Machine learning methods for calibrating radio interferometric data

Author: Simphiwe Nhlanhla Zitha
Supervisors: Dr Arun Aniyan, Prof Oleg Smirnov, Dr Laura Richter
Rhodes University & SARAO, South Africa

18 December 2018

A thesis submitted in fulfilment of the requirements for the degree of Masters of Science in the Centre for Radio Astronomy Techniques and Technologies

Department of Physics and Electronics

RHODES UNIVERSITY
Where leaders learn

Abstract

The applications of machine learning have created an opportunity to deal with complex problems currently encountered in radio astronomy data processing. Calibration is one of the most important data processing steps required to produce high dynamic range images. This process involves the determination of calibration parameters, both instrumental and astronomical, to correct the collected data. These parameters include instrumental as well as astronomical parameters. Typically, astronomers use a package such as Common Astronomy Software Applications (CASA) to compute the gain solutions based on regular observations of a known calibrator source. In this work we present applications of machine learning to first generation calibration (1GC), using the KAT-7 telescope environmental and pointing sensor data recorded during observations. Applying machine learning to 1GC, as opposed to calculating the gain solutions in CASA, has shown evidence of reducing computation, as well as accurately predict the 1GC gain solutions and antenna behaviour. These methods are computationally less expensive, however they have not fully learned to generalise in predicting accurate 1GC solutions by looking at environmental and pointing sensors. We call this multi-output regression model *ZCal*, which is based on random forest, decision trees, extremely randomized trees and K-nearest neighbor algorithms. The prediction error obtained during the testing of our model on testing data is $\approx 0.01 < \mathrm{rms_e} < 0.09$ for gain amplitude per antenna, and 0.2 $\mathrm{rad} < \mathrm{rms_e} <$0.5 rad for gain phase. This shows that the instrumental parameters used to train our model strongly correlate with gain amplitude effects than phase.

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my original work, and that any work done by others or by myself previously has been acknowledged and referenced accordingly.

Simphiwe Nhlanhla Zitha, 18 December 2018

Acknowledgements

I would like to express my sincere acknowledgement to my supervisors: Dr. Arun Aniyan, Prof. Oleg Smirnov and Dr. Laura Richter for their invaluable direction and mentorship. Prof. Oleg Smirnov who gave me the opportunity to work on this thesis, and introduced me to the SARAO machine learning group. Dr. Arun Aniyan who constantly gave me academic advice and encouraged me to become an independently machine learning thinker. Dr. Laura who helped me to learn the basics of CASA and dealing with KAT-7 data. Thanks to the RATT group for their inputs and comments to this thesis, particular Dr. Sandeep Sirothia who constantly helped me with every questions that are CASA related. I am also very thankful towards the SARAO, my science operations line manager Dr. Lindsay Magnus and science commissioning line manager Dr. Sharmila Goedhart for providing me with learning opportunities and funding to machine learning conferences to present this work.

Last but not least I would also like to thank the God almighty for giving me strength and ability through the course of this thesis. My family for their continuous support and prayer. My office mates and colleagues Isabella, Sean and Marisa for all the support and reviews of this thesis, Olorato for his continuous support with machine learning and coding. Nkgonne Waka for her prayers and always by my side encouraging and pushing me through the difficulties of this thesis.

Contents

Outline of this work

Chapter 1

In this chapter we present the basic introduction to radio astronomy including its history, evolution from single dish to interferometry, the required instrumentation to observe the sky; the steps undertaken during data processing to correct for the true sky and the importance of the daily "big data" generated by the instruments.

Chapter 2

In this chapter we present a brief introduction to machine learning and learn its effectiveness in situations where deep and predictive insights need to be uncovered from data sets that are complex and fast changing. Methods to fine-tune hyper-parameters per algorithm used to obtain better results and minimum processing time are examined.

Chapter 3

In this chapter we describe the machine learning methods and procedure used to approach the problem, the approach to radio astronomy data processing tools, the dataset used for training and testing, and ways in which different multi-output predictors respond to complex data set and results.

Chapter 4

This brings us to the discussion of the results obtained using our machine learning algorithms from the testing and validation data sets.

Chapter 5

Finally this brings us to the conclusion and future work planned to improve the performance of the algorithm.

Publications

Conference poster presentations

This work was presented in the following conferences:

1. Deep Learning Indaba 2018 in South Africa (Deep Learning Indaba).
2. Machine Learning Summer School (MLSS), Argentina, Buenos Aires, 2018 (MLSS).
3. Neural Information Processing Systems (NIPS) 2017 (NIPS).
4. SKA-SA postgraduate conference 2017.
5. Deep Learning Indaba 2017 in South Africa (Deep Learning Indaba).

1. Radio astronomy

1.1 Introduction

Modern-day astronomy is at an unprecedented stage, with a deluge of data from different telescopes. In contrast to conventional methods, today astronomical discoveries are data- driven. The upcoming Square Kilometer Array (SKA) is expected to produce terabytes of data every hour (The SKA telescope). With this exponential growth of data, challenges for data calibration, reduction and analysis also increase, making it difficult for astronomers to manually process and analyse the data. Therefore, intelligent and automated systems are required to overcome these challenges.

One of the main issues in radio astronomy is determining the quality of observational data. Astronomical signals are very weak by the time they reache the Earth's surface. They are easily corrupted by atmospheric interferences, incorrect observational parameters (e.g. telescope locations or telescope pointing parameters), malfunctioning signal receivers, interference from terrestrial man-made radio sources and tracking inaccuracies (Taylor et al., 1999). Therefore, it is required to do proper corrections to the observational data before processing the data. Radio astronomers spend a considerable amount of time performing a series of preprocessing steps called *calibration*, which involves the determination of a set of parameters to correct the received data. These generally include instrumental as well as astronomical parameters. The general strategy for doing these corrections makes use of a calibrator source. Calibrator sources are well suited for determining astronomical parameters for data corrections because they have known characteristics such as the brightness, shape, and frequency spectrum (Taylor et al., 1999). This process of calibration is iterative and time-consuming.

During scientific observations, different external parameters such as atmospheric pressure, temperature wind conditions, and relative humidity are collected through thousands of sensors attached to the telescopes and its adjoining instrumentation. The data coming from different sensors may provide information about the external conditions that may have corrupted the observed data. This piece of information is not always included in the conventional calibration steps. We propose to use machine learning methods to predict the calibration solutions, looking at pointing and environmental sensor data. This is mainly motivated by the fact that calibration steps make corrections to data that have been corrupted by environmental parameters.

In this project, we make use of data from the Karoo Array Telescope (KAT-7), an array consisting of seven telescopes, which is a precursor to the MeerKAT radio telescope. We look at pointing azimuth, elevation, scan, offset, temperature, wind speed, air pressure and relative humidity sensor data recorded during observations with a calibrator source PKS1613-586 to generate the training and testing dataset. The overall generated dataset contains sensor data per telescope and calibration solutions for package the signal received by each telescope in horizontal polarization (H-pol) and vertical polarization (V-pol). These calibrator solutions are calculated using the astronomy software called Common Astronomy Software Applications (CASA), the popular which is used for data calibration and imaging in radio astronomy.

1.2 Introduction to radio astronomy

Radio astronomy is one of the most fascinating fields of study about the Universe. Astronomers capture and analyze the electromagnetic signals emitted by distant objects, such as stars and galaxies, using a radio telescope. In Section 1.2 we present a brief introduction to radio astronomy, including interferometry vs single dish in Section 1.3. In Section 1.4 we define calibration in radio astronomy and describe the different techniques used for data calibration in Section 1.5.

1.2.1 History

Radio astronomy is the study of celestial sources emitting radio waves (Verschuur, 2015). A wave is an oscillatory motion of any kind, the most familiar ones being sound waves and vibrations of various material substances and waves on the surface of water. There are also wave disturbances found in electric and magnetic field. Such waves are responsible for what we experience daily, i.e. X-rays, visible light, or radio waves (Cassidy et al., 2002). Traditional astronomy is based on visible light (i.e. optical observations) wavelengths, but astronomical sources emit radiation not only as visible light but also across the electromagnetic spectrum from gamma rays at short wavelength(λ) to radio at long wavelength (low frequency), as shown in Figure 1.1 (Staats, 2016).

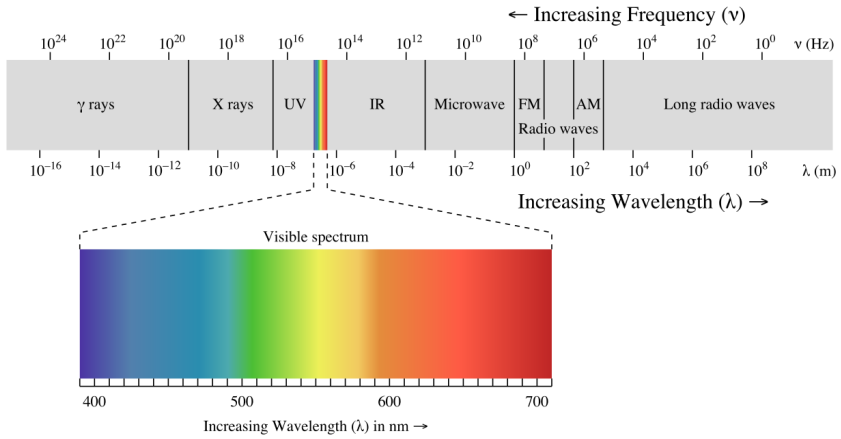


Figure 1.1: The electromagnetic spectrum (The Electromagnetic Spectrum)

Electromagnetic radiation is a group of waves created by fluctuations of electric and magnetic fields propagating through space at the speed of light ($c = 3 \times 10^8 m/s$) carrying electromagnetic radiant energy (The Electromagnetic Spectrum). There are numerous emission mechanisms in the universe that generate radio waves. Each portion of the electromagnetic spectrum reveals a unique view of the universe and so modern astronomy employs various instruments to look at each portion of the spectrum. The Earth's atmosphere absorbs electromagnetic radiation at most infra-red, ultraviolet, X-ray, and gamma-ray wavelengths. This means one can only observe the Universe from the ground in two windows namely the radio and visible wavebands. Radio waves can reveal objects that do not radiate in other parts of the electromagnetic spectrum, and they are able to pass through galactic dust clouds that blocks the view in the optical range and can penetrate the Earth's atmosphere (subject to some limits depending on frequency) (Thompson et al., 2001). After received by a radio telescope, the data are amplified and transferred to a computer for further data processing, which is discussed in the next chapter.

Radio astronomy was discovered accidentally by a radio engineer, Karl Jansky, in the 1930s. Jansky was assigned the task of investigating sources of radio interference radio-telephone communication system at short wavelengths (10-20 meters) (Verschuur, 2015). The purpose of this task was to determine the direction of the radio wave interference so they could easily tune out the interference by using directional antennas pointed away from the direction of the interference. In the process of conducting this task, Jansky constructed an antenna designed to receive radio waves at a frequency of 20.5 MHz (Society of Amateur Radio Astronomers). The antenna was mounted on a circular train track so that it could be rotated in all possible angles along the ground and could detect radio signal in all possible directions. After several months of recording and analysing the radio signals from all directions, Jansky's investigation was successful and he identified two known signals (from local and distant thunderstorms), which were interfering with the radio-telecommunication system, and one unknown signal (faint steady hiss). Jansky continued to investigate the unknown signal for over a year and eventually found out that the radiation was coming from the Milky Way. The signal was the strongest in the direction of the center of the Milky Way galaxy, in the constellation of Sagittarius. This was the first detection of non-black body radiation at radio wavelengths from an extraterrestrial source. Black body radiation is the radiation given off by any object related to its temperature (MIT Haystack Observatory).

After Jansky's project at Bell Laboratory ended, Bell was not interested in studying astronomy any more. In 1937, an astronomer, Grote Reber, who was fascinated by Jansky's ground-breaking discovery, decided to continue where Jansky had left off by further investigating the cosmic radio waves from the Milky Way galaxy. To pursue this research, Reber built the first single-dish radio telescope with a receiver strong enough to detect cosmic radio signals in his backyard. He spent hours at night scanning the sky at different frequencies. He was finally successful at detecting radio emission from the Milky Way galaxy, confirming Jansky's discovery (Verschuur, 2015).

Astronomical signals tend to be weak by the time they reach the ground. The need for greater sensitivity and higher resolution has resulted in the creation of telescopes that have a larger collecting area for more radio energy to be focused on the receiver (Verschuur, 2015). However, constructing a single-dish telescope large enough i.e. $> 500\ \mathrm{m}$ to achieve high angular resolution $\theta \approx \frac{\lambda}{D}$ (where λ is the wavelength of the electromagnetic radiation observed and D is the aperture diameter of the radio telescope), is physically impossible owing to the cost of the materials and the structure being at high risk to collapsing during high wind conditions. The angular resolution of a radio telescope measures its ability to detect fine details in the structure of the observed celestial source. For example, to obtain an angular resolution of 1 arc-second, would require a single-dish radio telescope's diameter to be thousands of metres (Verschuur, 2015).

1.3 Single dish vs interferometry

A single-dish survey can be complemented by an array of small radio telescopes providing higher sensitivity and higher resolution images (Thompson et al., 2001). This technique is called radio interferometry. Radio interferometry is mostly used to obtain high resolution images of astronomical sources, where-as a single-dish technique is mostly considered as a tool for low spacial resolution, i.e., it allows for imaging of very large sources (Thompson et al., 2001). Though these two techniques are quite different, they both provide measurements of the Fourier transform of some region in the sky (Cornwell, 1988). In this dissertation we focus on the interferometer, which consists of two or more sub-apertures grouped together to form a large array (Verschuur, 2015), as shown in Figure 1.2. Though the telescopes are placed at different positions separated by distance b, the resulting power output from all the telescopes

is combined. For an interferometric array, the angular resolution is defined as $\theta \approx \frac{\lambda}{B}$, where B represents the t maximum baseline in the array.

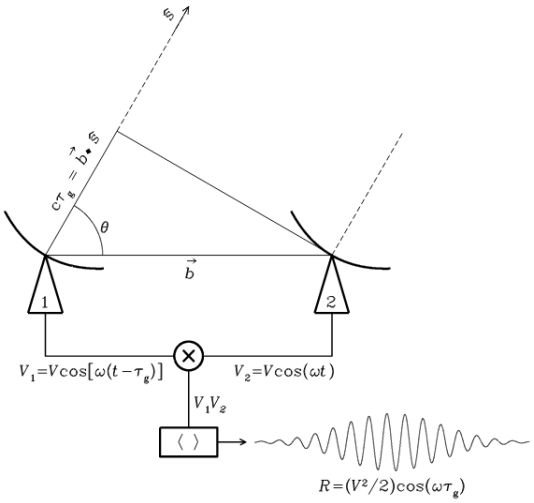


Figure 1.2: Two-antenna interferometer diagram [National Radio Astronomy Observatory]

Figure 1.2 illustrates a simple two two-element interferometer consisting of baseline $\overrightarrow{b}$ (we will refer to the left antenna as antenna 1 and the right antenna as antenna 2). This can be extended to multi-antenna arrays by adding more N antennas in the array such that the number of baselines $\frac{N(N-1)}{2}$ increases (Thompson et al., 2001). From Figure 1.2, suppose a distant source is observed in the direction $\widehat{s}$, the incoming signal will arrive in each radio telescope at slightly different times. To compensate for this time difference such that the signal arrives in each telescope at the same time, a time delay term $\tau_g = \frac{\overrightarrow{b} \cdot \widehat{s}}{c}$ is introduced to produce voltages V_1 and V_2 from antenna 1 and antenna 2 denoted by:

$$\begin{aligned} V_1(t) &= V\cos[\omega(t-\tau_g)], \text{ where } \nu = \frac{\omega}{2\pi} \text{ is the center frequency} \\ V_2(t) &= V\cos(\omega\tau_g). \end{aligned} \tag{1.3.1}$$

The inputs V_1 and V_2 in to the correlator are then multiplied and time averaged to yield an output response of amplitude R which is proportional to the point-source flux density and with its phase depending on the time delay and the frequency.

$$V_1 \times V_2 = V^2\cos(\omega\tau_g) \times \cos[\omega(t-\tau_g)] \tag{1.3.2}$$

From $\cos A\cos B = \frac{1}{2}(\cos(A+B) + \cos(A-B))$, we therefore have

$$\begin{aligned} V_1 \times V_2 &= \frac{V^2}{2}(\cos(\omega\tau_g + \omega t - \omega\tau_g)) \\ &= \frac{V^2}{2}(\cos(2\omega t - \omega\tau_g) + \cos(\omega\tau_g)) \end{aligned}$$

We then take the time average to remove the high frequency term $\cos(2\omega t - \omega\tau_g)$, such that we obtain the output

$$R =< V_1 V_2 >= \frac{V^2}{2}\cos(\omega\tau_g) \tag{1.3.3}$$

$$= \frac{V^2}{2}\cos\left(\frac{\omega}{c}\overrightarrow{b}\cdot\widehat{s}\right) \tag{1.3.4}$$

$$= \frac{V^2}{2}\cos\left(\frac{2\pi\nu}{c}\times\overrightarrow{b}\cdot\widehat{s}\right). \tag{1.3.5}$$

The amplitudes V_1 and V_2 are proportional to the electric field produced by the source multiplied by the the voltage gains of antennas 1 and 2. Thus the output amplitude $\frac{V^2}{2}$ is proportional to the point-source flux density S multiplied by $\sqrt{(A_1, A_2)}$, where A_1 and A_2 are the effective collecting areas of the two antennas (InteferometersI).

According to equation 1.3.1 radio interferometry has enabled astronomers to make measurements of fine angular details, including parameters such as frequency spectrum, polarization and the intensity (Thompson et al., 2001).

1.4 Calibration in radio astronomy

1.4.1 Interferometric visibility

In an interferometric array, the measured visibilities which are the two-dimensional (2D) Fourier transform of the sky's intensity at different baseline coordinates in the absence of instrumental effects, are defined by

$$V(u,v) \approx \mathcal{F}\{I\}(u,v) = \int\int A(l,m)I(l,m)e^{-2\pi i(ul+vm)}\mathrm{dldm}, \tag{1.4.1}$$

where (u,v) denotes the projected baseline coordinates in the 2D Fourier transform plane measured in wavelength along the axes in the east-west and north-south direction, and changes with time as the earth rotates (Taylor et al., 1999). The orthogonal coordinates (l,m) represent the position of a source or the phase reference position. In the 2D analysis l and m are defined as the cosines of the angles between the direction (l,m) and (u,v). $I(l,m)$ is the intensity distribution of a source and its Fourier transform $F\{I\}$ represents the amplitude and phase of a sinusoidal component of the intensity profile with spatial frequency u and v; $A(l,m)$ represents the effective collecting area of the antennas with respect to the direction of the incoming radiation (Thompson et al., 2001).

To simplify equation 1.4.1, the term $A(l,m)$ is most often assumed to be 1 such that the Van Cittert Zernike theorem states, (Thompson et al., 2017):

$$V(u,v) \approx \mathcal{F}\{I\}(u,v) = \int\int I(l,m)e^{-2\pi i(ul+vm)}\mathrm{dldm}. \tag{1.4.2}$$

The measured visibilities $V(u,v)$ and the sky brightness $I(l,m)$ are Fourier pairs $V(u,v) \rightleftharpoons I(l,m)$, where $\rightleftharpoons$ is the Fourier transformation. The sky brightness can readily be recovered from the measured visibilities by an inverse Fourier transform such that we have,

$$I^{\mathrm{True}}(l,m) \approx \mathcal{F}^{-1}\{V\}(l,m) = \int\int V(u,v)e^{-2\pi i(ul+vm)}\mathrm{dudv}. \tag{1.4.3}$$

Since for every sky's brightness I, there exists visibility function $V(u,v)$, the array of antennas with baselines $b = (u_i, v_i)$ measures only certain values in the set of continuous (u,v) coordinates in the visibility function $V(u,v)$. This measured set of values is given by the *Sampling function*, denoted by (Taylor et al., 1999),

$$S(u,v) = \begin{cases} 1 & \text{at uv point} \\ 0 & \text{otherwise} \end{cases}$$

for all the baselines. The actual data provided by the array is called sampled visibilities, denoted by $S(u,v) \times V(u,v)$. The Fourier transform of these sampled visibilities is the dirty image,

$$I^D(l,m) = \int\int S(u,v) \times V(u,v) e^{-2\pi i(ul+vm)} \mathrm{dudv}. \tag{1.4.4}$$

Using the *convolution theorem* for Fourier transforms which states that the Fourier transform of the convolution of two functions is the product of their Fourier transforms denoted by

$$f * g \rightleftharpoons \mathcal{F}\mathcal{G}, \tag{1.4.5}$$

where $f \rightleftharpoons \mathcal{F}$ and $g \rightleftharpoons \mathcal{G}$. From equation 1.4.4, we therefore have

$$I^D = PSF \circ I^{\mathrm{True}}, \tag{1.4.6}$$

where

$$PSF(l,m) = \int\int S(u,v) e^{-2\pi i(ul+vm)} dudv, \tag{1.4.7}$$

representing a point spread function corresponding to the sampling function $S(u,v)$ (Taylor et al., 1999).

1.4.2 Calibration

In radio astronomy, ideally one might think that after obtaining the observed visibilities the next step would be to directly retrieve the actual visibilities of the target source and perform imaging. However, in reality it is not as simple, as shown by equation 1.4.2. The measured visibilities V^{obs} are different from the actual visibilities V^{True} and this is due to instrumental and environmental effects (Thompson et al., 2017). An example of these effects on the signal measured by a radio interferometry include antenna gains (slowly and fast time-varying instrumental part), atmospheric effects, pointing errors (tracking inaccuracies) and incorrect observation parameters (antenna pointing parameters). Signal effects are classified into two types, direction-independent effects (affecting the signal from all directions equally) and direction-dependent effects (which vary based on the sky position of the signal) (Taylor et al., 1999). These effects can be corrected by estimating the errors associated with the measured visibilities, thereby recovering the true visibilities. This process is called calibration. In its simplest form, calibration minimizes the error between observed and predicted (model) visibilities by estimating the correct complex instrumental gain response (Grobler et al., 2016).

Suppose for baseline pair (i,j), the observed visibility is $V_{i,j}^{\mathrm{obs}}(t)$ and the true visibility is $V_{i,j}^{\mathrm{True}}(t)$ at observation time t. The basic calibration formula is written as,

$$V_{i,j}^{\mathrm{obs}} = G_{i,j} V_{i,j}^{\mathrm{True}} + \epsilon_{i,j}(t), \tag{1.4.8}$$

where $G_{i,j}(t)$ denotes the complex antenna gains for baseline (i,j) as a result of unwanted effects and may vary with time (Thompson et al., 2001). The extra term $\epsilon_{i,j}(t)$ is a stochastic complex noise (Taylor et al., 1999).

Most of the corruptions in data occur before the signal is correlated and the response associated with antenna i does not depend on the response of antenna j. The baseline-based complex gain $G_{i,j}$ can therefore be approximated by the product of amplitude A and phase ϕ such that one has,

$$G_{i,j}(t) = g_i(t)g_j^*(t) = A_i(t)A_j(t)e^{i(\phi_i(t)-\phi_j(t))}, \tag{1.4.9}$$

where $A_i(t)$, $A_j(t)$ are antenna-based gain amplitude solutions and $\phi_i(t), \phi_j(t)$ are antenna-based gain phase solutions. Calibration aims to find the appropriate A and ϕ for the corrupted visibilities. One standard technique for measuring the gain calibration solutions is continuously observe a part of the sky that contains a single known, relatively strong point source. This is to measure the instrumental phase, since the telescopes on the ground can only measure the phase differences as there is no absolute phase reference measure. Therefore, the reference for the phase visibilities can be accomplished by observing a stable point-like calibrator sources. Using these sources, one can determine the phase deviation from the desired reference point (Taylor et al., 1999). It is important to conduct observations of these sources regularly to track the phase and gain changes in an array. Above all, they have the paramount advantage of estimating time-dependent phase changes incurred by the atmosphere (Taylor et al., 1999).

The selection of a good calibrator source in the sky is based on the following desirable characteristics (Thompson et al., 2001):

1. They should have known positions in the sky, and the calibrator should be close to the target source (Taylor et al., 1999).
2. They should have strong flux densities to obtain a suitable signal-to-noise ratio (SNR) in a short time (Taylor et al., 1999).
3. The calibrator should be a point source (unresolved), if possible.
4. Their spectral properties should be known.

Note that the sources that are the subject of astronomical investigation will be referred to as "target sources" to distinguish them from calibrator sources (Thompson et al., 2001).

Suppose a target source is observed with measured visibility $V(u,v)^{\text{obs}}$ to calibrate the antenna-based complex gain factor $G_{i,j}(t)$ as a function of time and antenna pair (i,j), a point source calibrator is observed for which the measured visibility is,

$$V(u,v)_{\text{calibrator}} = G_{i,j}(t)S_{\text{calibrator}} \tag{1.4.10}$$

$$G_{i,j}(t) = \frac{V(u,v)_{\text{calibrator}}}{S_{\text{calibrator}}} \tag{1.4.11}$$

(Thompson et al., 2001), where S indicates the flux density of the calibrator. Now to calibrate the visibilities of the target source one can write,

$$V(u,v)^{\text{True}} = \frac{V(u,v)^{\text{obs}}}{G_{i,j}(t)} \tag{1.4.12}$$

$$= V(u,v)^{\text{obs}} \times \frac{S_{\text{calibrator}}}{V(u,v)_{\text{calibrator}}} \tag{1.4.13}$$

(Thompson et al., 2001).

1.5 Calibration techniques

With the improvement in complexities of the new radio astronomy instruments, various calibration techniques have been developed to overcome these challenges posed by the new instruments, thereby producing accurate calibration results. These techniques are loosely classified into first generation calibration (1GC), second generation calibration (2GC) and third generation calibration (3GC) (Noordam and Smirnov, 2010). In this dissertation thesis our focus will be on 1GC using CASA. CASA is a suite of functions for the reduction and analysis of radio astronomical data with IPython interface, and is currently maintained by the National Radio Astronomy Observatory (McMullin et al., 2007).

1.5.1 First generation calibration

This is the first step in the data reduction process where the received signal in each baseline is compared to the signal from a known source (the calibrator sources) to address the following errors in the response of the instrument:

- Bandpass calibration B is used to solve for gain variations in frequency (CASA cookbook, 2009). These variations in frequency arise as a result of non-uniform filter passbands or other frequency-dependent effects during signal transmission. These frequency-dependent effects usually vary on long time-scales. The CASA task that solves these frequency-dependant complex gains is called *bandpass*. This is done by observing a strong calibrator source with known model visibilities and spectrum characteristics for sufficient time to reach the required accuracy (Taylor et al., 1999).

- Delay calibration K is used to correct for phase delay errors. The arriving signal does not propagate to all the antennas at the same time. Delay calibration equalizes these propagation differences on individual signal paths at the input of the correlator and removes the largest phase difference across the observing band (Taylor et al., 1999).

- Gain calibration G is used to solve for time-dependant complex gain variation. These gain variations include the tropospheric amplitude and phase effects, amplitude response as a function of elevation (gain curve), phase and amplitude drifts in the electronics of each antenna, and relative amplitude and phase gain for each antenna (Taylor et al., 1999). Some of these calibrations are known prior and others must be determined by conducting observations of calibrator sources (Taylor et al., 1999). The CASA task used to solve for these antenna-based gains in each polarization in different specified time solution intervals is *gaincal* (CASA cookbook, 2009).

  Note that it is not always possible to find a calibrator source that satisfies the requirements mentioned in section 1.4.2, especially point number 2. In such cases it may be necessary to find a calibrator source that is a point source close to the target source and then calibrate it against one of the more commonly used flux density reference calibrators, such as PKS1934-638, 3C147 or 3C286 (Thompson et al., 2001).

- Flux calibration is used to calibrate the flux density of source calibrators with unknown flux densities. The flux density of the unknown calibrators is assumed to be $1\mathrm{Jy}$. This assumption is made such that the complex gain solutions G of the reference flux density calibrator and one

with unknown flux density are compared and scaled so that they match as much as possible. The CASA task used to perform this scaling or bootstrapping is *fluxscale* (CASA cookbook, 2009).

The solutions obtained from the tasks (*gaincal*, *bandpass*, *fluxscale*) are saved in tables and then applied to uncalibrated data to form calibrated data.

1.5.2 Second and third generation calibration

Once the calibrated data have been obtained by performing the 1GC, the first model image is computed. A process called *self- calibration* can be performed to obtain more accurate images by making further corrections to the antenna gains as a function of time. This application is very similar to the basic calibration. The main difference is that the model of the target source is more complex than assuming that it is a point source and the observed field is also used to calibrate itself (Wieringa, 1992). The 1GC and 2GC sections focus mainly on solving direction-independent effects. With the wide field of views of current and future generation radio telescopes such as the Extended Very Large Array, Low Frequency Array and the Square Kilometer Array (SKA), direction dependent-effects pose a challenge in calibration which result in 3GC.

1.6 Data deluge in radio astronomy

1.6.1 The Square Kilometre Array - South Africa

Figure 1.3: Subset of MeerKAT radio telescope in the Karoo, a semi-desert region of South Africa [SKA, South Africa]

The SKA is an international project to build the world's largest radio telescope array. When complete, it will be a radio telescope with an aperture of up to a million square meters. This powerful, sensitive instrument represents the advance in astronomy, engineering, research and development to construct a unique scientific instrument (The SKA telescope). Two sites chosen for the construction of this tremendous instrument, one of them being the Karoo wilderness of South Africa, where the 64-dish telescope (MeerKAT), a pathfinder to the SKA phase 1, is being constructed 1.3, while the other site is in the Murchison region of Western Australia (Hall et al., 2008).

When the SKA is complete, it will be a sophisticated instrument to view and track a bevy of cosmic wonders, including objects and events that were too dim or distant for modern astronomers' sky-gazing forebears. This instrument will be far faster ($10^6\times$) than any system currently available, and image the sky in sensitivity ($100\times$) and detail never before achieved (The SKA telescope).

1.6.2 Big data challenges in radio astronomy

Astronomy has been among the first scientific disciplines to experience a flood of data. The upcoming SKA is expected to produce terabytes of complex data every hour. The term complex applies both the data that consists of many separate datasets in a large number of different quantities or huge amount of data. This exponential growth in data illustrated in Figure 2.4 is pushing the requirements for data storage and analysis to the boundaries. The volume and variety of data obtained from the SKA telescope will outstrip the capacity of manual analysis, and in some cases also exceed the capacity of conventional databases currently existing. In view of these challenges caused by the massive data volume, rates, and complexity from next-generation radio telescopes, it is crucial to develop tools that can address astronomical problems faster than astronomers can analyse, such as artificial intelligence systems or algorithms that can connect large datasets to enable broader and deeper analyses than previously possible (Provost and Fawcett, 2013). Making new discoveries and revealing the secrets of the universe may not be as simple as previously envisaged. Gone are the days when astronomers could manually detect or notice something odd in their tables or graphs by just browsing through them (Big data in radio astronomy). The circumstances of the high volume and speed of data produced by the new, sophisticated astronomical instruments are a great challenge to humans interms of making unexpected discoveries. We wish to make use of computational machinery to find useful patterns in digital data, and translate these patterns into information that can be used to solve various complex problems, hence machine learning (Ball and Brunner, 2010). This learning must be returned in a useful manner to a human investigator, which will ideally result in human learning (Ball and Brunner, 2010).

2. Machine learning

Machine learning is programming computers to optimize a performance criterion using example data or past experience. In this chapter we present machine learning applications to challenges posed by big data. In Section 2.1, we give a brief definition of big data and the challenges currently faced in radio astronomy. In Section 2.2 we give a brief introduction and examples of machine learning, and in Section 2.3, 2.4 we present the process of machine learning and model complexity.

2.1 Big data

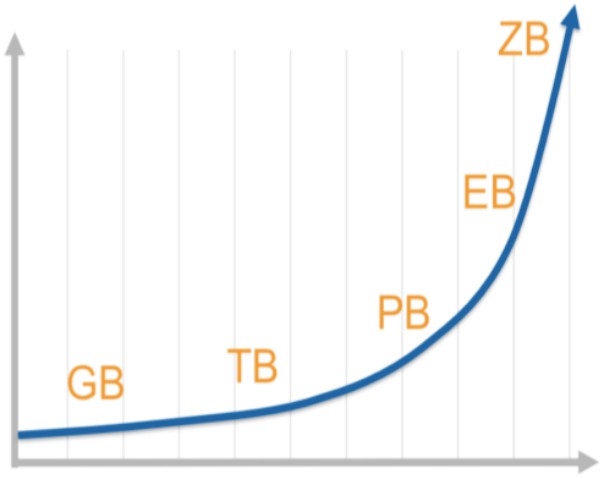


Figure 2.1: The exponential growth of data from gigabytes to zettabytes [Amazon].

In the past few decades, computing power has been rising exponentially, and as a related consequence, enormous quantities and complexity of the observed data, primarily in digital form. Such data are referred to as *big data* and traditional data-processing tools are inadequate to deal with them. In addition to the natural world, big data are now creating a digital world, in which extracting new and useful information from the data already taken and stored is becoming a major endeavor in itself (Ball and Brunner, 2010). This action of knowledge discovery in databases is commonly referred to as *data mining* (Ball and Brunner, 2010). In data mining, a large volume of data is processed to construct a simple model with valuable use, e.g., having high predictive accuracy (Alpaydin, 2014).

2.2 An Introduction to machine learning

For a computer to solve a problem, one need an algorithm. An algorithm is a sequence of instructions carried out to transform the input to output. e.g, one can design an algorithm for sorting, where the input is a set of unordered letters and the output is an ordered list from A to Z. For the same task, there may be various algorithms and one may be interested in finding the most optimum one, requiring the lowest number of instructions and memory to run. For some tasks, however, one does not have an algorithm, for example to differentiate between legitimate emails and spam emails (Alpaydin, 2014).

Therefore what we lack in knowledge, we make up for in the collected data. One can easily compile thousands of example messages, some of which one knows to be spam and then we get to learn what characterizes a spam email (Alpaydin, 2014).

Learning is what gives humans flexibility, the fact that humans can adjust and adapt to new circumstances and learn new tricks no matter at any age (Marsland, 2015). The important parts of human or animal learning are adapting, remembering and generalising. Being able to recognize old events, and to know which actions are to be taken, and also knowing how to distinguish between different situations is what makes learning useful (Marsland, 2015).

Machine learning is not only a database problem, but is also part of artificial intelligence. For a system to be intelligent, it needs to have the ability to learn and adapt in any changing environment. If the system can learn and adapt to changes, then it can easily predict future solutions not provided by the system designer (Alpaydin, 2014). This involves tasks such as recognition, prediction, diagnosis, robot control, planning etc (Nilsson, 1996). Machine learning also helps one find solutions to many problems in robotics, speech recognition and vision. Machine learning is is generally based on programming computers to optimize their performance criterion by using example data that consist of various events. A model is defined up to some parameters, and part of the learning process is to execute a computer program to find the optimum parameters of the model using the training data that is based on past experience with different events (Alpaydin, 2014). The model may be descriptive to gain knowledge from data or predictive to make predictions in the future. Machine learning uses the theory of statistics in building mathematical models, because the core task is making inference from different samples (Alpaydin, 2014).

Given a class of tasks T, an experience E and measure of performance P, a computer program is said to learn from E with respect to T if the measure P at T improves for more observations E provided (Michalski et al., 2013).

2.2.1 Examples of machine learning

We are surrounded by machine learning-based technology, such as online shopping, where search engines learn how to bring us the best results and a list of the most relevant products related to our search, email (spam filters, smart email categorization, etc) where anti-spam software learns to filter our email messages into spam or not-spam using simple rule filters. A similar approach is used by Gmail to categorize our emails into primary, social, and promotion inboxes, as well as labelling emails as important.

In a research paper, "The Learning Behind Gmail Priority Inbox" (Aberdeen et al., 2010), Google outlines its machine learning approach and notes: " When a user marks messages in a consistent direction, we perform a real-time increment to their threshold." Every time the user marks an email as important, Gmail learns. The researchers tested the effectiveness of Priority Inbox on Google employees and found that those with Priority Inbox "spent 6% less time reading email overall, and 13% less time reading unimportant email.", Facebook automatically highlights faces and suggests friends to tag when uploading a photo. A similar approach is used on digital cameras to learn how to detect faces. Smartphones today are equipped with a standard feature to convert voice to text; by pressing a button or saying a particular phrase ("OK Google", for example), you can start speaking and your phone converts the audio into text and recognizes voice commands (Techemergence). Machine learning is also widely used in scientific applications such as astronomy and health.

One common feature of all of these applications is that, in contrast to more traditional uses of computers,

in these cases, owing to the complexity of the patterns that need to be detected, a human programmer will find it difficult to provide finely detailed specification of how to tackle such tasks(Shalev-Shwartz and Ben-David, 2014). In recent years, the world's technology has become increased consistently, as shown in Figure 2.4, and this escalation has led to the amount of data available for learning increasing dramatically. We therefore provide several reasons why machine learning is important, given the data challenges.

- Machine learning methods can often be used to extract the important relationships and correlations of hidden information among big data.
- The amount of information available about certain tasks might be too large and complex for humans to encode, whereas machines can learn this knowledge accurately and be able to capture more of it than humans would (Nilsson, 1996).

Our focus will be based mainly on tasks that are beyond human capabilities. These are families of tasks that are related to the analysis of complex large datasets such as astronomical data. With more available stored digital data, it becomes obvious that there is meaningful information buried in the data that is complex for humans to make sense of. Learning to detect meaningful information in large datasets is a promising domain in which the combination of programs that learn with computers that have high processing power opens up new horizons (Shalev-Shwartz and Ben-David, 2014).

2.2.2 Types of machine learning

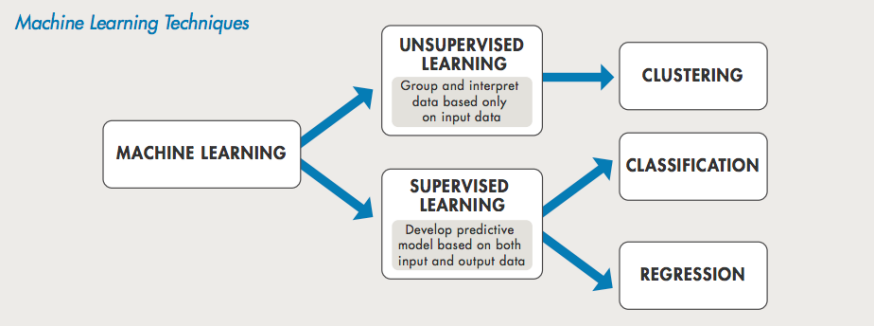


Figure 2.2: Machine learning techniques include both supervised and unsupervised learning (Machine learning).

Machine learning algorithms are broadly divided into unsupervised and supervised methods, also known as descriptive and predictive. In supervised learning, the goal is to predict the value of an output measure based on a number of input measures, i.e, finding the relationship between the input and output based on the training dataset. In contrast to supervised learning, unsupervised learning occurs in the sense that the data can speak for themselves without preconceptions such as expected classes being imposed (Ball and Brunner, 2010).

2.2.2.1 Supervised learning

The aim of supervised machine learning is to build a model that makes predictions based on some evidence in the presence. Suppose one wants to fit a model $\mathbf{Y} = f(\mathbf{X}) + \epsilon$ with errors ϵ being additive.

Supervised learning attempts to learn the function f by examples given through a teacher, given the training set with input and output observations $T = (x_i, y_i), i = 1, \ldots N$. The observed input values x_i are fed into a learning algorithm, which produces outputs $\widehat{f}(x_i)$ in response to the given inputs. The learning algorithm has the property that it can modify its input/output relationship $\widehat{f}$ in response to the computed error $y_i - \widehat{f}(x_i)$ between the original and predicted outputs. This process is known as learning by example. Upon completion of the learning process the hope is that the predicted and real outputs will be close enough with an error of approximately zero (Friedman et al., 2001).

Supervised learning uses regression and classification techniques to develop predictive models. Depending on the given dataset, variables can be characterized as either qualitative (also known as categorical) or quantitative. Quantitative variables take on numerical values. Examples include a person's height, age, air temperature, wind speed, etc. Whereas, qualitative variables take on input vectors and decide which of N classes they belong to as shown in Figure 2.5, based on training from exemplars of each class or category. Examples of qualitative variables include a person's gender (male or female) or whether a person defaults on a debt (yes or no) (Aitkin et al., 2009). Qualitative variables are typically represented numerically by codes. The easiest case is when there are only two classes or categories, such as "success" or "failure", "survived" or "died". These are often represented by a single binary digit or bit as 0 or 1, or else by -1 and 1. For reasons that will become apparent, such numeric codes are sometimes referred to as targets (Friedman et al., 2001). In machine learning, if the label is numerical, the task is called regression and the learner is also called a fitted regression model, while if the label is categorical, the technique is called classification and the learner is also called a classifier. For both techniques, the training process is conducted on data sets containing label information or examples (Zhou, 2012).

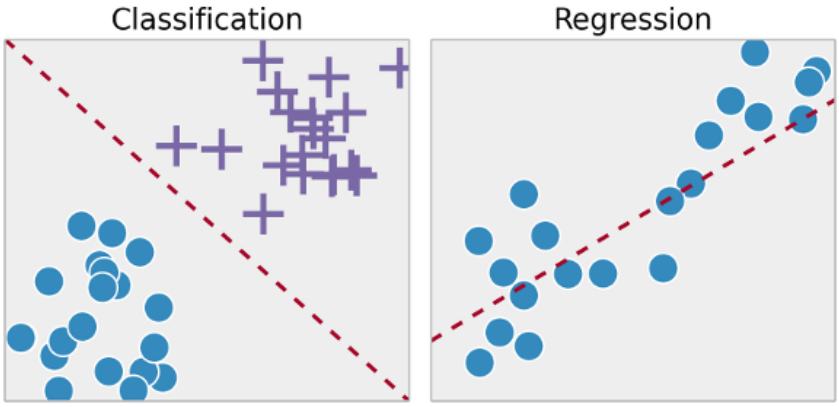


Figure 2.3: Example of classification and regression (Rossant, 2018).

Classification tasks are mostly known to be discrete, i.e. each example belongs to only one class, and the set of classes covers the whole possible output space; whereas regression tasks are known to be continuous (Stephen, 2009). However there are some overlaps between the algorithms for these tasks. A classification algorithm may predict a continuous value, but the continuous value is in the form of a probability for a class label and a regression algorithm may predict a discrete value, but the discrete value is in the form of an integer quantity. Some algorithms can be used for both classification and regression with small modifications, such as decision trees and artificial neural networks (Brownlee, 2013).

2.2.2.2 Unsupervised learning

In contrast to labelled data, data items without associated labels can often be obtained in great quantities without much effort to be done on it. The unsupervised learning technique aims at making use of the information provided by the unlabelled patterns to generate appropriate models.

As illustrated in Figure 2.2, in unsupervised learning one has only a set observations $T = x_1, x_2, \ldots, x_N$. In unsupervised learning one is not interested in prediction, because one does not have an associated output response variable $y_1, y_2, \ldots, y_N$. Rather the goal in unsupervised learning algorithms is to discover interesting information about the measurements on the training set T. One looks for an informative way to visualize the data and interesting patterns among the observations (James et al., 2013).

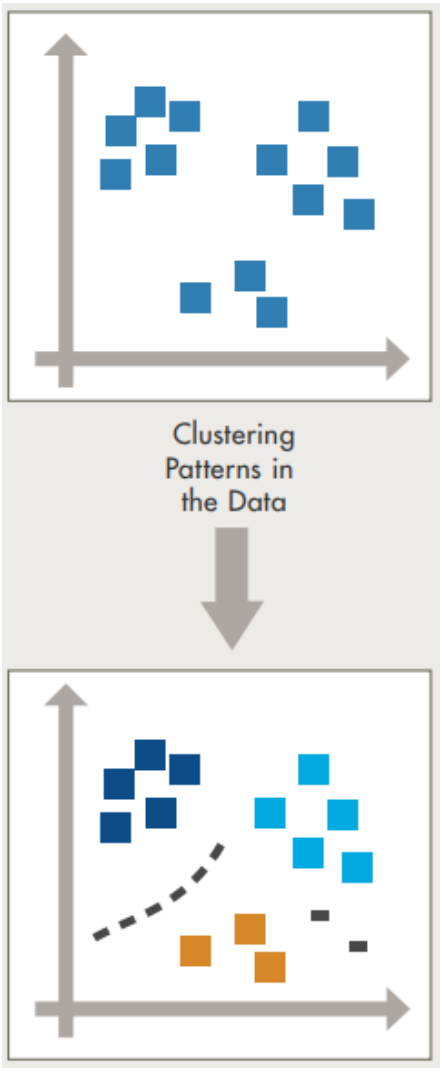


Figure 2.4: An example of clustering method which reveals hidden patterns in data (Machine learning).

Clustering is one of the most commonly used unsupervised methods that divide the data into clusters. The number of clusters must initially be specified with a value K. The algorithm uses a distance criterion for cluster membership, such as computing the Euclidean distance between the clustered points, and a stopping criterion for iteration(Ball and Brunner, 2010).

2.3 Process of machine learning

2.3.1 Data preparation

Machine learning algorithms learn from data, therefore it is critical that one feed them the right data for the specific problem one want to solve. In this section we will describe the processing steps required for getting data ready for a machine learning algorithm.

Three steps are commonly used in machine learning to process the data, namely data selection, data preprocessing and data transformation. Data selection has to do with the selection of the the subsets of all available data that one need to address the question or problem on which we are working. Usually assumptions will be made about the data and be tested at a later stage. After selecting the data, comes the preprocessing step where one decide how to use the data. Firstly, suitable formatting of the data is considered, which could be .csv file format or Numpy file format. In such cases there may be data instances that are incomplete or do not contain useful information. Such cases, the removal or fixing of such data is required. When generating the training dataset, it is crucial to also consider the computational cost and memory requirement. Sampling the data by taking a smaller representative sample of the selected data may be much faster for exploring the data space and prototyping solutions before considering the whole dataset. The final step is to transform data using scaling, where the preprocessed data may contain attributes with a mixture of scales for various quantities, which then are scaled to range between 0 and 1. This helps to stop the weights from getting too large unnecessarily (Marsland, 2015). One could also use decompositions, which involve attributes that represent a complex concept that may be more useful to a machine learning method when split into the constituent parts, and aggregations where attributes can be aggregated into a single attribute that would be more meaningful to the problem being solved (Brownlee, 2013). The most commonly used scaling methods are referred to as data normalisation, or standardisation (Marsland, 2015).

2.3.2 Feature engineering

In machine learning applications, whether classification or regression, observation dataset that one believes to contain information are taken as inputs and fed to the model for decision-making. Some of the goals in these learning algorithms are: improving the prediction performance of the model and reducing the model complexity. This could help reduce the estimation error and thus prevent overfitting. If one assumes that the observation dataset $\mathbf{X} = \mathbb{R}^d$ (Shalev-Shwartz and Ben-David, 2014), then the complexity of the model would depend on the number of input dimensions, d, as well as the sample size N. For reduced memory and computation, one is interested in reducing the dimensionality of the problem such that one has a new feature space $\mathbb{R}^k$ of dimension $k \ll d$. Decreasing d will also decreases the complexity of the algorithm and increase the learning process speed. The complexity is often broken into two parts: the complexity of training the algorithm, and the complexity of applying the trained algorithm to new data. Training does not happen very often, and is not usually time-critical. However, we often want a decision about a test point quickly, and there are potentially many test points when an algorithm is in use, so this needs to have low computational cost (Marsland, 2015).

When data can be explained with fewer features, one gets a better idea about the process that underlies the data and this allows knowledge extraction. There are two main methods for reducing dimensionality: feature selection and feature extraction. In feature selection, one is interested in finding k of the d dimensions that give the most information such that the other $(d - k)$ dimensions are discarded. In

feature extraction, one is interested in finding a new set of k dimensions that are combinations of the original d dimensions (Alpaydin, 2014).

During algorithm learning, a feature $x_i \in \mathbb{R}$ becomes relevant to a target concept t (what is being learned) if a pair of examples A and B in the instance space such that A and B exists differ only in their assignment to x_i (Blum and Langley, 1997). This implies that adjusting the value of x_i will affect the desired output of the learning algorithm. Hence it is important to identify the features that are more important for the problem. This invariably requires prior knowledge of the problem and the data. This process is called feature selection.

In any machine learning, algorithm technique selection is mostly driven by the structure of the dataset and type of problem one wishes to solve (regression or classification). Therefore it is important to have better understanding of the dataset and which features are useful prior to the selection and splitting of the dataset, as described in the previous paragraph. The output of the problem is real number, one therefore categorize the problem as regression. The identified algorithms that are applicable and practical to implement are multi-output regression algorithms to be discussed in more details in the next chapter.

2.4 Tuning model complexity

In prediction models, prediction errors can be decomposed into error due to variance and error due to bias of the model. There is a tradeoff between a model's ability to minimize the bias and variance during training (Fortmann-Roe, 2012).

2.4.1 Bias and variance

Understanding these two types of error can help us diagnose model results and avoid the mistake of over and underfitting. Understanding the bias and variance will help one to improve the data fitting process, resulting in accurate models (Fortmann-Roe, 2012). Error due to bias is the difference between the average prediction of models and the true value which one is trying to predict. For example, assuming one has multi-models built from a dataset generated randomly, this will result in the models yielding multi-predictions, hence the bias measures how far off in general these model's predictions are from the true value. Error due to variance is the variability of a model prediction for a given data point (Fortmann-Roe, 2012).

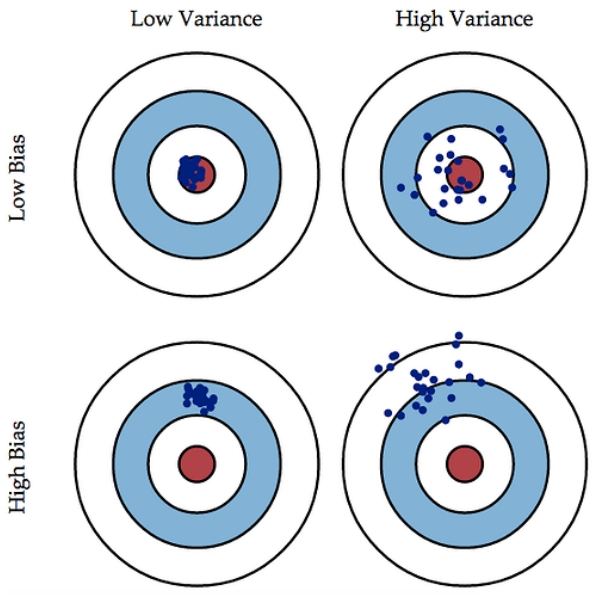


Figure 2.5: Graphical illustration of bias and variance. The center of each target represents a model that perfectly predicts the true value and as one moves away from the centre, the model gets worse at predicting the true value. Variability in the training dataset due to outliers or non-standard values will also result in poor prediction (Fortmann-Roe, 2012).

Suppose one has a training set $\mathbf{X} = \{x_1, \dots x_n\}$ and and a function $\mathbf{Y} = f(\mathbf{X}) + \epsilon$ with ϵ being an error term normally distributed with zero mean and variance σ^2. One wants to find a function $\widehat{f}(\mathbf{X})$ that estimates $f(\mathbf{X})$ as well as possible using some learning algorithm technique. In this case, one measures the mean squared prediction error at a point x as :

$$\begin{aligned} Err(x) &= \mathbb{E}\left[(\mathbf{Y} - \widehat{f}(x))\right] \\ &= \mathbb{E}\left[\mathbf{Y}^2 + \widehat{f}(\mathbf{x})^2 - 2\mathbf{Y}\widehat{f}(x)\right] \\ &= \mathbb{E}\left[\mathbf{Y}^2\right] + \mathbb{E}\left[\widehat{f}(x)^2\right] - 2\mathbb{E}\left[\mathbf{Y}\widehat{f}(x)\right] \end{aligned}$$

Since $\text{Var}(\mathbf{Y}) = \mathbb{E}\left[\mathbf{Y}^2\right] - \mathbb{E}\left[\mathbf{Y}\right]^2$, then we have $\mathbb{E}\left[\mathbf{Y}^2\right] = \text{Var}(\mathbf{Y}) + \mathbb{E}\left[\mathbf{Y}\right]^2$, such that

$$\begin{aligned} Err(x) &= \text{Var}(\mathbf{Y}) + \mathbb{E}\left[\mathbf{Y}\right]^2 + \text{Var}\left[\widehat{f}(x)\right] + \mathbb{E}\left[\widehat{f}(x)\right]^2 - 2f\mathbb{E}\left[\widehat{f}(x)\right] \\ &= \text{Var}(\mathbf{Y}) + \text{Var}(\widehat{f}) + (f(x) - \mathbb{E}\left[\widehat{f}\right]^2) \\ &= \sigma^2 + \text{Var}(\widehat{f}) + \text{Bias}\left[\widehat{f}\right]^2 \end{aligned}$$

where σ^2 represents the irreducible error (Fortmann-Roe, 2012).

There are various ways of managing the bias and variance during training, one of them being resampling and bootstrap aggregating. These techniques can be used to reduce the variance in model prediction. In bootstrap aggregating, replicates of the original dataset are created using a random selection with replacement. Each randomly selected data set is then used to construct a new model and the models are combined together into an ensemble. To make a prediction, all the models in the ensemble are averaged (Fortmann-Roe, 2012).

2.4.2 Overfitting and underfitting

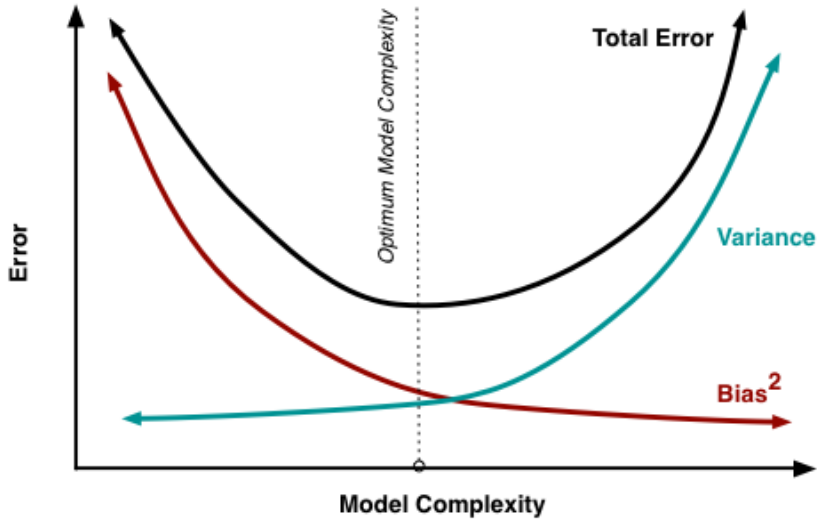


Figure 2.6: The figure is showing the bias and variance contribution to total error. Dealing with bias and variance is about dealing with overfitting and underfitting of the model. The bias is reduced and the variance is increased in relation to the model complexity. As the parameter space of the model increase, the more one experience complexity whereby the variance increases and becomes the primary concern while bias steadily decreases (Fortmann-Roe, 2012).

Understanding bias and variance is critical for understanding the behaviour of prediction models, but in general what one really cares about is the overall error, not the specific decomposition. The optimum point for any model is the level of complexity at which the increase in bias is equivalent to the reduction in variance as shown in the figure above. If model complexity exceeds this optimum point; we are in effect overfitting our model; while if our complexity falls short of the optimum point, we are underfitting the model (Fortmann-Roe, 2012).

2.5 Regression algorithm

Many different approaches are employed in machine learning regression. These approaches learn the relationship between the input and output by fitting a model directly from the data. The fitting process involves the adjustment and optimization of hyper-parameters to minimize the prediction error. This is usually done in an independent validation and testing dataset. In this study, we consider tree-based approaches: Decision tree, random forest, extremely randomized tree and the neighbourhood search approach: (K-nearest neighbor) to tackle our problem. Firstly, we formulate our regression estimation problem as follows. Suppose we have a feature matrix

$$\mathbf{X}_t = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{d1} & x_{d2} & x_{d3} & \dots & x_{dn} \end{bmatrix} = (x_{i,j}) \in \mathbb{R}^{d \times n}, i \in \{1, 2, \dots, d\}, j \in \{1, 2, \dots, n\}$$

where each column represents a vector of length d containing unique features,

$$\mathbf{x}_j = \begin{bmatrix} x_{1,j} \\ x_{2,j} \\ \vdots \\ x_{d,j} \end{bmatrix},$$

with each row a vector of length n, containing the n variable measurements for the ith observation, i.e

$$\mathbf{x}_i = \begin{bmatrix} x_{i,1} \\ x_{i,2} \\ \vdots \\ x_{i,n} \end{bmatrix},$$

and we also have a matrix of complex target variables to learn and predict on,

$$\mathbf{Y}_t = \begin{bmatrix} y_{11} & y_{12} & y_{13} & \cdots & y_{1m} \\ x_{21} & y_{22} & y_{23} & \cdots & y_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_{d1} & y_{d2} & y_{d3} & \cdots & y_{dm} \end{bmatrix} = (y_{k,l}) \in \mathbb{C}^{d \times m}, k \in \{1, 2, \ldots, d\}, l \in \{1, 2, \ldots, m\}$$

where each column represents a vector of length d, containing unique target variables as a function of time t

$$\mathbf{y}_l = \begin{bmatrix} y_{1,l} \\ y_{2,l} \\ \vdots \\ y_{d,l} \end{bmatrix},$$

with each row a vector of length m, containing the m variable measurements for the kth observation, that is

$$\mathbf{y}_k = \begin{bmatrix} y_{k,1} \\ y_{k,2} \\ \vdots \\ y_{k,m} \end{bmatrix},$$

The problem is to construct a learning machine, $M : \mathbf{X}_t \to \mathbf{Y}_t$, which when given a validation set of sensor examples, $\mathbf{X}_t^*$, minimises some measure of discrepancy between its prediction $M(\mathbf{X}_t^*) \approx \widehat{\mathbf{Y}}_t$, and the value of $\mathbf{Y}_t$, where M represents the predictor. We measure the discrepancy using four commonly used statistical measures in regression (Borchani et al., 2015): coefficient of determination, explained variance, mean squared error (MSE) and mean absolute error (MAE). The coefficient of determination is given by:

$$R^2\left[\mathbf{Y}_t, \widehat{\mathbf{Y}}_t\right] = 1 - \frac{\sum_k \left[(\mathbf{Y}_t)_k - (\widehat{\mathbf{Y}}_t)_k\right]^2}{\sum_k \left[(\mathbf{Y}_t)_k - \overline{\mathbf{Y}_t}\right]^2}, 0 \leq R^2 \leq 1, \tag{2.5.1}$$

which simply measures how well the future observations are to be predicted by M, with the best score being 1 for the best performing predictor where,

$$\overline{\mathbf{Y}_t} = \frac{1}{d_{\text{samples}}} \sum_k (\mathbf{Y}_t)_k, \tag{2.5.2}$$

is the mean value of $\mathbf{Y}_t$ draw from d samples, $\sum_k \left[(\mathbf{Y}_t)_k - \overline{\mathbf{Y}_t}\right]^2$ is the total sum of squares which measures the total variance in the predictor and $\sum_k \left[(\mathbf{Y}_t)_k - (\widehat{\mathbf{Y}}_t)_k\right]^2$ the residual sum of squares (deviations predicted from actual values of data), which measures the discrepancy between $\mathbf{Y}_t$ and $\widehat{\mathbf{Y}}_t$ (James et al., 2013).

The explained variance

$$V = \frac{\text{Var}\left[\mathbf{Y}_t - \widehat{\mathbf{Y}}_t\right]}{\text{Var}\left[\mathbf{Y}_t\right]}, 0 \leq V \leq 1, \tag{2.5.3}$$

measures the proportion to which the predictor accounts for the variation of the observed data $\mathbf{Y}_t$ (Bellinger et al., 2016). The best possible score is considered to be 1. where

$$\begin{aligned}
\text{Var}\left[\mathbf{Y}_t\right] &= \mathbb{E}\left[(\mathbf{Y}_t - \mu)^2\right], \ \mu = \mathbb{E}\left[\mathbf{Y}_t\right] \\
&= \mathbb{E}\left[\mathbf{Y}_t^2 - 2\mathbf{Y}_t\mu + \mu^2\right] \\
&= \mathbb{E}(\mathbf{Y}_t^2) - 2\mathbb{E}(\mathbf{Y}_t\mu) + \mu^2 \\
&= \mathbb{E}(\mathbf{Y}_t^2) - 2\mu^2 + \mu^2 \\
&= \mathbb{E}(\mathbf{Y}_t^2) - \mu^2,
\end{aligned}$$

(Fortmann-Roe, 2012)

MSE and MAE, which measure the average of the squared and absolute of the errors between the the observed values $\mathbf{Y}_t$ and the predicted $\widehat{\mathbf{Y}}_t$. For accurate prediction of the predictor, the value of MSE and MAE should converge to zero (James et al., 2013).

$$\text{MSE}\left(\mathbf{Y}_t, \widehat{\mathbf{Y}}_t\right) = \frac{1}{d_{\text{samples}}} \sum_k (\mathbf{Y}_t - \widehat{\mathbf{Y}}_t)_k^2 \tag{2.5.4}$$

$$\text{MAE}\left(\mathbf{Y}_t, \widehat{\mathbf{Y}}_t\right) = \frac{1}{d_{\text{samples}}} \sum_k \left|\mathbf{Y}_t - \widehat{\mathbf{Y}}_t\right|_k \tag{2.5.5}$$

.

The aim of this regression exercise is to predict multiple target variables $\widehat{\mathbf{Y}}_t$ hence it is referred to as multi-output regression. The learned model will then be used to predict multi-output values $\widehat{\mathbf{Y}}_{t+1}$ of all target variables of the new incoming unlabelled instances $\mathbf{X}_{t+1}$. It has been proven that multi-output regression methods provide means to model the multi-output datasets effectively and produce better predictive results. This method does not only consider the underlying relationships between the features and the corresponding targets but also the relationships between the targets themselves, thereby producing simpler models with better computational efficiency (Borchani et al., 2015). (Borchani et al., 2015) discuss several applications of multi-output regression including the challenges such as missing data, i.e., when some features or target variables are not observed.

2.5.1 Decision tree

Decision tree algorithms are popular methods for machine learning tasks such as classification and regression. These algorithms make it easy to handle categorical features, interpret, extend to the multi-output for regression settings and multiclass for classification setting. They can handle datasets that may have errors or missing values and are able to capture non-linearities and feature interactions (Cristina Petri, Cluj Napoca, 2010), hence they are widely used. One of the biggest advantages of using decision trees is the ease with which one can see what features or variables contribute to the classification or regression and their relative importance based on their location in the tree.

For a response variable that has classes, the tree algorithm (classification) organizes the dataset into groups by the response based on the similarities of the data, and when the response variable is numeric or continuous, the tree algorithm (regression) uses the data to predict the outcome by fitting the regression model in each of the independent variables (Morgan, 2014).

Decision trees are a simple but powerful form of multi-variable analysis. In effect, they perform a recursive binary partitioning of the feature space. The process of the parent node splitting into exactly two child nodes is refereed to as binary(Moisen, 2008), and it is said to be recursive if each child node wil, in turn, become a parent node, unless it is a terminal node, as illustrated in Figure 2.7, where the split occurs at the decision node.

Decision trees are built by growing a tree upside down, starting from the root node. The data sample passes down the tree through a series of splits, or nodes, at which a decision is made on the direction in which to proceed, either the left or right sub-tree, based on a sequence of questions about the features (Musicant et al., 2007). The next split or decision made depends on the answers to previous questions. Eventually, a terminal node is reached and the prediction is made.

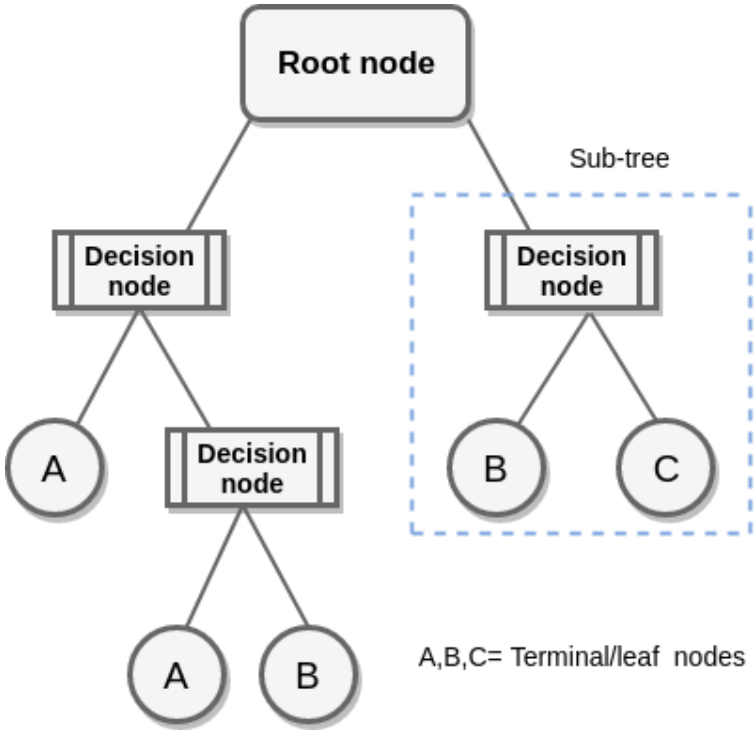


Figure 2.7: An illustration of a decision tree where a recursive partition is shown from the root node where there is an entire data sample to the internal nodes (decision nodes) where splitting occurs to form child nodes, which are either decision nodes or terminal nodes. A terminal node implies that after this split, further splitting of the samples does not provide relevant information in predicting the response variable (Moisen, 2008).

The binary recursive partitioning can apply to the fitting of both regression and classification trees. Each node in a tree is split based on its impurity, which is a measure of how badly the samples at that node fits the model defined in equation 2.5.6. Therefore, in regression and classification trees, variables and the locations of the split are chosen to minimize the impurity of the node at that point. However, the selection criteria of splitting each node are different for these two methods. In our case the for the regression scenario, the splitting rules used to minimize the impurity of each node c is the MSE and MAE as defined in equations 2.5.4 and 2.5.5 (Morgan, 2014). The sample mean of the response variable (estimated value μ_c) that represents the assigned prediction in each terminal node is defined as:

$$\mu_c = \frac{1}{N_c} \sum_{i \in N_c} \mathbf{Y}_i, \tag{2.5.6}$$

where N_c is the number of samples in each node, and $\mathbf{X}_c$ is the training samples.

These methods not only measure the impurity in each node, but are also used to measure the accuracy of the predictor, as in equations 2.5.4 and 2.5.5. Another commonly used strategy or measure in classification problems which also applies in regression problem, for selecting the best splits, from a set of possible split is to maximize the information gain at each tree node; i.e., the split chosen at each decision node is chosen from the the set $\mathrm{argmax}(\mathrm{IG}(\mathbf{X}_c, \mathrm{s}))$ where $\mathrm{IG}(\mathbf{X}_c, s)$ is the information gain when a split s is applied to sample data $\mathbf{X}_c$. This is the difference between the parent node (decision node) impurity and the weighted sum of the child nodes in equation 2.5.7.

$$\mathrm{IG}(\mathbf{X}_c, s) = \mathrm{Impurity}(\mathbf{X}_c) - \frac{N_{c(\mathrm{left})}}{N_c} \times \mathrm{Impurity}(\mathbf{X}_{c(\mathrm{left})}) - \frac{N_{c(\mathrm{right})}}{N_c} \times \mathrm{Impurity}(\mathbf{X}_{c(\mathrm{right})}), \quad (2.5.7)$$

where s is a split partitioning the training sample $\mathbf{X}_c$ of size N_c into two datasets $\mathbf{X}_{c(\mathrm{left})}$ and $\mathbf{X}_{c(\mathrm{left})}$ each of size $N_{c(\mathrm{left})}$ and $N_{c(\mathrm{right})}$. If the impurity measure is the MSE, we therefore define the information gain by

$$\mathrm{IG}(\mathbf{X}_c, s) = \mathrm{Impurity}(\mathbf{X}_c) - \frac{N_{c(\mathrm{left})}}{N_c} \times \mathrm{MSE}_{(\mathrm{left})} - \frac{N_{c(\mathrm{right})}}{N_c} \times \mathrm{MSE}_{(\mathrm{right})}. \quad (2.5.8)$$

If the impurity measure is the MAE, the information gain is defined by

$$\mathrm{IG}(\mathbf{X}_c, s) = \mathrm{Impurity}(\mathbf{X}_c) - \frac{N_{c(\mathrm{left})}}{N_c} \times \mathrm{MAE}_{(\mathrm{left})} - \frac{N_{c(\mathrm{right})}}{N_c} \times \mathrm{MAE}_{(\mathrm{right})}. \quad (2.5.9)$$

For multi-output decision trees that predict multiple continuous target attributes at once, the advantage over building a separate regression tree for each target is that the built tree is usually much smaller than the total size of the individual single-target decision trees for all variables; secondly they identify easily the dependencies between the different target variables.

The building of the multi-output decision trees follows the same steps as the standard decision tree in Figure 2.7, starting with all instances in the root node, then iteratively finding the optimal split and partitioning the nodes accordingly until a terminal node is reached, given a certain stopping criterion as shown in Figure 2.7. The only difference is the redefinition of the impurity measures in equations 2.5.4 and 2.5.5 of a node as the sum of the squared error over a multi-target response.

$$\sum_{k \in d} \frac{1}{N} \left[\sum_{l \in N} \left(\mathbf{y}_k^{(l)} - \mu^{(l)} \right)^2 \right], \quad (2.5.10)$$

where $\mu^{(l)}$ denotes the predicted output for the instance l in each node. Each split is selected to minimize the sum of the squared error such that each terminal leaf of the tree can be characterized by the multivariate mean of its instances and its defining feature values.

2.5.2 Random forest

Much work has been done in machine learning to improve the predictive ability of regression and classification trees by combining separate tree models into what is often called ensemble.

A random forest is a forest (ensemble) of decision tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest (Breiman, 2001). Figure 2.8 is illustrating how trees are built from randomly sampled data. The random forest algorithm was introduced by (Ho, 1995) after having noticed that decision trees were limited in their accuracy especially on large data samples owing to their inability to grow which led to overfitting. The random forest method was found not only to be accurate, but also less susceptible to overfitting.

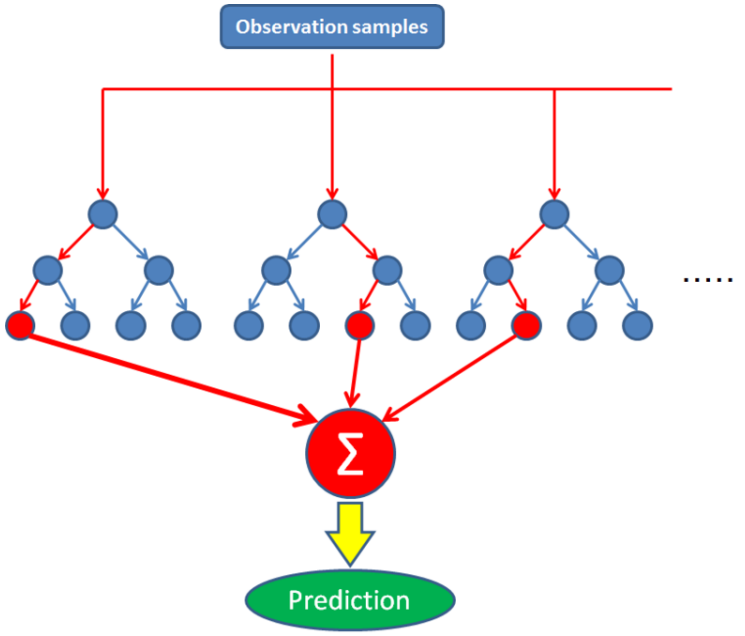


Figure 2.8: Random forest regression. [image by Data scientist TJO in Tokyo]

In random forest, the randomness is only introduced into the feature selection process, not into the choice of split points on the selected features. During the construction of a component decision tree, at each step of the split selection, a random forest first randomly selects a subset of features, and then carries out the split selection procedure for that particular tree within the selected feature subset. Random forest are considered as an extension of bootstrap aggregating (bagging), where the major difference with bagging is the incorporation of randomized feature selection (Zhou, 2012). The operational procedure of bagging is as follows:

- Given a set $\mathbf{X}$ of size N.
- Generate m subsets in vector Θ of length n by randomly selecting data from the set $\mathbf{X}$ with replacement (Breiman, 2001).
- Run k number of trees for each of the m sets of Θ to train the data, thereby Θ_m sets of trained data (Breiman, 2001).
- For the remaining n samples, trained trees from the previous two steps can be fitted onto the remaining data set of $\mathbf{X}_n$ to estimate the features of the data (Breiman, 2001).

The parameter k is the logarithm of the number of features (Breiman, 2001). This parameter controls the incorporation of randomness and this results in an increase in the bias. When $k =$ total number of features, then the constructed decision tree will be identical to the traditional deterministic decision tree, and when $k = 1$, a feature will be selected randomly.

The Random forest performs the prediction of its constructed trees by aggregating all the trees, i.e. (majority voting for classification or averaging for regression) to produce the final output. Suppose the underlying true function we try to learn is $f(\mathbf{x})$, and $\mathbf{x}$ is sampled according to a distribution p($\mathbf{x}$). The output of each decision tree learner can be written as the the true value plus an error item, i.e.,

$$h_i(\mathbf{x}) = f(\mathbf{x}) + \epsilon_i(\mathbf{x}), i = 1, \ldots, T. \tag{2.5.11}$$

such that we have the averaging and,

$$H(\mathbf{x}) = \frac{1}{T}\sum_{i=1}^{T} h_i(\mathbf{x}). \tag{2.5.12}$$

(Zhou, 2012)

Recall in Section 2.5.1, that the prediction of each tree is the sample mean, then from equation 2.5.11 we thus have the MSE of the output prediction h_i as

$$\int (f(\mathbf{x}) - h_i(\mathbf{x}))p(\mathbf{x})^2 d\mathbf{x} = \int \epsilon_i(\mathbf{x})^2 p(\mathbf{x}) d\mathbf{x}. \tag{2.5.13}$$

Similarly, we can derive the MSE of the combined decision tree learner (i.e., the ensemble) as

$$\int \left(f(\mathbf{x}) - \frac{1}{T}\sum_{i=1}^{T} h_i(\mathbf{x}) \right)^2 p(\mathbf{x}) d\mathbf{x} = \int \left(\frac{1}{T}\sum_{i=1}^{T} \epsilon_i(\mathbf{x}) \right) p(\mathbf{x}) d\mathbf{x} \tag{2.5.14}$$

The two main parameters of a random forest are the number of trees grown and the number of predictors randomly tried at each split. What one calls depth is sometimes found as the maximum node size, and controls the size of the trees that are grown. In the original implementation of random forest, trees are grown to the maximum potential extent so that they reach the lowest possible bias. Then, variance is reduced by growing many trees and averaging them. The main reason is that in random forest, one can only decrease error by reducing the variance, so the bias needs to be as low as possible in the first place. Therefore, in most cases, one does not really need to adjust this parameter (depth or max nodesize). but just has to make sure the default value allows the trees to grow as deeply as possible.

2.5.3 K-nearest neighbour

The K-nearest neighbor algorithm is a well known non-parametric algorithm commonly used for regression and classification problems. K-nearest neighbor is considered to be one of the simplest and yet powerful algorithms that does not make any assumptions about the distribution of the data being evaluated. The learning technique used in K-nearest neigbors is known as the instance-based learning. This technique is based on the memorization of the training dataset and the number of parameters is unbounded and grows with the size of the data.

The intuition underlying nearest neighbor regression is quite straightforward; the output predictions are obtained by looking into the memorized examples where the cost of the learning process is 0. The cost is in the computation of the prediction, with the prediction output being the continuous average values (or median) of the values of its K-nearest neighbors. The commonly used measure in K-nearest neighbors is the distance functions , i.e., consider $\mathcal{M}$ the m-dimensional Euclidean space $\mathbb{R}^m$, a training data set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathcal{M}$, a query point $q \in \mathcal{M}$, and a distance metric $d : \mathcal{M}^2 \to \mathbb{R}$. The K-nearest neighbor search is the task of finding the K closest (w.r.t. d) points to q from the data set $\mathbf{X}$, i.e., we find a set $\mathbf{A} \subset \mathbf{X}$ for which it holds that $|\mathbf{A}| = K$ and $d(q, \mathbf{x}) \leq d(q, \mathbf{y})\ \forall\ \mathbf{x} \in \mathbf{A}, \mathbf{y} \in \mathbf{X}$ or $\mathbf{A}$ (Hyvönen et al., 2015).

The distance function is defined as

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{m} (|x_i - y_i|^m)^{\frac{1}{m}} \tag{2.5.15}$$

known as the Minkowski distance. For a special case where $m = 1$, the distance function is a Manhantan distance also know as the L-1 norm

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{m} |x_i - y_i|, \tag{2.5.16}$$

and if $m = 2$, the distance function is a Euclidean distance, also know as the L-2 norm and,

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2} \tag{2.5.17}$$

(Cunningham and Delany, 2007).

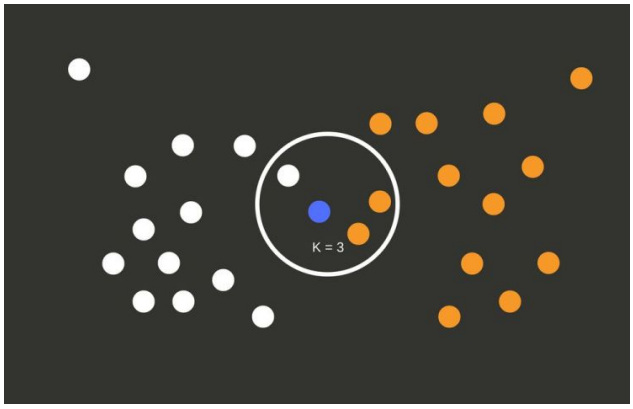


Figure 2.9: Example of K-nearest neighbor. The figure shows two different target classes orange and white circles, and one blue data point which has an unknown class. From the data points presented, we observe that the total number of the training samples with known classes is 26. To predict the target class for the blue circle, one consider the K value as three, and calculate the similarity distance using similarity measures such as Euclidean distance (K-nearest neighbor algorithm).

2.5.4 Extremely randomized tree

Extremely randomized tree is another tree-based ensemble method for supervised regression and classification problems. It builds an ensemble of decision trees according to the classical top-down procedure in Figure 2.8. Its differences from other tree-based ensemble methods like random forest are that the nodes are being split by choosing a threshold (cut-point) at random, and that it uses the whole training dataset rather than the bootstrap replica to grow the trees.

As in random forest, the predictions of the trees are also combined to produce the final prediction. This is done through a process of majority voting in classification problems and arithmetic averaging in regression problems as shown in equation 2.5.12. The explicit randomization of the cut-point and ensemble averaging reduces the variance even more, i.e. error = bias − variance (Geurts et al., 2006).

3. ZCal - calibrating radio interferometric data with machine learning

ZCal, in full "Zitha calibration" is a new machine learning radio data calibration experiment through which one makes use of the telescope sensor data to determine the relationship with calibrator solutions obtained using conventional calibration techniques. In Section 3.1, we further discuss the external parameters affecting the observed radio interferometric data, followed by a brief description of the KAT-7 telescope and Section 3.2 describes how the sensor data and calibration solutions are processed for training. In Section 2.5 we define and describe the regression algorithms used for our experiment, and in Section 3.3 we describe the training and testing procedure of the algorithms leading to the output results of our experiment in Section 3.4.1.

3.1 External parameters affecting the data

Astronomical observations done with interferometric radio telescopes suffer from various variable gain effects. The more the instrument, the more complex the gain effects the astronomers are likely to deal with. These can broadly be classified as direction-independent and direction-dependent errors. The complex instrumental gains due to the electronic devices in the telescope such as the receiver elements are directionally independent (Bhatnagar et al., 2008); as well as the tropospheric amplitude and phase effects; relative amplitude and phase gain for each antenna; amplitude response as a function of elevation (gain curve) and phase and amplitude drifts in the electronics of each antenna(Bhatnagar et al., 2008).

As defined in Chapter 1, *calibration* is the process of correcting for the signal corrupted by various mechanisms. Some of these mechanisms include external parameters such as fluctuations in air temperature, weather conditions on site and antenna pointing errors (Taylor et al., 1999). The antenna pointing error is the difference between the actual pointing position of the telescope and the requested (to go) position. Though an array consists of more than one antenna, each with a tracking centre, during a request ("POINT"), they must all follow the same required position. However, this is not always accurate, owing to misalignment of the elevation axis, atmospheric refraction, gravitational deformation of the structure, differential heating of the structure and wind loading (Taylor et al., 1999). Wind loading is an effect that degrades the pointing accuracy, thus the design of the telescope structure plays a role in terms of wind loads (Smith, 2002). Telescopes with a large dish structure are subjected to large wind loads. The wind speed and the geometric dimensions are the cause of highly turbulent flows around the antenna. There are violent shifts in airflow on the surface and around the antenna (Upnere et al., 2012). The following equation 3.1.1 shows the nominal expected sensitivity of the telescope without any errors:

$$\left[\frac{2kT_{sys}}{A_{eff}\sqrt{N(N-1)\tau\Delta\nu}}\right], \tag{3.1.1}$$

where (N is the number of antennas in the array, A_{eff} is aperture efficiency, τ is the integration time, $k = 1.38 \times 10^{-23} Jk^{-1}$ is the Boltzmann constant, $\Delta\nu$ is the bandwidth and $T_{sys} = \sum T_i$ an additive of noise contribution from different source such as atmosphere, ground, telescope surface and receivers) (Wilson et al., 2013).

3.2 KAT-7 telescope

Figure 3.1: The seven-dish MeerKAT precursor array, (Carignan et al., 2013)

The KAT-7 is a seven-dish interferometry that was built as an engineering prototype for techniques and technologies in preparation for the 64-dish Karoo Array Telescope (MeerKAT). These instruments are located in the Northern Cape Karoo desert region and are operated remotely from CapeTown. The construction of KAT-7 began in 2008 with the writing of the telescope requirements specification and was completed in 2010. It was then operated in engineering (commissioning) mode until its shut-down in 2016 (Foley et al., 2016). As seen in Table 3.1, KAT-7 antennas are extremely close together with baseline ranging between 26 and 185 m (Carignan et al., 2013). More parameter specifications are presented in the Tables 3.2 and 3.2:

Parameter	Value
Dish diameter	12 m
Number of antennas	7
Min baseline	26 m
Max baseline	185 m
Frequency range	1200-1950 MHz
Max instantaneous bandwidth	256 MHz
Polarization	Linear H & V
Systems temperature (T_{sys})	<35 K across the entire frequency band
Aperture effeciency	0.65
Location	latitude 30.7148° S, longitude 21.388 ° E
cryogenic temperature on receiver	80 K
Digital back end	Roach
Elevation	1054 m

Table 3.1: The instrument consists of 7 12-m dishes with linearly polarized feeds. The operational lower noise amplifier for each dish is $80\ K$ with system temperature $< 35\ K$. The frequency range of this instrument is $1200 - 1950$MHz with bandwidth of 256 MHz (Foley et al., 2016).

Parameter	Value
Pointing accuracy	25 ″
Wind (Operational)	36 km/h
Wind (Marginal Operation) km/h	45 km/h
Wind (Drive to Stow)	55 km/h
Wind (Survival)	160 km/h
Azimuth Rotation slew speed	2°/s
Azimuth limits	-175°, +285 °
Elevation slew speed	1°/s
Elevation limits	0°, 90 °
Feed/Cryo	Mass 75 kg

Table 3.2: The table shows the KAT-7 dish specification (Foley et al., 2016).

3.2.1 Sensor data

During scientific observations, data indexed by time are continuously collected from instruments and environment (atmospheric pressure, air temperature, wind conditions, and relative humidity, wind direction, etc). Samples as a function of time are recorded from software and sensors attached to the telescopes with their adjoining instrumentation (Slabber, 2017). Sampling strategies are set on sensors processed by the control and monitoring (CAM) system, ranging from several updates per second to infrequent updates. The typical sample in KAT-7 CAM contains the following attributes: a sensor name, sample timestamp, value timestamp, status and a value as shown in the following example, where red represents the sensor sample and green is the sensor sample description (Slabber, 2015; Slabber and Ockards, 2015).

```
{"name": wx_station_wind_speed,
"time": 1506430493.4778199196,
"status": nominal,
"value_ts": 1506430493.4774999619,
"value": 10.3}

#SENSOR_NAME A normalised form of a sensor name.

#SAMPLE_TS The UNIX timestamp of when the sample was
#first processed by the CAM system.

#VALUE_TS A UNIX timestamp when the acquisition
#was performed.

#STATUS indicate the state of the sensor when
#the sample was taken. Can have a value of UNKNOWN,
#WARN, NOMINAL, FAILURE, INACTIVE, ERROR, UNREACHABLE.
```

All the telescope components are interfaced to the CAM system via the Karoo Array Telescope Communication Protocol (KATCP), i.e KATCP is responsible for internal communication between CAM components and other subsystems (science data processing, correlator beamformer, etc) (Slabber, 2015). A KATCP interface exposes requests and sensors. To access the sensor information for analysis, users from different subsystems would connect to the CAM webserver using Python client and subscribe to sensor data of interest.

3.2.2 Preparation of training data

The objective of this study is to find correlations between calibration solutions and sensor information on the telescope. Therefore the main dataset for the study is the time-based sensor information of each antenna. The process of data collection encompasses all of the steps required to obtain the desired data in digital format. Methods of data collection include acquiring and archiving new observations, querying existing databases according to the science problem at hand, and performing as necessary any cross-matching or data combining (Ball and Brunner, 2010).

In every observation, the collected data are stored by the data capturing system in the Hierarchical Data Format (HDF5), which is a set of file formats designed to store and organize large amounts of data. The HDF5 file consists of two parts meta-data and observed visibilities. In meta-data one finds static information of the data set, including observer, dump rate and all the available subarrays and spectral windows in the data set selection criteria (antennas, channel frequencies, targets, scan information) and sensor data of interest as a function of time. The data observed by the radio telescope are in the form of complex numbers referred to as visibilities, as described in Chapter 1. Each source observed contains its own visibilities as a function of time along with sensor data, which keep a record of the telescope's activity and behaviour as these are observed.

In preparation for the training and testing dataset, we look at environmental sensors and instrumental sensors recorded during observations with a flux calibrator and a phase calibrator source PKS1613-586 in Figure 3.4. This was the most frequently used phase calibrator during the KAT-7 commis-

sioning phase. A series of observations for different tests with this calibrator 3.4 were taken in the period from December 2011 to November 2014. These tests includes Circinus X-1 monitoring at frequency $2022 - 1622.391$ MHz; OH maser monitoring at frequency $1666.481 - 1664.919$ MHz, $1666.381 - 1666.819$ MHz, $1666.281 - 1664.719$ MHz; spectral dynamic range test at frequency $1669.125-1662.877$ MHz, $1678.500-1653.506$ MHz; Circinus X-1 H absorption at frequency $1425.125-1418.827$ MHz, $1424.125-1417.877$ MHz and imaging new calibrators at frequency $2022-1622.391$ MHz. The chosen sensors of interest from each observation are:

1. Air temperature
2. Wind speed
3. Wind direction
4. Air pressure
5. Relative humidity
6. Actual refraction elevation
7. Actual refraction azimuth
8. Actual scan elevation
9. Actual scan azimuth
10. Actual pointing elevation
11. Actual pointing azimuth.

The environmental sensors measure the environmental conditions on site via the weather station installed around the telescope and the instrumental sensors (pointing) record the pointing azimuth and elevation activities of a telescope. These are examples of how each environmental and instrumental sensor are represented per antenna X.

```
{"name": anc_air_pressure,
"time": 1506430493.4778199196,
"status": nominal,
"value_ts": 1506430493.4774999619,
"value": 981.65}

{"name": anc_gust_wind_speed,
"time": 1506430493.4778199196,
"status": nominal,
"value_ts": 1506430493.4774999619,
"value": 3.96}

{"name": "anc_air_temperature",
"time": 1506430493.4778199196,
"status": "nominal",
"value_ts": 1506430493.4774999619,
"value": 23.96}

{"name": anc_air_relative_humidity,
"time": 1506430493.4778199196,
"status": nominal,
"value_ts": 1506430493.4774999619,
"value": 25.87}

{"name": anc_wind-direction,
"time": 1506430493.4778199196,
"status": nominal,
"value_ts": 1506430493.4774999619,
"value": 46.61}
```

```
{"name": antX_pos.actual-pointm-elev,
"time": 1506430493.4778199196,
"status": nominal,
"value_ts": 1506430493.4774999619,
"value": 89.97}

{"name": antX_pos.actual-pointm-azim,
"time": 1506430493.4778199196,
"status": nominal,
"value_ts": 1506430493.4774999619,
"value": -78.81}

{"name": antX_pos.actual-refrac-elev,
"time": 1506430493.4778199196,
"status": nominal,
"value_ts": 1506430493.4774999619,
"value": 89.80}

{"name": antX_pos.actual-refrac-azim,
"time": 1506430493.4778199196,
"status": nominal,
"value_ts": 1506430493.4774999619,
"value": -70.15}

{"name": antX_pos.actual-scan-elev,
"time": 1506430493.4778199196,
"status": nominal,
"value_ts": 1506430493.4774999619,
"value": 89.82}

{"name": antX_pos.actual-scan-azim,
"time": 1506430493.4778199196,
"status": nominal,
"value_ts": 1506430493.4774999619,
"value": 78.94}
```


We access this data from the HDF5 file using the KAT-7 data access library called katdal (The SKA telescope). We select the sensor data of interest with time index during the telescope tracking of the flux calibrator and phase calibrator PKS1613-586. The following is a Python code example showing how the sensor time index is selected from the observations for a specific calibrator.

```
#importing data access library
import katdal
import numpy
import time

#Opening the observation file
hdf5=katdal.open('1329697773.h5')
#Selecting data of interest from the the observation file
hdf5.select(scans='track',
            targets=['PKS1934-63','PKS1613-586'],
            channels=slice(199,800)
            )
#Obtaining sensor timestamps
h5df5_timestamps=[]
temp_time = np.empty((len(h5.timestamps)))
for i in range(0,len(h5.timestamps)):
    temp_time[i]=(h5.timestamps[i])
    timeh5.append(T.ctime(temp_time[i]))

#Selecting time during the scan of the calibrators
Time_Scan=[]
for scan in hdf5.scans() :
    #Taking average of start and endtime scan of calibrator source
    Average=((hdf5.timestamps[0]+
              hdf5.timestamps[len(hdf5.timestamps)-1])/2
             )
    Time_Scan.append(time.ctime(Average))

#obtaining time index of telescope sensors and scan time.
Time_index=numpy.where(np.in1d(numpy.unique(h5df5_timestamps),
                       Time_Scan))[0]
```

The katdal library can manipulate the HDF5 observation file in many ways, including file format conversion from HDF5 to measurement sets (MS). This option is useful for astronomers using CASA for data calibration and visualization (Foley et al., 2016). This conversion also provides an option whereby one flags all channels affected with known radio frequency interference (RFI). Once the observation file has been converted to ms format, hand flagging for unknown RFI is performed inside CASA using the task *flagdata* for H and V cross correlation products. After satisfactory flagging of the data, one then proceeds to the calibration step.

The systematic time-dependent complex gain errors denoted by J_{ij} (where i, j represent the baseline of two or more antennas) are the most dominant calibration effect in the visibility data, and a solution for them is necessary before proceeding with any other calibration (Ott and Kern, 2013). In this dissertation we only focus on the 1GC calibration with an interest in obtaining complex gain calibration solutions for the phase calibrator PKS1613-586. In the process of 1GC, as explained in chapter 1, the complex gain calibration (*gaincal*) CASA task determines the time-based complex antenna gains solutions G (amplitude and phase) for each polarization H & V and each spectral window, from the visibility data of the specified phase calibration source PKS1613-586 and stores these in calibration tables.

These solutions are calculated based on reference antenna 5 in our case, since it is an antenna that is

reasonably close to the center of the array and it was known to be particularly stable with no drop-outs during KAT-7 commissioning (Ott and Kern, 2013). The task *gaincal* in CASA factors $J_{i,j}$ into antenna-based components, so one does not see a set of values for each baseline, but rather only one for each antenna (J_i) (CASA cookbook, 2009).

Since our phase calibrator PKS1613-586 has an unknown flux density, the obtained values of the antenna based amplitude solutions differ from the correct ones by a factor depending only on the true source flux density by

$$\frac{1\ \mathrm{Jy}}{(\text{true source flux density})^{0.5}}, \text{ where } 1\ \mathrm{Jy} \text{ is the assumed flux density for PKS1613-586.} \quad (3.2.1)$$

We then used one of the following flux-density calibrators with known flux models: PKS1934-63, 3C147, 3C286 tied to the Perley-Taylor 99 to scale the correct gain amplitude of the phase calibrator in Figure 3.4 so that it match the scale amplitude solutions for the flux-density calibrators as well as possible by running the CASA task *fluxscale* (CASA cookbook, 2009). This process is called *bootstrapping*. After obtaining the final table with correct amplitude and phase complex gain solutions for the phase calibrator in Figure 3.4, we then used the complex solutions matrix written to the "CPARAM" column to generate the training and testing calibration solution data per antenna i. An example of the complex gain amplitude and phase solutions is shown in Figures 3.3 and 3.2. These solutions were obtained using the CASA task $gaincal$ described in Chapter 1.

The following code shows how the parameters were set in standard practice of 1GC as described in chapter 1 section 1.5.

```
#Set flux for flux calibrator
default(setjy)
vis = msfile
field = flux_calibrator
fluxdensity  = -1
standard = 'Perley-Taylor 99'
setjy()

#Bandpass calibration
default(bandpass)
vis = msfile
caltable = bandpasstable
field = flux_calibrator
refant = reference_antenna
solnorm = True
combine = 'scan'
solint = 'inf'
bandtype = 'B'
minsnr = 5
interp = ['nearest']
bandpass()
```

```
#Calculating G solution for flux calibrator
default(gaincal)
vis = msfile
caltable = gaintable
field = flux_calibrator
solint = 'inf'
refant = reference_antenna
gaintype = 'G'
calmode = 'ap'
solnorm = False
minsnr = 5
gaintable = [bandpasstable]
interp = ['nearest']
gaincal()

#G solution for gain calibrator
default(gaincal)
vis = msfile
caltable = gaintable
field = gain_calibrator
solint = 'inf'
refant = reference_antenna
gaintype = 'G'
calmode = 'ap'
append=True
solnorm = False
minsnr = 5
gaintable = [bandpasstable]
interp = ['nearest']
gaincal()

#Scaling flux density for phase calibrator
default(fluxscale)
vis = msfile
caltable = gaintable
fluxtable = fluxtable
reference = flux_calibrator
transfer = gain_calibrator
fluxscale()
```

(CASA cookbook, 2009)

A multi-frequency synthesis image deconvolution was carried out using the *CLEAN* CASA task with an image size of 256×256 and cell size of 30 arc seconds. The image rms noise in Figure 3.4 was measured using the rms noise of the 0 iteration residual image.

```
clean(vis='CircinusX-1.split.ms',
imagename='PKS1613586.im',
niter=10000,threshold='3.0mJy',
psfmode='hogbom',
interactive=False,
imsize=512,
cell='30arcsec',
stokes='I',
weighting='briggs',
mode='mfs',
usescratch=True)
```

(CASA cookbook, 2009)

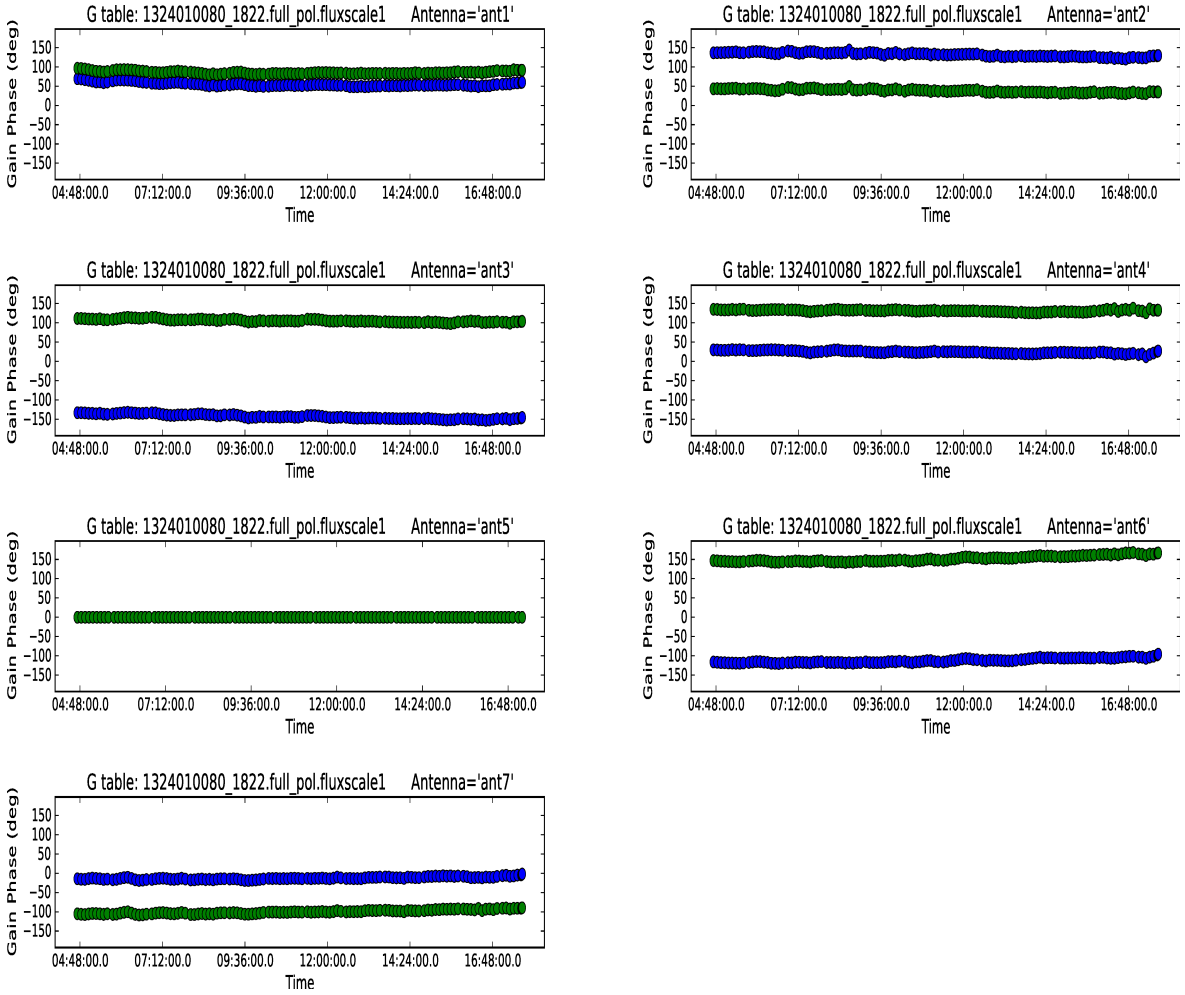


Figure 3.2: Complex gain phase solutions for phase calibrator PKS1613-586 obtained from CASA per antenna i. Blue is H-polarization and green is V-polarization. Antenna 5 shows zero phase since it is used as a reference antenna.

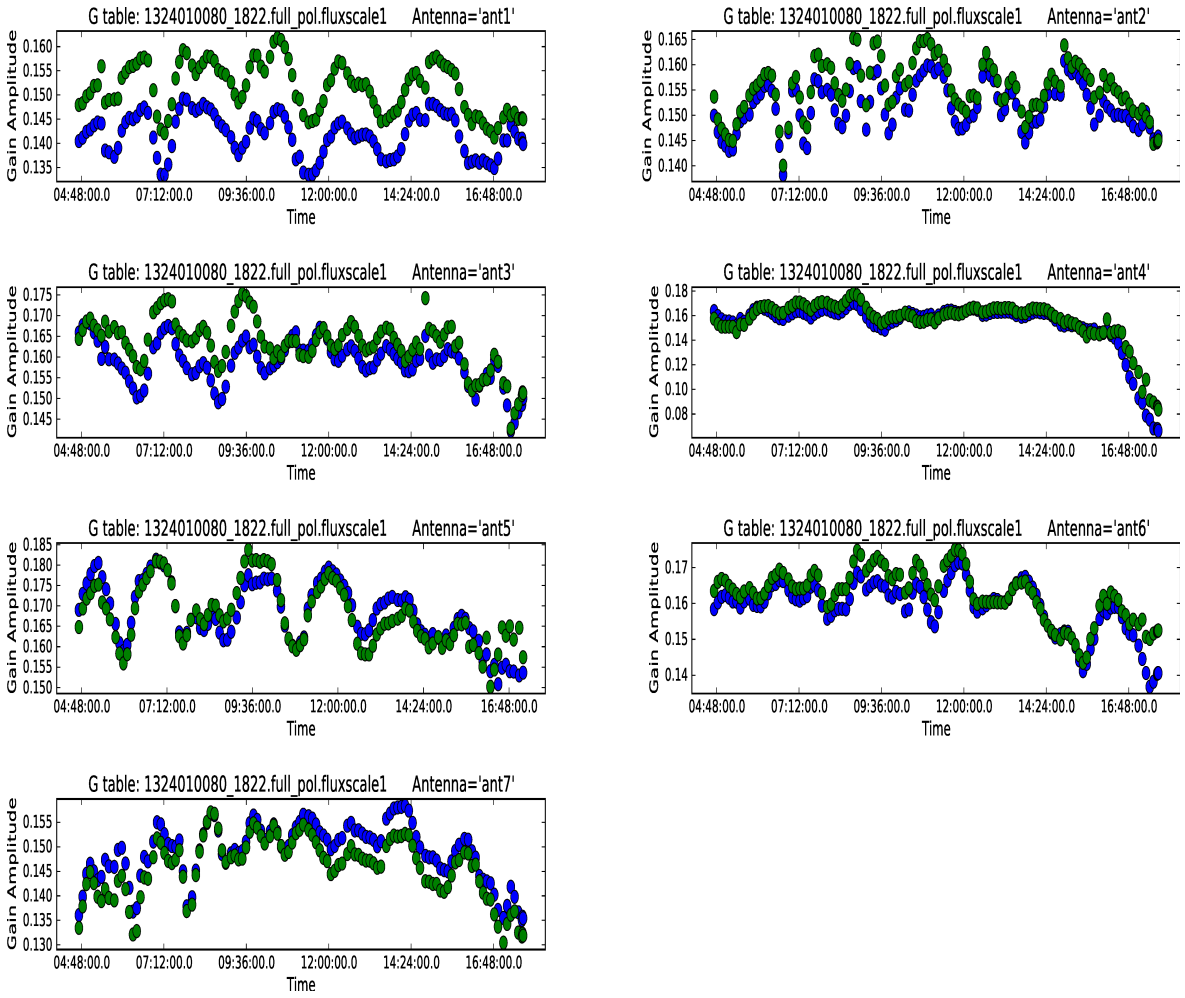


Figure 3.3: Complex gain amplitude solutions for phase calibrator PKS1613-586 obtained from CASA per antenna i. Blue is H-polarization and green is V-polarization.

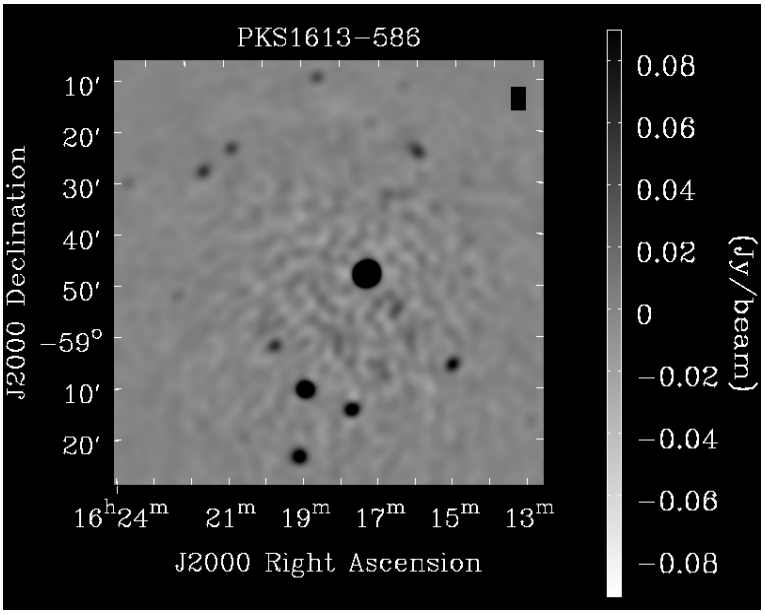


Figure 3.4: An image of the phase calibrator PKS1613-586 located at RA, Dec (J2000): 16:17:17, -58:48:07. Its flux density in Jy at frequency $= 1.83\mathrm{GHz}$ is 4.6 ± 0.015. This flux scaling was carried out in CASA using a strong flux calibrator. The expected rms noise of this image was calculated to be $0.08\mathrm{Jy}$ using *CLEAN* task in CASA.

3.3 Training and testing

The process of training and testing involves feeding a learning algorithm with training data (seen data) to learn from by finding suitable parameters and evaluate its performance using testing data (unseen), as illustrated in Figure 3.5. A fundamental assumption of machine learning is that the distribution of the training data are the representative from which future data will be chosen (Witten et al., 2016). Each instance in the training set contains "target value" and several "attributes" (namely features).

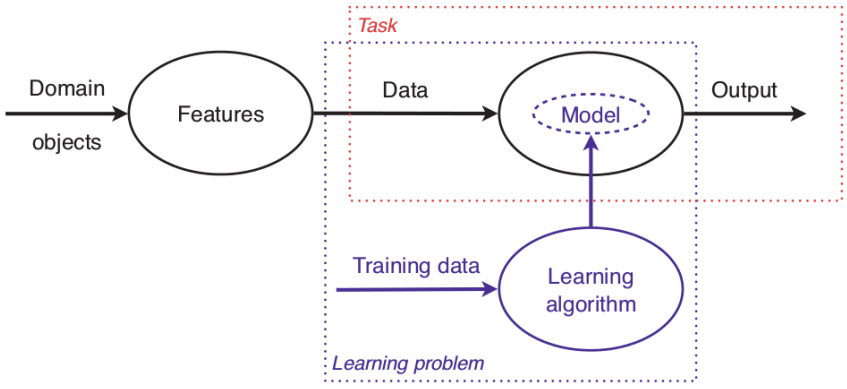


Figure 3.5: An overview of machine learning model training (Flach, 2012).

Referring to Section 3.2.2, some challenges and difficulties of gathering a larger data set were encountered

due to faulty radio telescopes leading to observations being conducted with fewer elements in an array, observations with target of interest PKS1613-586 conducted using only the baselines of interest (short or long), KAT-7's lifespan lasting for a short period, and radio frequency interference affecting calibration solutions. The advantage of having a large testing dataset is a more reliable estimate of accuracy, whereas a large training set is more representative of how much data are available for the learning process.

Despite the challenges mentioned above, we managed to construct a data matrix $\mathbf{M}$ of dimension 1760×75 containing sensor data $\mathbf{X}_t \in \mathbb{R}^{1760\times 47}$ and 1GC calibration solutions $\mathbf{Y}_t \in \mathbb{C}^{1760\times 14}$ from 28 different observations at integration time Δt ranging from $1 - 12$ hours dependending on each observation requirement. Take note that the 14 columns in plane $\mathbb{C}$ represent the amplitude and phase solutions for H,V polarization, i.e. 7×2; and the 47 columns in plane $\mathbb{R}$ are the number of selected sensors for all the antennas. Since each calibration solution in $\mathbf{Y}_t$ is represented as a complex variable

$$Ae^{i\phi} = A(\cos\phi + i\sin\phi), \tag{3.3.1}$$

for each polarization H $\&$ V. Due to different physical causes on the received signal, we therefore choose to treat the antenna phases and amplitudes separately by splitting equation 3.3.1, into gain amplitude solutions $\left|Ae^{i\phi}\right|$ and gain phase solutions ϕ in radians for both H $\&$ V. In every supervised machine learning problem, it is crucial to label the input and output of the dataset before starting the learning process. In our case, since this is a regression problem aiming to predict calibration solutions from sensor data, we therefore label the sensor data as input features and the calibration solutions as output targets.

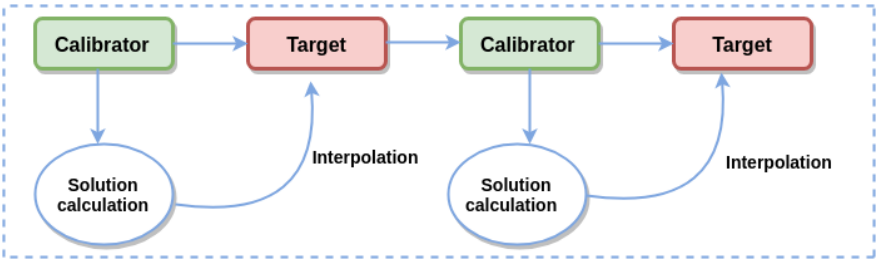


Figure 3.6: Observation procedure and calibration using CASA

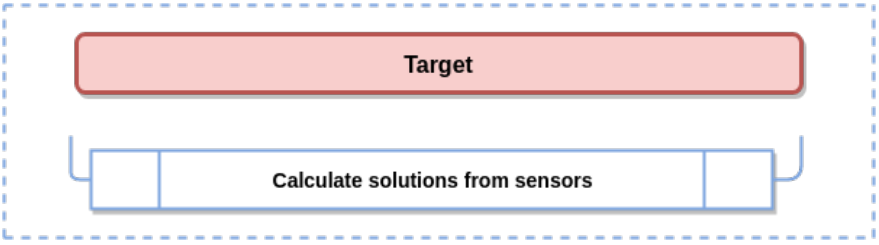


Figure 3.7: Observation procedure goal using machine learning

Figure 3.6 shows the observation procedure of KAT-7 where the calibrator is close to the target source (preferably within the range of 10° to 15°), with the scan switching between the target and the calibrator (Taylor et al., 1999). This is to minimize the phase perturbation caused by atmosphere. The calibration solutions are obtained from the phase calibrator ("green") in Figure 3.6 at different time intervals Δt of tracking called duty cycles. This is done assuming that the calibrator PKS1613-586 has invariant

characteristics (i.e. known positions in the sky or known properties from previous observations). One of the standard properties for an ideal instrument would be the 0° antenna-based gain phase response and constant amplitude for all baselines. In reality this is not always the case because of instrumental defects such as phase and amplitude drifts in the electronics of each antenna; amplitude response as a function of elevation (gain curve), and tropospheric amplitude and phase effects and other external effects. Hence, to calculate the calibration solutions, CASA considers each antenna-based gain J_i and the prior information about the calibrator to perform some adjustments or estimations G over the calibrator's duty cycle Δt for all the instruments i, j. These solutions at each duty cycle are then temporarily interpolated using the *interp* parameter inside the *gaincal* CASA task.

After obtaining solutions G, they are written into calibration tables and later applied to the target source of interest, T. The CASA task for this process is called *applycal*, which reads the specified calibration tables, applies them to the observed data column through an interpolation process at each single time unit for different target duty cycles $\Delta t, \Delta t_{n+1}, \ldots,$ following the order shown in Figure 3.6. The calibrated results are then written into the corrected column for imaging or other scientific processing (Ott and Kern, 2013). Since CASA "calculates" the solutions using interpolation and estimation methods given some prior information about the calibrators, the goal of the experiment is to produce a model (based on the training data) that learns the behaviour and changes of the calibrator owing to various external effects and able to predict the calibration solutions of the test data with high accuracy, given only the test sensors at different duty cycles. This will help in speeding up the calibration processes and decreasing the time duration for tracking the phase calibrator PKS1613-586 and therefore increasing the time duration for tracking the target observed as shown in Figure 3.7.

Since the sensor data obtained from the radio telescope are of different scales, i.e., unstructured. a pre-processing stage before proceeding with the training of the learning algorithms is necessarily. We propose to use a scaling statistical technique called Z-score normalization or standardization. This technique gives the normalized values or range of data from the original unstructured data using the concept of mean and standard deviation (Patro and Sahu, 2015). The features are rescaled such that they have properties of a standard normal distribution with mean $\mu = 0$ and standard deviation $\sigma = 1$.

$$Z = \frac{\mathbf{x} - \mu}{\sigma} \tag{3.3.2}$$

with mean:

$$\mu = \frac{1}{N} \sum_{j \in N} (\mathbf{x}_j),$$

and standard deviation:

$$\sigma = \sqrt{\frac{1}{N} \sum_{j \in N} (\mathbf{x}_j - \mu)^2}$$

Normalizing the features so that they are centered around 0 with a standard deviation of 1 is not only important if one is comparing measurements that have different units, but is also a general requirement for many machine learning algorithms as this will be helpful for prediction (Bott, 2014).

When training a learning algorithm in machine learning, two main things might happen as described in Section 2.4, namely overfitting and underfitting. To avoid these, we split the data into two subsets, i.e., training and testing data sets using the *scikit-learn* train-test split strategy (Buitinck et al., 2013).

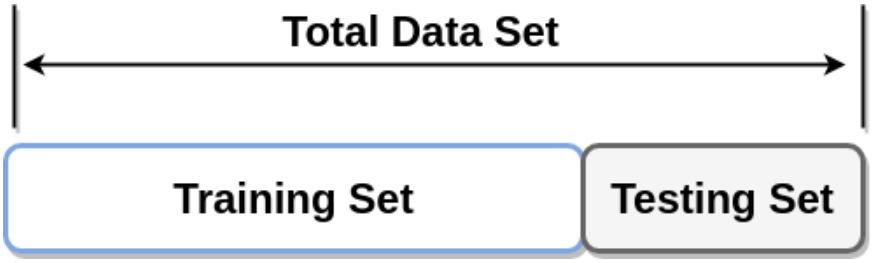


Figure 3.8: This is a train-test split strategy where a data matrix is randomly split into training and testing subsets.

From Figure 3.8, once the data have been normalized, we perform a 80% training and 20% testing split. The training set contains a known output and the model will learn using these data in order to be generalized to other data later on. The test dataset is used to test the model's prediction on this subset.

Model training in machine learning requires tuning of the hyper-parameters that determine the behaviour of the learning algorithm and hence the performance of the resulting model on unseen data. This involves finding the best combination of hyper-parameters with respect to the user-specified criterion (Buitinck et al., 2013). For this task, we make use of a randomized search cross-validation (RandomizedSearchCV) function by *scikit-learn* library. This library provides efficient and well-established machine learning tools in a programming environment that is user friendly and accessible to non-machine learning experts, and reusable in various scientific areas (Buitinck et al., 2013). In a randomized search cross-validation algorithm, not all hyper- parameters are tried out; instead it samples a fixed number of hyper-parameters from a specified probability distribution. i.e., given a set of parameters, p_i, each with N_i different values, search all combinations $\prod_i N_i$ with random sampling.

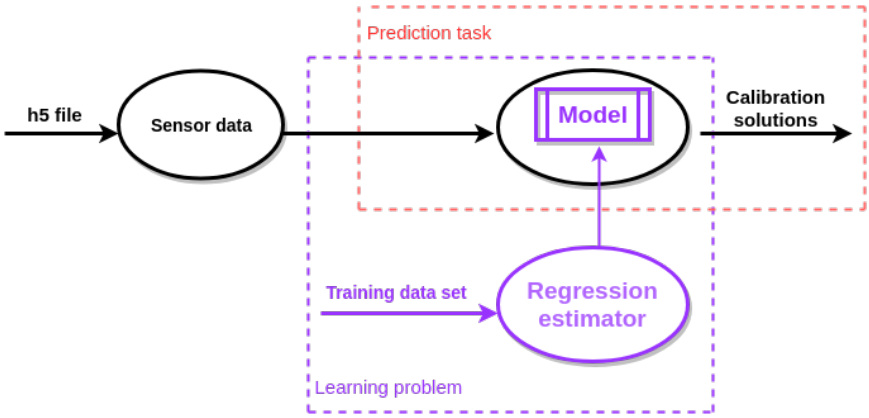


Figure 3.9: Regression learning diagram showing how we built a regression model estimator by taking in an observation file as input and extracting the sensor data per calibrator scan to obtain an accurate model that predict calibration solutions.

Decision tree estimator p_i					
splitter	max features	min sample split	min sample leaf	max depth	cv
[best,random]	[log2 sqrt auto]	$\{p_i : 30 \leq p_i \leq 60\}$	$\{p_i : 7 \leq p_i \leq 14\}$	$\{p_i : 700 \leq p_i \leq 1389\}$	10

Table 3.3: Decision hyper-parameters

Random forest estimator p_i				
n-estimators	max features	max depth	min sample leaf	cv
$\{p_i : 100 \leq p_i \leq 1200\}$	[log2 sqrt auto]	$\{p_i : 20 \leq p_i \leq 200\}$	$\{p_i : 3 \leq p_i \leq 50\}$	10

Table 3.4: Random forest hyper-parameters

Extremely randomized estimator p_i				
n-estimators	max features	max depth	min sample leaf	cv
$\{p_i : 100 \leq p_i \leq 1200\}$	[log2 sqrt auto]	$\{p_i : 20 \leq p_i \leq 200\}$	$\{p_i : 3 \leq p_i \leq 50\}$	10

Table 3.5: Extremely randomized tree hyper-paramaters

Note that since the random forest estimator 3.4 and extremely randomized estimator 3.5 are tree-based algorithms, they share the same hyper-parameters as the decision tree estimator 3.3, where

1. *splitter*: is the strategy used to choose the split at decision node, with p_i best representing the best split and random representing the random split.
2. *max features*: defines the maximum number of features to be considered in the tree when making the best split. The best parameters p_i to choose from are :
   - if p_i is $\log_2$ then the maximum features is $\log_2$ (*max features*)
   - if p_i is sqrt then the maximum features is $\sqrt{\textit{max features}}$
   - if p_i is auto, then this will simply take all the features that make sense in every tree. Here we simply do not put any restrictions on the individual tree.
3. *min sample split*: denotes the minimum number of samples required for a split in each decision node.
4. *max depth*: denotes the maximum depth of the tree, i.e. determines how deeply the tree should expand. If no parameter is given, then nodes are expanded until all leaves are pure or until all leaves contain less than min samples split.
5. *min sample leaf*: denotes the number of samples required for a node to be a terminal node (leaf node).

6. *n-estimator*: denotes the number of trees to be built before taking the maximum voting or averages of predictions. A higher number of trees give better performance but make decrease the speed of processing.

7. *cv*: denotes the cross-validation generator inside RandomizedSearch, which determines the cross-validation strategy, if p_i is an integer, then the number of folds in a KFold are specified (Pedregosa et al., 2011).

K-nearest neighbor estimator p_i					
n-neighbors	weights	algorithm	leaf size	p	cv
$\{p_i : 20 \leq p_i \leq 200\}$	[uniform,distance]	[auto ball-tree kd-tree brute]	$\{p_i : 30 \leq p_i \leq 150\}$	[2,3]	10

Table 3.6: K-nearest neigbor hyper-parameters

One of the most attractive features of the K-nearest neighbor algorithm is that is simple to understand and easy to implement. As shown in Table 3.6, few hyper parameters p_i are tried out, hence it is called a lazy learning algorithm.

1. *n-neighbors*: denotes the number of the K-nearest neighbors of the unknown observation. When K is small, one restrains the region of a given prediction and forces the predictor to limited in seeing the overall distribution, and a higher K averages more voters in each prediction and hence is more resilient to outliers.

2. *weights*: denotes the weight contribution of each point to the prediction of a query point in the local neighborhood.

   - If weight $= uniform$, then all points are weighted equally.
   - If weight $= distance$, then the weights proportional to the inverse of the distance from the query point is assigned.

3. *algorithm*: specifies which algorithm should be used to compute the nearest neighbors and takes in values such as:

   - If the algorithm is auto, then the algorithm attempts to determine the best approach from the training data.
   - the ball-tree, kd-tree algorithm are used to implement the ball tree algorithm. These data structures are used for fast high-dimensional nearest-neighbor searches.
   - the brute algorithm is used to implement the brute-force search algorithm.

4. *leaf size*: leaf size is passed to the ball-tree or kd-tree approach for finding k-nearest neighbors in.

5. *p*: denotes the power parameter for the Minkowski metric (Pedregosa et al., 2011).

We used a RandomizedSearchCV algorithm with number of iterations n-iter $= 50$ to obtain the following best estimator for each learning algorithm:

```
DecisionTreeRegressor(criterion='mse',
                      max_depth=890,
                      max_features='log2',
                      max_leaf_nodes=None,
                      min_impurity_split=1e-07,
                      min_samples_leaf=13,
                      min_samples_split=30,
                      min_weight_fraction_leaf=0.0,
                      presort=False,
                      random_state=0,
                      splitter='best')
```

	max-depth	max features	min samples leaf	min samples split	splitter	score
optimized p_i	890	$\log_2$	13	30	best	0.686

Figure 3.10: Decision tree optimized hyper-parameters

```
RandomForestRegressor(bootstrap=True,
                      criterion='mse',
                      max_depth=90,
                      max_features='auto',
                      max_leaf_nodes=None,
                      min_impurity_split=1e-07,
                      min_samples_leaf=3,
                      min_samples_split=2,
                      min_weight_fraction_leaf=0.0,
                      n_estimators=700,
                      n_jobs=1,
                      oob_score=0,
                      random_state=0,
                      verbose=0,
                      warm_start=False)
```

	max-depth	max features	min samples leaf	n-estimators	score
optimized p_i	90	auto	3	700	0.902

Figure 3.11: Random forest optimized hyper-parameters

```
KNeighborsRegressor(algorithm='brute',
                    leaf_size=30,
                    metric='minkowski',
                    metric_params=None,
                    n_jobs=1,
                    n_neighbors=20,
                    p=3,
                    weights='distance')
```

	algorithm	n-neighbors	p	weights
optimized p_i	brute	20	3	distance

Figure 3.12: KNN optimized hyper-parameters

```
ExtraTreesRegressor(bootstrap=False,
                    criterion='mse',
                    max_depth=90,
                    max_features='auto',
                    max_leaf_nodes=None,
                    min_impurity_split=1e-07,
                    min_samples_leaf=3,
                    min_samples_split=2,
                    min_weight_fraction_leaf=0.0,
                    n_estimators=700,
                    n_jobs=1,
                    oob_score=0,
                    random_state=0,
                    verbose=0, warm_start=False)
```

	max-depth	max features	min samples leaf	n-estimators	score
optimized p_i	90	auto	3	700	0.905

Figure 3.13: Extremely randomized tree optimized hyper-parameters

3.3.1 Feature importance

Random forests and extremely randomized trees are well established ensembles of tree models that not only produce good predictive performance, but also provide rich feature importance information (Kazemitabar et al., 2017). Generally, importance provides a score that indicates how valuable each feature was during the construction of each decision tree in the model. The importance of a feature is calculated by the amount that each split point improves the performance measure, weighted by the number of observations the node is responsible for in each decision tree (Kazemitabar et al., 2017). The performance measure can be the average of the impurity reductions over all nodes in the tree where a split was made on that variable, with impurity reductions weighted to account for the size of the node (Kazemitabar et al., 2017).

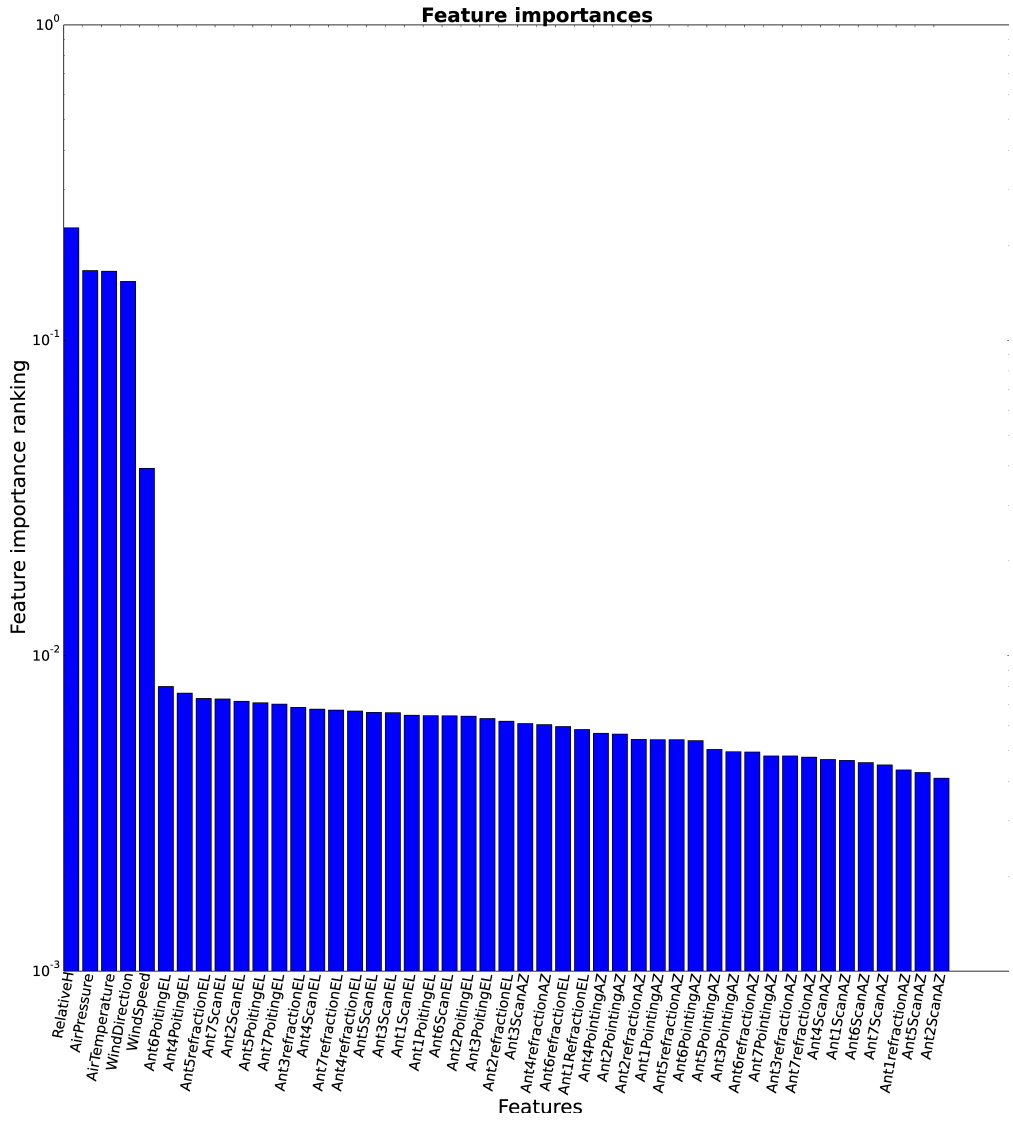


Figure 3.14: Feature importance

Figure 3.14 shows the importance of each feature in each learning algorithm i.e. the values with the highest contribution in building an accurate model. One observes that the environmental features such as air temperature, relative humidity, wind speed, wind direction and air pressure are higher ranked than the pointing sensors. The pointing features are ranked second, with the telescope elevation sensors being higher than the telescope azimuth sensors.

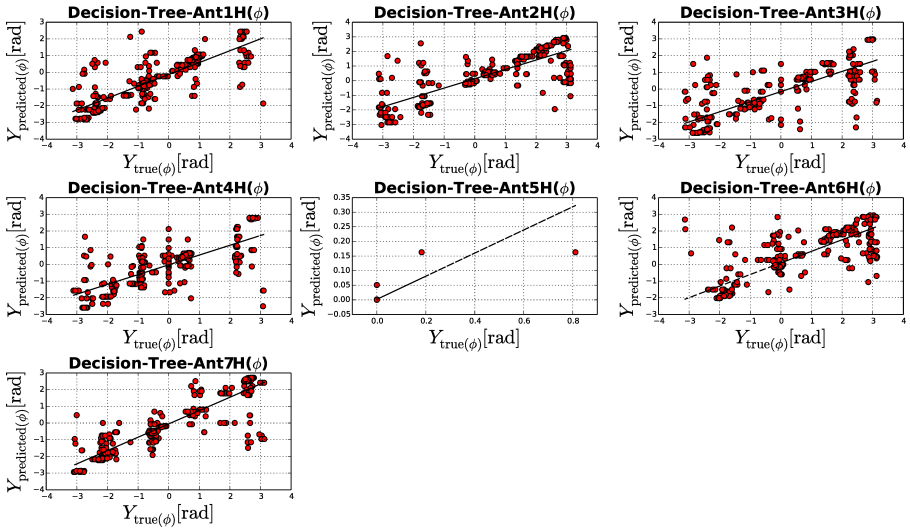
3.4 Results

3.4.1 Testing on test-data

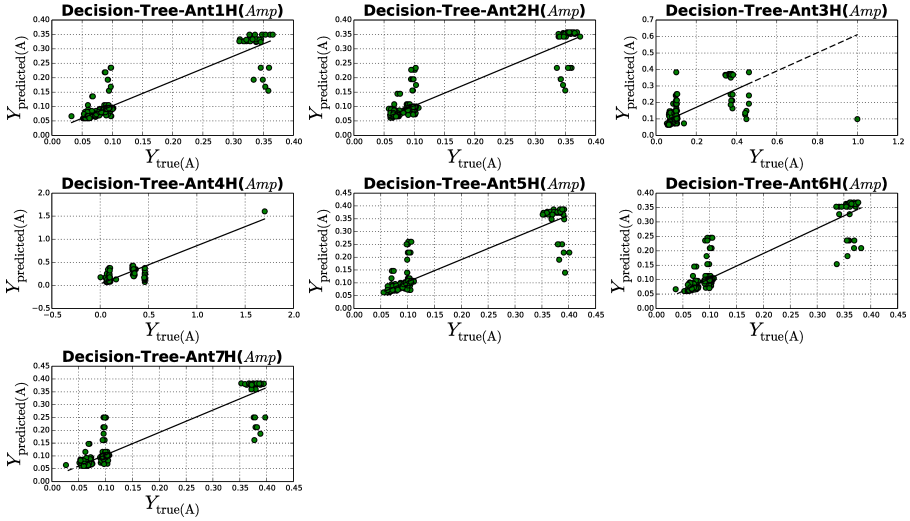
This section shows the plots of the predictions for the decision tree, random forest, extremely randomised trees and the K-nearest neighbor. The algorithms are run a number of times on the learning sample

and tested on the test samples to estimate the prediction errors.

3.4.2 Decision tree training

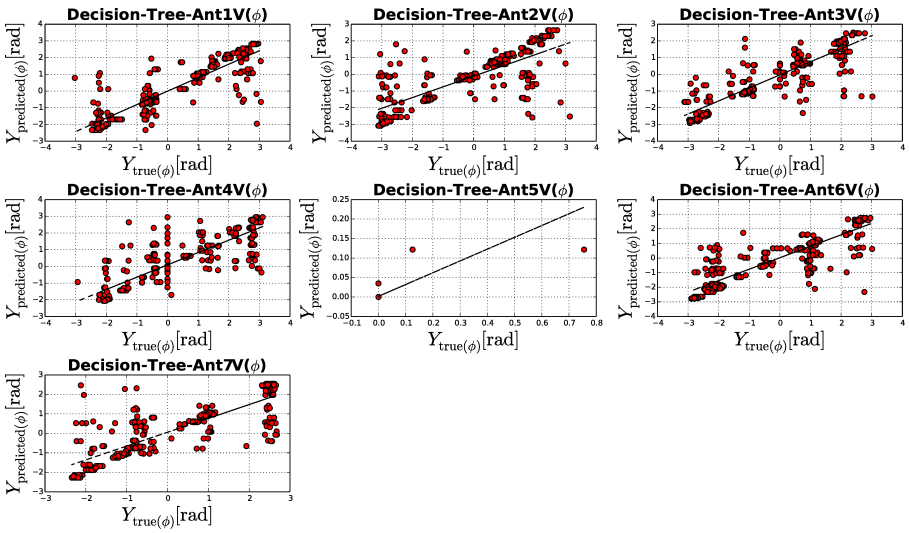


(a) Phase gain solutions for H-polarization

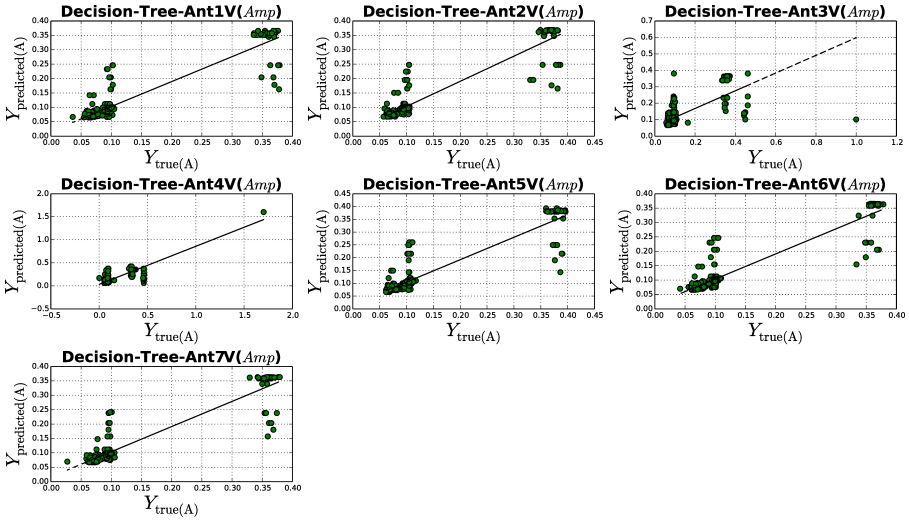


(b) Amplitude gain solutions for H-polarization.

Figure 3.15: Results obtained from the decision tree learning algorithm with randomized search optimization algorithm. (a) and (b)are the predicted gain solutions $\mathbf{Y}_{\text{predicted}}$ by the decision tree vs the true gain solutions $\mathbf{Y}_{\text{true}}$ (CASA) for H-polarization. We observe that the decision tree algorithm is performing badly in predicting the H-polarization amplitude and phase gain solutions



(a) Phase gain solutions for V-polarization



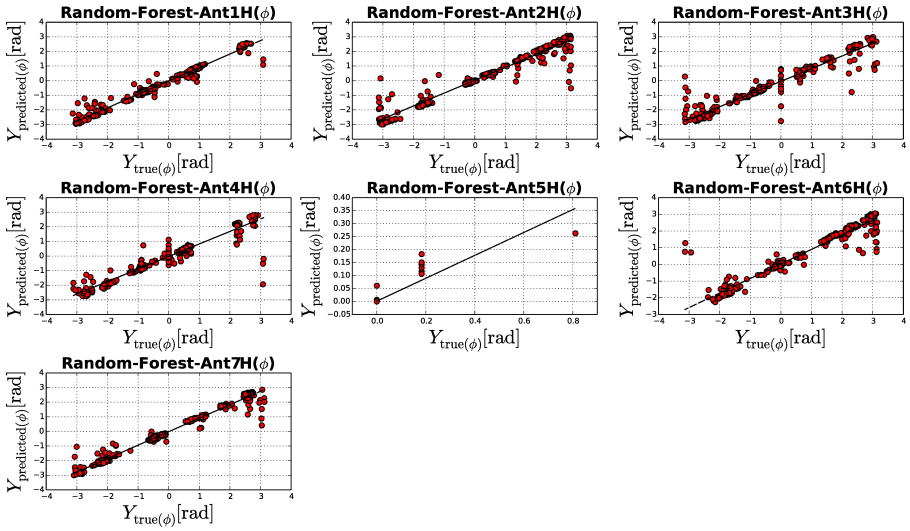
(b) Amplitude gain solutions for V-polarization

Figure 3.16: Results obtained from the decision tree learning algorithm with randomized search optimization algorithm. (a) and (b) are the predicted gain solutions $\mathbf{Y}_{\text{predicted}}$ by the decision tree vs the true gain solutions $\mathbf{Y}_{\text{true}}$ (CASA) for V-polarization. Similar to H-polarization above, the decision tree algorithm is failing to predict V-polarization amplitude and phase gain solutions.

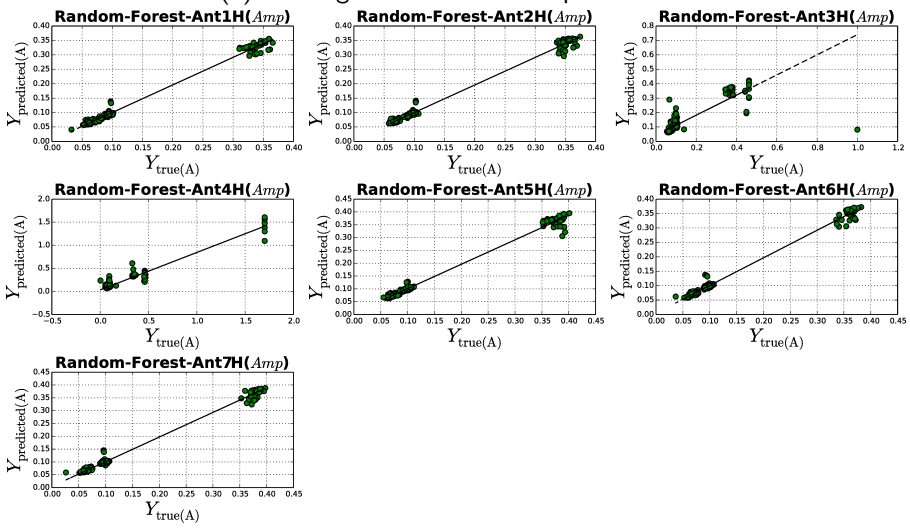
Antenna	**Decision tree phase**			
	Rmse	Rmae	R2score	Explained σ^2
Uniform average-H	0.573	0.417	0.820	0.821
Uniform average-V	0.435	0.367	0.879	0. 879
	Decision tree amplitude			
Uniform average-H	0.043	0.109	0.835	0.835
Uniform average-V	0.043	0.109	0.831	0.831

Table 3.7: The table shows the performance of the decision tree algorithm in predicting the amplitude and phase gain solutions for both H and V polarizations as shown in Figures 3.15 and 3.16. The values shown represent uniform average of all KAT-7 antennas, i.e., all output measures are averaged with uniform weight. Most of the predictions scatter away from the ideal truth values.

3.4.3 Random forest training

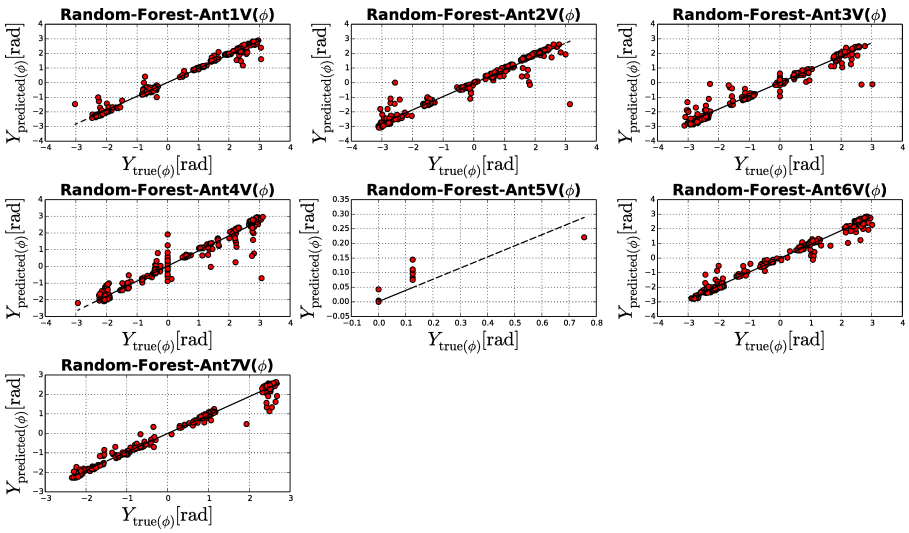


(a) Phase gain solutions for H-polarization

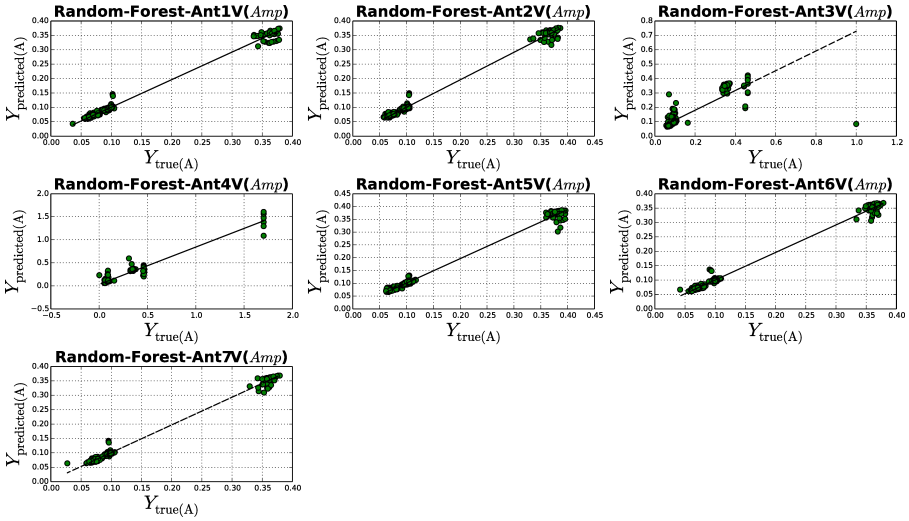


(b) Amplitude gain solutions for H-polarization

Figure 3.17: Results obtained from the random forest learning algorithm with randomized search optimization algorithm. Figures (a) and (b) are the predicted gain solutions $\mathbf{Y}_{\text{predicted}}$ by the random forest vs the true gain solutions $\mathbf{Y}_{\text{true}}$ (CASA) for H-polarization. We observe that the random forest algorithm is performing better in predicting the H-polarization amplitude and phase gain solutions with few outlier points.



(a) Phase gain solutions for V-polarization



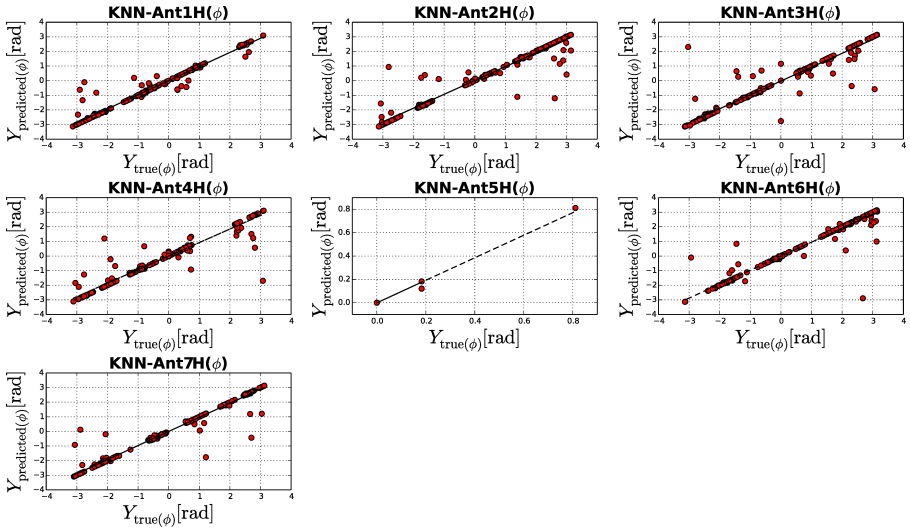
(b) Amplitude gain solutions for V-polarization

Figure 3.18: Results obtained from the random forest learning algorithm with randomized search optimization algorithm. (a) and (b) are the predicted phase gain solutions $\mathbf{Y}_{\text{predicted}}$ by the random forest vs the true gain solutions $\mathbf{Y}_{\text{true}}$ (CASA) for V polarization. We observe that the random forest is performing better in predicting the V-polarization amplitude and phase gain solutions with few outlier points.

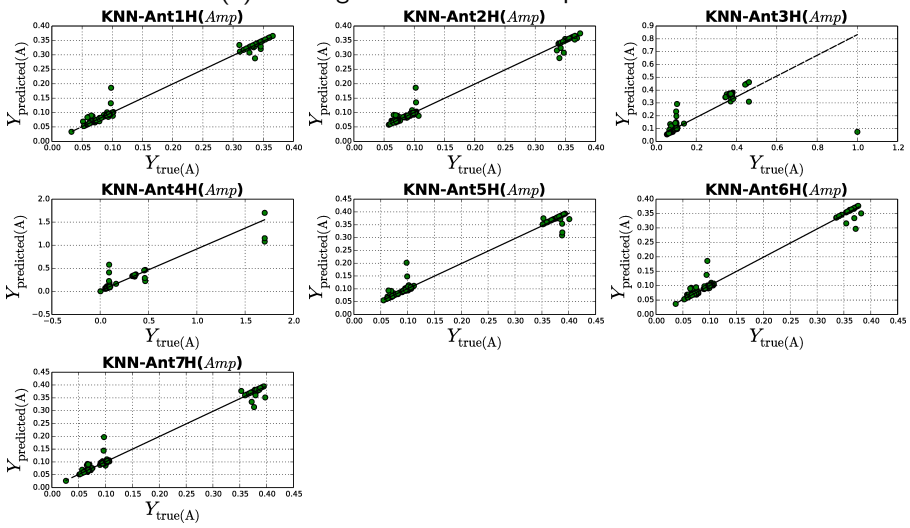
Antenna	**Random forest phase**			
	Rmse	Rmae	R2score	Explained σ^2
Uniform average-H	0.368	0.352	0.912	0.912
Uniform average-V	0.256	0.311	0.939	0.939
	Random forest amplitude			
Uniform average-H	0.028	0.087	0.918	0.919
Uniform average-V	0.028	0.088	0.916	0.917

Table 3.8: The table shows the performance of the random forest algorithm in predicting the amplitude and phase gain solutions for both H and V polarizations as shown in Figures 3.17 and 3.18. The values shown represent the uniform average of all KAT-7 antennas, i.e., all output measures are averaged with uniform weight. We observe that most of the predictions stay near the ideal truth values with rmse 2.5.4 and rmae 2.5.5 $\approx <$ 0.5, R^2 2.5.1 and explained variance V 2.5.3 converging to 1.

3.4.4 K-nearest neighbor training

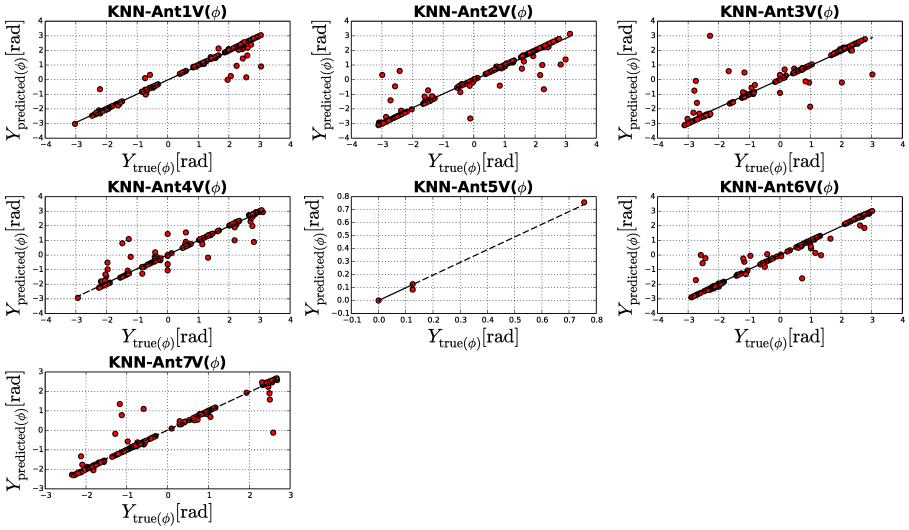


(a) Phase gain solutions for H-polarization

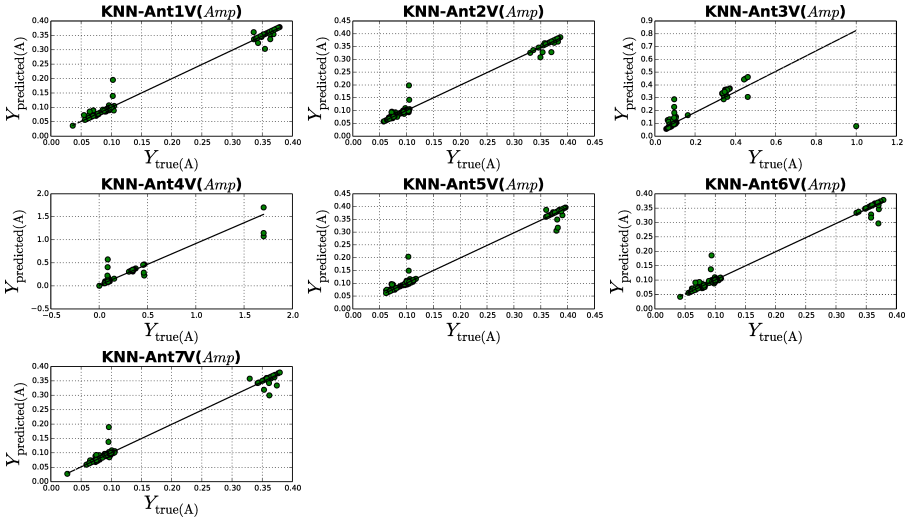


(b) Amplitude gain solutions for H-polarization

Figure 3.19: Results obtained from the K-nearest neighbor learning algorithm with randomized search optimization algorithm. (a) and (b) are the predicted gain solutions $\mathbf{Y}_{\text{predicted}}$ by the K-nearest neighbor vs the true gain solutions $\mathbf{Y}_{\text{true}}$ (CASA) for H-polarization. We observe that the K-nearest neighbor is performing better in predicting the H-polarization amplitude and phase gain solutions with few outlier points.



(a) Phase gain solutions for V-polarization



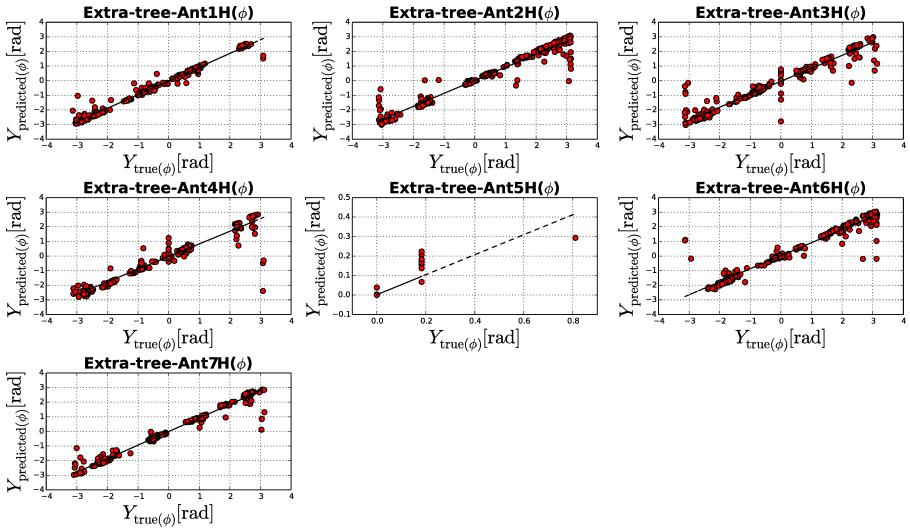
(b) Amplitude gain solutions for V-polarization

Figure 3.20: Results obtained from the K-nearest neighbor learning algorithm with randomized search optimization algorithm. (a) and (b) are the predicted phase gain solutions $\mathbf{Y}_{\text{predicted}}$ by the K-nearest neighbor vs the true phase gain solutions $\mathbf{Y}_{\text{true}}$ (CASA) for V-polarization. We observe that the K-nearest neighbor is performing better in predicting the V-polarization amplitude and phase gain solutions with few outlier points.

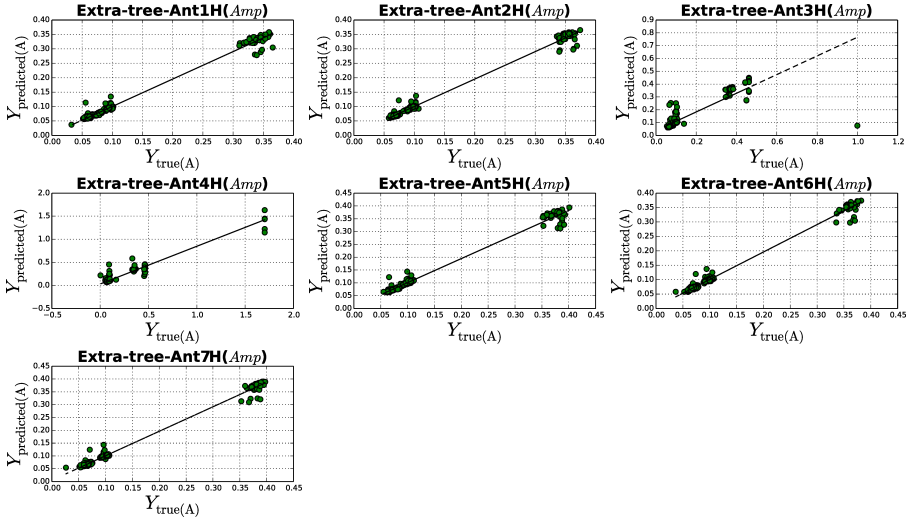
Antenna	**KNN phase**			
	Rmse	Rmae	R2score	Explainedσ^2
Uniform average-H	0.352	0.258	0.945	0.945
Uniform average-V	0.291	0.239	0.961	0. 961
	KNN amplitude			
Uniform average-H	0.352	0.258	0.945	0.945
Uniform average-V	0.291	0.239	0.961	0. 961

Table 3.9: The table shows the performance of the K-nearest neighbor algorithm in predicting the amplitude and phase gain solutions for both H and V polarizations as shown in Figure 3.19 and 3.20. The values shown represent the uniform average of all KAT-7 antennas, i.e., all output measures are averaged with uniform weight. Most of the predictions stay near the ideal truth values with RMSE 2.5.4 and rmae 2.5.5 $\approx <$ 0.5, R^2 2.5.1 and explained variance V 2.5.3 are converging to 1.

3.4.5 Extremely randomized tree training

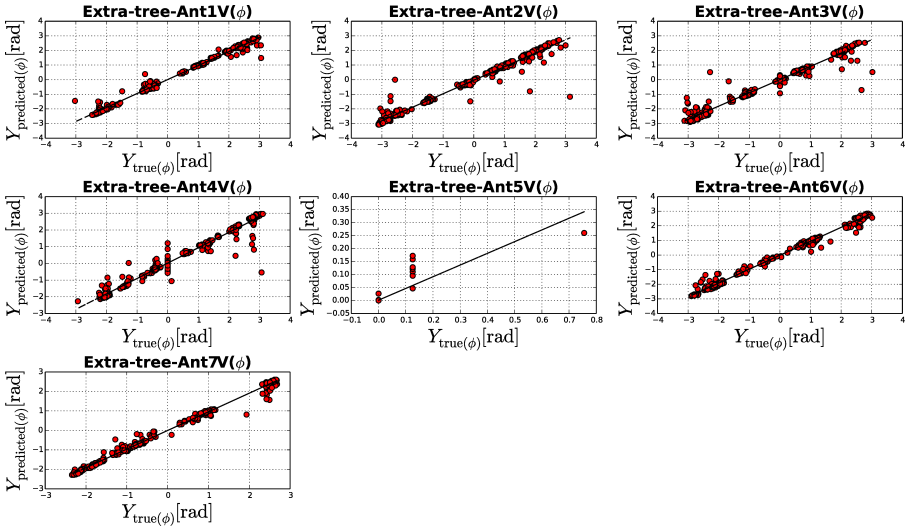


(a) Phase gain solutions for H-polarization

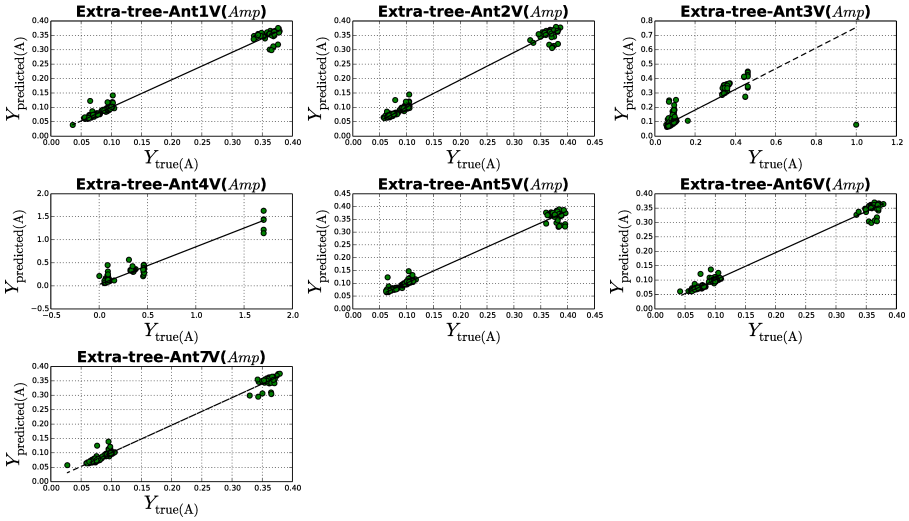


(b) Amplitude gain solutions for H-polarization

Figure 3.21: Results obtained from the extremely randomized tree learning algorithm with randomized search optimization algorithm. (a) and (b) are the predicted gain solutions $\mathbf{Y}_{\text{predicted}}$ by the extremely randomized tree vs the true gain solutions $\mathbf{Y}_{\text{true}}$ (CASA) for H-polarization. We observe that the extremely randomized tree is performing better in predicting the H-polarization amplitude and phase gain solutions with few outlier points.



(a) Phase gain solutions for V-polarization



(b) Amplitude gain solutions for V-polarization

Figure 3.22: Results obtained from the extremely randomized tree learning algorithm with randomized search optimization algorithm. (a) and (b) are the predicted gain solutions $\mathbf{Y}_{\text{predicted}}$ by the learning algorithm vs the true gain solutions $\mathbf{Y}_{\text{true}}$ (CASA) for V-polarization. We observe that the extremely randomized tree is performing better in predicting the V-polarization amplitude and phase gain solutions with few outlier points.

Antenna	Extremely randomized phase			
	rmse	Rmae	R2score	Explained σ^2
Uniform average-H	0.337	0.336	0.936	0.936
Uniform average-V	0.235	0.301	0.956	0. 957
	Extremely randomized tree amplitude			
Uniform average-H	0.027	0.083	0.925	0.925
Uniform average-V	0.027	0.083	0.923	0.923

Table 3.10: The table shows the performance of the extremely randomized tree algorithm in predicting the amplitude and phase gain solutions for both H and V polarizations as shown in figures 3.21 and 3.22. The values shown represents the uniform average of all KAT-7 antennas, i.e., all output measures are averaged with uniform weight. Most of the predictions stay near the ideal truth values with rmse 2.5.4 and rmae 2.5.5 $\approx<$ 0.5, R^2 2.5.1 and explained variance V 2.5.3 are converging to 1.

3.4.6 Summary of results

This section report the accuracy measure obtained for each learning algorithm when evaluating on testing dataset. In every model there are four plots, each for gain amplitude and gain phase H $\&$ V-polarization. The y axis in each subplot is the predicted gain amplitude and gain phase solutions using the testing sensor data, and the x axis is the ground truth gain amplitude and gain phase solution. Note that the two-cluster grouping of points in the predicted vs ground truth amplitude plots in Figures 3.16b, 3.17b, 3.18b, 3.19b, 3.20b, 3.21b, 3.22b are due to the sinusoidal variation of the gain amplitude solutions as shown in Figure 3.3. We tested our models using using the appropriate regression evaluation methods: RMSE, RMAE, R^2 score and the explained variance. The evaluation results obtained from the extremely randomized trees, random forest, and K-nearest neighbor are accurate with H-polarization RMSE of $\approx 0.3\ \mathrm{rad}$ and V-polarization of $\approx 0.2\ \mathrm{rad}$ for phase gain solution prediction as shown in Figures 3.17a, 3.18a, 3.19a, 3.20a, 3.21a, 3.22a, 3.22a. We observe that the amplitude gain solution prediction has better accuracy for random forest and extremely randomized tree on H $\&$ V polarization with RMSE error being ≈ 0.02, much less than the phase gain solution prediction, as shown in Figures 3.17b, 3.18b, 3.21b, 3.22b. We further observe that most of the predictions stay near the ground truth values with a coefficient of determination R^2 score converging to 1 for Random forest, extremely randomized trees and K-nearest neighbors with few outlying points that can be due to errors in calibration solution generation and radio frequency interference. On the other we observe that the decision tree algorithm is performing poorly in predicting the amplitude and phase gain solutions for H and V polarization. This is likely due to overfitting of the model.

3.4.7 Validating on new dataset

Once one is satisfied with the performance of the learning algorithms in predicting the calibration solutions accurately from the testing dataset, in this section new sensor data extracted from new (unseen) KAT-7 observations with the phase calibrator source PKS1623-586. This process is called model validation. Since the model was trained on normalized data, one first normalizes the new sensor data with mean μ and standard deviation σ obtained from the normalization of the training dataset.

$$\text{scaled-sensor data} = \frac{\text{sensor data} - \mu}{\sigma} \tag{3.4.1}$$

We make use of a three-observation dataset to perform the unbiased evaluation of the final model fit. These are a subset of the Circinus X-1 KAT-7 commissioning observations taken from December 2011

- June 2012. In each validation dataset, we compare the performance of decision tree, random forest, K-nearest neighbor and extremely randomized trees. we compare these results with gain amplitude and phase calibration solutions obtained from CASA 1GC for H & V-polarization.

3.4.7.1 Observation test-1

This section gives the plot results obtained from the validation observation. Each plot shows the gain calibration solutions predicted by the models: decision tree, random forest, K-nearest neghbors and extremely randomised trees using the new observation's sensor data. We compare the predicted gain calibration solutions with 1GC CASA generated solutions to validate each model's performance. The outliers at ≈ 0.45 on antenna 3 and antenna 4 for gain amplitude solutions are due to 1GC CASA calibration errors.

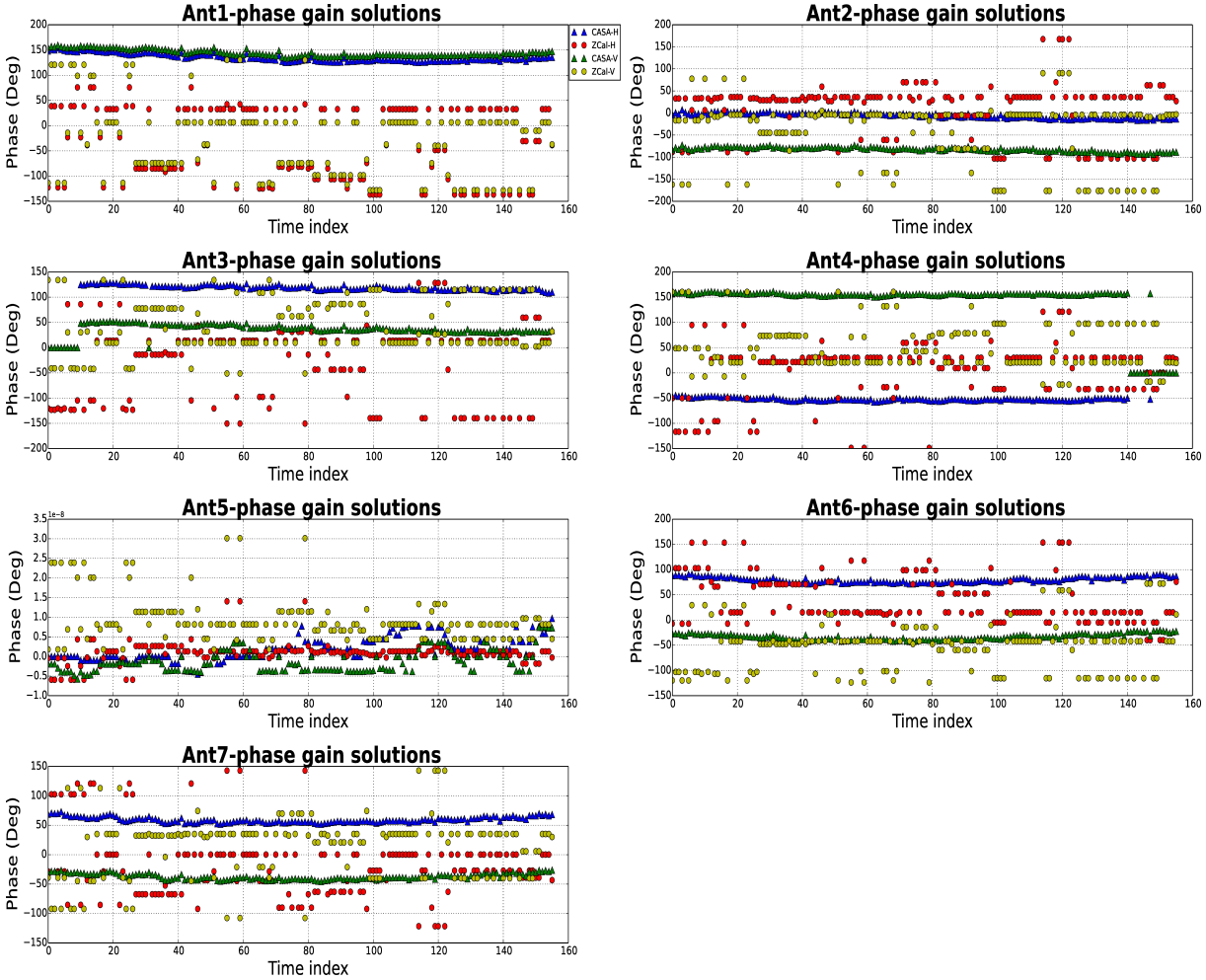


Figure 3.23: This figure shows the response of the decision tree learning algorithm in predicting the H & V phase gain solutions for the calibrator PKS1613-586 from the validation dataset-1. In each subplot we compare the predicted phase gain solutions (where H polarization is represented by red and V-polarization is represented by yellow) with the CASA generated phase gain solutions (where H polarization is represented by blue and V-polarization is represented by green). The decision tree algorithm is failing to predict the gain phase solutions on the new observation dataset-1.

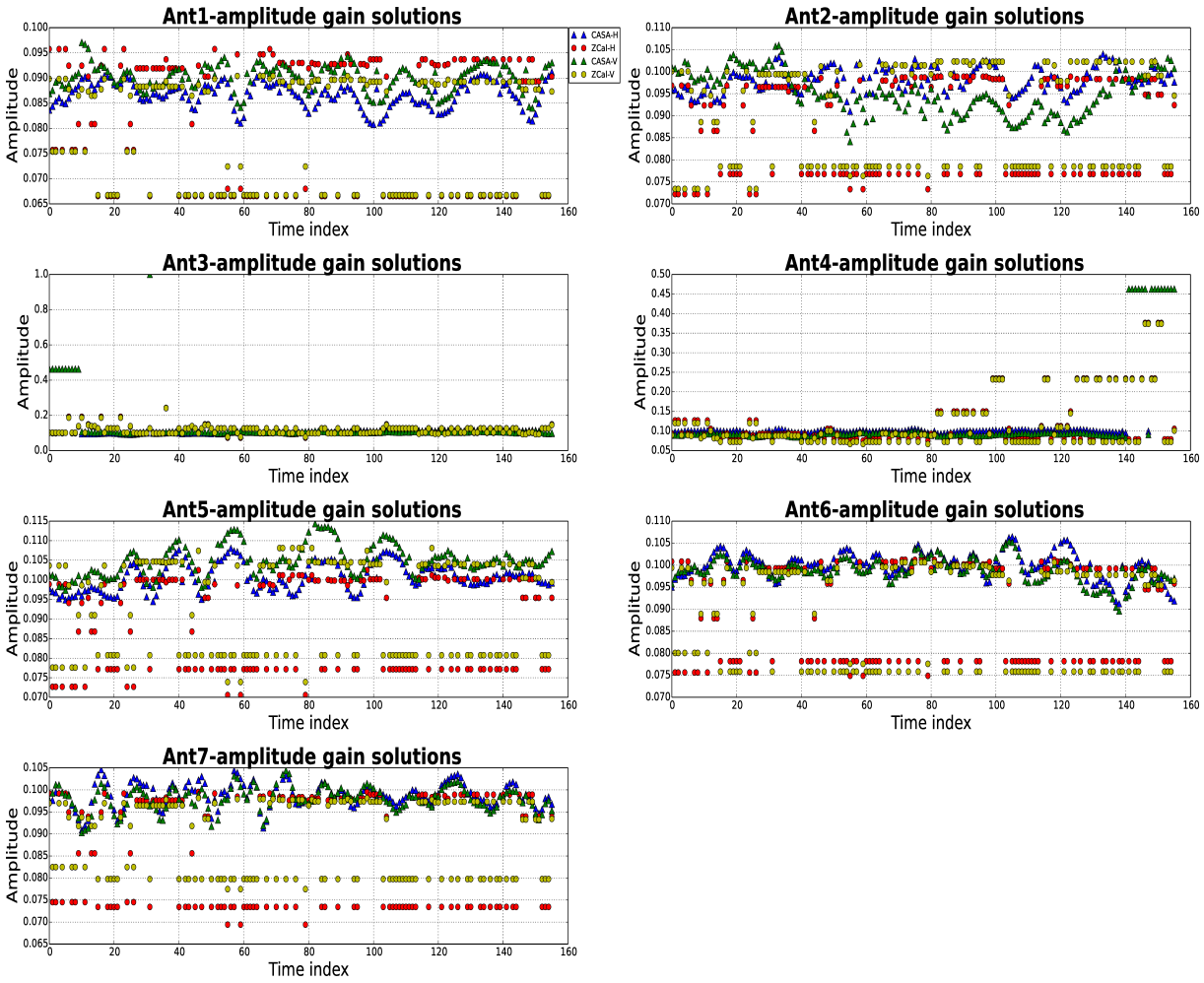


Figure 3.24: This figure shows the response of the decision tree learning algorithm in predicting the amplitude gain solutions for the calibrator PKS1613-586 from the validation dataset-1. In each subplot we compare the the predicted amplitude gain solutions (where H polarization is represented by red and V-polarization is represented by yellow) with the CASA generated phase gain solutions (where H polarization is represented by blue and V-polarization is represented by green). The decision tree algorithm is failing to predict the gain amplitude solutions on the new observation dataset-1.

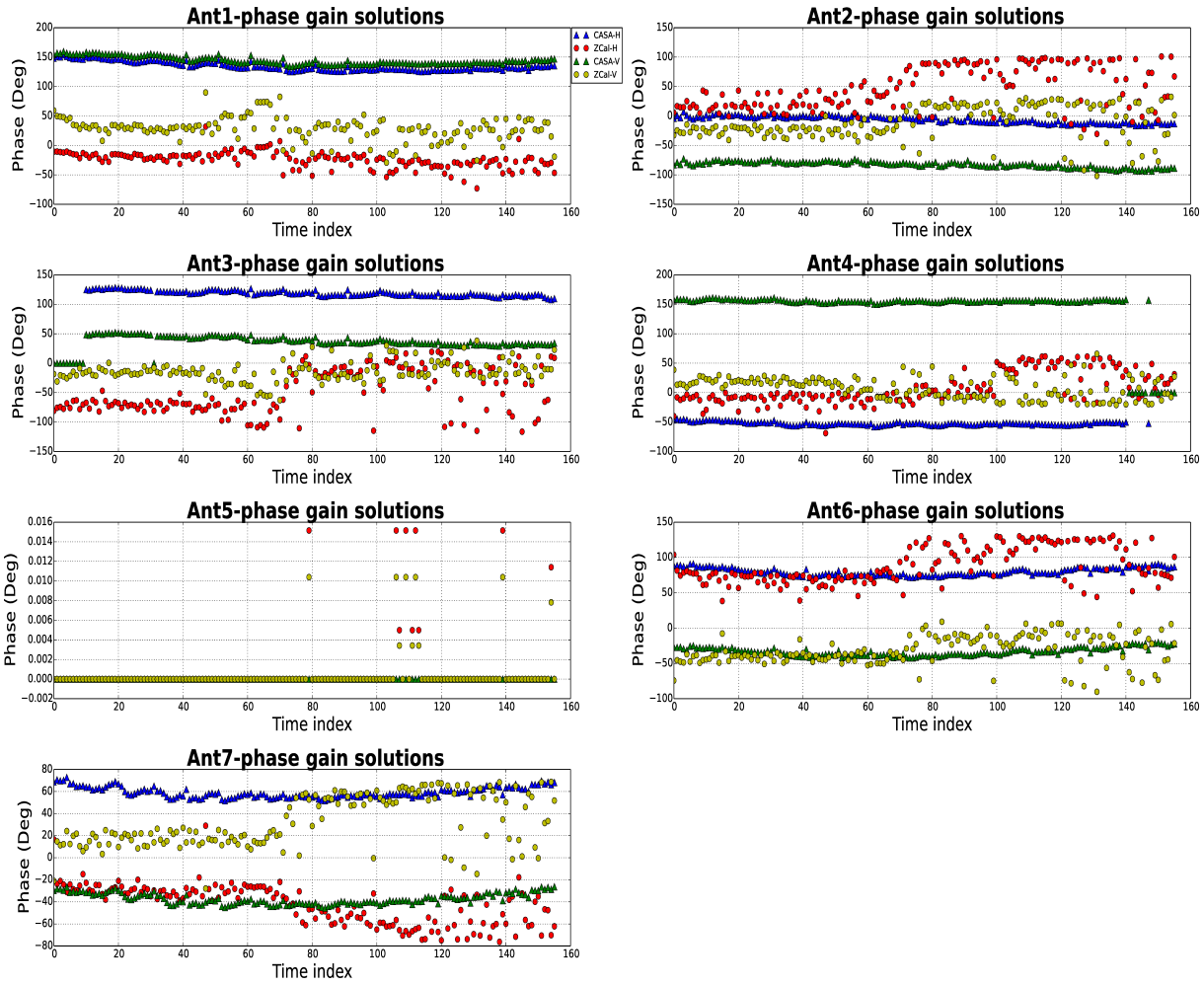


Figure 3.25: This figure shows the response of the random forest learning algorithm in predicting the phase gain solutions for the calibrator PKS1613-586 from the validation dataset-1. In each subplot we compare the predicted phase gain solutions (where H polarization is represented by red and V-polarization is represented by yellow) with the CASA generated phase gain solutions (where H polarization is represented by blue and V-polarization is represented by green). The random forest algorithm is failing to predict the gain phase solutions for some antennas and performing well in others with few outliers.

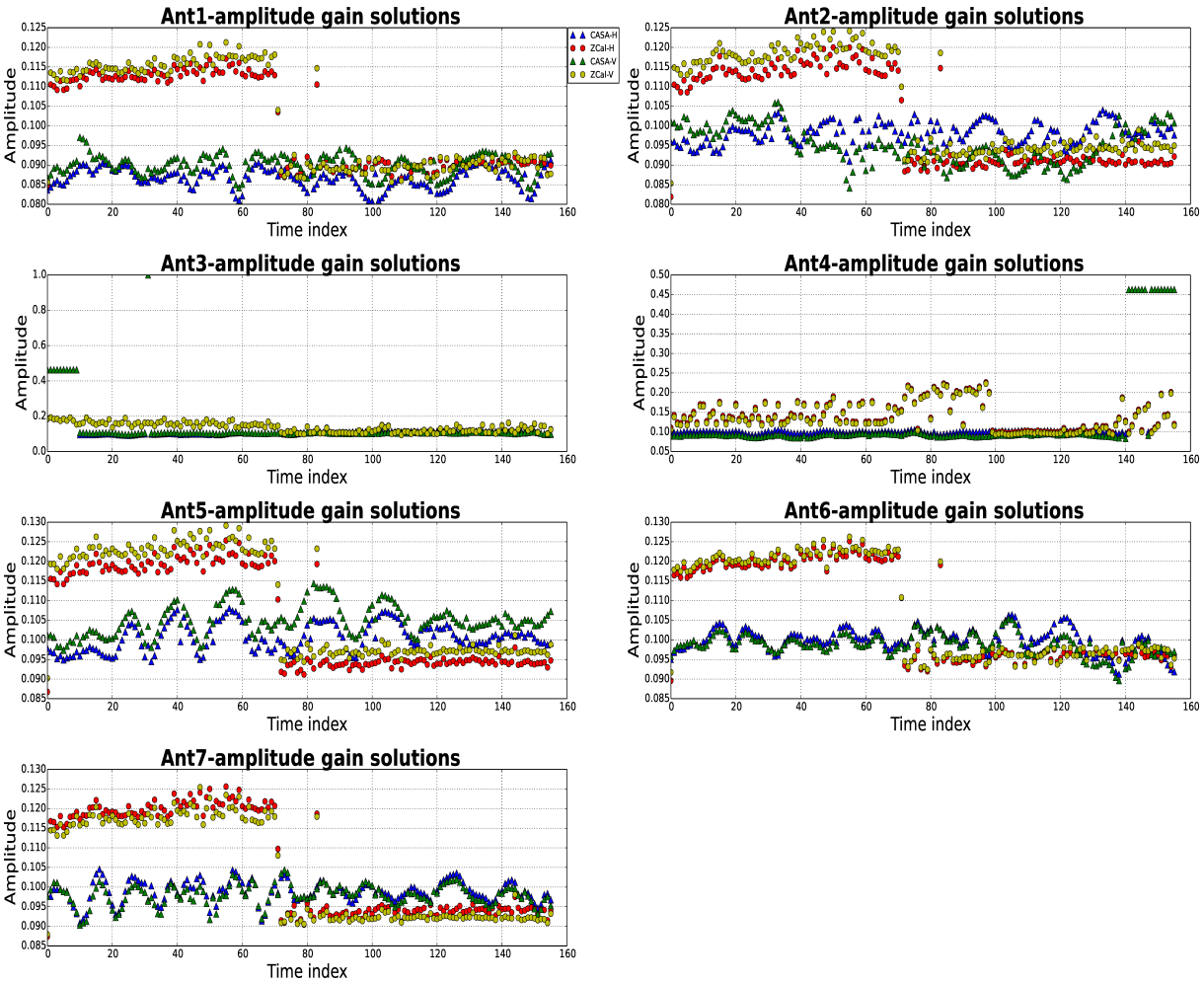


Figure 3.26: This figure shows the response of the random forest learning algorithm in predicting the amplitude gain solutions for the calibrator PKS1613-586 from the validation dataset-1. In each subplot we compare the predicted amplitude gain solutions (where H polarization is represented by red and V-polarization is represented by yellow) with the CASA generated phase gain solutions (where H polarization is represented by blue and V-polarization is represented by green). The random forest algorithm is failing to predict the gain amplitude solutions in all the antennas, for solutions at time index < 70. We further observe that at time index 80 the algorithm performing better in predicting the amplitude solutions, slightly close to CASA on the new observation dataset-1.

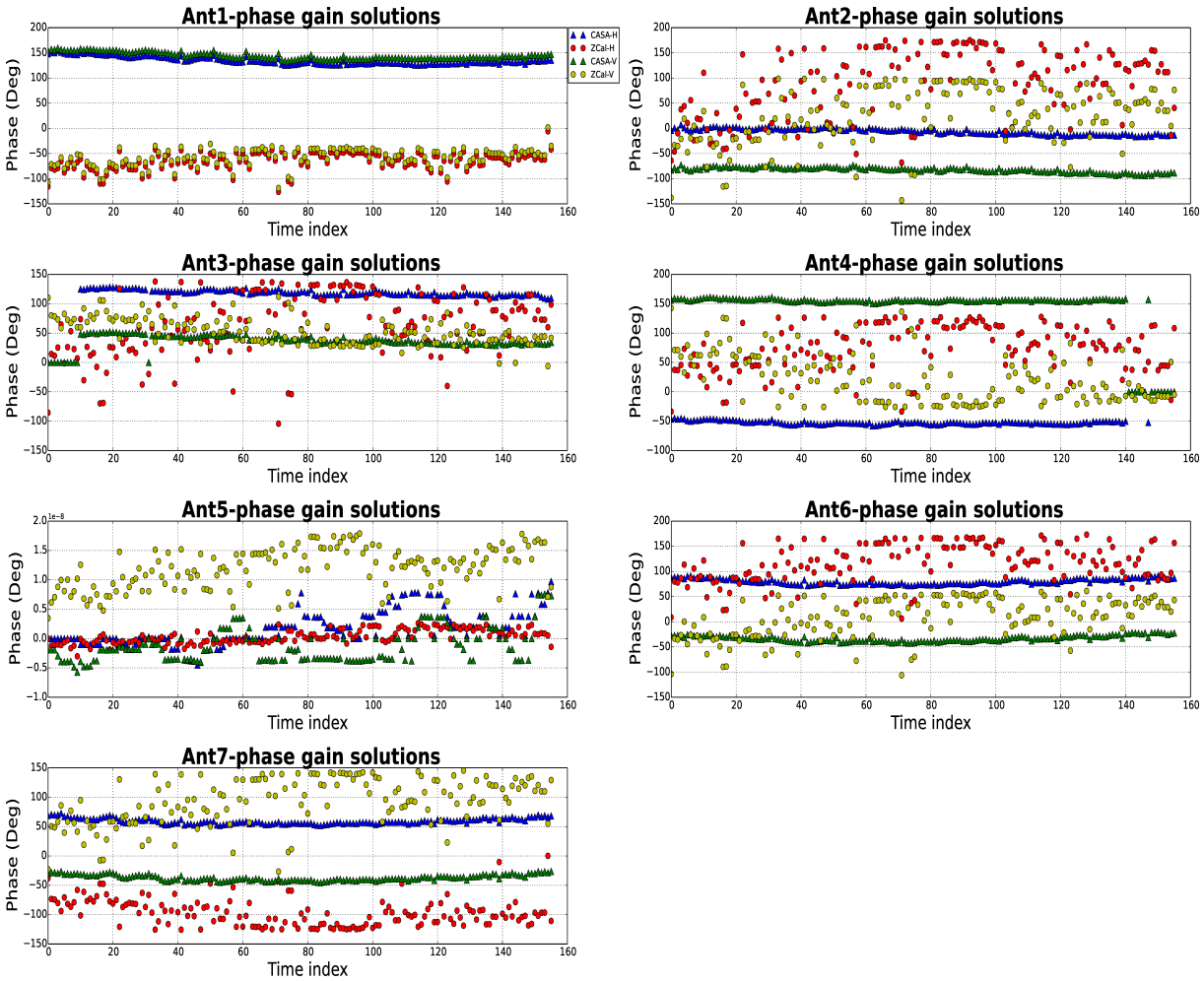


Figure 3.27: This figure shows the response of the K-nearest neighbor learning algorithm in predicting the phase gain solutions for the calibrator PKS1613-586 from the validation dataset-1. In each subplot we compare the predicted phase gain solutions (where H polarization is represented by red and V-polarization is represented by yellow) with the CASA generated phase gain solutions (where H polarization is represented by blue and V-polarization is represented by green). The K-nearest neighbor algorithm is failing to predict the gain phase solutions on the new observation dataset-1.

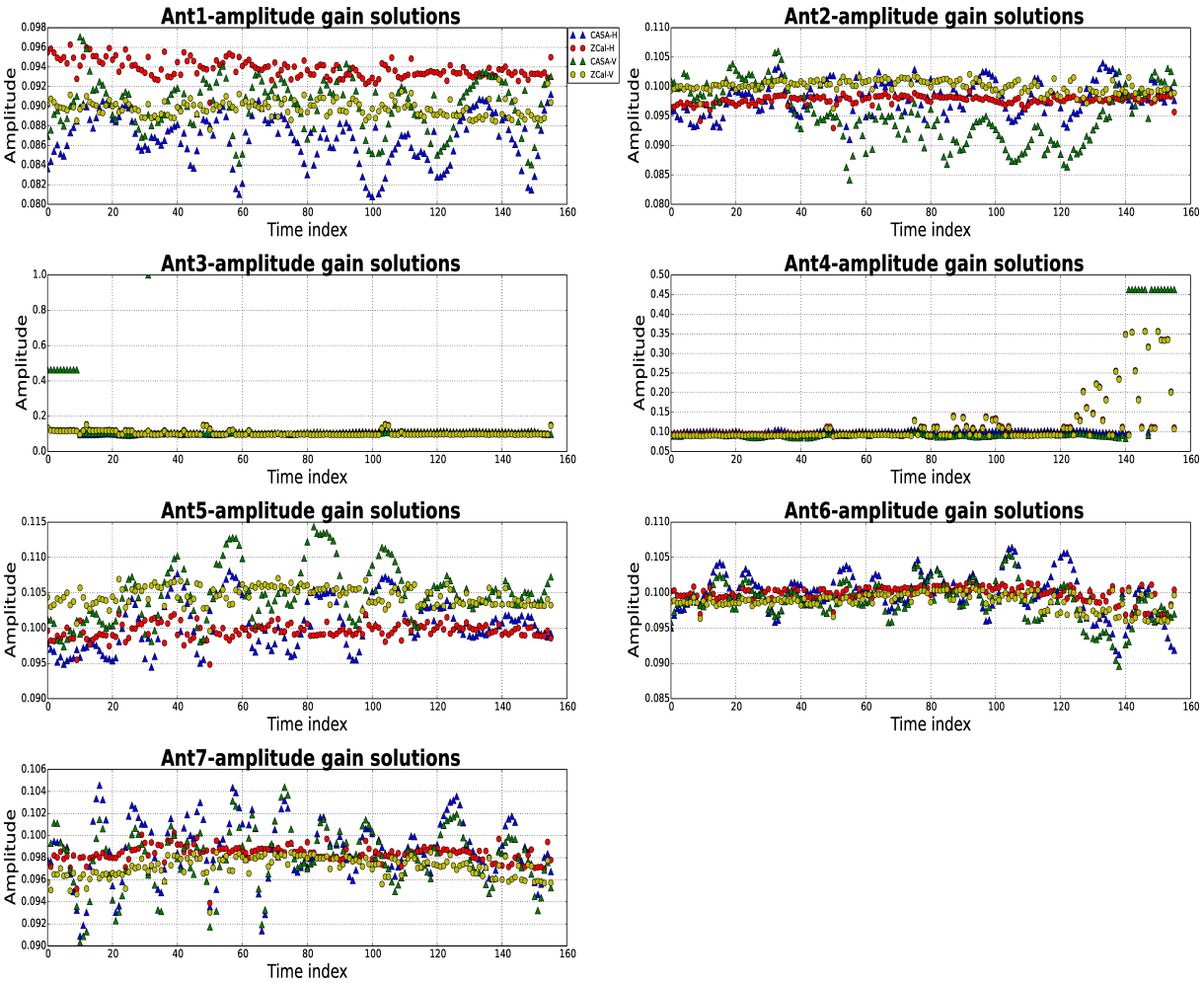


Figure 3.28: This figure shows the response of the K-nearest neighbor learning algorithm in predicting the amplitude gain solutions for the calibrator PKS1613-586 from the validation dataset-1. In each subplot we compare the predicted amplitude gain solutions (where H polarization is represented by red and V-polarization is represented by yellow) with the CASA generated phase gain solutions (where H polarization is represented by blue and V-polarization is represented by green). The K-nearest neighbor algorithm is predicting the gain amplitude solutions in all the antennas with a weak match when compared to CASA on the new observation dataset-1.

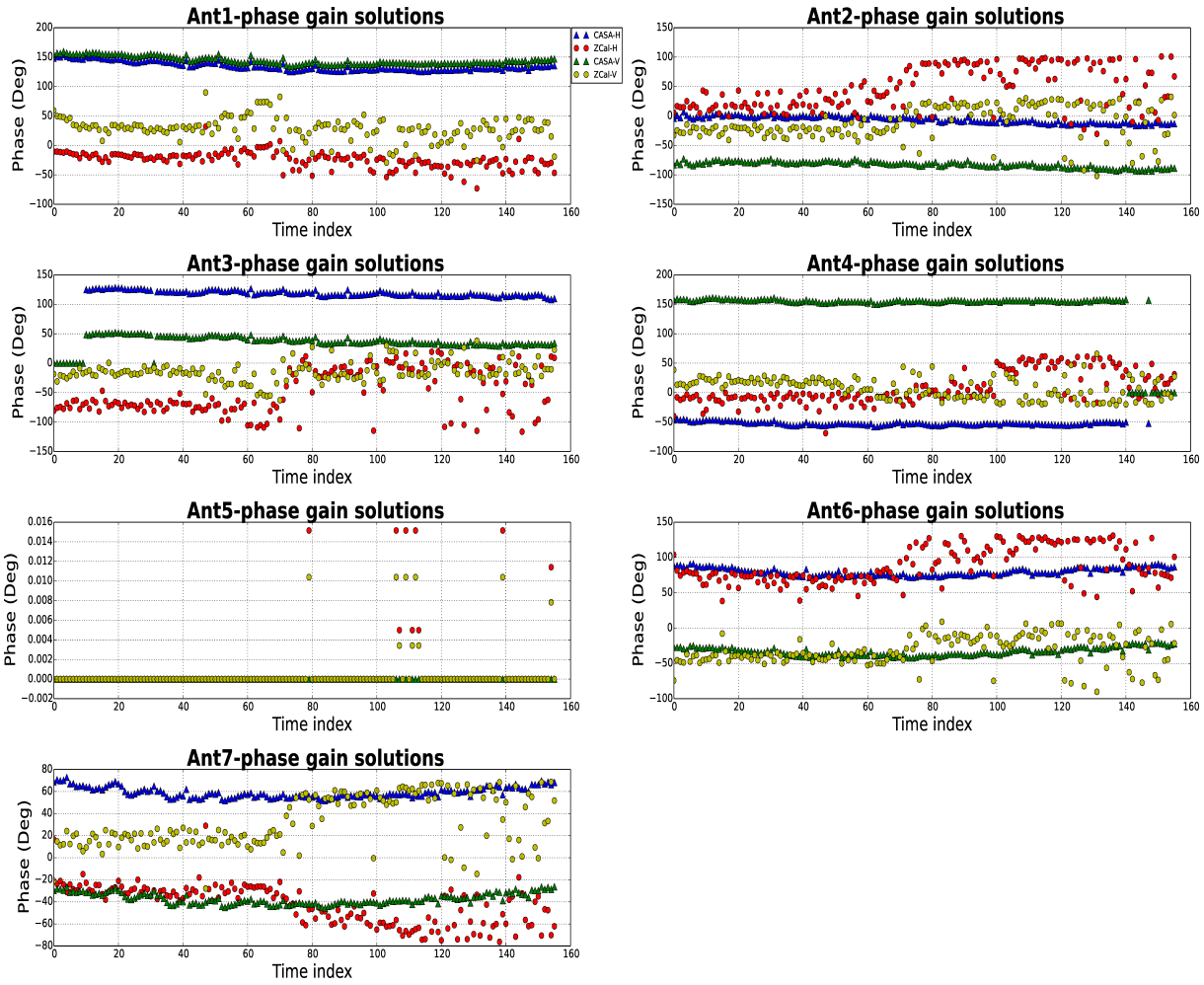


Figure 3.29: This figure shows the response of the extremely randomized tree learning algorithm in predicting the phase gain solutions for the calibrator PKS1613-586 from the validation dataset-1. In each subplot we compare the predicted phase gain solutions (where H polarization is represented by red and V-polarization is represented by yellow) with the CASA generated phase gain solutions (where H polarization is represented by blue and V-polarization is represented by green). The extremely randomized tree algorithm is failing to predict the gain phase solutions for some antennas and performing well in others with few outliers.

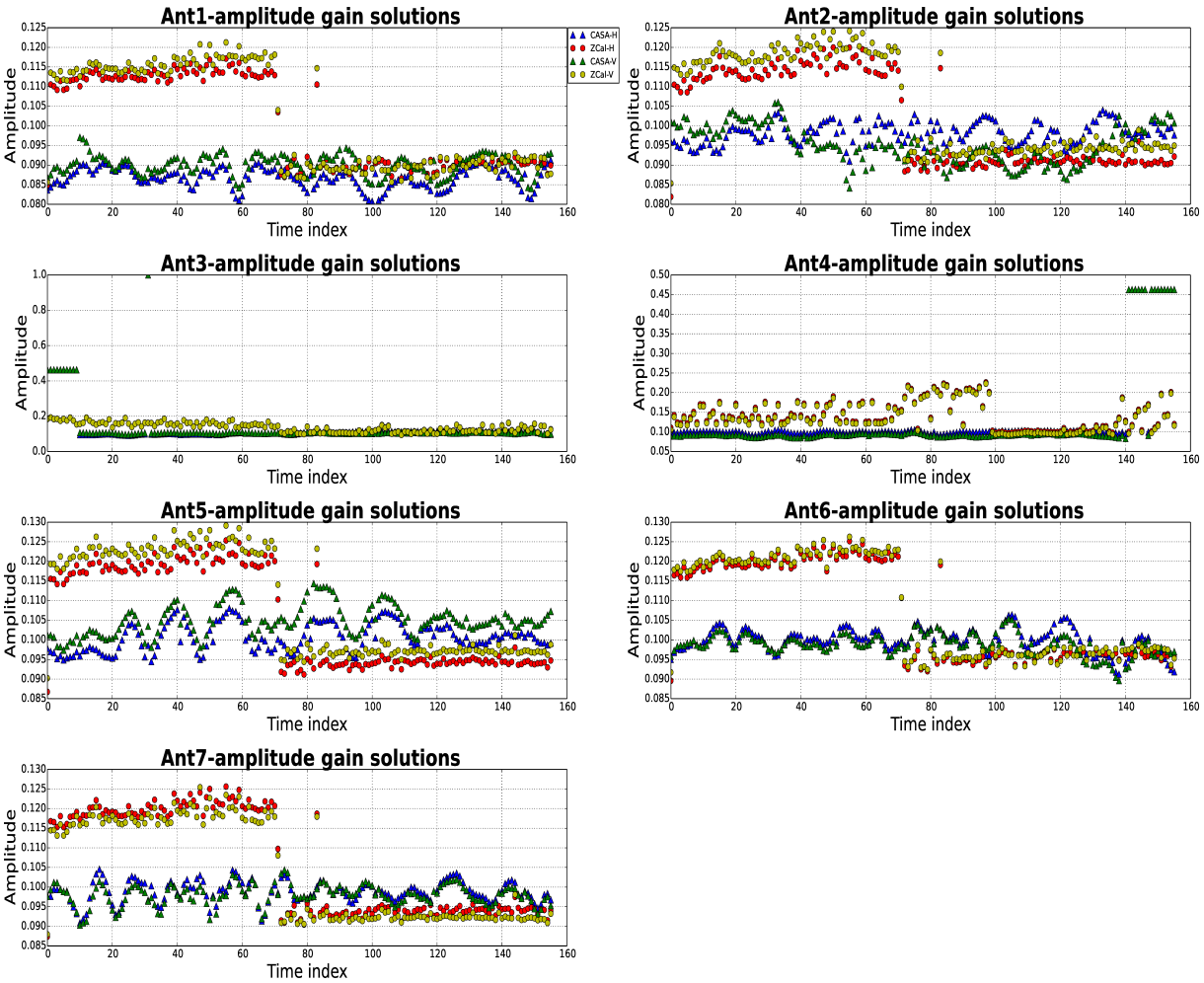


Figure 3.30: This figure shows the response of the extremely randomized tree learning algorithm in predicting the amplitude gain solutions for the calibrator PKS1613-586 from the validation dataset-1. In each subplot we compare the predicted amplitude gain solutions (where H polarization is represented by red and V-polarization is represented by yellow) with the CASA generated phase gain solutions (where H polarization is represented by blue and V-polarization is represented by green). The extremely randomized tree algorithm is failing to predict the gain amplitude solutions in all the antennas, for solutions at time index < 70. We further observe that at time index 80 the algorithm performing better in predicting the amplitude solutions, slightly close to CASA on the new observation dataset-1.

For dataset-1, one observes that the random forest, extremely randomized trees and the K-nearest neighbors have not fully learned to generalise in predicting the gain amplitude and phase calibration solutions when compared to CASA. We observe that these model's predictions fail in some part of the samples and successfully predict well in other. However, the extremely randomized trees algorithm is performing better than the rest as shown in Figure 3.29 and 3.30.

3.4.7.2 Observation test-2

In this section, the plot results obtained from the validation dataset-2 vs 1GC CASA solutions are given.

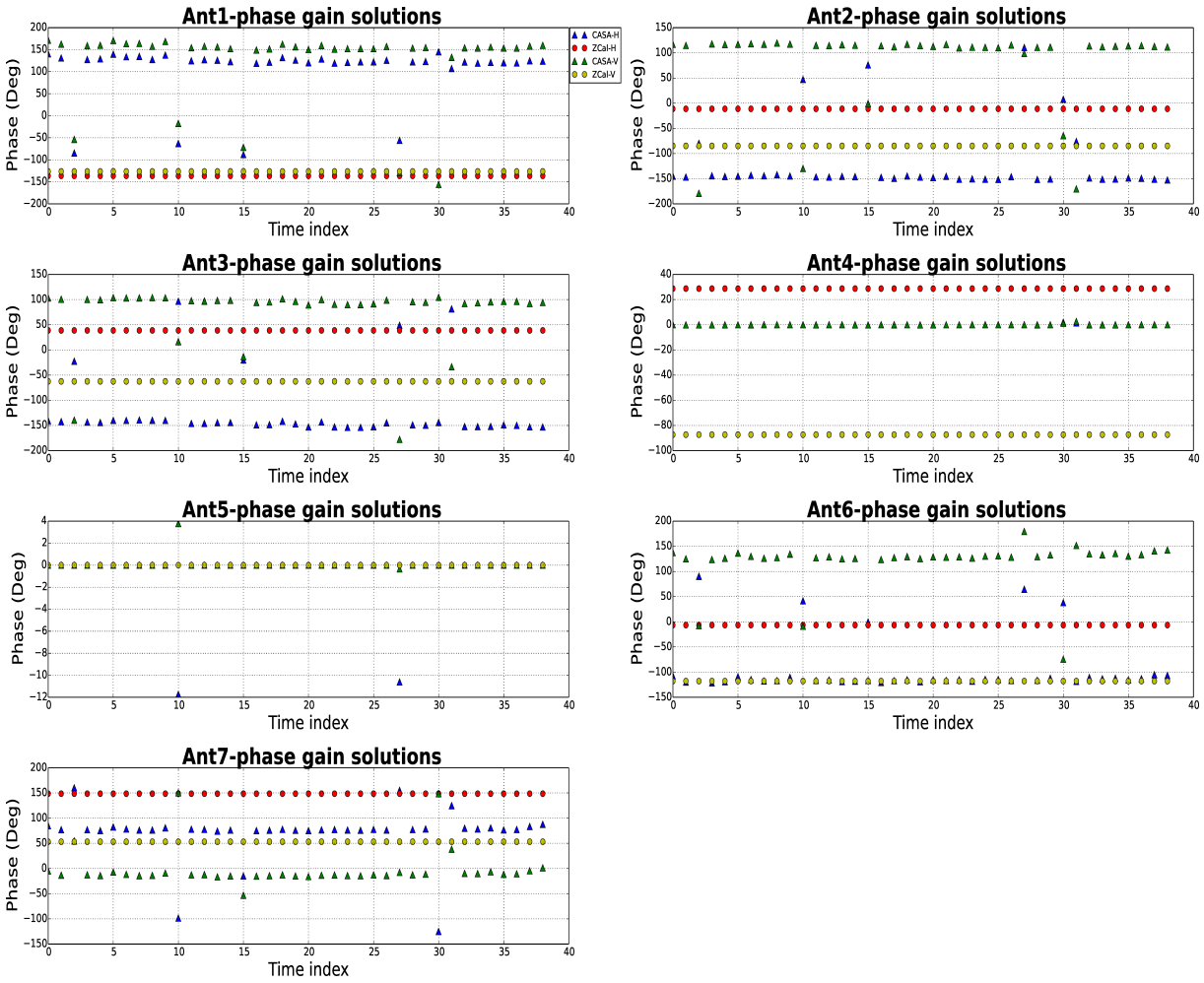


Figure 3.31: This figure shows the response of the decision tree learning algorithm in predicting the H & V phase gain solutions for the calibrator PKS1613-586 from the validation dataset-2. In each subplot we compare the predicted phase gain solutions (where H polarization is represented by red and V-polarization is represented by yellow) with the CASA generated phase gain solutions (where H polarization is represented by blue and V-polarization is represented by green). The decision tree algorithm is failing to predict the gain phase solutions for some antennas and performing well in others as seen for example in antenna 1 where CASA estimated solutions at 150° and the decision tree at -150° with no outliers. We also observe that it has simply learned that the reference antenna is flat.

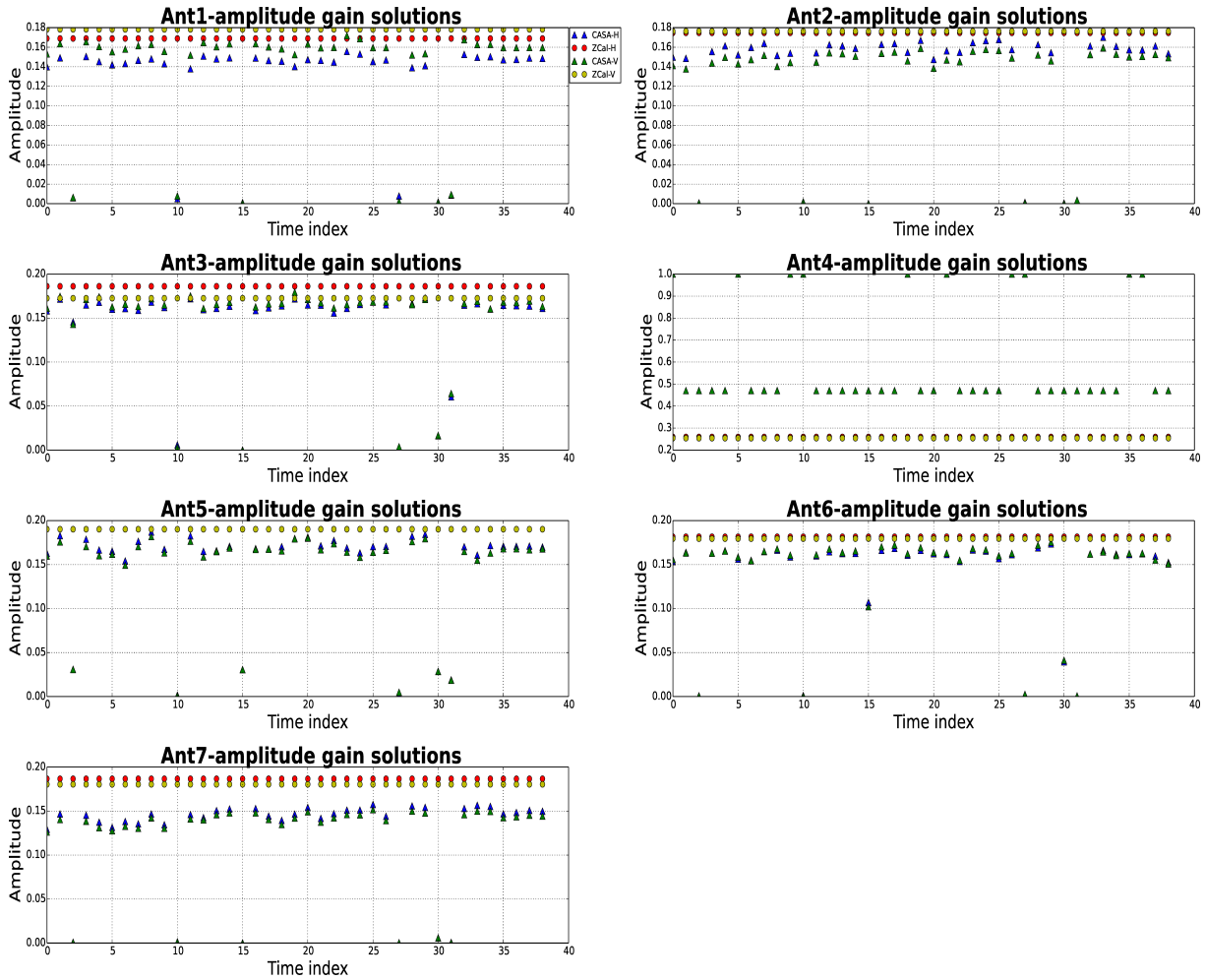


Figure 3.32: This figure shows the response of the decision tree learning algorithm in predicting the amplitude gain solutions for the calibrator PKS1613-586 from the validation dataset-2. In each subplot we compare the the predicted amplitude gain solutions (where H polarization is represented by red and V-polarization is represented by yellow) with the CASA generated phase gain solutions (where H polarization is represented by blue and V-polarization is represented by green). The decision tree algorithm is failing to predict the gain amplitude solutions on the new observation dataset-2. We also observe that it has not learned the sinusoidal behaviour of the amplitude solutions.

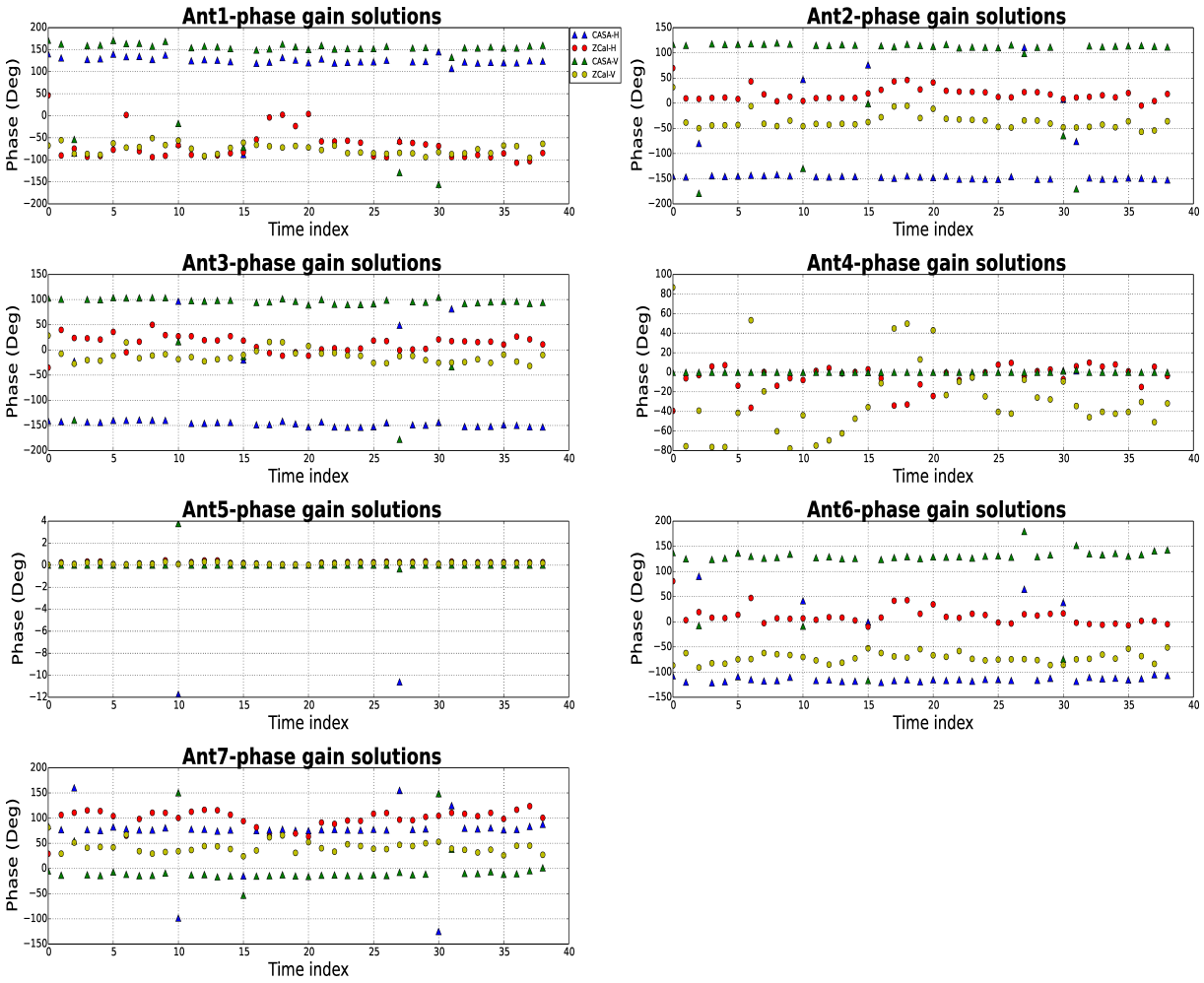


Figure 3.33: This figure shows the response of the random forest learning algorithm in predicting the H & V phase gain solutions for the calibrator PKS1613-586 from the validation dataset-2. In each subplot we compare the predicted phase gain solutions (where H polarization is represented by red and V-polarization is represented by yellow) with the CASA generated phase gain solutions (where H polarization is represented by blue and V-polarization is represented by green). The random forest algorithm is failing to predict the gain phase solutions for some antennas and slightly performing well in others as seen for example in antenna 1H where CASA estimated solutions at 150° and the random forest at -150°. We also observe that it has simply learned that the reference antenna is flat.

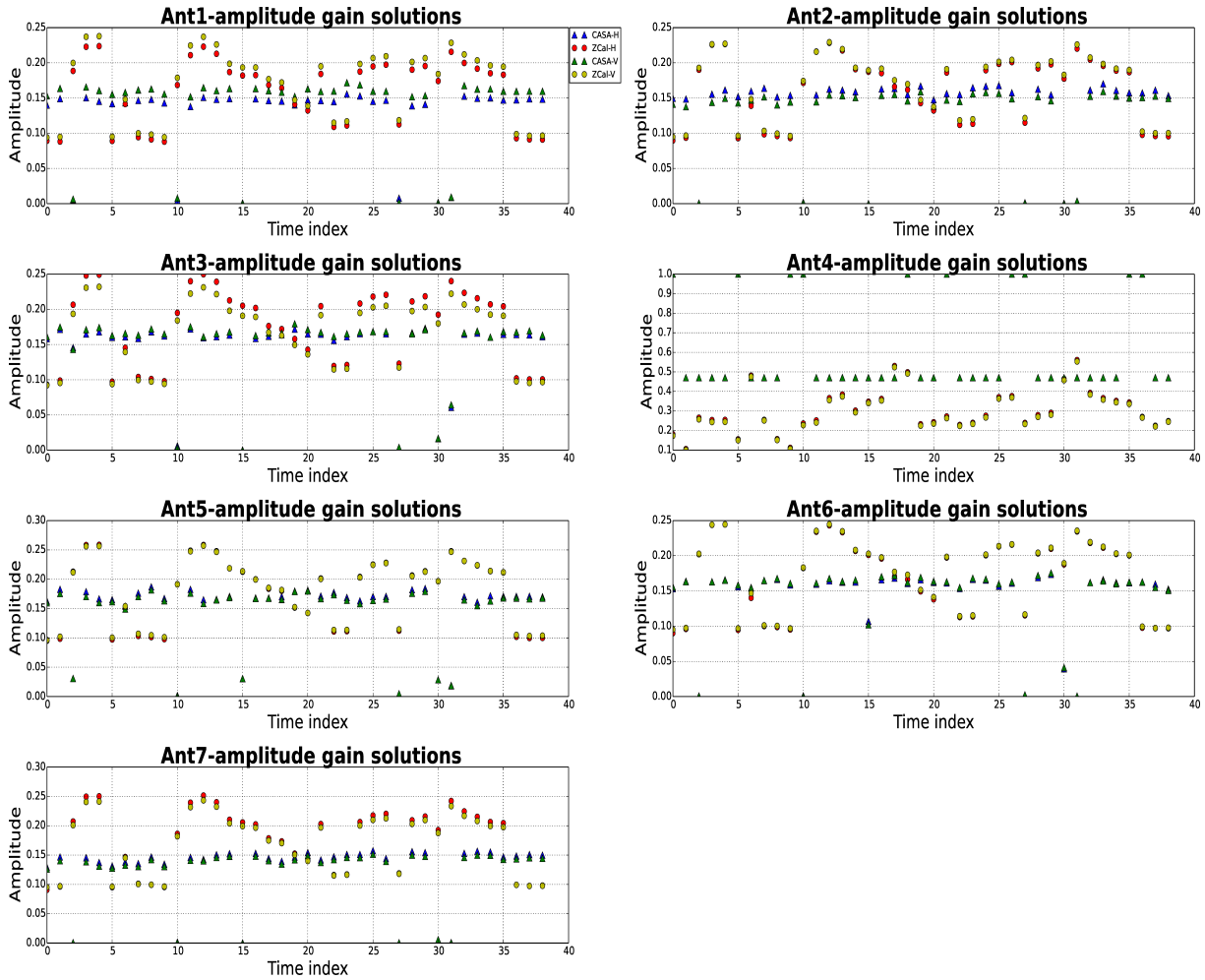


Figure 3.34: This figure shows the response of the random forest learning algorithm in predicting the amplitude gain solutions for the calibrator PKS1613-586 from the validation dataset-2. In each subplot we compare the the predicted amplitude gain solutions (where H polarization is represented by red and V-polarization is represented by yellow) with the CASA generated phase gain solutions (where H polarization is represented by blue and V-polarization is represented by green). The random forest algorithm is failing to predict the gain amplitude solutions on the new observation dataset-2.

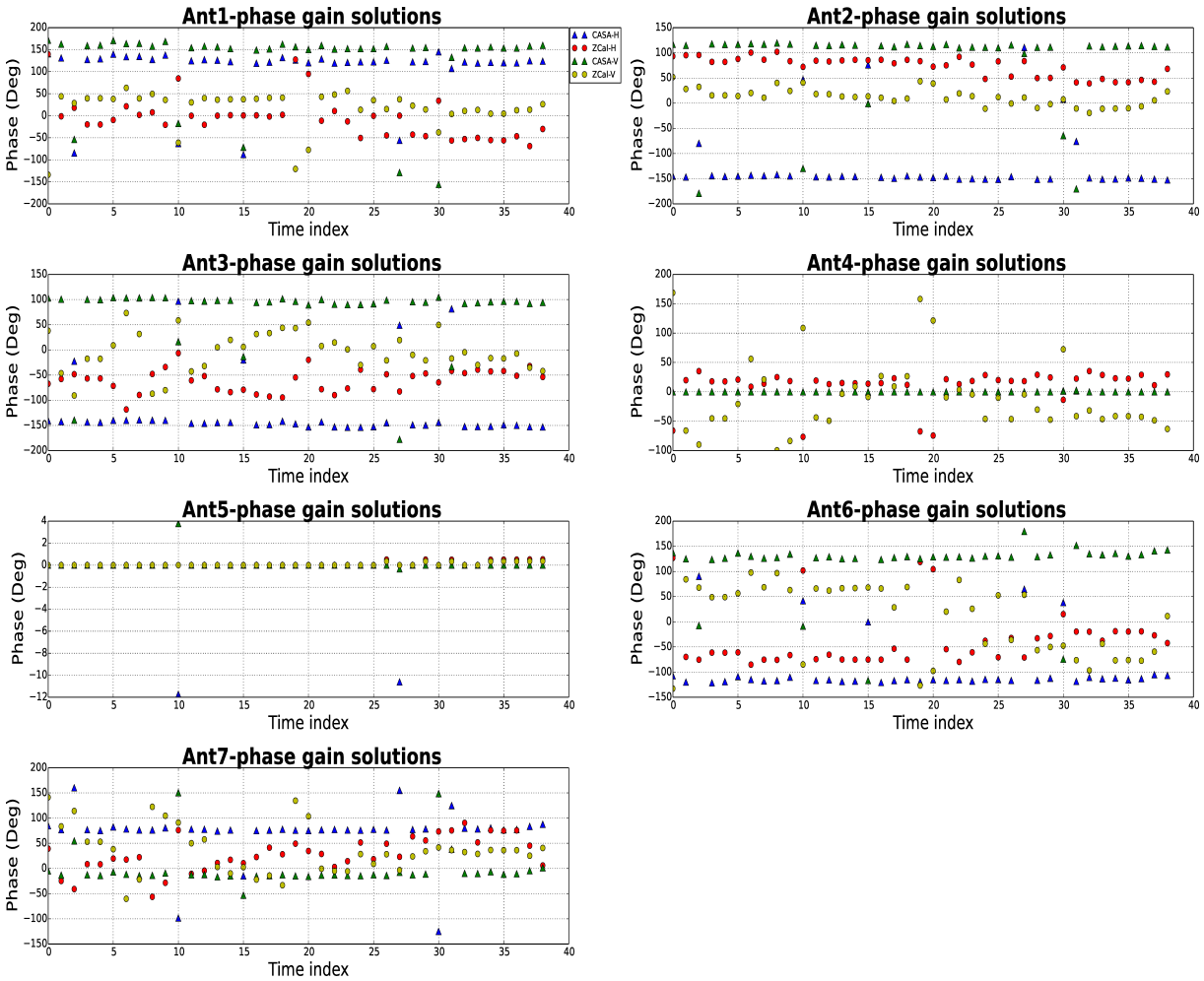


Figure 3.35: This figure shows the response of the K-nearest neighbor learning algorithm in predicting the H & V phase gain solutions for the calibrator PKS1613-586 from the validation dataset-2. In each subplot we compare the predicted phase gain solutions (where H polarization is represented by red and V-polarization is represented by yellow) with the CASA generated phase gain solutions (where H polarization is represented by blue and V-polarization is represented by green). The K-nearest neighbor algorithm is failing to predict the gain phase solutions, except for the reference antenna 5, where it learned that it is flat.

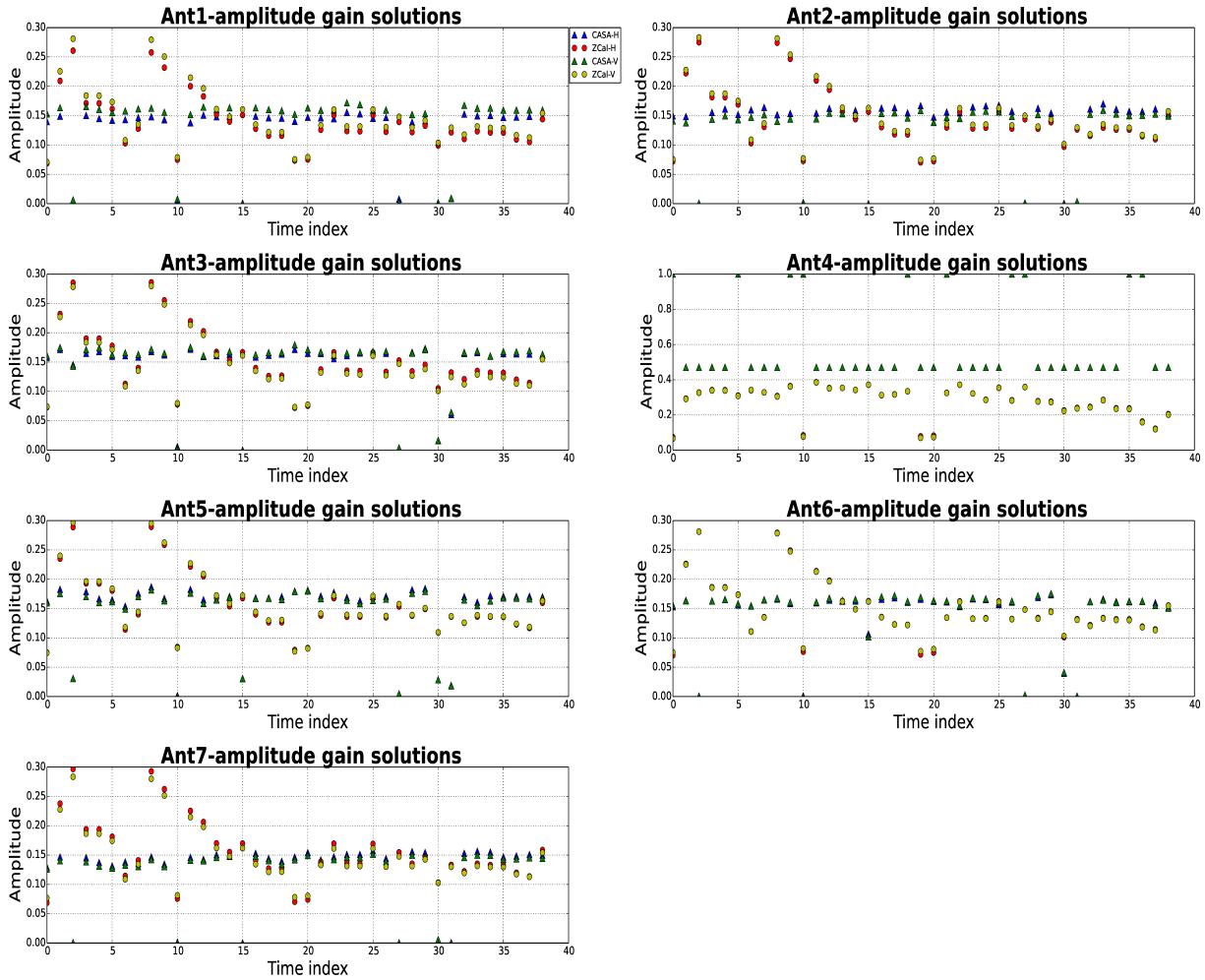


Figure 3.36: This figure shows the response of the K-nearest neighbor learning algorithm in predicting the amplitude gain solutions for the calibrator PKS1613-586 from the validation dataset-2. In each subplot we compare the the predicted amplitude gain solutions (where H polarization is represented by red and V-polarization is represented by yellow) with the CASA generated phase gain solutions (where H polarization is represented by blue and V-polarization is represented by green). The K-nearest neighbor algorithm is failing to predict the gain amplitude solutions on the new observation dataset-2. We observe that at time index > 12 the algorithm is predicting solutions slightly closer to what CASA estimated with few outliers for antenna 1,2,3,5, 6 and 7.

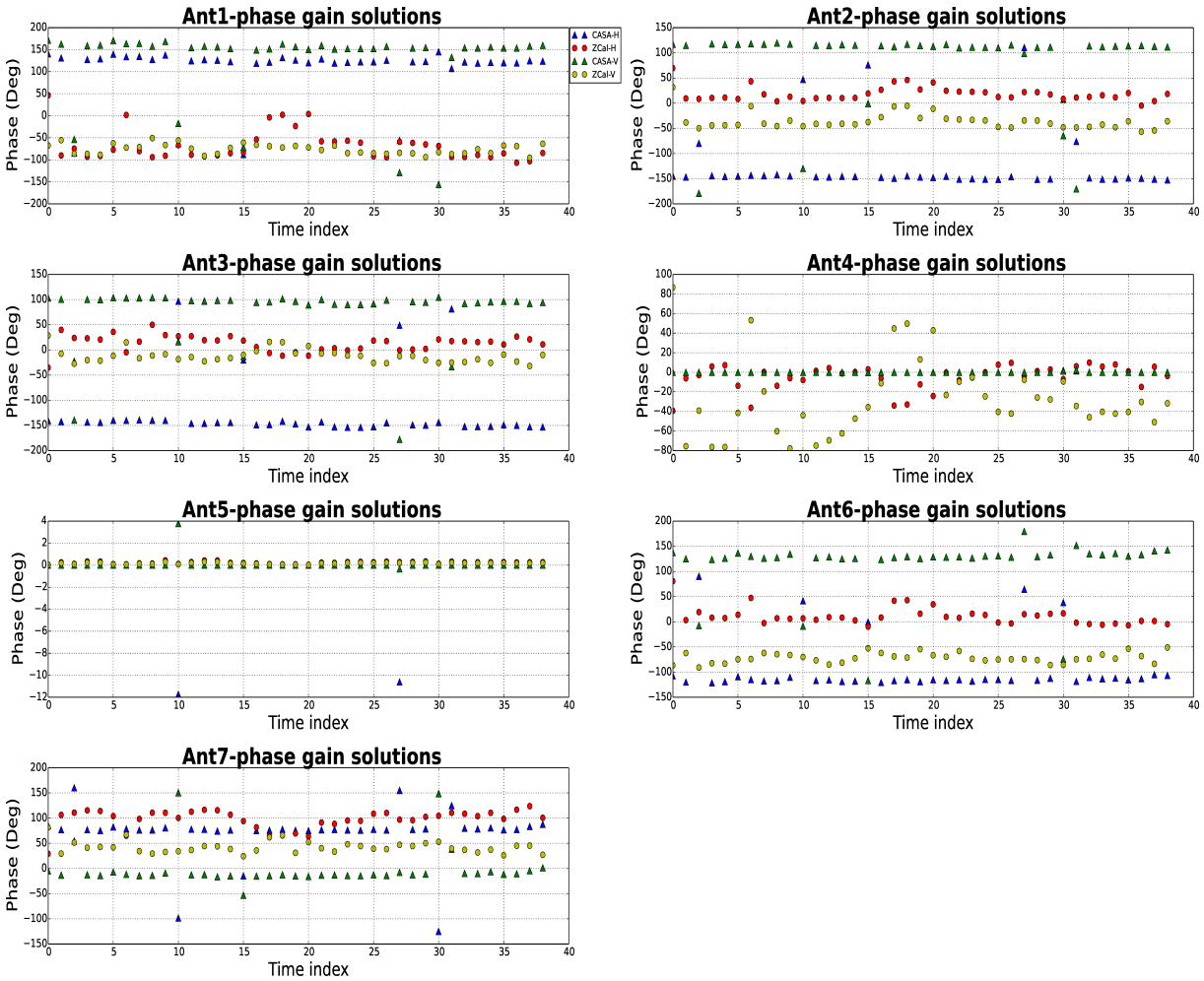


Figure 3.37: This figure shows the response of the extremely randomized tree learning algorithm in predicting the H & V phase gain solutions for the calibrator PKS1613-586 from the validation dataset-2. In each subplot we compare the predicted phase gain solutions (where H polarization is represented by red and V-polarization is represented by yellow) with the CASA generated phase gain solutions (where H polarization is represented by blue and V-polarization is represented by green). The extremely randomized tree algorithm is failing to predict the gain phase solutions for some antennas and slightly performing well in others as seen for example in antenna 1H where CASA estimated solutions at 150° and the extremely randomized tree at -150°. We also observe that it has simply learned that the reference antenna is flat.

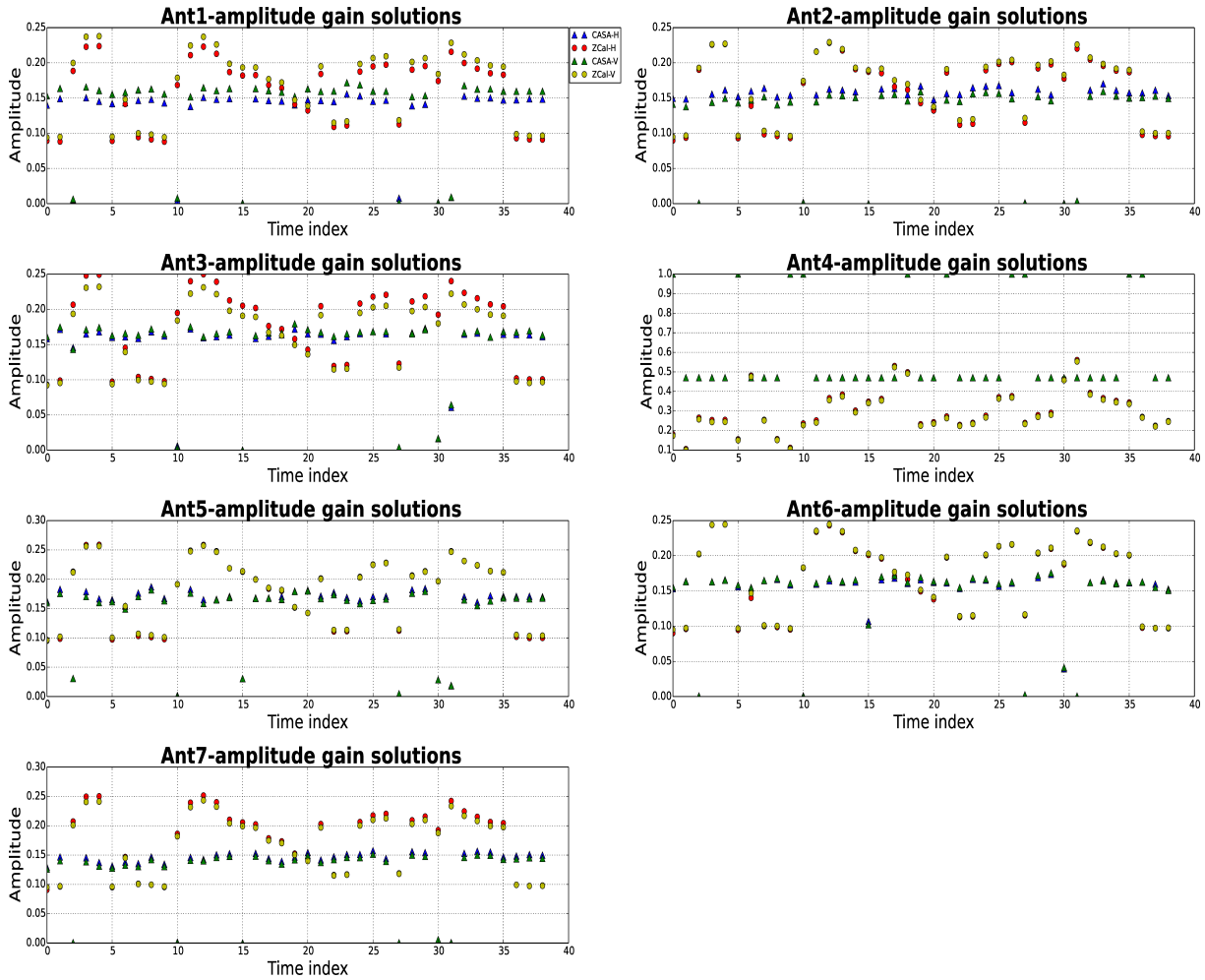


Figure 3.38: This figure shows the response of the extremely randomized tree learning algorithm in predicting the amplitude gain solutions for the calibrator PKS1613-586 from the validation dataset-2. In each subplot we compare the the predicted amplitude gain solutions (where H polarization is represented by red and V-polarization is represented by yellow) with the CASA generated phase gain solutions (where H polarization is represented by blue and V-polarization is represented by green). The extremely randomized tree algorithm is failing to predict the gain amplitude solutions on the new observation dataset-2.

For dataset-2, one observe that the random forest, extremely randomized trees, K-nearest neighbors and decision tree algorithm have not learned to generalise in predicting the gain amplitude and phase solutions when compared with CASA. Unlike from the observation dataset-1, there is no match with CASA solutions in all the models and data samples.

4. Behaviour analysis of antennas

In this chapter we analyse and discuss each antenna behaviour, looking at the results obtained in predicting the gain solutions from each experiment of our learning algorithms for both polarization H in Section 4.2 and V polarization in Section 4.3.

4.1 Pointing sensor distribution

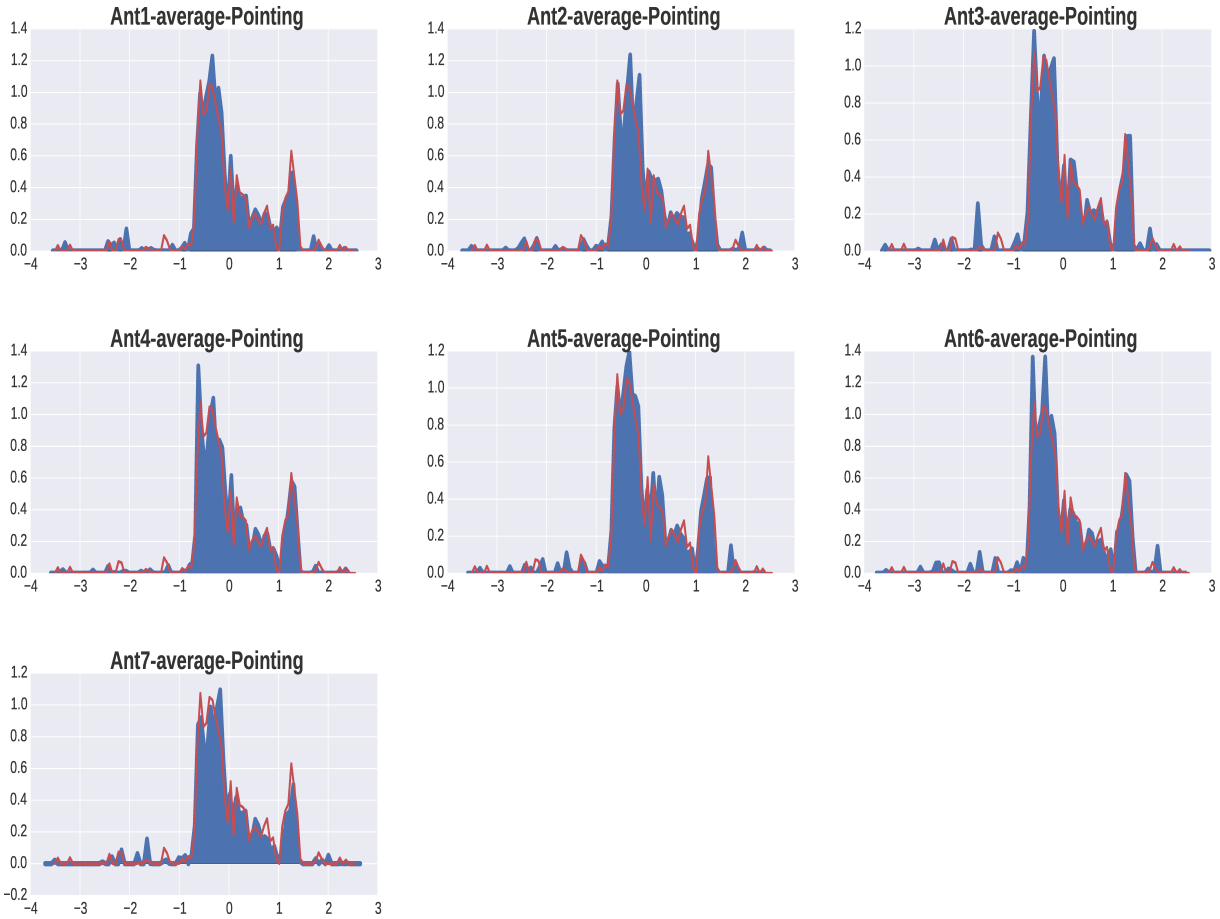


Figure 4.1: This figure shows the distribution of the averaged pointing data per antenna during the tracking of the calibrator source PKS1613-586. The blue plot represents the average of all the pointing sensors per antenna, and the red line is the reference probability distribution for all KAT-7 antennas.

Figure 4.1 illustrates how the pointing sensor data per antenna are distributed. This gives details about whether or not the antennas were pointing at the same position with reference red line as shown in Figure 4.1. One observes that each antenna pointing is distributed differently from the reference probability distribution. These small offsets might have contributed to prediction errors of amplitude and phase gain solutions.

4.2 H-polarization amplitude and phase

In this section we show the behaviour of each antenna phase and amplitude for H-polarization with respect to each model used. We evaluate each antenna behaviour using the RMSE computed during test-time in section 3.4.1.

As seen in Figure 3.2, the gain phase solution obtained from CASA for antenna 5 is zero. This is due to the selection of antenna 5 as a reference antenna during the calibration of G solutions. Fundamentally, the baseline visibility phases measured by interferometry are exclusively different between antennas, and so no absolute phase reference exists (Taylor et al., 1999). Antenna-based calibration solutions thus have phases that are constrained only to satisfy the observed visibility phases; further, each solution in time is formally independent of all others (Taylor et al., 1999), though stability (or at least continuity) in time is expected. Therefore, to assert phase continuity in time, it is conventional practice to assign a reference antenna whose phase will be held constant at zero in time and never drops out during an observation (CASA cookbook, 2009).

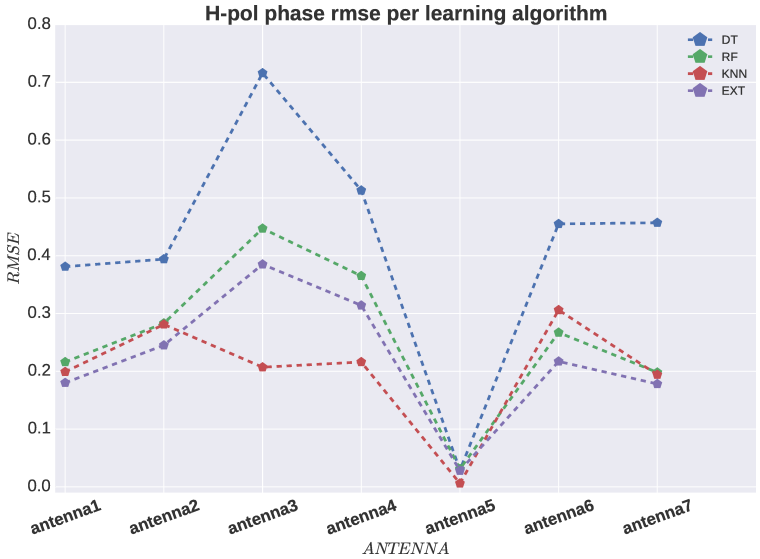


Figure 4.2: Root mean square error for each learning algorithm in predicting the h polarization phase gain solutions for each antenna. The blue, green, red and purple lines represent the different learning algorithms used in the experiment.

From the experiment in Section 3.4.1 and Figure 4.2, one observes that the learning algorithms have learned this behaviour pattern with rms error accuracy ≈ 0 Radians, resulting in predicting zero phase gain solutions for antenna 5. We validated our models looking at three observational validation data sets namely: observation test-1 and observation test-2. We provide our models with sensor data during tracking of the calibrator source PKS1613-586 as input. The phase output predictions in Figures 3.23, 3.25, 3.27, 3.29, 3.31, 3.33, 3.35, 3.37 shows that the models have learned to generalize for the reference antenna, by predicting zero phases for antenna 5 H and V polarization as trained, assuming that the antenna was stable without any drop-outs during the period of the observation. In Figure 4.2, though the models were supposed to perform differently because of their parameter settings, one notices that the random forest, K-nearest neighbor and the extremely randomised trees methods are very close to one another in rms error as a function of antenna 1H, 2H, 5H, 6H and 7H, whereas there are large large rms error bars in each model per antenna 3H and 4H. Such behaviour gives one an idea about the instability

of these two antennas. Figure 4.2 also illustrates which model can be used for better predictions of H polarization phase gain solutions. In this case, one observes that the decision tree model has larger rms error scatter than the random forest, K-nearest neighbors and the extremely randomized trees models. As discussed in Chapter 2, though we have used a randomised search algorithm to determine the optimal choice of parameters at each node, decision trees are still prone to over-fitting, especially when a tree is particularly deep, since it gets specific and complicated.Thus this leads to an increase in rms error for the decision tree model, as shown in 4.2. Such behaviour is referred to as model biasness. Figure 3.23 can be used as one of the examples to show that the decision tree model has not learned to generalize, since it predicts the wild phase solutions when compared to CASA. Thus we introduced the ensemble tree-based and nearest neighbor methods to prevent overfitting. The random forest and extremely randomized tree model in Figures 3.25 and 3.29 predict fairly stable H polarization phase solutions for observation test-1 and test-2 with antenna 6 and 7 close to CASA. As for the K-nearest neighbor algorithm, it performs poorly in predicting the H polarization phase solutions with large scattering and non-constant points over time when feeding it with the validation sets.

One of the causes of this poor prediction could be the small value of K chosen by the randomised search parameter optimisation algorithm. i.e., if the training data set is large but K is too small, then one still runs the risk of overfitting. For example, for K $= 3$ and a really large data set, there is high chance that two noisy data points close enough to each other can outvote the correct data points in some region. On the other hand, if K is chosen to be too large, then one may end up smoothing out and eliminating useful information in the distribution.

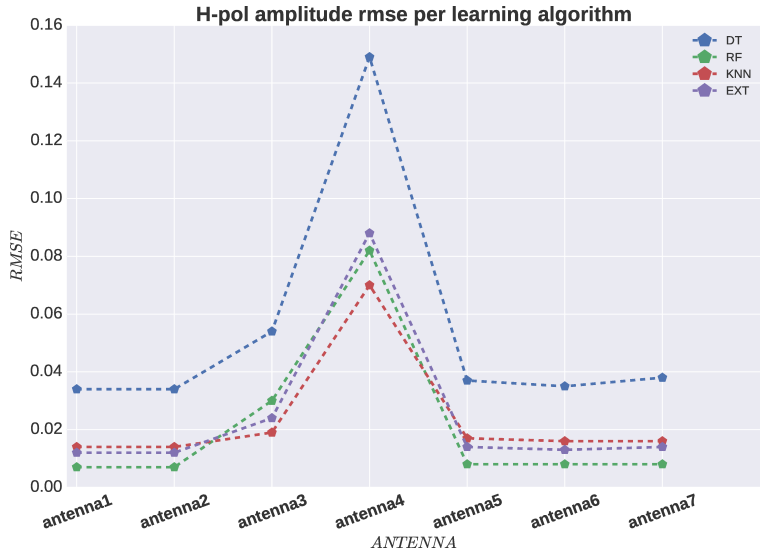


Figure 4.3: Root mean square error for each learning algorithm in predicting the h polarization amplitude gain solutions for each antenna. The blue, green, red and purple lines represent the different learning algorithms used in the experiment.

From the plot in Figure 4.3, one observes that the random forest, K-nearest neighbor, and extremely randomized trees learned to predict the H polarization amplitude gain solutions with an rms error accuracy of less than 0.02, which is much better than the phase prediction in Figure 4.2. Similar to H polarization phase prediction, one observes a large rms error variation on antenna 3H and 4H in all four models, with decision trees having larger rms error scattering than the rest. From validation data sets, Figures 3.24, 3.32 show the decision tree output prediction from all the observation tests, and one

observes that the decision tree model cannot generalize well for different new observational sets, thereby producing flat amplitude. However, these models managed to learn the most critical part, i.e., the sinusoidal variation of the gain amplitude solutions over time as shown in Figures 3.26, 3.34, 3.30, 3.38 for the validation dataset. Since the target of the experiment is the calibrator PKS1613-586, which was considered a bright source and a good calibrator during KAT-7 commissioning, we therefore compare the H polarization amplitude experimental solutions with ones obtained from CASA through calibration of the validation data sets. We observe that the machine learning amplitude is performing lower than CASA with a factor of 33.33% for observation test-1 and 22.22 % for observation test-2. On the other hand, the K-nearest neighbor seems to have failed in observation test-1 and predicted reasonably close to CASA for observation test-1 and test-2.

In Figure 3.26 and 3.30, The model predicts a drop in amplitude as a function of time in the middle of the observation from 0.12 to 0.09 level of CASA, while as the true amplitude have not dropped. From this unique behaviour observed, we can conclude that it is either due to prediction failure as the models have been trained on limited amount of observations and therefore failing on different observation settings, or it could be a behaviour that CASA did not detect as it does not include sensor data when calculating its calibration solutions.

4.3 V-polarization amplitude and phase

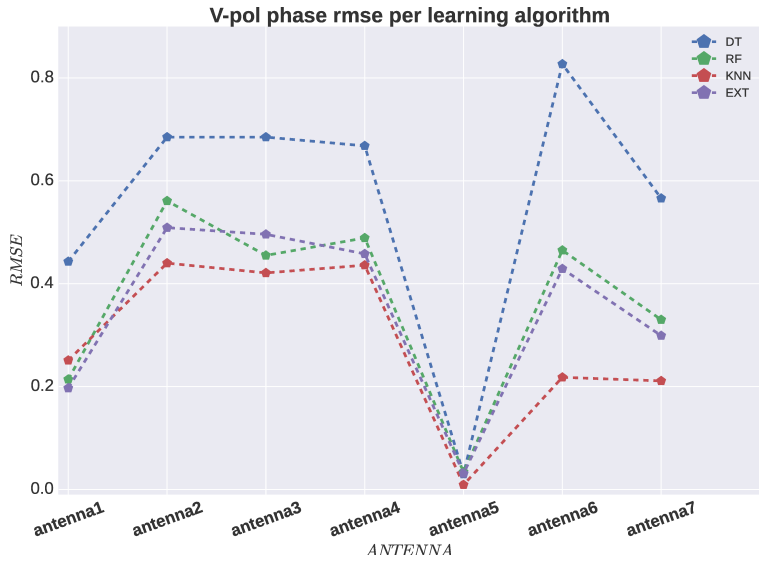


Figure 4.4: Root mean square error for each learning algorithm in predicting the V polarization phase gain solutions for each antenna. The blue, green, red and purple lines represent the different learning algorithms used in the experiment.

The model behaviour for the V polarization phase solutions experiment in Figure 4.4 is closely similar to Figure 4.2 H polarization in terms of rms error instabilities. We observed large rms error bars in each model per antenna 2V, 6V and 7V and much lower at antenna 1V, 3V, 4V, and 5V for random forest, K-nearest neighbors and the extremely randomized trees models. When feeding the models with the both both validation datasets test-1 and test-2, one observes that the random forest and extremely

randomized tree in Figure 3.25 and 3.29 predict fairly stable V polarization phase solutions in antenna 5, antenna 6 for time index < 80, antenna 7 for time index > 80 and fails on antenna 1, 2,3, and 4.

In Figure 4.2 and 4.4, we further observe that antennas with rms error > 0.5 $\mathrm{Radians}$ i.e, antenna 2,3,4 have wild phase solution for H and V polarization when predicting on validation dataset as shown in Figure 3.25, 3.27, 3.29,3.33, 3.35, 3.37.

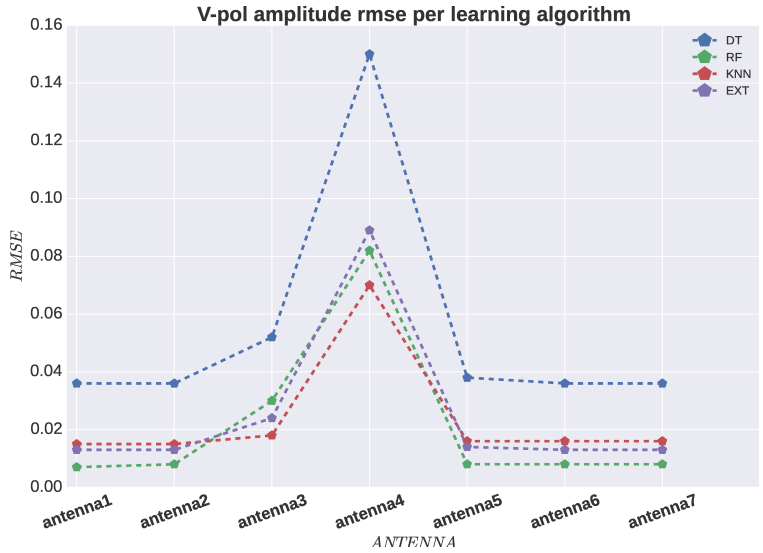


Figure 4.5: Root mean square error for each learning algorithm in predicting the V polarization amplitude gain solutions for each antenna with different colours representing different learning algorithms used in the experiment.

The V polarization rms error is no different from H polarizations as shown in Figures 4.3 and 4.5, where the rms error in both polarization for all the antennas is measured and rated the same. We observe that the machine learning amplitude is lower than CASA with a factor of 33.33% for observation test-1 and 22.22 % for observation test-2. In Figures 4.3 and 4.5 for H and V polarization, antennas with measured rms error > 0.05, i.e, are showing wild amplitude solutions prediction when feeding the random forest, K-nearest neighbor and extremely randomised models with validation datasets as shown in Figures 3.26, 3.28, 3.30, 3.34 ,3.36, 3.38 for antenna 4. This shows that the model have poorly learned the amplitude solution behaviour of this antenna.

4.4 Summary of analysis

From the analysis above, one therefore concludes that the random forest and extremely randomised trees are the better models for predicting the amplitude and phase gain solutions followed by the K-nearest neighbor. However, these models have learned to predict the amplitude gain solutions better than phase, as shown in Figures 4.3 and 4.5 with rms error less than 0.05. The machine has learned that most of the amplitude errors are caused by instrumental parameters and the environment around it. However, it has failed to learn the gain phase behaviour. This is most likely due to the fact that the ZCal was trained on instrumental based ancillary data and therefore it failed to predict the behaviour of the gain phase, since it varies quickly over time as a function of atmosphere and antenna-based (ground) effects.

Some of the limitations of our experiment may be due to training being done on time-based calibration solutions rather than frequency (bandpass).

When comparing the bias/variance of the tree-based methods, one observes that algorithm randomization increases the variance and bias of individual trees (Geurts et al., 2006). The variance increase is therefore cancelled out during the process of averaging over a sufficiently large ensemble of trees to produce an output. The bias and variance tradeoff is different in the classification and regression settings, Classification problems can tolerate high levels of biasness of a class probability estimates without yielding high classification error rates (Geurts et al., 2006).

During model training, the random forests method does the pre-sorting on the learning sample before growing all trees to avoid having re-sorting each time a node is split. This pre-sorting reduces the computing times of this method (Geurts et al., 2006). Pre-sorting makes the computational complexity of random forests depend linearly on the number of features since it requires the order of $nN \log N$ operations. The extremely randomized trees does not consider pre-sorting during training, which is a further advantage when dealing with very large data samples, as it does not keep sorted copies of the learning samples in its memory for each candidate feature, and hence the computation power increases (Geurts et al., 2006).

5. Conclusion

We have shown that the application of machine learning techniques to telescope sensor data opens a new avenue in the development of calibration methods. We have used the telescope sensor data to learn the variability of the complex gain calibration solutions for the calibrator PKS1613-586 as a function of time. We took advantage of the large amount of ancillary data generated by the telescope sensors during scientific observations, which is almost completely ignored in conventional calibration approaches. The implementation of the ZCal algorithm is based on regression machine learning algorithms, with the aim of predicting the calibration solutions and studying each antenna's behaviour. We have considered multiple observations and collected the observable quantities such as air temperature, relative humidity, wind speed, wind direction, air pressure and telescope pointings for each track on the calibrator PKS1613-586. Using the 1GC calibration solutions obtained with CASA, we constructed a matrix of training sample n_L and testing sample n_T to train the machine learning algorithms decision tree, random forest, extremely randomised trees, and K-nearest neighbor to be able to discern the patterns that relate complex gain solutions of to external parameters.

Since gain solutions are complex, we have implemented ZCal to learn on phase and amplitude separately. Each learning algorithm ran on the learning sample N times and its error was estimated on the test sample. We presented a statistical framework to measure the accuracy of each multi-output regression model and our results are encouraging with an rms error of $\approx< 0.5\mathrm{rad}$ during testing of our models using the testing data for gain amplitude and phase. Comparing the performance of these algorithms, the random forest, extremely randomized tree and K-nearest neighbours were shown to be the best for our purpose.

Applying these methods to the validation observation datasets test-1 and test-2 yielded mixed results. We observed that the environmental and the pointing sensor readings were more strongly correlated to the amplitude than phase. Consequently, the ability to predict gain-phase was overall poor; gain-amplitude prediction was accurate in some cases (capturing non-trivial behaviour such as occilations), and completely failed in others. With the benefit of physical intuition, the poor performance on phase solutions is hardly surprising – we know that water vapour in the troposphere is a significant contributor to gain-phase, and we know that our sensors are not particularly sensitive to it. However, our methods were quite deliberately designed with no inputs from any physical priors – the purpose of the project was to show that ML techniques can make available connections "blindly", without access to physical intuition. The accurate prediction of gain-amplitudes, in some cases, suggests that this is indeed feasible. It is not clear what caused the failed predictions, although we can always speculate on physical differences between observations that our sensors were not sensitive to. We can therefore expect that with access to a larger array of sensors, the ZCal approach will be able to make better gain predictions. In particular, telescopes working in the millimetre regime such as ALMA (where atmospheric effects are critical) are often equipped with water-vapour radiometers, which provide an independent measure of the atmospheric conditions. It would be very interesting to apply ZCal to such observations. The advantage of the ZCal approach is precisely in its lack of physics – as long as there is *some* correlation between sensors data and gains, we can expect that correlation to be learned without resorting to physical models.

6. Appendix

6.1 Decision tree H&V accuracy scores

Antenna	**Decision tree Phase**			
	Rmse	Rmae	R2score	Explained σ^2
Ant1-H	0.443	0.411	0.922	0.923
Ant2-H	0.685	0.471	0.869	0.87
Ant3-H	0.685	0.52	0.861	0.861
Ant4-H	0.668	0.511	0.816	0.817
Ant5-H	0.03	0.061	0.643	0.643
Ant6-H	0.827	0.54	0.784	0.784
Ant7-H	0.566	0.434	0.914	0.914
Ant1-V	0.381	0.387	0.957	0.957
Ant2-V	0.394	0.378	0.956	0.956
Ant3-V	0.716	0.519	0.839	0.839
Ant4-V	0.513	0.451	0.92	0.92
Ant5-V	0.029	0.055	0.571	0.572
Ant6-V	0.455	0.401	0.943	0.943
Ant7-V	0.457	0.397	0.913	0.914
	Decision tree Amplitude			
Ant1-H	0.034	0.106	0.824	0.824
Ant2-H	0.034	0.105	0.833	0.833
Ant3-H	0.054	0.145	0.707	0.709
Ant4-H	0.149	0.206	0.46	0.462
Ant5-H	0.037	0.11	0.83	0.83
Ant6-H	0.035	0.107	0.83	0.831
Ant7-H	0.038	0.11	0.828	0.828
Ant1-V	0.036	0.108	0.825	0.825
Ant2-V	0.036	0.11	0.822	0.822
Ant3-V	0.052	0.145	0.695	0.697
Ant4-V	0.15	0.207	0.46	0.462
Ant5-V	0.038	0.111	0.828	0.828
Ant6-V	0.036	0.107	0.826	0.826
Ant7-V	0.036	0.107	0.825	0.825

6.2 Random forest H& V accuracy scores

Antenna	**Random forest Phase**			
	Rmse	Rmae	R2score	Explained σ^2
Ant1-H	0.214	0.326	0.982	0.982
Ant2-H	0.561	0.459	0.912	0.912
Ant3-H	0.455	0.439	0.939	0.94
Ant4-H	0.489	0.418	0.901	0.902
Ant5-H	0.034	0.07	0.564	0.564
Ant6-H	0.465	0.417	0.931	0.932
Ant7-H	0.33	0.369	0.971	0.971
Ant1-V	0.216	0.318	0.986	0.986
Ant2-V	0.283	0.355	0.977	0.977
Ant3-V	0.447	0.43	0.937	0.937
Ant4-V	0.365	0.416	0.959	0.959
Ant5-V	0.031	0.062	0.516	0.516
Ant6-V	0.267	0.354	0.981	0.981
Ant7-V	0.198	0.299	0.984	0.984
	Random forest Amplitude			
Ant1-H	0.007	0.063	0.992	0.992
Ant2-H	0.007	0.064	0.992	0.992
Ant3-H	0.03	0.113	0.911	0.914
Ant4-H	0.082	0.17	0.838	0.838
Ant5-H	0.008	0.065	0.993	0.993
Ant6-H	0.008	0.063	0.992	0.992
Ant7-H	0.008	0.064	0.992	0.992
Ant1-V	0.007	0.063	0.992	0.992
Ant2-V	0.008	0.066	0.991	0.991
Ant3-V	0.03	0.113	0.9	0.903
Ant4-V	0.082	0.171	0.837	0.837
Ant5-v	0.008	0.066	0.993	0.993
Ant6-V	0.008	0.063	0.992	0.992
Ant7-V	0.008	0.062	0.992	0.992

6.3 K-nearest neighbor H&V accuracy scores

Antenna	**KNN Phase**			
	Rmse	Rmae	R2score	Explained σ^2
Ant1-H	0.251	0.24	0.975	0.975
Ant2-H	0.44	0.317	0.946	0.946
Ant3-H	0.421	0.296	0.948	0.948
Ant4-H	0.436	0.321	0.922	0.922
Ant5-H	0.009	0.033	0.967	0.967
Ant6-H	0.218	0.22	0.985	0.985
Ant7-H	0.211	0.206	0.988	0.988
Ant1-V	0.199	0.21	0.988	0.988
Ant2-V	0.281	0.251	0.978	0.978
Ant3-V	0.207	0.215	0.987	0.987
Ant4-V	0.216	0.248	0.986	0.986
Ant5-V	0.006	0.027	0.98	0.98
Ant6-V	0.306	0.25	0.974	0.974
Ant7-V	0.194	0.189	0.984	0.985
	KNN Amplitude			
Ant1-H	0.014	0.05	0.967	0.967
Ant2-H	0.014	0.048	0.972	0.973
Ant3-H	0.019	0.063	0.962	0.962
Ant4-H	0.07	0.107	0.883	0.883
Ant5-H	0.017	0.053	0.966	0.966
Ant6-H	0.016	0.053	0.965	0.965
Ant7-H	0.016	0.052	0.968	0.968
Ant1-V	0.015	0.05	0.969	0.969
Ant2-V	0.015	0.05	0.968	0.968
Ant3-V	0.018	0.062	0.963	0.963
Ant4-V	0.07	0.108	0.882	0.882
Ant5-V	0.016	0.052	0.968	0.968
Ant6-V	0.016	0.052	0.965	0.965
Ant7-V	0.016	0.05	0.967	0.968

6.4 Extremely randomized tree H&V accuracy scores

Antenna	**Extremely randomized Phase**			
	Rmse	Rmae	R2score	Explained σ^2
Ant1-H	0.197	0.305	0.985	0.985
Ant2-H	0.509	0.439	0.928	0.928
Ant3-H	0.496	0.42	0.927	0.928
Ant4-H	0.458	0.409	0.914	0.914
Ant5-H	0.031	0.062	0.63	0.63
Ant6-H	0.429	0.391	0.942	0.942
Ant7-H	0.299	0.348	0.976	0.976
Ant1-V	0.18	0.295	0.99	0.99
Ant2-V	0.245	0.331	0.983	0.983
Ant3-V	0.385	0.384	0.954	0.954
Ant4-V	0.314	0.383	0.97	0.97
Ant5-V	0.028	0.056	0.587	0.588
Ant6-V	0.217	0.332	0.987	0.987
Ant7-V	0.178	0.288	0.987	0.987
	Extremely randomized tree Amplitude			
Ant1-H	0.012	0.068	0.976	0.976
Ant2-H	0.012	0.068	0.978	0.979
Ant3-H	0.024	0.1	0.942	0.943
Ant4-H	0.088	0.161	0.811	0.811
Ant5-H	0.014	0.07	0.976	0.976
Ant6-H	0.013	0.069	0.976	0.976
Ant7-H	0.014	0.071	0.977	0.977
Ant1-V	0.013	0.069	0.977	0.977
Ant2-V	0.013	0.07	0.976	0.976
Ant3-V	0.024	0.1	0.937	0.938
Ant4-V	0.089	0.161	0.81	0.81
Ant5-V	0.014	0.071	0.977	0.977
Ant6-V	0.013	0.069	0.976	0.976
Ant7-V	0.013	0.069	0.976	0.976

References

E. Abebe. A study of potential calibrators using the kat-7 radio telescope. 2015.

D. Aberdeen, O. Pacovsky, and A. Slater. The learning behind gmail priority inbox. In *LCCC: NIPS 2010 Workshop on Learning on Cores, Clusters and Clouds*, 2010.

M. Aitkin, B. Francis, J. Hinde, and R. Darnell. *Statistical modelling in R*. Oxford University Press, 2009.

E. Alpaydin. *Introduction to machine learning*. MIT press, 2014.

A. Aniyan and K. Thorat. Classifying radio galaxies with the convolutional neural network. *The Astrophysical Journal Supplement Series*, 230(2):20, 2017.

N. M. Ball and R. J. Brunner. Data mining and machine learning in astronomy. *International Journal of Modern Physics D*, 19(07):1049–1106, 2010.

E. P. Bellinger, G. C. Angelou, S. Hekker, S. Basu, W. H. Ball, and E. Guggenberger. Fundamental parameters of main-sequence stars in an instant with machine learning. *The Astrophysical Journal*, 830(1):31, 2016.

S. Bhatnagar, T. Cornwell, K. Golap, and J. M. Uson. Correcting direction-dependent gains in the deconvolution of radio interferometric images. *Astronomy & Astrophysics*, 487(1):419–429, 2008.

Big data in radio astronomy. Expect the unexpected from the big-data boom in radio astronomy. https://theconversation.com/expect-the-unexpected-from-the-big-data-boom-in-radio-astronomy-84059, Accessed March 2018.

A. L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial intelligence*, 97(1):245–271, 1997.

H. Borchani, G. Varando, C. Bielza, and P. Larrañaga. A survey on multi-output regression. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(5):216–233, 2015.

R. Bott. About feature scaling and normalizationand the effect of standardization for machine learning algorithms. *Igarss 2014*, 2014.

L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

J. Brownlee. How to prepare data for machine learning. *Machine Learning Mastery*, 25, 2013.

L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, et al. Api design for machine learning software: experiences from the scikit-learn project. *arXiv preprint arXiv:1309.0238*, 2013.

C. Carignan, B. Frank, K. Hess, D. Lucero, T. Randriamampandry, S. Goedhart, and S. Passmoor. Kat-7 science verification: using hi observations of ngc 3109 to understand its kinematics and mass distribution. *The Astronomical Journal*, 146(3):48, 2013.

P. R. CASA cookbook. Casa user reference & cookbook. 2009.

D. Cassidy, G. Holton, and J. Rutherford. Wave motion. In *Understanding Physics*, pages 331–403. Springer, 2002.

T. Cornwell. Radio-interferometric imaging of very large objects. *Astronomy and Astrophysics*, 202: 316–321, 1988.

Cristina Petri, Cluj Napoca, 2010. Decision trees. Babes Bolyai University, http://www.cs.ubbcluj.ro/~gabis/DocDiplome/DT/DecisionTrees.pdf, Accessed February 2018.

P. Cunningham and S. J. Delany. k-nearest neighbour classifiers. *Multiple Classifier Systems*, 34:1–17, 2007.

Deep Learning Indaba. Deep learning indaba. http://www.deeplearningindaba.com/, Accessed July 2018.

P. Flach. *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press, 2012.

A. Foley, T. Alberts, R. Armstrong, A. Barta, E. Bauermeister, H. Bester, S. Blose, R. Booth, D. Botha, S. Buchner, et al. Engineering and science highlights of the kat-7 radio telescope. *Monthly Notices of the Royal Astronomical Society*, 460(2):1664–1679, 2016.

S. Fortmann-Roe. Understanding the bias-variance tradeoff. 2012.

J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

Gary, NJIT, 2014. Radio astronomy. NJIT, https://web.njit.edu/~gary/728/Lecture9.html, Accessed March 2018.

P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

T. Grobler, A. Stewart, S. Wijnholds, J. Kenyon, and O. Smirnov. Calibration artefacts in radio interferometry–iii. phase-only calibration and primary beam correction. *Monthly Notices of the Royal Astronomical Society*, 461(3):2975–2992, 2016.

P. Hall, R. Schillizzi, P. Dewdney, and J. Lazio. The square kilometer array (ska) radio telescope: Progress and technical directions. *International Union of Radio Science URSI*, 236:4–19, 2008.

T. K. Ho. Random decision forests. In *Document analysis and recognition, 1995., proceedings of the third international conference on*, volume 1, pages 278–282. IEEE, 1995.

V. Hyvönen, T. Pitkänen, S. Tasoulis, E. Jääsaari, R. Tuomainen, L. Wang, J. Corander, and T. Roos. Fast k-nn search. *arXiv preprint arXiv:1509.06957*, 2015.

Inteferometersl. Inteferometers i. https://www.cv.nrao.edu/course/astr534/Interferometers1.html, Accessed MArch 2018.

G. James, D. Witten, T. Hastie, and R. Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

K-nearest neighbor algorithm. Knn classifier, introduction to k-nearest neighbor algorithm. http://dataaspirant.com/2016/12/23/k-nearest-neighbor-classifier-intro/.

E. A. Kassaye. *A study of potential calibrators using the KAT-7 radio telescope*. PhD thesis, University of Cape Town, 2015.

J. Kazemitabar, A. Amini, A. Bloniarz, and A. S. Talwalkar. Variable importance using decision trees. In *Advances in Neural Information Processing Systems*, pages 425–434, 2017.

D. Levy. Introduction to numerical analysis. *Department of Mathematics and Center for Scientific Computation and Mathematical Modeling (CSCAMM) University of Maryland*, pages 2–2, 2010.

Machine learning. What is machine learning? https://www.mathworks.com/discovery/machine-learning.html, Accessed May 2018.

S. Marsland. *Machine learning: an algorithmic perspective*. CRC press, 2015.

J. McMullin, B. Waters, D. Schiebel, W. Young, and K. Golap. Casa architecture and applications. In *Astronomical data analysis software and systems XVI*, volume 376, page 127, 2007.

R. S. Michalski, J. G. Carbonell, and T. M. Mitchell. *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.

MIT Haystack Observatory. Brief introduction to radio astronomy. http://www.haystack.mit.edu/edu/pcr/RFI/RFI_page10-19.pdf, Accessed June 2017.

MLSS. Machine learning summer school. http://mlss2018.net.ar/, Accessed July 2018.

G. Moisen. Classification and regression trees. 2008.

G. Molenaar and O. Smirnov. Kern. *arXiv preprint arXiv:1710.09145*, 2017.

J. Morgan. Classification and regression tree analysis. *Report No1. Boston University School of Public Health*, 2014.

D. R. Musicant, J. M. Christensen, and J. F. Olson. Supervised learning by training on aggregate outputs. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 252–261. IEEE, 2007.

N. J. Nilsson. Introduction to machine learning. an early draft of a proposed textbook. 1996.

NIPS. Neural information processing systems. https://nips.cc/Conferences/2017, Accessed July 2018.

J. E. Noordam and O. M. Smirnov. The meqtrees software system and its use for third-generation calibration of radio interferometers. *Astronomy & Astrophysics*, 524:A61, 2010.

J. Ott and J. Kern. Casa synthesis & single dish reduction reference manual & cookbook. *Version May*, 29, 2013.

V. Pandey, J. van Zwieten, A. de Bruyn, and R. Nijboer. Calibrating lofar using the black board selfcal system. In *The Low-Frequency Radio Universe*, volume 407, page 384, 2009.

S. Patro and K. K. Sahu. Normalization: A preprocessing stage. *arXiv preprint arXiv:1503.06462*, 2015.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

F. Provost and T. Fawcett. Data science and its relationship to big data and data-driven decision making. *Big data*, 1(1):51–59, 2013.

C. Rossant. *IPython Interactive Computing and Visualization Cookbook: Over 100 hands-on recipes to sharpen your skills in high-performance numerical computing and data science in the Jupyter Notebook*. Packt Publishing Ltd, 2018.

S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

M. Slabber. Overview of the monitoring data archive used on meerkat. 2015.

M. Slabber. Scalable time series documents store icalepcs2017. 2017.

M. Slabber and M. T. Ockards. Illustrate the flow of monitoring data through the meerkat telescope control software. 2015.

D. R. Smith. Dynamic wind effects on large telescopes. 2002.

Society of Amateur Radio Astronomers. Introduction to radio astronomy. http://www.radio-astronomy.org/pdf/sara-beginner-booklet.pdf, Accessed June 2017.

K. Staats. *Genetic programming applied to RFI mitigation in radio astronomy*. PhD thesis, University of Cape Town, 2016.

S. Stanimirovic. Short-spacings correction from the single-dish perspective. *arXiv preprint astro-ph/0205329*, 2002.

M. Stephen. Machine learning: an algorithmic perspective. *USA: Chapman &Hall/CRC*, 2009.

G. B. Taylor, C. L. Carilli, and R. A. Perley. Synthesis imaging in radio astronomy ii. In *Synthesis Imaging in Radio Astronomy II*, volume 180, 1999.

Techemergence. Black scholes. techemergence,everyday-examples-of-ai, https://www.techemergence.com/everyday-examples-of-ai/, Accessed January 2018.

The Electromagnetic Spectrum. The electromagnetic spectrum. https://chem.libretexts.org/Textbook_Maps/Introductory_Chemistry/Map%3A_Introductory_Chemistry_(Tro)/09%3A_Electrons_in_Atoms_and_the_Periodic_Table/9.3%3A_The_Electromagnetic_Spectrum, Accessed May 2017.

The SKA telescope. The ska telescope. https://www.skatelescope.org/, Accessed May 2017.

P. Thévenaz, T. Blu, and M. Unser. Image interpolation and resampling. *Handbook of medical imaging, processing and analysis*, 1(1):393–420, 2000.

A. R. Thompson, J. M. Moran, and G. W. Swenson. Interferometry and synthesis in radio astronomy. 2001.

A. R. Thompson, J. M. Moran, G. W. Swenson, et al. *Interferometry and synthesis in radio astronomy*. Springer, 2017.

S. Upnere, N. Jekabsons, and R. Joffe. Characterization of wind loading of the large radio telescope. 2012.

G. Verschuur. *The invisible universe: the story of radio astronomy*. Springer, 2015.

M. H. Wieringa. An investigation of the telescope based calibration methods 'redundancy'and 'self-cal'. *Experimental Astronomy*, 2(4):203–225, 1992.

Wikipedia. Black scholes. Wikipedia, the Free Encyclopedia, http://en.wikipedia.org/wiki/Black%E2%80%93Scholes, Accessed April 2012.

T. L. Wilson, K. Rohlfs, and S. Hüttemeister. *Tools of radio astronomy*. Springer Science & Business Media, 2013.

I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

A. Wright. Single-dish radio astronomy. 2004.

J. A. Zensus, P. Diamond, P. Napier, et al. *Very long baseline interferometry and the VLBA*, volume 82. San Francisco: Astronomical Society of the Pacific, 1995.

Y. Zhang and Y. Zhao. Astronomy in the big data era. *Data Science Journal*, 14, 2015.

Z.-H. Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012.