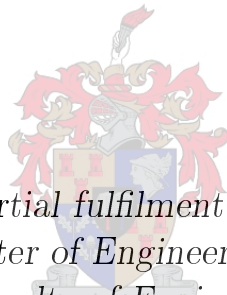


An RFI Simulation Pipeline to Help Teach Interferometry and Machine Learning

by

Insight Enya Aku Agbetsiafa



*Thesis presented in partial fulfilment of the requirements for
the degree of Master of Engineering (Electrical and
Electronic) in the Faculty of Engineering at Stellenbosch
University*

Supervisor: Dr. Trienko L. Grobler

Co-supervisor: Prof. Dirk de Villiers

April 2022

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: April 2022

Copyright © 2022 Stellenbosch University
All rights reserved

Abstract

An RFI Simulation Pipeline to Help Teach Interferometry and Machine Learning

Insight Enya Aku Agbetsiafa

Department of Electrical and Electronic Engineering,

University of Stellenbosch,

Private Bag X1, Matieland 7602, South Africa.

Thesis: MEng (Elec)

April 2022

An interferometer is a collection of radio antennas that together form one instrument. Machine Learning is the collective term that is used to refer to a set of algorithms that can automatically learn to perform a specific task if it is provided with training examples. Interferometry has become an intricate part of the scientific landscape in South Africa with the advent of MeerKAT. Similarly, utilizing Machine Learning (ML) to improve our lives has grown in popularity worldwide. Machine Learning is nowadays used to determine the likes of people, to interpret human utterings, to automatically classify images and the like. As these two fields grow in popularity and importance within the South African context, so does the development of tools that can aid in teaching these fields to undergraduate students.

A major problem for radio observatories worldwide is Radio Frequency Interference (RFI). RFI can be detected using ML. A simulator that can simulate interferometric observations that are corrupted by RFI can serve as a testbed for different ML approaches. Moreover, if the simulator is simplistic enough it can even be utilized as a teaching tool. In this thesis such a simulator is developed. This simulator can aid in teaching students how visibilities can be simulated and how RFI can be detected via ML. In effect, one tool that can help teach two relevant undergraduate topics, namely interferometry and ML. In particular, an experiment is proposed which an undergraduate student can repeat to gain a deeper understanding of interferometry and ML. In this experiment, visibilities are simulated, RFI is injected and detected using four different ML techniques, namely Naive Bayes, Logistic Regression, k -means

ABSTRACT

iii

and Gaussian Mixture Models (GMM). The results are then analysed and conclusions are drawn. For the simplistic setup considered here, the ranking of the four algorithms is from best to worst: Naive Bayes, Logistic Regression, GMM and then k -means. In the future, if the simulator is extended somewhat, it can also be used as a testbed for comparing numerous other ML algorithms. The thesis also provides a comprehensive review of all the theory that a student requires to master both interferometry and ML.

Uittreksel

'n RFI Simulasie Pyplyn om te Help met die Onderrig van Interferometrie en Masjienleer

(“An RFI Simulation Pipeline to Help Teach Interferometry and Machine Learning”)

Insight Enya Aku Agbetsiafa

Departement Elektriese en Elektroniese Ingenieurswese,

Universiteit van Stellenbosch,

Privaatsak X1, Matieland 7602, Suid Afrika.

Tesis: MIng (Elek)

April 2022

'n Interferometer is 'n versameling van radio antennas wat saam een instrument vorm. Masjienleer is die kollektiewe term wat gebruik word om te verwys na 'n stel algoritmes wat automaties kan leer hoe om 'n spesifieke funksies te verrig, gegee afrigtingsvoorbeelde. Interferometrie, het 'n belangrike deel van die wetenskaplike landskap in Suid-Afrika geword met die loots van MeerKAT. Soortgelyk, masjienleer se gebruik het wêreldwyd drasties gegroei. Masjienleer word deesdae gebruik om die voorkeure van mense te bepaal, om die woorde wat mense uiter te herken, om prentjies te klassifiseer en dies meer. Soos wat die twee velde se gewildheid groei, word dit al hoe meer belangrik om toepassings te ontwikkel wat gebruik kan word om te help om die twee velde aan voorgraadse studente te verduidelik.

'n Groot probleem wat radio-sterrewagte in die gesig staar is Radio Frekwensie Inmenging (RFI). RFI kan met behulp van masjienleer geïdentifiseer word. 'n Simulator wat sigbaarheidsmetings kan genereer wat besmet is met RFI kan gebruik word om verkillende masjienleer tegnieke met mekaar te vergelyk. Verder, as 'n simulator eenvoudig genoeg is, kan dit ook gebruik word as 'n onderrigstoepassing. In hierdie tesis word so 'n simulator ontwikkel. Die simulator kan gebruik word om beide, interferometrie en masjienleer, aan studente te verduidelik. Meer spesifiek, 'n eksperiment word voorgestel wat studente sal kan herhaal. In die eksperiment word sigbaarheidsmetings genereer wat vermeng word met RFI. Vier masjienleer algoritmes word dan

gebruik om die RFI te identifiseer. Die vier algoritmes is: Naïewe Bayes, Logistiese Regressie, Gausiese Mengsel Modelle (GMM) en k -gemiddeldes. Die akkuraatheidsrangorde van die vier algoritmes, soos in die studie bevind, is dieselfde as wat hier gegee is. As die simulator uitgebrei word kan dit ook gebruik word om verkeie ander masjienleeralgoritmes met mekaar te vergelyk. Die tesis bevat ook 'n oorsig van al die teorie wat 'n student sou kon help om beide velde te bemeester.

Acknowledgements

I would like to express my sincere gratitude to the following people and organisations for their contribution to making this project a success:

- The Almighty God who has been ever faithful and for seeing me through.
- My family and loved ones for their support and prayers.
- Dr. Grobler for his help, guidance, dedication and contributions throughout the journey of this research work.
- The South African SKA Project Office for the financial support granted to me.

God bless you all richly!

Dedications

To my late mum for her love, prayers and support

Contents

Declaration	i
Abstract	ii
Uittreksel	iv
Acknowledgements	vi
Dedications	vii
Nomenclature	xvi
1 Introduction	1
1.1 Background to the Study	1
1.2 Research Objectives	2
1.3 Organisation of Study	3
2 Interferometry	5
2.1 A Brief History of Radio Astronomy and Interferometry	5
2.1.1 Interferometers	7
2.2 Coordinate Systems	8
2.2.1 Celestial Coordinates	8
2.2.2 Direction Cosine Coordinates	11
2.2.3 Baseline Coordinates	13
2.3 Visibilities	15
2.4 Fourier Transform	17
2.5 Calibration and Synthesis Imaging	18
2.5.1 Imaging	18
2.5.2 Calibration	21
3 Radio Frequency Interference	24
3.1 Introduction	24
3.1.1 Interference	24
3.1.2 RFI Signals	25
3.1.3 Radio Frequency Spectrum	25

<i>CONTENTS</i>	ix
3.2 RFI Sources	26
3.2.1 Satellites	26
3.3 Detection of Interference	29
3.3.1 RFI Monitoring	29
3.4 Elimination of Interference	30
3.4.1 Pre-correlation Phase	31
3.4.2 Post-correlation Phase	31
3.5 RFI Simulators	33
3.5.1 HIDE and SEEK package Simulator	33
3.5.2 RFsim	34
3.5.3 HERA_sim	34
3.6 Generation of Noise and RFI	34
4 Machine Learning	36
4.1 Introduction	36
4.2 Machine Learning Applications	37
4.2.1 Steps Involved in Machine Learning	37
4.3 Machine Learning Methods	38
4.3.1 Supervised Learning	38
4.3.2 Unsupervised Learning	39
4.3.3 Semi-supervised Learning	39
4.4 Basic Overview of some Statistical Relationships	39
4.5 Classification	40
4.6 Probabilistic Generative Models (PGM)	40
4.6.1 Shared Covariance	41
4.6.2 Non-shared Covariance	43
4.6.3 Naive Bayes Classifier	44
4.7 Probabilistic Discriminative Models (PDM)	45
4.7.1 Logistic Regression Classifier	45
4.7.2 Newton-Raphson's method	47
4.8 Unsupervised Classification	48
4.8.1 k -means Clustering	48
4.8.2 Gaussian Mixture Models (GMM)	48
4.9 Confusion Matrix	49
4.9.1 Performance Metrics	50
4.10 <i>Iris</i> dataset	51
4.10.1 Results	52
4.10.2 Observations and conclusions	52
5 Testing, Results and Analysis	55
5.1 Introduction	55
5.2 Simulator Design	55
5.3 Simulator Settings and Programmatic Flow	57

<i>CONTENTS</i>	x
5.4 Simulator Limitations	60
5.5 Simulator Validation	60
5.6 Machine Learning and RFI Detection Results	61
5.6.1 Naive Bayes Results	63
5.6.2 Logistic Regression Results	63
5.6.3 Gaussian Mixture Model (GMM) Results	64
5.6.4 k -means Results	66
5.6.5 Average accuracy	67
5.6.6 Standard Deviation	68
5.6.7 True Positive	69
5.6.8 True Negative	73
5.7 Discussions, Summary and Conclusion	75
5.8 Teaching Tool	78
6 Conclusion and Recommendation	80
6.1 Conclusion	80
6.2 Recommendation for Future Studies	81
List of References	82

List of Figures

1.1	South African MeerKAT radio telescope	1
1.2	Images of some Radio Interferometry and Machine Learning Textbooks	3
	(a) <i>Interferometry and Synthesis in Radio Astronomy Text-</i> <i>book</i> [1]	3
	(b) <i>Deep Learning Textbook</i> [2]	3
	(c) <i>Hands-on ML using Scikit-learn, Keras and Tensorflow</i> <i>Textbook</i> [3]	3
2.1	Image of Karl Jansky	6
2.2	The Very Large Array (VLA)	7
2.3	Diagram of a single baseline interferometer	8
2.4	Diagram of the celestial sphere showing the equator and the poles	9
2.5	Earth shown covered in an imaginary grid of latitude and lon- gitude lines	10
2.6	The relationship between direction α , H and LST	11
2.7	Diagram showing the uvw and lmn coordinate systems	12
2.8	The relationship between direction cosines coordinates and the celestial coordinate system	12
2.9	Angles associated with the direction cosine coordinates	13
2.10	Relation between the horizontal frame $(\mathcal{A}, \mathcal{E})$ and the (E, N, U) cartesian frame [4].	14
2.11	Relationship between the XYZ and uvw coordinates	16
2.12	Self Calibration Framework [4]	22
3.1	Image showing data being flooded by a satellite transmission . .	25
3.2	Radio Frequency Spectrum	26
3.3	Frequency channels being corrupted in the MeerKAT L-band . .	27
3.4	A zoomed in version of the MeerKAT L-band's quietest portion	28
3.5	Spillover from an Orbiting Earth satellite	28
3.6	Block diagram of how the RFI Monitor works [5]	30
4.1	Image of a non-linear classifier	43
4.2	Structure of confusion matrix for binary classification (two classes)	50
4.3	Iris dataset	51

(a)	<i>Iris setosa</i>	51
(b)	<i>Iris versicolor</i>	51
(c)	<i>Iris virginica</i>	51
4.4	Scatter plot of the versicolor and virginica of the Iris dataset . .	52
4.5	Confusion matrix and decision boundary when the Naive Bayes classifier is employed	53
(a)	Confusion Matrix	53
(b)	Plot of decision boundary	53
4.6	Confusion matrix and decision boundary when the Logistic Regression classifier is employed	53
(a)	Confusion Matrix	53
(b)	Plot of decision boundary	53
4.7	Confusion matrix and decision boundary when the <i>k</i> -means classifier is employed	54
(a)	Confusion Matrix	54
(b)	Plot of decision boundary	54
4.8	Confusion matrix and decision boundary when the GMM classifier is employed	54
(a)	Confusion Matrix	54
(b)	Plot of decision boundary	54
5.1	Block diagram describing how the simulator works	56
5.2	Plot of KAT-7 antenna layout	58
5.3	<i>uv</i> -track of a single baseline (baseline 12 and 21)	61
5.4	<i>uv</i> -coverage	61
5.5	Real and imaginary part of visibilities plotted	61
5.6	Sampled visibilities with their real and imaginary components .	62
5.7	Waterfall plot of the phase and amplitude of visibilities	62
5.8	Waterfall plots of visibilities with thermal noise and RFI added	63
(a)	Visibility with noise	63
(b)	Visibility with RFI effects	63
5.9	Confusion matrix and decision boundary plot for Naive Bayes (0 dB and -11 dB)	64
(a)	Confusion Matrix	64
(b)	Plot of decision boundary	64
5.10	Scatter plot of the Naive Bayes classifier (0 dB and -11 dB) . . .	65
5.11	Confusion matrix and decision boundary plot for Naive Bayes (5 dB and -17 dB)	65
(a)	Confusion Matrix	65
(b)	Plot of decision boundary	65
5.12	Scatter plot of the Naive Bayes classifier (5 dB and -17 dB) . . .	66
5.13	Confusion matrix and decision boundary plot for Naive Bayes (10 dB and -13 dB)	66

(a)	Confusion Matrix	66
(b)	Plot of decision boundary	66
5.14	Scatter plot of the Naive Bayes classifier (10 dB and -13 dB) . .	67
5.15	Confusion matrix and decision boundary plot for Logistic Regression (0 dB and -11 dB	67
(a)	Confusion Matrix	67
(b)	Plot of decision boundary	67
5.16	Scatter plot of the Logistic Regression classifier (0 dB and -11 dB)	68
5.17	Confusion matrix and decision boundary plot for Logistic Regression (5 dB and -17 dB	68
(a)	Confusion Matrix	68
(b)	Plot of decision boundary	68
5.18	Scatter plot of the Logistic Regression classifier (5 dB and -17 dB)	69
5.19	Confusion matrix and decision boundary plot for Logistic Regression (10 dB and -13 dB	69
(a)	Confusion Matrix	69
(b)	Plot of decision boundary	69
5.20	Scatter plot of the Logistic Regression classifier (10 dB and -13 dB)	70
5.21	Confusion matrix and decision boundary plot for GMM (0 dB and -11 dB	70
(a)	Confusion Matrix	70
(b)	Plot of decision boundary	70
5.22	Scatter plot of the GMM classifier (0 dB and -11 dB)	71
5.23	Confusion matrix and decision boundary plot for GMM (5 dB and -17 dB	71
(a)	Confusion Matrix	71
(b)	Plot of decision boundary	71
5.24	Scatter plot of the GMM classifier (5 dB and -17 dB)	72
5.25	Confusion matrix and decision boundary plot for GMM (10 dB and -13 dB	72
(a)	Confusion Matrix	72
(b)	Plot of decision boundary	72
5.26	Scatter plot of the GMM classifier (10 dB and -13 dB)	73
5.27	Confusion matrix and decision boundary plot for the k -means classifier (0 dB and -11 dB	73
(a)	Confusion Matrix	73
(b)	Plot of decision boundary	73
5.28	Scatter plot of the k -means classifier (0 dB and -11 dB)	74
5.29	Confusion matrix and decision boundary plot for the k -means classifier (5 dB and -17 dB	74

LIST OF FIGURES

xiv

(a) Confusion Matrix	74
(b) Plot of decision boundary	74
5.30 Scatter plot of the k -means classifier (5 dB and -17 dB)	75
5.31 Confusion matrix and decision boundary plot for the k -means classifier (10 dB and -13 dB	75
(a) Confusion Matrix	75
(b) Plot of decision boundary	75
5.32 Scatter plot of the k -means classifier (10 dB and -13 dB)	76
5.33 A graph of average accuracies of all classifiers versus the SNR combinations	76
5.34 A graph of average standard deviation of all classifiers versus the SNR combinations	77
5.35 A graph of average true positives of all classifiers versus the SNR combinations	78
5.36 A graph of average true negatives of all classifiers versus the SNR combinations	79

List of Tables

3.1	RFI corrupted regions of the L-band range (900MHz - 1670MHz) [6]	27
5.1	The ENU (east-north-up) coordinates of KAT-7	57
5.2	Observation conducted with KAT7	58
5.3	The average confusion matrices for Naive Bayes	64
5.4	Average confusion matrices for Logistic Regression	64
5.5	Average confusion matrices for GMM	70
5.6	Average confusion matrices for k -means	71
5.7	Table referring to the teaching tool consisting of locations of python files in the github repository where one can go to learn how this simulator was created.	79

Nomenclature

Constants

$c =$	299792458 m/s ¹ (Speed of light in a vacuum)
$\pi =$	3.14159265

Variables/Symbols

τ_g	Geometric delay
α	Right ascension
δ	Declination
H	Hour angle
H_0	Starting hour angle
H_1	Stopping hour angle
B	Number of baselines
N	Number of antennas
\mathcal{A}	Azimuth
\mathcal{E}	Elevation
D	Baseline length
L	Latitude
f	Observational frequency
λ	Observational wavelength
$d\Omega$	Infinitesimal solid angle
V	Visibility
I	Sky distribution
$S(u, v)$	Sampling function
\mathcal{F}	Fourier Transform
\circ	Convolution
A	Astronomical source data
R	Sample of RFI corrupted data
N	Noise generated from a complex Gaussian distribution

T_{sys}	System temperature
σ	Either standard deviation or logistic sigmoid function
Δv	Observational bandwidth
τ	Integration time for a visibility
v	Frequency
t	Time
pq	Baseline
$\langle \cdot \rangle$	Averaging over v , t and pq
\odot	Hadamard product
D, \bar{D}	Noise-free visibility cube and its conjugate
N, \bar{N}	Noise cube and its conjugate
$P(X, Y)$	Probability of event X and event Y occurring
$P(X Y)$	Conditional probability that event X will occur given that Y has occurred
$D = \{\mathbf{X}, \mathbf{y}\}$	D is a dataset, the rows \mathbf{x}_i of \mathbf{X} represent observation vectors with corresponding label y_i , where $i \in \{1, \dots, N\}$
C_j	Class where $j \in \{1, \dots, k\}$
$P(C_j \mathbf{x})$	Class probability
$P(\mathbf{x} C_j)$	Class conditional density
$P(C_j)$	Class prior
$a(\mathbf{x})$	Log posterior odds
Σ	Covariance
$\boldsymbol{\mu}$	Mean
$\nabla E(\mathbf{w})$	Gradient of the log-likelihood function
$l(\mathbf{w})$	Negative log-likelihood
\mathbf{H}	Hessian
π_j	Mixture coefficients
\mathcal{N}	Gaussian density
a	Pareto parameter
f	Field of view
c	Number of channels
e	Number of experiments to perform
s	Signal-to-noise ratio value due to the thermal noise
r	Signal-to-noise ratio value due to the RFI
dB	Decibel

Coordinates

l, m, n	Direction cosine coordinates
XYZ	XYZ coordinates
uvw	uvw coordinates

Vectors and Tensors

\hat{s}	Unit vector
\vec{b}	Baseline vector
\mathbf{w}	Weight vector

Acronyms and Abbreviations

RFI	Radio Frequency Interference
ML	Machine Learning
ENU	East-North-Up
KAT	Karoo Array Telescope
LST	Local Sidereal Time
NCP	North Celestial Pole
VLA	Very Large Array
FoV	Field of View
FT	Fourier Transform
PSF	Point Spread Function
DDE	Direction Dependent Effect
SKA	Square Kilometre Array
ITU	International Telecommunication Union
kHz	Kilohertz
GHz	Gigahertz
VLF	Very Low Frequency
LF	Low Frequency
MF	Medium Frequency

HF	High Frequency
VHF	Very High Frequency
UHF	Ultra High Frequency
SHF	Super High Frequency
ELF	Extremely High Frequency
GPS	Global Positioning System
GSM	Global System for Mobile communication
GLONASS	Global Orbiting Navigation Satellite System
GNSS	Global Navigation Satellite Systems
PNT	Positioning, Navigation and Timing
BDS	BeiDou Navigation Satellite System
IRNSS	Indian Regional Navigation Satellite System
QZSS	Quasi-Zenith Satellite System
WSRT	Westerbork Synthesis Radio Telescope
LOFAR	Low Frequency Array
CUSUM	Cumulative Sum Method
CASA	Common Astronomy Software Applications
CNN	Convolutional Neural Network
SDP	Science Data Processing
ROC	Receiver Operating Characteristic
AUC	Area Under the ROC Curve
RFC	Random Forest Classifier
HIDE	HI Data Emulator
SEEK	Signal Extraction and Emission Kartographer
HERA	Hydrogen Epoch of Reionization Array
SNR	Signal-to-Noise Ratio

*NOMENCLATURE***xx**

GMM	Gaussian Mixture Model
AI	Artificial Intelligence
PGM	Probabilistic Generative Models
PDM	Probabilistic Discriminative Models
MLE	Maximum Likelihood Estimation
EM	Expectation Maximization
GUI	Graphical User Interface
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative

Chapter 1

Introduction

1.1 Background to the Study

Radio astronomy is a branch of astronomy that involves the study of radio emissions from celestial sources at radio frequencies. Radio telescopes receive radio waves being emitted from celestial sources in the universe. An image of the MeerKAT radio telescope is seen in Figure 1.1. Radio telescopes are discussed in greater detail in Chapter 2. The signals received by radio antennas



Figure 1.1: A view of two out of the 64 dishes of the South African MeerKAT radio telescope [7].

get corrupted by Radio Frequency Interference (RFI).

RFI lowers the quality of data; in the worst case the data becomes unusable. Radio data therefore needs to be cleaned thoroughly before use; by utilizing elimination or reduction techniques. This is discussed in greater detail in Chapter 3.

Radio observatories have also developed effective physical RFI mitigation strategies. These measures include the building of observatories in secluded places. Another is prohibiting the use of some bandwidths at and around the observatories and also using natural cover such as the mountains to protect radio telescopes from RFI [1].

Machine learning (ML) refers to the study of a set of algorithms that learns from data. It is discussed further in Chapter 4. It can be used in various ways but in this project, it is used for RFI detection.

It is important to highlight the fact that this project is multifaceted. An RFI simulator is proposed within this thesis which can serve as both a teaching tool and a testbed (if extended) of RFI detection algorithms. In the latter regard, four algorithms namely Naive Bayes, Logistic Regression, k -means and Gaussian Mixture Models (GMM) were considered. The performance of these methods were determined and their accuracy is reported in Chapter 5. The name of this simulator is *Simulator-Insight*, it can be downloaded here: <https://github.com/Insight-Agbetsiafa/Simulator-Insight>.

Let us now discuss the motivation behind the development of this simulator. The theoretical standard textbooks that are currently being used to teach interferometry and ML, respectively, are *Interferometry and Synthesis in Radio Astronomy* [1] and *Deep Learning* [2]. Recently, some textbooks have come out which are more practical in nature in both fields, namely *Fundamentals of Radio Interferometry* [4] and *Hands-on ML using Scikit-learn, Keras and Tensorflow* [3]. Images of some of the textbooks are seen in Figure 1.2. The practical interferometry textbook [4] can be further supplemented via simulators like, the *Friendly Virtual Radio Interferometer* [8], *APSYNSIM* [9] and *Pynterferometer* [10]. The aforementioned simulators are all radio interferometry related.

This notion can be extended further, by developing a simulator that can aid in teaching both interferometry and ML. This can be best accomplished by developing an RFI simulator. There exists a host of RFI simulators, namely *RFIsim* [11], *HIDE* [12] and *hera_sim* [13]. Most of these simulators are *deductive*, meaning they employ the physical laws of nature to produce output values. In this thesis, an *inductive* simulator is proposed. Inductive simulators reproduce the statistical properties of a dataset directly; it does not fall back on physics to do so. The fact that the proposed simulator is inductive makes it simplistic, which in turn implies that it can be effectively utilized as a teaching tool.

1.2 Research Objectives

To summarize, there are three main objectives:



(a) *Interferometry and Synthesis in Radio Astronomy Textbook* [1]

(b) *Deep Learning Textbook* [2]

(c) *Hands-on ML using Scikit-learn, Keras and Tensorflow Textbook* [3]

Figure 1.2: Images of some Radio Interferometry and Machine Learning Textbooks

Objective 1

Develop a simplistic inductive simulator which can be utilized as a teaching tool. This tool should enable students to master the fields of interferometry and ML. The objective of this project is to develop a simulation pipeline (using Python programming) which would serve as a teaching tool, i.e. it must enable students learn the fundamentals of interferometry and machine learning in a practical way.

Objective 2

Provide a concise summary of the theory a student would need to master both interferometry as well as ML.

Objective 3

Compare the efficacy of four ML algorithms when used to detect RFI, namely Naive Bayes, Logistic Regression, k -means and Gaussian Mixture Models (GMM) using the aforementioned simulator. Draw initial preliminary conclusions as to which algorithm is best suited to detect RFI.

Of course these conclusions should be confirmed, once the simulator is extended. Some of the limitations of the simulator are listed in Section 5.4 of Chapter 5.

1.3 Organisation of Study

This thesis is divided into six chapters:

1. The first chapter gives an overview of the background to the study, the research objectives and the organisation of the study.

2. The second chapter includes the background and theory on the fundamentals of radio interferometry.
3. The third chapter consists of research and reviewed literature in relation to RFI.
4. The fourth chapter discusses ML and includes a detailed explanation on the various classification algorithms that were used in this project.
5. The fifth chapter includes the methodology of the project, limitations, results and analysis.
6. The sixth chapter lists conclusions and future recommendations.

Chapter 2

Interferometry

In this chapter, the background regarding interferometry is presented. A historical introduction is given in Section 2.1, the coordinate systems that are used within interferometry are discussed in Section 2.2, the nature of visibilities and the Van Cittert-Zernike theorem are discussed in Section 2.3, the fact that the visibility function and the sky distribution function form a Fourier pair is stated in Section 2.4 and lastly, an overview of imaging and calibration is presented in Section 2.5.

2.1 A Brief History of Radio Astronomy and Interferometry

Radio astronomy is the study of celestial bodies that give off radio emission. Examples of such bodies include planets and galaxies. To detect the radio emissions coming from celestial bodies, a radio telescope can be used [14]. Karl G. Jansky was a radio engineer from Bell Telephone Laboratories. He was the first to conduct a radio astronomy experiment (in 1931). His task was to study and detect static interference which affected telephone signals [14]. In order for him to make further observations, he designed and built a directional antenna system which was around 30 metres long by 4 metres high and placed it on a platform with four wheels that could rotate horizontally (azimuth) [15]. Jansky made observations with this antenna which he grouped into three types of static. The first type was local thunderstorms, the second was distant thunderstorms and the third was a faint steady hiss from an unknown origin, which he later realized was emanating from the centre of the Milky Way Galaxy. He could not further his research because he had no financial support. Fortunately, Grote Reber who was also a radio engineer knew about Jansky's ideas and decided to extend it by building a parabolic reflector radio telescope. In addition to this antenna system moving along the azimuth, it could move in an upward and downward direction too [15].

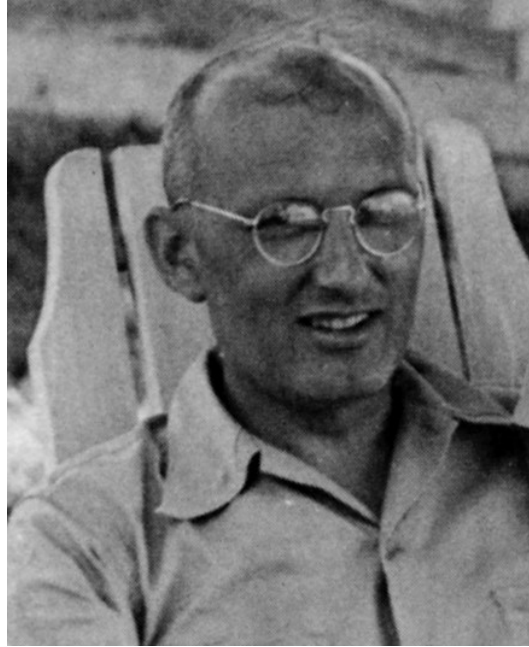


Figure 2.1: Image of Karl Jansky who was the first to conduct a radio astronomy experiment in 1931 [15].

With time, bigger antennas were designed and built in order to detect very weak signals from sources that are further away. A single dish telescope does not have a large collecting area since it is made of a single dish. This makes it impossible to achieve higher resolution. To solve this problem, a technique known as interferometry was developed, which entails creating an array using several individual radio telescopes [1].

In 1946, Martin Ryle and Derek Vonberg built a radio interferometer using this technique to examine cosmic radio emission which had been detected by previous scientists [1]. This was how interferometry originated.

An interferometer works by using interference principles to determine the locations of sources that radiate waves. This dates back to Albert Abraham Michelson who developed the Michelson interferometer which was used to measure the diameter of the red giant Betelgeuse in an experiment [1]. This interferometer comprised of a half-silvered mirror known as a beam-splitter and two other mirrors. When light falls on the beam-splitter, half of the beam goes to one mirror and the other half goes to the second mirror. After being reflected by the two mirrors, these beams merge again and are then processed by the detector. The difference in path length of the two light beams results in the formation of a fringe pattern which is then examined by the detector [16].



Figure 2.2: An array of radio telescopes known as the Very Large Array (VLA). Aside the MeerKAT in Figure 1.1, this is another type of radio telescope comprising of 27 dishes [17].

2.1.1 Interferometers

An interferometer consists of several individual antennas working together to form a single instrument. The instrument so obtained has a higher resolution than any of its constituent antennas. This is due to the fact that it is made of multiple antennas in an array, and the size of the array determines the amount of incoming radiation that can be collected or how best very weak signals can be detected. Several baselines can be formed from the various antennas. A baseline can be defined as the difference vector which is formed by the coordinates of two antennas. The interferometer tracks a point on the sky known as the *phase centre*. Regardless of the distance between other sources, time delays are introduced for waves arriving in phase across the array of antennas. When the time delayed signals from all antennas are correlated with one another, visibilities are created.

In Figure 2.3, the two dishes point in the same direction as the unit vector \hat{s} . The angle θ is the angle between the baseline vector \vec{b} and \hat{s} ; c is the speed of light in a vacuum. The *geometric delay*, τ_g is the difference in the arrival times of plane waves. This delay that is experienced by a plane wave arriving at antenna 1 in this figure is:

$$\tau_g = \frac{\vec{b} \cdot \hat{s}}{c} \quad (2.1)$$

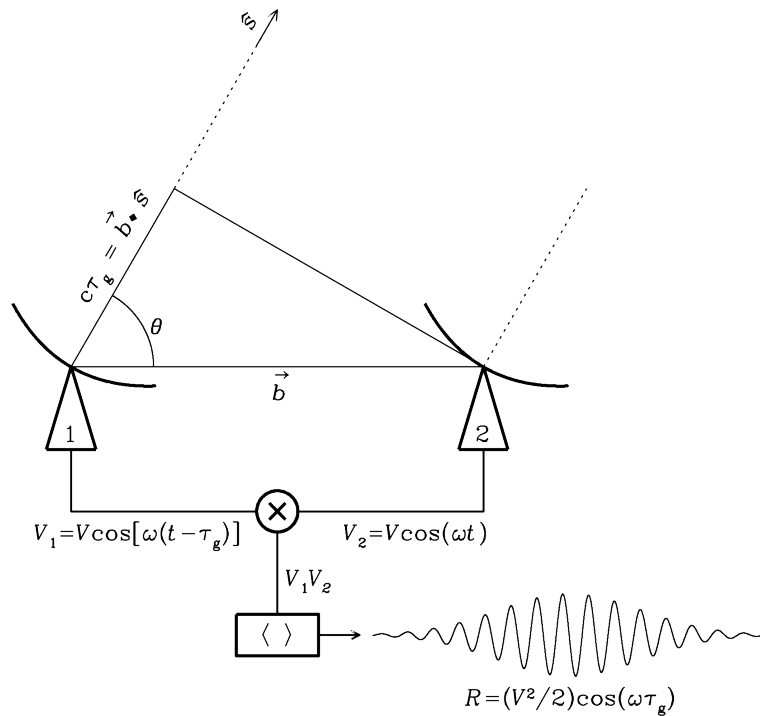


Figure 2.3: Diagram of a single baseline interferometer: two antennas separated by the baseline vector \vec{b} of length b pointing from antenna 1 to antenna 2 [18].

2.2 Coordinate Systems

Coordinate systems for astronomical source positions, antennae positions and their associated baseline vectors are important in interferometry. These coordinate systems are used to identify positions of objects in space [4].

One of the coordinate systems used in this project is the equatorial coordinate system. The coordinates used in this system include: right ascension (α) and declination (δ). These coordinates are of the J2000 epoch which reports the source locations of celestial objects as they were in the year 2000 [1]. The coordinate systems used for the antennas are the ENU (East-North-Up), XYZ and uvw coordinates. These coordinates are rotations of one another, measured in the same distance units. Further details are presented in section 2.2.3 on page 13.

2.2.1 Celestial Coordinates

Celestial coordinates are coordinates used to pinpoint positions of celestial bodies on the celestial sphere.

The universe can be projected onto an imaginary sphere which surrounds earth. This sphere is known as the celestial sphere [4]. The celestial equator

which is the projection of the earth's equator onto it can be found on the celestial sphere. The north and south celestial poles are also determined by projecting the Earth's poles onto the celestial sphere [4]. This is illustrated in Figure 2.4.

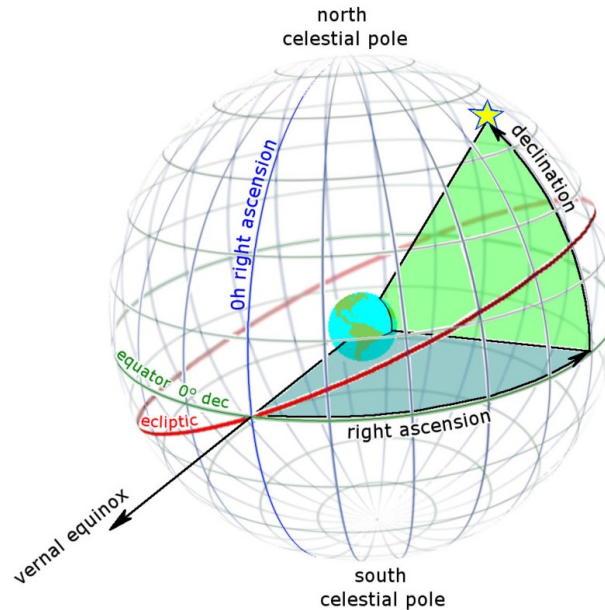


Figure 2.4: Diagram of the celestial sphere showing the equator and the poles with the ball-like structure inside the big sphere representing the Earth. The equatorial coordinates: right ascension and declination (in green) are being shown too. The red circle is the sun's apparent path around the sky, which defines the ecliptic [19].

2.2.1.1 Equatorial Coordinates

Equatorial coordinates is a popular coordinate system used to locate objects on the celestial sphere. These coordinates are independent of the location of the observer and the time of the observation. Thus, observers in different locations and on different times can use the same coordinates. Just as an individual can pinpoint locations on Earth using the latitude and longitude of that location as depicted in Figure 2.5, so can celestial objects be located using their equatorial coordinates.

This coordinate system is similar to the geographical coordinate system used on Earth. They use the same basic plane and poles as that of the geographic coordinate system. The main difference between the equatorial and geographic coordinate systems is that: the equatorial system is fixed to the stars which seems to rotate across the sky which is not actually the case. The geographic coordinates on the other hand is fixed to the earth [4].

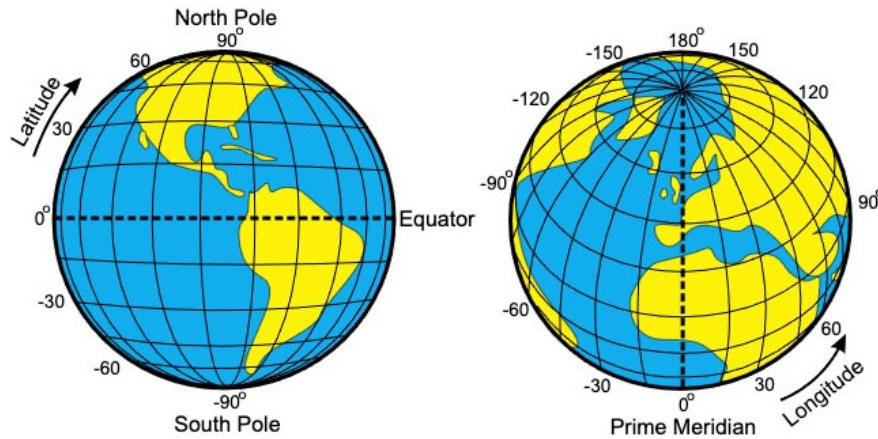


Figure 2.5: Earth shown covered in an imaginary grid of latitude lines (measured from 0° to 90° north and south of the equator) and longitude lines (measured from 0° to 180° east and west of the prime meridian). The East and West longitude suffixes are occasionally replaced by a negative sign for the western hemisphere and a positive sign for the eastern hemisphere [19].

Equatorial coordinates of an object is defined by its right ascension and declination. Declination refers to an angle and indicates by how much an object lies above or below the celestial equator. Its unit of measure is degrees, arcminutes and arcseconds. The right ascension measures the angle of objects east of the Vernal Equinox and its unit of measure is hours, minutes and seconds or in units of time [4]. Vernal Equinox is the point where the celestial equator intersects the ecliptic (path sun travels on celestial sphere) which serves as a reference point for all other celestial bodies [4].

2.2.1.2 Hour Angle:

Hour circle is any great circle on the celestial sphere that passes through the celestial poles, intersecting the celestial equator perpendicularly. Hour angle (H) is like a dynamic right ascension and is defined as the angular distance between the hour circle of a celestial object and the local meridian measured along the celestial equator [4]. This angle is commonly measured in hours, minutes and seconds [15]. The hour angle is expressed mathematically as $H = LST - \alpha$ where α is the right ascension and LST, the Local Sidereal Time (i.e the hour angle of the first point of Aries). Astronomers are interested in observing the stars. For this reason, they use a sidereal time-keeping system to keep track of the vernal equinox instead of the sun [4].

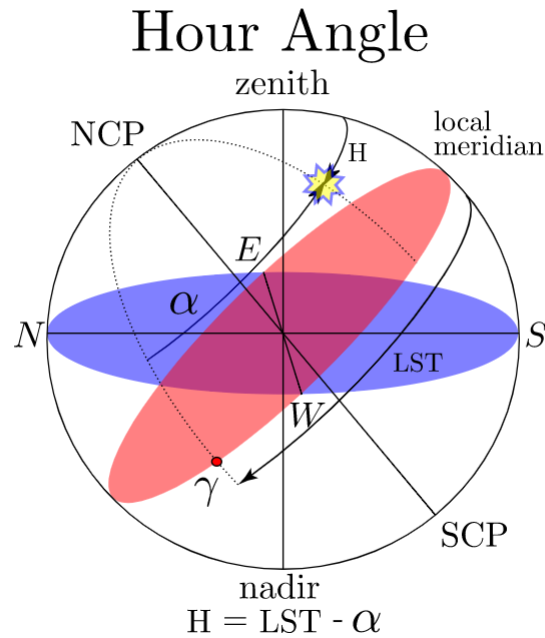


Figure 2.6: The relationship between direction α , H and LST. The red plane represents the fundamental plane of the celestial coordinate system. The blue plane also represents the fundamental plane of the horizontal coordinate system [4].

2.2.2 Direction Cosine Coordinates

Another useful coordinate system which is used to pinpoint a location on the celestial sphere are the l , m and n coordinates. The l , m and n coordinates are actually direction cosines and are aligned with the u, v, w coordinates as depicted in Figure 2.7.

Direction cosine coordinates can be used to highlight the fact that the sky brightness distribution and the visibility function form a Fourier pair. Figure 2.8 shows the fundamental plane of the direction cosine coordinate system and the fundamental plane of the equatorial system represented in blue and red respectively. The field centre, \mathbf{S}_c is used as an arbitrary reference point and the n -axis points toward it. \mathbf{S} represents the direction cosine position vector of a celestial body. If this vector is projected onto the lm -plane, then this resulting vector will have a length equal to $\sin\theta$. Here, θ denotes the angular distance between the field centre \mathbf{S}_c and \mathbf{S} measured along the surface of the celestial sphere [4]. If θ is small, the aforementioned length is also equal to $\sqrt{l^2 + m^2}$, which implies that $l^2 + m^2 \approx \theta^2$. Therefore, $\sqrt{l^2 + m^2}$ may be interpreted as the angular distance measured between the source at \mathbf{S} and the field centre \mathbf{S}_c measured along the surface of the celestial sphere. The l and m coordinates may be measured in degrees [4].

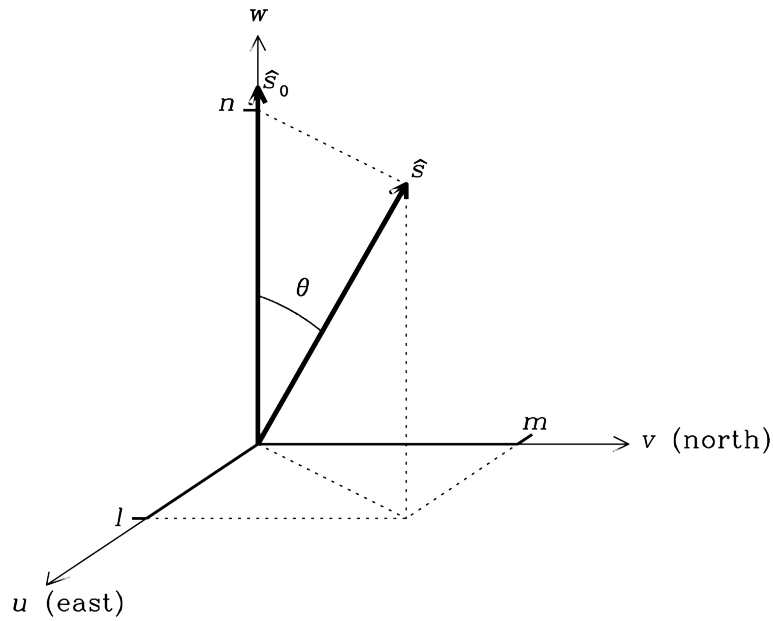


Figure 2.7: Diagram showing the two fundamental coordinate systems in interferometry. Namely, the uvw coordinate system and lmn coordinate system[18]

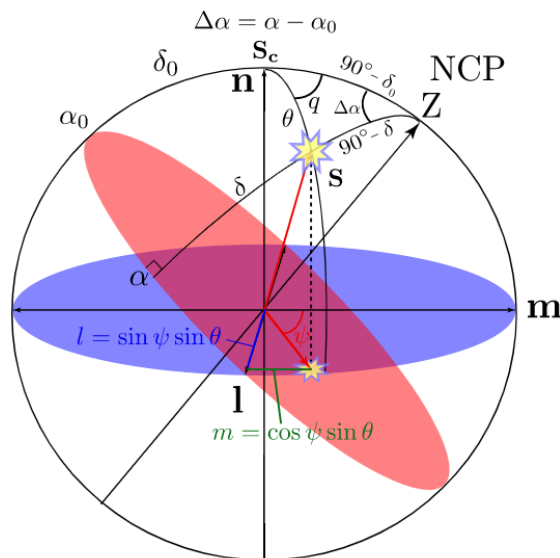


Figure 2.8: The relationship between direction cosines coordinates and the celestial coordinate system. The fundamental plane of the direction cosine coordinate system represented in blue and the fundamental plane of the equatorial system represented in red. Since the radius of the celestial sphere is equal to one, the orthogonal fundamental axes of the direction cosine coordinate system l , m and n can be labelled [4].

The l, m, n coordinates are dimensionless direction cosines but if $|\mathbf{a}|=1$, they become cartesian coordinates:

$$\begin{aligned} l &= \frac{a_1}{|\mathbf{a}|} = \cos(\alpha) \\ m &= \frac{a_2}{|\mathbf{a}|} = \cos(\beta) \\ n &= \frac{a_3}{|\mathbf{a}|} = \cos(\gamma) \end{aligned} \quad (2.2)$$

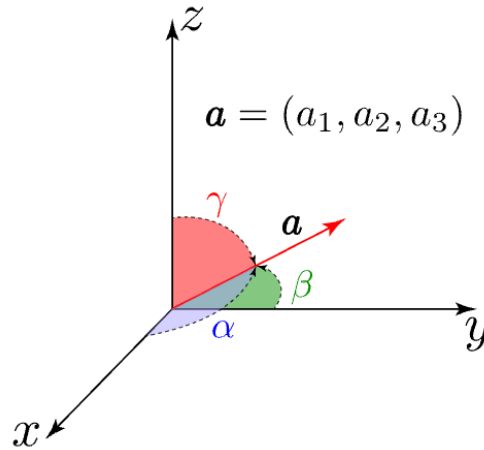


Figure 2.9: Direction cosine coordinates (l, m, n) and their angles represented by α , β and γ respectively [4].

From the right ascension (α) and declination (δ) of sources, the l and m coordinates can be obtained if the field-center is known. This is expressed in Equation (2.3).

$$\begin{aligned} l &= \sin(\theta) \sin(\psi) = \cos(\delta) \sin(\alpha - \alpha_0) \\ m &= \sin(\theta) \cos(\psi) = \sin(\delta) \cos(\delta_0) - \cos(\delta) \sin(\delta_0) \cos(\alpha - \alpha_0) \\ \delta &= \sin^{-1} (m \cos \delta_0 + \sin \delta_0 \sqrt{1 - l^2 - m^2}) \\ \alpha &= \alpha_0 + \tan^{-1} \left(\frac{1}{\cos \delta_0 \sqrt{1 - l^2 - m^2} - m \sin \delta_0} \right) \end{aligned} \quad (2.3)$$

2.2.3 Baseline Coordinates

A baseline is the distance or separation vector between two antennas in an interferometric array [4]. In an interferometric array comprising of antennas,

several baselines are formed by every pair of antennas in the array. For instance, for the 7-dish Karoo Array Telescope (KAT-7) consisting of 7 antennas, 21 baselines are formed. To find the number of baselines given the number of antennas, this formula is used:

$$B = \frac{N^2 - N}{2} \quad (2.4)$$

where N represents the number of antennas and B represents number of baselines.

2.2.3.1 ENU Coordinates

The ENU coordinate system also known as the East-North-Up coordinate system is a basic antenna coordinate system measured in meters. These coordinates are relative to the local horizon. Another useful coordinate system is the horizontal frame (\mathcal{A} , \mathcal{E}). Here \mathcal{A} denotes *azimuth* and \mathcal{E} denotes *elevation*. Figure 2.10 shows the relation between the horizontal frame (\mathcal{A} , \mathcal{E}) and the ENU cartesian frame. The azimuth is referred to as the angle in the plane of the local horizon measured in the clock-wise direction from North to East. The elevation also is the angle that is measured from the horizon to the local zenith [4].

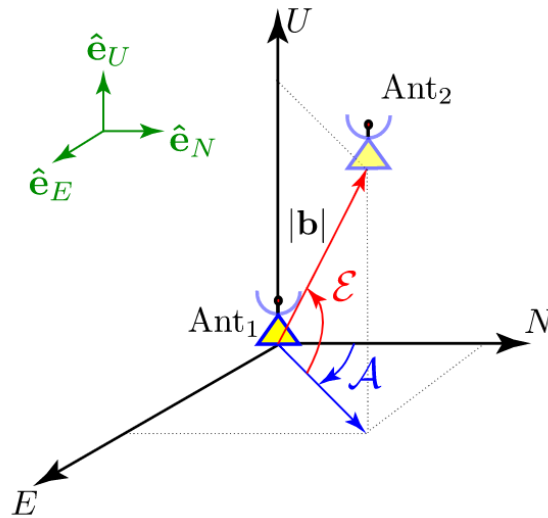


Figure 2.10: Relation between the horizontal frame (\mathcal{A} , \mathcal{E}) and the (E, N, U) cartesian frame [4].

Baseline vectors are differences between coordinates and to calculate a baseline between antenna one and antenna two, the following vector can be calcu-

lated:

$$\vec{b} = (b_x, b_y, b_z) = (x_2 - x_1, y_2 - y_1, z_2 - z_1) \quad (2.5)$$

2.2.3.2 XYZ Coordinates

A new coordinate system, XYZ, is obtained from a single rotation of the ENU coordinate system. The X-axis of the XYZ coordinate system points towards $(0^h, 0^\circ)$ depicting the point where the vernal equinox crosses the local meridian, the Y-axis points towards $(-6^h, 0^\circ)$ due East and the Z-axis passes through the North Celestial Pole (NCP) [4; 1].

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = D \begin{bmatrix} \cos L \sin \mathcal{E} - \sin L \cos \mathcal{E} \cos \mathcal{A} \\ \cos \mathcal{E} \sin \mathcal{A} \\ \sin L \sin \mathcal{E} + \cos L \cos \mathcal{E} \cos \mathcal{A} \end{bmatrix} \quad (2.6)$$

where D represents the baseline length and L represents the latitude.

2.2.3.3 uvw Coordinates

The *uvw* coordinate is yet another cartesian coordinate system. Antenna positions are also described when using this coordinate system.

Equation 2.6 above can be transformed into a *uvw* frame. This new frame is found in equation 2.7. Here δ denotes the declination of the field center and H_0 denotes its hour angle, while λ denotes the observational wavelength. This same equation shows that the *u* coordinate of a baseline depends only on the hour angle of the source but does not depend on the declination δ .

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \frac{1}{\lambda} \begin{bmatrix} \sin H_0 & \cos H_0 & 0 \\ -\cos H_0 \sin \delta_0 & \sin H_0 \sin \delta_0 & \cos \delta_0 \\ \cos H_0 \cos \delta_0 & -\sin H_0 \cos \delta_0 & \sin \delta_0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (2.7)$$

The *uv*-coverage of a baseline is generated by converting the baseline vectors in the XYZ coordinate system to the *uvw* coordinate system as the hour angle of the centre of the observational field varies over time. The *uvw* coordinate system is fixed to the centre of the observation field [20]. Figure 2.11 shows the relationship between the XYZ and *uvw* coordinate systems. The phase reference also known as the field centre of the observation is represented by \mathbf{s}_0 and the equatorial coordinates of \mathbf{s}_0 are (H_0, δ_0) .

2.3 Visibilities

Considering a quasi-monochromatic plane wave of frequency f , striking a pair of antennas such as those depicted in Figure 2.3. Assuming the voltage induced

$$V^2 e^{-2\pi i f \tau_g} \quad (2.12)$$

A small region around the phase centre is considered since the directional feedback of a parabolic dish is very small, thus around 1° on the sky. The output correlator in equation(2.12) is directly proportional to V^2 which is also proportional to the power received (i.e. the brightness distribution $I_v(\mathbf{s})$ per unit frequency at v in direction \mathbf{s}). More formally:

$$\begin{aligned} r(\boldsymbol{\tau}) &= \int I_v(\mathbf{s}) e^{-2\pi i f \tau_g} d\Omega \\ &= \int I_v(\mathbf{s}) e^{-2\pi i \frac{\mathbf{b} \cdot \mathbf{s}}{\lambda}} d\Omega \end{aligned} \quad (2.13)$$

$d\Omega$ describes an element of infinitesimal solid angle on the celestial sphere that can also be expressed in terms of (l, m, n) coordinates by using the Jacobian determinant, i.e.

$$d\Omega = \frac{dldm}{n} = \frac{dldm}{\sqrt{1-l^2-m^2}} \quad (2.14)$$

The Van Cittert-Zernike theorem now follows from Equation 2.1, Equation 2.13 and Equation 2.14:

$$V(u, v, w) = \int \int I(l, m) e^{-2\pi i [ul+vm+w(n-1)]} \frac{dldm}{n}$$

where $n = \sqrt{1-l^2-m^2}$. Note that for the sake of brevity, some of the steps needed to derive the Van Cittert-Zernike theorem are not given here. The reader interested in these steps is referred to [1].

2.4 Fourier Transform

Assuming the field of view (fov) of a telescope, defined by the primary antenna beam is small such that $l^2 + m^2 \ll 1 \implies n \approx 1$. It now follows that the w -term in the Van Cittert-Zernike theorem becomes negligible. This same equation, therefore, simplifies and becomes:

$$\begin{aligned} V(u, v) &= \int \int I(l, m) e^{-2\pi i (ul+vm)} dldm \\ &= \mathcal{F}\{I(l, m)\} \\ I(l, m) &\approx \mathcal{F}^{-1}\{V(u, v)\} \end{aligned} \quad (2.15)$$

Equation (2.16) reveals that the visibility function and the sky distribution function form a Fourier pair.

2.5 Calibration and Synthesis Imaging

A brief overview of calibration and imaging is provided in this section although they are not explored in our pipeline.

2.5.1 Imaging

Imaging entails producing an image of the sky from visibility measurements of an interferometric array [4]. The sky distribution, I and the visibility function, V form a Fourier pair. This can be written as:

$$V \Leftrightarrow I \quad (2.16)$$

Equation (2.16) is only valid when the uv -plane is sampled continuously such that the visibility is measured for all values of (u, v) . If there are finite number of antennas, the uv -plane is sampled at discrete points:

$$S(u, v) = \sum_k \delta(u_k, v_k) \quad (2.17)$$

where u_k and v_k are the (u, v) points measured by the telescope and $S(u, v)$ denote the sampling function. The feedback of an imaging system to a point source is known as the *Point Spread Function (PSF)*. The following equation shows the Fourier Transform relationship between the PSF and the sampling function:

$$S(u, v) \Leftrightarrow PSF(l, m) \quad (2.18)$$

Now the dirty image can be defined as:

$$I^D(l, m) \approx \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} S(u, v) V_{obs}(u, v) e^{-2\pi i(ul+vm)} dudv \quad (2.19)$$

where $V_{obs}(u, v)$ represents the observed visibilities, $S(u, v)$ represents the sampling function and $I^D(l, m)$ represents the dirty image. The sampled visibilities V^S become the product of the observed visibility function V_{obs} and the sampling function S :

$$V^S = S(u, v) V_{obs}(u, v) \quad (2.20)$$

Taking the fourier transform of equation 2.20 results in:

$$I^D = \mathcal{F}\{V^S\} = \mathcal{F}\{SV_{obs}\} \quad (2.21)$$

It now follows from the convolution theorem that the dirty image can be described as the convolution of two functions, $\mathcal{F}\{S\}$ and $\mathcal{F}\{V_{obs}\}$:

$$I^D = \mathcal{F}\{S\} \circ \mathcal{F}\{V_{obs}\} \quad (2.22)$$

$\mathcal{F}\{S\}$ is the PSF response of an array and depends on the array configurations. $\mathcal{F}\{V_{obs}\}$ is the Fourier transform of the visibility space that is fully sampled. The (l, m) and (u, v) symbols are discarded for simplicity.

It can therefore be concluded that, the dirty image results from the convolution of the Fourier transform of the sampling function i.e. the PSF with the Fourier transform of the visibility function i.e. the true sky/ideal image.

$$I^D = PSF \circ I^{ideal} \quad (2.23)$$

After a dirty image is made, deconvolution is performed. This is described in the following section.

2.5.1.1 Deconvolution

Visibility sampling causes a dirty image containing bright sources which are surrounded by PSF-like sidelobes [4]. These sidelobes in the PSF can be removed using deconvolution methods. Deconvolution is therefore defined as the process of correcting a dirty image by removing the sidelobes caused by the PSF from the image [4].

Residual image (noise left behind after removing the sky model) and restored image (residual image convolved with restoring beam) are the results of deconvolution.

$$I_{restored} = I_{skymodel} \circ PSF_{ideal} + I_{residual} \quad (2.24)$$

Approaches used for deconvolution include: Högbom's CLEAN algorithm (Image-Domain CLEAN), Clark's CLEAN algorithm (Gridded Visibility Domain CLEAN), Cotton-Schwab's CLEAN algorithm (Visibility Domain CLEAN), Maximum Entropy Method (MEM), Non-negative least-squares (NNLS) algorithm and many others. Some of the various CLEAN algorithms are discussed next in the sections below:

2.5.1.1.1 Högbom's Algorithm: This algorithm is widely used and was introduced by Jan Högbom in 1974. This deconvolving algorithm is used particularly in the image domain [1].

The pseudocode of Högbom's CLEAN Algorithm and a brief summary of it is given below [1; 4]:

1. A copy is made of the dirty image $I^D(l, m)$ to be cleaned which is called the residual image $I^{res}(l, m)$.
2. Find the position of the brightest pixel in the residual image.
3. At the maximum position, subtract from the residual image the dirty beam multiplied by the maximum strength f_{max} and a gain factor γ which is referred to as *loop gain*.

input: $I^D(l, m), PSF(l, m), \gamma, f_{thresh}, N$
initialize: $S^{model} \leftarrow \{\}, I^{res} \leftarrow I^D, i \leftarrow 0$
while any($I^{res} > f_{thresh}$) or $i \leq N$ **do**
 $l_{max}, m_{max} \leftarrow \underset{l, m}{argmax} I^{res}(l, m)$
 $f_{max} \leftarrow I^D(l_{max}, m_{max})$
 $I^{res} \leftarrow I^{res} - \gamma \cdot f_{max} \cdot PSF(l + l_{max}, m + m_{max})$
 $S^{model} \leftarrow S^{model} + \{l_{max}, m_{max} : \gamma \cdot f_{max}\}$
 $i \leftarrow i + 1$
end while
output: S^{model}, I^{res}

4. Go back to step 2 unless the intensity of all remaining pixels are below some user specified threshold or when the number of iterations reach the limit specified by the user.
5. The accumulated point source model is convolved with a clean beam which is a 2D Gaussian fitted to the central lobe of the dirty beam.
6. The residuals of the dirty image is added to the CLEAN image to form the restored image.

2.5.1.1.2 Clark's Algorithm: Clark introduced this algorithm in 1980. It is based on the subtraction of point source responses in the (u, v) plane. This algorithm has two cycles known as the minor cycle and the major cycle. The minor cycle uses a beam patch including the main beam and the major sidelobes to identify the components to be removed by doing approximate subtractions. In the major cycle, the point source models that were identified in the minor cycle are subtracted without approximation in the (u, v) plane. These point source models can be transformed using Fourier Transforms, multiplied by the inverse transform of the beam, transformed back and subtracted from the dirty image. The minor and major cycles are repeated until a stop condition is reached [1].

2.5.1.1.3 Cotton-Schwab's Algorithm: Another algorithm used is the Cotton-Schwab Algorithm which is a variant of the Clark Algorithm. The major cycle subtracts CLEAN components from the ungridded visibility data which removes aliasing noise and gridding errors [1]. This algorithm makes use of the gridded/de-gridded functions and can be more expensive but generates better results.

2.5.1.2 Comparison of Algorithms

The Högbom's algorithm (image-domain) and Clark's (gridded visibility-domain) are easy to use but accuracy in PSF removal is limited e.g. w-term effects. Cotton-Schwab's (visibility-domain) is preferable because it performs accurate subtraction of sky model but it is computationally costly [4].

2.5.2 Calibration

In radio interferometry, there are propagation phenomena that introduce errors into signals measured by a radio interferometer, which in turn affects the visibilities. The errors are sometimes caused by instrumental factors such as the antenna gains and positions, and at other times by environmental factors such as the atmosphere. Removal of these errors is known as calibration [1]. Calibration procedures allow antenna gains and phases to be determined during usual operation of the radio telescope. Consider the following equation:

$$\|\mathbf{r}\| = \|\mathbf{d} - \mathbf{m}\| = \sqrt{\sum_{i=1}^N (d_i - m_i)^2} \quad (2.25)$$

When given a model and some data, a set of parameters is determined. This set of parameters minimizes the difference between the model and data. The model vector and data vector is represented by \mathbf{m} and \mathbf{d} respectively. The Euclidean vector norm of their difference is minimized and the residual vector which is a measure of the difference between the predicted values by the model and the observed values is represented by \mathbf{r} . Also, \mathbf{m} is a parameterized function with input parameters such as (x_1, x_2, x_3, \dots) that forms the parameter vector \mathbf{x} . Minimizing the above equation is known as **Least Squares Minimization**. Using this approach, one can find the antenna gains which minimizes the difference between the observed and the predicted visibilities.

Calibration is divided into three generations. They are: 1GC calibration, 2GC calibration and 3GC calibration.

2.5.2.1 1GC Calibration

1GC calibration is the oldest form of calibration i.e. first generation calibration. This type of calibration is carried out using calibrator observations. These are observations of sources with parameters that have been well characterized, i.e their shape, flux and spectral response are well known. The calibrator observation is then used to obtain calibration solutions which is applied in order to correct the target field observation.

An observing strategy includes intermittent calibrator scans of a calibrator field which is known. The obtained calibration solutions can then be interpolated onto scans of the target field [21]. This procedure is effective when

removing large-scale errors in visibilities and is performed by using the least-squares approach [4].

2.5.2.2 2GC Calibration

2GC calibration also known as second generation calibration was introduced in the 1980s. It involves using the target field itself to calibrate the observation. [22]. 2GC calibration is also known as self-calibration.

The self-calibration framework is an iterative approach which helps reduce errors between predicted visibilities that are corrupted by an instrumental model (the free parameters) and observed visibilities in a *least squares* sense during each iteration. An initial sky model for *selfcal* (self-calibration) can be obtained by imaging visibilities that have been corrected by the interpolated calibrator solutions.

Self-calibration switches between the image domain where deconvolution is done and the visibility domain where calibration takes place. The self-calibration framework is depicted in Figure 2.12.

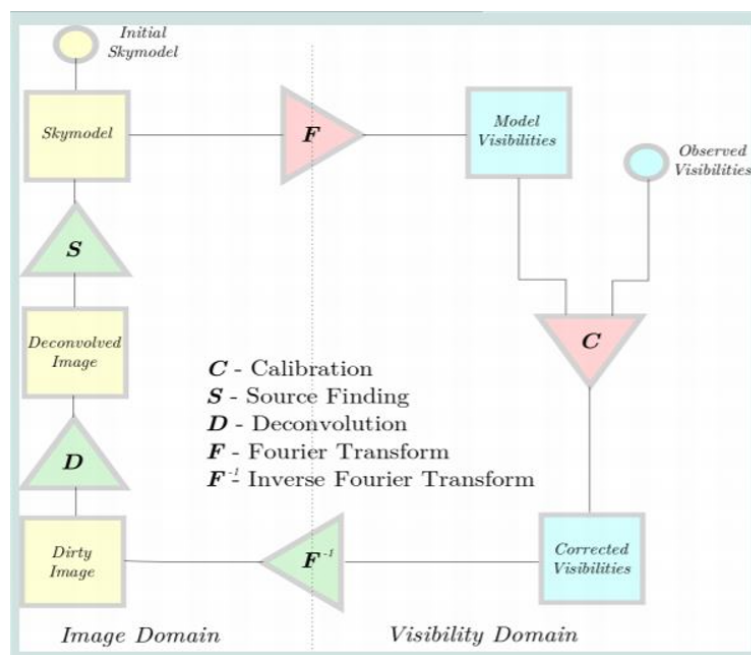


Figure 2.12: Self Calibration Framework [4]

The steps used in self-calibration are as follows [4; 1]:

1. Create an incomplete initial sky-model of a chosen field
2. The initial sky-model is then used to calibrate the observed visibilities.
3. The resulting image is deconvolved.
4. Run a source finder on the deconvolved image to create a more accurate sky model.
5. Go back to step 2, or stop the process if the target dynamic range has been reached.

2.5.2.3 3GC Calibration

3GC calibration (is known as the third generation calibration) is the contemporary way of calibrating. It is primarily used to remove direction-dependent effects (DDEs) eg. primary beam or the sensitivity pattern of antennas. These effects are non-uniform across the field of view, hence, they have to be removed in order to fully exploit the capabilities of telescopes such as the MeerKAT and the SKA [22; 4]. Procedures used in this form of calibration are add-ons of the 2GC self-calibration.

Techniques used in 3GC are grouped into two categories: physics-based and heuristic-only techniques. In physics based technique, the underlying physical phenomenon is known. Based on this underlying phenomenon, a parametrized model is constructed and the parameters are solved for. The results obtained are then used to correct the observed visibilities. An example of this approach is the Kalman filter [23]. The heuristic-only technique on the other hand deals with unknown underlying physical phenomenon. A number of free-parameters are solved for and are used to optimize [4]. An example of this approach is differential gains [24].

Chapter 3

Radio Frequency Interference

Section 3.1 of this chapter discusses interference, its effects on signals and the various divisions of radio frequencies. Following this, the sources of RFI which includes satellites are presented in Section 3.2. Some ways by which RFI can be detected and monitored, as well as some approaches from literature that are used in eliminating or reducing the effects are discussed in Section 3.3 and Section 3.4 respectively. An overview of some simulators that can be used to detect RFI are presented in Section 3.5. The generation of noise and RFI that was injected into the simulator presented in this thesis is as well discussed in Section 3.6.

3.1 Introduction

RFI is any undesirable signal that is present in data sets collected by receiving systems. These undesirable signals are as a result of radio waves that are emitted from “noisy” radio transmitters [25]. It is essential that radio astronomy observations be made without its data sets being contaminated by interfering signals.

Radio frequencies are divided into bands assigned for different purposes. Thus, a spectrum is allocated for use in radio astronomy in order to prevent interferences from transmissions by services using other bands. In the early years, the bands of radio astronomy systems were in the MHz range but as time went by, it was increased to above $\sim 100\text{GHz}$. Some bandwidth allocations were made for radio lines too. An example is the Hydrogen (H1) radio line that was assigned the 1420-1427 MHz band [1].

3.1.1 Interference

Interference occurs in many forms. Some are emissions from electrical and industrial appliances. Others are unwanted signals from Global System for

Mobile Communications (GSM) that originate from towers and cellular phones.

Radio observatories such as SKA, MeerKAT etc developed techniques to minimize the detrimental effects that RFI has on the observations they make. Measures taken to reduce RFI effects include the building of antennas at secluded places with no human movement or activities taking place. Moreover, they also make use of natural shields to protect the antennas from RFI [1].

3.1.2 RFI Signals

RFI signals are stronger in nature than the signals arriving on Earth from astronomical objects. These RFI signals can render data useless because they are capable of swamping the data collected by the receivers of radio telescopes [25].

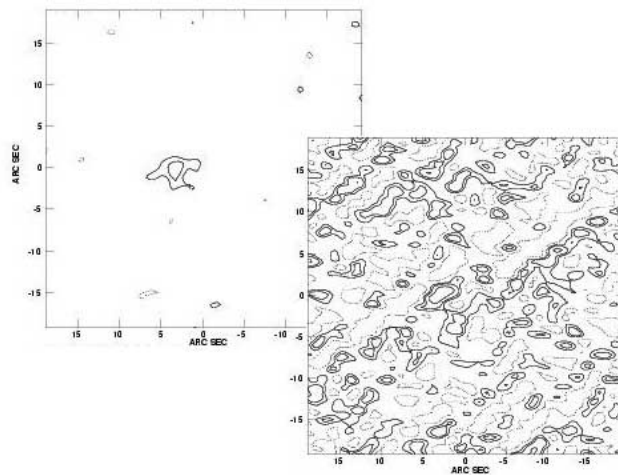


Figure 3.1: On the left is an image of a radio galaxy. This observation was made using the Very Large Array(VLA). On the right is the image of the same radio galaxy but this time around, a satellite passed within 25 degrees of the aforementioned radio galaxy. This image clearly shows that the signal from the satellite has completely overpowered the signal from the radio galaxy to such an extent that the original source is no longer visible [25]

3.1.3 Radio Frequency Spectrum

The radio frequency spectrum regulated by the International Telecommunication Union (ITU) describes the various divisions and allocations of radio frequencies for different uses. This is shown in Figure 3.2. It ranges from

the Very Low Frequency (VLF) band which starts from 3 kilohertz (kHz) to the Extremely High Frequency (EHF) band, which ends at 300 gigahertz (GHz). For instance, the frequency range for satellite communication is different from that of mobile phones. These radio frequencies are in terms of wavelength, which is the ratio between the speed of light in a vacuum and the radio frequency. Interference thereby occurs when two radio links share the same frequency band and are close to each other.

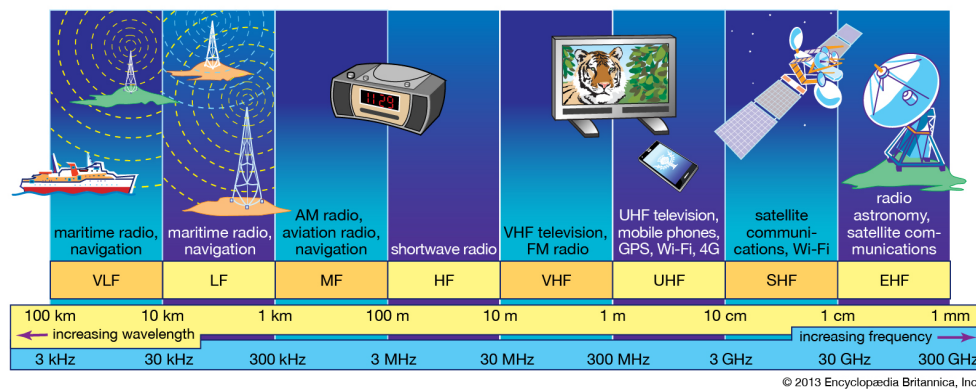


Figure 3.2: Radio Frequency Spectrum¹ showing the various radio frequency bands: VLF (Very Low Frequency) band, LF (Low Frequency) band, MF (Medium Frequency) band, HF (High Frequency) band, VHF (Very High Frequency) band, UHF (Ultra High Frequency) band, SHF (Super High Frequency) band and EHF (Extremely High Frequency) band.

3.2 RFI Sources

Table 3.1 shows the various frequency ranges of major sources of RFI [6]. Some details of each of these sources are discussed and are also shown in Figure 3.3 and Figure 3.4.

3.2.1 Satellites

Satellites are objects that rotate around larger objects. There are two types of satellites, namely: natural satellites eg. the moon and artificial satellites eg. Sputnik. The first artificial satellite was Sputnik which was launched by the Soviets on October 4, 1957 [26]. Subsequently, other satellites were launched; each one differing in size and being designed to perform a specific task. An example is the weather satellites which are used for weather forecasts.

Transmissions from satellites that are in orbit around the Earth are particularly troublesome for radio astronomers since those transmitters are located

RFI Source	Frequency Range (MHz)
GSM (Global System for Mobile Communication)	900 - 915 MHz uplink 925 - 960 MHz downlink
Aircraft transponders	Multiple <1 MHz bandwidth intermittent signals between 962 and 1213 MHz
GPS (Global Positioning System)	L1: 1565 - 1585 MHz L2: 1217 - 1237 MHz L3: 1375 - 1387 MHz L5: 1166 - 1186 MHz
GLONASS (Global Orbiting Navigation Satellite System)	L1: 1592 - 1610 MHz L2: 1242 - 1249 MHz
Galileo	1191 - 1217 MHz, 1260 - 1300 MHz
Inmarsat	1526 - 1554 MHz
Iridium	1616 - 1626 MHz

Table 3.1: RFI corrupted regions of the L-band range (900MHz - 1670MHz) [6]

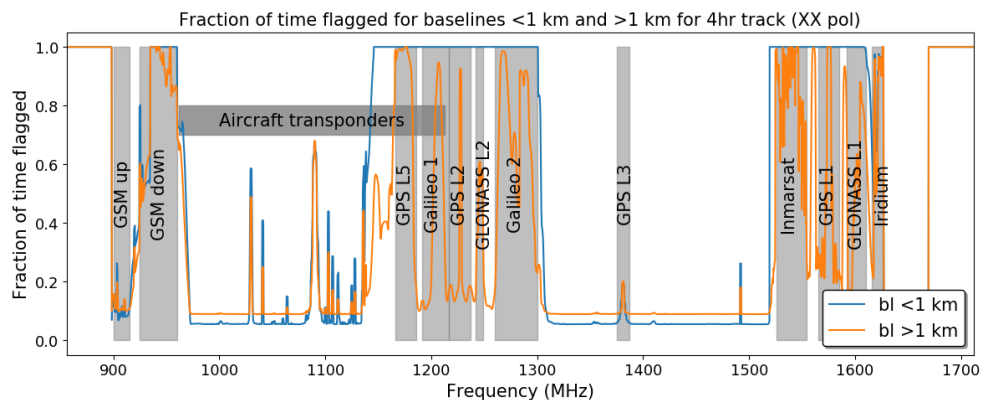


Figure 3.3: Figure showing the frequency channels in the MeerKAT L-band which are being corrupted by known RFI sources. The blue line represents the average across baselines shorter than 1km and the orange line also represents the average of baselines longer than 1km. The grey boxes are bands assigned to different RFI sources [6].

up above where radio telescopes are also pointing at in order to make observations [25]. Technology can be utilized to drastically reduce RFI interference.

3.2.1.1 GNSS Satellites

Global Navigation Satellite Systems (GNSS) refers to a constellation of satellites that provide positioning, navigation and timing (PNT) data to GNSS

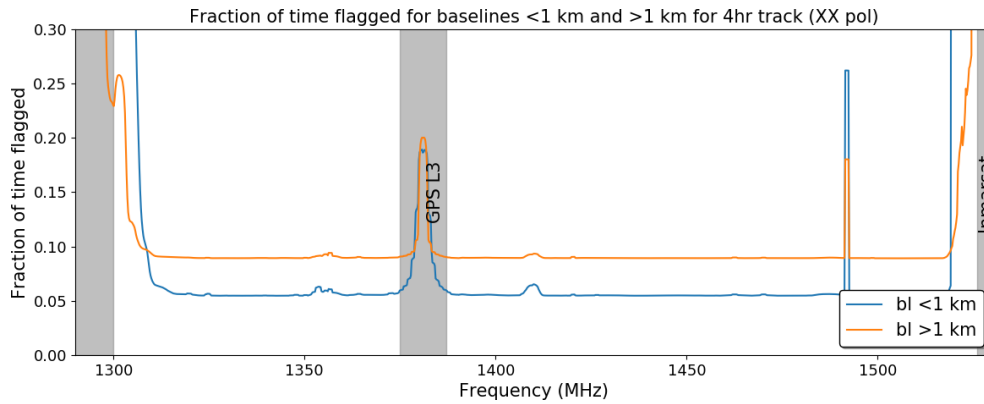


Figure 3.4: A zoomed in version of Figure 3.3, which is displaying the quietest portion of the L-band [6]

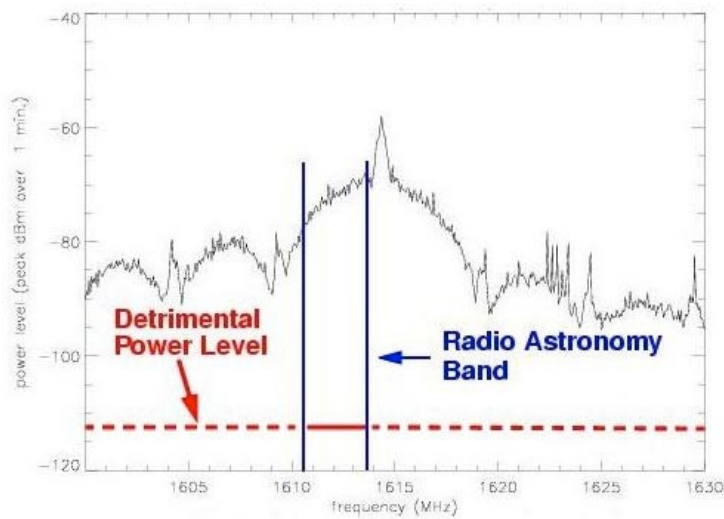


Figure 3.5: The signal represented in black is from an orbiting Earth Satellite which ignored regulations by an international agreement. The regulations are to prevent spillover from the satellites into the frequency ranges used by radio telescopes [25]

receivers. These GNSS receivers then use the PNT data to find locations in space. GNSS provide global services and are used in various forms of transportation eg. maritime, road and rail. The GNSS systems have two global systems that are up and running. They are the United States GPS and the Russian Federation's GLONASS. The following are the developing global and regional systems: China's BeiDou Navigation Satellite System (BDS), Europe's Galileo and India's Regional Navigation Satellite System (IRNSS) and

Japan's Quasi-Zenith Satellite System (QZSS) [27].

3.2.1.2 Telecommunication Satellites

Telecommunication satellites are satellites that allow long-distance telephone calls to be made. Also these satellites make live television broadcast from across the world possible [26].

3.3 Detection of Interference

Detecting of RFI is done by inspecting or monitoring the output of an observation. However, weak signals are hard to detect and when this happens, the channel bandwidths are compared to the bandwidth of an interfering transmitter. Unfortunately, this process cannot guarantee a 100 percent indication of RFI [1]. RFI can be detected in various ways. Some approaches used to detect interference are [1]:

- The operation of receivers used for monitoring; with antennas pointed towards interference sources.
- Inspection of the statistics of the output data of receivers.
- Utilizing multichannel receivers of high-resolution to detect interference from different channels.

3.3.1 RFI Monitoring

One way that detection and monitoring can be done is discussed in [5]. This document talks about an RFI monitoring system that was built for MeerKAT which utilized the Real Time Transient Analyser (RATTY). RATTY consists of an antenna that gathers electromagnetic radiation, a Reconfigurable Open Architecture Computing Hardware (ROACH) board that carries out signal processing in addition to analog-to-digital conversions and a computer for data storage and control. The system design is shown in Figure 3.6. The monitor automatically collects data from the RATTY system, the monitor then provides information of the RF environment, runs an automatic RFI detection on the obtained radio spectra and stores them on a regular basis. The statistics of the radio spectra obtained help infer the radio frequency environment and can be utilized to spot the incoming RFI sources. The spectral data, RFI detections and statistical data can be acquired via the web either by downloading the unprocessed data or by using visual tools.

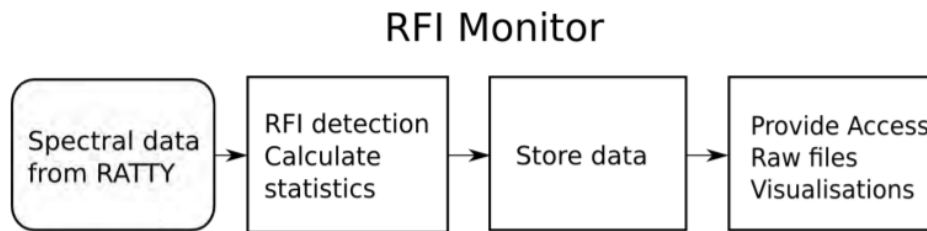


Figure 3.6: Block diagram of how the RFI Monitor works [5]

3.4 Elimination of Interference

RFI comes in different forms, hence there is no single algorithm that will work for all contamination instances. Therefore, this section discusses the various techniques used to eliminate or mitigate RFI.

Mitigation of RFI can be done in two phases: during the pre-correlation phase and during the post-correlation phase. Amongst the two, the pre-correlation phase tends to be more effective since the observational data remains untouched by the approach itself [28]. Also, mitigation of RFI in terms of historical approaches is grouped into three classes: linear methods, threshold-based algorithms eg. SumThreshold [28] and employing supervised machine learning approaches [29].

Other approaches that are commonly used to deal with RFI include: removing the corrupted data and cancelling the interference via spatial filtering; where a null is created in the reception pattern of an antenna towards the direction of an interfering object. Nulling exists in different forms, namely: spatial nulling, deterministic nulling and adaptive nulling. In spatial nulling, a null is formed by combining signals using a set of antennas in the direction of the interfering source. In another type of nulling known as deterministic nulling, the direction of the interfering source is known and a null is created in that same direction by measuring the received signals. Adaptive nulling also involves the removal of the consequences of a signal causing interference by positioning a null in the reception pattern of a group of antennas facing the direction of the interfering signal [1].

From all the aforementioned approaches, the threshold-based algorithms and the supervised machine learning approaches will be discussed since they are the most popular. How they eliminate or mitigate interference is discussed in the sections below. Whether a technique is pre-correlation or post-correlation is also distinguished.

3.4.1 Pre-correlation Phase

This section discusses an approach that takes place in the pre-correlation phase.

The preferable way of reducing interference is by cancellation instead of deleting the corrupted part [30]. For the cancellation of interference approach, adaptive interference cancelers are used which comes in two forms: analog adaptive interference and the digital adaptive interference cancelers. The analog adaptive interference cancelers were introduced in the 1960s for the cancellation of issues related to radio communications and radar but unfortunately, this system lacked certain traits such as versatility and dynamic range. The digital adaptive interference cancelers were then introduced in the mid-90s to minimize undesirable noise in audio systems. Cancellation does not only require detection of the interference signal but requires a precise estimation of the signal which is then applied to the removal procedure. In [30], excision of interference using real-time adaptive cancellation is implemented. An adaptive canceler was built comprising of two receivers namely: the primary channel and a different reference channel. In the case of RFI mitigation, the primary channel would be called the astronomy antenna which receives signal contaminated by RFI. The reference channel is pointed in the direction of the interferer which receives RFI only; the signal is then first processed using a digital adaptive filter and is subtracted from the primary channel input. Regulation of the weighting coefficients of the digital adaptive filter is done to reduce the RFI effect at the system output.

3.4.2 Post-correlation Phase

This section discusses the various approaches that take place in the post-correlation phase.

In this document [28], different post-correlation radio frequency interference classification methods were described and compared. These methods were tested on artificial data and on data that was observed from the Westerbork Synthesis Radio Telescope (WSRT) in the Low Frequency Array (LOFAR) frequency range. The following were taken into consideration during the selection of the RFI mitigation plan: the true or false-positive ratio of the classification of RFI, the speed of the algorithm as well as its detection and the effects of reducing RFI on noise. Thresholding is an effective technique that is mostly used to eliminate strong RFI. The threshold value is decided on or set with respect to the mode distribution variables per baseline. Values that are beyond a certain limit from the mean or median are identified as RFI and hence, flagged in order to obtain good data. The methods that were investigated are: the cumulative sum method (CUSUM), combinatorial thresholding, VarThreshold parameters and the SumThreshold method. It was concluded that the

best amongst all of the methods that were investigated is the SumThreshold method. The SumThreshold is an iterative method which evaluates signals by performing a surface fit in the time-frequency plane. This method is quick, powerful and can be carried out without using a data model.

RFI flagging is done to reduce RFI in signals to make data robust. If the RFI is determined to be above some critical threshold, the data sample is flagged. Almost all samples will have some amount of RFI in them. Manual flagging of data entails observing the data physically, checking if it is corrupted and then flagging it. This process is a tiresome one and requires a lot of effort. New methods were thereby introduced to flag data automatically since large data sets are being observed. Astronomical software packages such as CASA (Common Astronomy Software Applications) has helped radio astronomers in dealing with RFI when performing data reduction or flagging.

AOFlagger is a flagging software that is used for the LOFAR radio telescope, WRST etc to eliminate RFI in astronomical data and to examine measurement data. The software has a graphical interface (rfgui) that aids in viewing data in various ways. Its code was written in C++ [31]. Debatably, it is the best flagger in relation to speed and precision. As time went by, a new type of AOFlagger was invented which uses the MeerKAT SDP pipeline. This RFI flagger, i.e. the new AOFlagger is known as the SDP (Science Data Processing) Flagger.

In [32], flagging of RFI was done using two different methods. The first method uses visibilities that are redundant to spot contaminated data. The second method uses a strategy that was invented to identify weak RFI signals in the time-channel visibility plane of baselines. When powerful sources are subtracted improperly, ripples occur. These ripples minimize the ability to detect RFI in the residual visibilities. Some of the reasons why this occur is because of some direction dependent calibration errors and asymmetric primary beam. The ripples are thereby removed by cutting out the corresponding peaks in the related Fourier plane.

Deep learning can also be used to flag RFI as discussed in [29]. A convolutional neural network (CNN) contains convolutional layers. U-Net is a type of CNN. Its architecture shrinks and then enlarges images to their initial state by utilizing deconvolutional layers. Another CNN known as the ResNet Convolutional Neural Network (R-Net) is proposed by [29] for RFI detection. The ResNet architecture utilizes skip connections which makes it possible to train very deep networks. R-Net is an example of a supervised learning approach; the algorithm learns via existing RFI examples. The R-Net algorithm was tested on single-dish as well as RFI simulations and was trained on the magnitudes of the complex visibilities. When using supervised deep learning for RFI, large amounts of data consisting of many baselines have to be labelled for classification to take place, which is quite challenging. It is highly recommended that deep learning be performed on simulations when doing flag-

ging. In [29], the R-Net algorithm was compared to the deep U-Net algorithm and the standard SDP Flagger. The following metrics were considered: the F1-score metrics, the AUC (Area Under the ROC (Receiver Operating Characteristic) curve) metrics etc. In [33], a few conventional ML algorithms were used to detect RFI in KAT-7 data. These algorithms include Naive Bayes, k -Nearest Neighbour and the Random Forest Classifier (RFC). Data used was randomly divided according to the ratio of 70:30, that is to say 70% of the splitted data was used for cross-validation in order to train the algorithms and 30% for testing. It was finally concluded that the RFC detected RFI the best with an AUC of 98% and a recall of 91%.

Amongst the recent algorithms such as the U-Net and the standard SDP Flagger which was applied in the MeerKAT data reduction pipeline, the results proved that ResNet was most efficient. Some other comparisons showed that the execution time needed for R-Net is a bit less than the other algorithms; this is because the convolutional layers of the R-Net's architecture are of the same sizes which enables parallel handling of operations.

3.5 RFI Simulators

This section discusses the various RFI simulators that have been created and exist within the literature. Most of these simulators are deductive in nature. The simulator created in the work presented in this thesis is inductive. The approach used is discussed in the next section. The difference between an inductive and a deductive simulator is discussed in detail in Section 5.2 and Section 1.1 highlights the difference.

3.5.1 HIDE and SEEK package Simulator

Two data packages known as the HIDE (HI Data Emulator) and SEEK (Signal Extraction and Emission Kartographer) are presented in [12]. These data packages are used to simulate radio survey data. They can either be used hand in hand or separately. HIDE gathers astronomical signals in a single-dish radio telescope and simulates RFI signals. The single-dish radio telescope data is then processed by SEEK. Again, SEEK identifies RFI signals and masks them, utilizes flux calibration and focuses to restore the astronomical signal. The documentation² on these two packages can be found on GitHub under the GPLv3 license.

²<http://hideseek.phys.ethz.ch/>

3.5.2 RFSim

In [11], a simulator is proposed for radio interferometers and in particular for the MeerKAT telescope. The simulator utilized a physical RFI model (GPS satellite orbits) and generated visibilities which looked exactly like visibilities from real observations. This data set was used to develop an advanced flagging and mitigation technique.

3.5.3 HERA_sim

HERA_sim is a simulation package that is used to simulate interferometric arrays similar to Hydrogen Epoch of Reionization Array (HERA) which is a redundant array. HERA_sim has a highly configurable interface and as such it can be used to add several attributes to visibilities that exist [13].

3.6 Generation of Noise and RFI

Thermal noise that is injected into an interferometric system is proportional to the bandwidth and integration time [34]. Formally, it is expressed as follows:

$$\sigma \propto \frac{T_{sys}}{\sqrt{\Delta\nu\tau}}, \quad (3.1)$$

where σ is the standard deviation associated with the noise, T_{sys} is the system temperature, $\Delta\nu$ is the observational bandwidth and τ is the integration time for a visibility [34]. This equation can be referred to as the general noise model of an interferometric observation. T_{sys} is normally not changed because its associated to the instrument and is a constant as well. The integration time and the bandwidth however are not fixed and are changeable, which thereby creates a certain SNR (signal-noise-ratio). The parameter σ determines the power in the noise, thereby determining the SNR value of the contaminated signal. It is not necessary to know the integration time and bandwidth but rather, an SNR value can be chosen which is equivalent to choosing a value for either the integration time or/and the bandwidth. Simply put, integration time and bandwidth translates into a certain SNR value.

The shorthand, $\langle \cdot \rangle$ will be used to refer to averaging over frequency ν , time t and baseline pq . The equation below defines SNR:

$$\text{SNR} = 10 \log \left(\frac{\langle D \odot \bar{D} \rangle_{\nu,t,pq}}{\langle N \odot \bar{N} \rangle_{\nu,t,pq}} \right) \quad (3.2)$$

where D represents a noise-free visibility cube indexed by time t , frequency ν and baseline pq and \bar{D} is its conjugate. The symbol N is a noise cube and

$\langle \odot \rangle$ denotes the hadamard product. Equation 3.2 computes the power (P_{signal}) in D , divides it by the power in N (P_{noise}) and the resulting ratio is then converted to dB.

It now follows from Equation 3.2 that if given a certain SNR value and a data cube D (from which P_{signal} can be computed), then the power that the noise should have can be computed via

$$P_{\text{noise}} = P_{\text{signal}} \times 10^{-\frac{\text{SNR}}{10}} \quad (3.3)$$

How is N now generated so that it has the required amount of power P_{noise} ? It is a well established fact, that if the real and the imaginary components of N are drawn from a zero mean normal distribution with a standard deviation³ $\sigma = \sqrt{\frac{P_{\text{noise}}}{2}}$, then the power of N will be P_{noise} .

This same approach can be used to add thermal noise and to add RFI. Given two SNR values: one for the thermal noise level and one for the RFI noise level, two noise matrices are generated so that they have the correct power level as discussed earlier in this section. The two noise matrices are then added to the noiseless visibility cube. Note that P_{signal} is the same value whether the noise level is computed to add for the thermal noise as well as the RFI. Formula to add noise to visibilities of a certain SNR threshold where D now becomes the visibility with noise added:

$$\tilde{D} = D + N \quad (3.4)$$

The python code for the formulas to generate noise and RFI can be located in the `noise_visibility.ipynb` and `RFI_visibilities.ipynb` files respectively in the github repository.

³<https://dsp.stackexchange.com/questions/16216/adding-white-noise-to-complex-signal-complex-envelope>

Chapter 4

Machine Learning

In this chapter, Section 4.1 presents an overview of ML, Section 4.2 outlines some of the applications that make use of ML and the steps involved in setting up an ML framework. Section 4.3 discusses three of the machine learning methods, namely: supervised learning, unsupervised learning and semi-supervised learning. Section 4.4 presents the identities used in classification and Section 4.5 presents the two types of approaches used to create models: generative and discriminative approaches. Classifiers belonging to the generative grouping and the discriminative grouping are discussed in Section 4.6 and Section 4.7 respectively. The classifiers that do not belong to the aforementioned groupings but rather make use of the unsupervised learning are discussed in Section 4.8. An overview of the confusion matrix and the metric used in this work is given in Section 4.9. Lastly, the *Iris* dataset and results obtained when the four classifiers (Naive Bayes, Logistic Regression, k -means and GMM) were applied on on the *Iris* dataset is presented in Section 4.9.

4.1 Introduction

Machine learning is a sub-field of Artificial Intelligence (AI). ML is the collective name used to identify a set of algorithms that are capable of learning from data. Furthermore, after learning has taken place, these algorithms can then apply what they have learned to perform a specific task [35]. As an example, a piece of software that can accurately predict outcomes makes use of ML to do so.

The learning process involves finding patterns in data; these patterns are then used to predict future outcomes. Features of a dataset are referred to as attributes of the dataset. The number of features are also referred to as the dimension of the dataset.

4.2 Machine Learning Applications

Machine learning applications make use of statistics to discover patterns in a given dataset [35]. These applications use input data to predict an outcome. Examples of services that make use of machine learning¹ are:

- Digital Assistants eg. Google Assistant, Apple Siri
- Recommendations: Google, Facebook, Twitter
- Online advertisements
- Chatbots
- Fraud detection
- Cybersecurity
- Medical image analysis
- Self-driving cars
- Email spam filtering

For instance, Siri, the voice assistant application collects input data in the form of audible words. It then finds words in a dictionary that best match the recorded words. Other examples are websites that make recommendations based on what one watches, searches or listens to.

It can be concluded that machine learning algorithms are trained to discover patterns in data which can then be used to make decisions.

4.2.1 Steps Involved in Machine Learning

The steps involved in setting up an ML framework are [36; 37]:

1. Data gathering: This is the first step in machine learning. It involves gathering large amounts of high quality data, which in turn determines the accuracy of the predictive model.
2. Data preparation: After data is collected, it is prepared for training. A training dataset refers to the dataset which is required by an ML algorithm in order for learning to take place. It can be either labelled or unlabelled data. The order of data is randomized and cleaned in order to correct errors in the data, to get rid of duplicates and other data alterations. Data is then visualized to see if there are any imbalances and splitted into training and testing sets.

¹<https://www.ibm.com/cloud/learn/machine-learning>

3. Selecting a model: Choosing a model is dependent on the type of training data, the size of dataset and the expected outcome of the task at hand.
4. Training the model: This step entails using an algorithm to learn model parameters, checking the differences between the known output and the expected outcome and minimizing the aforementioned difference by adjusting the model parameters. These steps are repeated multiple times until the correct results are attained from the algorithm.
5. Evaluating the model: After the model is trained, it is tested to evaluate its performance. Depending on the problem that needs to be solved, the model created is tested with new data. This thereby increases the model's precision and efficiency.
6. Parameter tuning: This is also known as hyperparameter tuning. It involves tuning model parameters to improve performance. Examples of these parameters are learning rate, number of training steps, initialization values etc. The values of hyperparameters are determined by a validation set.
7. Making predictions: The test sets are then used to test the model to determine its performance in the real world. It can then, be used for prediction.

4.3 Machine Learning Methods

There are two major tasks expected from ML algorithms; regression and classification. These two algorithms fall under the supervised learning methods. Classification entails finding a model that divides input data into several classes and are used to identify discrete values. Regression on the other hand entails finding a model that identifies a continuous value depending on its input variables [38]. The difference between the two is that, the classification algorithm locates a mapping function in order to map the input (x) onto the discontinuous output (y) while the regression algorithm does the same but rather maps the input (x) onto the continuous output (y) [38]. The primarily focus in this thesis is the classification task.

Types of machine learning methods can be classified into three primary categories, namely: supervised learning, unsupervised learning and semi-supervised learning.

4.3.1 Supervised Learning

Supervised learning is a type of machine learning which uses labelled data for training [39; 2; 37]. A learning algorithm from this category is able to predict

future occurrences by applying what has been learned previously to new data using labelled examples. In other words, the algorithm receives inputs together with the desired outputs and then learns by comparing its generated output with the expected output. This comparison procedure helps in discovering errors and can in turn be used to improve the accuracy of the model.

Some techniques used in supervised learning include: Linear Regression, Logistic Regression, Naive Bayes and Random Forest [39]. One disadvantage of this method is that the production of labelled data is costly.

4.3.2 Unsupervised Learning

Unsupervised learning deals with data with no labels. This suite of algorithms are unable to tell the intended output but rather analyzes data and draws conclusions by finding hidden patterns from unlabelled data [2; 37]. Clustering software thereby use whichever patterns they find [39]. Examples of techniques used in unsupervised learning are k -means clustering and GMM.

4.3.3 Semi-supervised Learning

A semi-supervised machine learning algorithm uses both labelled and unlabelled data for training [40]. It mostly uses a smaller portion of labelled data for classification during training and a bigger portion of unlabelled data when extracting features. This algorithm is best used when labelled data is not enough for training and learning to occur.

4.4 Basic Overview of some Statistical Relationships

This section presents the relationship between variables that will be used from Section 4.5 to Section 4.7 to perform some derivations.

Consider two events X and Y . $P(X, Y)$ is the probability of event X and Y occurring. $P(X|Y)$ is the conditional probability that event X will occur given that event Y occurred. Consider the following identities [41; 37]:

1. Product Rule

$$P(X, Y) = P(X|Y)P(Y) \quad (4.1)$$

2. Bayes Theorem

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \quad (4.2)$$

3. Marginalization

$$P(Y|X) = \frac{P(X|Y)P(Y)}{\int P(X|y)P(y)dy} \quad (4.3)$$

4. Conditional Independence

$$P(X, Y|Z) = P(X|Y, Z)P(Y|Z) = P(X|Z)P(Y|Z) \quad (4.4)$$

4.5 Classification

This section closely follows the content presented in [41; 2; 37]. Here, the two different types of probabilistic models are discussed.

As has been mentioned already, classification is one of the major tasks for which ML is used. This section discusses the classification algorithms that are used in this work into details.

Let $D = \{\mathbf{X}, \mathbf{y}\}$. D is referred to as a dataset. The rows \mathbf{x}_i of \mathbf{X} represent observation vectors with corresponding label y_i , where $i \in \{1, \dots, N\}$. The p^{th} feature of \mathbf{x} is denoted by x_p . In other words, the observations can belong to one of k classes. The class label $y_i = j$ if \mathbf{x}_i belongs to class C_j , where $j \in \{1, \dots, k\}$.

Classification entails assigning an observation vector \mathbf{x} to one of k classes by first calculating the class probabilities $P(C_j|\mathbf{x})$. The observation vectors are assigned to the class with the highest probability:

$$C^* = \operatorname{argmax}_{C_j} P(C_j|\mathbf{x}) \quad (4.5)$$

A generative or a discriminative approach can be used to create the model $P(C_j|\mathbf{x})$. The generative approach employs Bayes Theorem. It follows from Bayes Theorem that:

$$P(C_j|\mathbf{x}) \propto P(\mathbf{x}|C_j)P(C_j), \quad (4.6)$$

where $P(C_j)$ is known as the class prior and $P(\mathbf{x}|C_j)$ is known as the class conditional density. To implement the generative approach, $P(\mathbf{x}|C_j)$ needs to be estimated. The class conditional density can be used to generate synthetic examples belonging to each class. Classifiers belonging to this grouping are called Probabilistic Generative Models (PGM). An example is Naive Bayes.

The discriminative approach dispenses with the class conditionals $P(\mathbf{x}|C_j)$; instead a predetermined model is chosen with which the posterior $P(C_j|\mathbf{x})$ is computed. Classifiers belonging to this grouping are called Probabilistic Discriminative Models (PDM). An example is the logistic regression classifier. Training data is used to differentiate between classes.

4.6 Probabilistic Generative Models (PGM)

In this section, the generative approach is discussed in more detail. Assume that \mathbf{x} is an observation vector and that it can belong to either C_1 or C_2 . It now follows from Bayes Theorem that:

$$\begin{aligned}
P(C_1|\mathbf{x}) &= \frac{p(\mathbf{x}|C_1)P(C_1)}{p(\mathbf{x}|C_1)P(C_1) + p(\mathbf{x}|C_2)P(C_2)} \\
&= \frac{1}{1 + \frac{p(\mathbf{x}|C_2)P(C_2)}{p(\mathbf{x}|C_1)P(C_1)}} \\
&= \frac{1}{1 + \exp(-a(\mathbf{x}))} \\
&= \sigma(a(\mathbf{x}))
\end{aligned} \tag{4.7}$$

In Equation 4.7, $a(\mathbf{x})$ denotes the log posterior odds, while σ denotes the logistic sigmoid function. The formal definition of $a(\mathbf{x})$ is:

$$a(\mathbf{x}) = \ln \frac{p(\mathbf{x}|C_1)P(C_1)}{p(\mathbf{x}|C_2)P(C_2)} \tag{4.8}$$

Moreover, the logistic sigmoid function is defined as:

$$\sigma(a) = \frac{1}{1 + \exp(-a)} \tag{4.9}$$

For k classes,

$$\begin{aligned}
P(C_n|\mathbf{x}) &= \frac{P(\mathbf{x}|C_n)P(C_n)}{\sum_{j=1}^k P(\mathbf{x}|C_j)P(C_j)} \\
&= \frac{\exp a_n(\mathbf{x})}{\sum_{j=1}^k \exp a_j(\mathbf{x})}
\end{aligned} \tag{4.10}$$

where

$$a_j(\mathbf{x}) = \ln p(\mathbf{x}|C_j) + \ln P(C_j) \tag{4.11}$$

It can easily be verified that $\sum_{j=1}^k P(C_j|\mathbf{x}) = 1$. The ratio of the exponentials in the numerator to the normalized sum of exponentials in the denominator in Equation 4.10 is called the softmax function. This function outputs posterior probabilities and provides a differentiable or smooth form of the max function. $P(C_n|\mathbf{x}) \approx 1$ for $j \neq n$ and $P(C_j|\mathbf{x}) \approx 0$ for $j \neq n$ if $a_n \gg a_j$.

To determine the posterior probability, the prior class probabilities $P(C_j)$ and class-conditional densities $p(\mathbf{x}|C_j)$ are required.

4.6.1 Shared Covariance

If the class-conditional densities are Gaussian, then:

$$P(\mathbf{x}|C_j) = \frac{1}{2\pi\sqrt{|\Sigma_j|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_j)^T \Sigma_j^{-1}(\mathbf{x} - \mu_j)\right) \tag{4.12}$$

where Σ is the covariance and μ is the mean. Now consider, once again a binary classification. If the two classes C_1 and C_2 share a covariance matrix, then $a(\mathbf{x})$ becomes linear, i.e Equation 4.7 simplifies and becomes:

$$P(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x} + w_0) \quad (4.13)$$

Under these conditions, Equation 4.8 now becomes:

$$a(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0 \quad (4.14)$$

where

$$\mathbf{w} = \Sigma^{-1}(\mu_1 - \mu_2) \quad (4.15)$$

and

$$w_0 = -\frac{1}{2}\mu_1^T \Sigma^{-1} \mu_1 + \frac{1}{2}\mu_2^T \Sigma^{-1} \mu_2 + \ln \frac{P(C_1)}{P(C_2)} \quad (4.16)$$

Take note of the following:

1. The bias term w_0 is dependent on the prior probabilities.
2. Due to the shared covariance, the quadratic terms in the Gaussians cancel out; resulting in a linear classifier (i.e the classifier has a linear decision boundary). To see why this is the case, consider the following argument. If $P(C_1|\mathbf{x}) > P(C_2|\mathbf{x})$ then \mathbf{x} is assigned to C_1 else it is assigned to C_2 . The decision boundary is, therefore, determined by $P(C_1|\mathbf{x}) = P(C_2|\mathbf{x}) = 1 - P(C_1|\mathbf{x})$ which is rewritten as $\sigma(\mathbf{w}^T \mathbf{x} + w_0) = 1 - \sigma(\mathbf{w}^T \mathbf{x} + w_0)$, or $\sigma(\mathbf{w}^T \mathbf{x} + w_0) = \frac{1}{2}$. The decision boundary then becomes $\mathbf{w}^T \mathbf{x} + w_0 = 0$.
3. The mean associated with each class needs to be estimated, as well as the shared covariance matrix if Equation (4.13) is to be implemented. For d -dimensional data this equates to $2d + \frac{1}{2}d(d + 1)$ parameters. If Equation (4.13) is inspected, \mathbf{w} and w_0 is estimated, which amounts to only needing to estimate $d + 1$ parameters instead. This is exactly what is effectively accomplished by rather following a discriminative approach.

Considering the k -classes scenario under the shared covariance assumption again, Equation 4.10 now becomes:

$$P(C_n|\mathbf{x}) = \frac{\exp a_n(\mathbf{x})}{\sum_{j=1}^k \exp a_j(\mathbf{x})} \quad (4.17)$$

where

$$a_j(\mathbf{x}) = \mathbf{w}_j^T \mathbf{x} + w_{j0} \quad (4.18)$$

$$\mathbf{w}_j = \sum_{i=1}^{-1} \mu_j$$

$$w_j^k = \frac{1}{2}\mu_j^T \sum_{i=1}^{-1} \mu_j + \ln P(C_j) \quad (4.19)$$

A shared covariance results in the creation of linear decision boundaries. On the other hand, if the shared covariance assumption is not adhered to, the decision boundaries becomes non-linear. This phenomenon is illustrated in Figure 4.1. How to estimate the parameters of $P(C_j|\mathbf{x})$ has already being shown under the shared covariance assumption. How then are the parameters of the class conditional densities and prior probabilities of a statistical model computed in general? This is the topic of the next section.

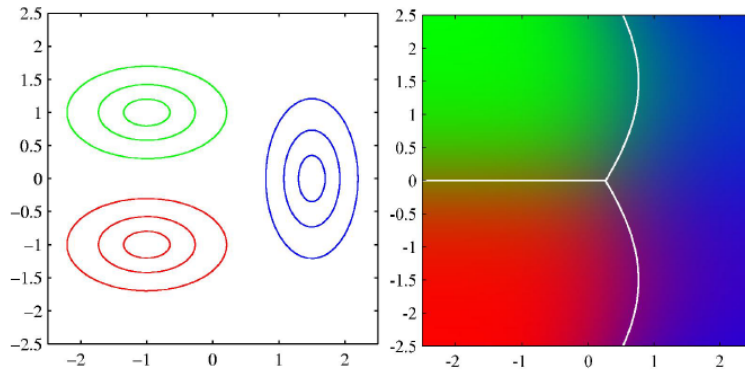


Figure 4.1: Image of a non-linear classifier showing its quadratic decision boundary [41]

4.6.2 Non-shared Covariance

Again let us consider the two-class case. The maximum likelihood estimates of the two means and the two covariance matrices are:

$$\boldsymbol{\mu}_1 = \frac{1}{N_1} \sum_{n=1}^N y_n \mathbf{x}_n \quad \text{and} \quad \boldsymbol{\mu}_2 = \frac{1}{N_2} \sum_{n=1}^N (1 - y_n) \mathbf{x}_n \quad (4.20)$$

and

$$\boldsymbol{\Sigma}_1 = \frac{1}{N_1} \sum_{n \in C_1} (\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^T \quad \text{and} \quad \boldsymbol{\Sigma}_2 = \frac{1}{N_2} \sum_{n \in C_2} (\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^T \quad (4.21)$$

where the sample sizes are represented by N_1 and N_2 , $y_n = 1$ if $\mathbf{x}_n \in C_1$ and $y_n = 0$ if $\mathbf{x}_n \in C_2$. Moreover, the maximum likelihood estimate of $P(C_1) = \pi$ and $P(C_2) = 1 - \pi$, where $\pi = \frac{N_1}{N}$.

Maximum Likelihood Estimation (MLE) can be defined as the statistical method of determining values for parameters of a model given a set of data [41; 40; 35].

4.6.3 Naive Bayes Classifier

Bayes' theorem which forms the basis of the Naive Bayes classifier was developed by Thomas Bayes (1702 to 1761) who was born in Hertfordshire². He studied Logic and Theology at the University of Edinburgh from the years 1719 to 1722. To obtain the Naive Bayes classifier, some naive independence assumptions need to be made between the observational dimensions [40; 37].

Assuming that the observation vector is conditionally independent, then $P(\mathbf{x}|C) = \prod_{n=1}^d P(x_n|C)$, where $P(\mathbf{x}|C)$ are class conditionals and x_n are the individual features of \mathbf{x} . Interestingly, this assumption forces the covariance matrix of $P(\mathbf{x}|C)$ to be diagonal. This is the definition of independence, which is why it is called "Naive". Bayes' theorem can now be used to calculate:

$$\begin{aligned} P(C_j|\mathbf{x}) &= \frac{P(C_j)P(\mathbf{x}|C_j)}{p(\mathbf{x})} \\ &= \frac{P(C_j) \prod_n p(x_n|C_j)}{\sum_i P(C_i) \prod_n p(x_n|C_i)}, \end{aligned} \quad (4.22)$$

to which class an observation is assigned to can, therefore, be computed using:

$$C^* = \operatorname{argmax}_{C_j} \frac{P(C_j) \prod_n p(x_n|C_j)}{\sum_i P(C_i) \prod_n p(x_n|C_i)} \quad (4.23)$$

Equation 4.22 can be simplified, since the denominator in Equation 4.23 depends on all C_j . The main difference now in comparison to the previous PGM models discussed earlier is that, here, the standard deviation and the mean associated with each class-conditional density of each dimension can be determined separately. In other words, in the previous PGM models all the entries of the covariance matrix had to be estimated, now only its diagonal entries need to be calculated. This leads to a massive reduction in the number of parameters that must be estimated. The maximum likelihood estimates of

²<https://holypython.com/nbc/naive-bayes-classifier-history/>

the aforementioned parameters are given by:

$$\begin{aligned}
 P(C_j) &= \frac{N_j}{N} \\
 \mu_{nj} &= \frac{1}{N_j} \sum x_{nj} \\
 \sigma_{nj}^2 &= \frac{1}{N_j} \sum (x_{nj} - \mu_{nj})^2
 \end{aligned}
 \tag{4.24}$$

where N_j represents the number of samples in class for $n = (1, \dots, d)$; $j = (1, \dots, k)$ and C_j ; summing over all the samples x_{nj} that belongs to class C_j .

4.7 Probabilistic Discriminative Models (PDM)

Classifiers utilizing a discriminative approach do not use class-conditionals; instead they use model weights. The model weights are chosen so that the likelihood of the data is maximized [41].

4.7.1 Logistic Regression Classifier

Logistic regression was introduced as a statistical model by Joseph Berkson who was a trained physicist, statistician and physician [42].

To model the probability of events occurring in statistics, a logistic model can be used. Consider an event; win (“1”) or loss (“0”). The logistic model determines which of the event has occurred or not which would be given a probability between “0” and “1”. Logistic regression can be used to classify the event by evaluating the logistic model parameters. A binary logistic model has a variable made of two values which are being labelled as “0” and “1”. The logarithm of the odds (log-odds) for the labelled value “1” represents the linear combination of one or multiple independent variables. These independent variables can be of a binary class or any real number. Logistic function is the name given to the function that transforms log-odds to probability. The log-odds scale is thereby measured in *logit* which is derived from **logistic unit** [42].

Again, in the binary classification problem, consider $D = \{\mathbf{X}, \mathbf{y}\}$, where as before \mathbf{X} represents observation matrix with corresponding label vector \mathbf{y} . Note that $y_i \in \{0, 1\}$. Equation 4.7 implies that if a linear decision boundary is chosen, then the following model choice naturally follows [2]:

$$P(C_1|\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{w}^T \mathbf{x}) \tag{4.25}$$

Here, \mathbf{w} contains w_0 and an extra one has been appended to the end of \mathbf{x} . The aim is now to find the weight vector \mathbf{w} which maximizes the likelihood of \mathbf{D} . To accomplish this, the joint distribution $P(D|\mathbf{w})$ has to be simplified first:

$$\begin{aligned} P(D|\mathbf{w}) &= P(\mathbf{X}, \mathbf{y}|\mathbf{w}) \\ &= P(\mathbf{y}|\mathbf{X}, \mathbf{w})P(\mathbf{X}|\mathbf{w}) \\ &= P(\mathbf{X}) \prod_{n=1}^N P(y_n|\mathbf{X}, \mathbf{w}) \\ &= P(\mathbf{X}) \prod_{n=1}^N P(y_n|\mathbf{x}_n, \mathbf{w}) \end{aligned} \quad (4.26)$$

where y_n is assumed to be conditionally independent given \mathbf{x}_n and \mathbf{X} which is independent of the weight vector \mathbf{w} . Since y_n follows a Bernoulli distribution [41; 2; 35]; it may be written:

$$P(y_n|\mathbf{x}_n, \mathbf{w}) = P(C_1|\mathbf{x}_n, \mathbf{w})^{y_n} (1 - P(C_1|\mathbf{x}_n, \mathbf{w}))^{1-y_n} \quad (4.27)$$

Equation 4.27 is substituted into Equation 4.26 to obtain:

$$P(D|\mathbf{w}) = p(\mathbf{X}) \prod_{n=1}^N P(C_1|\mathbf{x}_n, \mathbf{w})^{y_n} (1 - P(C_1|\mathbf{x}_n, \mathbf{w}))^{1-y_n} \quad (4.28)$$

Expressing Equation 4.28 in terms of Equation 4.25 results in

$$P(D|\mathbf{w}) = p(\mathbf{X}) \prod_{n=1}^N \sigma(\mathbf{w}^T \mathbf{x}_n)^{y_n} (1 - \sigma(\mathbf{w}^T \mathbf{x}_n))^{1-y_n} \quad (4.29)$$

Equation 4.29 is referred to as the likelihood of the data \mathbf{D} given the weight vector \mathbf{w} . For the sake of brevity, however, this same equation will simply be referred to as the likelihood function in this thesis from here onward. Minimizing the negative log-likelihood function $E(\mathbf{w})$ is the same as maximizing the likelihood function. The log-likelihood function is given by:

$$\begin{aligned} E(\mathbf{w}) &= -\ln P(D|\mathbf{w}) \\ &= -\sum_{n=1}^N \{y_n \ln \sigma(\mathbf{w}^T \mathbf{x}_n) + (1 - y_n) \ln (1 - \sigma(\mathbf{w}^T \mathbf{x}_n))\} - \ln p(\mathbf{X}) \end{aligned} \quad (4.30)$$

The weight vector \mathbf{w} which minimizes $E(\mathbf{w})$ can be determined by setting $\nabla E(\mathbf{w}) = 0$. The gradient of $E(\mathbf{w})$ is equal to

$$\nabla E(\mathbf{w}) = \sum_{n=1}^N (\sigma(\mathbf{w}^T \mathbf{x}_n) - y_n) \mathbf{x}_n \quad (4.31)$$

There is no analytic solution to $\nabla E(\mathbf{w}) = 0$. This solution can only be determined numerically, i.e the Newton-Raphson's method has to be employed to do so. This is discussed in greater detail in Section 4.7.2.

Note that there is a problem: if the weights are not constrained in some way, they can grow arbitrarily large. This results in overfitting. To see why this is the case, note that if \mathbf{x}_n is assumed to be far from the decision boundary then $\sigma(\mathbf{w}^T \mathbf{x}_n) \approx 1$ and $\sigma(\mathbf{w}^T \mathbf{x}_n) - y_n \approx 0$, i.e \mathbf{x}_n has little effect on this equation:

$$\sum_{n=1}^N (\sigma(\mathbf{w}^T \mathbf{x}_n) - y_n) \mathbf{x}_n = 0 \quad (4.32)$$

If an observation is close to the decision boundary the converse is true. To force an observation into the correct class, \mathbf{w} can simply be made arbitrarily large, i.e $\mathbf{w}^T \rightarrow \infty$ implies that $\sigma(\mathbf{w}^T \mathbf{x}_n) \rightarrow \infty$. To prevent this, the weight \mathbf{w} needs to be constrained which can be accomplished by adding a penalty term to $E(\mathbf{w})$.

4.7.2 Newton-Raphson's method

Consider

$$l(\mathbf{w}) = - \sum_{n=1}^N \{y_n \ln \sigma(\mathbf{w}^T \mathbf{x}_n) \ln(1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) + (1 - y_n) \ln \sigma(\mathbf{w}^T \mathbf{x}_n)\} + \frac{1}{2\lambda} \mathbf{w}^T \mathbf{w} + C \quad (4.33)$$

where C is a constant. Note here that the aforementioned penalty term is added. This approach of constraining the weight vector \mathbf{w} is known as regularization and λ is known as the regularization constant.

As before $l(\mathbf{w})$ can be minimized by setting $\nabla l(\mathbf{w}) = 0$. The derivative of $l(\mathbf{w})$ with respect to \mathbf{w} is given by

$$\nabla l(\mathbf{w}) = - \sum_{n=1}^N (y_n - \sigma(\mathbf{w}^T \mathbf{x}_n)) \mathbf{x}_n + \frac{1}{\lambda} \mathbf{w} = 0 \quad (4.34)$$

$\nabla l(\mathbf{w}) = 0$ can only be solved numerically. One approach is to employ Newton-Raphson's method:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \mathbf{H}^{-1} \nabla l(\mathbf{w}) \quad (4.35)$$

where $\mathbf{H}(\mathbf{w}_k)$ is the Hessian of $l(\mathbf{w})$. The Hessian \mathbf{H} is given by

$$\mathbf{H}(\mathbf{w}) = \sum_{n=1}^N \sigma(\mathbf{w}^T \mathbf{x}_n) (1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) \mathbf{x}_n \mathbf{x}_n^T + \frac{1}{\lambda} \mathbf{I} \quad (4.36)$$

where \mathbf{I} is the identity matrix. To use Equation 4.35, a starting value of \mathbf{w} has to be chosen where one choice would be the all one vector.

4.8 Unsupervised Classification

This section is about unsupervised learning that makes use of datasets that are unlabelled. Two algorithms are presented in this section, namely k -means and GMM.

4.8.1 k -means Clustering

In the year 1957, Stuart Lloyd of Bell Labs proposed the standard (k -means) algorithm as a method for pulse-code modulation (digital way of presenting analog signals) which was not published at that time, but only later in the year 1982. Before Lloyd's work was published, Edward W. Forgy had produced a publication (in the year 1965) that was very similar to Lloyd's. James MacQueen, however was the first to coin the term k -means in the year 1967 [43].

k -means is a clustering algorithm which sorts data points with similar features together into clusters, either by location, shape or density. " k " in k -means refers to *number of clusters*, hence can be referred to as *cluster means*. Given a dataset, each observation is assigned to the cluster with the closest mean. Observations belonging to each cluster is used to update the cluster mean. Using the updated cluster means, observations can be re-allocated to clusters and is repeated until there is no further change (convergence) [43]. Since k -means utilizes the Euclidean norm, it has a linear decision boundary [2].

4.8.2 Gaussian Mixture Models (GMM)

A mixture model is a statistical model which is used to characterize an event within a group of events. Datasets are however not needed to locate the events to which each observation belongs [44].

Karl Pearson (1894) who was a mathematician and a statistician handled decomposition issues in characterizing unusual behaviours (of forehead to body length ratios) that female shore crab populations possess. Pearson's work is mostly referenced although work based on identifying the various elements, as well as parameters of mixture distributions have been in existence since 1846. He had inspiration for his work by a zoologist named Walter Frank Raphael Weldon [44].

Gaussian Mixture Models consist of a mixture of Gaussians in relation to discrete latent variables z . Assuming X indicates an $n \times m$ dataset, the dataset X would consist of n observations and each observation \mathbf{x} would have m features [45]. This mixture of Gaussians with k being the number of components

can be represented as

$$P(\mathbf{x}|\boldsymbol{\theta}) = \sum_{j=1}^k \pi_j \mathcal{N}(\mathbf{x}|\mathbf{u}_j, \boldsymbol{\Sigma}_j) \quad (4.37)$$

where π_j is the prior probability of the j -th Gaussian component, \mathbf{u}_j is the mean, $\boldsymbol{\Sigma}_j$ is the covariance matrix and $\mathcal{N}(\mathbf{x}|\mathbf{u}_j, \boldsymbol{\Sigma}_j)$ is a Gaussian density with the mean vector and the covariance matrix. Also, $\boldsymbol{\theta} = \{\pi_j, \mathbf{u}_j, \boldsymbol{\Sigma}_j | j \in \{1, 2, \dots, k\}\}$. However, it can be estimated by applying the Expectation Maximization (EM) algorithm [40]. Using the EM algorithm, $\boldsymbol{\theta}$ is initialized by applying the k -means algorithm. In the **E-step**, the responsibilities are evaluated,

$$\gamma_{nj} = \frac{\pi_j \mathcal{N}(\mathbf{x}_n|\mathbf{u}_j, \boldsymbol{\Sigma}_j)}{\sum_{i=1}^k \pi_i \mathcal{N}(\mathbf{x}_n|\mathbf{u}_i, \boldsymbol{\Sigma}_i)} \quad (4.38)$$

In the **M-step**, the parameters (in $\boldsymbol{\theta}$) are calculated using the current responsibilities:

$$\begin{aligned} \mathbf{u}_j &= \frac{1}{N_j} \sum_{n=1}^N \gamma_{nj} \mathbf{x}_n \\ \boldsymbol{\Sigma}_j &= \frac{1}{N_j} \sum_{n=1}^N \gamma_{nj} (\mathbf{x}_n - \mathbf{u}_j)(\mathbf{x}_n - \mathbf{u}_j)^T \\ \pi_j &= \frac{N_j}{N} \end{aligned} \quad (4.39)$$

where $N_j = \sum_{n=1}^N \gamma_{nj}$. The E-step and the M-step is repeated until convergence. Since the various components do not share a covariance matrix, the decision boundary of GMM is non-linear.

4.9 Confusion Matrix

A confusion matrix is a metric which is used to report the performance of a classification algorithm. It is represented in a tabular form with each entry being different combinations of actual and predicted values [35]. In Figure 4.2, the various divisions of a confusion matrix are shown. This same figure is associated with a binary classification problem which is made of two classes: the positive and negative class. Given a data, the positive class refers to the actual positive instances in the data whilst the negative class refers to the actual negative instances.

The following are terms used to describe the structure of a confusion matrix in terms of the positive and negative classes [39]:

		True Class	
		Positive	Negative
Predicted Class	Positive	TP	FP
	Negative	FN	TN

Figure 4.2: Structure of confusion matrix for binary classification (two classes)

1. True Positive (TP): This refers to the situation where a classifier correctly predicts that a positive sample is positive.
2. True Negative (TN): This refers to the situation where a classifier correctly predicts that a negative sample is negative.
3. False Positive (FP): This refers to the situation where a classifier incorrectly predicts that a negative sample is positive.
4. False Negative (FN): This refers to the situation where a classifier incorrectly predicts that a positive sample is negative.

4.9.1 Performance Metrics

Confusion matrices involve calculations which gives a better understanding on what the model is doing right and what it is doing wrong. To further aid in understanding how the model is performing, the following metrics can be calculated: recall, accuracy, specificity and many others when doing classification [37]. Out of the numerous metrics that were just stated, only that of the accuracy was used in this thesis.

4.9.1.1 Accuracy

It provides the relation between the total number of correct predictions and the total predictions that were made. It shows how often a classifier is correct

and it is formally expressed as

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \quad (4.40)$$

4.10 *Iris* dataset

In this section, the four algorithms presented in this chapter were applied to a dataset. This was done to highlight some of the differences between these algorithms in a practical setting. The dataset that has been chosen for this purpose is the *Iris* dataset.

The *Iris* flower data set also known as Fisher's *Iris* dataset is a multivariate data set which was created by Ronald Fisher, who was a statistician and biologist, in the year 1936. An American botanist by the name of Edgar Anderson gathered the data to vary the *Iris* flowers of three species morphologically. These species were called: *Iris setosa*, *Iris versicolor* and *Iris virginica* [46]. Due to this, the *Iris* flower data set is sometimes referred to as the Anderson's *Iris* dataset. They are shown in Figure 4.3:

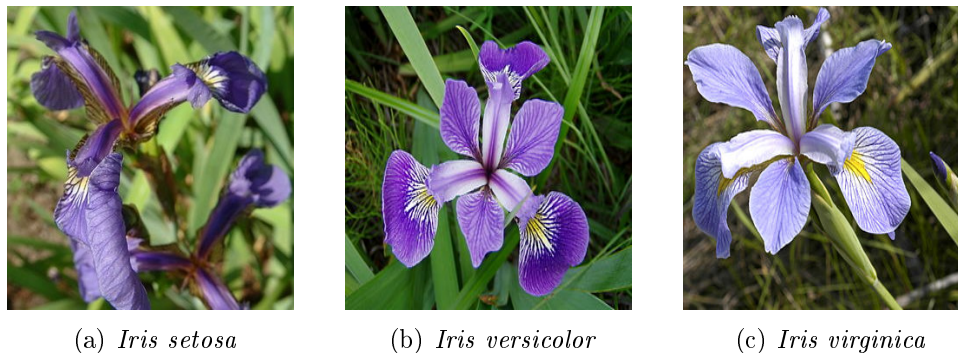


Figure 4.3: Figures representing the *Iris* dataset: *Iris setosa*, *Iris versicolor* and *Iris virginica* respectively.

This data set contains 50 samples; each of these samples can be associated with one of the three *Iris* species. The individual samples/observations has four features: the width and length of sepals and that of petals. Also, the unit length of measure used by Fisher was centimetres. He employed a linear discriminant model to differentiate the different species from one another using the aforementioned features [46]. It has become common place to use this dataset as a testbed for a variety of classification algorithms [46].

The *Iris* dataset is easily accessible via R and Python (scikit-learn package) and can be used by ML novices.

4.10.1 Results

This section consists of the various results which were obtained by the four classifiers presented in this chapter when they were applied on the Iris data set. Only two of the three classes were considered, i.e. versicolor and virginica and two out of the four features, namely: petal width and petal length. All of the data was used for training and testing. The decision boundaries that the various approaches find is what is of interest in this thesis and to relate this with the theory presented, and not the optimal classification accuracy of these methods.

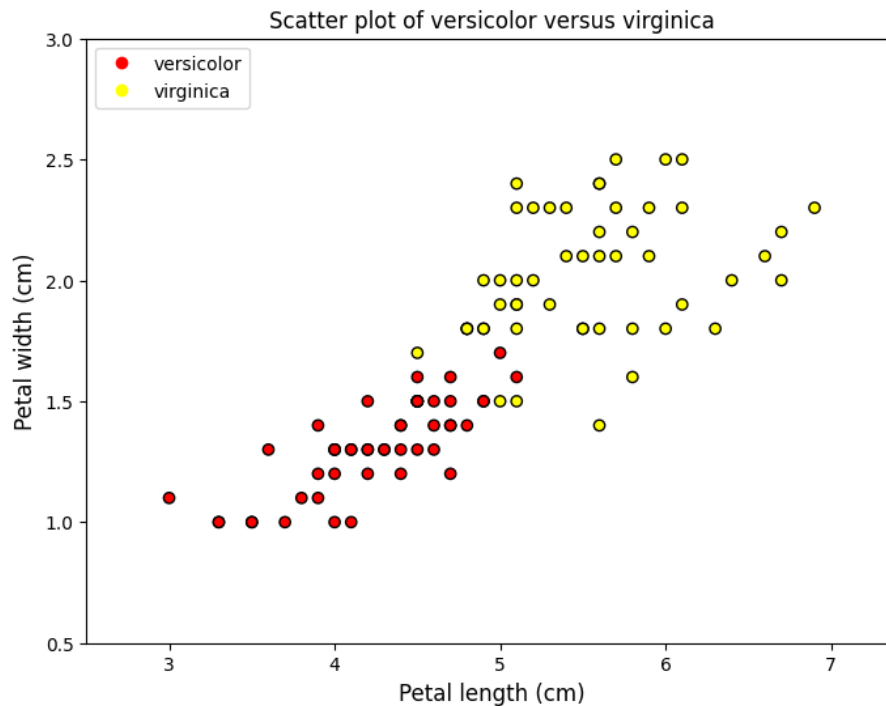


Figure 4.4: Scatter plot of two out of the three classes of the Iris dataset (versicolor and virginica) using the petal length on the x-axis and petal width on the y-axis

4.10.2 Observations and conclusions

1. Scatter plot: From the scatter plot, it can be concluded that versicolor and virginica classes are linearly separable.
2. Confusion matrix: In terms of confusion matrices, the Naive Bayes and Logistic Regression methods predicted both classes equally well. No versicolor was detected as virginica and vice-versa, unlike that of the

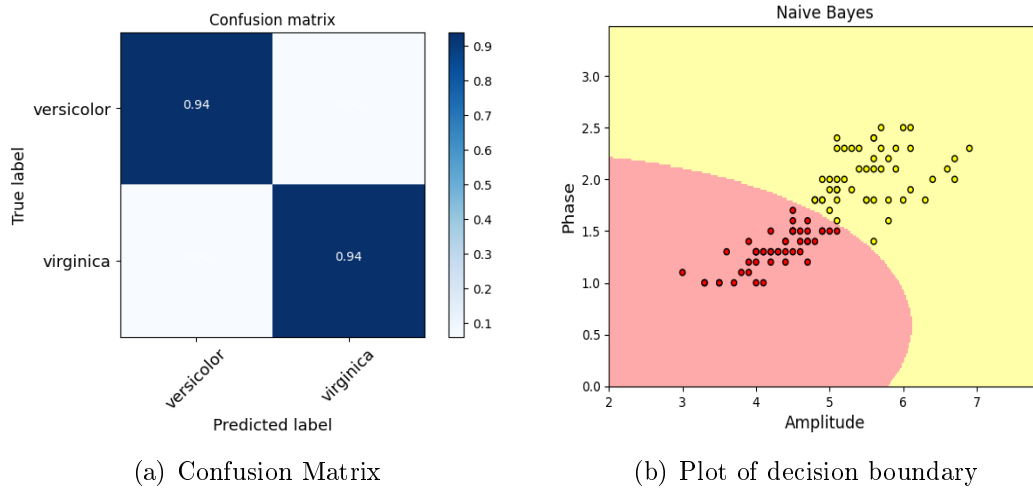


Figure 4.5: The confusion matrix and decision boundary plots of the versicolor and virginica classes of the Iris data set that were obtained if the Naive Bayes classifier is employed.

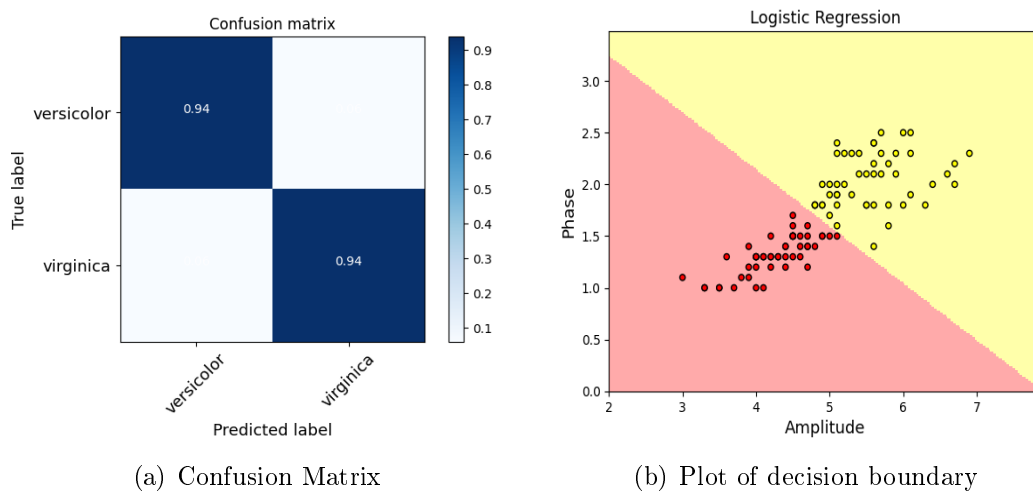


Figure 4.6: The confusion matrix and decision boundary plots of the versicolor and virginica classes of the Iris data set that were obtained if the Logistic Regression classifier is employed.

GMM and k -means that predicted more of versicolor than the virginica class. Hence, the supervised methods outperformed the unsupervised.

3. Decision boundary: The decision boundaries of the Naive Bayes and GMM were observed to be non-linear, whereas that of logistic regression and k -means are linear or have linear boundaries which aligns with the theory aspect presented in this thesis.

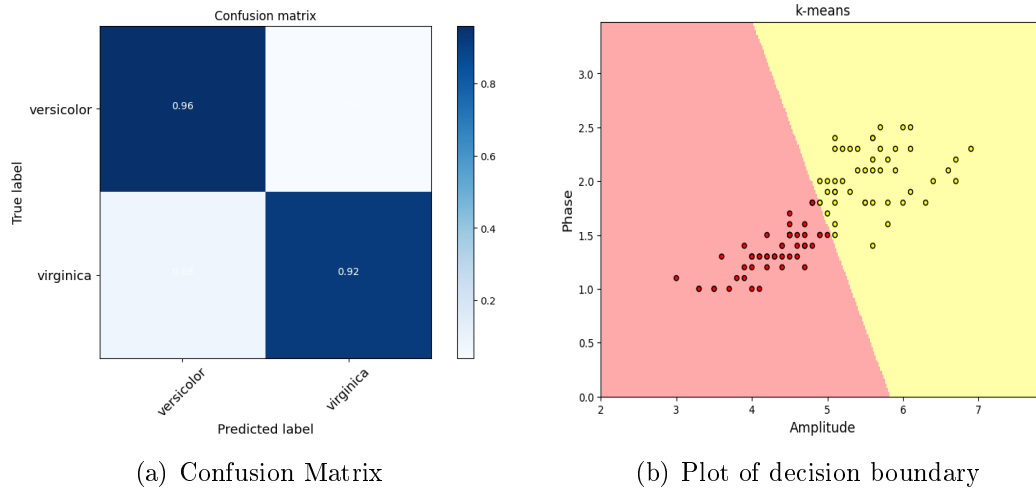


Figure 4.7: The confusion matrix and decision boundary plots of the versicolor and virginica classes of the Iris data set that were obtained if the *k*-means classifier is employed.

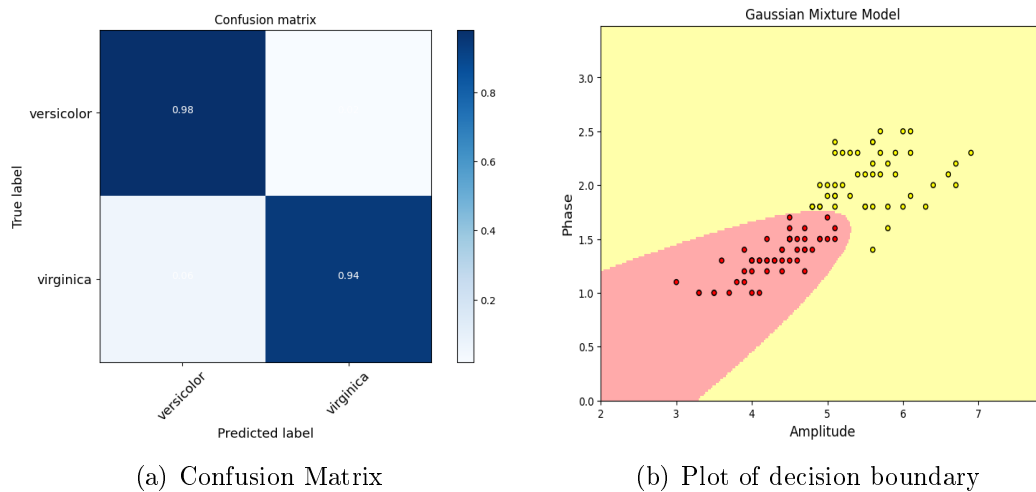


Figure 4.8: The confusion matrix and decision boundary plots of the versicolor and virginica classes of the Iris data set that were obtained if the Gaussian Mixture Model classifier is employed.

Chapter 5

Testing, Results and Analysis

In this chapter, Section 5.1 presents an overview of the simulator, its benefits to students and the programming language that was used. Section 5.2 presents a flowchart on how the simulator was created and what one can achieve using this simulator. Section 5.3 discusses the programmatic flow and how the simulator was tested. The limitations of the simulator are outlined in Section 5.4 and the plots of the results obtained from the simulator runs are seen in Section 5.5. The average TP, TN, FN, FP and standard deviation values of each algorithm that are tabulated, as well as the results after applying the classification algorithms are presented in Section 5.6. Furthermore, Section 5.7 outlines the summary and conclusions of the observations that were made. Section 5.8 lists the python files that can be located in the github repository (see Section 1.1), which can be used to teach students the basics of interferometry and ML.

5.1 Introduction

The main target of this work is to create an RFI pipeline that would serve as a teaching tool for students who wish to learn interferometry and machine learning. This pipeline would thereby be of benefit to students learning about machine learning and interferometry.

This is a software based project which made use of the Python programming language. The textbook Python Cookbook [47] is a useful reference for students who wish to master programming. As per the objectives this project sought to achieve, the simulator was tested by running a couple of experiments to ensure that it functions properly.

5.2 Simulator Design

One first needs to design the software architecture before development commences. Below is a block diagram describing how the simulator works.

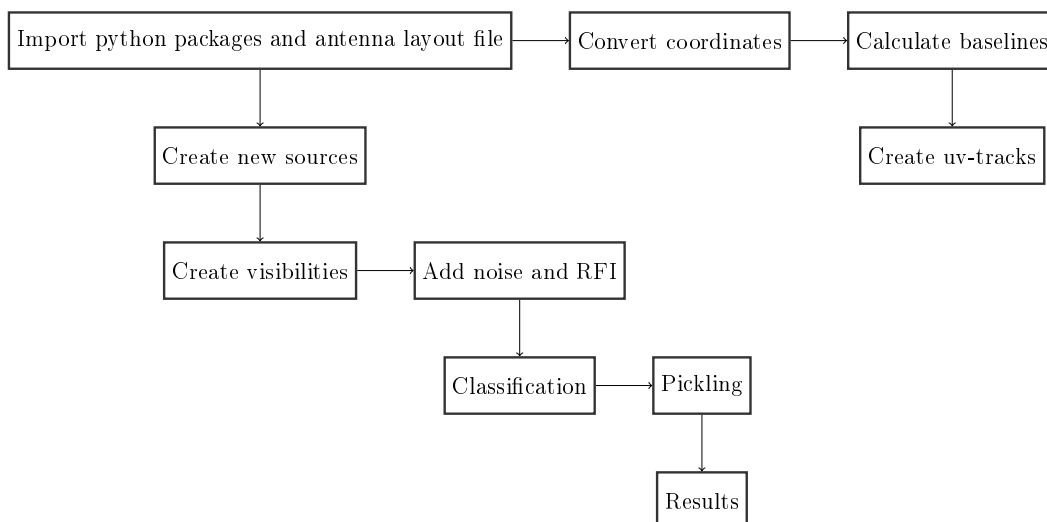


Figure 5.1: Block diagram describing how the simulator works

This block diagram is discussed in detail in Section 5.3. The software can plot *uv-tracks* (see Figure 5.3), visibilities (see Figure 5.7), visibilities corrupted via noise (see Figure 5.8(a)), visibilities corrupted via RFI (see Figure 5.8(b)) and the results obtained from four classifiers (see section 5.6). The run time of a simulation is dependent on the complexity of the model in addition to the number of runs. The simulator is user-friendly, easy to operate and performs input validation. It was coded in Python 3 and also makes use of the following external packages: *matplotlib*¹, *numpy*², *sklearn* [48], *pickle*³ etc. The user interface of the simulator is the command prompt.

The simulator consists of a custom built Python object. An inductive based RFI model was used, which entails the replication of a dataset via a statistical model. The simulator presented in [11] on the other hand made use of a deductive model. A deductive simulation model replicates the physical mechanism which resulted in the dataset being considered. All RFI sources were modelled on the physical level and then sources were created which were RFI objects. They were then observed and the visibilities gotten when the sources were observed are simulated. By definition⁴, a deductive model is one which begins with a general statement or hypothesis and aims at testing theories which already exists whereas an inductive model generates new theory from specific observations, thus, conclusions are drawn from the data itself. In an inductive model, observations are made, patterns are observed and then theories are developed which is exactly what was done in this study. In-depth

¹<https://matplotlib.org/stable/contents.html>

²<https://numpy.org/doc/stable/numpy-user.pdf>

³<https://docs.python.org/3/library/pickle.html>

⁴<https://www.livescience.com/21569-deduction-vs-induction.html>

discussions of the two models can be seen in [49]. Also, the differences between these two models is discussed in Section 1.1. Some of the existing deductive RFI simulators were reviewed in Section 3.5. Some detail as to how to go about implementing an inductive RFI simulator is given in Section 3.6.

5.3 Simulator Settings and Programmatic Flow

The command line arguments which are passed into the simulator is parsed via *getopt*⁵. The command line arguments are: number of sources to create represented as n , the pareto parameter as a , the field of view as f , the number of channels as c , the number of experiments to perform as e , the signal-to-noise ratio value due to the thermal noise as s and the signal-to-noise ratio value due to the RFI as r and which classifier to use for the experiment.

The antenna layout used was that of KAT-7. The ENU coordinates of KAT-7 is depicted in Table 5.1. The observational parameters used in our simulator is given in Table 5.2, they include the latitude, starting hour angle H_0 , stopping hour angle H_1 , declination δ_0 , right ascension α_0 and the observational frequency f . These parameters except frequency were converted into radians from degrees. This information was imported into our simulator via a package called *pandas*⁶. The observation wavelength was also calculated from the observation frequency f .

Antenna	E (x)	N (y)	U (z)
Antenna 1	25.095 m ($x1$)	-9.095 m ($y1$)	0.045 m ($z1$)
Antenna 2	90.284 m	26.380 m	-0.226 m
Antenna 3	3.985 m	26.893 m	0.000 m
Antenna 4	-21.605 m	25.494 m	0.019 m
Antenna 5	-38.272 m	-2.592 m	0.391 m
Antenna 6	-61.595 m	-79.699 m	0.702 m
Antenna 7	-87.988 m	75.754 m	0.138 m

Table 5.1: The ENU (east-north-up) coordinates of KAT-7

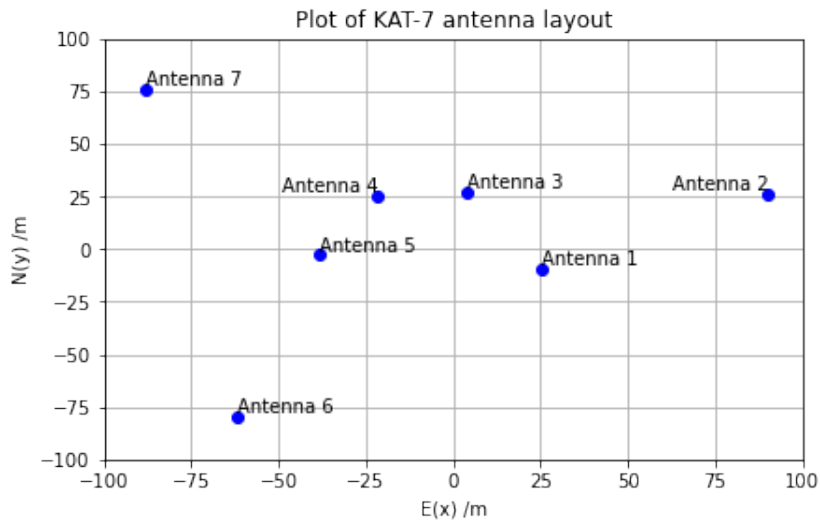
Table 5.2 provides information about an observation that was conducted with KAT-7.

Since the information in Table 5.1 tells us about the number of antennas in KAT-7, the number of baselines of our interferometer can be computed. The formula that was used to calculate the number of baselines is given in Equation 2.4 of Chapter 2.

⁵<https://docs.python.org/3/library/getopt.html>

⁶<https://pandas.pydata.org/docs/>

Name	Value
Latitude L	$-30^\circ - 43' - 17.34''$
Starting hour angle H_0	-4h
Stopping hour angle H_1	4h
Field center δ_0	$-74^\circ - 39' - 37.481''$
Field center α_0	4h 44m 6.686s
Observational frequency f	1.4 GHz

Table 5.2: Observation conducted with KAT7**Figure 5.2:** Plot of the positions of the KAT-7 antenna given in Table 5.1.

The ENU baseline difference vector of the various baselines, the length D of the baselines, the azimuth angle \mathcal{A} and the elevation (altitude) angle \mathcal{E} of the baselines were calculated and are then used to calculate the XYZ coordinates. The XYZ coordinates are then converted into uvw coordinates. As the hour angle changes, the computed uvw coordinates start to form uv -tracks. One can refer back to Section 2.2 to see how these calculations are to be performed.

The pareto distribution is an exponential decaying probability distribution used to express types of events eg. social and scientific events [50]. The shape of the pareto distribution is affected by a parameter known as the pareto parameter. The pareto distribution is however used in this case to describe sources; as there are few bright sources and many faint sources. In relation to this thesis, one hundred flat spectrum sources were created using the pareto parameter and are located uniformly in a pre-specified field of view (FoV). The amplitude of these sources were drawn from a pareto distribution with a pre-specified pareto parameter. The values of the pareto parameter and the FoV used for the simulation were 2 and 10 respectively. Subsequently, a completely

filled in uv -plane is created with visibilities which is then sampled (see Section 2.3). The sky model is expressed as $I(l, m) = \sum_k A_k \delta(l - l_k, m - m_k)$ where A_k represents the amplitude, l_k and m_k represents the l and m coordinates of the k -th source. The FT of $I(l, m)$ is then $V(u, v) = \sum_k A_k \exp(-2\pi i(ul_k + vm_k))$, this is how the sky model is constructed.

A mask which indicates which channels to corrupt is then constructed. This mask is also used to generate the waterfall plots in Figure 5.8(a) and Figure 5.8(b). The number of channels simulated is 200, five of which were corrupted.

Thermal noise was generated with a specific SNR value and was added to the visibilities. The same was done to add RFI to our simulation (this is discussed in more detail in Section 3.6) but measured relative to the astronomical sources and not to the thermal noise that is added to the visibility. Thus, the RFI is added under the assumption that the astronomical component is the power. Hence, it is not added on the noise induced visibility but rather computed based on the power in the original signal. The RFI SNR value always needs to be less than the SNR value of the thermal noise which is added because RFI is much stronger than noise and the more negative a value is, the more the noise level. Three different thermal and RFI SNR combinations are considered in this study. They were chosen by performing a grid search. These three combinations can be associated with three different RFI categories, namely a strong RFI use case (SNR = 5 dB and RFI = -17 dB), a strong-to-medium RFI use case (SNR = 10 dB and RFI = -13 dB) and medium RFI use case (SNR = 0 dB and RFI = -11 dB).

Machine learning was also incorporated into the pipeline. ML is used to perform RFI detection (see Section 5.6). Plots of the decision boundaries, confusion matrices and scatter plots are seen in the aforementioned section as well. Four classifiers were used, namely: Naive Bayes classifier, the Logistic Regression classifier, GMM classifier and k -means classifier (see Chapter 4). The data associated with a single baseline was then randomly split into a training set (X_{train}) and a test set (X_{test}) of equal size. The classifiers were trained using X_{train} and tested using X_{test} . This experiment was repeated 100 times for each classifier.

The confusion matrices of each run were all pickled or stored to different pickle files. These files were stored in different directories which were named after each classifier type. The individual files in each directory contained the following information in their names: the classifier type, experiment number, SNR value for thermal noise and RFI. For example, all GMM experiments were saved to a GMM experiments directory, and in that directory, there were different folders which were used to group the results of the 100 experiments for each of the three SNR combinations. A different script was written to read in the pickle files which calculates the average of all confusion matrices, accuracies, true positives and true negatives (also see Section 4.9). The standard

deviations of each classifier were calculated and then averaged as well. How this is calculated is discussed in Section 5.6.

5.4 Simulator Limitations

This simulator is of a simplistic nature. Below are some of the limitations of the project and the simulator. However, all these limitations can be worked on and if it is improved enough, it could possibly be useful in real-world applications. Its current simplistic design makes it an ideal teaching tool.

1. No median filter is applied before RFI detection is performed. The performance of the classifiers would improve if the visibilities were filtered first. This is so because median filters are used to remove noise from signals hence if applied, the noise content from visibilities will reduce, thereby increasing the performance of the classifiers.
2. No hyper-parameter tuning/no regularization was performed when running the Logistic Regression classifier experiment. The default value produced accurate enough results.
3. Only one baseline is considered.
4. Polarization is ignored.
5. Only a limited number of channels were corrupted
6. Same SNR value was used for each of the channels that were corrupted.
7. A very simple first order normal distribution was assumed as the distribution for RFI noise for the sake of simplicity. Channel independence was also assumed. This should be tested via experiments and improved upon if need be.
8. A Graphical User Interface (GUI) should be added to improve its efficacy as a teaching tool
9. More advanced ML metrics can be considered to evaluate the different RFI detection methods.

5.5 Simulator Validation

The results plotted in Figures 5.3 to Figures 5.8(b) were obtained by running the simulator and are shown here for validation purposes.

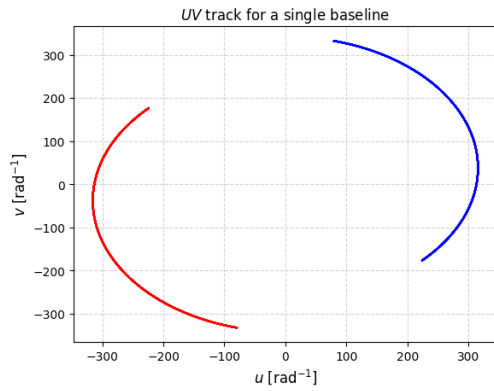


Figure 5.3: uv -track of a single baseline (baseline 12 and 21)

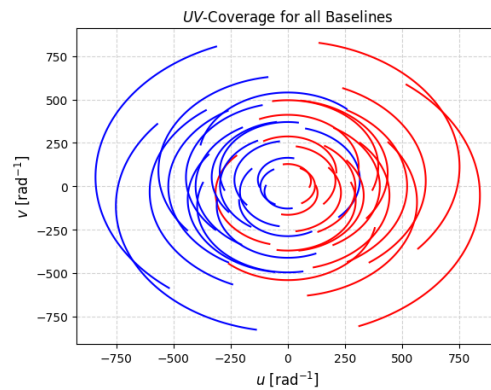


Figure 5.4: uv -coverage

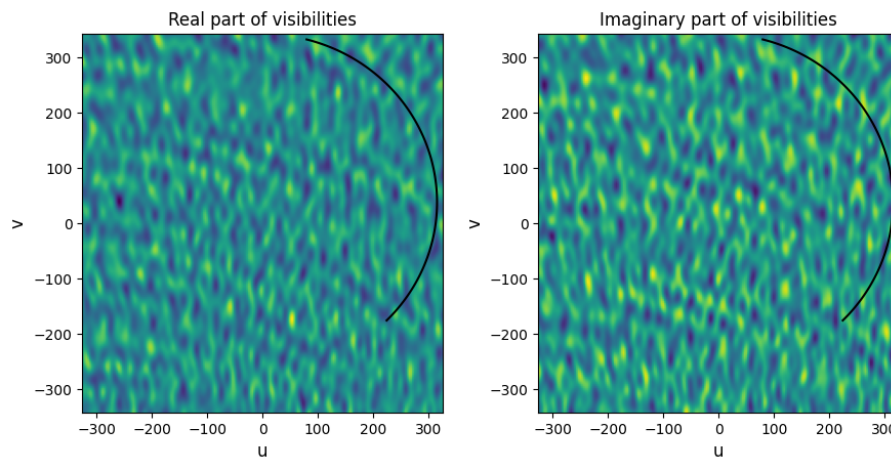


Figure 5.5: Real and imaginary part of visibilities plotted with 100 sources, pareto number of 2 and Fov value of 10

5.6 Machine Learning and RFI Detection Results

This section comprises of tables which includes the average confusion matrices (True Positive (TP), False Positive (FP), False Negative (FN), True Negative (TN)) of each SNR combination of the four algorithms (Naive Bayes, Logistic Regression, GMM and k -means algorithms). It also contains a multitude of plots that tell us how effective the aforementioned algorithms were at detecting RFI. The values reported are normalized and are not percentages.

The standard deviation was calculated by adding the diagonals of each of the saved confusion matrices (TP and TN) and averaged (divided by 2). The

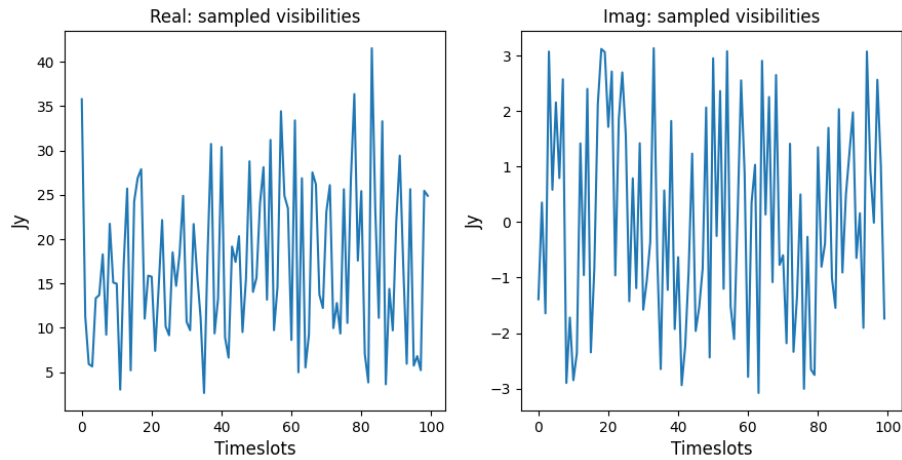


Figure 5.6: Sampled visibilities with their real and imaginary components of 100 sources, pareto number of 2 and Fov value of 10 plotted

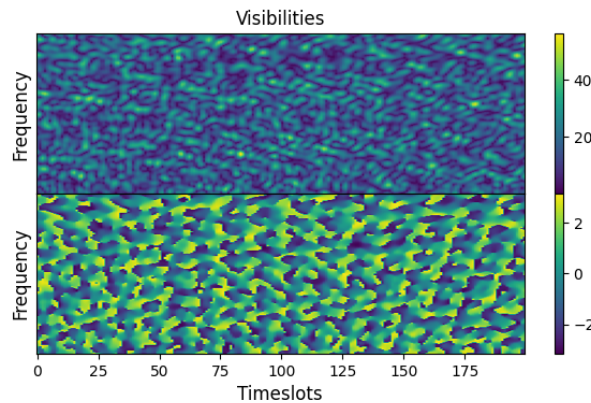


Figure 5.7: The phase and amplitude of the visibilities of baseline 12 (Antenna 1 and 2) as a function of frequency and timeslots. This plot is commonly referred to as a waterfall plot within the literature. The top panel is the amplitude and the bottom panel is the phase.

average computed values were then plotted in the bar graph (see Figure 5.34). It is the same value which is reported in the tables.

Three SNR combinations were investigated. In the first use case, 5 dB and -17 dB were used. This represents the use case in which the RFI is very high. The thermal noise level is relatively low. For the second use case, 10 dB and -13 dB was used. In this use case the RFI noise level which is induced is medium-to-high. The thermal noise level is lower than that in the first use case (this is of secondary importance, the main thing that is of interest in modelling for each of the three use cases is the level of RFI noise which was

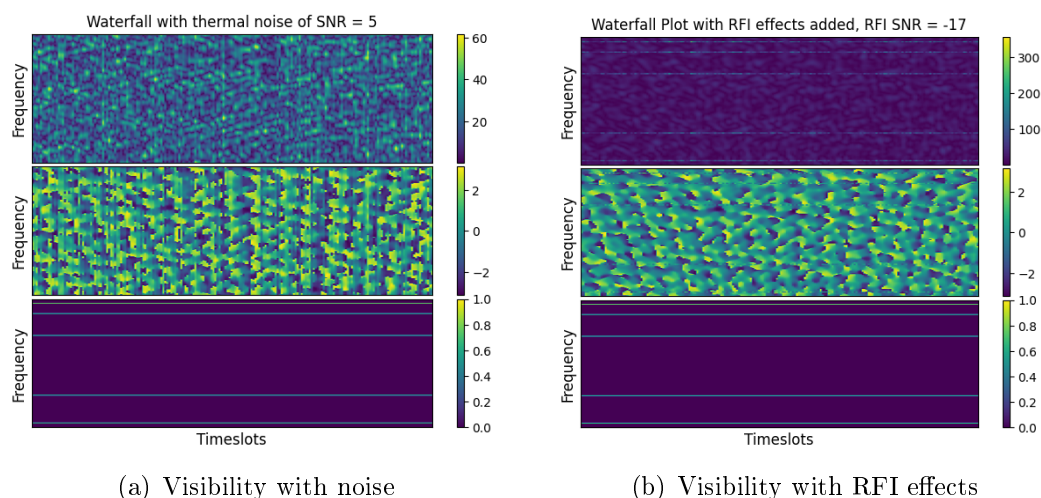


Figure 5.8: The plots in the figures above are visibilities with thermal noise and RFI added, respectively. The visibilities are plotted as a function of frequency and time-slots. In the top panel the amplitude is plotted, in the middle panel the phase and in the bottom panel the RFI corruption mask.

added). In the last use case, 0 dB and - 11 dB was used. In other words, a medium RFI noise level and a relatively high level of thermal noise were used.

The green and red labels used in the scatter plots and decision boundary plots correspond to the “g” and “r” labels that are used in the confusion matrices respectively. For the sake of readability, these labels are used in all plots in this section. It is clear from the scatter plot that the red label indicates true visibilities and the green label indicates RFI. Furthermore, the visibilities represent the positive class and RFI represents the negative class.

5.6.1 Naive Bayes Results

The confusion matrix associated with Naive Bayes is diagonally dominant. This method is performing the best. The average confusion matrices for the three use cases for the Naive Bayes classifier is presented in Table 5.3. The confusion matrix, a labelled visibility scatter plot and a decision boundary plot of a randomly selected experiment involving the Naive Bayes classifier for the three use cases considered is depicted in Figures 5.10 to 5.14.

5.6.2 Logistic Regression Results

The LR classifier performs similarly to the NB classifier. The average confusion matrices for the three use cases for the Logistic Regression classifier is presented in Table 5.4. The confusion matrix, a labelled visibility scatter plot and a decision boundary plot of a randomly selected experiment involving the

SNR value	RFI SNR	Confusion Matrix				Standard deviation
		TP	FP	FN	TN	
0	-11	0.9985	0.0014	0.3561	0.6438	± 0.0287
5	-17	0.9995	0.0004384	0.1188	0.8811	± 0.0156
10	-13	0.9990	0.0009687	0.2515	0.7484	± 0.0237

Table 5.3: The average confusion matrices of the Naive Bayes classifier for the three use cases that were considered in this study.

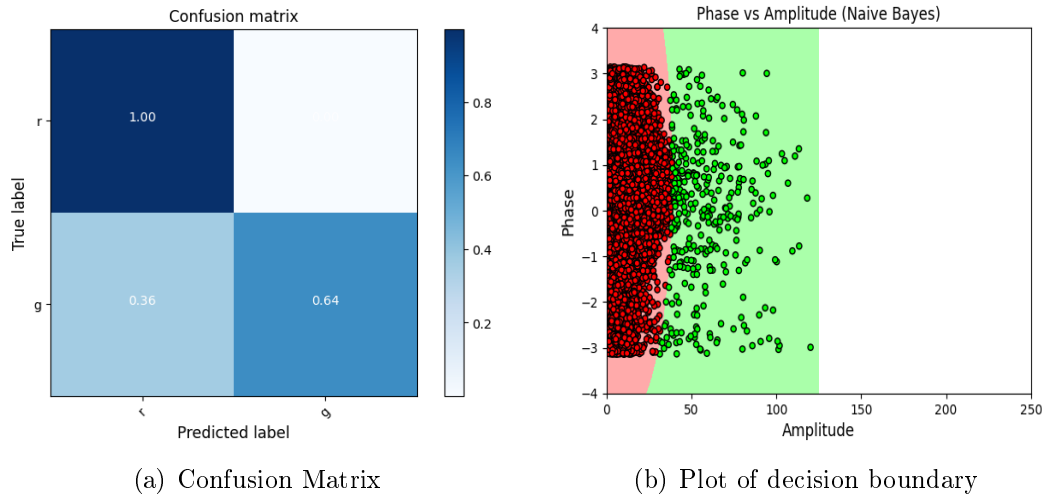


Figure 5.9: The confusion matrix and decision boundary plots for the SNR combination 0 dB and -11 dB for the Naive Bayes classifier

Logistic Regression classifier for the three use cases considered is depicted in Figures 5.15(a) to 5.19(a).

SNR value	RFI SNR	Confusion Matrix				Standard deviation
		TP	FP	FN	TN	
0	-11	0.9997	0.0002866	0.4007	0.5992	± 0.0394
5	-17	0.9999	0.00007538	0.1324	0.8675	± 0.0189
10	-13	0.9998	0.0001841	0.2878	0.7122	± 0.0274

Table 5.4: The average confusion matrices of the Logistic Regression classifier for the three use cases that were considered in this study.

5.6.3 Gaussian Mixture Model (GMM) Results

In terms of unsupervised classification, this model outperforms k -means. The average confusion matrices for the three use cases for the GMM classifier is

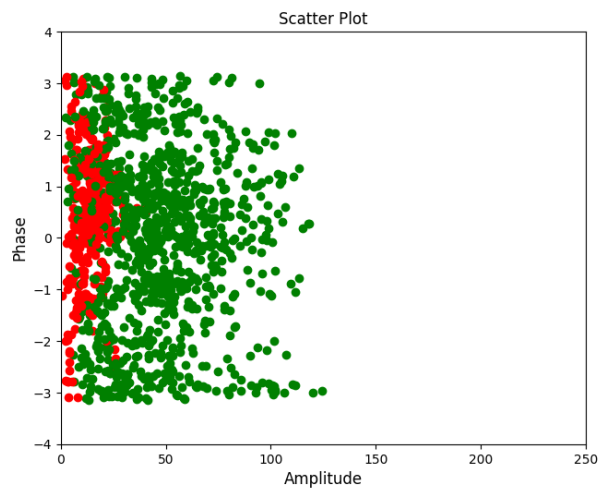
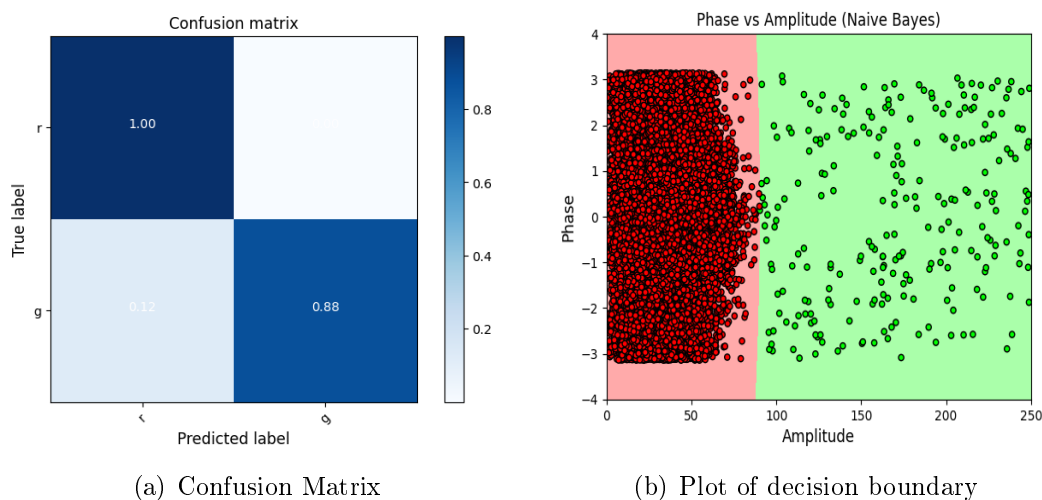


Figure 5.10: Scatter plot of the Naive Bayes classifier for the SNR combination 0 dB and -11 dB



(a) Confusion Matrix

(b) Plot of decision boundary

Figure 5.11: The confusion matrix and decision boundary plots of the SNR combination 5 dB and -17 dB for the Naive Bayes classifier

presented in Table 5.5. The confusion matrix, a labelled visibility scatter plot and a decision boundary plot of a randomly selected experiment involving the GMM classifier for the three use cases considered is depicted in Figures 5.21(a) to 5.25(a).

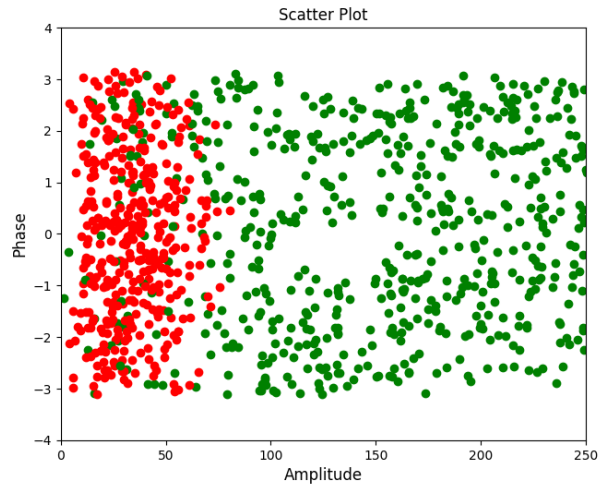


Figure 5.12: Scatter plot of the Naive Bayes classifier for the SNR combination 5 dB and -17 dB

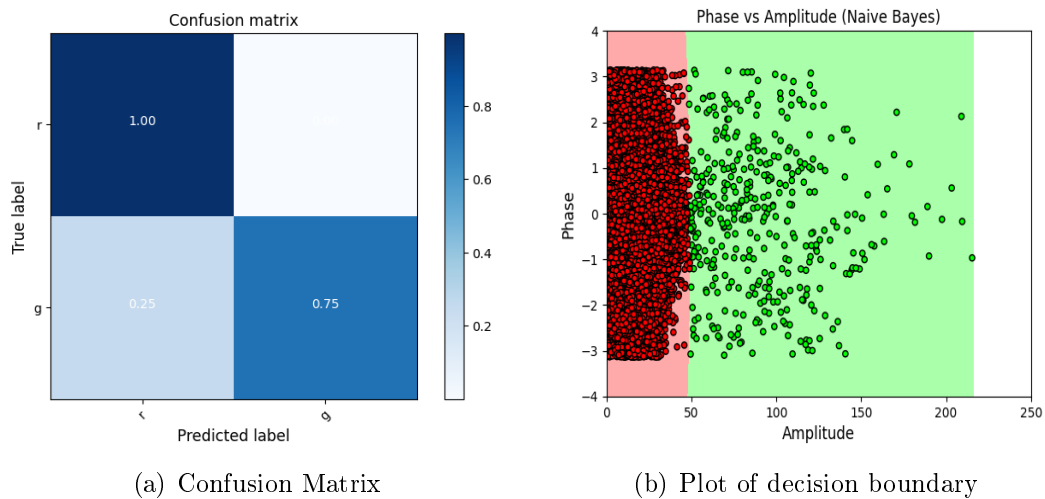


Figure 5.13: The confusion matrix and decision boundary plots for the SNR combination 10 dB and -13 dB for the Naive Bayes classifier

5.6.4 *k*-means Results

This method performed the worst. The average confusion matrices for the three use cases for the *k*-means classifier is presented in Table 5.6. The confusion matrix, a labelled visibility scatter plot and a decision boundary plot of a randomly selected experiment involving the *k*-means classifier for the three use cases considered is depicted in Figures 5.27(a) to 5.32.

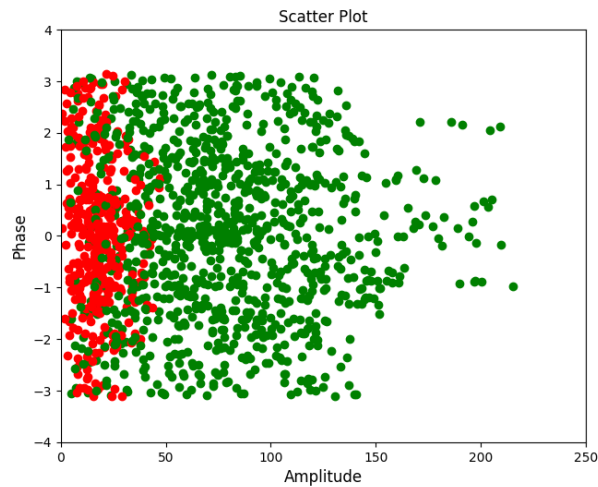


Figure 5.14: Scatter plot of the Naive Bayes classifier for the SNR combination 10 dB and -13 dB

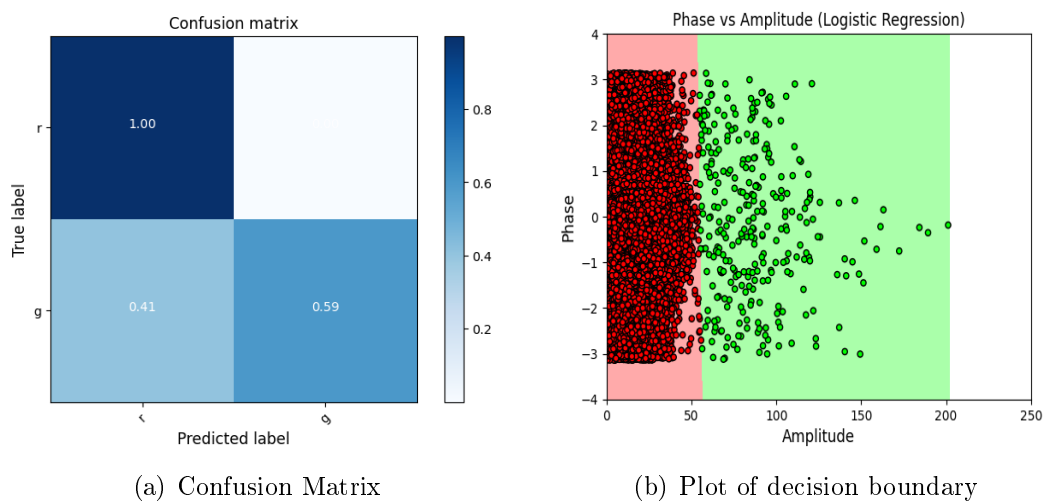


Figure 5.15: The confusion matrix and decision boundary plots for the SNR combination 0 dB and -11 dB for the Logistic Regression classifier

5.6.5 Average accuracy

The bar graph in Figure 5.33 shows that k -means is performing the worst of all the methods but the remaining methods are all performing similarly. Not one of the methods is significantly outperforming the other methods. The average accuracy of each method is reducing as the RFI SNR value increases.

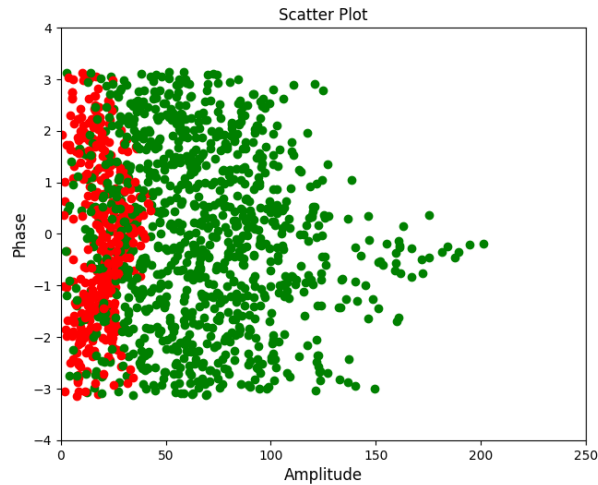
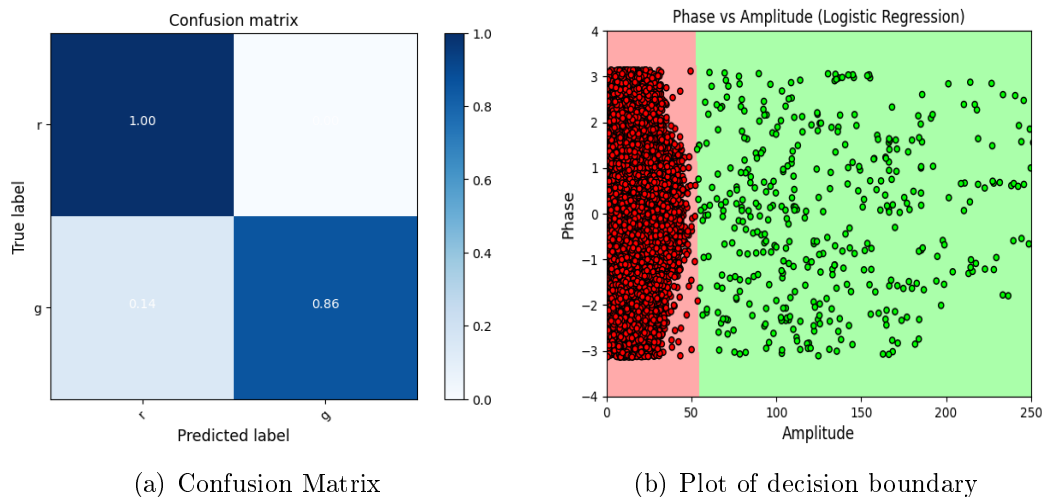


Figure 5.16: Scatter plot of the Logistic Regression classifier for the SNR combination 0 dB and -11 dB



(a) Confusion Matrix

(b) Plot of decision boundary

Figure 5.17: The confusion matrix and decision boundary plots for the SNR combination 5 dB and -17 dB for the Logistic Regression classifier

5.6.6 Standard Deviation

The bar graph in Figure 5.34 is a plot of standard deviation versus the SNR combinations. It indicates that the classifiers are very stable (the Naive Bayes classifier in particular). The only exception (quite notably) is k -means (and to a lesser degree GMM as well), which is to be expected since it is a linear unsupervised approach. It was observed that the TP and TN of the k -means classifier changed quite significantly during the simulator runs. The same applies to GMM for the SNR combination of $\{0 \text{ dB}, -11 \text{ dB}\}$. The Naive

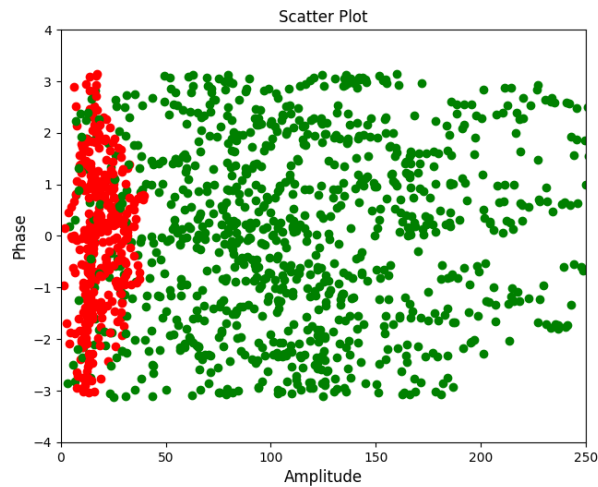


Figure 5.18: Scatter plot of the Logistic Regression classifier for the SNR combination 5 dB and -17 dB

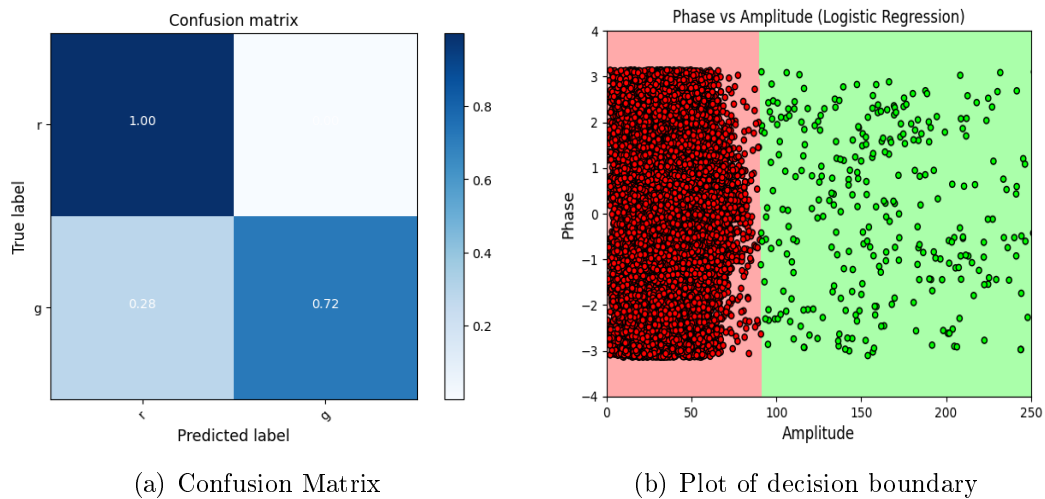


Figure 5.19: The confusion matrix and decision boundary plots for the SNR combination 10 dB and -13 dB for the Logistic Regression classifier

Bayes classifier was more stable than the other methods.

5.6.7 True Positive

TP is the upper left cell in a confusion matrix which tells us how well it labelled true visibilities as such and how well the visibilities can be distinguished from the corrupted ones. Its graph is shown in Figure 5.35. In the TP results, there is not much difference in GMM and k -means for the first two use cases

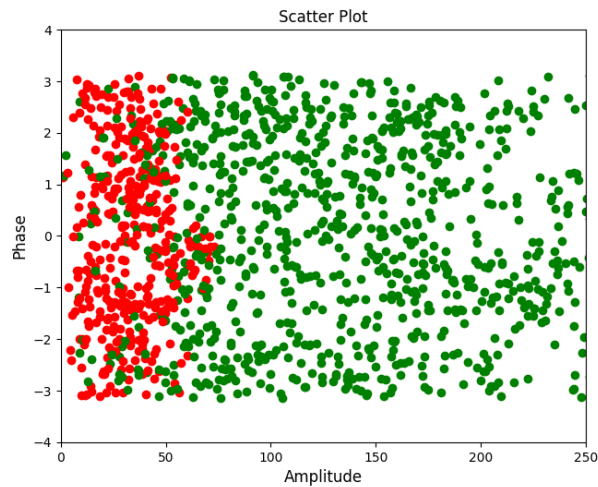


Figure 5.20: Scatter plot of the Logistic Regression classifier for the SNR combination 10 dB and -13 dB

SNR value	RFI SNR	Confusion Matrix				Standard deviation
		TP	FP	FN	TN	
0	-11	0.9809	0.0190	0.3120	0.6879	± 0.0530
5	-17	0.9995	0.0004164	0.1236	0.8763	± 0.0157
10	-13	0.9987	0.001256	0.2554	0.7445	± 0.0228

Table 5.5: The average confusion matrices of the GMM classifier for the three use cases that were considered in this study.

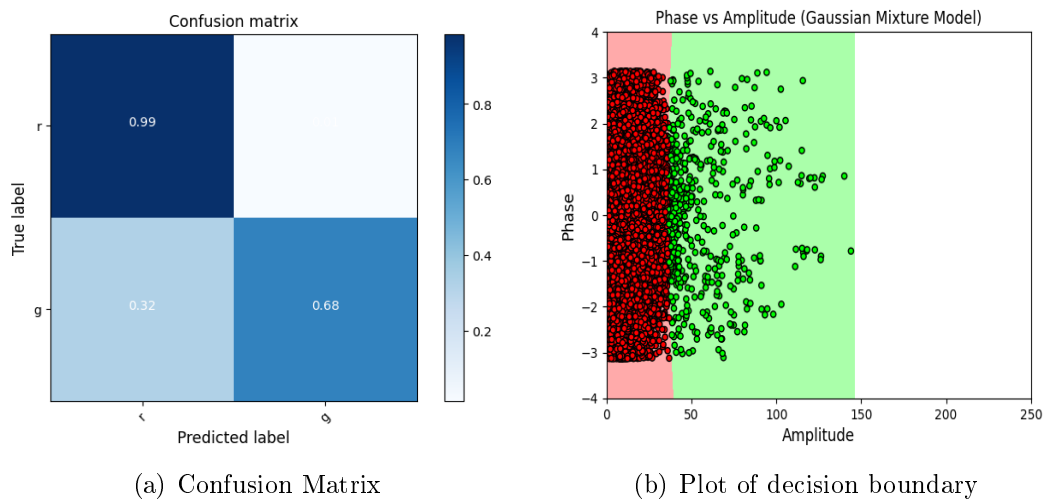


Figure 5.21: The confusion matrix and decision boundary plots for the SNR combination 0 dB and -11 dB for the GMM classifier

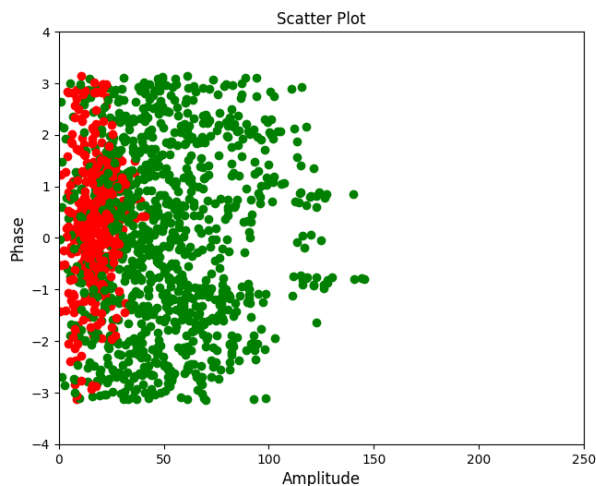


Figure 5.22: Scatter plot of the GMM classifier for the SNR combination 0 dB and -11 dB

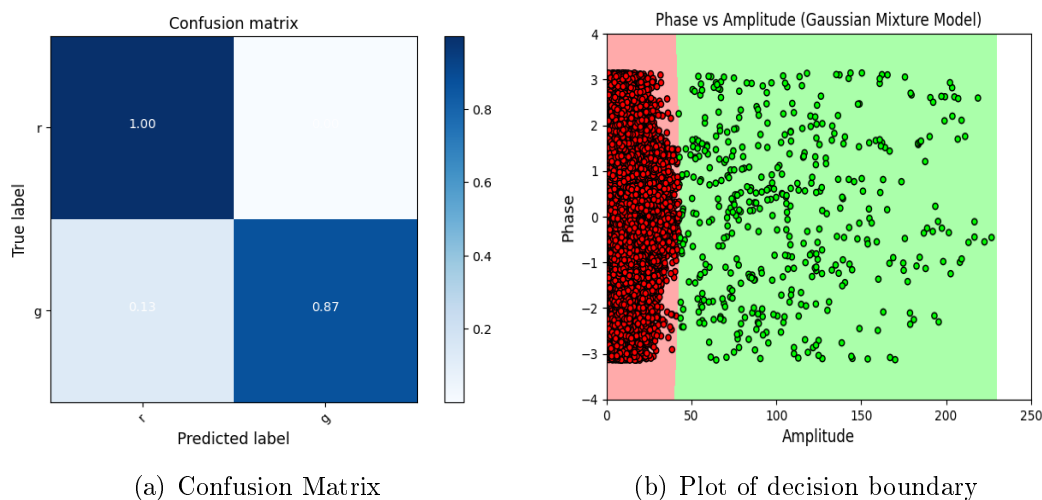


Figure 5.23: The confusion matrix and decision boundary plots for the SNR combination 5 dB and -17 dB for the GMM classifier

SNR value	RFI SNR	Confusion Matrix				Standard deviation
		TP	FP	FN	TN	
0	-11	0.8367	0.1632	0.2419	0.7581	± 0.1497
5	-17	1	0	0.3299	0.6701	± 0.0212
10	-13	0.9923	0.0076	0.3691	0.6309	± 0.0510

Table 5.6: The average confusion matrices of the k -means classifier for the three use cases that were considered in this study.

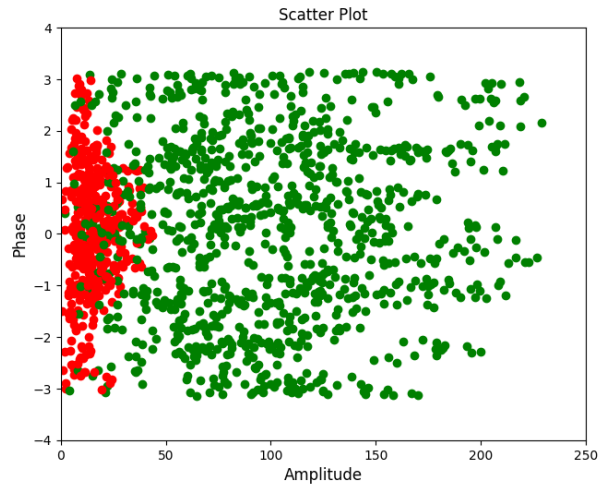
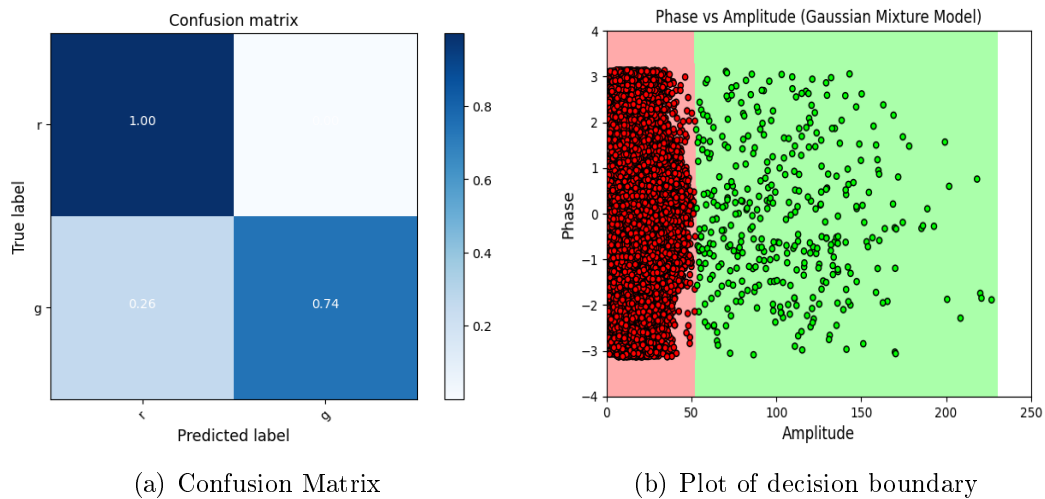


Figure 5.24: Scatter plot of the GMM classifier for the SNR combination 5 dB and -17 dB



(a) Confusion Matrix

(b) Plot of decision boundary

Figure 5.25: The confusion matrix and decision boundary plots for the SNR combination 10 dB and -13 dB for the GMM classifier

plotted in Figure 5.35. For the SNR combination $\{0 \text{ dB}, -11 \text{ dB}\}$, k -means performed significantly worse than GMM. It was observed during the course of the experiments that the TPs of k -means and GMM often deviated significantly from one, lowering the overall accuracy of these two methods. The supervised approaches on the other hand were able to find decision boundaries which maximized their TP scores. This explains their consistency and the results reported in Figure 5.34 as well.

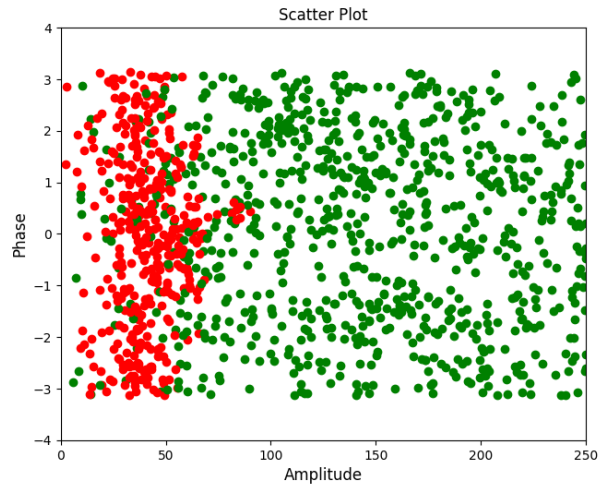


Figure 5.26: Scatter plot of the Gaussian Mixture Model classifier for the SNR combination 10 dB and -13 dB

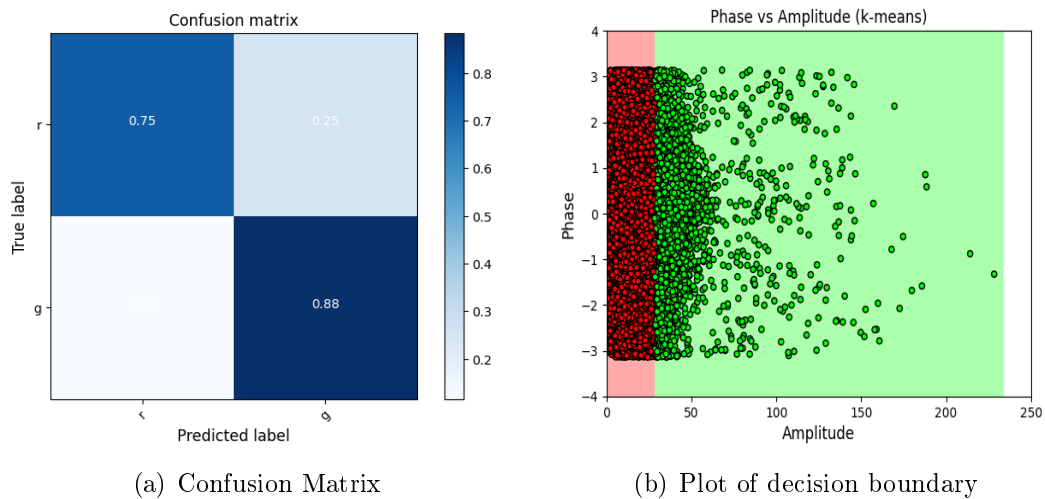


Figure 5.27: The confusion matrix and decision boundary plots for the SNR combination 0 dB and -11 dB for the k -means classifier

5.6.8 True Negative

TN is the lower right cell in a confusion matrix which tells how well it labelled RFI as such. All TN values were averaged and plotted in Figure 5.36. It was observed that k -means did the worst in detecting RFI. The TN results of k -means did improve for the SNR combination use case $\{0 \text{ dB}, -11 \text{ dB}\}$. The fact that k -means has such a low TN value is what is lowering its overall accuracy, making it one of the weakest RFI approaches considered in the study. Moreover, the improvement it shows for the use case $\{0 \text{ dB}, -11 \text{ dB}\}$ SNR com-

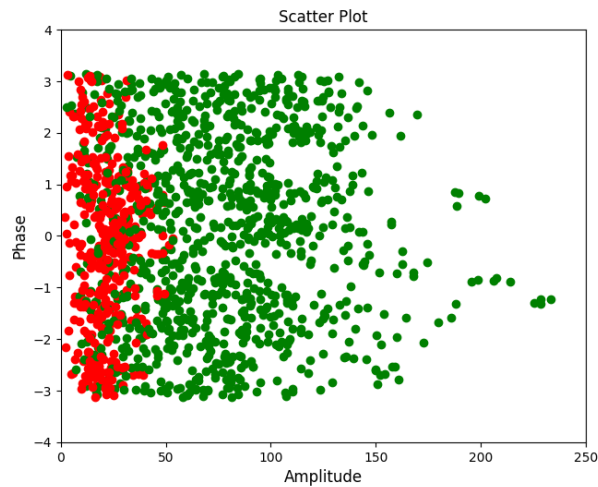


Figure 5.28: Scatter plot of the k -means classifier for the SNR combination 0 dB and -11 dB

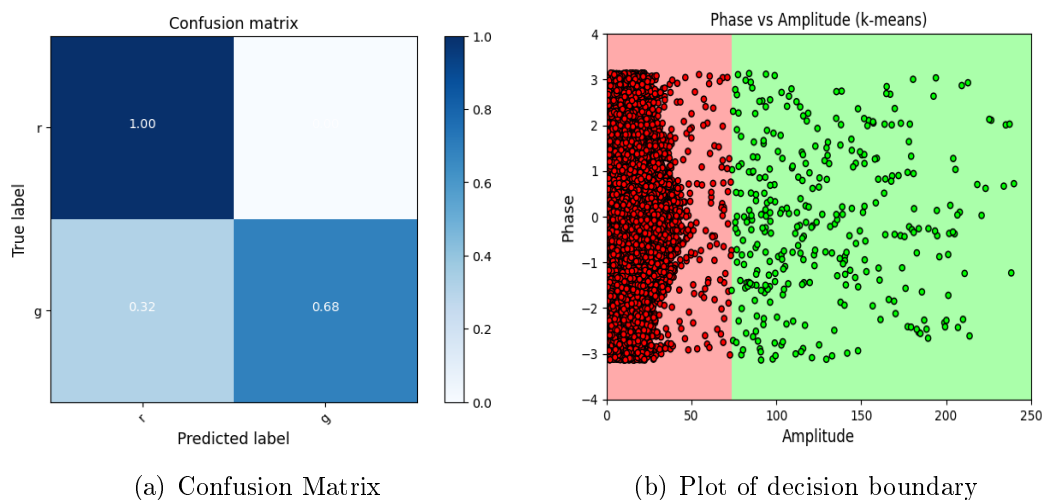


Figure 5.29: The confusion matrix and decision boundary plots for the SNR combination 5 dB and -17 dB for the k -means classifier

combination is overshadowed by a drop in its TP value for that same combination (as noted in Section 5.6.7). In contrast with the other methods, it is much better at detecting RFI when the RFI is weak, but under those conditions it loses the ability to identify a true visibility when it is given one. It sets to high a threshold when the RFI is strong. A similar trend is observable for the GMM classifier, albeit to a much lower degree.

All methods, are less capable of detecting RFI as the RFI becomes weaker, which is to be expected. The only exception being k -means (and to a lesser

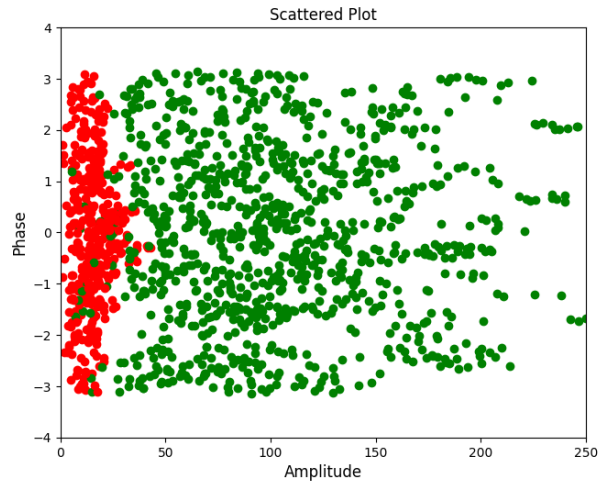


Figure 5.30: Scatter plot of the k -means classifier for the SNR combination 5 dB and -17 dB

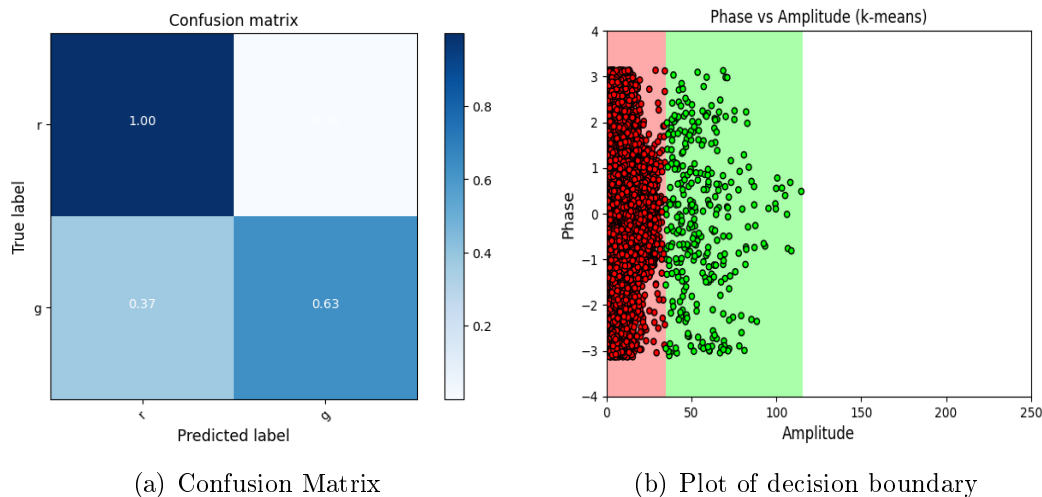


Figure 5.31: The confusion matrix and decision boundary plots for the SNR combination 10 dB and -13 dB for the k -means classifier

degree GMM). Having said this, k -means is still performing the worst. When the RFI is strong it struggles to detect it but when it is weak it detects it well, but in that case it is no longer able to detect uncorrupted visibilities accurately (and on average has a lower accuracy than the other methods).

5.7 Discussions, Summary and Conclusion

The following general conclusions can be drawn:

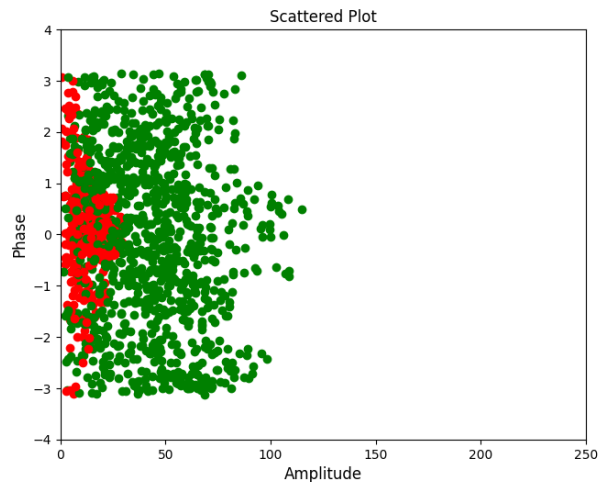


Figure 5.32: Scatter plot of the k -means classifier for the SNR combination 10 dB and -13 dB

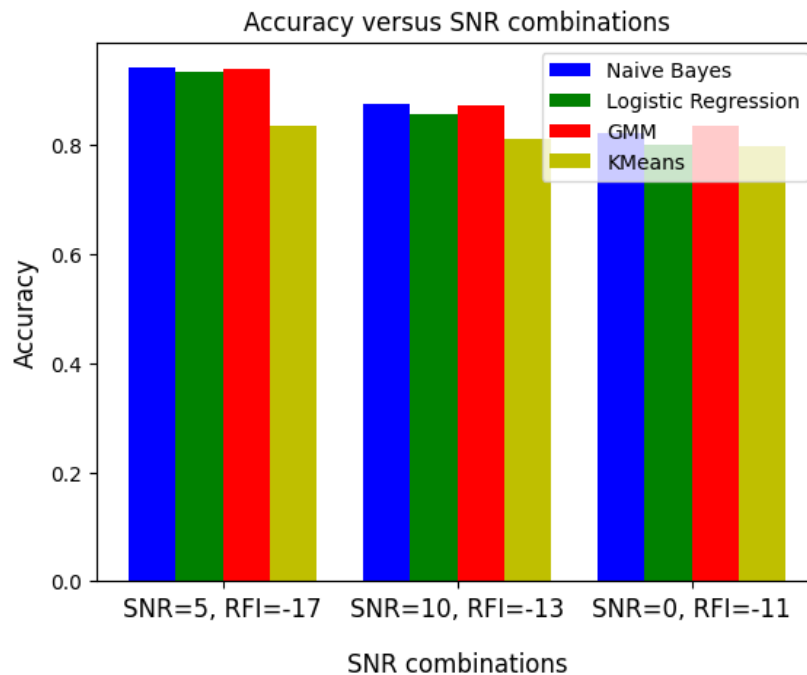


Figure 5.33: A graph of average accuracies of all classifiers versus the SNR combinations

1. TP: Most methods can identify when a visibility is real or not, this degrades when the RFI becomes weaker.
2. TN: Most methods can identify RFI well. This degrades as the RFI be-

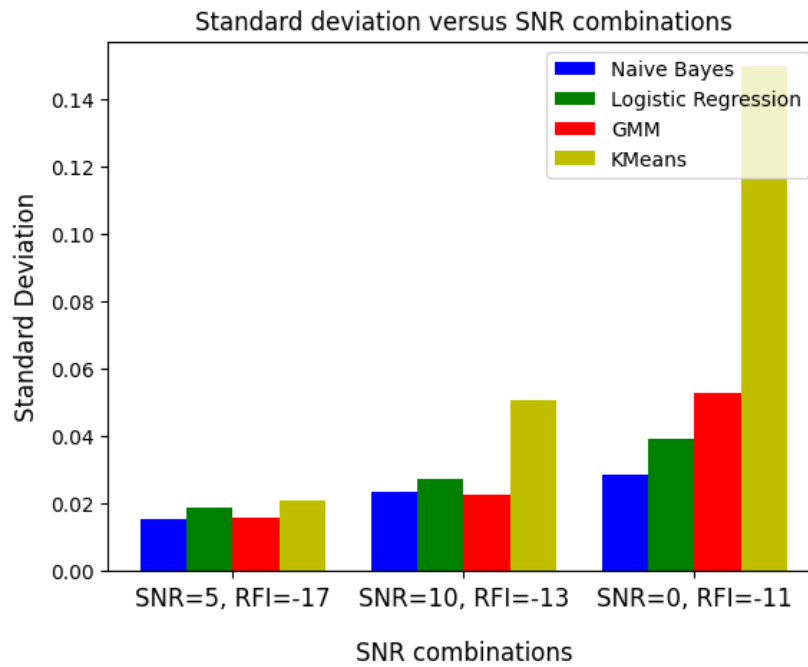


Figure 5.34: A graph of average standard deviation of all classifiers versus the SNR combinations

comes weaker. The most notable exception is *k*-means because of a drop in its TP value for the $\{0 \text{ dB}, -11 \text{ dB}\}$ SNR combination. However, it remains the weakest method with the lowest accuracy of all the methods (the standard deviation results corroborate this and shows that GMM also follows this trend).

3. Accuracy: The accuracy of each method is reducing as the RFI SNR value increases. Accuracy improves as the RFI SNR value changes from -11 to -13 to -17; which is to be expected.
4. The supervised classification methods did better than the unsupervised methods; which is to be expected.
5. The two unsupervised methods performed on par (although it seems that Naive Bayes slightly outperformed the Logistic Regression method possibly due to the fact that it is a non-linear method, whilst Logistic Regression is a linear method).
6. Non-linear methods outperformed linear methods; which is to be expected.
7. The supervised methods produced more consistent results compared to the unsupervised methods.

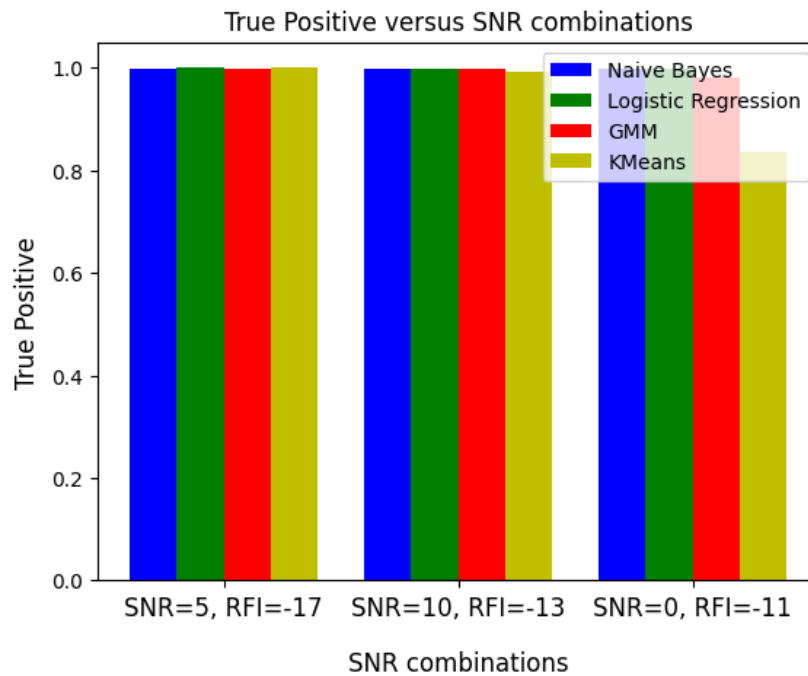


Figure 5.35: A graph of average true positives of all classifiers versus the SNR combinations

- The ranking of the four algorithms according to the accuracy graph is: Naive Bayes, GMM, Logistic Regression and k -means. Due to the fact that GMM had unstable results, the aforementioned ranking is changed to: Naive Bayes, Logistic Regression, GMM and k -means.

Recall that the main aim of this simulator is for it to be used as a teaching tool. The conclusions that were drawn from this study, therefore, may differ to what one might observe if these approaches were applied to real world data.

5.8 Teaching Tool

This section focuses on how this simulator can be utilized as a teaching tool. It was created to help teach students the basics of interferometry and machine learning. Table 5.7 lists all the main concepts a student can master by using the created pipeline. It also lists the relevant python file which should be studied to master these concepts.

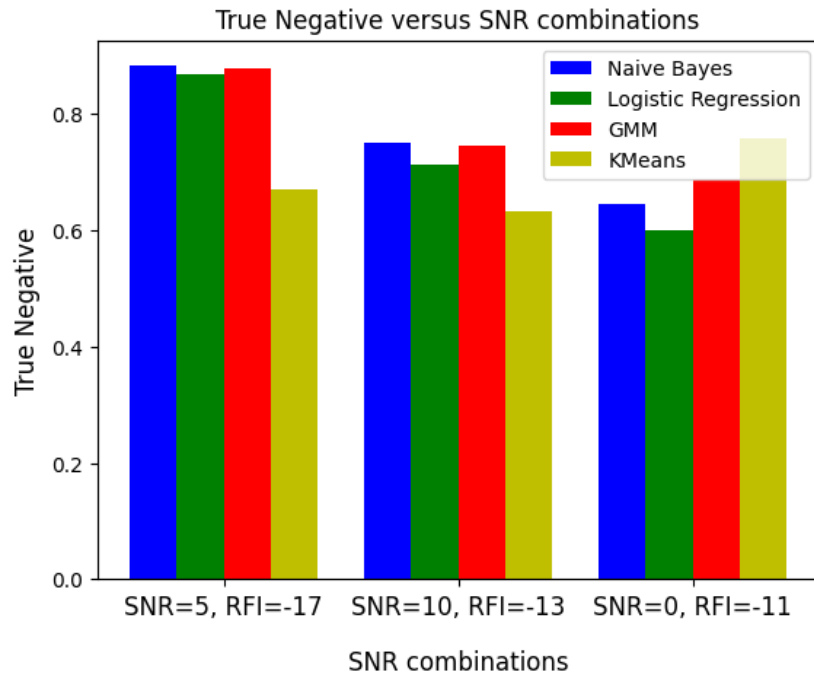


Figure 5.36: A graph of average true negatives of all classifiers versus the SNR combinations

Teaching tool	Location in Github repo
How to convert coordinates to radians	coordinate_conversion.ipynb
How to create <i>uv</i> -tracks	uv-track.ipynb
How to create sources	source_creation.ipynb
How to create visibilities	visibility.ipynb
How to add noise to visibilities	noise_visibility.ipynb
How to add RFI to visibilities	RFI_visibilities.ipynb
How to perform classification	classification_parameters.ipynb
How to generate classification plots	classification_plottings.ipynb

Table 5.7: Table referring to the teaching tool consisting of locations of python files in the github repository where one can go to learn how this simulator was created.

Chapter 6

Conclusion and Recommendation

6.1 Conclusion

As stated in Section 1.2 of Chapter 1, the thesis had three main objectives. The creation of an RFI simulator to serve as a teaching tool, a succinct review of the theory underpinning interferometry and ML; and the utilization of the aforementioned simulator to rank four classification algorithms. Let us see if these three objectives were achieved.

The main contribution of this work is a new inductive RFI simulator. The details of this simulator was presented in Chapter 5. This simulator is multi-faceted and was designed to serve as both a teaching tool and a testbed for ML algorithms. Students can use this simulator to learn the basics of interferometry and ML. The simulator can generate visibilities from a skymodel, create uv -tracks, induce RFI onto said visibilities and employ ML to detect the aforementioned RFI. The simulator was validated in Chapter 5. How the simulator can be used to learn specific concepts is made clear in Section 5.3. The theory underpinning interferometry and ML is thoroughly presented in Chapter 2 and Chapter 4 respectively. An ML experiment which can be repeated by a student wanting to learn more about ML is presented in Section 4.10. Moreover, a naive initial accuracy ranking of the four classification algorithms namely Naive Bayes, Logistic Regression, GMM and k -means in terms of using them to detect RFI is presented in Section 5.7 of Chapter 5. It was observed that the accuracies of the algorithms decreased as the RFI SNR value increased. This implies that, the more negative the RFI SNR value, the more the RFI noise level. In short, the thesis clearly achieved all three objectives.

To conclude, a simplistic approach was used and the outcome thereof was a teaching and training tool.

6.2 Recommendation for Future Studies

The limitations of the simulator is clearly outlined in Section 5.4. Each of these limitations should be improved upon in future work. Moreover, only four relatively basic ML algorithms were ranked in this study. As discussed in Chapter 3, there exists a multitude of ML algorithms which can be employed to detect RFI. Once the simulator presented in this thesis is improved to an extent that it can generate realistic RFI effects, then it can be used as a testbed for all the algorithms presented in Chapter 4. This, however, is beyond the scope of the current work. A study to evaluate the effectiveness of the developed teaching tool should also be conducted as part of a future endeavour.

List of References

- [1] Thompson, A., Moran, J. and Swenson, G.: In: *Interferometry and Synthesis in Radio Astronomy, Third Edition*. Astronomy and Astrophysics Library, SpringerOpen, 2017.
- [2] Goodfellow, I., Bengio, Y. and Courville, A.: In: *Deep Learning*. MIT Press, 2016. Available at: https://books.google.co.za/books/about/Deep-Learning.html?id=Np9SDQAAQBAJ&source=kp_book_description&redir_esc=y.
- [3] Géron, A.: In: *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition*. O'Reilly Media, Inc., 2019. Available at: <https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/>.
- [4] Smirnov, O. *et al.*: Fundamentals of radio interferometry and aperture synthesis. Available at: https://github.com/ratt-ru/fundamentals_of_interferometry.
- [5] Schollar, C.: *RFI monitoring for the MeerKAT Radio Telescope*. Masters, University of Cape Town, 2015. Available at: <https://core.ac.uk/download/pdf/185420039.pdf>.
- [6] Desk, S.S.: RFI (L-band). 2020. Available at: <https://skaafrica.atlassian.net/servicedesk/customer/portal/1/topic/bc9d6ad2-8321-4e13-a97a-d19d6d019a1c/article/305332225>.
- [7] SRAO: Meerkat radio telescope. 2021. Available at: <https://www.srao.ac.za/gallery/meerkat/>.
- [8] Purcell, C.R. and Truelove, R.: A virtual radio interferometer application. 2017. Available at: <https://crpurcell.github.io/friendlyVRI/>.
- [9] Marti-Vidal, I.: APSYNSIM: An interactive tool to learn interferometry. *arXiv: Instrumentation and Methods for Astrophysics*, 2017. Available at: <https://arxiv.org/pdf/1706.00936.pdf>.
- [10] Avison, A. and George, S.J.: A graphical tool for demonstrating the techniques of radio interferometry. *European Journal of Physics*, vol. 34, no. 1, pp. 7–17, 2012. Available at: <https://arxiv.org/pdf/1211.0228.pdf>.

- [11] Finlay, C.: *Radio Frequency Interference: Simulations for Radio Interferometry Arrays*. Masters, University of Cape Town, 2020. Available at: <https://open.uct.ac.za/handle/11427/33716>.
- [12] Joel, A., Sebastian, S., Chihway, C., Christian, M., Adam, A. and Alexandre, R.: HIDE & SEEK: End-to-end packages to simulate and process radio survey data. *Astron. Comput.*, vol. 18, pp. 8–17, 2017. Available at: <https://arxiv.org/pdf/1607.07443.pdf>.
- [13] Aguirre, J. *et al.*: Validation of the HERA Phase I Epoch of Reionization 21 cm Power Spectrum Software Pipeline. 2021. Available at: <https://arxiv.org/pdf/2104.09547.pdf>.
- [14] ASTRON: What is radio astronomy? 2021. Available at: <https://www.astron.nl/education/what-is-radio-astronomy/>.
- [15] Kraus, J.D.: Radio astronomy. In: *Radio Astronomy 2nd edition*. Cygnus-Quasar Books, 1986.
- [16] Ling, S., Sanny, J. and Moebs, W.: The michelson interferometer. In: *University Physics*, vol. 3. 2016. Available at: <https://opentextbc.ca/universityphysicsv3openstax/chapter/the-michelson-interferometer/>.
- [17] National Radio Astronomy Observatory: 20th anniversary of the VLA. 2021. Available at: <https://public.nrao.edu/news/20th-anniversary-of-the-vla/>.
- [18] National Radio Astronomy Observatory: Radio telescopes and radiometers. 2018. Available at: <https://www.cv.nrao.edu/~sransom/web/Ch3.html>.
- [19] King, B.: Right ascension and declination: Celestial coordinates for beginners. 2019. Available at: <https://skyandtelescope.org/astronomy-resources/right-ascension-declination-celestial-coordinates/>.
- [20] Wijnholds, S., Grobler, T. and Smirnov, O.: Calibration artefacts in radio interferometry - II. ghost patterns for irregular arrays. *Monthly Notices of the Royal Astronomical Society*, vol. 457, no. 3, pp. 2331–2354, 2016. Available at: <https://academic.oup.com/mnras/article/457/3/2331/2588910>.
- [21] Grobler, T., Nunhokee, C., Smirnov, O., van Zyl, A. and de Bruyn, A.: Calibration artefacts in radio interferometry - I. ghost sources in westerbork synthesis radio telescope data. *Monthly Notices of the Royal Astronomical Society*, vol. 439, no. 4, pp. 4030–4047, 2014. Available at: <https://academic.oup.com/mnras/article/439/4/4030/1174814>.
- [22] Square Kilometre Array: 3rd Generation calibration - SKA enews. 2015. Available at: <https://india.skatelescope.org/3rd-generation-calibration-skaenews-july2015/>.

- [23] Tasse, C.: Non-linear Kalman filters for calibration in radio interferometry. *Astronomy & Astrophysics*, vol. 566, no. 11, p. A127, 2014. Available at: https://www.aanda.org/articles/aa/full_html/2014/06/aa23503-14/aa23503-14.html.
- [24] Smirnov, O.: Revisiting the radio interferometer measurement equation. *Astronomy & Astrophysics*, vol. 527, p. A106, 2011. Available at: <http://dx.doi.org/10.1051/0004-6361/201016082>.
- [25] National Radio Astronomy Observatory: Radio frequency interference. 2021. Available at: <https://public.nrao.edu/telescopes/radio-frequency-interference/>.
- [26] Howell, E.: What is a satellite. 2020. Available at: <https://www.space.com/24839-satellites.html>.
- [27] UNITED NATIONS: Global navigation satellite systems (GNSS). 2021. Available at: <https://www.unoosa.org/oosa/en/ourwork/psa/gnss/gnss.html>.
- [28] Offringa, A., de Bruyn, A., Biehl, M., Zaroubi, S., Bernardi, G. and Pandey, V.: Post-correlation radio frequency interference classification methods. *Monthly Notices of the Royal Astronomical Society*, vol. 405, no. 1, pp. 155–167, 2010. Available at: <https://academic.oup.com/mnras/article/405/1/155/1020990>.
- [29] Alireza, V.S., Bassett, B.A., Oozeer, N., Fantaye, Y. and Finlay, C.: Deep learning improves radio frequency interference classification. *Monthly Notices of the Royal Astronomical Society*, vol. 499, no. 1, pp. 379–390, 2020. Available at: <https://arxiv.org/pdf/2005.08992.pdf>.
- [30] Barnbaum, C. and Bradley, R.F.: A new approach to interference excision in radio astronomy: Real-time adaptive cancellation. *The Astronomical Journal*, vol. 116, pp. 2598–2614, 1998. Available at: <https://iopscience.iop.org/article/10.1086/300604/fulltext/>.
- [31] Offringa, A.: AOflogger: RFI software. Astrophysics Source Code Library, 2010.
- [32] Sekhar, S. and Athreya, R.: Two procedures to flag radio frequency interference in the uv plane. *The Astronomical Journal*, vol. 156, no. 1, p. 9, 2018. Available at: <https://iopscience.iop.org/article/10.3847/1538-3881/aac16e>.
- [33] Mosiane, O., Oozeer, N., Aniyani, A. and Bassett, B.A.: Radio frequency interference detection using machine learning. vol. 198, pp. 1–2, 2017. Available at: <https://iopscience.iop.org/article/10.1088/1757-899X/198/1/012012>.
- [34] Grobler, T., Bernardi, G., Kenyon, J., Parsons, A. and Smirnov, O.: Redundant interferometric calibration as a complex optimization problem. *Monthly Notices*

- of the Royal Astronomical Society*, vol. 476, no. 2, pp. 2410–2420, 2018. Available at: <https://ui.adsabs.harvard.edu/abs/2018MNRAS.476.2410G>.
- [35] Alpaydin, E.: In: *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*, Second Edition. MIT Press, 2010.
- [36] Eisler, S. and Meyer, J.: Visual Analytics and Human Involvement in Machine Learning. *ArXiv*, vol. abs/2005.06057, 2020. Available at: <https://arxiv.org/pdf/2005.06057.pdf>.
- [37] Marsland, S.: In: *Machine Learning: An Algorithmic Perspective*, Chapman & Hall/CRC Machine Learning & Pattern Recognition Series, Second Edition. CRC Press, 2009.
- [38] Leopord, H., Cheruiyot, D.W.K. and Kimani, D.S.: A survey and analysis on classification and regression data mining techniques for diseases outbreak prediction in datasets. *The International Journal of Engineering and Sciences (IJES)*, vol. 5, no. 9, pp. 2319–1805, 2016. Available at: <https://www.javatpoint.com/regression-vs-classification-in-machine-learning>.
- [39] Shobha, G. and Rangaswamy, S.: Chapter 8 - machine learning. In: Gudivada, V.N. and Rao, C. (eds.), *Computational Analysis and Understanding of Natural Languages: Principles, Methods and Applications*, vol. 38 of *Handbook of Statistics*, pp. 197–228. Elsevier, 2018. Available at: <https://doi.org/10.1016/bs.host.2018.07.004>.
- [40] Barber, D.: In: *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.
- [41] Bishop, C.M.: In: *Pattern Recognition and Machine Learning (Information Science and Statistics)*, First Edition. Springer New York, 2007.
- [42] Walker, S. and Duncan, D.: Estimation of the probability of an event as a function of several independent variables. *Biometrika*, vol. 54, no. 1–2, pp. 167–179, 1967. Available at: <https://doi.org/10.1093/biomet/54.1-2.167>.
- [43] MacQueen, J.: Some methods for classification and analysis of multivariate observations. vol. 5.1, pp. 281–297. 1967. Available at: <http://www.cs.cmu.edu/~bhiksha/courses/mlsp.fall2010/class14/macqueen.pdf>.
- [44] Amendola, C., Faugere, J.-C. and Sturmfels, B.: Moment varieties of gaussian mixtures. *Journal of Algebraic Statistics*, vol. 7, no. 1, 2016. Available at: <http://dx.doi.org/10.18409/jas.v7i1.42>.
- [45] Grobler, T., Kleynhans, W., Salmon, B. and Burger, C.: Unsupervised sequential classification of MODIS time-series. In: *IGARSS 2020 IEEE International Geoscience and Remote Sensing Symposium*, pp. 2244–2247. 2020.

- [46] Andrews, D. and Herzberg, A.: Iris Data. In: *Data*, pp. 5–8. Springer, New York, NY, 1985. Available at: https://doi.org/10.1007/978-1-4612-5098-2_2.
- [47] Beazley, D. and Jones, B.: Python Cookbook. In: *Recipes for Mastering Python*. O'Reilly Media, 3rd edition, 2013.
- [48] Pedregosa, F. *et al.*: Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. Available at: https://en.wikipedia.org/wiki/Pareto_distribution.
- [49] Grobler, T.L.: *Sequential and Non-sequential hypertextual Classification and change Detection of MODIS time-series*. PhD, University of Pretoria, 2012. Available at: <https://repository.up.ac.za/bitstream/handle/2263/25427/Complete.pdf?sequence=7&isAllowed=y>.
- [50] Hussain, S., Bhatti, S., Ahmad, T., Aftab, M. and Tahir, M.: Parameter estimation of Pareto Distribution: Some modified moment estimators. *Maejo International Journal of Science and Technology*, vol. 12, no. 1, pp. 11–27, 2018. Available at: https://en.wikipedia.org/wiki/Pareto_distribution.