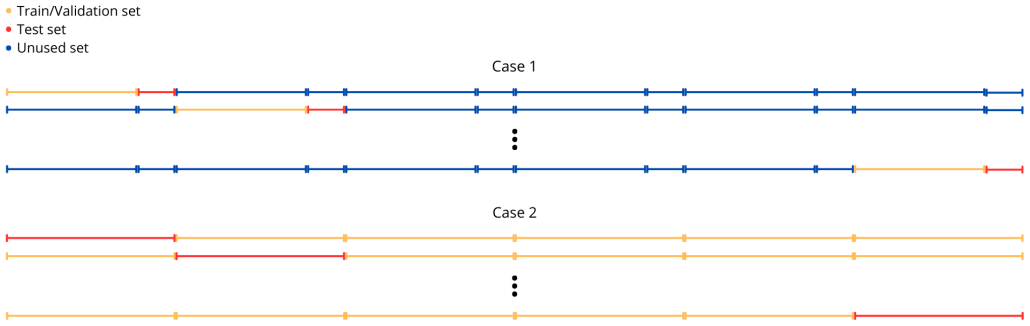


Figure 5.1: This figure shows two cross-validation cases: the orange region represents the train and validation set, the red region represents the test set, and the blue region is unused for evaluation. In **Case 1**, the dataset is split into six equal-sized folds sorted by date. For the selected fold, we use the last part (colored red) for testing and the remaining part (colored orange) for training and validation. This process is repeated six times, once for each fold. In **Case 2**, the dataset is again split into six equal-sized folds sorted by date. However, we use one whole fold for testing this time and the remaining five for training and validation. This process is repeated six times, with each fold used exactly once for testing.

5.1.1 Feature Selection

We trained models using a range of features, specifically $k = [2, 5, 10, 20, 30, 40, 50]$ features. For each model, we selected the k features that had the greatest mutual information (3.30) with the target value on the training and validation set. This approach helps us identify the most important features to improve the model's performance. Selecting a subset of features can reduce the noise in the data. By selecting different numbers of features, we can explore the trade-off between model complexity and performance.

5.1.2 Model Architecture

We performed a hyperparameter search for each model using the parameter space in Table 5.1. The search space includes eight hyperparameters that affect the model's complexity, such as the maximum depth of the trees, the regularization strength, and the learning rate. We used a uniform or log-uniform distribution to sample each hyperparameter within a specific range. We evaluated 200 different combinations of hyperparameters (for each dataset, cross-validation case, target variable, and the number of features selected) to find the optimal values for each model. The models were validated using the MSE.

Table 5.1: This table presents a list of parameters we sampled during hyperparameter tuning for XGBoost the pointing correction model. The table includes names, sampled distributions, and corresponding ranges.

Parameter	Sample Distribution	Range
`max_depth`	Uniform	[1, 5]
`reg_lambda`	Uniform	[0, 1]
`colsample_bytree`	Uniform	[0.5, 1]
`n_estimators`	Uniform	[20, 500]
`learning_rate`	Log-Uniform	$[10^{-5}, 1]$
`subsample`	Uniform	[0.5, 1]
`gamma`	Log-Uniform	$[10^{-5}, 1]$
`min_child_weight`	Uniform	[1, 10]

5.1.3 Model Evaluation

To evaluate the performance of the models, we calculated the RMS on each test fold and compared it to the current RMS of the telescope on the same data. We calculated the RMS for azimuth and elevation separately since an XGBoost model only can predict one target. Given fold j and target either azimuth or elevation, we calculate the RMS by

$$RMS_{\text{target},j} = \sqrt{\frac{1}{N_j}\sum_{i=1}^{N_j}(\tilde{\delta}_{\text{target},ji} - \delta_{\text{target},ji})^2}, \tag{5.1}$$

where $\tilde{\delta}_{\text{target},ji}$ is the predicted pointing offset and $\delta_{\text{target},ji}$ is the true pointing offset for the ith pointing scan in fold j. N_j is the number of pointing scans in fold j.

The measure we used for evaluating an analyzing the results is the RMS ratio between the XGBoost pointing correction model and the current model, given by

$$r_{RMS,j} = \frac{RMS_{\text{target},j}}{RMS_{\text{current},j}}. \tag{5.2}$$

This measure is useful because it compares our results to the current performance of the telescope. If $r_{RMS,j} < 1$, it indicates that the XGBoost model provides an improvement over the current performance of the telescope for a given fold.

To obtain an overall measure of the model's performance compared to the current performance of the telescope, we averaged the ratios $r_{RMS,j}$ over all six test folds

$$\bar{r}_{RMS} = \sum_{i=1}^{6}\frac{RMS_{model,j}}{RMS_{current,j}}. \tag{5.3}$$

This gives us an average ratio $\bar{r}_{RMS}$, which measures the expected performance of the pointing correction model. If $\bar{r}_{RMS} < 1$, it indicates that the XGBoost model outperforms the current pointing correction method on average across all test folds. By comparing the average ratio $\bar{r}_{RMS}$ for the two different cross-validation cases in Figure 5.1, we can identify which models provide the best performance.

5.2 Experiment 2: Pointing Model using Neural Networks

This experiment uses the raw dataset containing input coordinates, Az_{input} and El_{input} respectively, and corresponding true observed values Az_{observed} and El_{observed}.

The goal is to find a model f such that

$$f(X) \approx (\delta_{\text{Az}}, \delta_{\text{El}}) = (Az_{\text{observed}} - Az_{\text{input}}, El_{\text{observed}} - El_{\text{input}}) \tag{5.4}$$

We split the data into a train, validation, and test set. The last 15% of the data, which we sorted by date, is used for testing. We use the remaining 85% of the data for training and validation and split this set into 20% for training and 80% for validation. This results in $\sim 76\%$ and $\sim 24\%$ of the total dataset used for training and validation.

5.2.1 Feature Selection

Selecting the right features is essential in improving the pointing model's accuracy. This model uses two types of features: geometric and harmonic terms (some of which are part of the current analytical pointing model [(2.3),(2.5)]) and new features extracted from the telescope's database. For the geometric and harmonic terms, we analyzed Pearson and Spearman's rank correlation to the target values in equation (5.4) and picked a subset of features that showed a strong correlation. We did the same for the other features, except we picked out the features that showed a correlation of either type larger than 0.1. Tables A.10 and A.11 list the features we extracted from the database with a correlation equal to or greater than 0.1. During model training, we randomly selected $n \in [2, 19] \cap \mathbb{Z}$ features from these lists and used them to train the model. This way of choosing features does not consider complex dependencies between the features that can affect the offsets. However, training neural networks is computationally heavy, so we had to select the features we tested carefully.

5.2.2 Model Architecture

This experiment utilized four different model architectures. The first architecture involved feeding all input data into one or two hidden layers. The other three architectures incorporated machine learning techniques by separating the geometric and harmonic terms of the input data from the other features and processing them using distinct architectures. The approaches aimed to keep the current model's simplicity and performance while incorporating new features.

The following are the four different architectures:

1. **Regular Neural Network:** All features are passed through the same layers, all with a nonlinear activation function. See Figure 5.2a

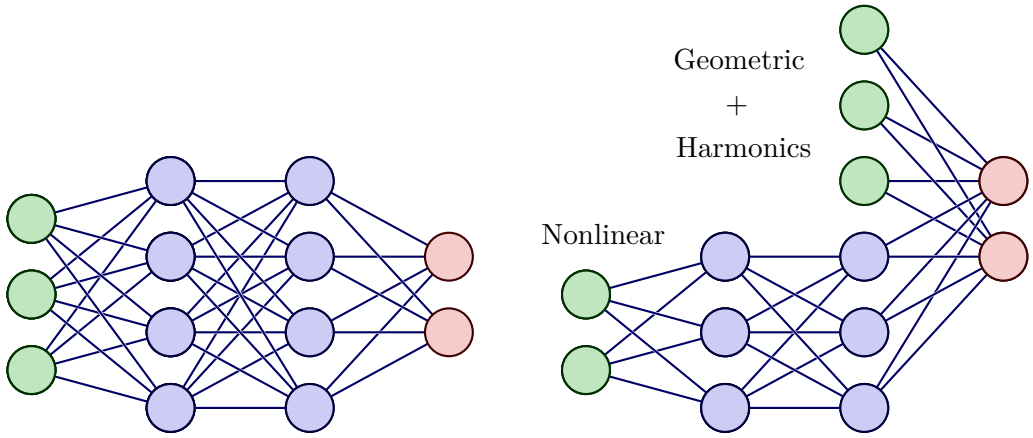
2. **Neural Network with Separated Features 1:** This architecture separates the input features into two groups: geometric and harmonic features and the rest of the features. The geometric and harmonic features are connected directly to the linear output layer, while we pass the remaining features through layers with nonlinear activation functions. See Figure 5.2b

3. **Neural Network with Separated Features 2:** This architecture is similar to the previous architecture, but we feed the geometric and harmonic features through an additional layer with a nonlinear activation function before connecting them to the output layer. See Figure 5.2c

4. **Neural Network with Separated Features 3:** This architecture combines the previous two architectures by passing the regular features through a few

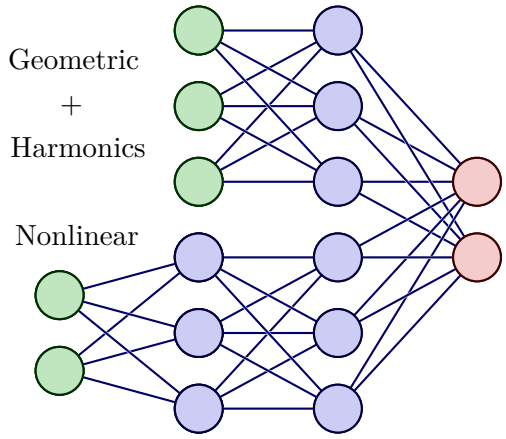
hidden layers with nonlinear activation functions before concatenating them with the geometric and harmonic features. We then pass the combined features through a final layer before connecting them to the output layer. See Figure 5.2d

These are visualized in Figure 5.2.

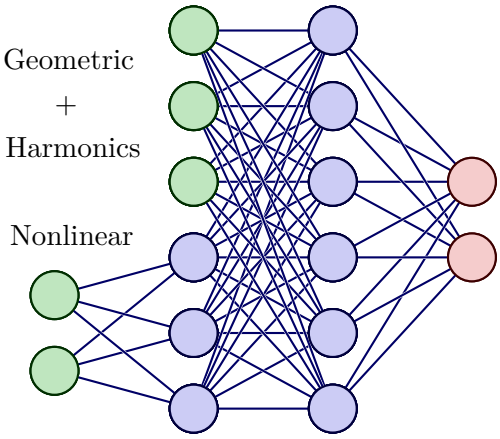


(a) **Regular neural network:** This is the standard neural network architecture without any feature separation. All features are connected to the same layers.

(b) **Neural network with separated features 1:** In this architecture, the geometric and harmonic features are separated from the other features and directly connected to the output layer without any nonlinear activation function.



(c) **Neural network with separated features 2**: Similar to the previous architecture, the geometric and harmonic features are separated from the other features. However, they are also processed by a nonlinear activation function before being connected to the output layer.



(d) **Neural network with separated features 3:** In this architecture, we concatenate the processed regular features to the geometric and harmonic features before being connected to the output layer.

Figure 5.2: The different architectures tested for the pointing model.

The hyperparameters for the neural networks were randomly sampled from different distributions, as presented in Table 5.2. We used the Adam optimization algorithm for all models and trained 100 networks of each architecture for 200 epochs. We picked the model from the epoch with the best performance on the validation set.

Table 5.2: This table presents a list of parameters we sampled during hyperparameter tuning for the base pointing model. The table includes names, the distribution we sampled from, and corresponding ranges.

Name	**Distribution Type**	**Range**
hidden layers	uniform integer	[1,3]
hidden layer size	uniform integer	[20, 120]
learning rate	uniform	[0.001, 0.02]
batch size	uniform integer	[32, 512]
activation	categorical	[gelu, tanh]
loss function	categorical	[MSE (3.2), MSD (3.3)]

5.2.3 Loss Function and Model Evaluation

To evaluate the performance of the models, we used the root mean squared (RMS), measured in arcseconds, on the test set. We calculate the RMS as follows:

$$\text{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\Big((\tilde{\delta}_{Az,i} - \delta_{Az,i})^2 + (\tilde{\delta}_{El,i} - \delta_{El,i})^2\Big)}, \tag{5.5}$$

where $\tilde{\delta}_{Az}$ and $\tilde{\delta}_{El}$ are the predicted offsets, while δ_{Az} and δ_{El} are the true values. N is the number of observations in the test set.

This RMS is used to compare the performance of the models. It will also be compared with a benchmark linear regression model to see if a machine learning approach offers any improvements.

Part III

Results & Discussion

Chapter 6

Results & Discussion

6.1 Results of Experiment 1: Pointing Correction Model

In this section, we present the results obtained from experiment 1 introduced in section 5.1. Prior to presenting the results, we provide a reminder of the RMS ratio measure (5.3), which we frequently used in this section. We used this measure to compare the current pointing model to the machine learning model, and a value less than 1 indicates an improvement of the current model.

Table 6.1 presents the validation and test RMS ratios for all folds of the NFLASH230 model in case 1 and case 2. Recall that models were trained with different numbers of features k, and these results are from the model with the best performance on the validation set (can be different k for each fold). The results show that the model's performance on the validation set is very good in both case 1 and 2. However, this does not generalize well to the test set, with the performance on the test set for case 1 being significantly worse than the current model for all folds. In contrast, for case 2, the performance on the test set is better than the current model for most of the folds. Table 6.2 presents the same results for the model predicting the offsets of all instruments. This model exhibits similar trends, although its performance on the test set is not as good as the NFLASH230 model.

Table 6.1: Validation and test performance measured in RMS ratio (5.2) in case 1 and 2 (see Figure 5.1) for the pointing correction model trained on NFLASH230 data. The performance is given for the model complexity k that yields the best results on the validation data for the given fold.

		Case 1 RMS ratio		Case 2 RMS ratio	
Target	Fold	Validation	Test	Validation	Test
Azimuth	1	0.848	1.188	0.846	1.043
	2	0.841	1.427	0.870	0.962
	3	0.840	1.462	0.923	0.882
	4	0.837	1.266	0.873	0.989
	5	0.846	1.242	0.879	0.944
	6	0.837	1.318	0.907	0.930
Elevation	1	0.835	1.173	0.887	1.030
	2	0.831	1.188	0.889	0.973
	3	0.831	1.204	0.886	1.025
	4	0.812	1.198	0.826	0.844
	5	0.815	1.166	0.802	0.870
	6	0.810	1.262	0.825	0.906

Table 6.2: Validation and test performance measured in RMS ratio (5.2) in case 1 and 2 (see Figure 5.1) for the pointing correction model trained on all data. The performance is given for the model complexity k that yields the best results on the validation data for the given fold.

		Case 1 RMS ratio		Case 2 RMS ratio	
Target	Fold	Validation	Test	Validation	Test
Azimuth	1	0.870	1.642	0.881	1.233
	2	0.861	1.626	0.971	0.928
	3	0.876	1.784	0.897	1.016
	4	0.866	1.613	0.938	0.950
	5	0.862	1.935	0.927	0.942
	6	0.874	1.779	0.923	1.027
Elevation	1	0.832	1.193	0.948	0.965
	2	0.831	1.141	0.924	1.051
	3	0.824	1.129	0.929	1.094
	4	0.816	1.172	0.822	0.922
	5	0.818	1.196	0.828	0.978
	6	0.822	1.186	0.831	0.951

In addition to Table 6.1 and Table 6.2, we also evaluated the performance of the NFLASH230 model with different complexities. Table 6.3 shows the mean RMS ratio (5.3) on the test set, and the associated standard deviation for the azimuth and elevation models when using the same number of features k on all folds. By inspection, we see that the machine learning model does not provide any improvement over the current pointing model for case 1, apart from a slight improvement of an average of 1.8% reduced RMS with a standard deviation of 1.4% for the azimuth model with number of features $k = 2$. However, case 2 shows more promising results.

For azimuth, the best RMS ratio is 0.948 with a standard deviation of 0.056, which is an average improvement of 5.2% reduced RMS on all folds, with a standard deviation of 5.6%. The number of features for these results is $k = 2$. Using $k = 50$ features shows similar results, with 0.945 RMS ratio and a standard deviation of 0.073.

For elevation, the best RMS ratio is 0.940 with a standard deviation of 0.075, using $k = 50$ features. Table 6.4 shows the same results for case 1 and 2, but for the model trained on data from all instruments. We see the same trends, with case 1 showing no improvement and case 2 showing a slight improvement for azimuth and elevation. The best RMS ratio for azimuth in case 2 is 0.980 with a standard deviation of 0.059, using $k = 2$ features. For elevation, the best model is the one with $k = 50$ features, with a RMS ratio of 0.955 and a standard deviation of 0.029. Overall, the elevation models show slightly better results than the azimuth models. Additionally, the model predicting only NFLASH230 offsets performs better than the model predicting offsets from all instruments.

Table 6.3: Resulting mean RMS ratio (5.3) on unseen test sets in case 1 and 2 (see Figure 5.1) for the model trained on NFLASH230 data, using different number of features k in the model.

	Case 1				Case 2			
	Azimuth		Elevation		Azimuth		Elevation	
k	Mean	STD	Mean	STD	Mean	STD	Mean	STD
2	0.982	0.014	1.020	0.024	0.948	0.056	0.972	0.081
5	1.366	0.077	1.198	0.034	0.983	0.142	0.953	0.097
10	1.383	0.087	1.155	0.047	0.957	0.080	0.967	0.087
20	1.252	0.119	1.126	0.071	0.972	0.131	0.949	0.069
30	1.335	0.226	1.094	0.041	0.963	0.093	0.959	0.077
40	1.146	0.036	1.058	0.020	0.961	0.089	0.948	0.077
50	1.202	0.131	1.062	0.022	0.945	0.073	0.940	0.075

Table 6.4: Resulting mean RMS ratio (5.3) on unseen test sets in case 1 and 2 (see Figure 5.1) for the model trained on all data, using different number of features k in the model.

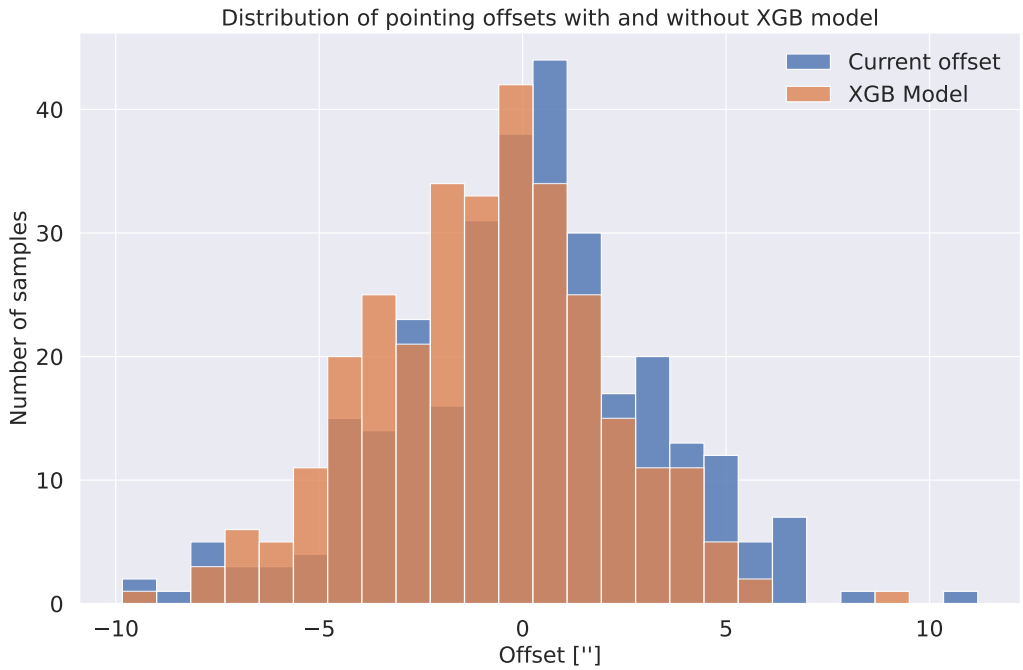
	Case 1				Case 2			
	Azimuth		Elevation		Azimuth		Elevation	
k	Mean	STD	Mean	STD	Mean	STD	Mean	STD
2	1.007	0.003	1.232	0.055	0.980	0.059	0.964	0.016
5	1.003	0.003	1.170	0.028	0.990	0.067	0.964	0.016
10	1.288	0.101	1.116	0.015	1.001	0.102	0.979	0.059
20	1.580	0.082	1.121	0.023	1.018	0.130	0.971	0.036
30	1.606	0.110	1.107	0.018	1.026	0.151	0.957	0.018
40	1.528	0.111	1.068	0.010	1.026	0.137	0.973	0.044
50	1.758	0.121	1.061	0.027	1.018	0.114	0.955	0.029

Table **??** presents an unbiased estimate of the performance of this approach, since we chose the model to use on the test set purely based on performance on the validation set. It shows the mean RMS ratio for the test set on all folds for both models in Tables 6.1 and 6.2. The results indicate that there is no improvement over the current pointing model for case 1. However, for case 2, the model predicting only NFLASH230 offsets shows a small improvement over the current model, with an RMS ratio of 0.958 for azimuth and 0.941 for elevation, both with standard deviations of 0.055 and 0.079, respectively.

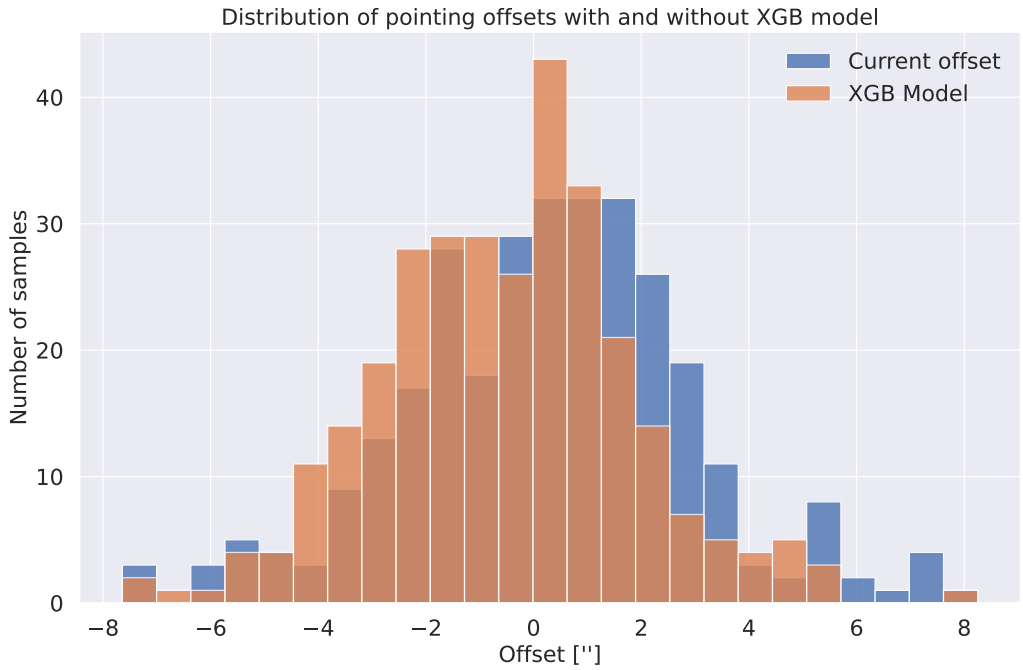
Although the results of case 1 have not shown any improvement over the current pointing model, case 2 has demonstrated potential for improving the pointing accuracy. However, it is important to note that the test data used in the cross-validation process for case 2 is either before or in the middle of the training and validation sets in time, except for the last fold. In the last fold, the test set falls after the training and validation in time, and for the NFLASH230 model, the RMS ratio on this fold was 0.930 for azimuth and 0.906 for elevation, which represents a 7.0% and 9.4% improvement, respectively. In contrast, for all instruments, the RMS ratio was 1.027 for azimuth and 0.951 for elevation, representing a 2.7% worse performance for azimuth and a 4.9% improvement for elevation. This result is the most realistic and unbiased estimate we have on the performance of these models.

For a list of the 50 features with the greatest mutual information to the target variable, please refer to Table A.3 in Appendix A.

Figure 6.1 shows the distribution of NFLASH230 offsets with and without the machine learning model corrections. These are the distribution of offsets of the unseen test set for the last fold.



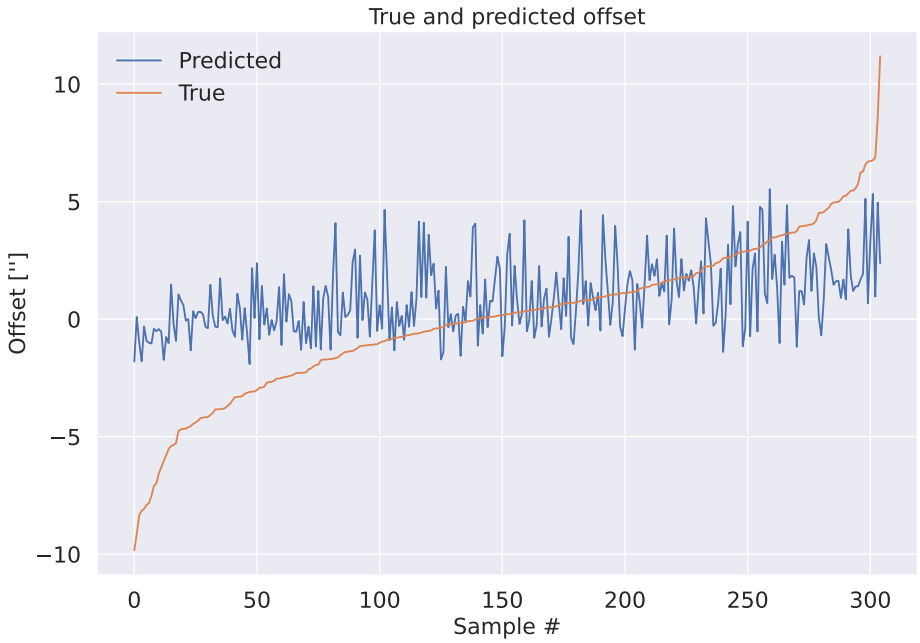
(a) The azimuth model, with offsets reduced on the unseen test set by 7.0%.



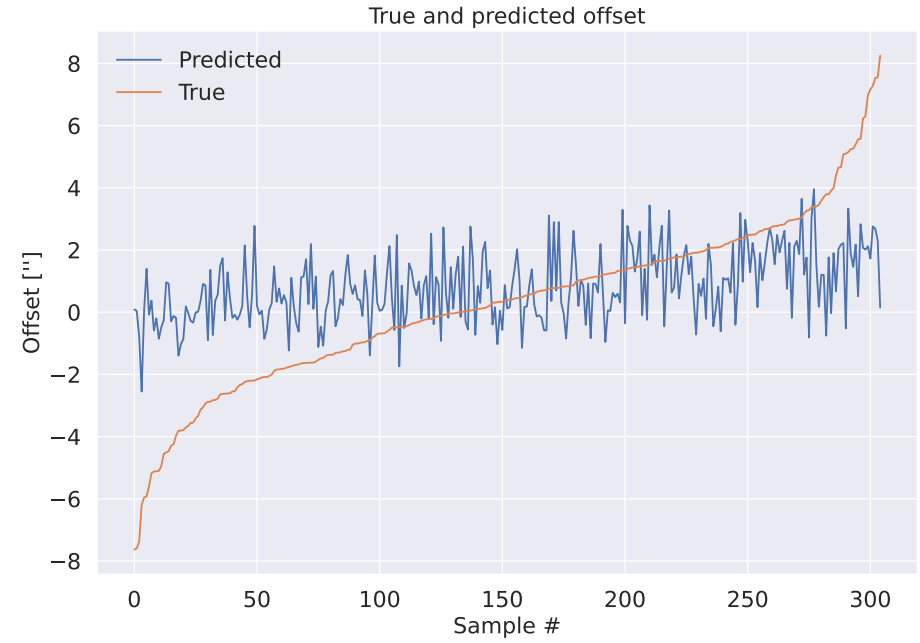
(b) The elevation model, with offsets reduced on the unseen test set by 9.4%.

Figure 6.1: Distribution of offsets with and without the NFLASH230 pointing correction model on the unseen test set for the last fold. **a)** Azimuth model and **b)** elevation model.

Figure 6.2 show the predicted and observed pointing NFLASH230 offsets for all samples in the unseen test set of the last fold. We sorted the samples by the true value in ascending order.



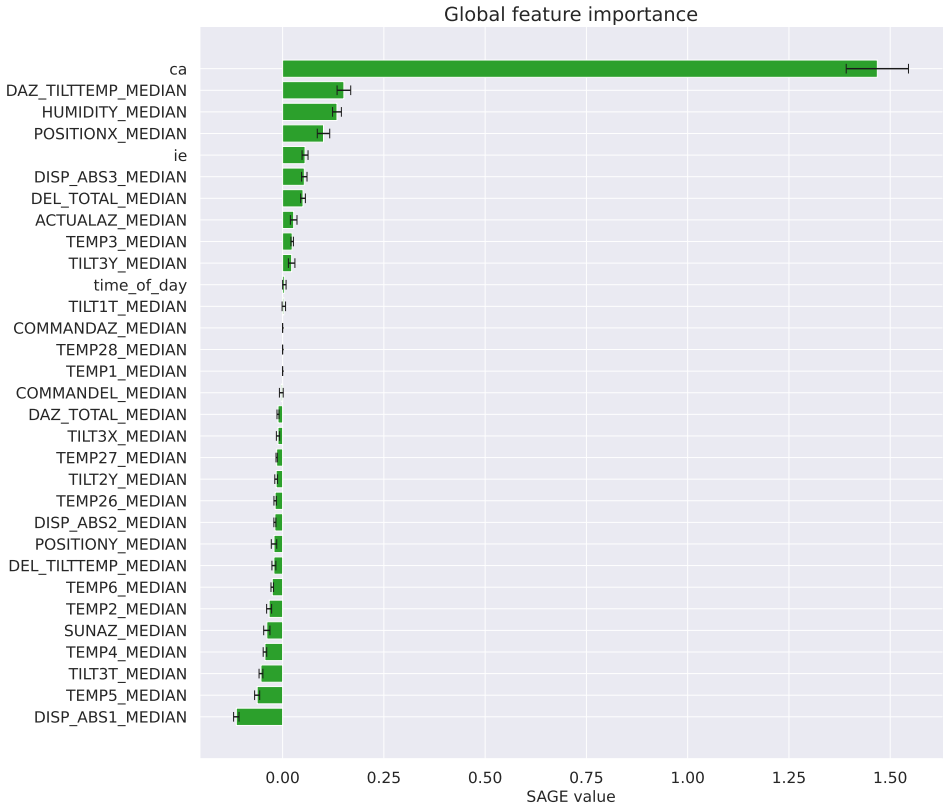
(a) The azimuth model, with offsets reduced on the unseen test set by 7.0%.



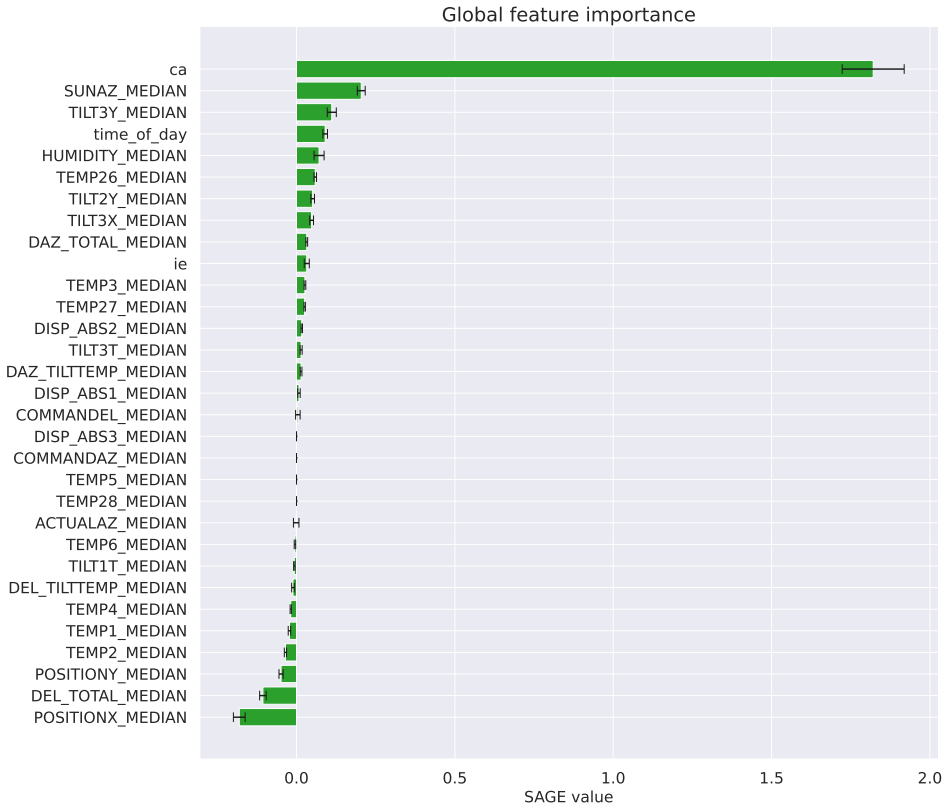
(b) The elevation model, with offsets reduced on the unseen test set by 9.4%.

Figure 6.2: Offset predictions by the NFLASH230 pointing correction model on the unseen test set for the last fold. Predicted and observed values, sorted in ascending order by observed value. **a)** Azimuth model and **b)** elevation model.

Figure 6.3 show the SAGE values for the NFLASH230 azimuth pointing correction model on the validation set (Figure 6.3a) and test set (Figure 6.3b). The same plots for the elevation models are presented in Figure 6.4.

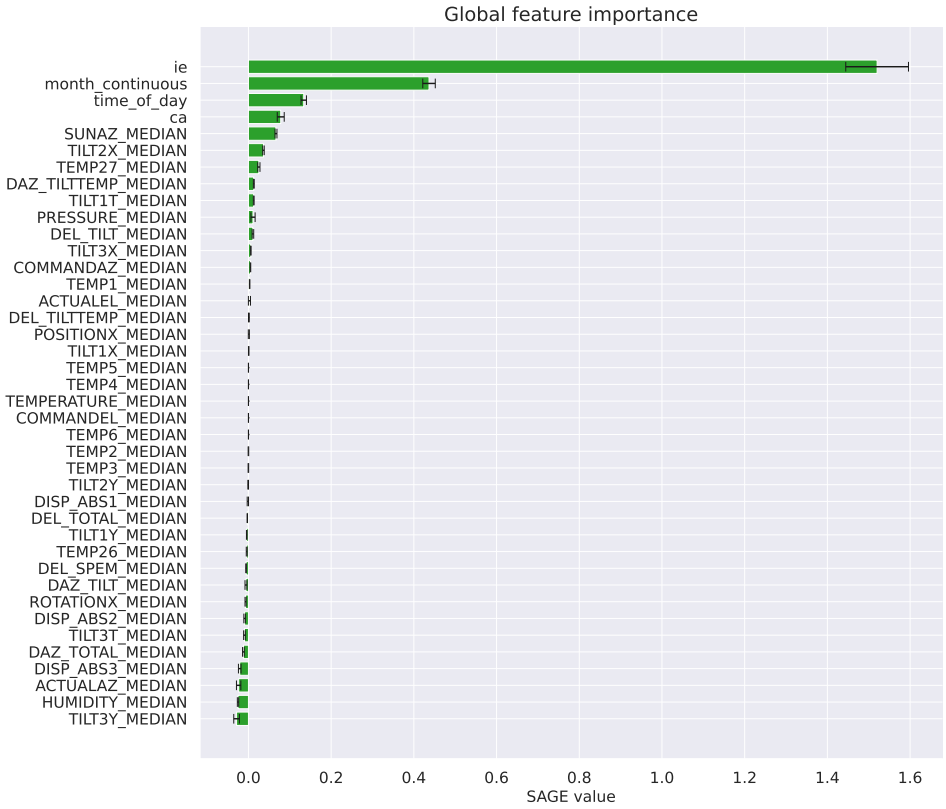


(a) Validation set

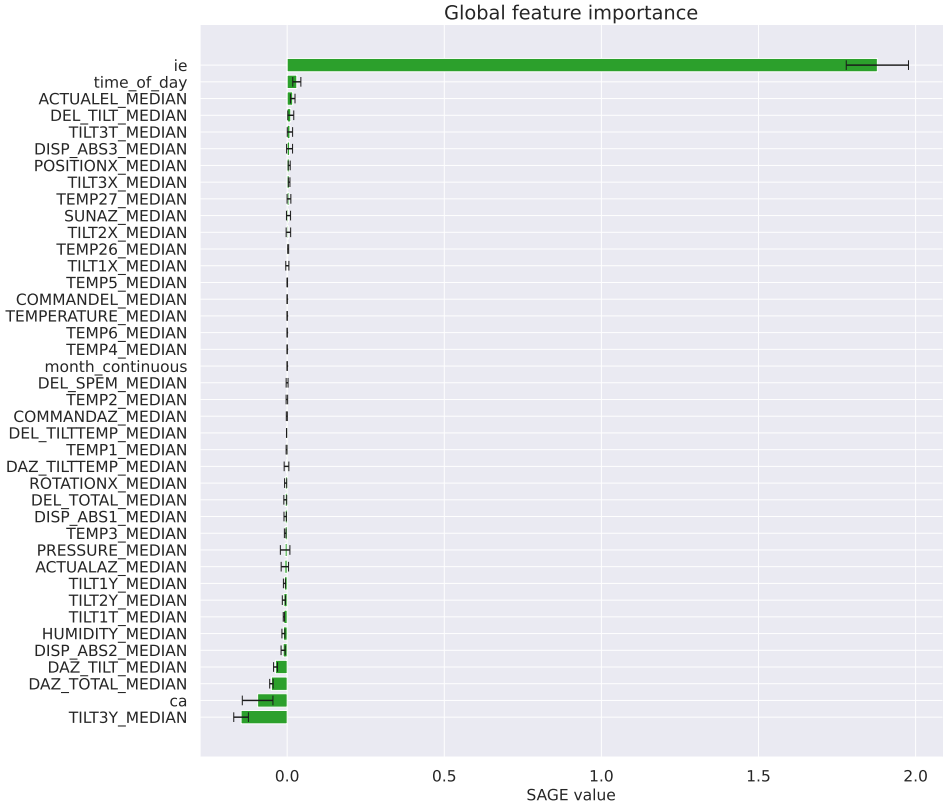


(b) Test set

Figure 6.3: SAGE values for the NFLASH230 azimuth pointing correction model on the a) validation set, and b) test set.



(a) Validation set



(b) Test set

Figure 6.4: SAGE values for the NFLASH230 elevation pointing correction model on the **a)** validation set, and **b)** test set.

6.2 Discussion of Experiment 1: Pointing Correction Model

The first research question addressed in this thesis is whether machine learning can enhance the pointing accuracy of a radio telescope using the current pointing strategy. To investigate this question, we explore a realistic scenario (case 1) in which a model is trained on a smaller period of data and used to predict the offset of consecutive scans for a period afterward. We focus now on the model predicting the offsets of only NFLASH230. The results from this case, presented in Table 6.1, demonstrate that the model's performance on the validation set is promising, with the root-mean-square (RMS) ratio in the range of approximately 0.80-0.85 for azimuth and elevation, which corresponds to a 15-20% reduction in pointing offset. However, this performance does not transfer to the following test period, in which the RMS ratios are 1.16-1.46, indicating a 16-46% increase in pointing offset. There are several possible reasons for the model's poor performance on the test set. One of the limitations of tree-based models, such as XGBoost, is that they typically do not generalize well to new data that is different from the training data, as they predict solely based on logical conditions seen in the training set. If the factors affecting the pointing offset change over time and the new data is very different from the training data, the model will likely perform poorly. Furthermore, another potential explanation for the poor performance could be that the dataset is too small, and the model overfits on the validation set. The results of this experiment suggest that learning the relationships in the data that affect pointing offset is challenging, and a complex model may be necessary. To train a proper complex model, a larger amount of data is required, at least more than the number of samples in the training and validation sets for case 1. The findings also indicate that choosing the complexity of the model with the best performance on the validation set may not necessarily lead to the best performance on the test set. We further explore this aspect by examining Table 6.3, which demonstrates the mean RMS ratio on the test set using the same number of features for all the folds. This provides an idea of the complexity that might provide the best-performing models on the test set. Even though the model with the best performance on the validation set is not chosen, which could be a lucky or overfitted performance, no improvement is observed in the current pointing model on the test set. However, the results show better performance than the first Table 6.1. The same trends are observed when predicting offsets from all instruments in Tables 6.2 and 6.4.

Moving on to case 2, which tests whether the amount of data is a limitation for enhancing the pointing accuracy. This test case is less realistic because the data is split into 6 folds and cross-validation is performed, and for all folds except for the last one, the test period will be either before or between the training/validation set, making the results less applicable in practice. Results from this case indicate that a larger time period helps the model generalize better, which is expected as a larger period includes more variation that can help the model capture the relationships between features. We start by looking at the same table 6.1. Here, we also see a good performance on the validation set across all folds, with a 9-20% reduced pointing offset on the validation set. This shows that more training data helps the model generalize, which is somewhat expected considering that a larger time period includes more variation, which helps the model learn what causes the offsets. The average RMS ratio on all folds on the test set for case 2 is 0.958 for azimuth and 0.941 for elevation. With the standard deviations, the 95% confidence intervals would

be put at $[0.850, 1.066]$ for azimuth and $[0.786, 1.010]$ for elevation. Given that the upper bound of both confidence intervals is larger than one and the testing case is not realistic, we cannot conclude that the model can reduce the pointing offset robustly and consistently. For an unbiased result that reflects expected performance in practice, we look at the RMS ratio for the last fold in Table 6.1, showing a 7.0% and 9.4% reduced RMS for azimuth and elevation, respectively. These results are the ones visualized in the histograms and plots in Figures 6.1 and 6.2, respectively. It is not apparent from the histogram with the azimuth model results that the model performs better than the current model. What we can see, however, is a shift in offsets towards the left, indicating more negative offsets. This is consistent with the sorted prediction plot 6.2a, which shows a bias towards positive predictions. There also seem to be slightly less large negative offsets. Looking at the elevation histogram 6.1b and plot 6.2b, we can make the same analysis. The histogram shows a shift towards the left, and the sorted prediction plot shows a bias toward positive predictions. We do also see that there are more offsets in the center bin in the histogram (meaning more smaller offsets). The tails in the sorted prediction plots also seem steep, indicating that the cleaning could have been better.

We now discuss the results of the SAGE analysis on the validation and test sets for the model discussed in the previous paragraph, as shown in Figures 6.3 and 6.4. This analysis helps us understand the factors that improve and worsen the models' performance on unseen data, thereby providing insights into how the models can be improved.

For the azimuth model, we observe that the azimuth correction *ca* is the most important feature, contributing significantly to the model's performance on both the validation and test sets. However, the elevation correction *ie* shows a mixed effect on the model's performance. While it improves the performance on the validation set, it worsens the performance on the test set. This indicates that there are some correlation between the azimuth and elevation correction that change over time. Notably, the COMMANDAZ and COMMANDEL features, which are included in every model, do not affect the azimuth or elevation model's performance. This suggests that the analytical pointing model, which is based solely on azimuth and elevation, is performing very well, and the sources of offsets are likely attributed to factors other than the pointing direction. We also find that the continuous month feature is the second most important feature in the elevation model, after the elevation correction *ie*. However, the model's performance on the test set does not benefit from this feature, as its value is larger in the test set than in the training and validation sets. This is one of the weaknesses of tree-based models. Training the model on the previous years' data could solve this problem, as long as there are not other big changes in the structure or mounting of the instruments.

In summary of the results from case 2, we see promising results with machine learning models further reducing the offsets slightly. This indicates that a possible pointing strategy could be training a model on multiple months worth of data and then using the model for a couple of weeks. However, given the limited data available in this project, this could not be tested thoroughly. If more data were available, a similar analysis could be performed with the start and end time of both the train/validation and test set moved by two weeks and iteratively train new models to predict the offsets for the next time period. This could verify whether the improved performance repeats.

In case 2, the model predicting offsets from all instruments shows results similar to the model predicting only NFLASH230 offsets, though with slightly worse performance. The reduced performance is likely due to less training data available for the other instruments. The difference in performance between these two models provides some insights. First, it suggests that various factors influence the different instruments differently. Otherwise, we expect the larger model to perform better than the smaller model. The reason that each instrument can be affected differently is because of the different mounting locations at the telescope. Different mounting locations mean that the path of photons leading to the receiver can vary, leading to distinct factors affecting the pointing.

There are several measures we could take to enhance the analysis further. One such measure is to improve feature engineering. When the quantity of training data is limited, it is vital to ensure that the features are as informative as possible. For example, instead of including six distinct temperature measurements in a model, it may be preferable to create terms based on the difference between multiple temperature measurements, as they did in [28].

Another step is to integrate corrections for ca and ie into the offset that we intend to predict. These corrections are applied at each pointing scan and display the strongest correlation to the target offset of all the features, and we, therefore, incorporate them in all the models. By transforming the offsets $\delta_{az} = \delta_{az} + ca$ and $\delta_{el} = \delta_{el} + ie$, we can then eliminate ca and ie from the input to the models. This could remove a layer of complexity for the model.

Another option to consider for predicting the pointing offsets is to use neural networks. These models offer several advantages if trained successfully. For instance, they can handle multiple outputs and thus only require a single model. By training the neural network with two outputs, it can simultaneously consider the offsets in azimuth and elevation and explore the correlation between them. It would also be beneficial to fine-tune the network as new pointing scans become available continuously. Despite their potential benefits, our initial tests with neural networks did not yield satisfactory results, likely due to the limited training data available. Neural networks also require more tuning, which makes them more difficult to use with limited datasets.

To further improve the analysis, there are other possible areas for exploration, such as minimizing the number of pointing scans conducted by the astronomers. While performing scans every one to two hours is standard practice, this can be time-consuming and may disrupt science observations. A possible approach to investigate this possibility is to identify periods when pointing scans are conducted every hour or so. Rather than providing the model with the corrections ca and ie obtained following the previous pointing scan, the model can be fed with the corrections from a pointing scan that occurred six hours earlier to evaluate its performance. In this case, reducing pointing offsets is not a primary objective of the machine learning approach. Instead, maintaining the same level of pointing accuracy while conducting fewer pointing scans would be a satisfactory outcome.

6.3 Results of Experiment 2: Pointing Model using Neural Networks

Table 6.5 shows the RMS in arcseconds on the test set overall folds for the different model architectures. The models presented in this table are the ones with the lowest mean RMS across all test folds. A linear regression model is included for comparison. It is clear that neural network architectures significantly outperform the linear regression model. From the mean RMS overall folds, we see that the different neural network architectures offer similar performance. We also see that the RMS of fold 1 is by far worse than the other folds. The lowest mean RMS is from the architecture where the non-linear features are connected to the geometrical and harmonic features.

Table 6.5: The RMS of the pointing model on the test set in each fold for all neural network architectures. The different architectures are described in Figure 5.2.

	RMS [″] on test fold							
Network	1	2	3	4	5	6	Mean	STD
Regular	28.06	19.34	12.28	13.25	17.19	16.33	17.74	5.19
Separated 1	30.69	16.93	13.76	10.04	15.77	13.61	16.80	6.57
Separated 2	27.34	20.75	12.65	24.17	13.64	16.69	19.21	5.38
Separated 3	30.27	20.59	14.01	10.76	13.67	10.34	16.61	6.97
Linear regression	70.55	40.19	51.17	49.69	36.86	37.29	47.63	12.81

Table 6.6 shows the hyperparameter used for the best models. All architectures perform better with a single hidden layer. The regular neural network uses ReLU activation and MSE loss, while the other architectures use Tanh activation and MSD loss. The regular neural network also has more neurons in the hidden layer and a higher learning rate. The batch size is also varying. There seem to be similarities between the hyperparameters chosen for the three architectures with separate features. However, given the large standard deviation of the mean RMS, there are probably bigger issues than hyperparameter tuning.

Table 6.6: Hyperparameters used in the best-performing model for each architecture.

Architecture	Activation	Hidden Layers	Learning Rate	Batch Size	Loss
Regular	ReLU	[82]	0.0199	334	MSE
Sep1	Tanh	[40]	0.0098	101	MSD
Sep2	Tanh	[40]	0.0098	101	MSD
Sep3	Tanh	[26]	0.0039	358	MSD

Table 6.7 lists the features selected for the best-performing models.

6.4 Discussion of Experiment 2: Pointing Model using Neural Networks

To address the second research question of whether machine learning models can replace the traditional analytical linear regression models commonly used in radio/(sub-

Table 6.7: Features used in the best-performing model for each architecture.

Feature	Sep 1	Sep 2	Sep 3	Regular
COMMANDAZ	x	x	x	x
COMMANDEL	x	x	x	x
DISP ABS3	x	x	x	x
CA	x	x	x	
NPAE	x	x	x	
Constant	x	x	x	
$\cos(\text{COMMANDEL})$	x	x	x	
$\cos(2 \cdot \text{COMMANDEL})$	x	x	x	
$\cos(3 \cdot \text{COMMANDEL})$	x	x	x	
$\cos(4 \cdot \text{COMMANDEL})$	x	x	x	
$\cos(5 \cdot \text{COMMANDEL})$	x	x	x	
$\sin(\text{COMMANDEL})$	x	x	x	
$\sin(2 \cdot \text{COMMANDEL})$	x	x	x	
$\sin(3 \cdot \text{COMMANDEL})$	x	x	x	
$\sin(4 \cdot \text{COMMANDEL})$	x	x	x	
$\sin(5 \cdot \text{COMMANDEL})$	x	x	x	
Turbulence	x	x	x	
DEL TILTTEMP	x	x	x	
DAZ DISP			x	
POSITIONY			x	
TILT1Y			x	
TEMPERATURE			x	
TILT1T			x	

)mm telescopes, we utilized raw data from the apex telescope containing input and observed coordinates for azimuth and elevation. Typically, linear regression models are fitted to this raw data using multiple geometrical and empirical terms. As a benchmark, we used a linear regression model and compared its performance against four different neural network architectures (see Figure 5.2). Our goal was to determine if a machine learning approach could outperform the linear regression model and, if so, which type of architecture performs the best. We employed cross-validation with six folds and found that all neural network architectures significantly outperformed the linear regression model. The mean RMS of the linear regression model was $47.63''$ with a standard deviation of $12.81''$, whereas all neural network architectures fell within the range of $16''$-$19''$ with deviations of $5''$-$7''$. Although all the neural network architectures had similar performance and standard deviation, we cannot conclude that any of them is better than the others. Overall, our results indicate that machine learning models can outperform traditional analytical models for pointing offset prediction in radio telescopes, and we encourage further research in this area to explore the full potential of machine learning techniques.

Analysis of the hyperparameters selected for the top-performing models reveals that most of them are relatively simple in structure. Interestingly, none of the models employed two hidden layers. Additionally, the architectures incorporating distinct layers for the terms employed in the analytical pointing models exhibited lower neuron counts in their nonlinear layers. This is likely because the level of complexity required for the input features provided to the nonlinear layers is assumed to be lower when other aspects of the model are responsible for handling certain factors. Thus, the observed pattern of hyperparameter selection is consistent with the expectation based on architecture.

Table 6.7 presents the features selected by each model. All the models included COMMANDAZ and COMMANDEL as inputs for the nonlinear layers. The "Separated" neural networks incorporated the harmonic features as inputs in the linear part of the network, shown in Figure 5.2. We sampled the remaining features randomly. Interestingly, the models used few nonlinear features, indicating that the factors influencing pointing error are difficult to model with the current training set. Obtaining more data may be necessary to establish more complex relationships between features that can improve the performance of the pointing models. Notably, the regular neural network utilized only one additional feature, namely DISP_ABS (which still requires explanation). This finding suggests that a neural network utilizing only the azimuth and elevation angle could outperform a linear regression model. However, linear regression models have advantages in terms of transparency and robustness, as they are inherently linear, and all the features in the model are independent of each other, resulting in a model that is unlikely to result in significant pointing errors. This is an important consideration when using machine learning models, which can be challenging to interpret. Their robustness must be thoroughly tested before being deployed in practical settings. These considerations also underlie the design of the "Separated" neural network architectures. We wanted to utilize the robustness of linear regression on the terms used in the current analytical pointing model while incorporating new features. Considering that the experiment demonstrated negligible performance differences between the separate neural networks and the regular one and that the best models only choose a few features, we conclude that additional

training data is required to see if this is a feasible approach.

While the experiment did not demonstrate the performance that would be expected from using such models in practice, it indicates that a neural network could provide a more efficient and effective model than a linear regression model. Moreover, a neural network can be fine-tuned with newly obtained data, making it an appealing model. If this approach is shown to be reliable and high-performing, it could be the preferred choice for more advanced telescopes, as it may require less manual analysis to develop and maintain a pointing model. However, further testing with additional data is needed to confirm the feasibility of this approach.

Chapter 7

Conclusion

7.1 Experiment 1: Pointing Correction Model

In conclusion, our study addressed the research question of whether machine learning could enhance the pointing accuracy of a radio telescope. Our findings suggest that while machine learning has the potential to improve pointing accuracy, a more extensive dataset and a complex model may be necessary for consistent and robust performance. We tested two cases, one where we trained on a smaller period and tested on unseen data in a consecutive period, and another where we trained and tested on a longer period. The results indicated that a longer period or more training data helps the model generalize better, and the NFLASH230 model provided a reduction in offset for azimuth and elevation. However, given that the testing case is not entirely realistic, we cannot conclude that the model can reduce pointing offset in a robust and consistent manner. Therefore, further research is needed to verify the performance improvements of such a strategy.

Furthermore, our study highlighted several potential avenues for future research to improve the accuracy and efficiency of pointing offset prediction in radio telescopes using machine learning techniques. One key area for improvement is feature engineering, where more informative features could be created to enhance model performance, especially when dealing with limited training data. We also suggest exploring neural networks, which offer advantages such as handling multiple outputs and continuous fine-tuning as new data becomes available. Although our initial tests did not yield satisfactory results, further investigation is necessary with more extensive training data to explore the potential of using neural networks for pointing offset prediction in radio/(sub-)mm telescopes. Finally, minimizing the number of pointing scans conducted by astronomers while maintaining similar pointing accuracy is another potential area for exploration. Overall, our study provides insights into future research directions to optimize the performance of machine learning models for pointing offset prediction in radio telescopes and highlights the potential for machine learning to improve this critical aspect of radio telescope operations.

7.2 Experiment 2: Pointing Model using Neural Networks

In this study, we investigated whether machine learning models can replace traditional analytical linear regression models for predicting pointing offsets in radio/(sub-)mm telescopes. To address this question, we used raw data from the APEX telescope

containing input and observed coordinates for azimuth and elevation. We fitted a linear regression model to the raw data and compared its performance against four different neural network architectures using cross-validation with six folds.

Our results showed that all neural network architectures significantly outperformed the linear regression model, with mean RMS falling within the range of $16''$-$19''$ with deviations of $5''$-$7''$, while the mean RMS of the linear regression model was $47.63''$ with a standard deviation of $12.81''$. Although all the neural network architectures had similar performance and standard deviation, we cannot conclude that any of them is better than the others.

The top-performing models we analyzed had relatively simple structures, with most of them not employing two hidden layers. Interestingly, the architectures incorporating distinct layers for the terms employed in the analytical pointing models exhibited lower neuron counts in their nonlinear layers. We also found that the models used few features for the nonlinear layers, suggesting that the factors influencing pointing error are difficult to model with the current data set. Obtaining more data may be necessary to establish more complex relationships between features that can improve the performance of the pointing models.

While our experiment did not demonstrate the performance that would be expected from using such models in practice, it indicates that a neural network could provide a more efficient and effective model than a linear regression model. Moreover, a neural network can be fine-tuned with newly obtained data, making it an appealing model. If this approach is shown to be reliable and high-performing, it could be the preferred choice for more advanced telescopes, as it may require less manual analysis to develop and maintain a pointing model. However, further testing with additional data is needed to confirm the feasibility of this approach.

Part IV

Appendices

Appendix A

Results & Tables

Table A.1: Performance when choosing min validation for the last fold.

	Case 1		Case 2	
Dataset	Azimuth	Elevation	Azimuth	Elevation
Transformed	2.016	1.067	1.050	0.926
Transformed NF230	1.243	1.009	0.954	1.008
Regular	1.779	1.186	1.027	0.951
Regular N230	1.204	1.262	0.930	0.906

Table A.2: Performance when choosing min validation for each fold. Train/val split on days. Test size 0.43.

	Case 1				Case 2			
	Azimuth		Elevation		Azimuth		Elevation	
Dataset	Mean	STD	Mean	STD	Mean	STD	Mean	STD
Transformed	2.140	0.498	1.073	0.040	1.008	0.073	0.949	0.022
Transformed NF230	1.242	0.047	1.006	0.009	0.999	0.096	1.074	0.201
All instruments	1.730	0.126	1.170	0.028	1.016	0.114	0.994	0.065
Only NFLASH230	1.251	0.131	1.198	0.033	0.958	0.055	0.941	0.079

A.1 Raw data correlation

Table A.3: The 50 features with the greatest mutual information to the target value (unsorted).

	All Instruments		NFLASH230	
Features	Az	El	Az	El
ACTUALAZ_MEDIAN	x	x	x	x
ACTUALEL_MEDIAN	x	x	x	x
COMMANDAZ_MEDIAN	x	x	x	x
COMMANDEL_MEDIAN	x	x	x	x
DAZ_DISP_MEDIAN	x		x	x
DAZ_TILTTEMP_MEDIAN	x	x	x	x
DAZ_TILT_MEDIAN	x	x	x	x
DAZ_TOTAL_MEDIAN	x	x	x	x
DEL_DISP_MEDIAN	x	x	x	x
DEL_SPEM_MEDIAN	x	x	x	x
DEL_TILTTEMP_MEDIAN	x	x	x	x
DEL_TILT_MEDIAN	x	x	x	x
DEL_TOTAL_MEDIAN	x	x	x	x
DEWPOINT_MEDIAN	x	x	x	x
DISP_ABS1_MEDIAN	x	x	x	x
DISP_ABS2_MEDIAN	x	x	x	x
DISP_ABS3_MEDIAN	x	x	x	x
DSUNAZ_CHANGE_P5	x	x	x	x
HUMIDITY_MEDIAN	x	x	x	x
POSITIONX_MEDIAN	x	x	x	x
POSITIONY_MEDIAN	x	x	x	x
POSITIONZ_MEDIAN	x	x	x	
PRESSURE_MEDIAN	x	x	x	x
ROTATIONX_MEDIAN	x	x		x
SUNAZ_MEDIAN	x	x	x	x
SUNEL_MEDIAN			x	x
TEMP1_MEDIAN	x	x	x	x
TEMP26_MEDIAN	x	x	x	x
TEMP27_MEDIAN	x	x	x	x
TEMP28_MEDIAN	x	x	x	x
TEMP2_MEDIAN	x	x	x	x
TEMP3_MEDIAN	x	x	x	x
TEMP4_MEDIAN	x	x	x	x
TEMP5_MEDIAN	x	x	x	x
TEMP6_MEDIAN	x	x	x	x
TEMPERATURE_MEDIAN	x	x		x
TILT1T_MEDIAN	x	x	x	x
TILT1X_MEDIAN	x	x	x	x
TILT1Y_MEDIAN	x	x	x	x
TILT2T_MEDIAN	x	x	x	x
TILT2X_MEDIAN	x	x	x	x
TILT2Y_MEDIAN	x	x	x	x
TILT3T_MEDIAN	x	x	x	
TILT3X_MEDIAN	x	x	x	x
TILT3Y_MEDIAN	x	x	x	x
WINDDIRECTION_MEDIAN		x	x	x
WINDSPEED_MEDIAN	x	x		
WINDSPEED_VARIANCE_P5	x	x	x	x
ca	x	x	x	x

Table A.4: All

Target	Fold	Colsample by tree	γ	η	Max Depth	Min child weight	Number of estimators	λ	Subsample
Az	1	0.536	0.013	0.018	5	8	269	1	0.699
	2	0.526	0.013	0.010	1	10	393	0.175	0.756
	3	0.970	0.063	0.023	2	2	460	0.071	0.800
	4	0.501	0.008	0.019	3	2	400	0.578	0.979
	5	0.530	0.031	0.187	2	3	69	0.277	0.931
	6	0.668	0.056	0.139	2	2	77	0.426	0.783
El	1	0.577	0.247	0.017	5	1	111	0.966	0.517
	2	0.645	0.078	0.063	5	2	33	0.010	0.787
	3	0.600	0.017	0.009	5	5	86	0.660	0.598
	4	0.608	0.019	0.037	5	10	368	0.329	0.600
	5	0.999	0.031	0.022	4	1	427	0.178	0.716
	6	0.788	0.369	0.097	2	1	354	0.552	0.796

Table A.5: NFLASH230

Target	Fold	Colsample by tree	γ	η	Max Depth	Min child weight	Number of estimators	λ	Subsample
Az	1	0.720	0.125	0.126	1	4	201	0.247	0.864
	2	0.777	0.988	0.017	1	6	468	0.763	0.962
	3	0.517	0.298	0.014	2	7	470	0.509	0.762
	4	0.972	0.773	0.116	5	1	30	0.006	0.802
	5	0.897	0.112	0.008	5	10	435	0.658	0.819
	6	0.935	0.011	0.064	1	6	224	0.228	0.748
El	1	0.600	0.156	0.007	5	2	140	0.047	0.792
	2	0.999	0.148	0.029	5	1	38	0.998	0.580
	3	0.955	0.237	0.014	3	6	210	0.613	0.712
	4	0.537	0.014	0.034	2	10	352	0.537	0.594
	5	0.686	0.324	0.081	1	1	371	0.368	0.966
	6	0.862	0.306	0.184	1	10	157	0.029	0.701

Table A.6: NFLASH230

Fold	Colsample by tree	γ	η	Max Depth	Min child weight	Number of estimators	λ	Subsample
Azimuth								
1	0.720	0.125	0.126	1	4	201	0.247	0.864
2	0.777	0.988	0.017	1	6	468	0.763	0.962
3	0.517	0.298	0.014	2	7	470	0.509	0.762
4	0.972	0.773	0.116	5	1	30	0.006	0.802
5	0.897	0.112	0.008	5	10	435	0.658	0.819
6	0.935	0.011	0.064	1	6	224	0.228	0.748
Elevation								
1	0.600	0.156	0.007	5	2	140	0.047	0.792
2	0.999	0.148	0.029	5	1	38	0.998	0.580
3	0.955	0.237	0.014	3	6	210	0.613	0.712
4	0.537	0.014	0.034	2	10	352	0.537	0.594
5	0.686	0.324	0.081	1	1	371	0.368	0.966
6	0.862	0.306	0.184	1	10	157	0.029	0.701

Table A.7: Validation and test performance for Case 2, all instruments left and nflash230 right. Performance when choosing the number of features showing best performance.

Target	Fold	Val RMS 1	Test RMS 1	Val RMS 2	Test RMS 2
Az	1	0.915	1.084	0.846	1.043
	2	0.971	0.928	0.886	0.953
	3	0.927	1.011	0.928	0.872
	4	0.945	0.952	0.882	0.962
	5	0.933	0.972	0.898	0.938
	6	0.937	0.935	0.930	0.923
El	1	0.964	0.975	0.916	1.014
	2	0.930	1.000	0.889	0.973
	3	0.966	0.957	0.886	1.025
	4	0.822	0.922	0.839	0.839
	5	0.830	0.934	0.802	0.888
	6	0.836	0.941	0.827	0.901

Table A.8: Validation and test performance for Case 1 and 2, all instruments. Performance when choosing the model complexity that yields the best results on the validation set for the given fold.

Target	Fold	Val RMS 1	Test RMS 1	Val RMS 2	Test RMS 2
Az	1	0.870	1.642	0.881	1.233
	2	0.861	1.626	0.971	0.928
	3	0.876	1.784	0.897	1.016
	4	0.866	1.613	0.938	0.950
	5	0.862	1.935	0.927	0.942
	6	0.874	1.779	0.923	1.027
El	1	0.832	1.193	0.948	0.965
	2	0.831	1.141	0.924	1.051
	3	0.824	1.129	0.929	1.094
	4	0.816	1.172	0.822	0.922
	5	0.818	1.196	0.828	0.978
	6	0.822	1.186	0.831	0.951

Table A.9: Validation and test performance for Case 1 and 2, only NFLASH230. Performance when choosing the model complexity that yields the best results on the validation set for the given fold.

Target	Fold	Val RMS 1	Test RMS 1	Val RMS 2	Test RMS 2
Az	1	0.848	1.188	0.846	1.043
	2	0.841	1.427	0.870	0.962
	3	0.840	1.462	0.923	0.882
	4	0.837	1.266	0.873	0.989
	5	0.846	1.242	0.879	0.944
	6	0.837	1.318	0.907	0.930
El	1	0.835	1.173	0.887	1.030
	2	0.831	1.188	0.889	0.973
	3	0.831	1.204	0.886	1.025
	4	0.812	1.198	0.826	0.844
	5	0.815	1.166	0.802	0.870
	6	0.810	1.262	0.825	0.906

Table A.10: Features with Spearman's rank correlation ≥ 0.1 to either one of the target values.

Feature	δ_{Az}	δ_{El}	$\delta_{\mathrm{Az}} \cos \mathrm{El}$
WINDSPEED_VAR_5	0.10	0.12	0.11
DAZ_TILT_MEDIAN_1	0.09	0.03	0.14
DAZ_TILTTEMP_MEDIAN_1	0.00	-0.06	0.30
TILT1Y_MEDIAN_1	0.08	0.02	0.13
TEMP26_MEDIAN_1	0.06	0.13	−0.19
TEMP27_MEDIAN_1	0.06	0.13	−0.21
TEMP28_MEDIAN_1	0.04	0.11	−0.25
TEMPERATURE_MEDIAN_1	0.04	0.12	−0.26
POSITIONZ_MEDIAN_1	0.05	0.11	−0.21
DEWPOINT_MEDIAN_1	0.08	0.12	−0.15
DAZ_TOTAL_MEDIAN_1	0.09	0.03	0.13
WINDSPEED_MEDIAN_1	0.02	0.02	0.13
HESE2	0.46	0.44	0.44
HESE3	0.98	0.94	0.77
HESE4	0.97	0.92	0.74
HESE5	0.34	0.31	0.15
HACA5	0.06	0.05	0.10
HECE	1.00	0.96	0.78
HECE2	1.00	0.96	0.78
HECE3	0.76	0.72	0.52
HSCA5	0.12	0.11	0.14

Table A.11: Features with Pearson's correlation ≥ 0.1 to either one of the target values.

Feature	δ_{Az}	δ_{El}	$\delta_{\mathrm{Az}} \cos \mathrm{El}$
TEMP26_MEDIAN_1	0.05	0.11	−0.12
TEMP27_MEDIAN_1	0.05	0.11	−0.13
DAZ_TOTAL_MEDIAN_1	0.12	0.08	0.07
WINDSPEED_MEDIAN_1	0.03	0.03	0.10
AWAZ	0.00	−0.01	0.57
HESE2	0.56	0.54	0.45
HESE3	0.97	0.90	0.43
HESE4	0.91	0.82	0.26
HESE5	0.29	0.22	−0.11
HECE	1.00	0.92	0.45
HECE2	0.98	0.89	0.38
HECE3	0.70	0.62	0.10
HSCA	0.01	−0.00	0.53

Appendix B

Additional Methods

B.1 Transformation of Pointing Offsets and Corrections

Table B.1 show examples of pointing offsets and corrections applied during the pointing scans. The column "Original" show raw pointing scan data. The pointing corrections *ca* and *ie* are normally updated according to equations (2.17) and (2.18), as shown in the first two rows of the table. However, observers may choose not to update the pointing, particularly when the pointing offset is small, as illustrated in the consecutive row of the table. In other cases, the corrections may be updated but not according to equations (2.17) and (2.18). This can occur when a new science project is loaded and the pointing correction from the previous time that project was used is applied. These factors introduce several challenges for the training of a model:

- *ca* and *ie* should represent the optimal correction using all the information we have about the current state of a system. If we do not update the corrections, there is some information about the system (the previously observed pointing offset) that the model is not receiving.
- Some features are constructed as the change in variables since the last correction. If the corrections are not updated, this interval is longer than if they were, and the resulting features could be more prone to uncertainties and noise. A problem with the integration also occurs if the corrections are not updated according to the equations (2.17) and (2.18). Then, we do not know when those corrections represent the system, resulting in inaccurate features.

A possible solution to this problem is a two step procedure where we first transform the offsets and corrections to represent the system at the most recent pointing scan, and second use these transformed offsets as training labels and the transformed corrections as training inputs.

In practice, we do this by assuming the corrections *ca* and *ie* are updated after every pointing scan. This changes the correction applied during the next pointing scan, which further affects the observed pointing offset. This effect propagates throughout the whole dataset. The following formulas

$$\tilde{ca}_i = \tilde{ca}_{i-1} + \tilde{\delta}_{az,i-1} \tag{B.1}$$

$$\tilde{ie}_i = \tilde{ie}_{i-1} - \tilde{\delta}_{el,i-1} \tag{B.2}$$

$$\tilde{\delta}_{az,i} = \delta_{az,i} + ca_i - \tilde{ca}_i \tag{B.3}$$

$$\tilde{\delta}_{el,i} = \delta_{el,i} - ie_i + \tilde{ie}_i \tag{B.4}$$

Where the "$\sim$" denotes a transformed variable. Using these transformations, the corrections used for training and the resulting offset are similar to the ones that would be observed if the corrections were made according to the equations (2.17) and (2.18) after every pointing scan. The column "Transformed" in the table shows the transformed variables.

Table B.1: **Original:** Example from the dataset of the observed pointing offsets and the corrections applied during the pointing scan. **Transformed:** Pointing offsets and corrections according to equations (B.1), (B.2), (B.3), and (B.4).

	Original				Transformed			
i	δ_{az}	δ_{el}	ca	ie	$\tilde{\delta}_{az}$	$\tilde{\delta}_{el}$	$\tilde{ca}$	$\tilde{ie}$
1	1.2	0.1	2.1	1.7	1.2	0.1	2.1	1.7
2	0.0	0.5	3.3	1.6	0.0	0.5	3.3	1.6
3	−1.1	0.0	3.3	1.6	−1.1	−0.5	3.3	1.1
4	0.6	0.7	2.2	1.6	0.7	0.7	2.2	1.6
5	0.9	1.4	2.2	1.6	0.2	0.7	2.8	0.9
6	1.0	1.1	2.2	1.6	0.1	−0.3	3.1	0.2
7	−0.9	1.2	3.1	0.5	−1.0	1.3	3.2	0.5
8	0.5	1.5	2.2	−0.7	0.5	1.4	2.2	−0.7
9	−0.3	0.4	2.2	−0.7	−0.8	−1.1	2.7	−2.2

B.2 Feature Engineering for Transformed Offsets and Corrections

There are two main features engineered for this project; features that represent the system during a pointing scan and features that represent changes since the last correction. The idea behind this is simple. The correction used during a pointing scan represents the ideal correction for the system during the previous pointing scan. As there are a lot of factors and complex relationships, and we do not have large amounts of training data, it might be easier for the model to learn how these changes affect the pointing rather than learning all the relationships.

B.2.1 Feature Transformation

Median values

The median value of variables during a pointing scan is the most used feature.

Sum of all change

To capture systematic error in pointing due to the telescope moving back and forth in azimuth and elevation, we sum over the positive and negative changes in these variables.

Given the time of the last pointing correction t_1 and the start of a pointing scan t_2, the sum over the positive changes in a variable x_i is given by

$$X = \sum_{i=t_1+1}^{t_2} \max(0, x_i - x_{i-1}) \tag{B.5}$$

Similarly, the sum of negative changes in a variable is

$$X = \sum_{i=t_1+1}^{t_2} \min(0, x_i - x_{i-1}) \tag{B.6}$$

We make these features with azimuth and elevation.

Change since the last correction

This feature is self-explanatory and is just the change in a variable since the pointing was corrected.

$$\Delta x = x_{t_2} - x_{t_1} \tag{B.7}$$

In order to make this feature more robust against noisy data, we instead consider the change in the median for a time interval around the last correction t_1 and the start of a pointing scan t_2

$$\Delta x = median(x_{t_2}, x_{t_2-1}, \ldots, x_{t_2-p}) - median(x_{t_1}, x_{t_1+1}, \ldots, x_{t_1+p}), \tag{B.8}$$

where p is the number of data points needed to cover a period of P minutes, given by $p = P \cdot frequency$. The unit of frequency is data points per minute, found in Table 2.3.

Max change in time interval

In case the speed of the temperature change affects the deformation of the telescope's structure, we find the maximum temperature change in a given time interval since the last pointing correction.

$$X = \max(x_{t_1+p} - x_{t_1}, x_{t_1+p} - x_{t_1}, \ldots, x_{t_2} - x_{t_2-p}), \tag{B.9}$$

Position of the sun

Observers at the telescope report that the sun is affecting the pointing. It is most drastically affected when the sun sets or rises, likely due to rapid temperature change leading to deformation in the telescope structure. We also think the sun's position affects the pointing. For instance, if the sun is shining on the left side of the telescope, it will affect the pointing differently than if it is on the right side. Obtaining the sun's position for the telescope's location is done using the python module PyEphem [19].

Using the azimuth angle of the sun and the telescope, we can calculate the position of the sun with respect to the pointing with

$$\Delta Az_\odot = Az_t - Az_\odot \tag{B.10}$$

This will result in values outside the $[-180°, 180°]$. An example is if $Az_\odot = 179°$ and $Az_t = -179°$. The calculation in equation (4.2) yield $-179° - 179° = -358°$, which corresponds to the sun being 358° to the right of the telescope, while it ideally should be 2° to the left. Therefore, we adjust the values accordingly

$$\Delta Az_\odot = Az_\odot + 360°, \text{ for } \Delta Az_\odot < 180° \tag{B.11}$$

$$\Delta Az_\odot = Az_\odot - 360°, \text{ for } \Delta Az_\odot > 180° \tag{B.12}$$

Here, the interval of the difference in azimuth is fixed to the interval $(-180°, 180°)$, where $0°$ means the telescope is pointing towards the sun in the azimuth direction. $\Delta Az_\odot = 90°$ corresponds to the sun being direct to the left of the pointing direction.

Another measure tested is the total angle between the pointing and the sun's position. We calculate this using the following formula

$$\theta = \cos Az_t \cdot \cos El_t \cdot \cos Az_\odot \cdot \cos El_\odot + \sin Az_t \cdot \cos El_t \cdot \sin Az_\odot \cdot \cos El_\odot + \sin El_t \cdot \sin El_\odot \tag{B.13}$$

Appendix C

Supplementary Theory

C.1 XGBoost Hyperparameters

This section provides an explanation of the hyperparameters used in the XGBoost models.

`n_estimators`: This parameter denotes the number of trees used in the ensemble. Increasing this number will allow for a more complex model.

`max_depth`: This parameter denotes the maximum depth of a tree. A tree could have a shorter depth if no new splits improve the model, but it cannot surpass this parameter value. A larger depth can improve the complexity of the model but also lead to overfitting.

`reg_lambda`: This parameter is responsible for the L2 regularization on leaf weights. Increasing this value reduces overfitting and can improve generalization.

`Colsample_by_tree`: This parameter controls the fraction of samples used when building a new tree. It can have values in the range $(0, 1]$.

`learning_rate`: This parameter controls the shrinkage of the weights of each tree during the learning process. Lowering this value will require more trees to be used in the ensemble but can improve generalization.

`subsample`: This parameter denotes the fraction of samples used when constructing a tree. Using a subset of the samples of the full dataset can lead to better generalization. The range of this parameter is $(0, 1]$.

`min_child_weight`: This parameter denotes the minimum number of samples required to create a new child node during tree construction.

`gamma`: This parameter specifies the minimum reduction in loss needed to split a leaf node.

Bibliography

[1] Jacob Baars, B. Hooghoudt, P. Mezger, and M. Jonge. The iram 30-m millimeter radio telescope on pico veleta, spain. *Astronomy and Astrophysics*, 175:319–326, 02 1987.

[2] Sindre Stenen Blakseth, Adil Rasheed, Trond Kvamsdal, and Omer San. Deep neural network enabled corrective source term approach to hybrid analysis and modeling. *Neural Networks*, 146, 2022.

[3] W. Van Breugel, C. De Breuck, S.A. Stanford, D. Stern, H. Röttgering, and G. Miley. A radio galaxy at z = 5.19. *Astrophysical Journal*, 518, 1999.

[4] Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, 1995.

[5] M. Coriat, S. Corbel, L. Prat, J. C.A. Miller-Jones, D. Cseh, A. K. Tzioumis, C. Brocksopp, J. Rodriguez, R. P. Fender, and G. R. Sivakoff. Radiatively efficient accreting black holes in the hard state: The case study of h1743-322. *Monthly Notices of the Royal Astronomical Society*, 414, 2011.

[6] A. Corstanje, P. Mitra, A. Bonardi, S. Buitink, H. Falcke, B. M. Hare, J. R. Hörandel, K. Mulrey, A. Nelles, J. P. Rachen, L. Rossetto, P. Schellart, O. Scholten, S. Ter Veen, S. Thoudam, T. N.G. Trinh, and T. Winchen. The effect of the atmospheric refractive index on the radio signal of extensive air showers using global data assimilation system (gdas). *Proceedings of Science*, 2017.

[7] Ian Covert. Understanding shap and sage. https://iancovert.com/blog/understanding-shap-sage/, 2020.

[8] Ian Covert, Scott Lundberg, and Su-In Lee. Understanding global feature contributions with additive importance measures, 2020.

[9] Ian Covert, Scott Lundberg, and Su-In Lee. Understanding global feature contributions with additive importance measures, 2020.

[10] Jian Dong, Li Fu, Qinghui Liu, and Zhiqiang Shen. Measuring and analyzing thermal deformations of the primary reflector of the tianma radio telescope. *Experimental Astronomy*, 45, 2018.

[11] Wodek Gawronski and Kamal Souccar. Control systems of the large millimeter telescope. *IEEE Antennas and Propagation Magazine*, 47, 2005.

[12] Daniel George and E.A. Huerta. Deep learning for real-time gravitational wave detection and parameter estimation: Results with advanced LIGO data. *Physics Letters B*, 778:64–70, mar 2018.

[13] D. B. Haarsma, R. B. Partridge, R. A. Windhorst, and E. A. Richards. Faint radio sources and star formation history. *The Astrophysical Journal*, 544, 2000.

[14] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions, 2017.

[15] Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre G.R. Day, Clint Richardson, Charles K. Fisher, and David J. Schwab. A high-bias, low-variance introduction to machine learning for physicists. *Physics Reports*, 810:1–124, may 2019.

[16] Vishal Morde. Xgboost algorithm: Long she may rein. Downloaded January 2023.

[17] Izaak Neutelings. Neural networks with tikz. https://tikz.net/neural_networks/, 2021. Accessed: March 15, 2023.

[18] C E Petrillo, C Tortora, G Vernardos, L V E Koopmans, G Verdoes Kleijn, M Bilicki, N R Napolitano, S Chatterjee, G Covone, A Dvornik, T Erben, F Getman, B Giblin, C Heymans, J T A de Jong, K Kuijken, P Schneider, H Shan, C Spiniello, and A H Wright. LinKS: discovering galaxy-scale strong lenses in the Kilo-Degree Survey using convolutional neural networks. *Monthly Notices of the Royal Astronomical Society*, 484(3):3879–3896, January 2019. _eprint: https://academic.oup.com/mnras/article-pdf/484/3/3879/28572367/stz189.pdf.

[19] Brandon Rhodes. Ephem, 2021. https://pypi.org/project/ephem/.

[20] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[21] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27, 1948.

[22] Lloyd Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2:307–317, 1953.

[23] Peter Stumpff. Astronomical pointing theory for radio telescopes. *Kleinheubacher Reports*, 15:431, 1972.

[24] Till Tantau. The PGF/TikZ manual, 2021.

[25] Tomruen. Azimuth-altitude schematic, 2011. Accessed: March 13, 2023.

[26] Tpoint Software. *TPOINT*, 2009. Version 13.5.

[27] Sebastian von Hoerner and Woon Yin Wong. Gravitational deformation and astigmatism of tiltable radio telescopes. *IEEE Transactions on Antennas and Propagation*, 23, 1975.

[28] E. White, F. D. Ghigo, R. M. Prestage, D. T. Frayer, R. J. Maddalena, P. T. Wallace, J. J. Brandt, D. Egan, J. D. Nelson, and J. Ray. Green Bank Telescope: Overview and analysis of metrology systems and pointing performance. *Astronomy & Astrophysics*, 659:A113, March 2022.