

RADAR PACKET ANALYSER

Report submitted in partial fulfilment of the requirement for the degree of

Your Degree

In

Your Degree Specialization

By

YOUR NAME

To

Your College/University logo and address

20__-__

DECLARATION

I hereby state that the project entitled “RADAR Packet Analyser” is an authentic work under the guidance of Mr/Ms.**Instructor** of **Org** for the fulfilment of the project by me, **Name**, a ___ year student of **YOUR DEGREE** of **YOUR SPECIALIZATION**.

NAME

Blank Page for pasting Certificate

ACKNOWLEDGEMENTS

It is with a sense of gratitude, I acknowledge the efforts of the entire team of well-wishers who have in some way or other contributed in their own special ways to the success and completion of this summer training project.

First of all, I would like to thank my summer training instructor Mr. **Instructor, Org** for his valuable guidance and mentorship which helped me a lot in this project. He was thorough with all the concepts and dedicated towards teaching. Without his guidance it would have been impossible to complete this project.

Further, I sincerely express my thanks to our lecturers for their valuable guidance and support. Last but not the least, I am indebted to my family and friends for their support.

Name

Enrolment No.:

ABSTRACT

This project demonstrates the decoding of data in the packets which were received by RADAR systems and then predicting the values of coordinates for the aircrafts. RADAR is a detection system that uses radio waves to determine the range, angle, or velocity of objects. It can be used to detect aircraft, ships, spacecraft, guided missiles, motor vehicles, weather formations, and terrain. A radar system consists of a transmitter producing electromagnetic waves in the radio or microwaves domain, a transmitting antenna, a receiving antenna (often the same antenna is used for transmitting and receiving) and a receiver and processor to determine properties of the object(s). Radio waves (pulsed or continuous) from the transmitter reflect off the object and return to the receiver, giving information about the object's location and speed.

In this project, the packets are stored in .pcap files and the data read and decoded is stored in a Comma Separated Values (.csv) file. The aim of storing the output data in the .csv file is to make future analytic operations on the data feasible. The packets contained in the .pcap file contains packets received by RADAR systems from aircrafts. Data in these packets is stored in the Category-48 format specified by European Organisation for the Safety of Air Navigation.

The data stored in the csv file is used to predict the values of coordinates of aircraft using machine learning.

The code is written in Python 3 and the project can be run from python console.

CONTENTS

Declaration.....	ii
Abstract.....	v
Chapter 1: Project Overview.....	1
1.1. Introduction.....	1
1.2. Need & Objective	2
1.3. Methodology	3
1.4. Software Used.....	4
1.5. About Organisation.....	11
Chapter 2: Project Design	12
2.1. Dataframe.....	12
2.2. Packet.....	13
2.3. Linear Regression	14
Chapter 3: Implementation	16
3.1. Reading .Pcap File	16
3.2. Decoding The Messages And Writing Into File	17
3.3. Applying Regression.....	20
Chapter 4: Result & Discussion.....	22
Chapter 5: Future Scope & Conclusion	28
References.....	24

CHAPTER 1

PROJECT OVERVIEW

1.1. INTRODUCTION

This project takes a file containing data packets received by RADAR systems. This file is in .pcap format. The packets contained by this file comprise of messages sent by various aircrafts to the RADAR. These messages are in the ASTERIX format specified by the European Organisation for the Safety of Air Navigation used for Transmission of Monoradar Target Reports. This format describes the message structure for the transmission of monoradar target reports from a radar station (conventional Secondary Surveillance Radar (SSR), monopulse, Mode S, conventional primary radar or primary radar using Moving Target Detection (MTD) processing), to one or more Surveillance Data Processing (SDP) Systems. This project only focuses on packets which follow Category-48 specification of the ASTERIX format.

The data in packets contains various fields like polar coordinates of the aircraft, its Mode-3/A code, time of day at which the packet was transmitted etc. Values for all these fields are decoded from the packets and are stored in a data structure and also written in a Comma Separated Values file. Then, the fetched and decoded values are used to train a linear regression model to predict further values of polar coordinates and visualized using scatter plots to show the projected trajectory of every aircraft.

1.2.NEED & OBJECTIVE

The objective of this project is automate the analysis of packets received from RADAR systems. Though there are tools like Wireshark available for reading the contents of packets stored in .pcap files, the same cannot be used for applying analysis on the data and getting valuable insights. Also tools like Wireshark do not specifically tell the significance of any bits for the corresponding field in the message as they only tells the bit-value for a parameter according to the message format but do not tell what that bit-value means with respect to that particular field. Also, this project also sorts all the messages/records according to the unique identification code of the aircrafts and predict the values for their polar coordinates using machine learning regression techniques and hence tries to predict their trajectory. Storing the packet data in csv format makes it a lot more feasible to operate upon especially for data science purposes. Comma Separated Values can be straight away read with functions of Pandas library of Python and can be read manually using a spreadsheet processor like MS-Excel as well. Predictions about the values of polar coordinates of aircrafts are also made using a linear regression model and displayed through scatter plots. These predictions are useful for visualizing the projected trajectory of the aircraft for better understanding. This project only takes name of file containing packets and file in which records are to be stored as inputs and stores and visualizes the data in a very easy to understand, thus allowing even a user from non-computer science background to use it as well.

1.3.METHODOLOGY

The project takes the name of the target .pcap file containing the packets received from RADAR systems and the name of Comma Separated Values file in which the values for various fields of the messages are to be written as inputs from the user. The name of .pcap file to be processed is input by the user. The file is then opened in read-binary mode. Function of Pypcapfile library is then used to read data and metadata from the packets. File object is passed as parameter to this function. Packet metadata consists of header, packet length, capture length and timestamp. Metadata of the packet extends upto 42 bytes, packet data starts from 43rd byte. Packet data is read as a stream of hexadecimal which are then converted to decimal integer or stream of binary bits based on field of the message being read. Category of the packet is read and checked to be 48, if so, then other fields like packet data length, FSPEC are decoded from the packet data. Based upon the values of different bits of FSPEC, it is decided whether the corresponding field such as target report descriptor, time of day, polar coordinates, aircraft code etc. exists in the message or not. If it exists, number of bits equal to the length of that field are read and stored as an integer or stream of bits in a list. A separate list is used for each field, if field is present in the message its value is appended to that list else an empty string is appended to maintain the size and indices of all lists equal. Many fields in the message are of variable length which can be found out using the field extension which is the last bit in that field. If it is 0, the field is not extended, if it's 1, then the field is extended. Also a single packet can contain

Finally all lists are combined to form a dataframe and written to a .csv file. The dataframe is checked for all unique values of Mode-3/A Code which can be used to uniquely identify an aircraft. Dataframe is split into multiple based on the value of Mode-3/A code. These parts of dataframe are then sorted on the basis of the value of *time of day field*. The values of polar coordinates of these sorted dataframes are used to predict further values by applying linear regression. The *time of day* field is used as feature while polar coordinates as target for regression.

1.4.SOFTWARE USED

1.4.1. PYTHON

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter, map, and reduce functions; list comprehensions, dictionaries, sets and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

1.4.2. NUMPY

NumPy is the fundamental package for scientific computing with Python.

Some of its features are:

- i. a powerful N-dimensional array object
- ii. sophisticated (broadcasting) functions

- iii. tools for integrating C/C++ and Fortran code
- iv. useful linear algebra, Fourier transform, and random number capabilities
- v. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.
- vi. NumPy is based on two earlier Python modules dealing with arrays. One of these is Numeric. Numeric is like NumPy a Python module for high-performance, numeric computing, but it is obsolete nowadays. Another predecessor of NumPy is Numarray, which is a complete rewrite of Numeric but is deprecated as well. NumPy is a merger of those two, i.e. it is built on the code of Numeric and the features of Numarray. Numpy array is used to store values of predictions of polar coordinates in this project.

1.4.3. PANDAS

pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language.

Pandas is the most popular python library that is used for data analysis. It provides highly optimized performance with back-end source code is purely written in C or Python. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables.

There are several ways to create a DataFrame. One way way is to use a dictionary. Another way to create a DataFrame is by importing a csv file using Pandas. Now, the csv cars.csv is stored and can be imported using `pd.read_csv`. There are several ways to index a Pandas DataFrame. One of the easiest ways to do this is by using square bracket notation.

In the example below, you can use square brackets to select one column of the cars DataFrame. You can either use a single bracket or a double bracket. The single bracket with output a Pandas Series, while a double bracket will output a

Pandas DataFrame. Square brackets can also be used to access observations (rows) from a DataFrame. You can also use `loc` and `iloc` to perform just about any data selection operation. `loc` is label-based, which means that you have to specify rows and columns based on their row and column labels. `iloc` is integer index based, so you have to specify rows and columns by their integer index like you did in the previous exercise.

1.4.4. PYPCAPFILE

`pypcapfile` is a pure Python library for handling `libpcap` savefiles. The core functionality is implemented in `pcap.savefile`. It currently supports very basic support for Ethernet frames and IPv4 packet parsing. `Packet` is an attribute of the object returned by the `load_savefile` function of the `savefile`. It itself has six further attributes, namely, `header`, `timestamp`, `timestamp_us`, `capture_len`, `packet_len` and `packet`. The `packet` attribute returns the actual data of the packet and contains values in hexadecimal format, rest return the metadata of the packet

1.4.5. SCIKIT-LEARN

Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. Scikit-learn is largely written in Python, with some core algorithms written in Cython to achieve performance. Support vector machines are implemented by a Cython wrapper around LIBSVM; logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR.

Scikit-learn is probably the most useful library for machine learning in Python. It is on NumPy, SciPy and matplotlib, this library contains a lot of efficient tools for machine learning and statistical modelling including classification, regression and clustering and dimensionality reduction.

Scikit-learn comes loaded with a lot of features. Few of them are:

- i. **Supervised learning algorithms:** Starting from Generalized linear models (e.g Linear Regression), Support Vector Machines (SVM), Decision Trees to

Bayesian methods – all of them are part of scikit-learn toolbox. The spread of algorithms is one of the big reasons for high usage of scikit-learn.

- ii. **Cross-validation:** There are various methods to check the accuracy of supervised models on unseen data
- iii. **Unsupervised learning algorithms:** There is a large variety of algorithms in the offering – starting from clustering, factor analysis, and principal component analysis to unsupervised neural networks.
- iv. **Sample datasets:** This is useful handy while learning scikit-learn and building models.
- v. **Feature extraction:** Useful for extracting features from images and text

1.4.6. MATPLOTLIB

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc. Matplotlib comes with a wide variety of plots. Plots helps to understand trends, patterns, and to make correlations. They're typically instruments for reasoning about quantitative information.

NumPy is the fundamental package for scientific computing with Python.

Some of its features are:

- i. useful linear algebra, Fourier transform, and random number capabilities
- ii. Pyplot is a matplotlib module which provides a MATLAB-like interface. Matplotlib is designed to be as usable as MATLAB, with the ability to use Python, and the advantage of being free and open-source.
- iii. It supports a very wide variety of graphs and plots namely - histogram, bar charts, power spectra, error charts etc. It is used along with NumPy to provide an environment that is an effective open source alternative for MatLab. It can also be used with graphics toolkits like PyQt and wxPython.

1.4.7. JUPYTER

The Jupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access or can be installed on a remote server and accessed through the internet. Project Jupyter is a non-profit, open-source project, born out of the IPython Project in 2014 as it evolved to support interactive data science and scientific computing across all programming languages. Jupyter will always be 100% open-source software, free for all to use and released under the liberal terms of the modified BSD license.

Jupyter is developed in the open on GitHub, through the consensus of the Jupyter community.

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning etc.

A Jupyter Notebook document is a JSON document, following a versioned schema usually ending with the ".ipynb" extension. Jupyter Notebook provides the user with options to download the code as Python file (.py), Portable Document (PDF), HTML (.html), Markdown (.md), reST (.rst) or LaTeX (.tex). It also provides the user with feasibility of automatic saving of code at short intervals.

Jupyter Notebook provides a browser-based REPL built upon a number of popular open-source libraries:

- i. Python
- ii. ØMQ
- iii. Tornado (web server)
- iv. jQuery
- v. Bootstrap (front-end framework)
- vi. MathJax
- vii. Jupyter Notebook can connect to many kernels, (by default Jupyter Notebook ships with the IPython kernel) to allow programming in many languages.

1.4.8. WIRESHARK

Wireshark is a free and open-source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education. Originally named Ethereal, the project was renamed Wireshark in May 2006 due to trademark issues.

Wireshark is cross-platform, using the Qt widget toolkit in current releases to implement its user interface, and using pcap to capture packets; it runs on Linux, macOS, BSD, Solaris, some other Unix-like operating systems, and Microsoft Windows. There is also a terminal-based (non-GUI) version called TShark. Wireshark, and the other programs distributed with it such as TShark, are free software, released under the terms of the GNU General Public License. It is used in this project to verify the read contents of the packets manually. Wireshark is the world's foremost network protocol analyzer. It lets you see what's happening on your network at a microscopic level. It is the de facto (and often de jure) standard across many industries and educational institutions. Capturing raw network traffic from an interface requires elevated privileges on some platforms. For this reason, older versions of Ethereal/Wireshark and tethereal/TShark often ran with superuser privileges. Taking into account the huge number of protocol dissectors that are called when traffic is captured, this can pose a serious security risk given the possibility of a bug in a dissector. Due to the rather large number of vulnerabilities in the past (of which many have allowed remote code execution) and developers' doubts for better future development, OpenBSD removed Ethereal from its ports tree prior to OpenBSD 3.6.^[22] Elevated privileges are not needed for all operations. For example, an alternative is to run `tcpdump` or the *dumpcap* utility that comes with Wireshark with superuser privileges to capture packets into a file, and later analyze the packets by running Wireshark with restricted privileges. To emulate near realtime analysis, each captured file may be merged by *mergcap* into growing file processed by Wireshark. On wireless networks, it is possible to use the Aircrack wireless security tools to capture IEEE 802.11 frames and read the resulting dump files with Wireshark.

As of Wireshark 0.99.7, Wireshark and TShark run `dumpcap` to perform traffic capture. Platforms that require special privileges to capture traffic need only `dumpcap` run with those privileges. Neither Wireshark nor TShark need to or should be run with special privileges.

Wireshark development thrives thanks to the contributions of networking experts across the globe. It is the continuation of a project that started in 1998. Wireshark has a rich feature set which includes the following:

- i. Deep inspection of hundreds of protocols, with more being added all the time
- ii. Live capture and offline analysis
- iii. Standard three-pane packet browser
- iv. Multi-platform: Runs on Windows, Linux, OS X, FreeBSD, NetBSD, and many others
- v. Captured network data can be browsed via a GUI, or via the TTY-mode TShark utility
- vi. The most powerful display filters in the industry
- vii. Rich VoIP analysis
- viii. Read/write many different capture file formats: tcpdump (libpcap), Pcap NG, Catapult DCT2000, Cisco Secure IDS iplog, Microsoft Network Monitor, Network General Sniffer (compressed and uncompressed), Sniffer® Pro, and NetXray, Network Instruments Observer, NetScreen snoop, Novell LANalyzer, RADCOM WAN/LAN Analyzer, Shomiti/Finisar Surveyor, Tektronix K12xx, Visual Networks Visual UpTime, WildPackets EtherPeek/TokenPeek/AiroPeek, and many others
- ix. Capture files compressed with gzip can be decompressed on the fly
- x. Live data can be read from Ethernet, IEEE 802.11, PPP/HDLC, ATM, Bluetooth, USB, Token Ring, Frame Relay, FDDI, and others (depending on your platform)
- xi. Decryption support for many protocols, including IPsec, ISAKMP, Kerberos, SNMPv3, SSL/TLS, WEP, and WPA/WPA2
- xii. Coloring rules can be applied to the packet list for quick, intuitive analysis
- xiii. Output can be exported to XML, PostScript®, CSV, or plain text

1.5.ABOUT ORGANISATION

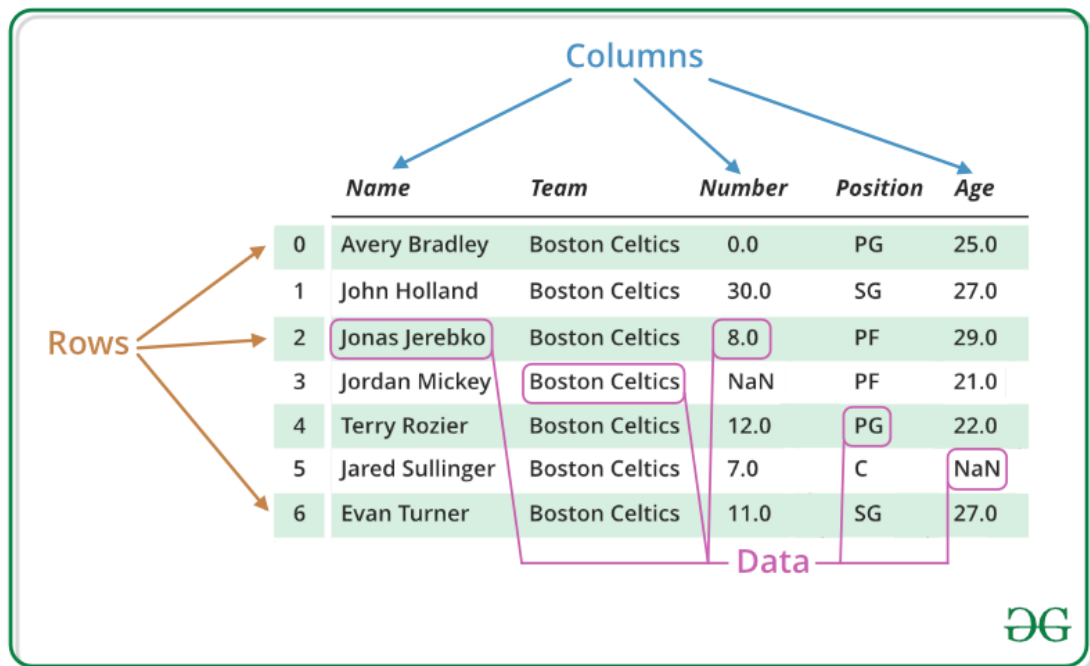
CHAPTER 2

PROJECT DESIGN

The source code of this project is completely written in Python. This section discusses about the few analogies required for the understanding of project:

2.1. DATAFRAME

Pandas DataFrame is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas DataFrame consists of three principal components, the data, rows, and columns.



2.1 A DataFrame

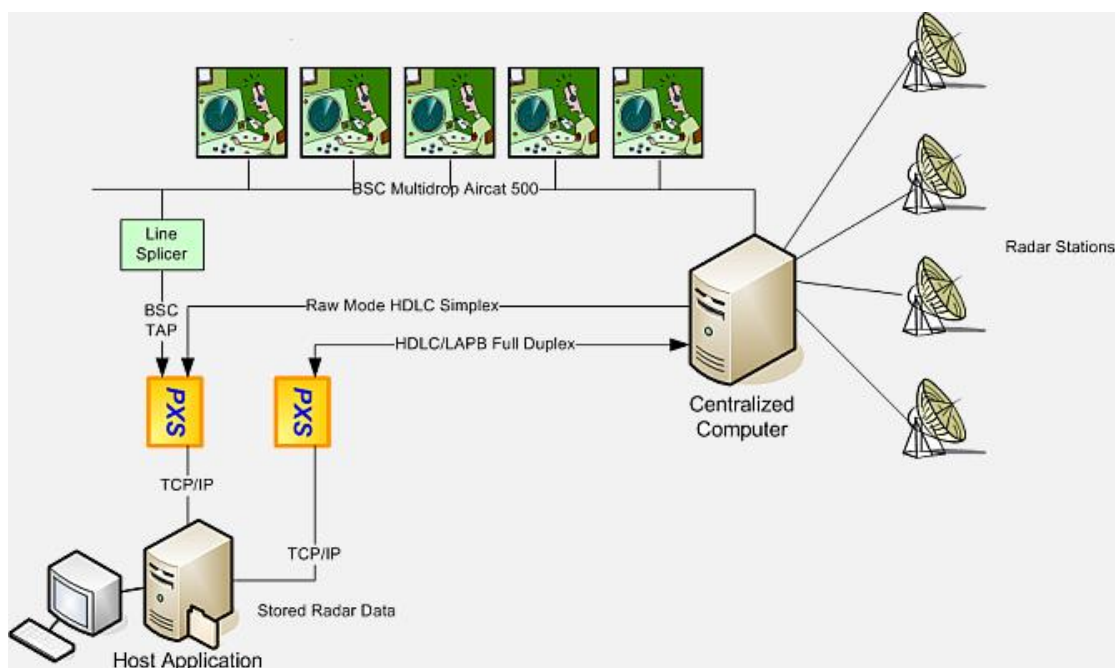
To create DataFrame from dict of narray/list, all the narray must be of same length. If index is passed then the length index should be equal to the length of arrays. If no index is passed, then by default, index will be range(n) where n is the array length.

Indexing a Dataframe using indexing operator `[]`: Indexing operator is used to refer to the square brackets following an object. The `.loc` and `.iloc` indexers also use the indexing operator to make selections. In this indexing operator to refer to `df[]`.

2.2. PACKET

It is an attribute of the object returned by the *load_savefile* function of the *savefile*. It itself has six further attributes, namely, *header*, *timestamp*, *timestamp_us*, *capture_len*, *packet_len* and *packet*. The *packet* attribute returns the actual data of the packet and contains values in hexadecimal format, rest return the metadata of the packet.

With or without compression, the size of a CAT-48 packet comprising header and data is likely to be several Kbytes long, perhaps 10s of Kbytes. Consideration must then be given to the delivery of such a large packet through standard IP protocols. The ASTERIX CAT-48 standard does not define the way that the packet is delivered, leaving this to the underlying transport protocol. In the common distribution method of UDP, a large packet size needs to be broken into smaller units each less than the packet size for the UDP protocol (called the maximum transmission unit or MTU). Although the transport process will, in theory, handle the delivery of large packets of data by automatically breaking them into smaller units and reassembling them on the other side, a problem exists in the IPv4 protocol which means the process would not work as expected in a practical implementation.



2.2 Path followed by a packet from RADAR to host application

No.	Time	Source	Destination	Protocol	Length	Info
604	22.806600	3comLtd_dd:fa:81	Broadcast	ARP	60	Who has 10.250.183.255? T
605	22.838664	CadmusCo_f6:e2:59	Broadcast	ARP	60	Who has 10.250.176.203? T
606	22.872101	10.250.176.238	10.250.183.255	BROWSER	243	Host Announcement ADM-PC,
607	22.888514	10.250.176.2	224.0.0.2	HSRP	62	Hello (state Active)
608	22.902613	10.250.180.0	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
609	22.914334	MitacInt_50:8d:f1	Broadcast	ARP	60	Who has 10.250.181.10? Te
610	23.080466	10.250.176.2	224.0.0.2	HSRP	62	Hello (state Active)
611	23.096641	10.110.32.66	10.250.180.106	UDP	62	1935→52148 Len=20
612	23.096858	10.250.180.106	10.110.32.66	RTCP	62	Application specific (
613	23.227174	LcfCHefe_bf:bd:e8	Broadcast	ARP	60	Who has 10.250.176.65? Te
614	23.288604	10.250.176.2	224.0.0.2	HSRP	62	Hello (state Active)
615	23.429380	fe80::9d74:1d2e:5f7...	ff02::1:2	DHCPv6	151	Solicit XID: 0x4329c CID:
616	23.449517	10.250.179.72	255.255.255.255	UDP	82	58288→1947 Len=40
617	23.474220	10.250.180.106	65.52.108.74	TLSv1.2	699	Application Data

Frame 2: 668 bytes on wire (5344 bits), 668 bytes captured (5344 bits) on interface 0
 Ethernet II, Src: AsustekC_2b:1b:81 (34:97:f6:2b:1b:81), Dst: All-HSRP-routers_48 (00:00:0c:07:ac:48)
 Internet Protocol Version 4, Src: 10.250.180.106, Dst: 95.213.4.212

```

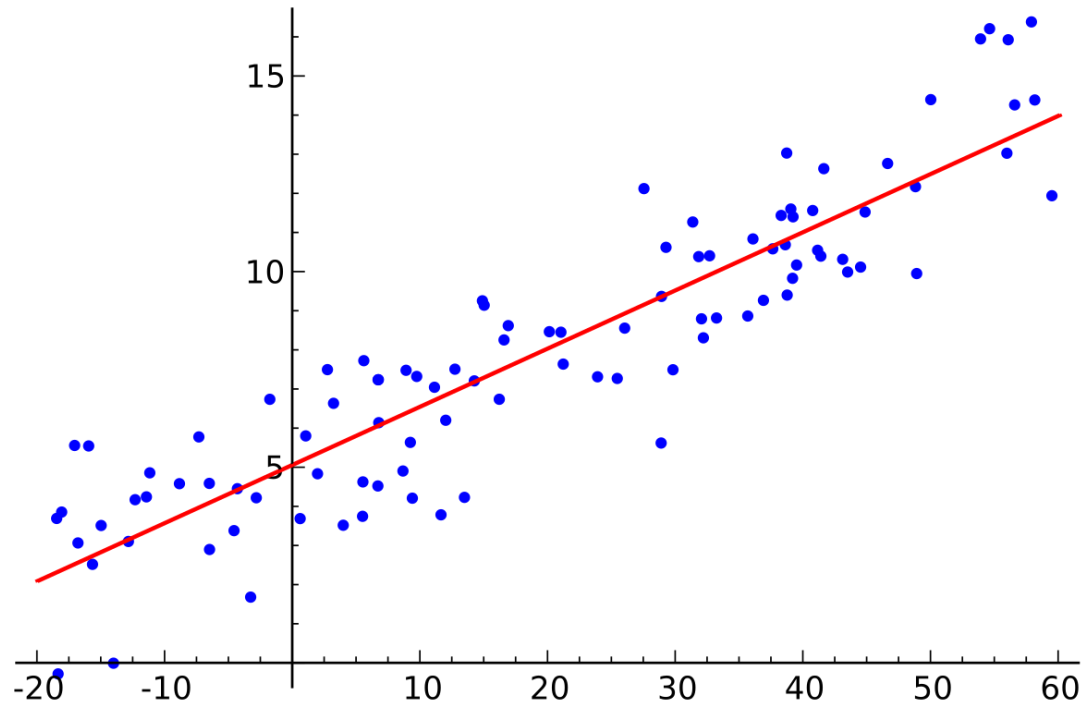
0000  00 00 0c 07 ac 48 34 97 f6 2b 1b 81 08 00 45 00  ....H4. .+...E.
0010  02 8e 36 43 40 00 80 06 00 00 0a fa b4 6a 5f d5  ..6C@... ..j_.
0020  04 d4 d1 dd 01 bb 19 34 2a 9f a3 bb 26 7a 50 18  ....4 *...&zP.
0030  01 00 26 8e 00 00 17 03 03 02 61 00 00 00 00 00  ..&..... .a.....
0040  00 02 53 2b e4 17 94 11 2b 4f 55 04 59 1d f1 0d  ..S+... +OU.Y...
0050  e3 3f 5b 2b 16 69 a2 a1 df b1 98 45 ff 98 b3 9d  .?[+.i.. ..E....
  
```

Ethernet: <live capture in progress> | Packets: 1030 · Displayed: 1030 (100.0%) | Profile: Default

2.3 Contents of a pcap file in wireshark

2.3. LINEAR REGRESSION

Linear regression is a linear approach to modelling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). The case of one explanatory variable is called simple linear regression. For more than one explanatory variable, the process is called multiple linear regression. This term is distinct from multivariate linear regression, where multiple correlated dependent variables are predicted, rather than a single scalar variable.



2.4 Linear Regression

In linear regression, the relationships are modelled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models. Most commonly, the conditional mean of the response given the values of the explanatory variables (or predictors) is assumed to be an affine function of those values; less commonly, the conditional median or some other quantile is used. Like all forms of regression analysis, linear regression focuses on the conditional probability distribution of the response given the values of the predictors, rather than on the joint probability distribution of all of these variables, which is the domain of multivariate analysis.

Linear regression was the first type of regression analysis to be studied rigorously, and to be used extensively in practical applications. This is because models which depend linearly on their unknown parameters are easier to fit than models which are non-linearly related to their parameters and because the statistical properties of the resulting estimators are easier to determine.

CHAPTER 3

IMPLEMENTATION

This section would discuss the steps involved in building the project. The project can be divided into three major parts: reading .pcap file, decoding messages and writing into file, and applying regression.

3.1. READING .PCAP FILE

Name of the .pcap file containing RADAR packets is input by the user and it is opened in read-binary mode. The file object is passed to *savefile.load_savefile* function which reads the packets in the file. This function returns an object which has an attribute *packets*, a list of all packets in the file. The packets can be read by iterating every packet in this packet list.

```
7 file=input('Enter pcap file name: ')
8 file=file+'.pcap'
```

3.1 Taking pcap file name as input

```
21 def process_pcap(file_name):
22     print('Opening {}'.format(file_name))
23     count = 0
24     testcap=open(file_name,'rb')
25     capfile=savefile.load_savefile(testcap,verbose=True)
```

3.2 Opening the file and creating savefile object

Empty lists are created corresponding to the fields of the message and packet metadata to be read and their values for every suitable packet are appended into the list.

```
37 header=[]
38 timestamp=[]
39 timestamp_us=[]
40 capture_len=[]
41 packet_len=[]
42 packet=[]
```

3.3 Creating empty lists to store metadata

```

57     sac=[]
58     sic=[]
59     time_of_day=[]
60     target_report_descriptor_typ=[]
61     target_report_descriptor_sim=[]
62     target_report_descriptor_rdp=[]
63     target_report_descriptor_spi=[]
64     target_report_descriptor_rab=[]
65     target_report_descriptor_fx=[]
66     polar_coord_x=[]
67     polar_coord_y=[]
68     mode3acode=[]

```

3.4 Creating empty lists to store message fields

3.2 DECODING THE MESSAGES AND WRITING INTO FILE

ASTERIX Category of every packet being read is checked, if it is found to be of Category -48, its metadata and message fields are appended to corresponding lists. As described earlier, the *packet* attribute return stream of hexadecimal values which are to be either converted to integer or into 4 binary bits one-by-one.

```

103     category=int(pkt.packet[84:86],16)
104     #print(category)
105     if category==48:
106         count += 1
107         header.append(pkt.header)
108         timestamp.append(pkt.timestamp)
109         timestamp_us.append(pkt.timestamp_us)
110         capture_len.append(pkt.capture_len)
111         packet_len.append(pkt.packet_len)
112         packet.append(pkt.packet)

```

3.5 Checking ASTERIX Category and appending metadata to lists

Whether a particular field is present in the message or not is defined by bits of the FSPEC, therefore it is to be converted to binary format. Though built-in functions provide the facility to convert hexadecimal to decimal and that to binary, the number of bits generated in output varies depending upon the input and also, the whole hexadecimal stream is considered as a single where this project requires each digit to be converted separately. Slicing the hexadecimal also does not give desired results. Hence, a function is required to convert each hexadecimal digit into exactly 4 binary bits.

```
9 def to_bits(a,i):
10     a=int(a[i:i+2],16)
11     b=bin(a%16)
12     b=b[2:]
13     while(len(b)!=4):
14         b='0'+b
15     c=bin(a//16)
16     c=c[2:]
17     while(len(c)!=4):
18         c='0'+c
19     return b+c
```

3.6 Converting hexadecimal into binary one-by-one

Value of every bit of FSPEC is checked, if it's found to be 1, it means corresponding field is present in the message, hence it is appended to the list and the iterator used for reading packet data is incremented, else the list is appended with empty string.


```

f1=to_bits(pkt.packet,90)
if f1[7]=='1':
    f2=to_bits(pkt.packet,92)
else:
    f2=''
if f2!='' and f2[7]=='1':
    f3=to_bits(pkt.packet,94)
else:
    f3=''
f=f1+f2+f3
#print('f={}'.format(f))
fspec1.append(f)
x=96
if f1[0]=='1':
    sac.append(int(pkt.packet[x:x+2],16))
    sic.append(int(pkt.packet[x+2:x+4],16))
    x=x+4
else:
    sac.append('|')
    sic.append('')
if f1[1]=='1':
    time_of_day.append(int(pkt.packet[x:x+6],16)/128)
    x=x+6
else:
    time_of_day.append('')

```

3.7 Checking FSPEC value to know about presence of corresponding fields

The lists are combined under a single dataframe and written into a csv file.

```

df['HEADER']=header
df['TIMESTAMP']=timestamp
df['TIMESTAMP_US']=timestamp_us
df['CAPTURE_LEN']=capture_len
df['PACKET_LEN']=packet_len
df['PACKET']=packet
df['CATEGORY']=cat
df['DATA_LENGTH']=length

```

3.8 Assigning a dataframe column for each list.

```

308 df.to_csv('{}'.format(csv))
309 print('{} contains {} CAT48 packets'.format(file_name, count))
310 print('Data and MetaData of packets saved in {}'.format(csv))
311 return df

```

3.9 Writing dataframe to .csv file

3.3 APPLYING REGRESSION

The dataframe obtained consists of data of multiple aircrafts. Therefore, to predict the trajectory of an aircraft, dataframe is to be split into parts such that each part consists of data of a single aircraft. It is done by obtaining a list of distinct Mode-3/A codes and then making list of all entries for each of that code. Finally these lists are sorted on the basis of *time of day* field and linear regression is applied using *time of day* as feature and polar coordinates as target. Standard deviation of *time of day* is calculated and its multiples are added to last (highest actual value) to calculate testing features.

```
316     l=df['MODE-3A_CODE'].unique()
317     for i in range(len(l)):
318         x=[]
319         y=[]
320         time=[]
321         for j in range(len(df['MODE-3A_CODE'])):
322             if l[i]==df['MODE-3A_CODE'].iloc[j] and df['TIME_OF_DAY'].iloc[j]!='':
323                 x.append(df['X_POLAR_COORDINATE'].iloc[j])
324                 y.append(df['Y_POLAR_COORDINATE'].iloc[j])
325                 time.append(df['TIME_OF_DAY'].iloc[j])
326         if len(x)>5:
327             d=pd.DataFrame()
328             d['x']=x
329             d['y']=y
330             d['t']=time
```

3.10 Splitting dataframe according to Mode-3/A code values of aircrafts

```
324     d=pd.DataFrame()
325     d['x']=x
326     d['y']=y
327     d['t']=time
328     print(d.head())
329     d=d.reset_index().sort_values('t',axis=0,ascending=True,na_position='last')
```

3.11 Sorting dataframe on the basis of *time of day*

```

340     t_std=d['t'].values.std()
341     x_t=np.zeros((8,1))
342     x_t[0]=d['t'].iloc[-1]+t_std
343     for i in range(1,8):
344         x_t[i]=x_t[i-1]+t_std

```

3.12 Calculating standard deviation of *time of day* field and adding it recursively to highest value to get testing features (base for prediction)

```

345     lr=LinearRegression()
346     lr.fit(d['t'].values.reshape(-1,1),d[['x','y']])

```

3.13 Creating a linear regression model using *time of day* as feature and polar coordinates as targets

```

347     pred=lr.predict(x_t.reshape(-1,1))

```

3.14 Predicting values of polar coordinates using the linear regression model with multiples of standard deviation added to *time of day* as feature

```

356     plt.scatter(x,y,c='red')
357     plt.scatter(pred[:,0],pred[:,1],c='green')
358     plt.show()

```

3.15 Plotting the actual and predicted values of polar coordinates using a scatter plot

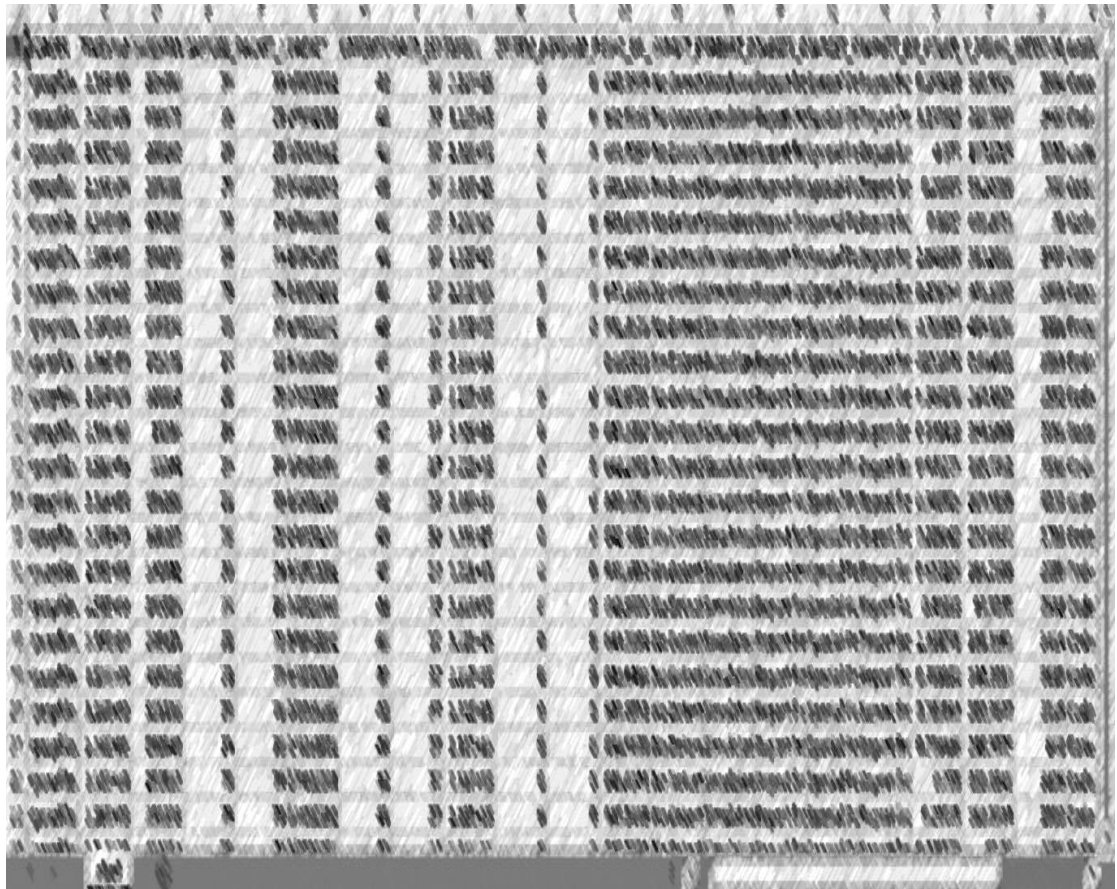
CHAPTER 4

RESULT & DISCUSSION

The contents of ASTERIX Category-48 packets were successfully read and decoded from .pcap file and written to .csv file and polar coordinates of the aircrafts were predicted successfully using linear regression.

```
C:\Users\SuperUser\Desktop>python project.py
Enter pcap file name: cat048
Opening cat048.pcap...
[+] attempting to load cat048.pcap
[+] found valid header
[+] loaded 75433 packets
[+] finished loading savefile.
Enter File name for saving data & metadata: abcd
cat048.pcap contains 32467 CAT48 packets
Data and MetaData of packets saved in abcd.csv
```

4.1 Running the project Python script



4.2 Contents written into csv file

CHAPTER 5

FUTURE SCOPE & CONCLUSION

This scope of this project is currently limited to analyzing packets received beforehand and stored in .pcap files. In future, it may be extended to analyzing real-time incoming packets and predicting the trajectories of aircrafts.

Secondly, this project only focuses on Category-48 ASTERIX packets and support for other ASTERIX categories like Category-34 may be added in future.

Also, the user interface is command line-based which may be upgraded to a graphical user interface which would allow the user to browse the input file (if any) and choose the location for storing output .csv file and plots of predictions.

Linear regression is used for predicting the values of polar coordinates which does not give quite accurate predictions. A more advanced algorithm like XGBoost (boosted random forest), Ridge Regressor, Recurrent Neural Networks or any other good algorithm might be used for getting better results.

REFERENCES

- [1] Python Software Foundation, <https://www.python.org>
- [2] Eurocontrol, <https://www.eurocontrol.int/>
- [3] Wikipedia, <https://www.wikipedia.org/>
- [4] Analytics Vidhya, <https://www.analyticsvidhya.com>
- [5] Thispointer, <https://thispointer.com/>
- [6] The Python Package Index, <https://pypi.org>