

DOCTOR OF PHILOSOPHY

An investigation into inductive parameter learning in complex hierarchical knowledge structures representing clinical expertise

Sherif Hegazy

2012

Aston University

If you have discovered material in AURA which is unlawful e.g. breaches copyright, (either yours or that of a third party) or any other law, including but not limited to those relating to patent, trademark, confidentiality, data protection, obscenity, defamation, libel, then please read our takedown policy at <http://www1.aston.ac.uk/research/aura/aura-take-down-policy/> and contact the service immediately eprints@aston.ac.uk.

An investigation into

**Inductive Parameter Learning in Complex
Hierarchical Knowledge Structures Representing
Clinical Expertise**



Sherif E. Hegazy

Doctor of Philosophy

ASTON University

July 2012

@Sherif E. Hegazy, 2012

Sherif E. Hegazy asserts his moral right to be identified as the author of this thesis

This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with its author and that no quotation from the thesis and no information derived from it may be published without proper acknowledgement.

**An investigation into
Inductive Parameter Learning in Complex Hierarchical Knowledge
Structures Representing
Clinical Expertise**

Sherif E. Hegazy

*PhD Thesis
2012*

Summary

This dissertation investigates the very important and current problem of modelling human expertise. This is an apparent issue in any computer system emulating human decision making. It is prominent in Clinical Decision Support Systems (CDSS) due to the complexity of the induction process and the vast number of parameters in most cases. Other issues such as human error and missing or incomplete data present further challenges.

In this thesis, the Galatean Risk Screening Tool (GRiST) is used as an example of modelling clinical expertise and parameter elicitation. The tool is a mental health clinical record management system with a top layer of decision support capabilities. It is currently being deployed by several NHS mental health trusts across the UK. The aim of the research is to investigate the problem of parameter elicitation by inducing them from real clinical data rather than from the human experts who provided the decision model. The induced parameters provide an insight into both the data relationships and how experts make decisions themselves. The outcomes help further understand human decision making and, in particular, help GRiST provide more accurate emulations of risk judgements. Although the algorithms and methods presented in this dissertation are applied to GRiST, they can be adopted for other human knowledge engineering domains.

Keywords:

Clinical Decision Support, Parameter Elicitation, Missing Data, Knowledge Engineering

Dedication

To my family, especially my father who is the inspiration and the driving force behind my achievements...

Thank you.



Acknowledgement

I would like to express my sincere gratitude to my family, without whose support I would have never achieved so much in my life. I would like to thank my father, who was the inspiration for pursuing my studies to this stage.

A special thanks to my supervisor, Dr. C. D. Buckingham, who was extremely understanding and supportive in many ways, not just academically but also in my life over the past four years.

I would also like to thank Dr. Ann Adams, and her team at Warwick University, who supplied and processed some of the test data used in this work.

Sherif E. Hegazy



Contents

1. Introduction	12
1.1 Knowledge Representation	13
1.2 Parameter Learning	14
1.3 Present Investigation	
1.3.1 Motivation	15
1.3.2 Objectives	16
1.4 Thesis Organization	18
2. Background	19
2.1 Introduction	20
2.2 Clinical Decision Support Systems	20
2.2.1 Definition	
2.2.2 Overview	21
2.2.2.1 History	21
2.2.2.2 Classification	24
2.2.3 Structure of CDSS	25
2.3 Design Methodologies	26
2.3.1 Data Collection	26
2.3.1.1 User Initiated	26
2.3.1.2 Automated	26
2.3.2 Logical Expression	27
2.3.2.1 Decision Tables	27
2.3.2.2 Venn Diagrams	28
2.3.2.3 Logical Conditions	28
2.3.3 Artificial Intelligence	29
2.3.3.1 Artificial Neural Networks	29
2.3.3.2 Decision Trees	31
2.3.3.3 Bayesian Belief Networks	32
2.3.3.4 Data Mining	33
2.3.3.5 Statistical Pattern Matching	34
2.3.3.6 Genetic Algorithms	35

2.3.4 Heuristic Modelling	36
2.3.4.1 Rule-Based Systems	36
2.3.5 Case Studies	38
2.4 GRiST: Galatean Risk Screening Tool	43
2.4.1 Overview	43
2.4.2 Structure	45
2.4.3 Interface	46
2.4.4 Features	47
2.4.5 Parameter Learning in GRiST	47
2.5 Summary	52
3. Methodology	53
3.1 Introduction	54
3.2 Tree Analysis	54
3.2.1 GRiST Tree	54
3.2.2 Mathematical Model	58
3.3 Parameter Learning	67
3.3.1 Introduction	67
3.3.2 ARRIVE Algorithm	67
3.3.2.1 Induction	67
3.3.2.2 Algorithm in a Nutshell	74
3.3.2.3 Case Study	78
3.3.3 iARRIVE Algorithm	80
3.3.4 Error in Solution	83
3.3.5 Case Study	84
3.3.6 Proof of Concept	87
3.4 Summary	92
4. Data Conditioning	93
4.1 Introduction	94
4.2 Membership Grade Functions (MG)	94
4.3 MGM: Membership Grade Modulation	96
Algorithm	
4.3.1 Background	96

4.3.2 The Algorithm	97
4.3.3 Error Estimation	100
4.3.4 Case Studies	103
4.3.5 Analysis of MGM	113
4.4 Summary	115
5. Data Pre-processing	116
5.1 Introduction	117
5.2 Handling Missing Data	118
5.3 Predicting Output	120
5.3.1 Break Points (BP)	120
5.4 Predicting Inputs	123
5.4.1 Using the Mean Method	123
5.4.2 Using Correlation	126
5.4.2.1 The Coefficient of Determination	127
5.4.2.2 The Correlation Matrix	128
5.4.2.3 The Correlation Matrix for GRiST	130
5.4.2.3 Example	131
5.4.3 An Algorithm for Predicting Input by Correlation (APIC)	134
5.4.3.1 Induction	134
5.4.3.3 Error Estimation	139
5.4.3.2 Case Study	140
5.4.3.4 APIC and the Analysis of the Tree	145
5.4.3.5 APIC and iARRIVE	146
5.4.3.6 Case Study	149
5.4.3.7 Analysis	165
5.4.3.8 Error Analysis in Results	166
5.5 Summary	169
6. Results	170
6.1 Introduction	171
6.2 Using Real Data	171

6.3 Data Description	171
6.3.1 Data Cleaning	176
6.4 Results Analysis	177
6.5 Error Distribution	181
6.6 Summary	185
7. Conclusions and Future Work	186
7.1 Conclusions	187
7.2 Future Work	188
References	190
Appendix A: GRiST Tree Structure	199
Appendix B: Results	203
Appendix C: Software	237
Appendix D: Code and Data Structures	245



List of Figures

Figure	Caption
2.1	A schematic drawing of the four-phase model for the development of clinical decision support systems
2.2	A simple Venn Diagram.
2.3	A basic Artificial Neural Network.
2.4	A decision Tree for basic Diabetes screening.
2.5	A simple Bayesian Belief Network for diagnosis of Bronchitis and its relation to smoking and lung cancer.
2.6	The CBR Cycle [adapted from Aamodt & Plaza, 1994].
2.7	A GRiST system layout.
2.8	The General GRiST DSS Tree.
3.1	The General GRiST DSS Tree.
3.2	A simple two level GRiST tree.
3.3	A three level GRiST tree.
3.4	The relationship between the number of RIs, nodes (n), internal nodes (I) and leaf nodes (L) in a tree.
3.5	A leaf node with seven children.
3.6	ARRIVE flow chart.
3.7	A sample decision sub-tree
3.8	A sample decision tree.
3.9	A sample GRiST tree.
3.10	The complement of the tree in Figure 9.
4.1	Membership functions as inputs to a GRiST tree.
4.2	A Membership Grade (MG) function.
4.3	MGM tries to find V, a concatenation of V1 and V2.
4.5	A two sets of MGs results in MGM.
4.6	Area between two curves.
4.7	MG functions provided by six different experts for the same input to the tree.
4.8	The six MGs and Combined MG using the Mean Method.
4.9	The Combined MG using Regression,
4.10	The original MGs as provided by the nine experts for MG1.
4.11	The combined MGM for MG2in regression order 1.
4.12	The combined MGM for MG2in order 2.
4.13	The combined MGM for MG2 in order 3.
4.14	An MGM from the actual trials with 43 experts.
4.15	Union and Intersection in Fuzzy Sets.
5.1	The threshold points, BP.
5.2	Errors associated with missing inputs
5.3	Different Correlation Coefficient values and their meaning.
5.4	A sample GRiST tree with MG inputs A1 to A5.
5.5	A sample tree with three inputs (m1, m2, m3) and five test cases.
5.6	m1 versus m2

5.7	m1 versus m3
5.8	m2 versus m3
5.9	APIC Algorithm flowchart.
5.10	m1 versus m2
5.11	m1 versus m3
5.12	m1 versus m2
5.13	Relationship between m1 with m2.
5.14	Analysis of the relationship between A and B.
5.15	Analysis of the relationship between A and C.
5.16	Analysis of the relationship between A and D.
5.17	Analysis of the relationship between A, B and C using APIC.
5.18	Analysis of the relationship between A, C and D using APIC.
5.19	Analysis of the relationship between A, B and D using APIC.
5.20	Analysis of the relationship between A, B, C and D using APIC.
5.21	Complete APIC flowchart.
6.1	Actual Risk Judgements vs calculated/ predicted judgements using iARRIVE
6.2	Absolute Error distribution across the training set of 771 records..
6.3	Error distribution across the training set of 771 records.
6.4	Absolute Error distribution across the test sample.
6.5	Distribution of absolute error in the training set, given in the % of the predicted values.
6.6	Normal Distribution properties based on its parameters.
6.7	The Error represented in a Normal Distribution form.
6.8	The probabilistic distribution of Data in a Normal Distribution
6.9	Probable Percentage of errors in Confidence Intervals of 1*SD (right) and 2*SD (left).
6.10	The GRiST Risk classification scale.
6.11	The system performance in real life situations, showing impact on clinicians' decisions.



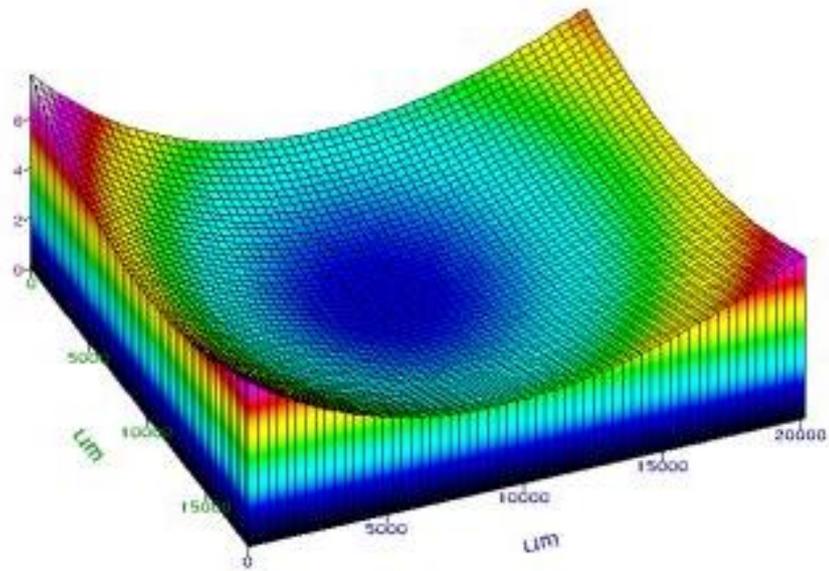
Publications

1. Hegazy, S. E., Buckingham C. D., “**An Algorithm for Robust Relative Influence Values Elicitation (ARRIVE)**,” *Proceedings of the 3rd International IEEE Conference on Computing in the Global Information Technology (ICCGI 2008)*, 2008, Athens, Greece, pp. 91- 96.
2. Hegazy, S. E., Buckingham C. D. (2009), “**An Incremental Algorithm for Robust Relative Influence Values Elicitation (iARRIVE)**,” *Proceedings of eTelemed, IARIA/IEEE, Cancun, Mexico, 2009*, pp. 239-244.
3. Hegazy, S. E., Buckingham C. D. , “**A Method for Automatically Eliciting node Weights in a Hierarchical Knowledge Based Structure for Reasoning with Uncertainty**,” *International Journal on Advances in Software*, issn 1942-2628, 2009 vol 2. nr. 1, page 76 - 83 , <http://www.iariajournals.org/software/>
4. Hegazy, S. E., Buckingham C. D. , (2010), “**Modulating Membership Grades to Gain Consensus for Fuzzy Set Uncertainty Values in a Clinical Decision Support System**,” *Proceedings of the 2010 Third International IEEE Conference on Advances in Human-Oriented and Personalized Mechanisms, Technologies and Services, (CENTRIC 2010)*, Nice, France, August 2010.



Chapter 1

Introduction



This Chapter covers the following:

- Introduction
- Knowledge Representation
- Parameter Learning
- Present Investigation
- Thesis Organization

This work investigates the elicitation of parameter values in a hierarchical structure representing clinical expertise. The work is based on the Galatean Risk Screening Tool (GRiST) [GRiST, 11]. GRiST is a clinical decision support system that provides clinicians with risk assessments based on a psychological model developed through the project. It is currently being deployed in several NHS Trusts across the UK. We have used real data from GRiST to test the algorithms presented in this work. Although GRiST is used as a platform, the algorithms presented here can be easily applied to any knowledge engineering domain.

1.1 Knowledge Representation

In any real life system, knowledge representation is a complicated issue. Computers are at their very definition merely computational machines, but mapping real life data and simulating human thinking is very often a challenge.

Knowledge Engineering is a field of computer science that deals with representing and conditioning knowledge to be suitable for computer processing. This is one of the cornerstones of Decision Support Systems (DSS). One very complicated aspect of knowledge representation is simulating human decision making. This is due to the huge number of parameters involved many of which are unknown. It is also fair to say that it has been a challenge to mirror the human decision making process. Once the knowledge elicitation process is complete, this can then be used for decision support and analysis of new data.

This dissertation focuses on the challenges associated with representing expert knowledge in the clinical field. This is one of the most complicated decision support areas due to the lack of a formal approach by experts. The decision making process in the clinical field is very subjective and lacks formal procedures and steps to describe or back decisions. The Galatean Risk Screening Tool (GRiST) system tries to emulate the decision making process followed by clinicians in their day to day diagnosis. Although the algorithms and methods developed in this dissertation focus on the GRiST system, which represents psychological risk analysis, it can be easily adopted

for other fields of knowledge engineering. This is due to the fact that throughout our work, as explained in detail in this thesis, we have assumed a generic model for data representation that does not rely on specific clinical models. In fact, our induction of the algorithms presented in this dissertation does not make use of any clinical information, but deduces the solution based on the mathematical model of the tree. Thus the tree could represent any knowledge domain, not necessarily clinical, as long as the structure of the tree is provided by the experts.

1.2 Parameter Learning

In order to be able to represent the knowledge elicitation process accurately in any decision support system, a set of parameters such as relative influences or weights of input data; needs to be identified and instantiated. These can be tuned from the data to achieve better results and more accurate decision process.

One of the problems associated with parameter learning in decision support systems in general, is identifying the parameters in the first place. This is due to the fact that in many cases the actual parameters are either unknown themselves, or their behaviour and distributions are not clear. Hence the training phase of a Decision Support System (DSS) is very important when the parameters are learned. The wrong parameters could, and probably would, invalidate the entire system operation.

In Clinical Decision Support Systems (CDDS), this is especially important, as the types and distributions of the parameters play a major part in the credibility of the system among clinicians and patients alike. In other words, we need to be able to justify the use of these parameters and their respective distributions and link them to the actual problem in a semantic way.

There is also the issue of human factors and human errors, which could negatively affect any elicitation process. Obtaining consensus over the elicitation process is a major issue with CDSS. We need to have methodologies in place to generate consensus from very often inconsistent data.

The final obstacle with DSS design in general and CDDSs in particular, is the problem of missing or corrupt data. Whether at the training stage or the deployment

stage, missing data is inevitable. This is due to human factors, the nature of the problem or both. In the medical field, working with hundreds of parameters is not unusual. Hence, the missing data problem becomes more apparent. The more parameters the system has; the bigger the chance of missing some of them due to human error or simply lack of information.

We need to have mechanisms to deal with missing data, otherwise the training would be inaccurate and subsequently, the decision making process would become inaccurate too.

1.3 Present Investigation

1.3.1 Motivation

This work addresses the challenges of missing parameter values learning and elicitation in CDSSs. The algorithms developed in this work have been applied on the GRiST CDSS as a platform to test the methods we developed. Real clinical patient data has been used and the inputs of 43 experts were deployed in the testing process.

Large hierarchical knowledge structures such as GRiST with relative influences of each node (i.e. the contribution of each node in the tree to the top or decision, see Chapter Two) require a large number of parameter values that experts cannot and probably would not provide. The focus of this work is to automate the parameter value elicitation process using available data. Since the GRiST tree structure is predefined through years of elicitation by the experts, the main contribution of this research would be calculating the values of the parameters based on previous known cases in a large hierarchical structure representing clinical expertise. However, the algorithms presented in this thesis can be adapted to other knowledge engineering domains, which is part of the contribution of our work.

The often large number of parameters -and the complicated relationships between them- involved in a CDSS makes it very desirable to have automated methods for the elicitation of its parameters. In some cases, as in GRiST, this may be the only way to obtain objective information with viable proof due to the large number of parameters involved.

Another aspect of a CDSS that is challenging and we aim to address is the missing and corrupt data issue. As with any DSS, and real life system, it is inevitable that some data may be missing and wrong data may be entered by users. The system needs to be able to identify irregularities and warn the user, as well as possibly having methodologies to replace the missing data with valid assumptions. Although this is not the main contribution of this work, we touch on it out of necessity to be able to test our algorithms on real data.

The final challenge that we aim to address is the issue of expert consensus at the training stage of a CDSS. In our case, over 40 experts were prompted to devise some of the parameter distributions. This meant that we could potentially have over 40 different expert alternatives for each of the parameters.

We need a way to be able to concatenate the expert opinions into one global function. The astronomical number of different possible combinations would make it impossible for experts to agree and would make it impossible for us to analyze experts' suggestions. It would be extremely difficult (if not impossible) to collate the data provided by different experts into one coherent model, with meaningful values.

1.3.2 Objectives

The project aims at devising a methodology to automatically elicit parameter values in a decision tree, based on previous or expert knowledge. Though in our case it is applied to a clinical decision support system, the approach could be adopted to other applications that involve similar decision tree structures.

The **objectives** are summarized as follows:

1. Developing a parameter value elicitation methodology for the GRiST decision tree. This includes devising the necessary algorithms to learn the values automatically and using the available data to learn the values. This is the main contribution of this work.

2. Testing the methods on real data. We have a large number of patient data available through the GRiST online system. Several NHS trusts in the UK are currently using GRiST.

3. Exploring the challenges associated with eliciting real life data. This includes – but not limited to- missing values in the test set, incomplete or inaccurate patient records and errors in the original data.

4. Consensus analysis and representation. As the system uses experts knowledge to test the algorithms, it is difficult to make sure that one expert's decision will be similar to another expert even on the same case. This creates the problem of generating consensus and integrity of the test data set.



1.4 Thesis Organization

The rest of the thesis is organized as follows: Chapter Two presents background information and literature survey on the current state of knowledge engineering and clinical decision support. Chapter Three presents our methodology in tackling our problem in representing mental health clinical expertise.

Chapter Four presents our work in tackling the issue of obtaining expert consensus, since the system was developed using the feedback of over 40 experts, their opinions had to be concatenated. We present our approach to this problem.

Chapter Five covers the problem of missing and inaccurate data. We present methods of tackling this issue in GRiST patient records.

Chapter Six presents analysis of the real life data, obtained from our software that uses the algorithms presented in this thesis on real patient data.

Chapter Seven concludes the work and presents future work.

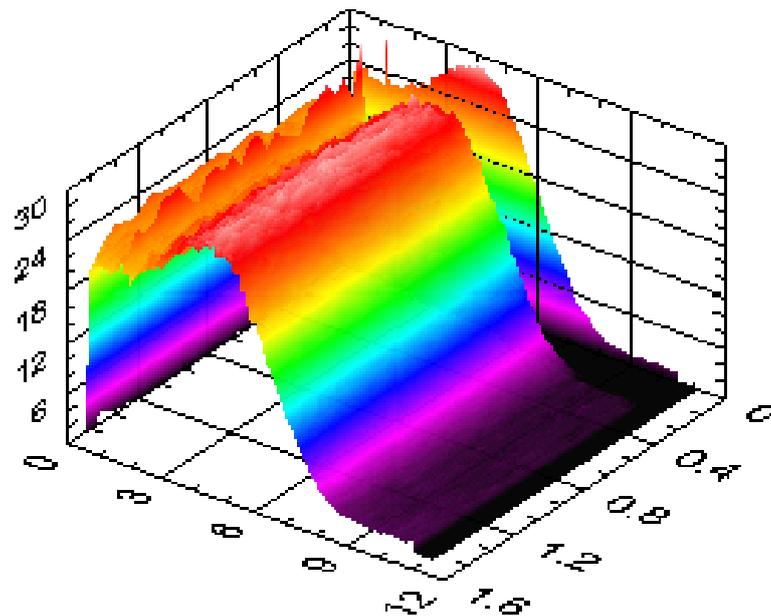
Appendix A through D present some material from the GRiST system (our clinical decision support system), and overview of the software we developed and statistical results from our algorithm.

References are provided in [ABCD, YY] format, where “ABCD” represents the first four letters of the main author’s surname, “YY” is the year of publication. For example, [BUCK, 08], represents a reference by “C.D.Buckingham” as the main author, published in 2008.



Chapter 2

Background



This Chapter covers the following:

- Introduction
- Clinical Decision Support Systems
- Design Methodologies
- GRiST
- Summary

2.1 Introduction

In this chapter, we introduce a background and literature survey of Clinical Decision Support systems, their history, classification, techniques and examples. As the work will be focused on GRiST, this survey will in turn have emphasis on clinical systems. The final part of this chapter focuses on GRiST as our platform and discusses where it is situated among clinical decision support systems and techniques. We will also show how GRiST differs from conventional systems. The survey covers a variety of aspects due to the complex nature of clinical decision support systems. GRiST itself deploys various techniques.

2.2 Clinical Decision Support Systems

2.2.1 Definition

Clinical Decision Support Systems (CDSSs) have been defined broadly as any computer programme which helps health professionals make clinical decisions [SHOR, 90]. They are "active knowledge systems which use two or more items of patient data to generate case-specific advice" [WYAT, 91] and are typically designed to integrate a medical knowledge base, patient data and an inference engine to generate case specific advice. This is a basic definition, but in most clinical systems the data used to generate advice consists of complex and complicated relationships between many pieces of information, as in the case of GRiST [GRIST, 11] [BUCK, 08].

The main purpose of a modern CDSS is to assist clinicians at the point of care [BERN, 07]. Their four key functions are [PERR, 99]:

1. Administrative: Supporting clinical coding and documentation, authorization of procedures, and referrals.
2. Managing clinical complexity and details: Keeping patients on research and chemotherapy protocols; tracking orders, referrals follow-up, and preventive care.
3. Cost control: Monitoring medication orders; avoiding duplicate or unnecessary tests.

4. Clinical Advice: Supporting clinical diagnosis and treatment plan processes; and promoting use of best practices, condition-specific guidelines, and population-based management.

Clinical decision support systems are often complicated and required to work in sensitive knowledge domains [HEEK, 06], [FOX, 02]. Risk screening in the mental health field is a particularly complex procedure but lacks much assistance beyond paper-based approaches [HAWL, 06]. Unfortunately, computerised decision support systems to assist mental-health clinicians are hindered by limited knowledge about the cognitive processes involved in risk assessment, which compounds the inherent difficulty of knowledge elicitation [BUCK, 07].

2.2.2 Overview

2.2.2.1 History

The concept of decision support has evolved from two main areas of research: The theoretical studies of organizational decision making done at the Carnegie Institute of Technology during the late 1950s and early 1960s, and the technical work on interactive computer systems, mainly carried out at the Massachusetts Institute of Technology in the 1960s [KEEN, 78].

It is considered that the concept of DSS became an area of research of its own in the middle of the 1970s, before gaining in intensity during the 1980s [SPRAG, 93]. In the middle and late 1980s, executive information systems (EIS), group decision support systems (GDSS), and organizational decision support systems (ODSS) evolved from the single user and model-oriented DSS [EFRA, 05].

The definition and scope of DSSs has been migrating over the years. In the 1970s a DSS was described as "a computer based system to aid decision making" [SOL, 87]. In the late 1970s the DSS movement started focusing on "interactive computer-based systems which help decision-makers utilize data bases and models to solve ill-structured problems" [SOL, 87]. In the 1980s DSS should provide systems "using suitable and available technology to improve effectiveness of managerial and professional activities", and the end of 1980s DSS faced a new challenge towards the design of intelligent workstations [BERN, 99]. Beginning in about 1990, data

warehousing and on-line analytical processing (OLAP) began broadening the realm of DSS. As the turn of the millennium approached, new web-based analytical applications were introduced [ANAH, 97].

Since the early days of computers, as early as the 1950's, the prospect of using computers to assist in clinical decision making has been recognized by physicians and computer scientists [GREE, 07]. Several groups began to analyze the process of medical diagnoses in order to automate it [LEDL, 59]. This in turn led to another domain of research, more concerned with the representation and use of knowledge, beyond data and numbers which was later to become known as knowledge engineering. This led to the exploration of new symbolic reasoning approaches [GREE, 07].

Clinical decision support systems followed the developments in decision support systems, and they started to emerge as early as the 1960's CDSS [SHOR, 90]. Their major obstacle was negative human perception and the reluctance of clinicians to give up their knowledge and use the systems afterwards. The lack of structure and rules in the clinical process did not help either, as many experts rely on experience rather than set-in-stone rules [GRAY, 01]. Experience changes with time and is amended or modified, whereas rules are fixed. This is apparent in the medical field as new advances in medicine can change previous rules.

This problem is still around today, although progress has been made in many areas.

A brief history of early clinical decision support systems is listed in Figure 2.1 below. More details on these systems are given in section 2.3.5: Case studies. Some of these systems (or their derivatives) are still in use today, such as INTERNIST, others form the basis of many of the current systems on the market, like MYCIN.

Wright and Sitting [WRIG, 08] have formulated a model charting the developments in the clinical domain with four distinct architectural phases for decision support:

1. Standalone decision support systems, beginning in 1959:
 - a. These were the early attempts to develop tools to help clinicians and gather medical data. These were disjoint efforts with no standardisation or integration ability.
2. Integrated systems, beginning in 1967:

- a. These were systems that could be integrated into existing clinical systems, although rigid formats had to be followed and often were not portable among different platforms.
3. Standards-based systems, beginning in 1989:
 - a. Attempts to create a standard syntax for medical decision support systems and specifically rules were introduced. Some of these, like ARDEN are still in commercial use today.
4. Service models, beginning in 2005:
 - a. These used Application Programming Interface (API) framework to create portable systems that can be integrated more easily into existing clinical records. They eased the restrictions on standards on the data side, but imposed new restrictions on the application side. They act as gateways between the data and the Decision Support System.



Figure 2.1: A schematic drawing of the four-phase model for the development of clinical decision support systems [WRIG, 08].

2.2.2.2 Classification

Most CDSSs consist of three parts, the knowledge base, inference engine, and mechanism to communicate. There are two main types of CDSS, as classified by [BERN, 99]:

- Knowledge-Based
- Non Knowledge-Based

i. Knowledge-Based CDSS

In this model, the knowledge base contains the rules and associations of compiled data which most often take the form of IF-THEN rules. If this was a system for determining drug interactions, then a rule might be that IF drug X is taken AND drug Y is taken THEN alert user.

Using another interface, an advanced user could edit the knowledge base to keep it up to date with new drugs [BERN,99]. The inference engine combines the rules from the knowledge base with the patient's data. The communication mechanism will allow the system to show the results to the user as well as have input into the system. [BERN, 07], [OPEN, 09].

This is covered in more detail in the heuristic modelling section.

ii. Non-Knowledge-Based CDSS

These are CDSS that do not use a knowledge base but use a form of artificial intelligence or machine learning, which allows computers to learn from past experiences and/or find patterns in clinical data. An example is using neural networks. These systems generally do not require a rule base, as they generate their rules from the data and previous cases. There are several methodologies used for these types of systems, mainly statistical and mathematical. These are listed in this chapter in section 2.3.3. Non-knowledge-based systems often focus on a narrow list of symptoms such as ones for a single disease as opposed to the knowledge based approach which cover many different diseases to diagnosis [BERN, 07].

The advantage of knowledge based systems is that the decisions can be easily justified and are supported by the experts who supplied the rules. They are also easy

to implement. They have a limitation on the number of rules (as they need to be supplied by experts) and they are difficult to update.

Non-knowledge based systems on the other hand, learn directly from the data, which makes them up to date and reflect the previous cases. It is difficult to justify the decisions and outcomes and the design and testing process of these systems is quite complicated. They require fairly large amounts of data for accurate learning.

2.2.3 Structure of CDSS

Shortliffe [CLAN, 84] classifies the structure of CDSSs into five areas:

i. Functionality:

Broadly speaking they are intended to assist with forming diagnoses or selecting treatment.

ii. Output:

Passive systems require the physician to access the advice, whereas active systems give unsolicited advice and have a higher level of information processing [ELSO, 95], e.g., PRODIGY [PURV, 88].

iii. Consultation Type:

This may be by a response to entered data, which is the commonest type, or by a critiquing of the doctor's treatment suggestions, e.g., CAPSULE <http://www.infermed.co.uk>.

iv. Design Methodology:

The mathematical model or method used in creating the engine behind the DSS could be used to classify such systems. For example, rule based experts systems, belief networks and decision trees have all been used to model clinical expertise. [BERN, 99]. These are covered in more detail later.

v. Interface:

The Human Computer Interface (HCI) is an important aspect of any compute system. If the system is not user friendly and intuitive, it is likely to be blocked by the target users and eventually discarded. This is more apparent in the clinical field, as

clinicians are used to certain paper interfaces and documents, which they would be very reluctant to replace. The interface design will depend on the application and inputs of the system [BERN, 07].

2.3 Design Methodologies

Any decision support system relies on knowledge that originates from a variety of sources [GREE, 07]. In clinical DSSs, however, due to the large number of the sources, selecting and integrating this knowledge is a complicated task [COIE, 94].

There are many different methodologies that can be used by a CDSS In order to process the data and prepare it in order to provide support to the health care professional, The basic components of a CDSS include a *dynamic* (medical) knowledge base and an *inference mechanism*. In the following sections, we will present an overview of both [COIE, 94]. The dynamic knowledge base is mainly concerned with the data, how it is retrieved, modeled and pre-processed. The inference mechanism is mainly responsible for the decision support process, and intelligence.

2.3.1 Data Collection

Finding information relevant to the problem or the case is a basic form of CDSS. A simple check to determine whether a laboratory result is normal or not requires searching a table or literature to find the ranges of a 'normal' result.

There are two main types of Information retrieval, user initiated and automated.

2.3.1.1 User Initiated

This could be simply described as a search tool for clinicians to search literature. An early example is MEDLINE (1964), which started as a bibliographic retrieval system for biomedical literature for use by librarians in the US National Library of Medicine (NLM). The system still exists in other forms with over 21 million citations as cited by the National Institute of Health [Pubmed.gov, accessed on 1 July 2012].

2.3.1.2 Automated

In this case, the system can automatically come up with suggestions that are relevant to the current context, and perform automated searches using automatically generated parameters. An example is the Infobutton [THOM, 10] where a visual icon is

placed on all screens indicating that information relevant to the display is present in other areas. The automated system offers more flexibility and advice but require far more development and testing. It also relies on experts input in the design cycle.

2.3.2 Logical Expressions

Logical expressions are indeed very common in clinical decision support systems. They represent a true or false decision node, and can be represented using many common techniques, such as decision tables. Complex expressions can be constructed from a set of simple true or false statements.

The deterministic nature of the Logical Expression makes it easy to account for the decision and track the decision process. For example, if patient BMI is larger than 26, then obesity alert is raised. Alerts and reminders have been shown to help increase physician compliance with many different guidelines; however, the risk exists that creating too many alerts and reminders could overwhelm doctors, nurses, and other staff and cause them to ignore the alerts altogether. There are many different ways to analyze the data and represent them using logical expressions and these are described next.

2.3.2.1 Decision Tables

These are among the earliest techniques used to represent logical conditions. The advantages of a decision table include the ease of sorting the data based on a certain column (criteria), and the ease of mapping between paper based human approaches and the computerised representation. Decision tables use Boolean expressions to represent the rules.

Example: A simple decision table for the GP or nurse at a clinic is shown in Table 2.1.

Condition	Rules			
Fever?	Y	Y	N	N
Headache?	Y	N	Y	N
Actions				
Refer to consultant	Y	Y		
Paracetamol	Y		Y	

Table 2.1: A simple Decision Table for a nurse in GP surgery.

2.3.2.2 Venn Diagrams

Another classic approach to representing logic in general and clinical logic specifically, is Venn diagrams. Venn diagrams were conceived around 1880 by John Venn. They are used to teach elementary set theory, as well as illustrate simple set relationships. Shapes or circles are used to enclose groups of entities representing certain characteristics. The overlap of these circles indicates subgroups that have a combination of characteristics from several groups.

The main limitation of the Venn Diagram is its visual aspects, as it becomes significantly difficult to represent more than three or four classes visually. An Example is shown in Figure 2.2. Classes A, B and C could represent different symptoms of three diseases. The intersection area would represent the shared symptoms of these three diseases.

Venn diagrams are very useful in mindmaps and the design stage, but later in the design process they become more difficult and complicated to use.

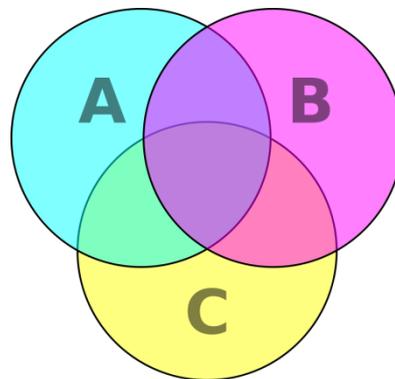


Figure 2.2: A simple Venn diagram.

2.3.2.3 Logical Conditions

A common way of representing logical conditions is by the use of logical expression, of If ... then statements. The evaluation of n expressions yields a true or false result. Boolean operators can also be used in these expressions. Logical expressions are widely used in knowledge based systems, such as rule based reasoning, where decisions are made using If-Then statements.

2.3.3 Artificial Intelligence

As in most human decision and judgment, most clinical judgement is not deterministic. Many factors contribute to various errors, imprecision of tests and limitations of data. Many decisions are based on expert opinions rather than actual facts or figures. There are several techniques and methods used to incorporate probabilities into the process of clinical decision making [MICH, 93].

2.3.3.1 Artificial Neural Networks

Artificial Neural Networks (ANN) use a form of artificial intelligence, that allows the systems to learn from past experiences / examples and recognizes patterns in clinical information. They consist of nodes called neurons and weighted connections that transmit signals between the neurons in a unidirectional fashion. A simple ANN consists of three main layers (Figure 2.3): Input (data receiver or findings), Output (communicates results or possible diseases) and Hidden (processes data). The system becomes more efficient with known results for large amounts of data [TANG, 07]. Artificial neural networks use nodes and weighted connections between them to analyze the patterns found in the patient data to derive the associations between the symptoms and a diagnosis [BURK, 97].

The advantages of ANN include the elimination of needing to program the systems and providing input from experts. ANN systems do not require large databases to store outcome data with its associated probabilities.

Some of the disadvantages are that the training process may be time consuming leading users to not make use of the systems effectively. Parameter learning could be sensitive and requires rigorous testing to obtain the best learning parameters. They also require large amounts of testing data to gain acceptable accuracy. The ANN systems derive their own formulas for weighting and combining data based on the statistical recognition patterns over time which may be difficult to interpret and doubt the system's reliability [GALU, 07]. Another disadvantage is that since the system cannot explain the reason why it uses the data the way it does, most clinicians don't use them for reliability and accountability reasons [BERN, 07].

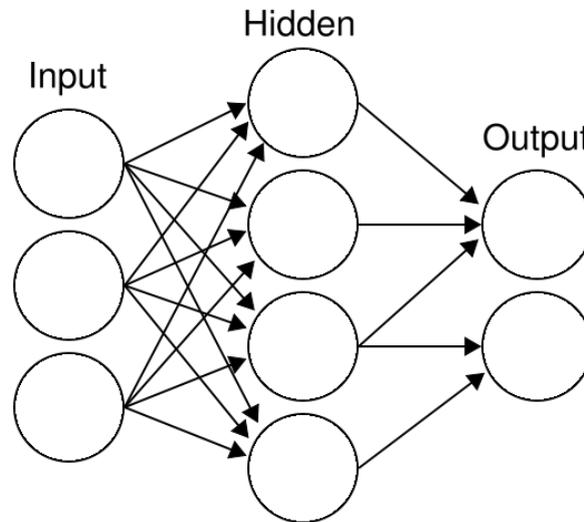


Figure 2.3: A basic Artificial Neural Network.

Examples include the diagnosis of appendicitis, back pain, myocardial infarction, psychiatric emergencies and skin disorders. The ANN's diagnostic predictions of pulmonary embolisms were in some cases even better than physician's predictions. Additionally, ANN based applications have been useful in the analysis of Electrocardiogram (ECG) (A.K.A. EKG) waveforms [DYBO, 01].

There are many variations and derivations from the basic neural network, all of them however share the same limits. Many of them only work on thresholds, i.e. not exact values, and most need large amounts of test data. This means that the results of the neural net are approximate or in a range form, which makes it difficult when exact values are needed. The error margins depend highly on the data and the number of test cases, as well as learning parameters. This makes it quite sensitive.

2.3.3.2 Decision Trees

Decision trees are a very useful and intuitive way of representing decisions, as they emulate human thinking [AHO, 74]. The model in which every decision is based on the comparison of two numbers within constant time is called simply a decision tree model. It was introduced to establish computational complexity of sorting and searching [QUIN, 89]. Figure 2.4 shows a basic decision tree for early diabetes screening of a patient.

Simple decision trees can be generated manually, when all the parameters are known. In more complex cases, when the structure of the tree is not known, techniques have been developed to generate the structure based on known data. Most algorithms that have been developed for learning decision trees are variations on a core algorithm that employs a top-down, greedy search through the space of possible decision trees. These algorithms generally construct a decision tree T from a set of training cases [QUIN,92].

J. Ross Quinlan developed the first algorithm, ID3 [QUIN, 75]], and based it on the Concept Learning System (CLS) algorithm [QUIN, 75]. Other methods like Classification and Regression Trees (CART) were introduced for the induction of a tree [BREI, 84]. Variations on the above methods usually deal with the type of the input variables, the data pool or set properties, or the output type (i.e. continuous or discrete data) [DRUC, 96], [JANI, 96]. Most of these methods attempt to construct the tree without prior knowledge of the desired tree structure. This means, they attempt to predict the layout of the tree and number of nodes based on the training cases. The trees are then pruned and optimized to the minimum structure that satisfies the classes in the training instances. A simple decision tree is shown in Figure 2.4.

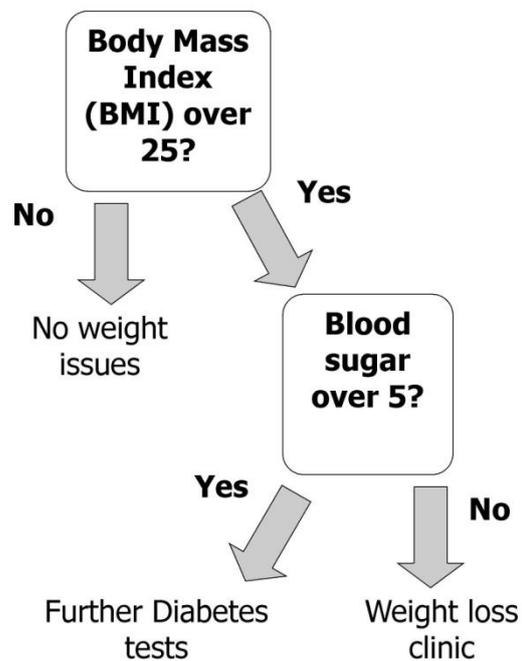


Figure 2.4: A decision tree for basic diabetes screening.

2.3.3.3 Bayesian Belief Networks

Bayesian Belief Networks (BBN) model causality and the probabilistic dependencies of events on one another [NEAP, 04]. They are based on conditional probabilities, the probability of an event given the occurrence of another event [HEIS, 75]. In the context of CDSS, the Bayesian network can be used to compute the probabilities of the presence of the possible diseases given their symptoms [JENS, 96].

They were developed by Pearl in the 1970's at Stanford [GREE, 07]. The first application in medical problems was by Cooper in the 1980's. This model permits known probabilities to be entered into the system, and estimate or infer unknown ones based on the known probabilities [JENS, 07].

An example is shown in Figure 2.5. The BBN is used to determine whether a smoker will develop Bronchitis or cancer due to smoking. The X-Ray will be able to correctly determine cancer with a certain probability. Correct diagnosis is also measured in probability.



Figure 2.5: A simple Bayesian Belief Network for diagnosis of Bronchitis and its relation to smoking and lung cancer [GADEW, 10].

In a clinical BBN, nodes are used to represent items such as symptoms, patient states or disease categories. Connections between nodes indicate a cause and effect relationship. A system based on this logic will attempt to trace a path from symptom nodes all the way to disease classification nodes, using probability to determine which path is the best fit.

Some of the advantages of this approach are the fact that it helps to model the progression of a disease over time, based on linking time and probability; and the interaction between diseases; however, it is not always the case that medical experts know exactly what causes certain symptoms, and it can be difficult to choose what level of detail to build the model to. BBN's also have the advantage of representing knowledge and conclusions of experts in the form of probabilities and are based on unbiased probabilities that are applicable to many models.

Some of the disadvantages of Bayesian network include the difficulty to get the probability knowledge for possible diagnosis and not being practical for large complex systems given multiple symptoms. The Bayesian calculations on multiple simultaneous symptoms could be overwhelming for users.

The first clinical decision support system to use a causal probabilistic network was CASNET, developed by Kulikowski (1982), used to assist in the diagnosis of glaucoma. CASNET featured a hierarchical representation of knowledge, splitting all of its nodes into one of three separate tiers: symptoms, states and diseases. Another example of a Bayesian network in the CDSS context is the Iliad system [WARN, 94] which makes use of Bayesian reasoning to calculate posterior probabilities of possible diagnoses depending on the symptoms provided. The system now covers about 1500 diagnoses based on thousands of findings. A further example is the DXplain system (<http://www.lcs.mgh.harvard.edu/dxplain.htm>) that uses a modified form of the Bayesian logic. This CDSS produces a list of ranked diagnoses associated with the symptoms. It is covered later in the Case Studies section.

2.3.3.4 Data Mining

As databases became larger and computer hardware became faster and more capable of faster processing, the number of patient records and data stored digitally

soared in the 1980's [TAN, 05]. With such massive data volumes, it became difficult to extract meaningful information, or relationships that can be used in a CDSS.

Data mining uses statistical and mathematical techniques to find relationships between field and substructures in a database based on occurrences in data [BOCA, 03]. Newer techniques such as fuzzy logic have also been deployed in recent years [LARO, 06].

Data mining and machine learning, also known as Knowledge Discovery from Databases (KDD), have also benefited from other artificial intelligence techniques, such as artificial neural networks [LARO, 06]. A combination of AI techniques and data mining algorithms could produce more in depth knowledge and pattern analysis of the system. Data mining could use a variety of methods to deduce relationships and information from the data.

2.3.3.5 Statistical Pattern Matching

Pattern Recognition techniques define the mathematical relationship between measurable features and classification of objects. [KANA, 74]. In clinical practice, the presence or absence of each of several signs and symptoms in a patient may be definitive for the classification of a patient [SHOR, 79].

In order to find the diagnostic pattern, or discriminant function, the method requires a training set of objects for which the correct classification is already known. If the form and parameters are not known for the statistical distributions underlying the features, then they must be estimated [CLAN, 84]. After training, the pattern can be compared to new, unclassified objects to aid in deciding the category to which the new object belongs.

Three of the best known training criteria for the discriminant function are [SHOR, 79]:

1. **Least squared error criterion:** Choose the function that minimizes the squared differences between predicted and observed measurement values. This includes a variety of statistical regression methods, and Regression Analysis of the results to decide on the quality of fit and when to stop the regression process [CHAT, 77].

2. **Clustering Criterion:** choose the function that produces the tightest clusters;
3. **Bayes' Criterion:** Choose the function that has the minimum cost associated with incorrect diagnoses. The method also involves probability, which helps to model more complex problems.

There are numerous papers on the use of pattern recognition methods in medicine, as early as (Armitage and Gehan, 1974), who discuss three examples of prognostic studies, with an emphasis on regression methods [CLAN, 84]. Medical imaging pattern recognition is widely used as a decision support tool to highlight possible abnormalities in medical MRI scans and X-Rays [BAES, 03].

2.3.3.6 Genetic Algorithms

A Genetic Algorithm (GA) is a non-knowledge-based method developed in the 1940s at the Massachusetts Institute of Technology based on Darwin's evolutionary theories that dealt with the survival of the fittest [GOLD, 89]. These algorithms rearrange to form different re-combinations that are better than the previous solutions. Similar to neural networks, the genetic algorithms derive their information from patient data.

In a genetic algorithm, a population of strings (called chromosomes or the genotype of the genome), which encode candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem, evolves toward better solutions [FALK, 97]. These systems go through an iterative process to produce an optimal solution. The so-called fitness function measures the quality of potential solutions.

A disadvantage is the lack of transparency in the reasoning involved for the decision support systems making it undesirable for physicians.

The main challenge in using genetic algorithms is in defining the fitness criteria [FRAS, 70].

2.3.4 Heuristic Modelling

In the previous section, we introduced some of the methods used to modelling the CDSS based on data. An alternative approach is to try to emulate human reasoning processes and model human expertise. These models are often well understood by clinicians, and the output can be justified. This is sometimes called “naturalistic” decision making [KLEI, 93]. It differs from that found in the artificial intelligence / expert system literature in that these models are typically focused on emulating the outcomes of expert decision making by emulating the process a human decision maker might use in reaching the outcome [MORR,98]. The human expert will typically look at a situation, and use some general heuristic derived from his/her previous experience to choose an action, rather than computationally exhausting all possible outcomes.

2.3.4.1 Rule-Based Systems

A rule-based expert system attempts to capture knowledge of domain experts into expressions that can be evaluated known as rules. These are generally simple linguistic rules or logical expressions that represent the knowledge of the expert. An example rule might read, "If the patient has high blood pressure, he or she is at risk for a stroke." Once enough of these rules have been compiled into a rule base, the system can then use them to give diagnosis on new cases, by looking for the closest match.

The main difficulty in such systems is the formulation of the rules themselves. Questionnaires or interviews are often used to extract the rules from the experts. The rules may be in several levels and the output of one level would be the input to another. An extension of the Rule based-system is a Case-Based system, which tries to overcome the problem of rule formulation or at least updating. A Case-Based system is more advanced, in the sense that it uses rules, but learns from new cases, and new rules could be inferred based on the new cases entering the system.

Figure 2.6 shows the Case Based Reasoning (CBR) system’s life cycle.

The current working knowledge will be evaluated against the rule base by chaining rules together until a conclusion is reached [KOLO, 93]. Some of the advantages of a rule-based expert system are the fact that it makes it easy to store a large amount of information, and coming up with the rules will help to clarify the logic used in the decision-making process. However, it can be difficult for an expert to transfer their

knowledge into distinct rules, and many rules can be required for a system to be effective.



Figure 2.6: The CBR Cycle [adapted from Aamodt & Plaza, 1994]

Rule-based systems can aid physicians in many different areas, including diagnosis and treatment. An example of a rule-based expert system in the clinical setting is MYCIN [SHOR, 76]. The Stanford AI group subsequently developed ONCOCIN, another rules-based expert system coded in Lisp in the early 1980's [SHOR, 81].

The system was intended to reduce the number of clinical trial protocol violations, and reduce the time required to make decisions about the timing and dosing of chemotherapy in late phase clinical trials. As with MYCIN, the domain of medical knowledge addressed by ONCOCIN was limited in scope and consisted of a series of eligibility criteria, laboratory values, and diagnostic testing and chemotherapy treatment protocols that could be translated into unambiguous rules. Oncocin was put into production in the Stanford Oncology Clinic.

2.3.5 Case studies

CDSS software can be commercially available or home grown to meet specific needs. Either way, applications of information technology must be motivated by clinical problems in order to be useful in the clinical setting [COIE, 97].

A brief overview of some systems is given below:

CASNET/Glaucoma

CASNET (Causal ASSociational NETworks), developed in the 1960s, was a general tool for building expert system for the diagnosis and treatment of diseases. The most significant expert system application based on CASNET was CASNET/Glaucoma for the diagnosis and treatment of glaucoma.

Expert clinical knowledge was represented in a causal-associational network (CASNET) model for describing disease processes. CASNET/Glaucoma was developed at Rutgers University and implemented in FORTRAN.

AAPHelp: de Dombal's system for Acute Abdominal Pain (1972):

This was an early attempt to implement automated reasoning under uncertainty [HORR, 72] [DOMB, 93]. De Dombal's system, developed at Leeds University, was designed to support the diagnosis of acute abdominal pain and, based on analysis, the need for surgery. The system's decision making was based on the naive Bayesian approach.

INTERNIST I (1974)

Pople and Myers began work on INTERNIST, one of the first clinical decision support systems, designed to support diagnosis, in 1970.

INTERNIST-I was a rule-based expert system designed at the University of Pittsburgh in 1974 for the diagnosis of complex diagnosis of complex problems in general internal medicine. It uses patient observations to deduce a list of compatible disease states (based on a tree-structured database that links diseases with symptoms). By the early 1980s, it was recognized that the most valuable product of the system was its medical knowledge base. This was used as a basis for successor systems including CADUCEUS and Quick Medical Reference (QMR), a commercialised diagnostic DSS for internists [MILL, 89].

HELP

“HELP was the first hospital information system to collect patient data needed for clinical decision-making and at the same time incorporate a medical knowledge base and inference engine to assist the clinician in making decisions” [GARD, 98]. It was developed by Department of Medical Informatics, University of Utah, Salt Lake City in 1975. It supports not only the routine applications of an Hospital Information System (HIS) including order entry/charge capture, pharmacy, radiology, nursing documentation, ICU monitoring, but also supports a robust decision support function. The decision support system has been actively incorporated into the functions of the routine HIS applications. Decision support has been used to provide alerts/reminders, data interpretation, patient diagnosis, patient management suggestions and clinical protocols. Activation of the decision support is provided interactively within the applications and asynchronously through data and time drive mechanisms.

MYCIN (1976)

MYCIN was a rule-based expert system designed to diagnose and recommend treatment for certain blood infections (antimicrobial selection for patients with bacteremia or meningitis). It was later extended to handle other infectious diseases.

Clinical knowledge in MYCIN is represented as a set of IF-THEN rules with certainty factors attached to diagnoses. It was a goal-directed system, using a basic backward chaining reasoning strategy (resulting in exhaustive depth-first search of the rules base for relevant rules though with additional heuristic support to control the search for a proposed solution). MYCIN was based on around 600 rules and was used to help identify the type of bacteria causing an infection. While useful, MYCIN can help to demonstrate the magnitude of these types of systems by comparing the size of the rule base (600) to the narrow scope of the problem space.

MYCIN was developed in the mid-1970s by Edward Shortliffe and colleagues at Stanford University [SHOR, 76]. It is probably the most famous early expert system, described by Mark Musen as being "the first convincing demonstration of the power of the rule-based approach in the development of robust clinical decision-support systems" [MUSE, 99].

The EMYCIN (Essential MYCIN) expert system shell, employing MYCIN's control structures was developed at Stanford in 1980. This domain-independent framework

was used to build diagnostic rule-based expert systems such as PUFF, a system designed to interpret pulmonary function tests for patients with lung disease.

ONCOCIN

A rule-based medical expert system for oncology protocol management developed at Stanford University in 1976. Oncocin was designed to assist physicians with the treatment of cancer patients receiving chemotherapy. ONCOCIN was one of the first DSS which attempted to model decisions and sequencing actions over time, using a customised flowchart language. It extended the skeletal-planning technique to an application area where the history of past events and the duration of actions are important.

DXplain

DXplain is a decision support system which uses a set of clinical findings (signs, symptoms, laboratory data) to produce a ranked list of diagnoses which might explain (or be associated with) the clinical manifestations. DXplain provides justification for why each of these diseases might be considered, suggests what further clinical information would be useful to collect for each disease, and lists what clinical manifestations, if any, would be unusual or atypical for each of the specific diseases.

DXplain includes 2,200 diseases and 5,000 symptoms in its knowledge base. It was developed in 1984 by Laboratory of Computer Science, Massachusetts General Hospital, and Harvard Medical School. (<http://www.lcs.mgh.harvard.edu/dxplain.htm>). It is still available today.

QMR : Quick Medical Reference

QMR is a diagnostic decision-support system with a knowledge base of diseases, diagnoses, findings, disease associations and lab information, with information from the primary medical literature on almost 700 diseases and more than 5,000 symptoms, signs, and labs." QMR was designed for 3 types of use: "as an electronic textbook; as an intermediate level spreadsheet for the combination and exploration of simple diagnostic concepts; as an expert consultant program" [MILL, 89].

It was developed (1980) by the University of Pittsburgh and First Databank, California (<http://www.firstdatabank.com/>).

PRODIGY:

Prescribing RatiOnally with Decision support In General practice studY

PRODIGY is a computerised prescribing decision support system for general practitioners. It provides information for patients, supports the regular review of clinical management, and presents prescribing recommendations as well as advice on non-drug treatments for a range of conditions.

This system is integrated with computerised patient records and offers evidence-based treatment advice, prescribing recommendations and patient information leaflets. Support is offered for a wide variety of clinical conditions, covering 70% of GP consultations [PURV, 98].

APACHE

APACHE was one of the first medical decision support systems to be commercialised - in 1988 by Apache Medical Systems Inc, a company founded specifically to carry this out. APACHE III is today marketed by Cerner Inc.

ARDEN:

Started in 1989 [ARBO,07], it is one of the most successful attempts of creating a standard syntax for rule representation in CDSS. It used Medical Logic Modules (MLM) as standard units to represent the rules. Arden is still in use commercially today, and major clinical systems vendors such as McKesson and Siemens offer support for the Arden syntax. Its latest version 2.6 has been accepted as a standard by the American National Standards Institute (ANSI).

GLIF: Guideline Interchange Format

GLIF is often seen as an extension to ARDEN, with more extended records and flexibility. It is yet to be seen in commercial systems. From 2000, GLIF was being tested in Stanford, Columbia and Harvard [OHNO, 98].

ISABEL

Isabel Healthcare provides a diagnosis decision support application that assists physicians with getting the diagnosis right the first time. Accessed over the web or fully integrated with an Electronic Medical Record (EMR) system. Started in 2003 as an

independent provider, it is still in use commercially today and has new version that is supported on smart phones and mobile apps (www.isabelhealthcare.com).

Shareable Active Guideline Environment project (SAGE) [RAM, 04]:

The system was started in 2005, and it uses API to connect the decision support side to the clinical data, hence the terminology service phase. It uses a standard API interface to connect to the clinical data, which is called the Virtual Medical Record (VMR). It requires the data to be stored in a standard strict format to be able to access it through the systemd

SEBASTIAN:

First attempts took place in 2005, and although it shares SAGE's API approach, it has a more generic API interface, that acts as a translator between the Decision Support system and the Clinical data, posing less restrictions on the clinical side. This means, the SEBASTIAN API interface can be integrated more easily into existing clinical systems.

SANDS (Service-oriented Architecture for NHIN Decision Support):

Another system that is based on a service-oriented architecture [NADK, 07], that seamlessly integrates into current clinical systems. It has been applied in several case studies in the US [WRIG, 08b].

The above brief history shows the evolution of clinical decision support systems over the decades. Many practical considerations govern the design of these systems.. In most systems, simpler rule based or logical expressions are used. This allows for practical implementation and testing. More complex methods require far more time and data, and are very complex to verify. This leads to our system, GRiST, which is introduced in the next section. GRiST uses more complex and higher level hierarchical structures that mimic clinical decision process which makes it different from all above systems.



2.4 GRiST: Galatean Risk Screening Tool

2.4.1 Overview

The Galatean Risk Screening Tool, GRiST is a web-based Clinical Decision Support system for risk assessments in Mental Health [BUCK, 07B]. The project aims to improve mental-health risk assessments by developing a much needed, universally accessible, and innovative computerised decision support system (DSS). The DSS will contain a risk-screening tool that records client data (cues) and provides risk estimates for suicide, self-harm, self-neglect, and harm to others. The tool is intended for use without specialist training and by any relevant professionals, not just those within health and social care. It will provide a new educational and clinical resource linking validated human expertise from mental-health professionals with statistical information extracted from a dedicated client database.

Current research has resulted in a pilot screening tool that collects relevant cues relating to risk but does not itself quantify risk. The project seeks to build on the work by creating a DSS with the following facilities: a database of client risk information; a psychological model of how expert clinicians process cues to predict risk; graphical tools for displaying the flow of uncertainty through the psychological model from cues to risk estimates; data-analysis tools to determine the empirical relationships between cues and risks in the database; and a web-based DSS providing a supportive environment for using the tool, including quantification of risk, explanations for how the quantifications were obtained, and supporting material that may help inform the users' actions.

The linkage of human expertise with empirical data analysis will show how clinical decisions can be improved and enable explanations of risk to be presented in a form comprehensible to all people working with mental-health clients. Health and social benefits include multidisciplinary cooperation between health-care providers, fewer inappropriate referrals, and the reduction of unidentified risk. In addition, the methods and technology developed can be generalised to many areas of expert decision-making.

Risk screening in the mental health field is a particularly complex procedure but lacks much assistance beyond paper-based approaches [BUCK, 07] , [BUCK, 07b]. Unfortunately, computerised decision support systems to assist mental-health clinicians

are hindered by limited knowledge about the cognitive processes involved in risk assessment, which compounds the inherent difficulty of knowledge elicitation [BUCK, 02], [BUCK, 08]. The Galatean Risk Screening Tool, GRiST [GRIST, 10], addresses the problem by directly investigating how mental-health clinicians carry out assessments and incorporating their expertise within a decision support system, by using a psychological model of classification [BUCK, 02].

However, when applied to a large, complex, and hierarchical knowledge structure, the number of parameters that need setting for accurate simulation of expert judgements is extremely large; several thousand for GRiST. The main components of the GRiST system are shown in Figure 2.7.

The GRiST server holds the learned parameters and the software for learning and evaluations. It uses an image of the patient data and records on the client side. Clinicians can tune the learning parameters and check the quality of the learning. End users use the system to enter new patient records or run online assessments using the learned parameters on the GRiST server. All the components are accessible via the Internet through various security and authentication layers.

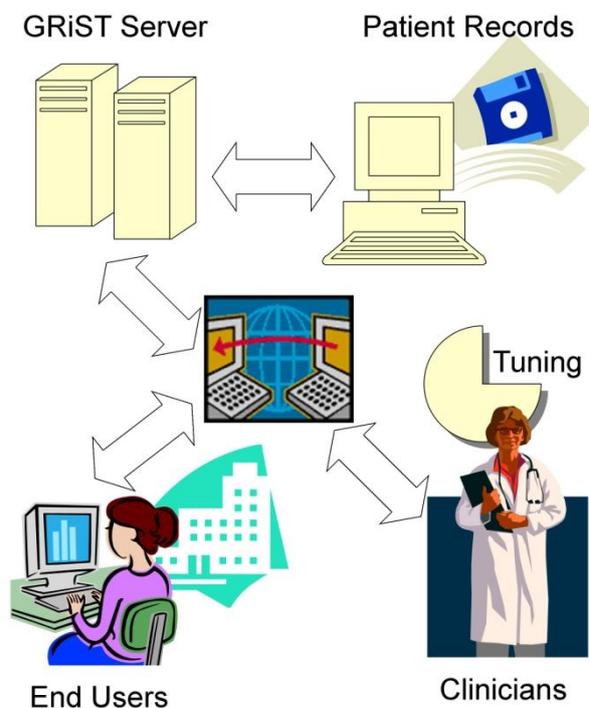


Figure 2.7: A GRiST system layout

2.4.2 Structure

The basic structure of the decision support system expertise in GRiST is a weighted decision tree [BUCK, 08]. Input is represented by questions that are associated with each leaf node of the tree as shown by Figure 2.8. See Appendix A for more details on the GRiST Tree structure. The questions form part of an electronic data-gathering tool and the value given in response to a question is passed into a function associated with the leaf node for that question to produce Membership Grades (MG), which is then propagated up the tree. Importance of items and concepts in the tree are represented by weightings or Relative Influences (RIs) that moderate the contribution of MGs to risk.

The GRiST server contains the elicitation engine, web interface, security handlers as well as report generators. Patient records are kept separate on the organizations' servers and are only linked through coded keys to GRiST. Clinicians can tune GRiST, and customize the interface. The end users can use the system on site or through a web interface.

The structure of the tree has been developed from the psychological model that has been induced from over 100 experts in the UK. Input is represented by questions that are associated with each leaf node of the tree. The questions form part of an electronic data-gathering tool and the value given in response to a question is passed into a function associated with the leaf node for that question to produce an MG, which is then propagated up the tree. Importance of items and concepts in the tree is represented by weightings (RIs) that moderate the contribution of MGs to risk.

The RIs also need to be set so that they reflect the expertise of mental-health practitioners. Getting them to do it themselves as part of the knowledge elicitation process is an arduous task when the tree may have several thousand nodes.

This makes it unlikely that a large enough set of participants can be obtained to ensure the consensus for each RI is reliable, as opposed to eliciting the leaf node parameters, which are far fewer. This could be seen as a generic problem, and the algorithms developed in this work can be extended to other domains of knowledge with similar structures.

2.4.3 Interface

GRiST incorporates various online web interfaces that can be accessed via a security gateway. It also provides parameter elicitation tools, as well as expertise representation and modification. It has a suite of reporting and graphing tools [BUCK, 07], to assist users and visualize data.

GRiST is currently being used in four NHS Trusts in the UK, and more trusts are adopting it. It was recommended by the Department of Health as one of only three tools in Mental Health DSS best Practice Guide [GRIST, 10]. Appendix C contains some more information about the GRiST interface and snapshots of the various tools we developed.

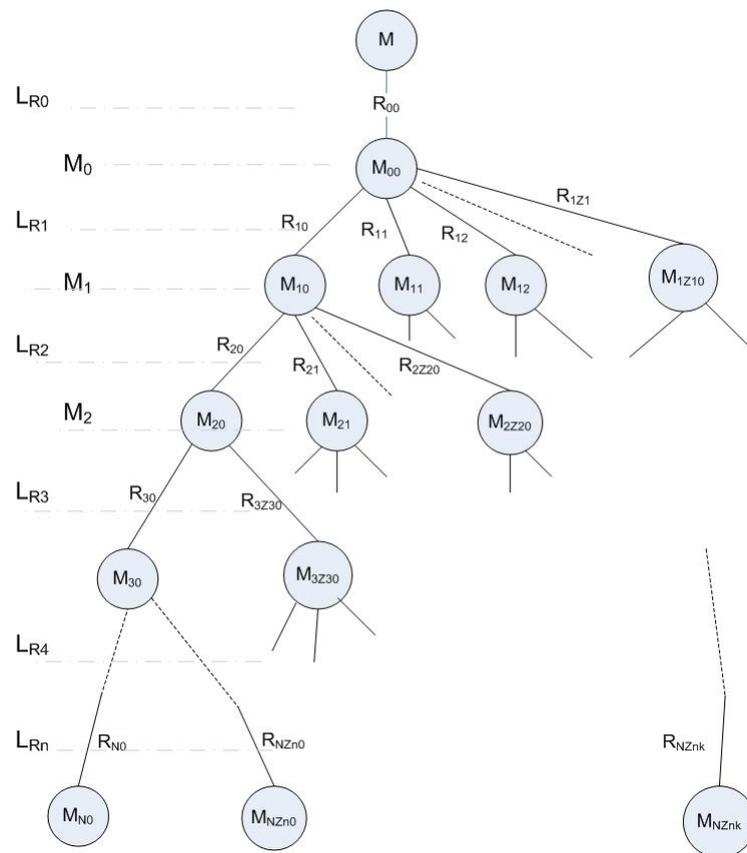


Figure 2.8: The General GRiST DSS Tree.

2.4.4 Features

The GRiST CDSS consists of the following elements:

- A tool for gathering risk assessment data.
- A structured data storing and reporting tool.
- A psychological model of clinical expertise that will generate quantifications of risk associated with the data.
- Statistical and pattern-recognition tools for generating mathematical risk assessments to support the clinical ones.
- A database of client risk information and associated clinical judgements of risk.

Client data will be analysed by the DSS to produce estimates of the degree of risk exhibited in areas such as suicide, self-harm, self-neglect, and various forms of risk to others such as violence and neglect of dependants. The data-gathering tool and overall DSS will be designed for use by any person who may need to make risk assessments of clients from the population of working-age adults. Hence it will not require specialised knowledge or extensive training.

GRiST stores the questionnaires in a structured way that is easy to handle and analyse, it also provides various reporting tools.

2.4.5 Parameter Learning in GRiST

In most CDSSs developed so far, (as in the previous overview), there are two main types of CDSS engines: Automated parameter learning and user specified parameters.

The user specified parameter allocation process is when the experts provide the data and information for the system to use. A straightforward example is rule based systems, where previous cases are explicitly provided by the experts and the system then matches up new cases with known ones. The main problem with such systems is that they rely highly on the accuracy of the rules provided by the experts. If the rules

are inaccurate, then the systems output will be inaccurate as well. There is also no way of scrutinizing these systems' performance.

Another drawback is that the system does not automatically evolve or learn from new cases and as the database grows. So in reality, it is a hard-coded static system. Any new rules have to be added by the experts. One major advantage of such systems, especially in the clinical field, is the ease of justifying the results semantically, as the rules have been entered by the experts originally.

On the other extreme, automated parameter learning does not involve experts or human intervention. A good example for that are data mining techniques, which mainly use statistical methods to induce relationships in the data, regardless of the semantic relationships. This means that some of the results may not be possible to justify clinically. The system does not take any expert opinions into account and is reliant purely on numbers and statistics to draw conclusions.

One major advantage of these systems is their adaptability and ongoing evolution. Particularly in the incremental versions, the system updates its parameters with every new case, and by doing so, it evolves and improves its error margins and eventually renders the error negligible.

The major problem in such automated systems is that it is virtually impossible to justify the results clinically as they do not rely on a semantic foundation. It is also very difficult to sway clinicians to use or trust the system, as they do not understand the basis and have no input into it. This alienates such systems from the clinical domain. Many believe that this human factor was one of the main reasons such systems were never advanced as far as they should have.

In GRiST, we have combined the two approaches for the first time. Clinicians have the tools to input and tweak the initial parameters in the tree, such as the membership functions at the leaf nodes, and the automated process that we introduce in the work, calculates the rest of the required parameters in the tree (Relative Influence RI, values). (See Figure 2.9).

The RI values themselves can then be further manipulated by the clinicians to assess the effect of such changes and to link to the semantic expertise and rules. Having the automated process allows the system to evolve and update its parameters

incrementally. This makes it flexible and up to date. It also allows for error estimation and even checking the consistency of the original data.

This has the advantage of both types of design approaches, as it involves clinicians in the process, allowing for semantic justification of the results, and at the same time allows for evolution and incremental learning. It gives the opportunity to analyse the data and produce error margins and accuracy and consistency measures to evaluate the systems performance without alienating clinicians.

In GRiST, two main types of parameters will need to be elicited: relative influence values and membership grades. RI values, which determine how much each child contributes to the risk in the parent node. Due to the large number of these (3000+), this will be done automatically using the algorithms we developed in this work.

The other parameter is the Membership Grade (MG) function, which determines the relationship between the inputs from the questionnaire with the actual value that will be used for the risk component at the leaf nodes. This is provided by the experts using the online tools we developed as part of this project, which are illustrated in Appendix C.

In our project, parameter value learning is the main challenge, due to the large number of parameters, as explained above.

There are several techniques for learning parameter values in a decision support system. The main ones are data mining, Bayesian Belief Networks, Neural Networks and Decision Trees.

In data mining, previously unknown relationships are induced from the data based on statistical analysis, and the results are represented using statistical parameters such as confidence and sample size [LARO, 06]. It is useful in case the required relationships are not exactly known and the results depend highly on the data. This means the outcomes will dramatically differ depending on the data set, and different parameters will be obtained with different sets of data. This is different from our problem, where the parameters are known and provided by the experts, whereas the values are unknown.

In Bayesian belief networks, the relationships between the parameters and the nodes are represented by probabilities [JENS, 07]. The Bayesian belief networks can be extended to decision graphs, which provide a more general and flexible way of

representing the network [JENS, 07]. This is a different currency from that we use in our system, which are weights that represent contributions to the total risk.

Neural networks are one popular way to learn parameter or weight values in a decision support system or a decision support system in general [SIMO, 07]. The main drawback in that case as far as our system is concerned is the lack of transparency in the layers. The neural network does not provide the internal structure of the network. Instead it calculates the weights with no explanation of the internal structure apart from the number of layers. This means we cannot map our system onto the network, as our internal tree is predefined and supplied by the expert. Neural networks therefore, make it difficult to justify the results and are thus not suitable for our work.

Decision trees are used to represent knowledge, and extract relationships on a path that were not obvious from the data. Most decision tree algorithms however, focus on the construction of the tree itself, as classifiers, to rearrange the tree and generate a structure, such as Quinlan et. Al., whose ID3 is the basis for many Decision tree algorithms [QUIN, 84]. Other recent research suggested using different techniques from the top-down classifiers to induce the tree structure from data, such as [BARR, 12]. This is not necessary in our case, as the tree is already provided by the experts.

The function of the tree after construction can be described as either classification or regression [BREI, 84]. In the case of classification, the tree classifies the input value into one of x predefined classes, thus using a range rather than an exact number. In the case of regression, the output of the tree is a number or a value.

Our system is unique in terms that it uses both expert inputs and automated algorithms to calculate the parameter values. Most of the systems we presented use either one or the other, and in case of parameter learning, they are mostly automated.

As in all real life systems, GRiST faces the problem of dealing with distorted and missing data. This along with the approximation errors, need to be minimized and quantified to be able to have an acceptable representation of the expertise and at the decision support phase.

These will be the focus and the contribution of this work.

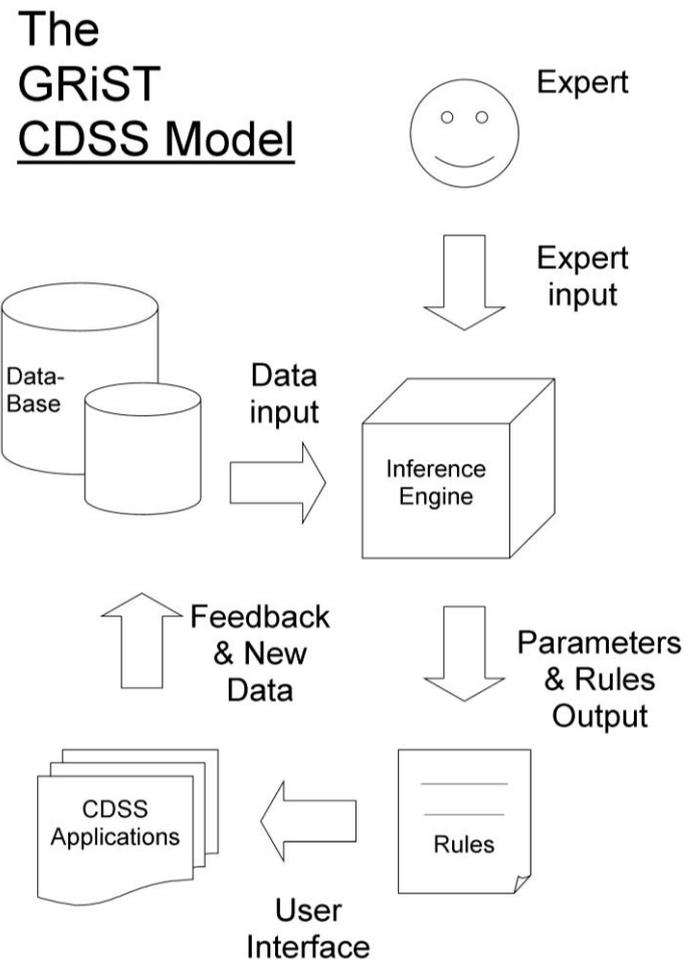


Figure 2.9: The GRiST CDSS Model

2.5 Summary

In this chapter, we presented an overview of Clinical Decision Support Systems, including their history, classification, components, as well as their different design methodologies. Most real life CDSS involve eliciting parameters based on experts knowledge and using these parameters at the inference stage to assess new cases.

We also gave a description of GRiST, the clinical decision support system that we will be using throughout this work as a case study for the algorithms presented in the next two chapters. GRiST is currently deployed by several major NHS trusts in the UK. We use the data obtained from GRiST to test our algorithms.

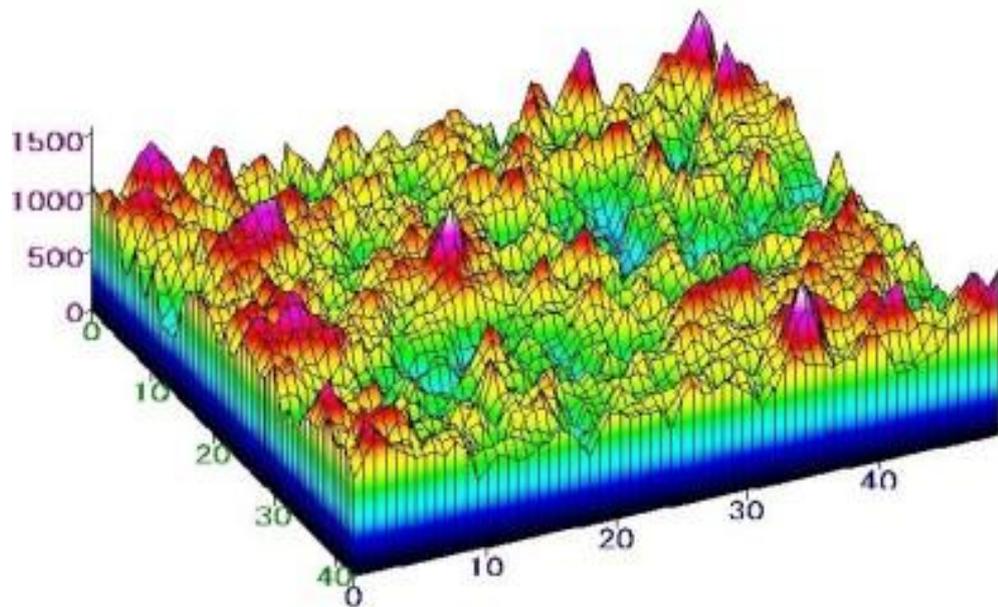
We also illustrated how GRiST is different from other clinical decision support systems and subsequently, why commonly used techniques are not suitable to solve our problem.

The next chapter presents our methodology in developing our algorithm to elicit the parameters in the GRiST tree, namely the Relative Influence values.



Chapter 3

Methodology



This Chapter covers the following:

- Parameter Learning
- ARRIVE Algorithm
- iARRIVE Algorithm

3.1 Introduction

This chapter investigates the elicitation of parameters in a hierarchical structure representing clinical expertise in the form of a decision support tree. The work is based on the GRiST (Galatean Risk Screening Tool). GRiST is a tool that provides the clinician with risk assessments based on a psychological model. Although the methods and algorithms presented in this work are applied to GRiST, many are applicable to other knowledge engineering domains.

We present our ARRIVE (an Algorithm for Robust Relative Influence Values Elicitation). We then present an extension, iARRIVE, an incremental version, which allows updating the values based on new cases. At the end of this Chapter, we present case studies and error analysis of the algorithm.

3.2 Tree Analysis

3.2.1 GRiST Tree

The Galatean Risk Screening Tool, GRiST, addresses the problem of modelling the clinical knowledge by directly investigating how mental health clinicians carry out assessments and incorporating their expertise within a decision support system, by using a psychological model of classification [BUCK, 02]. However, when applied to a large, complex, and hierarchical knowledge structure, the number of parameters that need setting for accurate simulation of expert judgements is extremely large; several thousands for GRiST [BUCK, 08].

The basic structure of the decision support system expertise in GRiST is a weighted decision tree. Risk is represented by fuzzy-set membership grades (MGs) [ZADE, 65] that are associated with each node of the tree. Data associated with a patient assessment (case) generates a MG at the matching leaf node using a function that depends on some parameters given by the experts for each leaf node.

The MGs then propagate up the risk hierarchy and eventually to the top level risks, where the MG associated with a risk represents the simulated clinical risk judgement.

The relative influence of each node in the hierarchy is a parameter that decides how much risk is propagated up the tree by a node compared to its siblings [BUCK, 08].

This parameter also needs to be set so that it reflects the expertise of mental-health practitioners. Asking them to provide it themselves as part of the knowledge elicitation process is an arduous task when the tree may have several thousand nodes. This makes it unlikely that a large enough set of participants can be obtained to ensure the consensus for each RI is reliable, as opposed to eliciting the leaf node parameters, which are far fewer.

In this section, we devise an algorithm that induces the RIs from the clinical judgements given by expert mental-health practitioners for patient cases. If we can do this, it means the RIs are modelled on the clinicians' own risk judgements because the RIs are set to the exact values required for simulating those judgements. This depends on knowing the MGs at the leaf nodes along with the associated clinical risk judgements, which means the elicitation process only requires experts to provide parameters for the leaf nodes.

It is important, if not mandatory, for having an automated system to elicit RIs because their sheer number is likely to mean experts don't do it accurately themselves. Even if they did, there would need to be additional elicitation rounds to obtain consensus on the collated data.

Figure 3.1 shows a generic GRiST tree, where:

L_{Rn} : denotes the RIs in level n.

M_n : is a set of the MGs (Membership Grades) in level n.

M_{xy} : denotes the Membership Grade of node y in level x. M_{xy} is part of M_n . $y=0$ to Z_{jh} , where Z_{jh} is the number of children of node number h-1 at level j.

R_{ti} : denotes the RI of node number i at level t on the total MG at level t-1 which equals $M_{(t-1)y}$. y is an index representing the number of the parent node of R_{ti} .

To find M, the total membership grade of the tree (which represents the overall diagnoses or risk of the patient's mental health), there are several methodologies we could follow.

One way would be to train the model using known cases and, assuming that leaf MG values and M are given, we could use a Neural Networks simulation.

Note that in GRiST: $\sum_{y=0}^{Zx^y} R_{xy} = 1$, is an inherent property in the GRiST tree.

This will mean that $R_{00} = 1$.

This is due to the fact that the GRiST model assumes that each node contributes to the total risk, and to the intermediate risk to the parent node. Thus, the children share the contribution to the risk in the parent by a percentage, and the total maximum contribution is 100% (or 1).

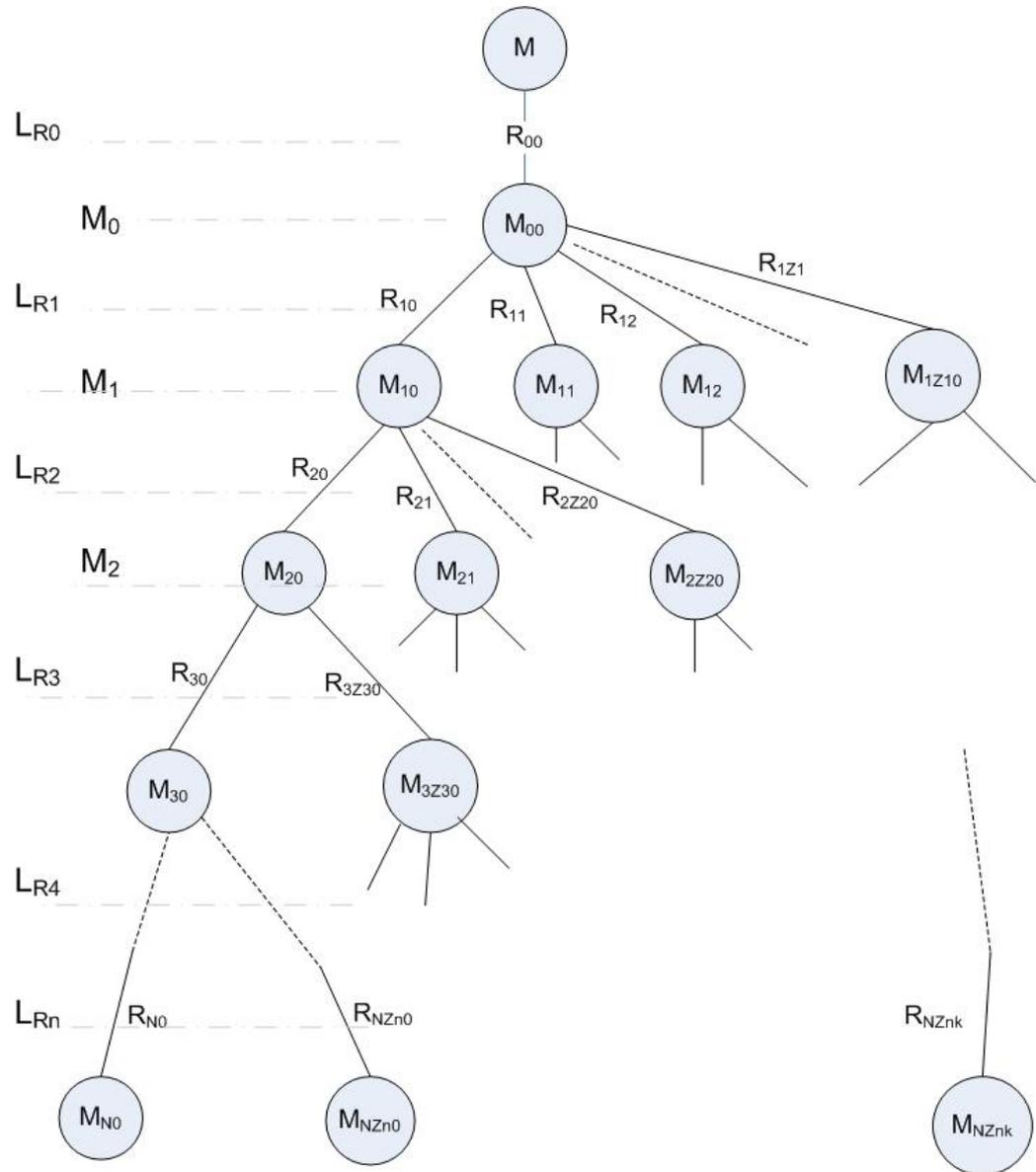


Figure 3.1: The General GRiST DSS Tree.

3.2.2 Mathematical Model

To model the tree mathematically, we follow the model GRiST introduced to calculate the overall result (membership, M).

$$\begin{aligned}
 M &= R_{00} M_{00} \\
 R_{00} &= 1, \text{ thus, } M = M_{00} \\
 M &= R_{00} (R_{10} M_{10} + R_{11} M_{11} + R_{12} M_{12} + \dots + R_{1Z10} M_{1Z10}) \\
 &= R_{00} (R_{10} (R_{20} M_{20} + R_{21} M_{21} + R_{22} M_{22} + \dots + R_{2Z10} M_{2Z20}) + \\
 &\quad R_{11} (R_{2(Z20+1)} M_{2(Z20+1)} + R_{2(Z20+2)} M_{2(Z20+2)} + R_{2(Z20+3)} M_{2(Z20+3)} + \dots + \\
 &\quad R_{2(Z20+Z21)} M_{2(Z20+Z21)}) + \\
 &\quad R_{12} (\dots) + \\
 &\quad \dots + \\
 &\quad \dots + \\
 &\quad R_{1Z10} (\dots))
 \end{aligned}$$

Eqn. (3.1)

By simplifying the above equation, we obtain the following (multiplying the brackets):

$$\begin{aligned}
 M &= R_{00} R_{10} R_{20} M_{20} + R_{00} R_{10} R_{21} M_{21} + \\
 &\quad R_{00} R_{11} \dots + R_{00} R_{10} \dots + \\
 &\quad R_{00} R_{12} \dots + R_{00} R_{12} \dots + \\
 &\quad \dots
 \end{aligned}$$

Eqn. (3.2)

If we continue this process, till we reach the leaves, the resulting expression will look like the Eqn (3.2), with all Rxy multiplied in front of the leaf MGs then added together.

Note that there is a certain pattern for the multiplication expression. We will clarify how we will make use of it in Section 3.3 as it is part of our algorithm induction. To illustrate the above, we use a simpler example of a tree with two levels.

Example 1:

As shown in Figure 3.2, for simplicity, we will rename the leaves as follows:

$$a, b, c, d, e, f, g = (M_{20} \text{ to } M_{26})$$

Eqn.(3.3)

Note that, MG values will be present as we are using pre-assessed cases for training.

$$\begin{aligned} M &= R_{00} (R_{10} M_{10} + R_{11} M_{11} + R_{12} M_{12}) \\ &= R_{00} (R_{10} (R_{20} a + R_{21} b) + R_{11} (R_{22} c + R_{23} d) + R_{12} (R_{24} e + R_{25} f + R_{26} g) \end{aligned}$$

Or:

$$\begin{aligned} M &= R_{00} R_{10} R_{20} a + R_{00} R_{10} R_{21} b + \\ &\quad R_{00} R_{11} R_{22} c + R_{00} R_{11} R_{23} d + \\ &\quad R_{00} R_{12} R_{24} e + R_{00} R_{12} R_{25} f + R_{00} R_{12} R_{26} g \end{aligned}$$

Eqn. (3.4)

Since, a to g are given, the unknowns are R's.

The above M is given by the experts (consultation results carried out by the clinicians on previous patients), and this is available for many patients that were assessed before (training set). Hence, we have several of the above equations (several patients' records).

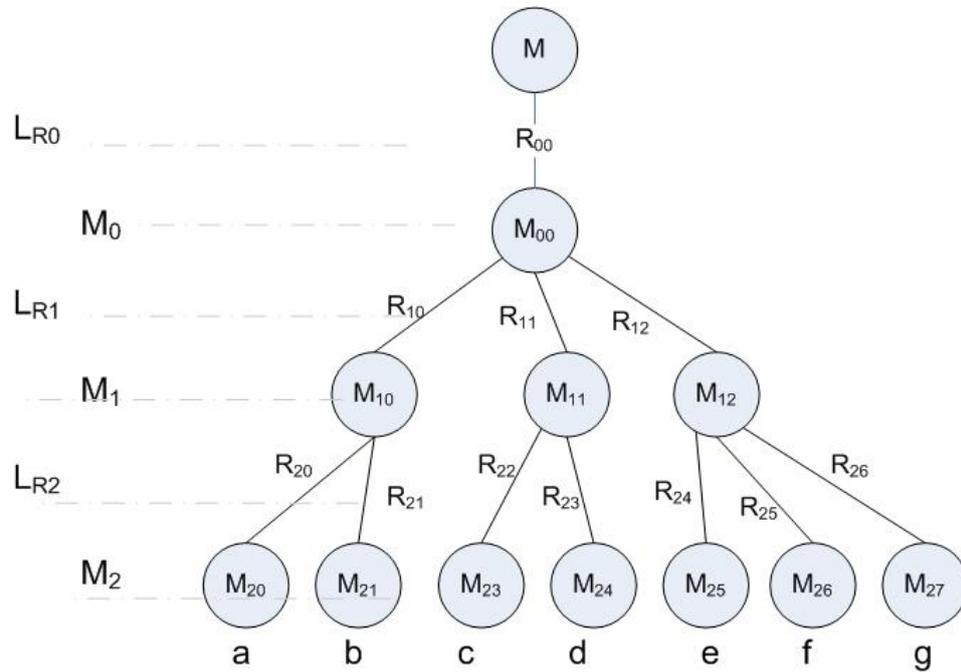


Figure 3.2: A simple two level GRiST tree.

We can look at the above mentioned system of equations as a system of linear simultaneous equations. So what we have done, in effect is transforming the original problem with RIs as the unknown into a new domain, with less variables, and where the RIs are not separate variables, but combinations of RIs.

Even though in the GRiST tree, some of the leaf questions may be repeated, the MGs are only used as coefficients to the new variables, and thus the new variables can still be treated as independent. It is important to keep the repeated MGs to preserve the structure representation on the tree. Because we use a multiplication along the path of the MG as the new variable (see below), this means the new variables will be independent and this is important for the simultaneous equations.

We now define new variables as the new unknowns to be solved. The new unknowns are:

$$A = R_{00} R_{10} R_{20}$$

$$B = R_{00} R_{10} R_{21}$$

$$C = R_{00} R_{11} R_{22}$$

$$D = R_{00} R_{11} R_{23}$$

$$E = R_{00} R_{12} R_{24}$$

$$F = R_{00} R_{12} R_{25}$$

$$G = R_{00} R_{12} R_{26}$$

Eqn. (3.5)

That is seven unknowns. Note that $R_{00} = 1$ (from the RI properties).

But originally, we need to find the values of R's, i.e., eleven unknowns (see Figure 3.2).

Seven linear equations will suffice to find the seven unknowns (A through G) as follows:

$$M1 = a1. A + b1. B + c1. C + d1. D + e1. E + f1. F + g1. G$$

$$M2 = a2. A + b2. B + c2. C + d2. D + e2. E + f2. F + g2. G$$

.....

$$M7 = a7. A + b7. B + c7. C + d7. D + e7. E + f7. F + g7. G$$

In General form:

$$Mi = ai . A + bi . B + ci . C + di . D + ei . E + fi . F + gi . G, \quad i=1, \dots, 7$$

Eqn. (3.6)

Solving the above is straight forward (using matrices).

Now, we have A to G.

But originally, we had eleven unknowns, (eleven RIs), so to determine RIs, we need an extra four equations in addition to the above seven.

For this use an inherent property of RIs (for GRiST) [BUCK, 02]:

$$\sum_{y=0}^{Zxn} R_{xy} = 1$$

Eqn. (3.7)

In our case this gives us:

$$R_{10} + R_{11} + R_{12} = 1$$

$$R_{20} + R_{21} = 1$$

$$R_{22} + R_{23} = 1$$

$$R_{24} + R_{25} + R_{26} = 1$$

Eqn. (3.8)

These are the extra four equations needed to fully determine the eleven RIs. So we have eleven equations in eleven unknowns.

Now, to find the solution, we will use A to G. We will expand this step in the next examples.

Note that the new unknowns have no meaning in the original GRiST tree. They are merely the multiplication of the paths along the leaf nodes to the root. They serve however in reducing the number of overall variables to solve. This means the new problem has fewer unknowns and at the same time can be reformulated in a format for which many established techniques can be used to solve. We have turned a problem of eliciting thousands of individual values into one of learning them from clinical judgements. And we have done this by reformulating the problem as a straightforward weight learning exercise that is then used to learn internal RIs recursively.

The unknowns can now be found using several methods and two of them are described in the following sections: We have chosen the Gaussian method for solving simultaneous equation as a straight forward solution, although it has many limitations, including limiting the number of equations and possibility of not having a solution in case of singular matrices. We then present another solution, using multiple and polynomial regression, which is more practical and adaptable. This is the method we applied to the real data in testing.

Example 2:

We illustrate that the pattern of equations and the number of equations holds for larger trees. In this case using a three level GRiST tree. The aim of this example is to demonstrate that the solution can be found, and there is enough information to fully determine the unknowns. The full algorithm will be presented in the next section.

As shown in Figure 3.3. For simplicity, we rename the leaves as follows:

$$a, b, c, d, e, f, g, h, i = (M_{30} \text{ to } M_{38})$$

Eqn. (3.9)

These values will be provided by the experts (we are using assessed cases with known outcomes. MG on the leaves are given by the experts). Using the same procedure as in example two, and simple arithmetic manipulation, we will obtain the following:

$$\begin{aligned} M &= R_{10} M_{10} + R_{11} M_{11} \\ &= R_{10} (R_{20} (R_{30}a + R_{31} b)) + R_{21} (R_{32} c + R_{33} d + R_{34} e) \\ &\quad + R_{11} (R_{22} (R_{35} f + R_{36} g)) + R_{23} (R_{37} h + R_{38} i) \end{aligned}$$

Or:

$$\begin{aligned} M &= R_{10} R_{20} R_{30} a + R_{10} R_{20} R_{31} b + \\ &\quad + R_{10} R_{21} R_{32} c + R_{10} R_{21} R_{33} d + R_{10} R_{21} R_{34} e \\ &\quad + R_{11} R_{22} R_{35} f + R_{11} R_{22} R_{36} g + \\ &\quad + R_{11} R_{23} R_{37} h + R_{11} R_{23} R_{38} i \end{aligned}$$

Eqn. (3.10)

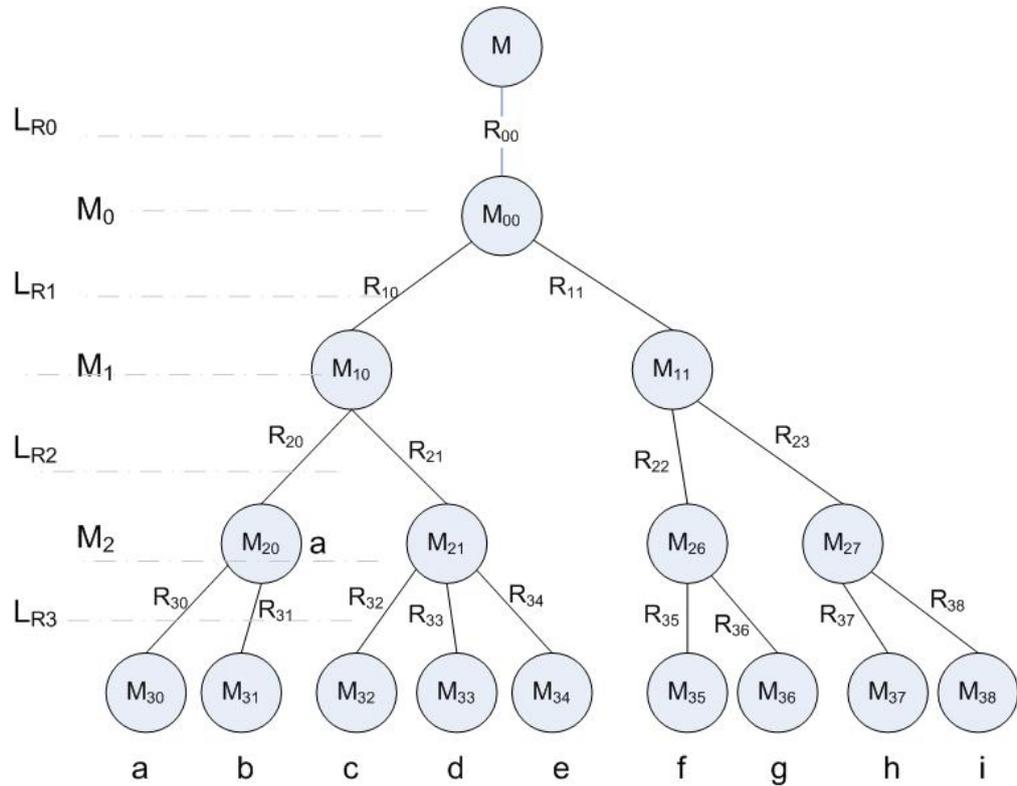


Figure 3.3: A three level GRiST tree.

Similar to the previous case we rewrite:

$$A = R_{10} R_{20} R_{30}$$

$$B = R_{10} R_{20} R_{31}$$

$$C = R_{10} R_{21} R_{32}$$

$$D = R_{10} R_{21} R_{33}$$

$$E = R_{10} R_{21} R_{34}$$

$$F = R_{11} R_{22} R_{35}$$

$$G = R_{11} R_{22} R_{36}$$

$$H = R_{11} R_{23} R_{37}$$

$$I = R_{11} R_{23} R_{38}$$

Eqn. (3.11)

As there are 16 unknown RIs, for a fully determined equation system in addition to these nine equations we will use the seven equations provided as constraints on the RIs. (see Figure 3.3).

$$R_{10} + R_{11} = 1$$

$$R_{20} + R_{21} = 1$$

$$R_{22} + R_{23} = 1$$

$$R_{30} + R_{31} = 1$$

$$R_{32} + R_{33} + R_{34} = 1$$

$$R_{35} + R_{36} = 1$$

$$R_{37} + R_{38} = 1$$

Eqn. (3.12)

Note that R_{00} although considered an unknown in the general form of the model, it has to be equal to 1. From the above two examples, it can be seen how the RIs can be fully determined using the proposed model.

To generalize, we derive a general equation for the number of equations that can be obtained from a tree. Use the example in Figure 3.4. where:

n = total number of nodes in the tree.

RI = total number of RIs. These are the connecting branches in effect.

L = total number of leaf nodes (nodes without children).

I = number of internal nodes (non-leaf). $I=n-L$.

We know that we get L equations for the Gaussian elimination (see previous section). We need to show that we have extra equations for the solution to be possible, the number of those is at least $E_{min}=RI-L$.

From figure 3.4, we notice that each internal node, will add an extra equation. This is because it will have children, and the sum of RIs at those children has to be 1 (Eqn. 3.7). So the number of extra equations we get in a tree is equal to the number of internal nodes (I). But the number of internal nodes = $I = n - L$.

From figure 3.4, we notice that $RI = n - 1$. This is because each node will have a parent (and thus an RI linking it to its parent) apart from the top node.

So the total number of extra equations we get in a tree will equal:

$$E = n - L = (RI + 1) - L = (RI - L) + 1 = E_{min} + 1.$$

This proves that the system will be solvable in the general case.

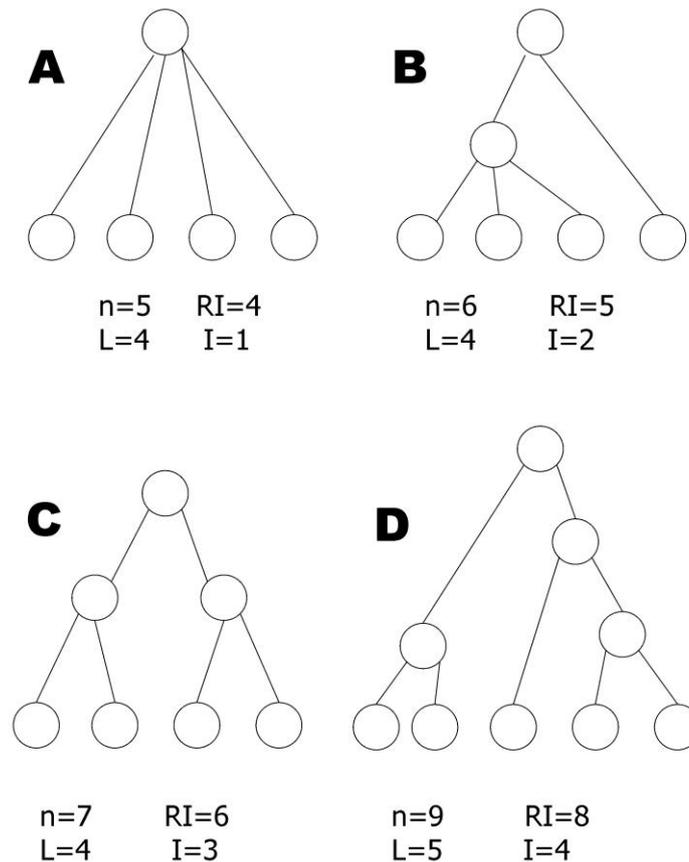


Figure 3.4: The relationship between the number of RIs, nodes (n), internal nodes (I) and leaf nodes (L) in a tree.

3.3 Parameter Learning

3.3.1 Introduction

In order to devise a procedure for solving the above equations and thus obtaining the RI values, we need to look at the pattern of the equations.

If we look at the values of A to G in example two, we find that each corresponds to the multiplication of RIs from the bottom of the tree upwards starting at the leaf that has the same name but in lower case (i.e. leaf a or M_{20}).

The input to the algorithm would be n vectors of known and diagnosed cases given by experts. That is the number of patient records. In example 1, that vector will contain the following:

$$V = (M, a, b, c, d, e, f, g)$$

Eqn. (3.13)

The algorithm proposed here can be divided into two steps: solving for the multipliers (A to G), and then solving for RIs.

3.3.2 ARRIVE Algorithm

3.3.2.1 Induction

Step 1: Solving for Multipliers

The first step of the algorithm will be solving n simultaneous linear equations, where n is the total number of leaves of the GRiST tree (in example 1 that is seven, a through g), see Figure 3.2.

$$M1 = a1. A + b1. B + c1. C + d1. D + e1. E + f1. F + g1. G$$

$$M2 = a2. A + b2. B + c2. C + d2. D + e2. E + f2. F + g2. G$$

.....

$$M7 = a7. A + b7. B + c7. C + d7. D + e7. E + f7. F + g7. G$$

Or:

$$M_k = \sum_{t=1}^7 x_t. X_t$$

Where $X=\{A, B, C, \dots G\}$
 $x=\{a, b, c, \dots g\}$

Eqn. (3.14)

Or in a matrix form:

$$\begin{pmatrix} M1 \\ M2 \\ M3 \\ M4 \\ M5 \\ M6 \\ M7 \end{pmatrix} = \begin{pmatrix} a1 & b1 & c1 & d1 & e1 & f1 & g1 \\ a2 & b2 & c2 & d2 & e2 & f2 & g2 \\ a3 & b3 & c3 & d3 & e3 & f3 & g3 \\ a4 & b4 & c4 & d4 & e4 & f4 & g4 \\ a5 & b5 & c5 & d5 & e5 & f5 & g5 \\ a6 & b6 & c6 & d6 & e6 & f6 & g6 \\ a7 & b7 & c7 & d7 & e7 & f7 & g7 \end{pmatrix} \times \begin{pmatrix} A \\ B \\ C \\ D \\ E \\ F \\ G \end{pmatrix}$$

Eqn. (3.15)

By solving the above equations, we obtain:

$$S = (A, B, C, D, E, F, G)$$

Eqn. (3.16)

Now to determine RIs or R_{xy} , we will use vector S. Note that $R_{00} = 1$

$$S = \begin{pmatrix} A \\ B \\ C \\ D \\ E \\ F \\ G \end{pmatrix} = \begin{pmatrix} R_{10} & R_{20} \\ R_{10} & R_{21} \\ R_{11} & R_{22} \\ R_{11} & R_{23} \\ R_{12} & R_{24} \\ R_{12} & R_{25} \\ R_{12} & R_{26} \end{pmatrix}$$

Eqn. (3.17)

Using the RI properties, we have these equations:

$$R_{10} + R_{11} + R_{12} = 1$$

$$R_{20} + R_{21} = 1$$

$$R_{22} + R_{23} = 1$$

$$R_{24} + R_{25} + R_{26} = 1$$

Eqn. (3.18)

By substitution from S into the above we can solve the system. This also has a pattern.

$$\begin{aligned} A/B &= (R_{10} \cdot R_{20}) / (R_{10} \cdot R_{21}) \\ &= R_{20} / R_{21} \end{aligned}$$

$$\text{So: } R_{21} = (B / A) R_{20}$$

Eqn. (3.19)

Substituting in the relevant equation, we get:

$$R_{20} + R_{21} = R_{20} + (B / A) R_{20} = 1$$

$$\text{Or: } R_{20} (1 + (B/A)) = 1$$

$$\text{Or: } R_{20} ((A+B) / A) = 1$$

$$\text{Thus: } R_{20} = A / (A+B)$$

Eqn. (3.20)

By substituting for R_{20} , we get:

$$R_{21} = 1 - R_{20}$$

Eqn. (3.21)

By continuing in the same manner, we can obtain the rest of RIs. The general form is deduced later in this chapter.

What we ideally want is a systematic approach for solving these equations. This would be the only way to automate the solution (especially with over 200 RIs in a GRiST tree)

Step 2: Solving for RIs

This is the second step of the algorithm.

To do this, we look at a general leaf node and its children. We assume this is the leftmost node in the tree (this doesn't make a difference, just makes notations easier). See Figure 3.5.

The challenge is to advice a systematic way for the solution.

From the S matrix, we know that:

$$S = \begin{pmatrix} A \\ B \\ C \\ D \\ E \\ F \\ G \\ \dots \\ \dots \\ \dots \\ \dots \end{pmatrix} = \begin{pmatrix} R_{10} & \dots & R_{(n-1)0} & R_{n0} \\ R_{10} & \dots & R_{(n-1)0} & R_{n1} \\ R_{10} & \dots & R_{(n-1)0} & R_{n2} \\ R_{10} & \dots & R_{(n-1)0} & R_{n3} \\ R_{10} & \dots & R_{(n-1)0} & R_{n4} \\ R_{10} & \dots & R_{(n-1)0} & R_{n5} \\ R_{10} & \dots & R_{(n-1)0} & R_{n6} \\ \dots & \dots & \dots & \dots \end{pmatrix}$$

Eqn. (3.22)

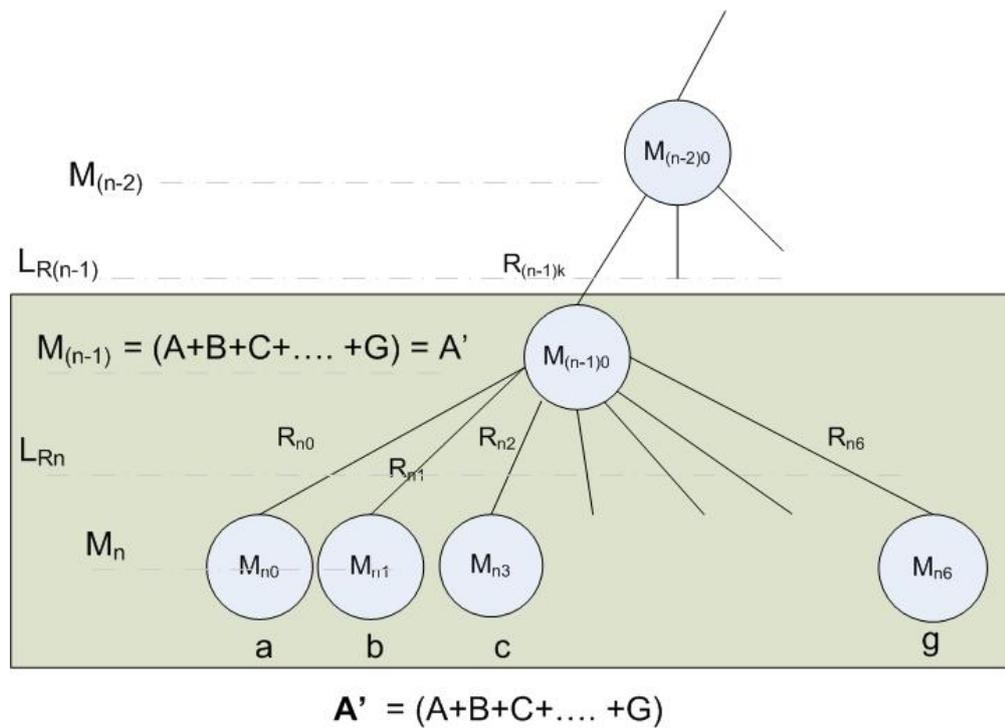


Figure 3.5: A leaf node with seven children.

Now, we take a slice of the matrix, and call it S' for simplicity (it only contains entries starting with R_{10}) :

$$S' = \begin{pmatrix} A \\ B \\ C \\ D \\ E \\ F \\ G \end{pmatrix} = \begin{pmatrix} R_{10} & \dots & R_{(n-1)0} & R_{n0} \\ R_{10} & \dots & R_{(n-1)0} & R_{n1} \\ R_{10} & \dots & R_{(n-1)0} & R_{n2} \\ R_{10} & \dots & R_{(n-1)0} & R_{n3} \\ R_{10} & \dots & R_{(n-1)0} & R_{n4} \\ R_{10} & \dots & R_{(n-1)0} & R_{n5} \\ R_{10} & \dots & R_{(n-1)0} & R_{n6} \end{pmatrix}$$

Eqn. (3.23)

We know from the GRiST tree properties that:

$$R_{n0} + R_{n1} + R_{n2} + R_{n3} + R_{n4} + R_{n5} + R_{n6} = 1$$

Eqn. (3.24)

We will convert the above equation into a function of only one variable, e.g. R_{n0} . To do this we use S' .

$$B/A = R_{n1} / R_{n0}$$

$$R_{n1} = (B/A) \cdot R_{n0}$$

$$C/A = R_{n2} / R_{n0}$$

$$R_{n2} = (C/A) \cdot R_{n0}$$

$$D/A = R_{n3} / R_{n0}$$

$$R_{n3} = (D/A) \cdot R_{n0}$$

$$E/A = R_{n4} / R_{n0}$$

$$R_{n4} = (E/A) \cdot R_{n0}$$

$$F/A = R_{n5} / R_{n0}$$

$$R_{n5} = (F/A) \cdot R_{n0}$$

$$G/A = R_{n6} / R_{n0}$$

$$R_{n6} = (G/A) \cdot R_{n0}$$

Eqn. (3.24)

Substituting in the sum:

$$R_{n0} + (B/A) \cdot R_{n0} + (C/A) \cdot R_{n0} + (D/A) \cdot R_{n0} + (E/A) \cdot R_{n0} + (F/A) \cdot R_{n0} + (G/A) \cdot R_{n0} = 1$$

Or:

$$R_{n0} \cdot (1 + B/A + C/A + D/A + E/A + F/A + G/A) = 1$$

Further:

$$R_{n0} \cdot (A + B + C + D + E + F + G) / A = 1$$

$$R_{n0} = \left(\frac{A}{A + B + C + D + E + F + G} \right)$$

Eqn. (3.25)

If we repeat the above process for the rest of the RIs, we will get:

$$R_{n1} = \left(\frac{B}{A + B + C + D + E + F + G} \right)$$

$$R_{n2} = \left(\frac{C}{A + B + C + D + E + F + G} \right)$$

$$R_{n3} = \left(\frac{D}{A + B + C + D + E + F + G} \right)$$

$$R_{n4} = \left(\frac{E}{A + B + C + D + E + F + G} \right)$$

$$R_{n5} = \left(\frac{F}{A + B + C + D + E + F + G} \right)$$

$$R_{n6} = \left(\frac{G}{A + B + C + D + E + F + G} \right)$$

Eqn. (3.26)

Hence the general rule in the algorithm, to find a certain RI in the leaf nodes, would be:

$$RI_k = \left(\frac{S'(k)}{\sum_k S'(k)} \right)$$

Eqn. (3.27)

Where k is the leaf node index.

Step three: Shrinking the tree

Having found the RIs of the leaf nodes (see Figure 3.5), we can now reduce the children of node $M_{(n-1)0}$ into one leaf of the bigger tree. Since only the total MG value is propagated up the tree, we can now reduce S' further to obtain parent RIs.

From S' , and given that now we know the values of RIs of the leaf:

$$R_{10} \cdot R_{20} \dots\dots\dots R_{(n-1)0} \cdot R_{n0} = A$$

$$R_{10} \cdot R_{20} \dots\dots\dots R_{(n-1)0} = A / R_{n0} = A+B+C+D+E+F+G$$

Eqn. (3.28)

This is confirmed by the rest of the equations as they are now redundant (since we know the RIs of the leaf).

$$R_{10} \cdot R_{20} \dots\dots\dots R_{(n-1)0} \cdot R_{n1} = B$$

$$R_{10} \cdot R_{20} \dots\dots\dots R_{(n-1)0} = B / R_{n1} = A+B+C+D+E+F+G$$

Eqn. (3.29)

Now S' is no longer needed for the parent node $M_{(n-1)0}$.

But the parent will have two or more children, and if this process is repeated for the other siblings of our node above, the parent will have the following equations:

$$R_{10} \cdot R_{20} \dots\dots\dots R_{(n-1)0} = A+B+C+D+E+F+G$$

$$R_{10} \cdot R_{20} \dots\dots\dots R_{(n-1)1} = H+ I + J$$

$$R_{10} \cdot R_{20} \dots\dots\dots R_{(n-1)2} = K+ L + M + N + O$$

.....

And so on.

Or, in general form:

$$M_{(n-1)0} = \sum_0^c M_{nk}$$

Where: c = number of children of $M_{(n-1)0}$

Eqn. (3.30)

The above equations are then solved again using the same algorithm (it can be seen it is recursive/ iterative).

When the top of the tree is reached, all RIs will be known.

3.3.2.2 The Algorithm in a nutshell

To generalize the algorithm as of the tree in Figure 1, we summarize it as follows.

Inputs:

$$V1 = (M1, M_{n01}, M_{n11}, \dots, M_{nk1})$$

To

$$Vk = (Mk, M_{n0k}, M_{n1k}, \dots, M_{nkk})$$

Eqn. (3.31)

Where:

M1 to Mk : are the k different cases outcomes. K is defined below.

M_{ny} : is the input MG at the leaf on the nth level (lowest level) of the GRiST tree of the yth input vector (Vy). We need k vectors to solve the resulting k simultaneous equations.

K = the number of leaf nodes of the GRiST tree.

Outputs:

RI values.

Procedure:

Step one:

Solve the following simultaneous equations:

$$\begin{pmatrix} M1 \\ M2 \\ M3 \\ \dots \\ \dots \\ Mk \end{pmatrix} = \begin{pmatrix} M_{n01} & M_{n11} & M_{n21} & \dots & \dots & \dots & M_{nk1} \\ M_{n02} & M_{n12} & M_{n22} & \dots & \dots & \dots & M_{nk2} \\ M_{n03} & M_{n13} & M_{n23} & \dots & \dots & \dots & M_{nk3} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ M_{n0k} & M_{n1k} & M_{n2k} & \dots & \dots & \dots & M_{nkk} \end{pmatrix} \times \begin{pmatrix} A1 \\ A2 \\ A3 \\ \dots \\ \dots \\ Ak \end{pmatrix}$$

Eqn. (3.32)

The above matrix is kXk in dimension.
The solution yields vector S:

$$S = \begin{pmatrix} A1 \\ A2 \\ A3 \\ \dots \\ \dots \\ \dots \\ Ak \end{pmatrix}$$

Eqn. (3.33)

Step two:

We use S' to denote a sub tree of each node at level (n-1).
Hence we have: S'1 to S'h where h is the number of nodes at level (n-1) in the GRiST tree.
For each subtree, S'j, we solve to find its RIs.

$$R_{nr} = \left(\frac{S'j(r)}{\sum_r S'j(r)} \right)$$

Eqn. (3.34)

Where r is the leaf node number (e.g. a, b, c,..). r starts from 0 to the number of leaves of node j at level (n-1). j = 0, ...,h.

R has values from 0 to k at the leaf level of the tree.

Step three:

Once RIs are found on that level, the tree can now be shrunk by a level, by replacing the leaf level and its parents with one new level.

MGs for the new level are calculated by the sum of corresponding (A+B+C+....) of each node. (See Figure 3.5).

So,

$$M_{(n-1)h} = \sum_r S' j(r)$$

Eqn. (3.35)

Once the new parent MGs are found for the new level, we can go to step two and repeat step two and three for the new tree. This process is continued n times. At the end, we will have all RIs in the tree.

A simple flowchart of the algorithm is outlined on the next page (Figure 3.6).

If more training records are available, than the minimum required, then iARRIVE algorithm needs to be used, which is introduced in the next section.

Complexity:

The main step of the algorithm is the Gaussian elimination of complexity $O(n^3)$ [ATKI, 89], where n is the number of leaf nodes in the tree. Back propagation will involve k iterations, where k is the total number of RI values in the tree.



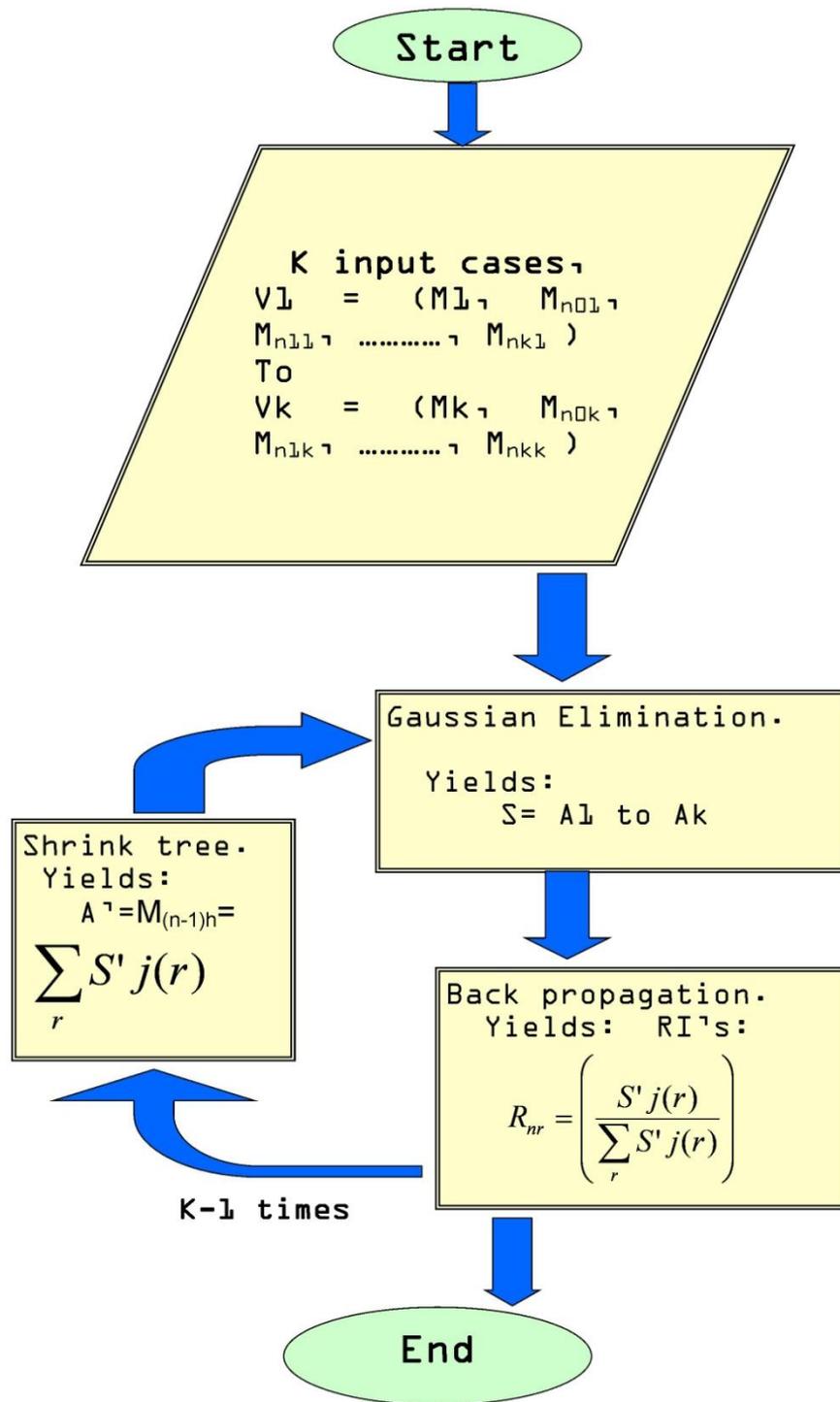


Figure 3.6: ARRIVE flow chart.

3.3.2.3 Case Study

In this study we will use our algorithm to calculate the RI values in the tree shown in Figure 3.6. The tree has six leaves (A to F), hence we need six training cases. We use a synthetic training data set, in the form of the following matrix equation (as in Eqn. (3.32)):

$$\begin{pmatrix} 0.3 \\ 0.4 \\ 0.9 \\ 0.7 \\ 0.8 \\ 0.1 \end{pmatrix} = \begin{pmatrix} 0.1 & 0.3 & 0.4 & 0.2 & 0.3 & 0.6 \\ 0.1 & 0.2 & 0.5 & 0.4 & 0.6 & 0.2 \\ 0.2 & 0.1 & 0.3 & 0.7 & 0.8 & 0.7 \\ 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 0.3 \\ 0.2 & 0.3 & 0.4 & 0.3 & 0.9 & 0.2 \\ 0.3 & 0.1 & 0.6 & 0.5 & 0.5 & 0.4 \end{pmatrix} \times \begin{pmatrix} A \\ B \\ C \\ D \\ E \\ F \end{pmatrix}$$

Eqn. (3.36)

Using Gaussian Elimination, we obtain:

$$\begin{aligned} A &= -0.44 & D &= 0.44 \\ B &= 0.92 & E &= 0.964 \\ C &= -1.067 & F &= 0.196 \end{aligned}$$

Eqn. (3.37)

Note that we use 3 decimal points approximation for simplicity (rounding).

Using Equation (3.34) and the propagation technique in (3.35), we obtain all the RI values as in Figure 3.7.

To verify the model, we use the first training case (first line in (3.36)) as an input (in blue in each leaf node). Propagating through the decision tree using the new RI values, we finally reach a decision ($M = 0.298$). This is almost the same as the desired output in the original test case, in equation (3.36), i.e. 0.3. The error is due to approximation and using only three decimal points precision. ($0.3 - 0.298 = 0.02$). This amounts to only 6.66 %, which is acceptable in GRiST, as it makes no sense to be any more accurate than one decimal point with respect to RIs and their influence. A more detailed analysis on real data is presented in the Chapter Six, Results.

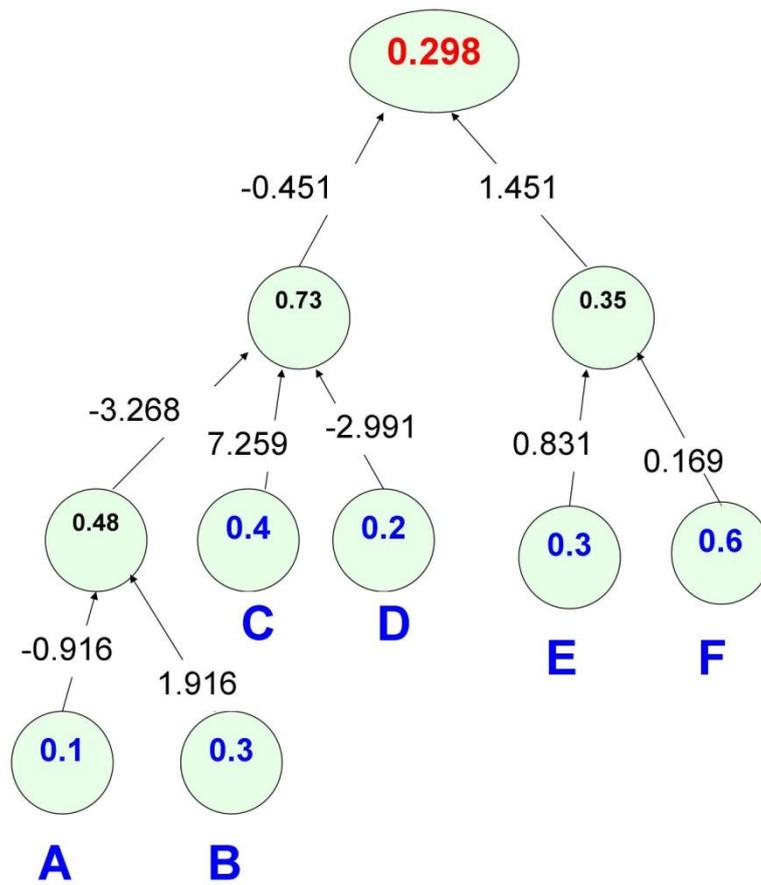


Figure 3.7: A sample decision sub-tree.



3.3.3 iARRIVE Algorithm

In ARRIVE [18], we developed a mathematical representation for GRiST that provided the basis for a model to solve. We then devised an algorithm to find the solution for that model and, ultimately, all RI values in the tree. The restriction of the ARRIVE model is that it requires exactly k cases, where k is the number of leaf nodes in the GRiST tree.

This means that if more cases than k are available, we will have to filter the inputs to chose only k cases, to use the algorithm. This can be done in several ways by, for example averaging techniques to combine several cases together or using a credibility or confidence index to choose the “best” k cases to use.

The disadvantages are that some cases will either be discarded or have less influence on the overall solution and incorporating new cases into the RI values solution is difficult. There is also the problem of learning from new cases, as in case of ARRIVE the whole procedure needs to be started over. In order to overcome these restrictions, we have developed an incremental version of ARRIVE, we call iARRIVE [HEGA, 08].

Assume we have t previously known cases where $t > k$. In this case, the matrix representation of the tree will look like:

$$\begin{pmatrix} M1 \\ M2 \\ M3 \\ \dots \\ \dots \\ \dots \\ Mt \end{pmatrix} = \begin{pmatrix} M_{n01} & M_{n11} & M_{n21} & \dots & \dots & \dots & M_{nk1} \\ M_{n02} & M_{n12} & M_{n22} & \dots & \dots & \dots & M_{nk2} \\ M_{n03} & M_{n13} & M_{n23} & \dots & \dots & \dots & M_{nk3} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ M_{n0t} & M_{n1t} & M_{n2t} & \dots & \dots & \dots & M_{nkt} \end{pmatrix} \times \begin{pmatrix} A1 \\ A2 \\ A3 \\ \dots \\ \dots \\ \dots \\ Ak \end{pmatrix}$$

Eqn. (3.38)

or in the general mathematical representation:

$$M_i = M_{n0i} \cdot A_1 + M_{n1i} \cdot A_2 + M_{n2i} \cdot A_3 + \dots + M_{nki} \cdot A_k$$

Eqn. (3.39)

where $i=0$ to t (t is the number of equations).

A1 to Ak are the unknowns, or the coefficients of Equation (3.39), and k = number of leaf nodes in the GRiST tree. Since $t > k$ (more test cases than the number of unknowns), Equation (3.39) cannot be solved as a Gaussian elimination problem.

This is a multivariate regression problem with k variables (M) and k regression coefficients (A1 to Ak).

The problem is finding the initial coefficients at the leaf nodes (A1 to Ak), which must be done using iARRIVE, but after that, we revert to the same approach as for ARRIVE to find the higher level RIs (step two and three). The first step is only required to calculate the coefficients at the leaf nodes, and this can be done in different ways. After that, back propagation is used to calculate the RIs, regardless of how the leaf node coefficients were calculated in the first step. This is a very important and useful characteristic of our methodology and overall approach to solving the RI problem. Once the problem is converted into a different domain, step 1, solving the simultaneous equations can be carried out using many possible methods. We have chosen the once most suitable and straight forward to give a robust solution. Once the new unknowns are determined, the second part of the algorithm is independent and finds the RI values recursively regardless of the way the unknowns are found in the initial step. We believe that this is a powerful feature in our approach and makes the algorithm more flexible and adaptable to other problems and knowledge domains. It also allows for two systems with identical inputs and outputs to be interpreted in different ways, by changing the internal tree connections in step two and three. Note that step one does not take into account the path of the RIs, they are simply replaced with a new variable.

There are many methods for solving multivariate regression problems (e.g. [SHEL, 72], [GOVI, 06], [KLEI, 98]) and several software packages that provide the required functions such as linear, splines, ([MATLAB], [NLREG], [CAMO, 08]). For a more robust solution we have chosen Matlab [MATLAB].

For simplicity, we rewrite Equation (3.39) in a general form as:

$$M = m_1 \cdot A_1 + m_2 \cdot A_1 + m_3 \cdot A_1 + \dots + M_k \cdot A_k$$

Eqn. (3.40)

Or in matrix form, for t cases:

$$[M] = [m] \cdot [A]$$

Eqn. (3.41)

This yields:

$$[A] = ([m]'. [m])^{-1} [m]'. [M]$$

Eqn. (3.42)

Hence we obtain all As and can continue as in ARRIVE to obtain RIs.

To generalize the algorithm using the notation in Figure 3.3, we summarize it as follows:

Inputs:

$$V1 = (M1, M_{n01}, M_{n11}, \dots, M_{nk1})$$

To:

$$Vt = (Mt, M_{n0t}, M_{n1t}, \dots, M_{nkt})$$

Eqn. (3.43)

where M1 to Mt are the t different cases outcomes and $t > k$. M_{ny} is the input MG at the leaf on the nth level (lowest level) of the GRiST tree of the yth input vector (Vy) and $y = 1$ to t. K = the number of leaf nodes of the GRiST tree.

Outputs:

RI values.

Step one:

Solve the following equations for A, using Multivariate (multiple) Regression methods:

$$\begin{pmatrix} M1 \\ M2 \\ M3 \\ \dots \\ \dots \\ \dots \\ Mt \end{pmatrix} = \begin{pmatrix} M_{n01} & M_{n11} & M_{n21} & \dots & \dots & \dots & M_{nk1} \\ M_{n02} & M_{n12} & M_{n22} & \dots & \dots & \dots & M_{nk2} \\ M_{n03} & M_{n13} & M_{n23} & \dots & \dots & \dots & M_{nk3} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ M_{n0t} & M_{n1t} & M_{n2t} & \dots & \dots & \dots & M_{nkt} \end{pmatrix} \times \begin{pmatrix} A1 \\ A2 \\ A3 \\ \dots \\ \dots \\ \dots \\ Ak \end{pmatrix}$$

Eqn. (3.44)

The M matrix is t by k in dimension and the solution of Equation (3.44) yields vector S = A1 to Ak.

We then continue with steps two and three as in the previous section.

Complexity of the algorithm:

As in simple linear regression with least-squares, our algorithm has time complexity of $O(k.n^2)$ [ATKI, 89], where k is the sample size, and n is the number of parameters to be estimated, in our case that is the number of leaf nodes in the tree.

3.3.4 Error in Solution

In this section, we analyse the output prediction method and the error in the output decision due to missing data or sample inconsistencies (i.e., noise in input data, inaccurate inputs,..etc).

To conduct analysis on the error in the output, due to using the newly inducted RIs, we can use the mapping of the interpolated function. This will also give an indication of the quality (consistency) of the input data. If the mapping fits most inputs, intuitively this means that the input data is clean, with not much noise.

Assume A is the mapping matrix from iARRIVE :

$$\begin{pmatrix} M1 \\ M2 \\ M3 \\ \dots \\ \dots \\ Mt \end{pmatrix} = \begin{pmatrix} M_{n01} & M_{n11} & M_{n21} & \dots & \dots & \dots & M_{nk1} \\ M_{n02} & M_{n12} & M_{n22} & \dots & \dots & \dots & M_{nk2} \\ M_{n03} & M_{n13} & M_{n23} & \dots & \dots & \dots & M_{nk3} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ M_{n0t} & M_{n1t} & M_{n2t} & \dots & \dots & \dots & M_{nkt} \end{pmatrix} \times \begin{pmatrix} A1 \\ A2 \\ A3 \\ \dots \\ \dots \\ Ak \end{pmatrix}$$

Eqn. (3.45)

$$M_i = (M_{0i} \ M_{1i} \ M_{2i} \ \dots \ M_{ki})$$

M_i is an input matrix (one of the training cases) and M is the corresponding output judgement. Assume m_1 to m_n are the test cases used to generate A , with M_1 to M_n as the corresponding outputs.

The error between any actual M_i and calculated $M'i$ would be:

$$\xi = M_i - M'i$$

$i=1$ to n .

Eqn. (3.46)

But $M'i = m_i \cdot A$

Eqn. (3.47)

Hence the total absolute error for the n test cases would be:

$$\xi_{abs} = \sum_{i=1}^n |Mi - mi.A|$$

Eqn. (3.48)

The average error for this interpolation would be:

$$\xi_{ave} = \frac{\sum_{i=1}^n |Mi - mi.A|}{n}$$

Eqn. (3.49)

The total average percentage error would be:

$$\% \xi = \frac{\sum_{i=1}^n |Mi - mi.A|}{\sum_{i=1}^n |mi.A|} \times 100$$

Eqn. (3.50)

This is very useful in terms of analysing the original data. If the average percentage error is relatively high, this could indicate the accuracy (or lack of it) of the results. At the same time, a higher error in the output may indicate a problem in the original training set.

This could be due to noise in the original data, or discrepancies in the information obtained from the different experts. This does not necessarily indicate that the original data is inconsistent or inaccurate. In fact, it may be a matter of difference of opinions among the experts, which does occur in real life.

3.3.5 Case Study

We will use our algorithm to calculate the RI values in the tree shown in Figure 3.8. The tree has six leaves (A to F), hence we need at least six training cases. In this example, we use nine.

We use a synthetic training data set, in the form of the following matrix equation (as in (3.38)):

$$\begin{pmatrix} 0.3 \\ 0.4 \\ 0.9 \\ 0.7 \\ 0.8 \\ 0.1 \\ 0.1 \\ 0.2 \\ 0.3 \end{pmatrix} = \begin{pmatrix} 0.1 & 0.3 & 0.4 & 0.2 & 0.3 & 0.6 \\ 0.1 & 0.2 & 0.5 & 0.4 & 0.6 & 0.2 \\ 0.2 & 0.1 & 0.3 & 0.7 & 0.8 & 0.7 \\ 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 0.3 \\ 0.2 & 0.3 & 0.4 & 0.3 & 0.9 & 0.2 \\ 0.3 & 0.1 & 0.6 & 0.5 & 0.5 & 0.4 \\ 0.2 & 0.2 & 0.7 & 0.5 & 0.5 & 0.3 \\ 0.3 & 0.1 & 0.4 & 0.5 & 0.3 & 0.1 \\ 0.09 & 0.2 & 0.4 & 0.2 & 0.4 & 0.5 \end{pmatrix} \times \begin{pmatrix} A \\ B \\ C \\ D \\ E \\ F \end{pmatrix}$$

Eqn. (3.51)

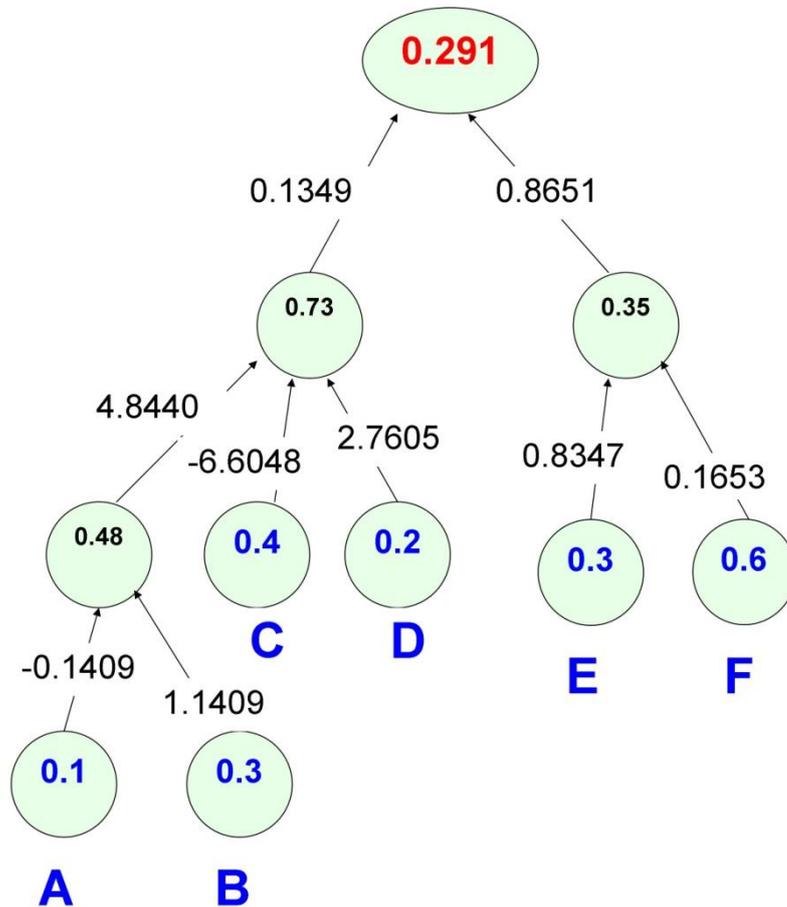


Figure 3.8: A sample decision tree.

Using multiple regression (Equation 3.42), we obtain:

$$A = -0.114$$

$$D = 0.461$$

$$\begin{aligned} B &= 0.923 & E &= 0.894 \\ C &= -1.103 & F &= 0.177 \end{aligned}$$

Eqn. (3.52)

Note that we use four decimal points approximation for simplicity (rounding).

Using Equation (3.34) and the propagation technique in (3.35), we obtain all the RI values as in Figure 3.8.

To verify the model, we use the first training case (first line in (3.51)) as an input (inside each leaf node (A, B, C, ..)). Propagating through the decision tree using the new RI values, we finally reach a decision ($M = 0.291$). This is almost the same as the desired output in the original test case, in equation (3.51), i.e. 0.3. The error is due to approximation and using only three decimal points precision ($0.3 - 0.291 = 0.009$). This amounts to only 3%, which indicates a good interpolation and acceptable approximation.

The error is due to the use of first order polynomial regression and the inaccuracy of the original data. By using higher order regression and analysing the data before hand, the error could be reduced further as will be shown in the next chapter.



3.3.6 Proof of Concept

The original structure and the GRiST model [BUCK, 08] rely on set theory and fuzzy logic [ZADE, 65]. One of the main assumptions that should hold if our algorithm and the mathematical model presented in this chapter are semantically correct, is the complement set. From the definition of the Risk in GRiST, a complement of a certain risk would result from complementing all the inputs to that risk [BUCK, 08]. If the total risk for example is 0.8 (at the top of the tree), then the complement risk, defined as $1 - 0.8 = 0.2$ should result from complementing every input to the original tree that produced the 0.8 risk. To illustrate this, assume the GRiST tree in Figure 3.9.

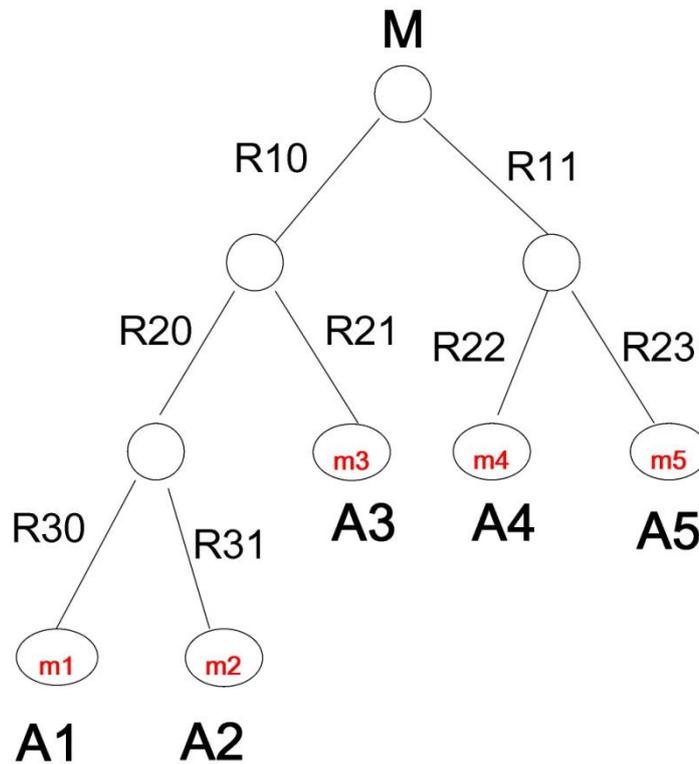


Figure 3.9: A sample GRiST tree.

(A1, A2, A3,) represent the multipliers, as per Equation (3.33).

Thus :

$$A = \begin{pmatrix} A1 \\ A2 \\ A3 \\ A4 \\ A5 \end{pmatrix}$$

Eqn. (3.55)

$$A1 = R10.R20.R30$$

$$A2 = R10.R20.R31$$

$$A3 = R10.R21$$

$$A4 = R11.R22$$

$$A5 = R11.R23$$

Eqn. (3.56)

The fundamental assumption is that the same tree, with complement inputs (m_i'), should produce a complement judgement (M'). (See Figure 3.10).

$$\text{where : } m_i' = 1 - m_i$$

$$\text{and } M' = 1 - M$$

To prove this, we go back to Equation (3.41):

$$M = m.A$$

Now we replace m with ($m' = 1 - m$):

$$M' = m'.A = (1 - m).A$$

Eqn. (3.57)

Or, from matrix calculus:

$$M' = I.A - m.A$$

Eqn. (3.58)

Or:

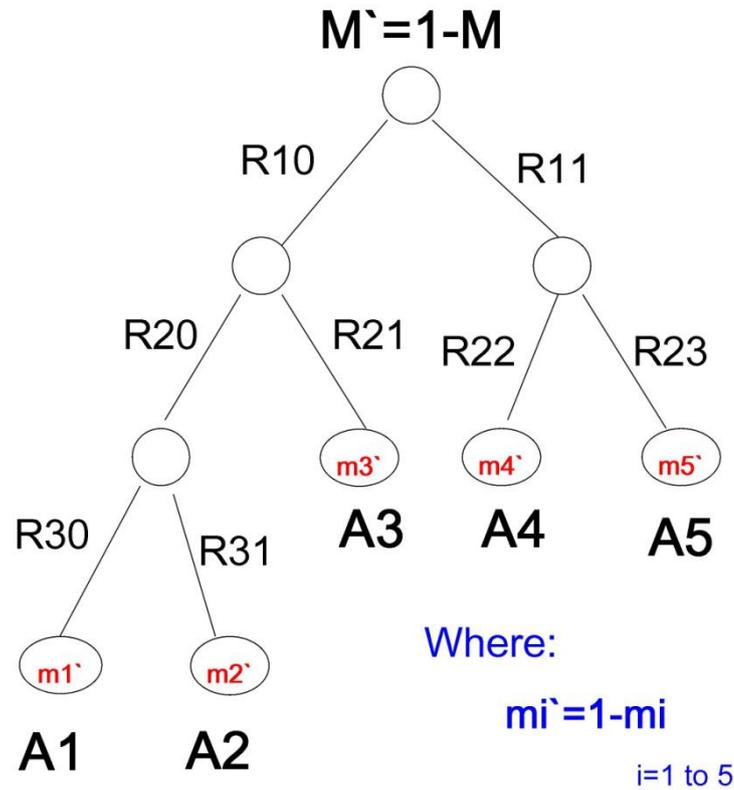


Figure 3.10: The complement tree of the tree in Figure 3.8.

$$M' = I.A - M$$

Eqn. (3.59)

where I is the unity matrix:

$$I = (1 \ 1 \ 1 \ 1 \ 1)$$

Eqn. (3.60)

From matrix operations:

$$I.A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} A1 \\ A2 \\ A3 \\ A4 \\ A5 \end{pmatrix}$$

Eqn. (3.61)

$$I.A = \sum_{i=1}^5 A_i$$

Eqn. (3.62)

If the expression in Equation (3.62) evaluates to 1 then our model is correct, and the complement tree condition is satisfied.

By substituting for RI values in (3.62) using (3.56) we obtain:

$$I.A = R_{10}.R_{20}.R_{30} + R_{10}.R_{20}.R_{31} \\ + R_{10}.R_{21} + R_{11}.R_{22} + R_{11}.R_{23}$$

Eqn. (3.63)

By grouping:

$$I.A = R_{10}.R_{20}.(R_{30} + R_{31}) \\ + R_{10}.R_{21} + R_{11}.(R_{22} + R_{23})$$

Eqn. (3.64)

From RI properties, Equation (3.7), we know that: $\sum_{y=0}^{Z_{xn}} R_{xy} = 1$

So, Equation (3.63) becomes:

$$I.A = R_{10}.R_{20} + R_{10}.R_{21} + R_{11}$$

Eqn. (3.65)

Grouping again:

$$I.A = R_{10}.(R_{20}+R_{21}) + R_{11}$$

$$\text{Or } I.A = R_{10} + R_{11}$$

Eqn. (3.66)

And from Equation (3.7) this yields:

$$I.A = 1$$

Eqn. (3.67)

The above process is valid for any GRiST tree, as grouping is done bottom up.

This proves that the complement tree assumption holds by substituting I.A from Eqn. (3.67) in Eqn. (3.57)

$$M^c = 1 - M$$

Eqn. (3.68)

This is a necessary proof of concept according the properties of GRiST [BUCK, 08]. We have shown that our model is correct mathematically, and in this section, we have proven its semantic correctness on the basis of the constraints on GRiST and the way it handles fuzzy set membership grades [BUCK, 08] .

Equations (3.59) and (3.68) also prove the important property that If all input MG =1, the total M will also equal 1. If input risk is maximum then output risk is also maximum. If all input MGs are 0, the M=0.



3.4 Summary

In this chapter we have tackled the main problems associated with parameter elicitation in hierarchical structures representing knowledge and expertise.

We have introduced the GRiST decision tree structure and highlighted the challenges of determining the relative influence values or parameters in the tree.

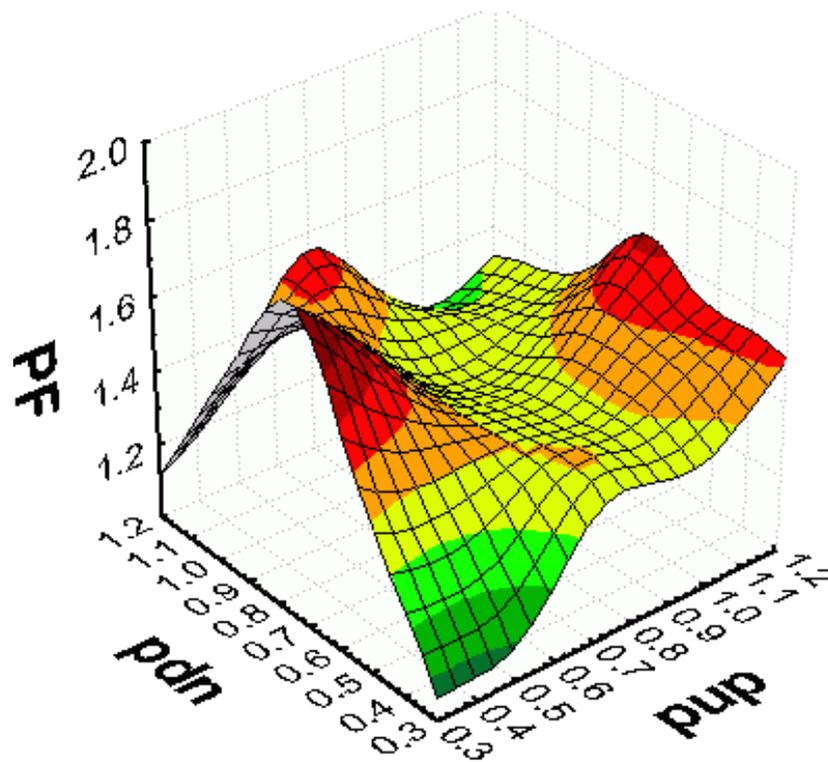
We introduced our ARRIVE (an Algorithm for Robust Relative Influence Values Elicitation) as well as an incremental extension of the method, iARRIVE. Our algorithm consists of two stages; the first stage transforms the problem into a different domain with a new set of unknowns. Once these are determined, the method uses back propagation to calculate the RIs in the tree. This has many advantages, primarily the flexibility of using several possible methods to solve step one.

In the next chapter, we address the problem of varying expert opinions and ways of concatenating their input and expertise to the tree in GRiST, which is vital for accurate RIs calculations using iARRIVE.



Chapter 4

Data Conditioning



This Chapter covers the following:

- Data Conditioning
- Quantifying Experts inputs
- MGM: Membership Grade Modulation Algorithm

4.1 Introduction

Experts in general may have different opinions based on their experience, background and expertise. This is very apparent in the clinical field. In the case of GRiST, over 40 experts contributed to the knowledge elicitation process, which inevitably meant varied inputs.

In this chapter, we develop the MGM (Membership Grade Modulation) Algorithm, which will be used to combine the various expert opinions into one usable function. We also present general approaches to measure the consistency of the expert input and ways of filtering out or smoothing the overall expertise.

In order to be able to map data from the GRiST questionnaire into usable information for the algorithms presented in the previous chapter, (i.e., numbers), we need a knowledge representation method. One possible candidate could be a rule base. But due to the large variations in input data and the large number of possibilities (on average 200 questions, with 10 possibilities each on average), this is not feasible. Another reason is that any modification in the rule base would be very difficult to track and map.

4.2 Membership Grades (MG)

We have opted for fuzzy logic [ZADE, 65] to do the mapping from expert inputs to the questionnaires to the actual data used by iARRIVE. Membership functions will provide us with membership grades (MG, or fuzzy-set membership grades) to map any inputs to numbers for the GRiST tree (Figure 4.1).

The advantage of using fuzzy set membership grade functions [ZADE, 65] is that a simple amendment to them can alter the representation of expertise. Experts provide graphs representing the effect of the input values on the decision or risk. The functions (e.g. Figure 4.2) are provided by the expert clinicians and we have developed tools for helping to elicit them (See Appendix C). Input patient values are mapped to their associated fuzzy function to obtain a value or MG.

The main challenge is to concatenate the functions: experts' opinions may vary but we could only use one function (MG) for each leaf node.

As in Figure 4.1, MG2 may originally have two different functions provided by two different experts. In the real life case of GRiST, we had 44 Experts with potentially up to 44 different functions for the same MG. These have to be concatenating somehow to come up with one function representing consensus (MG2).

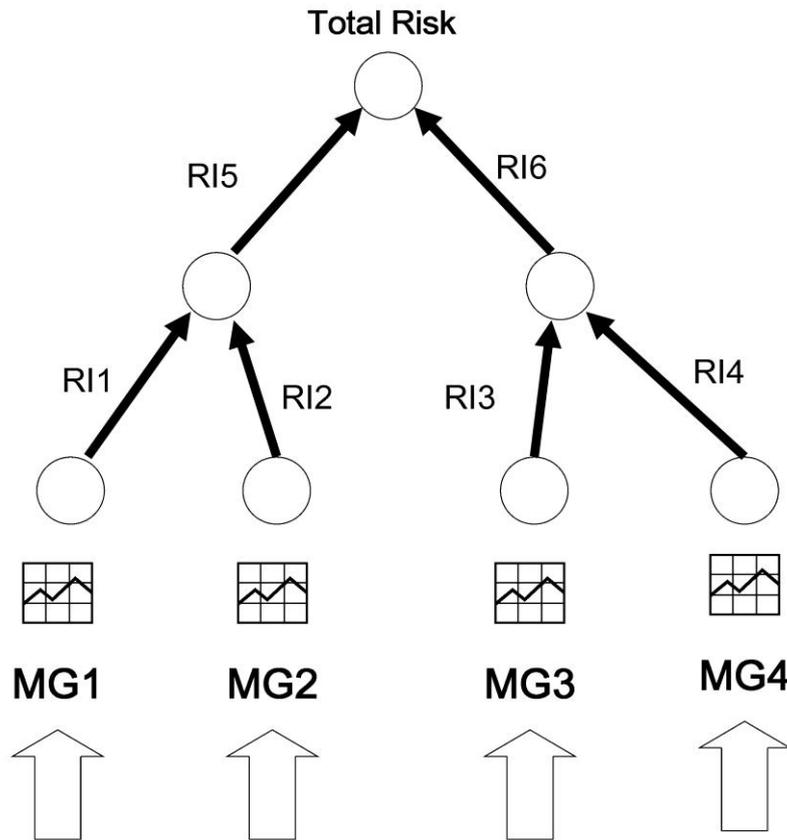


Figure 4.1: Membership functions as inputs to a GRiST tree.

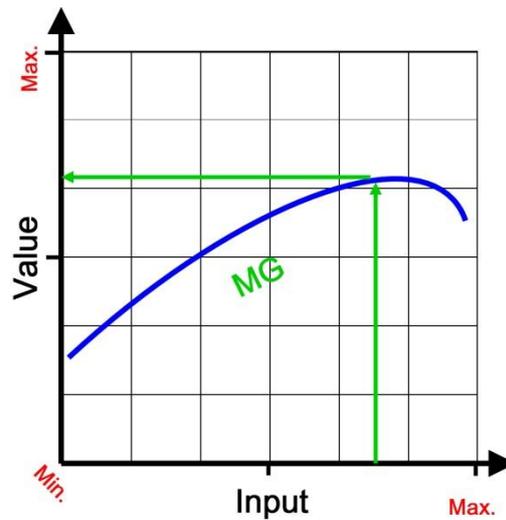


Figure 4.2: A Membership Grade (MG) function.

Our aim is to find a function that represents the concatenation of all opinions (note: this could be n experts with n opinions in the general case). As this function is to find some sort of a consensus, it should provide a value V somewhere between the two values provided by the experts, V_1 , V_2 (Figure 4.3). We also need a measure of the accuracy of this solution.

4.3 MGM: Membership Grade Modulation Algorithm

4.3.1 Background

In real life, no two people entirely agree. This couldn't be truer in the medical and psychiatric fields. Experts always have differences, based on their experience, background and many other factors. In the GRiST model, in order to obtain the various parameters in the tree, we have had the cooperation of over 40 experts. This meant that potentially, each of the MG functions in Figure 4.1 could have over 40 different possibilities.

The challenge is to find a method of combining them, to fairly represent all experts with acceptable deviations. We also need to be able to judge the quality and the consensus in the original data supplied by the experts.

To do this, we developed the MGM Algorithm.

4.3.2 The Algorithm

Our algorithm is called the MGM: Membership Grade Modulation algorithm. For the purposes of this work, MGM will represent the concatenated function that is calculated by the algorithm. Assume the two functions provided by the experts are MG1 and MG2. We need to find a function MGM that combines both. We represent a simple two function example in Figure 4.3. MG1 is the top line, MG2 is the lower line. V1 and V2 represent two values for the same input based on two experts' functions (represented by two lines). V represents the concatenation between the two values.

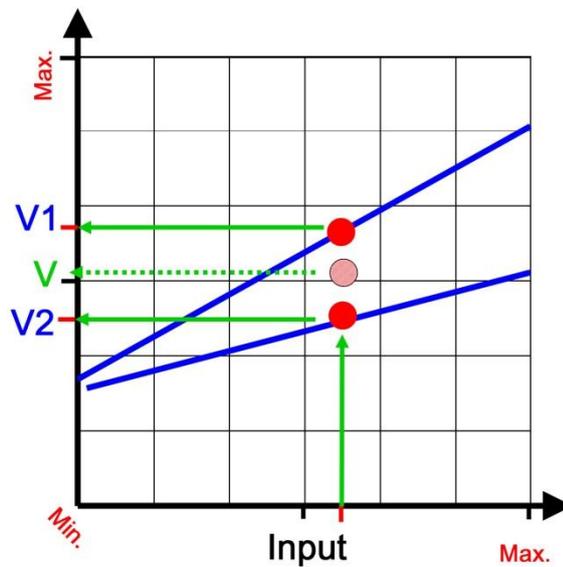


Figure 4.3: MGM tries to find V, a concatenation of V1 and V2.

One straight forward approach would be to find the average of V1 and V2, and this should represent the value V. We will use this approach.

$$\text{Thus : } V = \frac{V1 + V2}{2}$$

Eqn. (4.1)

In a more general case, for n functions:

$$V = \frac{\sum_{k=1}^n V_k}{n}$$

Eqn. (4.2)

To have a more consistent approach, we need to find MGM, a new function that represents the above equation. To do this we use function properties [11], which means that functions can be added to each other like single values. Thus:

$$MGM(x) = \frac{MG1(x) + MG2(x)}{2}$$

Eqn. (4.3)

In a generalized form, for n functions:

$$MGM(x) = \frac{1}{n} \sum_{k=1}^n MGk(x)$$

Eqn. (4.4)

We now further analyze the graph in figure 4.3. What if expert 1 is more prominent or has more weight or expertise in the field of this particular MG (or question in the GRiST questionnaire). The function described in equation 4.4 assumes MG1 and MG2 have equal weight.

To address this concern, we introduce a new parameter, w. This will represent the weight of each function. This will determine its contribution to the overall new MGM, thus the experts' contribution. We use normalized weights, as we need to represent the weight of each expert compared to the others in the set. Thus the sum of the weights should be 1. We can then add the MGs multiplied by their respective weights to get the overall MGM. Or:

$$MGM(x) = \sum_{k=1}^n w_k MGk(x)$$

Eqn. (4.5)

Where: $\sum_{k=1}^n w_k = 1$

Eqn. (4.6)

If all experts have equal weights, Equation 4.5 becomes Equation 4.4, as each weight will be 1/n.

This new function, MGM, represents the concatenation of all experts into one function, that can now be used straight forward in the GRiST tree to defuzzify input questionnaires and calculate risk.

Extension:

To generalize the MGM algorithm, we will use polynomial regression methods [KLEI, 98] to fit a polynomial to the combined graph representing points from all MG graphs. Regression will produce an overall MG function that will best represent the combined data pool and minimize the overall error.

Regression will produce an MG function that will best represent the combined data pool and minimize the overall error. The order of this function can be set to obtain better representation, which gives more flexibility. In practice, though, orders higher than three don't offer much improvement of the fit [DRAP, 98] (as we will show in the case study).

The advantage of using regression in real life is that it will produce a function for the overall MG even if the original data is scattered (which is usually the case, as the input MG curve functions are unknown. The input data or the MG functions are provided by the experts using our online graphical Elicitation Tool (See appendix C).

These are represented as points connected by straight lines. This was designed based on the feedback of the clinical trial, to simplify the input process for clinicians. The graphs are stored as a collection of points and then digitized at a fixed frequency using the software.

Complexity:

As in simple linear regression with least-squares, our algorithm has time complexity of $O(k.n^2)$ [ATKI, 89], where k is the sample size, and n is the number of parameters to be estimated, in our case that is the number of leaf nodes in the tree.

4.3.3 Error Estimation

As in the case of any real life expert system, the above algorithm depends highly on the consistency of the data. Since the opinions of medical experts are highly unlikely to be unified, the functions supplied by the experts (MG) will vary, hence the need for MGM. If the differences are minor between experts, MGM will produce consistent results.

The problem would arise in the case of large discrepancies in expert opinions and thus MG functions. In this case, MGM might not represent any of them and the output would be confusing. This is however not a flaw in the algorithm itself. It is a natural conclusion which we could reach as well if we analyze the inputs. If one expert says high and the other says low, it is natural that the outcome would be none, or close to zero (as they negate each other)! We need a method to detect these discrepancies and raise alerts if they exist in the data.

To quantify these discrepancies, we will use a distance measure to measure the average distance between curves. This is a complicated process and might require a vast amount of calculation [KLEI, 98]. Instead we propose a different method that is more suited to our case here, as we only need an indication of the distance and not the exact value. This will in turn give an indication of both the error in the output data and the consistency of the experts' elicitation. See Figure 4.4.

To measure the distance between two curves we will use the area between them as an indication (Figure 4.5). Assuming the two graphs are fairly similar, (i.e. similar slopes and shape) which is the general case in our test set as we assume the clinicians will agree to some extent but may deviate at certain points), the larger the area between the two curves, the larger the average distance.

To calculate the area enclosed between any two curves, we can look at it as the difference between the area under curve A and the area under curve B. An area under a curve (see Figure 4.5) can be calculated by integrating the curve between two points [GIAQ, 03]. For an MG, this would be:

$$A = \int_{x=\min}^{\max} MG(x)$$

Eqn. (4.7)

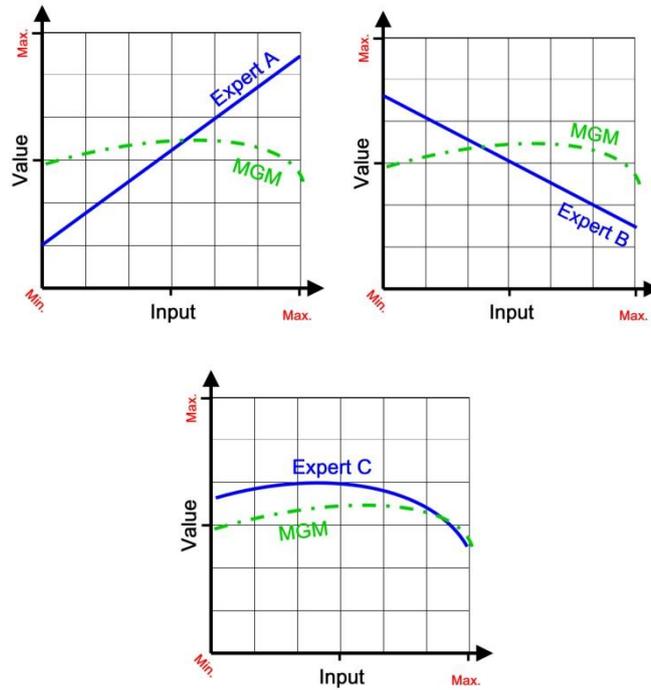


Figure 4.4a: A poor set of MGs results in poor MGM.

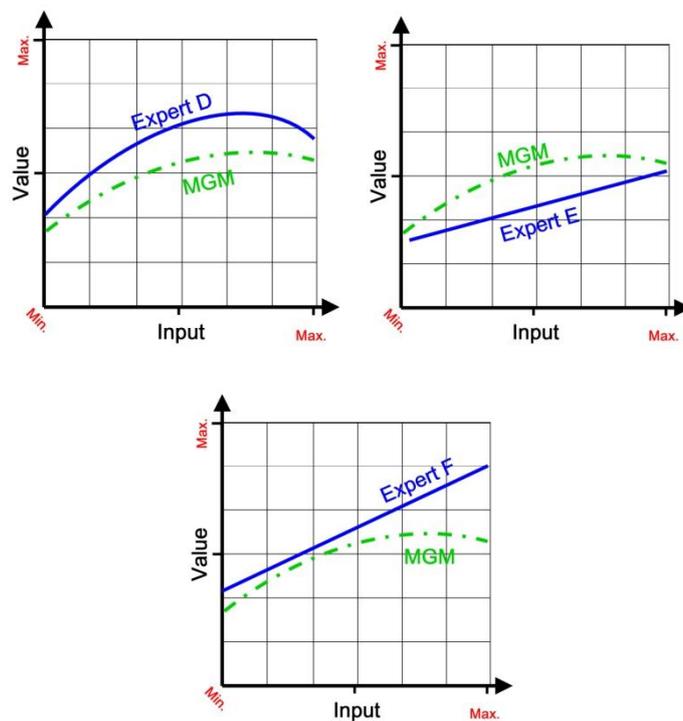


Figure 4.4b: A good set of MGs results in consistent MGM.

Hence, to calculate the area between MG1 and MGM (see Figure 4.5):

$$A_1 = \xi_1 = \left| \int_{x=\min}^{\max} MG_1(x) - \int_{x=\min}^{\max} MGM(x) \right|$$

Eqn. (4.8)

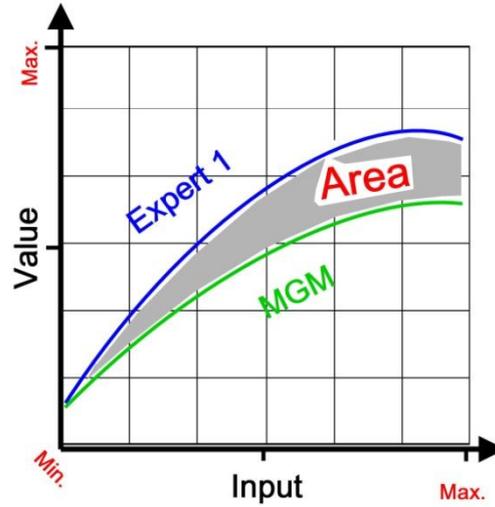


Figure 4.5: Area between two curves.

Note that we are interested in the positive value for the area and need the absolute value. For n MGs, the indication of total error and the accuracy of results would be:

$$\xi = A_{total} = \sum_{k=1}^n \left(\left| \int_{x=\min}^{\max} MG_k(x) - \int_{x=\min}^{\max} MGM(x) \right| \right)$$

Eqn. (4.9)

The overall relative error would be a good indication per MG (although not comparable across different MGs), as it eliminates the need for units (it is relative) and makes adjusting the thresholds more natural for experts. The normalized form would be:

$$\% \xi = \frac{\sum_{k=1}^n \left(\left| \int_{x=\min}^{\max} MG_k(x) - \int_{x=\min}^{\max} MGM(x) \right| \right)}{n \int_{x=\min}^{\max} MGM(x)}$$

Eqn. (4.10)

This gives an overall indication of the consistency of the input functions by the experts and thus the quality of the risk assessment process as a whole.

The threshold of the error can be set by clinicians from trials, or by simply eliminating the graphs with the highest error.

Another interesting aspect for using the error from the above equations is to use it as an indication for the experts' opinion weight: The more the deviation from the MGM, the less weight this expert should have. These weights can then be used in Equation 4.4 to generate the overall updated MG (or more accurate). This works as a recursive procedure, which should reduce the overall error.

The error for expert k is:

$$\xi_k = \left| \int_{x=\min}^{\max} MG_k(x) - \int_{x=\min}^{\max} MGM(x) \right| \quad \text{Eqn. (4.11)}$$

We will define the weight of an expert's opinion as the complement of the error, divided by the total of all experts' errors (to normalize the weights, and ensure that the total of weights equals 1).

$$w_k = 1 - \frac{\xi_k}{\sum_{j=1}^n \xi_j} \quad \text{Eqn. (4.12)}$$

This equation gives the weight of each expert compared to others, based on their errors. The higher the relative error of an expert compared to the total error from all, the less their weight is.

4.3.4 Case Studies

Case Study 1:

We apply the MGM algorithm on the data in Figure 4.6. Six MG functions obtained from six experts need to be concatenated into a single MGM function representing the set. The MGM of synthetic data is shown in Figure 4.7 using the mean method for consensus. The combined MG is shown as the thicker line along with the six MG

functions from which it was derived. The case study in the next section shows the MGM using polynomial regression.

In this example, we apply the polynomial algorithm on real life data, obtained from the GRiST system, and the graphical online MG Elicitation tool [11]. For simplicity, we will show the MGs supplied by nine different clinicians, where each expert plotted an MG function that best represents the weights of each answer to the questionnaire and its contribution to the overall decision. In the actual elicitation exercise, between 40 and 50 experts provided MG distributions for each question, which shows the importance of an automated process to combine the MGs.

The data from the visual tool had to be prepared first and validated, in what we call the data cleaning stage.

The original graphs were saved as a collection of points joined up by straight lines, which posed a problem for the regression operation. For example, take the hypothetical Expert 1 in Figure 4.6.

The original data supplied by the elicitation tool would have defined four joined lines and be in the form: E1= (0,0) (0.2,0.4) (0.4,0.6) (0.8,0.4) (1,0.6).

This means that each expert graph could have a different number of points depending on the number of lines they used to describe the MG graph.

So the tool had to resample each graph at a set step (and thus equal frequency) to get the same number of sampled points per expert for the same MG.

For E1 in Figure 4.6, the resulting sampled set of points using a step of 0.1 would be: E1= (0,0) (0.1,0.2) (0.2,0.4) (0.3,0.5) (0.4,0.6) (0.5, 0.55) (0.6,0.5) (0.7,0.45) (0.8,0.4) (0.9,0.5) (1.0,0.6).

When this is done for all experts, they will have the same weight in the regression process because all the points from all experts for a certain MG have to be combined into one graph as one set.

If one expert had more points on the graph, this would give the expert exaggerated influence or pull on the MGM. The data pre-processing thus ensures that at each point X there will be n Y values, where n is the number of experts in the trial. The functions are then concatenated by performing regression on the overall set of points. The sampling step is one of the tuning parameters that can be adjusted for more

accurate results. It depends on the shape and the complexity of the original functions but, for our mental-health domain, a step of 0.5 was sufficient.

Figure 4.8 shows the combined MGM using polynomial regression. The R-squared coefficient represents the quality/accuracy of the regression process. It represents the percentage of the data that can be explained by the fitting function.. It is also called the Determination Coefficient [KLEI, 98]. The larger it is, the less the fitting error is present.

This means that the polynomial represents the data better. In our case, $R^2 = 0.5689$ is larger than the linear regression; hence it is a better function. Case study two will illustrate the effect of the order of regression on the results accuracy.

In the above example, the combined MGM will be of the form:

$$\text{MGM} = y = -1.2461x^2 + 1.6348x + 0.0582$$

Eqn. (4.13)

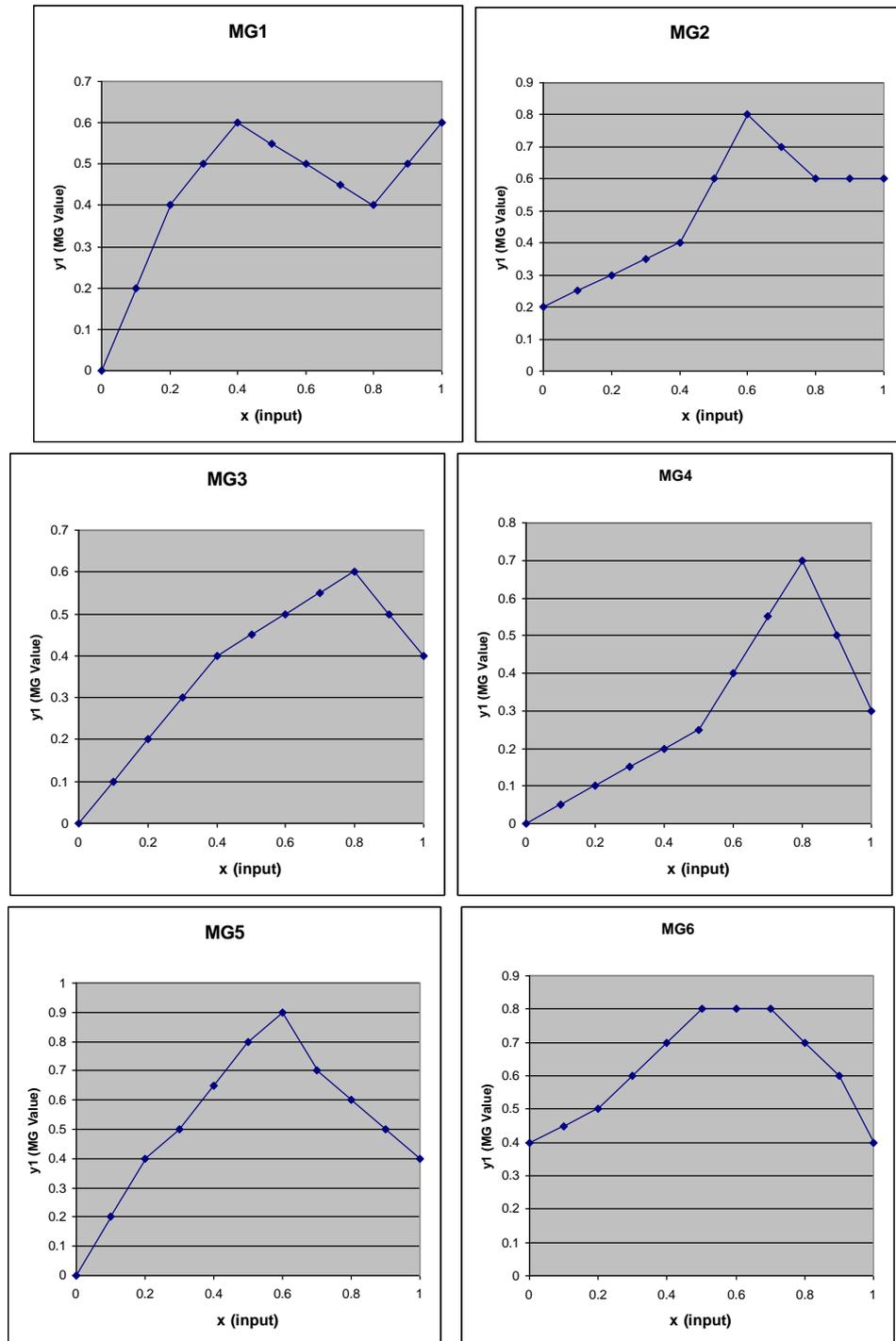


Figure 4.6: MG functions provided by six different experts for the same input to the tree.

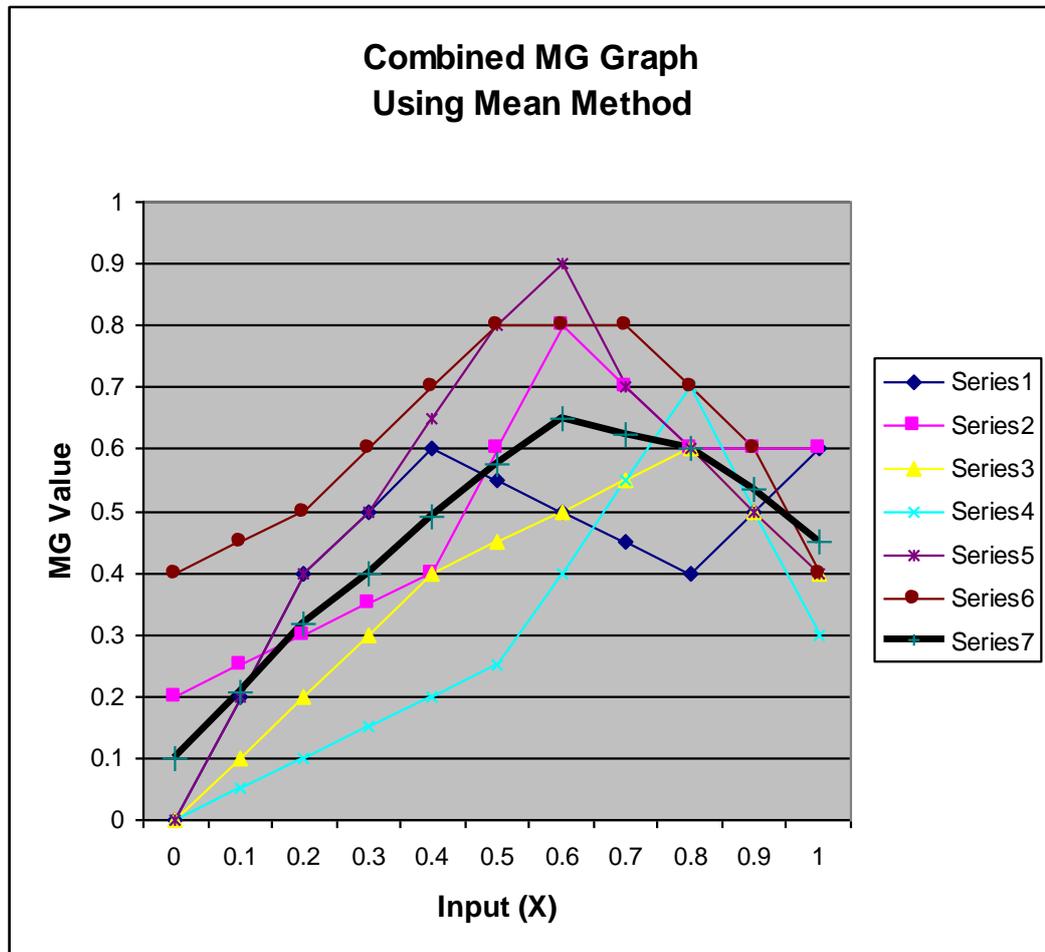


Figure 4.7: The six MGs and Combined MG using the Mean Method.

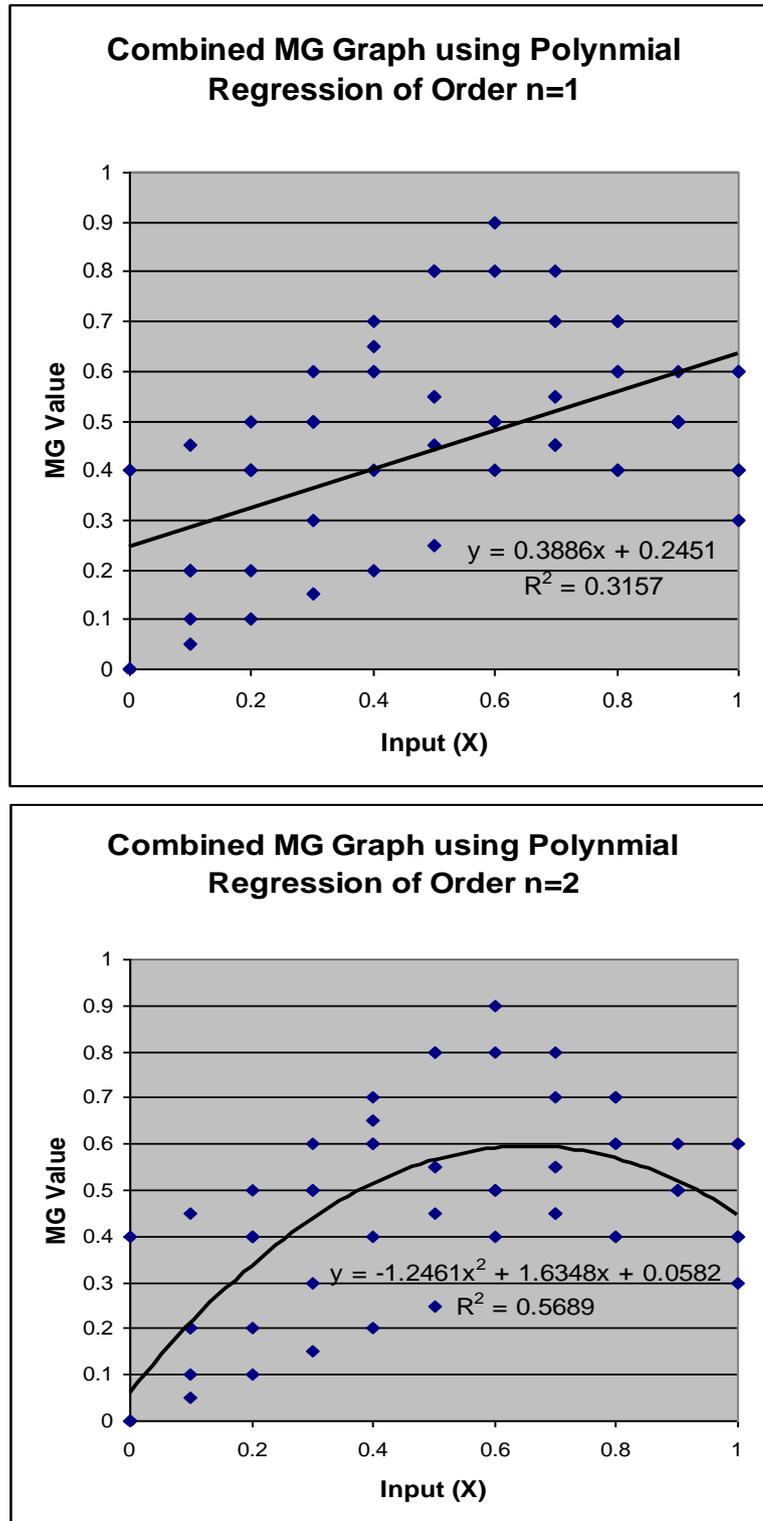


Figure 4.8: The Combined MG using Regression, Order n=1 (top) and n=2 (bottom).

Case Study 2:

In this section, we apply the algorithm on real life data, obtained from the GRiST system, and the graphical online MG Elicitation tool, See Appendix C.

For simplicity, we will show the Membership Grades supplied by nine different clinicians, where each expert plotted an MG function that best represents the weights of each answer to the questionnaire and its contribution to the overall decision.

In the actual elicitation exercise, between 40 and 50 experts provided MG distributions for the question answers. This shows the importance of an automated process to combine the MGs.

Figure 4.9 shows the original MG functions submitted by nine experts for MG2, which represents the effect of the number of suicide attempts so far on the overall risk. As shown in Figure 4.9, some experts agreed on function forms while others disagreed which is expected.

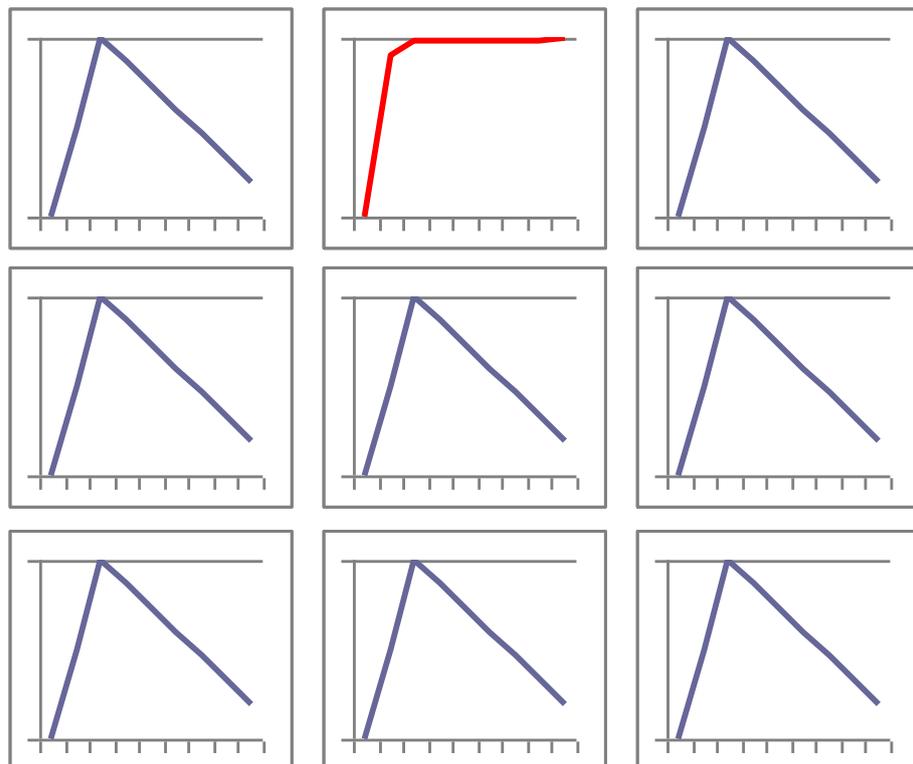


Figure 4.9: The original MGs as provided by the nine experts for MG2.

Figures 4.10 to 4.12 show the output of the MGM (combined membership grade) for MG2, using first, second, third, fourth, fifths and sixth order regression, as well as the

quality of the fit (R^2) and the equation of the MGM. The R-squared coefficient in the figures represents the quality/accuracy of the regression process or fit (in MS Excel). The higher it is, the better the polynomial regression represents the combined data. The improvement in quality of fit is reduced as order increases but the improvement tends to diminish for higher orders (see Figure 4.11 and 4.12, where $R^2 = 0.79$, 0.0795 and 0.807, for regression orders 4,5 and 6 respectively, which is not a large improvement over 0.7761 in respect to complexity). Third order should be sufficient, which, for this example provides a solution for MGM as follows:

$$\text{MGM} = y = 0.0112 * X^3 + 0.7322 * X^2 + 0.7322 * X + 0.0126$$

$$R^2 = 0.7761$$

Eqn. (4.14)

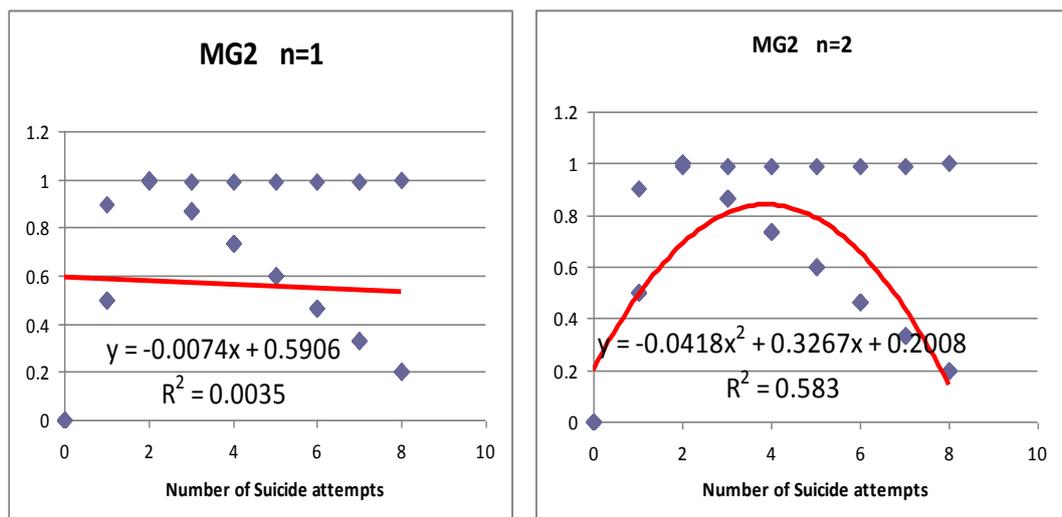


Figure 4.10: The combined MGM for MG2 in regression order 1 and 2.

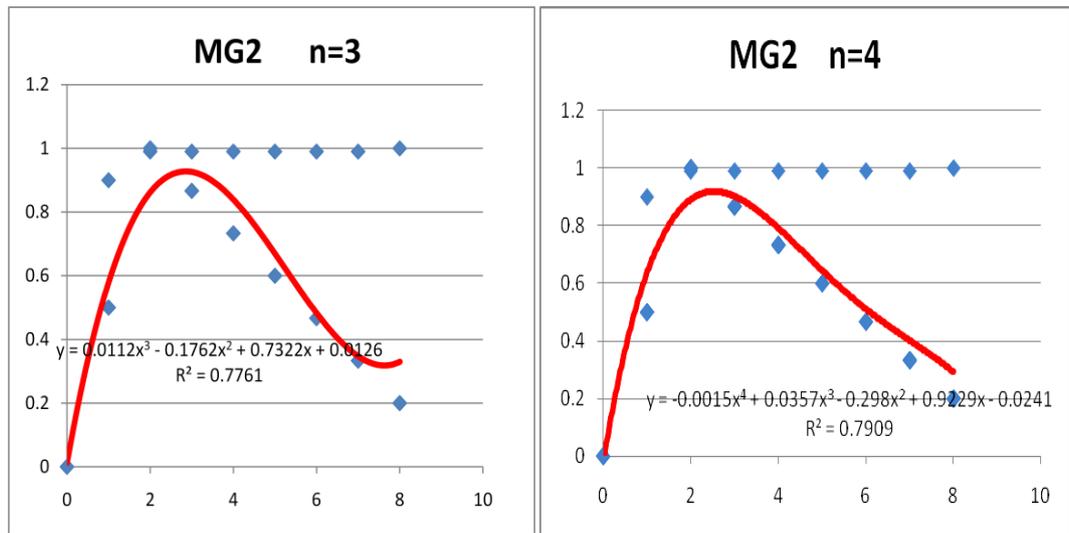


Figure 4.11: The combined MGM for MG2 in order 3 and 4.

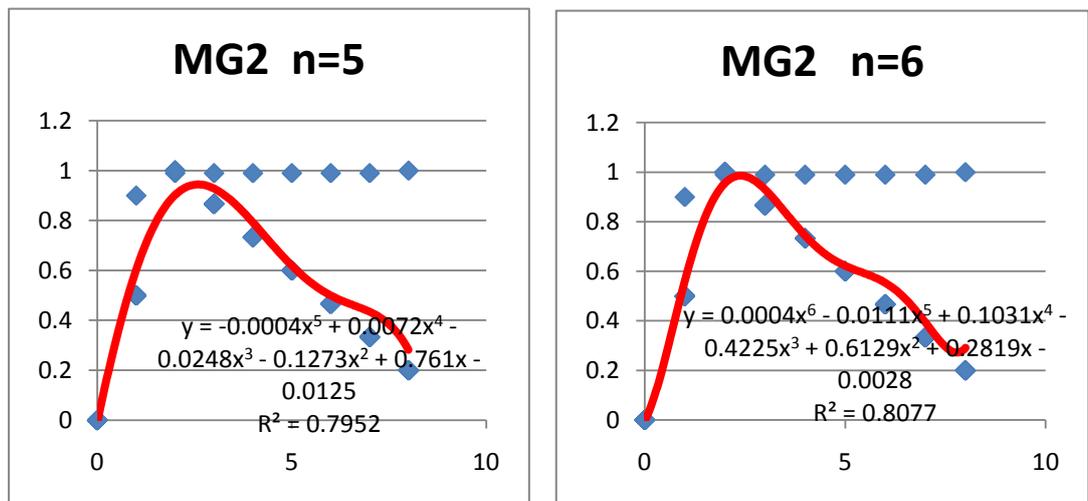
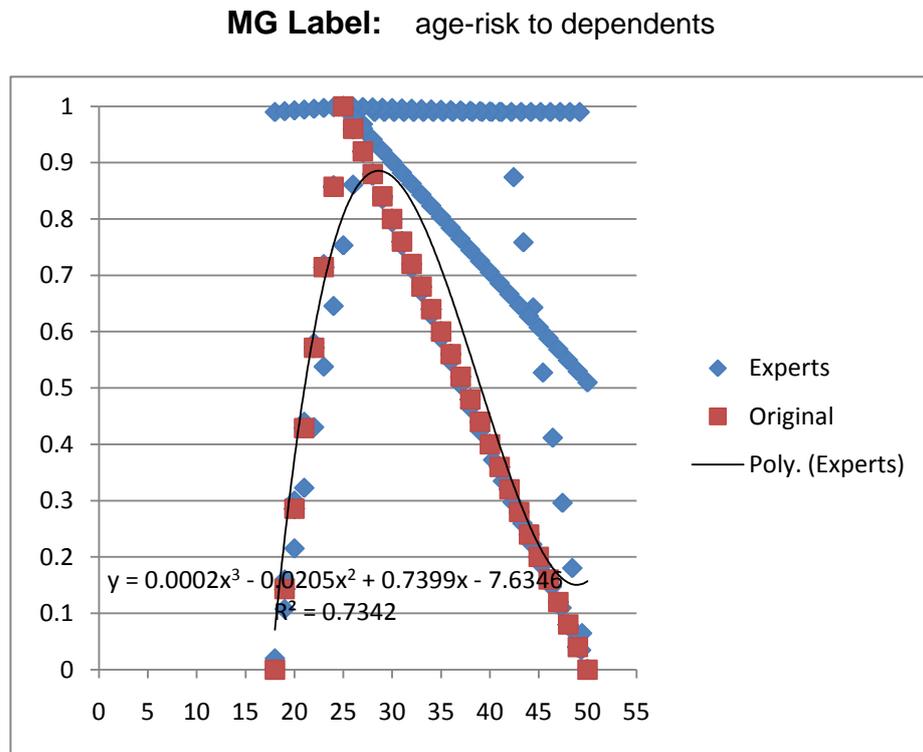


Figure 4.12: The combined MGM for MG2 in order 5 and 6.

See appendix B for the complete MGM analysis results, which includes GRiST data supplied by 43 Experts.

An example is shown in Figure 4.13 below. It also indicates the comments of some of the experts during the elicitation process using the online tool, in the table below. Our tool stores all the data in a database [BUCK, 07], which can then be used to draw the curves in MS Excel.



Start: (18,0.08)

End: (50,0.16)

Max.: (30,0.88)

Comment	Clinician No.
skipped - no experience	12
0.5 at 18, 0.2 at 50	16
not relevant	18
WOULD NOT ANSWER QUESTION AS COULD NOT ISCOLATE OTHER FACTORS	28
WOULD NOT ANSWER	31
Query impact - not done	5

Figure 4.13: An MGM from the actual trials with 43 experts.

4.3.5 Analysis of MGM

It is very important to analyse the results of MGM. The output of the algorithm is a single function that should provide a reasonable representation of the combined effect of all the functions supplied by the experts to represent that particular MG. We also assume that the original MG functions are pretty much similar, which will help the MGM result. This is a fair assumption in a clinical system as we assume some general consensus among clinicians, though the details can vary.

In fact, the actual combined function, MGM, does not represent any of the input functions (i.e., it is a new function, that doesn't agree with any of the original ones, see Figure 3.19 to 3.21). This is because, by definition, it is the best possible compromise or fit, to produce the minimum error between all the expert functions and the new MGM.

Although this is mathematically and statistically acceptable, it could pose semantic problems when it comes to the meaning of the actual data. In GRiST, for example some of the MG functions have specific semantic restrictions on them, relating to the medical data. For example, some of the functions are required to have a minimum of 0, and a maximum at 1.

A combined function using the MGM could lose those localized minima and maxima, due to the nature of regression and curve fitting. As the process smoothes the sharp changes, it is very likely that the new MGM will not have those properties. This begs the question of exploring alternative techniques for combining the MG functions, for example Fuzzy theory analysis.

The fuzzy operations [ZADE, 65] on the MGs will have their limitations too, as they will mostly either flatten the combined MGM function or raise the whole curve, resulting in loss of one or both feature points (i.e., Min. and Max.). In some cases, it may create new Maxima and Minima, which would be confusing for the GRiST tree.

Standard intersection:

$$(A \cap B) = \min [A(x), B(x)] \quad \text{Eqn. (4.15)}$$

Standard union:

$$(A \cup B) = \max [A(x), B(x)] \quad \text{Eqn. (4.16)}$$

This is illustrated graphically in Figure 4.14 (www.doc.ic.ac.uk).

A future avenue to explore would be to investigate the possibility of combining MGM with the fuzzy set operations, in order to better reflect the semantic minima and maxima restrictions specific to the GRiST model.

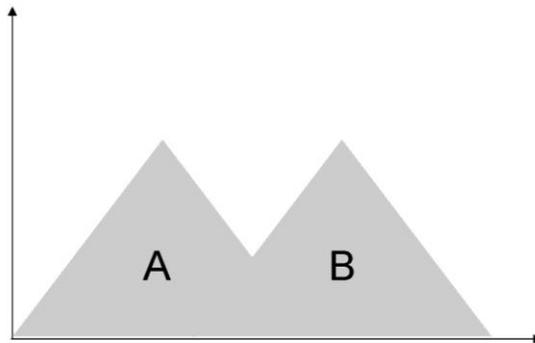


Figure 4.14a: Union in fuzzy sets ($A \cup B$).

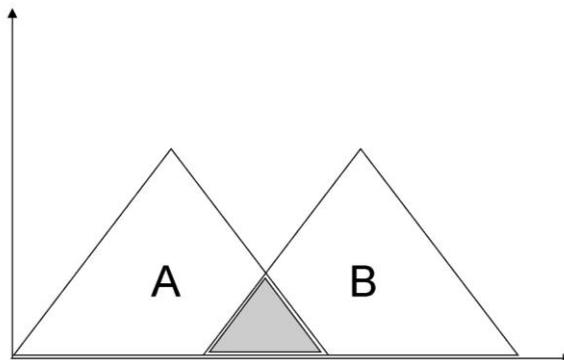


Figure 4.14b: Intersection in Fuzzy Sets.

4.4 Summary

In this chapter we have investigated one of the main problems associated with expert systems; namely discrepancies in expert input. Since all experts differ in their opinions and backgrounds, their input to the systems will inevitably be varied. This is very much true and in fact is a problem in the clinical domain. In case of GRiST, this is evident in the membership grade functions elicitation process. MGs are fuzzy functions used to map inputs (logical or numerical) into meaningful quantifiable values that are used as inputs to the system.

We explored the problem of concatenating experts' opinions as the input filters to the GRiST tree leaf nodes. We call these the membership grade functions.

As the project involved over 40 experts, each leaf node could have potentially over 40 different functions supplied by the different experts. These functions are then used to decode inputs by users into meaningful data that are used to calculate the RIs, and when the system is eventually deployed. The challenge is to combine these functions to produce one MG function that represents the combined information with acceptable accuracy. To do this, we developed the MGM Algorithm.

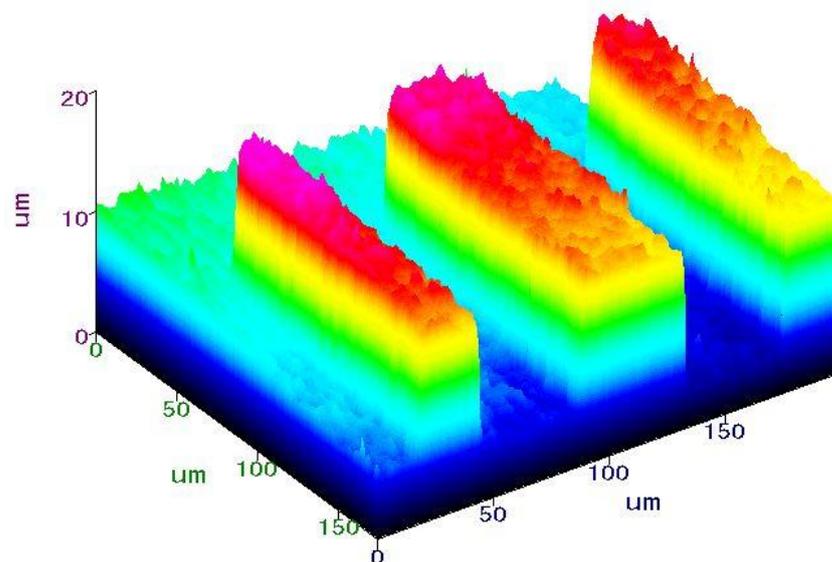
As it is impossible to represent all the functions in the set accurately (because they may differ between experts), we have also presented accuracy and error estimations of the MGM, and showed the acceptable levels of errors.

In the next chapter, we further analyse the iARRIVE method, and tackle the real life problem associated with missing and noisy data. We presented error analysis of the algorithm and a real life case study.



Chapter 5

Pre-processing



This Chapter covers the following:

- Introduction
- Handling Missing Data
- Predicting Output
- Predicting Input
- Error Analysis
- Summary

5.1 Introduction

The problem of missing data is widespread in any application that involves questionnaires or large amounts of information entered by the users. An additional problem is often the accuracy of the input data and how much noise it contains.

In the specific case of GRiST, with an excess of 200 questions, it is inevitable that some of the data will either be missing (i.e. not entered/ answered) or distorted (i.e. inaccurate). In the data available to us from the Mental Health Trusts we work with in the UK, a number of cases do contain missing data.

The model we have developed so far in the ARRIVE and iARRIVE algorithms assumes the data set is complete in order to be able to calculate the values or RIs throughout the tree. It is also vital to calculate the errors in the output due to our prediction.

But in real life data, missing values are inevitable. The location of the missing inputs could vary dramatically, according to the interviewer, his level of knowledge and relevance of the questions to the specific case. Hence we need somehow to deal with the problem of missing data. This needs to be done both at the training stage and the analysis/decision support stage.

This chapter introduces some ways of handling the missing data in GRiST, with some error analysis associated with it. This will give a good indication of the performance of our algorithm iARRIVE, as well as the behaviour of the decision support system in real life situations and accompanying missing data.

5.2 Handling Missing Data

In general, there are several approaches to handling missing data in a system. And as a rule, there will be an error associated with the output due to the missing data.

The main categories those methods fall into are:

1. Substitution:

In some systems, missing data is substituted by a default value, for example, 0. An example would be vending machine software that assumes 1 as default for example. Thus no user input will be substituted by 1.

2. Intelligent Substitution:

In this case, a missing value is always replaced by a certain value, which is not necessarily 0, min or max. This could be a value obtained from analysing the system such as the mean. Note that this value will depend on the data itself, and could therefore vary along the system's life time as an adaptive process. It will also require prior knowledge of the system and learning some of its characteristics. Previous cases need to be known in order to calculate the mean.

3. Prediction:

This is a more intelligent approach, where the system attempts to predict missing data. This is done based on the current condition of the system, e.g. values of other parameters or inputs. For example, for a system on patient information in a hospital, if a person's weight is missing, then based on the person's height the system may be able to estimate or predict the weight.

This approach requires intensive analysis of the system beforehand but has the advantage of producing more justifiable results, as they are based on the system at hand and actual local data.

4. Elimination:

In this case, the system attempts to reconfigure its structure with new parameters to reach a decision without the missing input, thus bypassing the missing data all together. This approach may appear to be appealing but it can be argued that the resulting system will be different to the original one. It is also

difficult to trace the error resulting in this case because the system is effectively restructured. It will be very difficult to compare the new system to the old one. It also requires intensive knowledge of the system.

5. Modulation:

Depending on the structure of the system, it might be possible to reach a partial decision without the missing data. This requires the system to have independent paths that could be separated, and not one overall output. This approach does not solve the missing value problem, but allows the system to produce the part of the decision based on the available inputs.

The table below compares the different approaches with advantages and disadvantages.

Method	Complexity	Accuracy	Advantages
Substitution	Trivial	low	Simple, Fast
Intelligent Substitution	Low	Medium/ low	Fast , Proven
Prediction	Medium	Medium	Flexible
Elimination	High	Medium/High	Emulates human decision making
Modulation	Very High	High	Better in Complex systems

Table 5.1: A comparison of various input predicting techniques.

The substitution approach is trivial and would not be plausible, whereas the elimination and modulation approaches will both result in losing a chunk of the semantic structure of the tree, which would not be acceptable.

In the next sections we will demonstrate the use of two of the above techniques on our system. The most commonly used one is the intelligent substitution, and the more

efficient prediction approach. In both methods, inputs can be statistically and mathematically justified, which makes them more appealing to clinical applications, where decisions require transparency and traceability. They also provide better prediction, despite the more processing that is required. They are also both adaptable and change with the data, and update the values based on new cases.

5.3 Predicting Output

In many cases, it is useful to predict the output, even as an indication, even if some of the inputs are missing. This is particularly useful in the case of risk analysis, as we have known risk thresholds. So if we reach a certain predefined threshold, we can stop the interview process and safely give an indication of the outcome, such as high risk and low risk, without actual values [GIGE,96]. This means that the actual numerical value may not be exact but the range in which it falls will provide sufficient information for decision making. This highly depends on the domain, and it is valid in GRiST [BUCK, 08].

This could also be helpful in speeding up the input process and give early alerts to high risk areas even before completing the analysis. To do this, we introduce the idea of Break Points (BP).

5.3.1 Break Points

An interesting analysis of the GRiST tree would be to predict the risk based on some of the input and not all. That is, if the user doesn't input all the MGs, or answers all the questions. This could be in the case of end users who are not experts or don't find all questions relevant.

Is there a way through the properties of our model to predict the nature of the risk using an incomplete set of inputs? Indeed there is. Consider the tree in Figure 5.1. Assume we only have m_1 and m_2 given. We will try to use them to have an indication of the risk. To do this, we will define a BP as a Break Point. It is the value threshold at a node above which the total M will be high risk (or above the risk threshold supplied by experts). If a node reaches that value, even without knowing the rest of the node values, we can assume the total risk to be high.

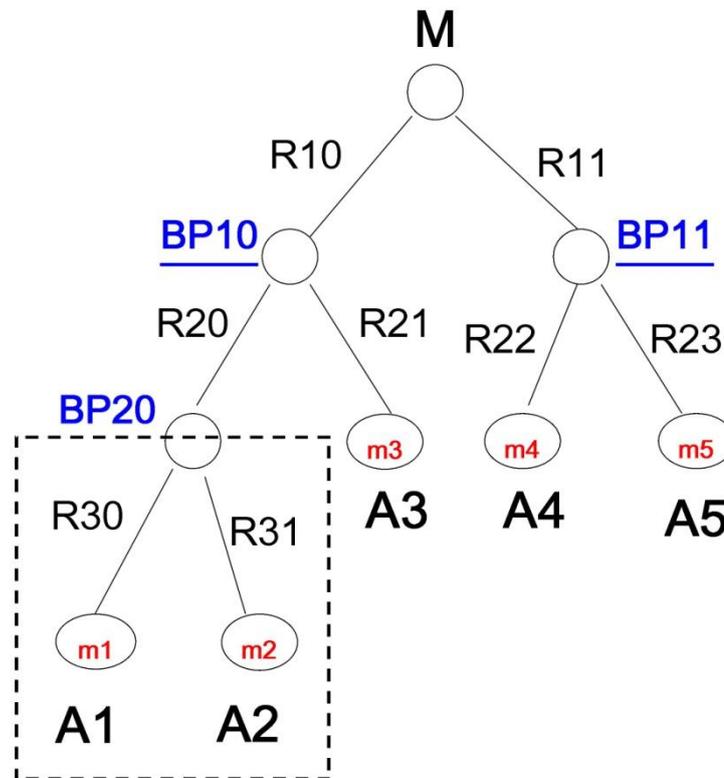


Figure 5.1: The threshold points, BP.

Now we need to find the values of the Break Points in the tree. To do this, we use threshold risk inputs supplied by the experts (m1 to m5). These indicate high risk regions in the data. These could also be calculated from previous data as average values (see later in this section).

A_i are the multipliers as in (3.5).

$$A_1 = R_{10}.R_{20}.R_{30}$$

$$A_2 = R_{10}.R_{20}.R_{31}$$

$$A_3 = R_{10}.R_{21}$$

$$A_4 = R_{11}.R_{22}$$

$$A_5 = R_{11}.R_{23}$$

Eqn. (5.7)

We know from our algorithm that the total Mental Health risk at the top of the tree, in GRiST is of the form (See Chapter Three, iARRIVE algorithm) :

$$\text{Risk} = M = A1.m1 + A2.m2 + A3.m3 + A4.m4 + A5.m5 \quad \text{Eqn. (5.8)}$$

Now, for a sub tree of A1 and A2, assume the total MG would be m12,

$$m12 = R30.m1 + R31.m2 \quad \text{Eqn. (5.9)}$$

Now to find the threshold of m12 that makes the total risk higher than certain threshold M, assuming the rest of the values are average (m3 to m5), we can rewrite the above equation as (see Chapter 3, iARRIVE algorithm):

$$M = m12.R22.R10 + A3.m3 + A4.M4 + A5.m5 \quad \text{Eqn. (5.10)}$$

Let:

$$A20 = R22.R10 \quad \text{Eqn. (5.11)}$$

To find the minimum m12 (call that BP20) that satisfies Eqn. (5.10):

$$BP20 = \frac{M - A3.m3 + A4.M4 + A5.m5}{A20} \quad \text{Eqn. (5.12)}$$

That is the threshold value above which the risk would most probably be above the high threshold M, set by the experts. So if during a questionnaire, after answering m1 and m2, m12 is larger than BP20, then there is a high probability that the total risk M would be high even without knowing the answers to the rest of the questions.

In general, to calculate the break points of any node in the tree, we can generalize the above equation:

$$BP_{xy} = \frac{M - \sum A_i.m_i}{A_{xy}} \quad \text{Eqn. (5.13)}$$

where: $m_i \notin BP_{xy}$ subtree.
and

$$A_{xy} = \prod RI_{hj} \quad \text{Eqn. (5.14)}$$

As a generalization, M would be BP00. (Total Risk threshold Break Point), and is supplied by the experts.

5.4 Predicting Inputs

We have analyzed above the situation where one or more of the input values are missing, and how we were able to predict cut off points (BP) which would give a rough indication of the total risk (not the actual value). But it would be useful to find a method of predicting the value of the total risk, even though some inputs are missing, that is applying the GRiST classification algorithm but with missing values that are supplied via prediction. Predicting the error in that calculation would be greatly beneficial too.

To do this, we need a method to predict the values of the missing data, based on analysis and association from previous cases. Several methods have been developed to do this, starting with the simple mean method, on towards AI methods. In the following sections we introduce the application of a couple of these methods to our algorithm.

5.4.1 Using the Mean

This is classed as intelligent substitution, as described in Section 5.2. The missing values m_i are substituted with their statistical average, m^i . To calculate m^i , we use previously known cases, or expert opinions. We also calculate the standard deviation of each m^i . Let that be σ_i .

The total M for a GRiST tree where m_i are known and m_i' are missing would be:

$$M = \sum A_i . m_i + \sum_i A_i . m^i + \xi \quad \text{Eqn. (5.15)}$$

where t is an index spanning the known values at input level (leaf nodes), and i is the index of leaf nodes with missing values with the mean value substitute in them.

where: ξ is the error due to using m^i .

Let $M^$ be the calculated risk with the known values, thus:

$$M^ = \sum A_t . m_t \quad \text{Eqn. (5.16)}$$

So:

$$M = M^ + \sum_i A_i . m^i + \xi \quad \text{Eqn. (5.17)}$$

The maximum error we should get on average, would be σ_i , thus the maximum error in M would be:

$$\xi = \pm \sum_i A_i . \sigma_i \quad \text{Eqn. (5.18)}$$

Hence the Maximum Percentage Error (MPE) would be:

$$\% \xi = \frac{\pm \sum_i A_i . \sigma_i}{M} \times 100 \quad \text{Eqn. (5.19)}$$

So, we can actually calculate the percentage error based on the missing data.

We can now take this a step further and calculate the error associated with each input. That is, the error in M if a certain input was missing and substituted by its average, m^i .

This would be very useful in determining the important inputs or questions. The higher the error associated with missing a question's data, the more important that question is to the decision process. Thus we can analyze the structure of the tree, and order the questions by their importance. This would give an insight into the nature of the tree through the data. Questions can be given weights and priorities, thus guiding the clinicians' clues to the important ones, which should not be missed in analysis. It will also help us study in depth the psychological model underpinning GRiST.

To determine the errors associated with each input or question, (Figure 5.2), we use the above error equation, assuming only one question was left out, for that specific input. Hence from Figure 5.2, we get:

$$\% \xi_1 = \frac{\pm A_1 \cdot \sigma_1}{M} \times 100$$

$$\% \xi_2 = \frac{\pm A_2 \cdot \sigma_2}{M} \times 100$$

$$\% \xi_3 = \frac{\pm A_3 \cdot \sigma_3}{M} \times 100$$

$$\% \xi_4 = \frac{\pm A_4 \cdot \sigma_4}{M} \times 100$$

$$\% \xi_5 = \frac{\pm A_5 \cdot \sigma_5}{M} \times 100$$

Eqn. (5.20)

We can also calculate the complement, which would be the accuracy of the algorithm in calculating M given missing inputs, which would be:

$$\%M = 100 - \% \xi$$

Eqn. (5.21)

This would indicate the accuracy of the results.

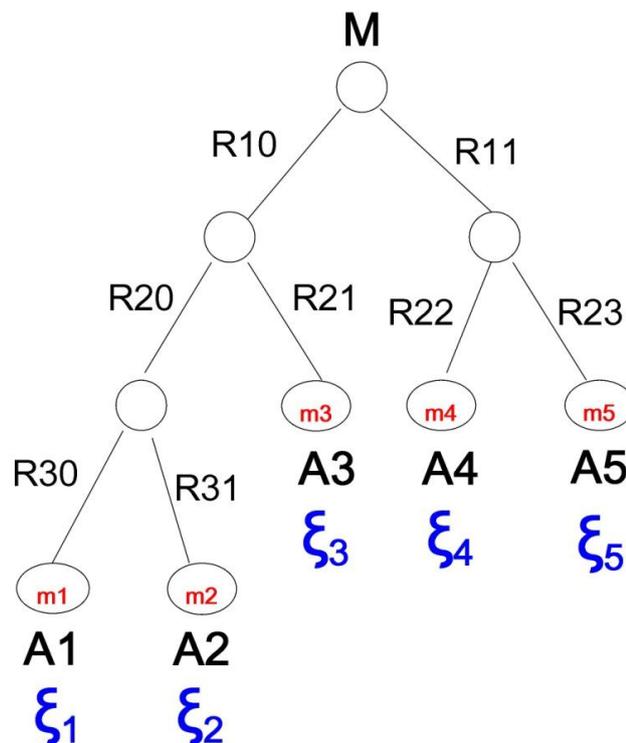


Figure 5.2: Errors associated with missing inputs.

5.4.2 Using Correlation

In statistics, correlation (often measured as a correlation coefficient, ρ) indicates the strength and direction of a *linear* relationship between two random variables [EDWA, 76]. That is in contrast with the usage of the term in colloquial speech, which denotes any relationship, not necessarily linear. In general statistical usage, *correlation* or *co-relation* refers to the departure of two random variables from independence. In this broad sense there are several coefficients, measuring the degree of correlation, adapted to the nature of the data [BRUC, 72].

In our case it would be interesting to see if any of the inputs to the tree are related or correlated in any way.

In the previous Figure 5.2, for example, if we know that inputs m_2 and m_4 are always equal; then if m_2 is missing, we can always substitute $m_4=m_2$ and avoid any missing data. This will not only give us an accurate result in the prediction, but could also save time, as we can then avoid asking one of the questions (inputs) all together. This will reduce redundancy and make our process more efficient. It will also give us a better insight into the GRiST tree and the various dependencies in the data and structure. This is similar to principal component analysis [EDWA,76], but whereas in our case one can work on every input, principal component analysis only works on the top correlated sets (sets with highest correlations).

Correlation is one measure we can use to do this. It gives an indication of the relationship between each two inputs of the GRiST questionnaire. Thus we can see which inputs are related, or correlated, or behave in a similar way. This information could then be used to substitute the missing input or remove that question from the input all together as it may be redundant in the first place.

In case of a small number of inputs, e.g. two, this could be visually checked by plotting them and checking if they follow a trend. Figure 5.3 shows several sets of (x, y) points, with the correlation coefficient of x and y for each set. Note that the correlation reflects the noisiness and direction of a linear relationship (top row), but not the slope of that relationship (middle), nor many aspects of nonlinear relationships (bottom). N.B.: the figure in the centre has a slope of 0 but in that case the correlation coefficient is undefined because the variance of Y is zero.

A correlation coefficient is obtained by dividing the covariance of the two variables by the product of their standard deviations [EDWA, 76].

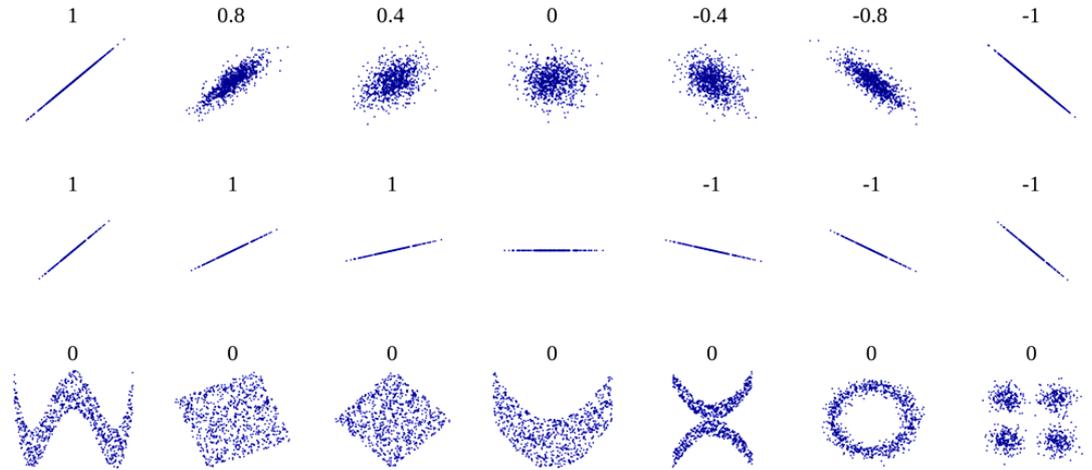


Figure 5.3: Different Correlation Coefficient values and their meaning [WIKI, 11]

The correlation coefficient $\rho_{X, Y}$ between two random variables X and Y with expected values (mean) μ_X and μ_Y and standard deviations σ_X and σ_Y is defined as:

$$\rho_{X,Y} = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y} = \frac{E((X - \mu_X)(Y - \mu_Y))}{\sigma_X \sigma_Y},$$

Eqn. (5.22)

Or:

Since $\mu_X = E(X)$, $\sigma_X^2 = E[(X - E(X))^2] = E(X^2) - E^2(X)$ and likewise for Y , and since $E[(X - E(X))(Y - E(Y))] = E(XY) - E(X)E(Y)$, we may also write:

$$\rho_{X,Y} = \frac{E(XY) - E(X)E(Y)}{\sqrt{E(X^2) - E^2(X)} \sqrt{E(Y^2) - E^2(Y)}}.$$

Eqn. (5.23)

Using correlation in estimating the missing values could be important as we show later, as variables with strong correlation can be used to substitute each other's values.

5.4.2.1 The Coefficient of Determination

Another important measure for dependency between two variables is the coefficient of determination [EDWA, 76].

The coefficient of determination is the ratio of the explained variation to the total variation [SHEL, 72]. It has a value between 0 and 1. It represents how strong two variables are linearly associated. That is, 0.9 means that 90% of the data is close to the line of best fit, or the regression line represents the data in 90% of the cases reasonably.

5.4.2.2 The Correlation Matrix

In the case of higher dimensions (as in GRiST, with almost 200 inputs), the number of possible combinations is huge (the factorial of 198), hence we need the correlation coefficient matrix as a formal method to indicate dependencies and trends within the inputs.

To deduce this, we first need to look at the covariance and covariance matrix.

The covariance between two random variables X and Y , with means $E(X) = \mu$ and $E(Y) = \nu$ is defined as [PLAC, 60] :

$$\text{Cov}(X, Y) = E((X - \mu)(Y - \nu)), \quad \text{Eqn. (5.24)}$$

where E is the Mean operator. This can also be written as:

$$\text{Cov}(X, Y) = E((X - \mu)Y) = E(X(Y - \nu)) \quad \text{Eqn. (5.25)}$$

Or alternatively

$$\text{Cov}(X, Y) = E(X \cdot Y) - \mu E(Y) - \nu E(X) + \mu\nu, \quad \text{Eqn. (5.26)}$$

Random variables whose covariance is zero are called uncorrelated.

If entries in the column vector:

$$\mathbf{X} = \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix} \quad \text{Eqn. (5.27)}$$

X are random variables (in our case, GRiST inputs at leaf nodes), each with a finite variance, then the covariance matrix Σ is the matrix whose (i, j) entry is the covariance:

$$\Sigma_{ij} = \text{cov}(X_i, X_j) = E[(X_i - \mu_i)(X_j - \mu_j)] \quad \text{Eqn. (5.28)}$$

Where:

$$\mu_i = E(X_i) \quad \text{Eqn. (5.29)}$$

is the expected value (mean) of the i th entry in the vector X . In other words, we have [BRUC, 72]:

$$\Sigma = \begin{bmatrix} E[(X_1 - \mu_1)(X_1 - \mu_1)] & E[(X_1 - \mu_1)(X_2 - \mu_2)] & \cdots & E[(X_1 - \mu_1)(X_n - \mu_n)] \\ E[(X_2 - \mu_2)(X_1 - \mu_1)] & E[(X_2 - \mu_2)(X_2 - \mu_2)] & \cdots & E[(X_2 - \mu_2)(X_n - \mu_n)] \\ \vdots & \vdots & \ddots & \vdots \\ E[(X_n - \mu_n)(X_1 - \mu_1)] & E[(X_n - \mu_n)(X_2 - \mu_2)] & \cdots & E[(X_n - \mu_n)(X_n - \mu_n)] \end{bmatrix} \quad \text{Eqn. (5.30)}$$

The covariance matrix alongside Eigen Vectors are the basis of Principle Component Analysis, which is used widely in pattern recognition techniques and trend analysis [EDWA, 74]. When n is very large, the matrix is very useful in storing the data and provides an easy access technique. It is built in many commercial software packages (e.g. using Matlab).

Hence the correlation matrix, from the definition of correlation, would be:

$$\rho = \begin{bmatrix} \frac{\text{Cov}(X_1, X_1)}{\sigma_{x1}\sigma_{x1}} & \frac{\text{Cov}(X_1, X_2)}{\sigma_{x1}\sigma_{x2}} & \cdots & \frac{\text{Cov}(X_1, X_n)}{\sigma_{x1}\sigma_{xn}} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\text{Cov}(X_n, X_1)}{\sigma_{xn}\sigma_{x1}} & \frac{\text{Cov}(X_n, X_2)}{\sigma_{xn}\sigma_{x2}} & \cdots & \frac{\text{Cov}(X_n, X_n)}{\sigma_{xn}\sigma_{xn}} \end{bmatrix} \quad \text{Eqn. (5.31)}$$

5.4.2.3 The Correlation Matrix for GRiST

Analysis of the Correlation matrix of the inputs to GRiST will give us an indication of the hidden relationships between the various variables (inputs) and thus redundancies and tendencies within the data. This will enable us to substitute missing data or even refine the tree structure based on the findings. Redundant questions can be pruned and inputs can then be predicted.

Variables with high positive values for correlation are the ones of interest, as this indicates a linear relationship in the same direction (i.e. increase together and decrease together). The Correlation Coefficient always has an absolute value of less than one.

To obtain the correlation matrix for a sample GRiST tree, we use the example shown in Figure 5.4.

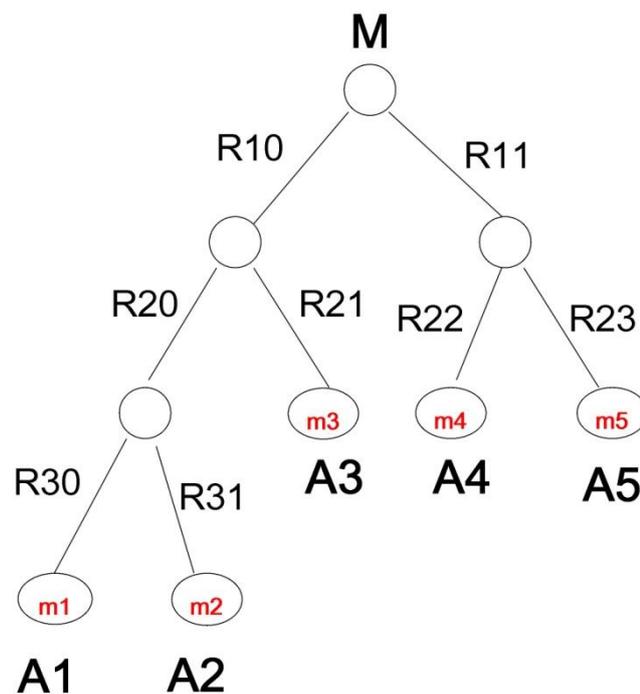


Figure 5.4: A sample GRiST tree with MG inputs A1 to A5.

A1 to A5 represent matrices of various test cases (i.e. various patients) to the knowledge tree. We now need to find the Correlation matrix for that tree. To do this, we need to use equation 5.31. We need to calculate the correlation between each two inputs, i.e.: $\text{Corr}(m1,m2)$, $\text{Corr}(m1,m3)$, ,...etc.

This will give us:

$$\rho = \begin{bmatrix} \rho(m1, m1) & \rho(m1, m2) & \rho(m1, m3) & \rho(m1, m4) & \rho(m1, m5) \\ \rho(m2, m1) & \rho(m2, m2) & \rho(m2, m3) & \rho(m2, m4) & \rho(m2, m5) \\ \rho(m3, m1) & \rho(m3, m2) & \rho(m3, m3) & \rho(m3, m4) & \rho(m3, m5) \\ \rho(m4, m1) & \rho(m4, m2) & \rho(m4, m3) & \rho(m4, m4) & \rho(m4, m5) \\ \rho(m5, m1) & \rho(m5, m2) & \rho(m5, m3) & \rho(m5, m4) & \rho(m5, m5) \end{bmatrix} \quad \text{Eqn. (5.32)}$$

Where:

$$\rho(mx, my) = \frac{Cov(mx, my)}{\sigma_{mx} \sigma_{my}} = \frac{E(mx * my) - E(mx) * E(my)}{\sqrt{E(mx^2) - E^2(mx)} \sqrt{E(my^2) - E^2(my)}} \quad \text{Eqn. (5.33)}$$

5.4.2.4 An Example

We will now demonstrate the above with a practical example, using a very simple sample tree. Consider the inputs of the tree in Figure 5.5 below, with five different test cases, to the three inputs, m1, m2, m3. The input cases matrices are A1, A2, A3 respectively.

Where:

$$A1 = \begin{bmatrix} 0.3 \\ 0.6 \\ 0.5 \\ 0.4 \\ 0.7 \end{bmatrix}, \quad A2 = \begin{bmatrix} 0.4 \\ 0.7 \\ 0.6 \\ 0.5 \\ 0.8 \end{bmatrix}, \quad A3 = \begin{bmatrix} 0.2 \\ 0.4 \\ 0.3 \\ 0.1 \\ 0.5 \end{bmatrix} \quad \text{Eqn. (5.34)}$$

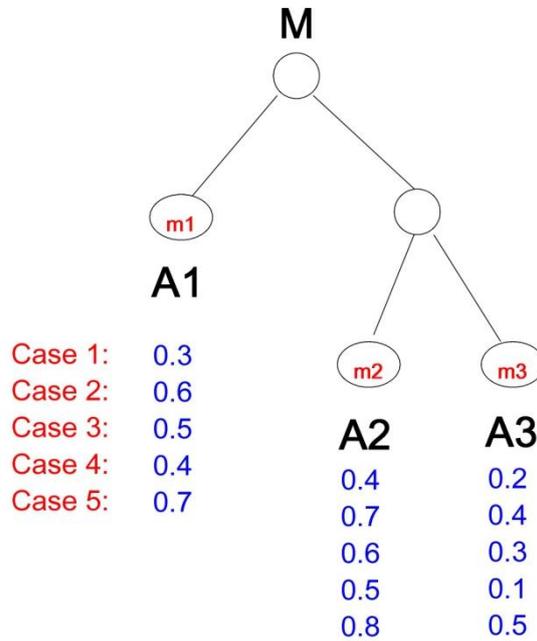


Figure 5.5: A sample tree with three inputs (m1,m2,m3) and five test cases.

The Correlation matrix will be on the form:

$$\rho = \begin{bmatrix} \rho(m1, m1) & \rho(m1, m2) & \rho(m1, m3) \\ \rho(m2, m1) & \rho(m2, m2) & \rho(m2, m3) \\ \rho(m3, m1) & \rho(m3, m2) & \rho(m3, m3) \end{bmatrix}$$

Eqn. (5.35)

Now:

$$\rho(mx, my) = \frac{Cov(mx, my)}{\sigma_{mx} \sigma_{my}} = \frac{E(mx * my) - E(mx) * E(my)}{\sqrt{E(mx^2) - E^2(mx)} \sqrt{E(my^2) - E^2(my)}}$$

Eqn. (5.36)

We need to find, E(m1), E(m2), E(m3), and all the various combinations. The results using MS Excel are shown in Table 5.2.

An example is shown below:

$$\begin{aligned} \rho(m1, m2) &= \frac{E(A1 * A2) - E(A1) * E(A2)}{\sqrt{E(A1^2) - E^2(A1)} \sqrt{E(A2^2) - E^2(A2)}} \\ &= \frac{0.32 - 0.5 * 0.6}{\sqrt{0.27 - 0.25} \sqrt{0.38 - 0.36}} = \frac{0.02}{0.02} = 1 \end{aligned}$$

Eqn. (5.37)

This is an expected result, as we deliberately choose the data to be correlated to show the results graphically.

Note that:

$$\rho(x, x) = 1 \quad \text{Eqn. (5.38)}$$

By calculating the rest of the entries, the Correlation matrix for this example would be:

$$\rho = \begin{bmatrix} 1 & 1 & 0.9 \\ 1 & 1 & 0.9 \\ 0.9 & 0.9 & 1 \end{bmatrix} \quad \text{Eqn. (5.39)}$$

Set	A1	A2	A3	A1*A2	A1*A3	A2*A3
Case 1:	0.3	0.4	0.2	0.12	0.06	0.08
Case 2:	0.6	0.7	0.4	0.42	0.24	0.28
Case 3:	0.5	0.6	0.3	0.3	0.15	0.18
Case 4:	0.4	0.5	0.1	0.2	0.04	0.05
Case 5:	0.7	0.8	0.5	0.56	0.35	0.4
E():	0.5	0.6	0.3	0.32	0.168	0.198
Var():	0.025	0.025	0.025	0.0306	0.01667	0.02092
StD():	0.158114	0.158114	0.158114	0.174929	0.129112	0.144637
Corr():				1	0.9	0.9
Det():				1	0.81	0.81

Table 5.2: Results of calculating the various coefficients.

Table 5.2 shows some basic calculations and the relationship between the different coefficients for the example shown above. Most Mathematical software packages provide these functions; the table was constructed using MS Excel.

5.4.3 An Algorithm for Predicting Inputs by Correlation (APIC)

This was a simple straight forward example, where a clear correlation is present between A1 and A2. This means that A1 follows A2 in trend, but not in value. So A1 and A2 are dependent. $A1 \leftrightarrow A2$.

This means there is a relationship between the values of A1 and A2. In other words, a formula should exist describing A1 in terms of A2 or vice versa.

The above matrix also shows a certain relationship between A3 and both A1 and A2, although it is not as strong. This is clearer from the value of the Coefficient of determination, $R^2(A2, A3) = 0.81$

How do we use Correlation and Determination coefficients to predict missing inputs? The following section explains an algorithm for doing this.

5.4.3.1 Induction

Let's assume the missing value in one of the data sets is A1. The first step would be to find the variable (input) with the largest Correlation to A1. In the above Correlation matrix, we simply look at the first row (A1), and find the maximum value. We need to ignore the diagonal values, as they are always equal to 1 (Correlation of a variable to itself).

The maximum value should be larger than a predefined threshold to assume enough correlation to justify dependency. This is to be found based on the data and domain. Typically, values above 0.5 are considered acceptable in statistical applications [DRAP, 98].

In the above example, the second value (representing the second column is the largest. Hence A1 is more strongly correlated to A2 than to any other variable in the inputs (in this example we only have three variables, but this could be n).

Now, we know that a strong relationship exists between A1 and A2, we need to find a formula that would describe this relationship. If we do, then using any new input to the system, A2, we can predict the value of A1.

We will look at this problem from a different perspective. Looking at the graphs representing each pair of the inputs, we can clearly see which ones have a trend. m1 and m2 clearly follow a line, which is consistent with our Correlation matrix findings.

We thus can deduce the formula using curve fitting or more general, regression [CHAT, 77].

The order of the fitting or the type of the fitting function is pretty much dependent on the data; this could be linear, or non-linear. This requires knowledge of the domain and testing using different fitting functions to reach the ones that yield minimum error.

To determine the best order to use, we need to run the regression algorithms several times using various orders and calculating the resulting interpolation error in each case. This will give us the function best suited to that specific set of data or domain [DRAP, 98]. The stopping criteria will depend on the domain and statistical nature of the data. This is illustrated later in the chapter.

We demonstrate the algorithm using Linear Regression (or Polynomial Interpolation for the general case). In this case, from the graph, it is obvious that the relationship between m1 and m2 is a linear one.

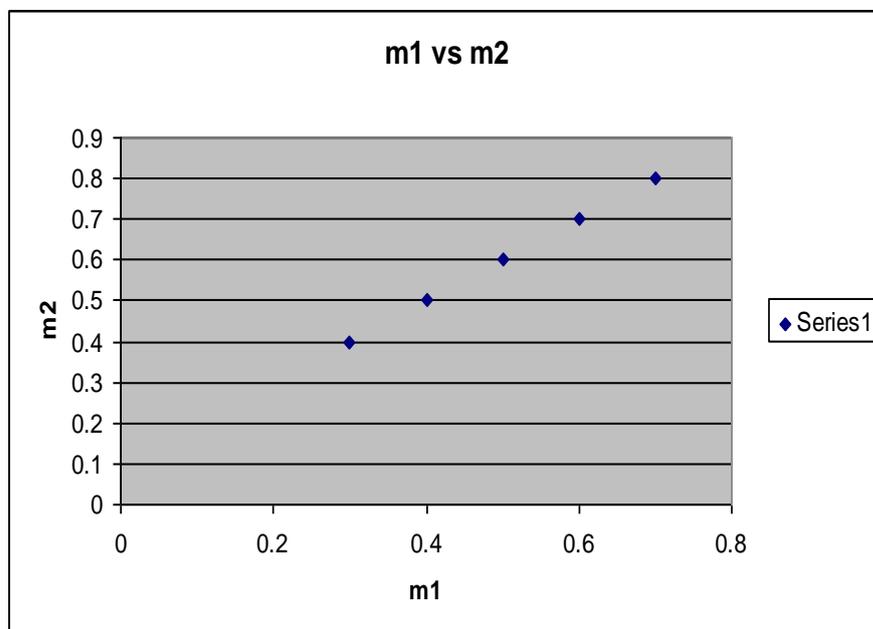


Figure 5.6: m1 versus m2

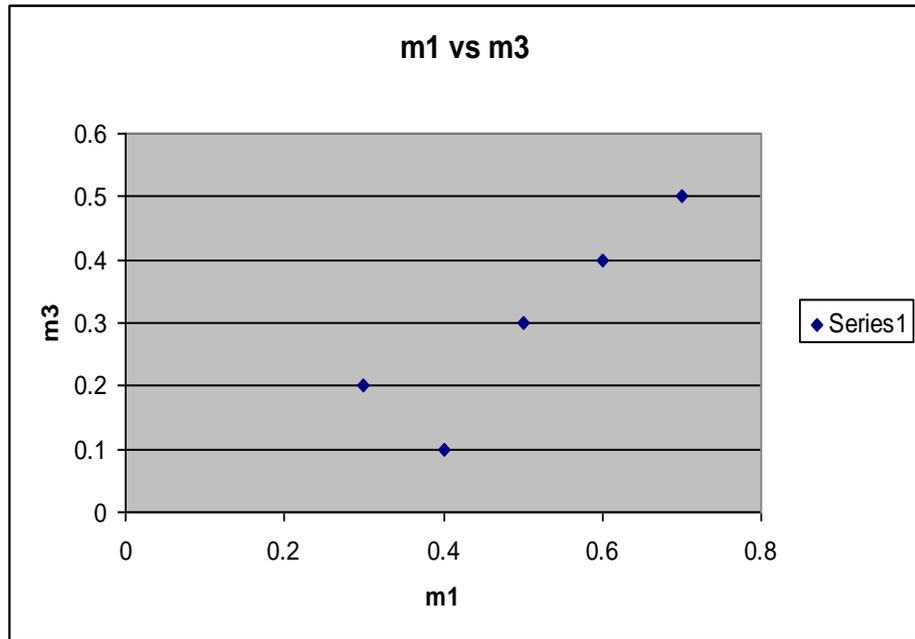


Figure 5.7: m1 versus m3

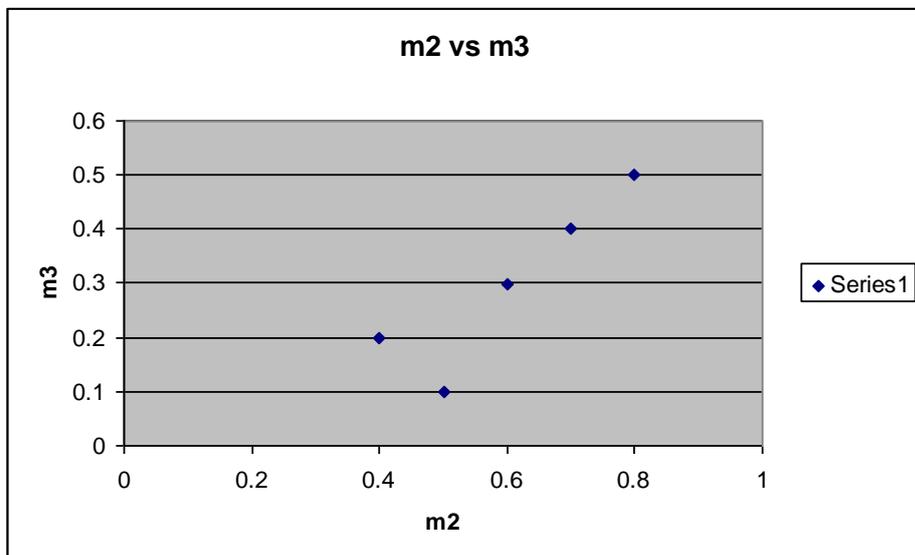


Figure 5.8: m2 versus m3

Using simple fitting algorithms (or complicated ones for higher order), we can calculate the formula coefficients.

Matlab [MATLAB] and other mathematical software provide efficient functions to generate the parameters of the fitted curves using higher order Polynomial regression.

It also provides functions to assess the quality of the fitted curves and parameter estimation errors.

In our example:

Assume the polynomial is of the form:

$$m2 = a + b.m1$$

Eqn. (5.40)

Thus:

By applying polynomial fitting, we get:

$$a = 0.1$$

$$b = 1$$

Eqn. (5.41)

Thus the relationship between $m1$ and $m2$ can be described as:

$$m2 = 0.1 + m1$$

Eqn. (5.42)

Hence, if we have an input data set, with a missing $m2$, we can always calculate or predict the value of $m2$ using the above equation.

The error resulting from the approximation of APIC can then be calculated as in the Mean method we presented in previous sections. This is revisited in the next section.

A flowchart of the algorithm is shown in Figure 5.9.

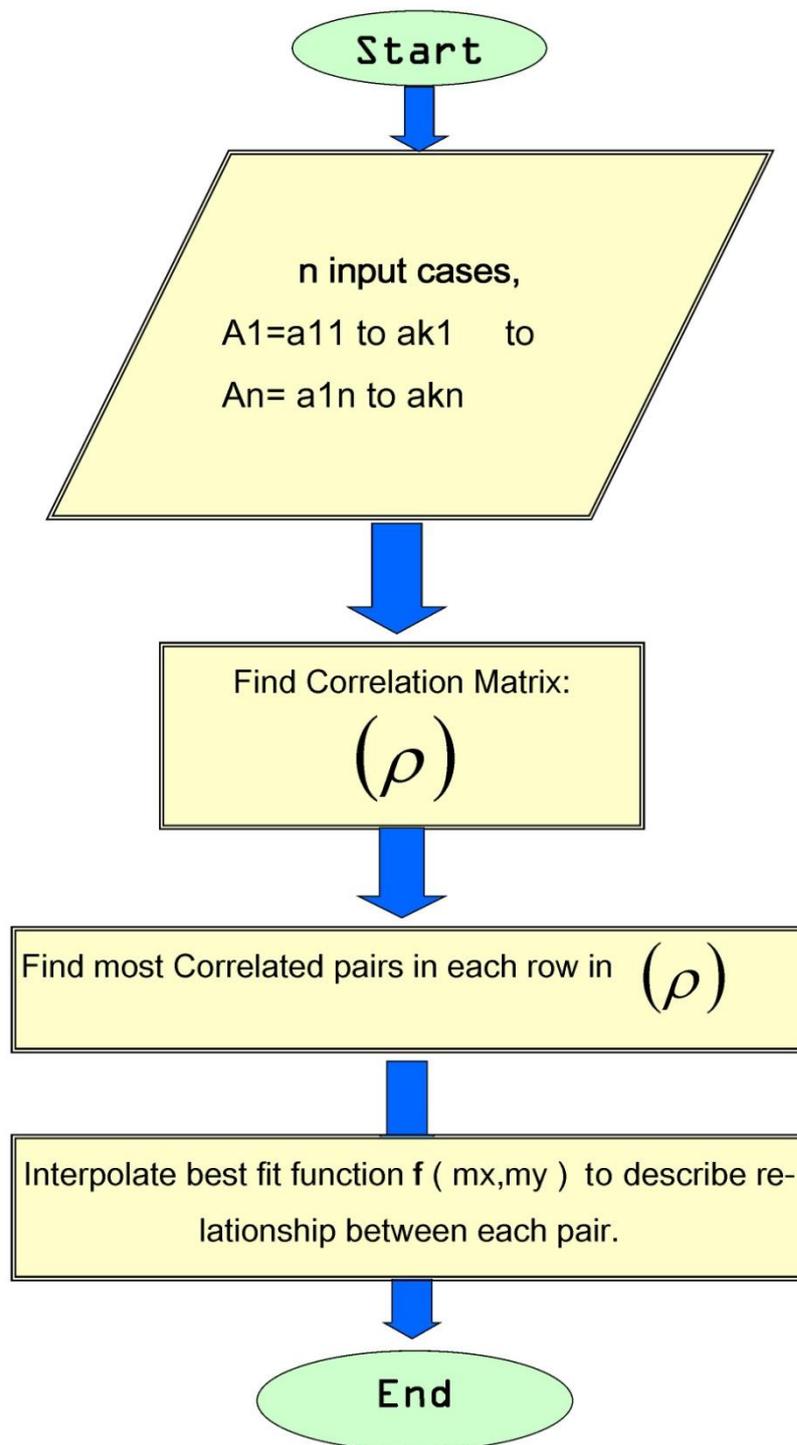


Figure 5.9: APIC Algorithm flowchart.

We will assume a linear relationship between inputs for simplicity. Analysis of different applications and different types of data/domains will give a better understanding of the nature of the required interpolation.

The same algorithm can be applied with higher order regressions, which gives the flexibility to use this method depending on the nature of the data and the model in hand. The complexity of the algorithm will be of order n^2 as the calculation of the correlation matrix will need to go through all the pair combinations of input.

5.4.3.2 Estimation Errors in APIC

The algorithm above presents an acceptable solution to the problem of missing data at the leaf node level (inputs). The output of the algorithm will be a viable substitution of the missing input with a justifiable prediction.

It is important however, to present the user with an indication about how the obtained output might differ from what would have been obtained had there been no missing data in the first place. To do this, we calculate the average error resulting from substituting the missing input with a predicted one.

For a specific input of the tree (leaf node) MG, assuming M' is a vector representing the different outputs obtained using prediction (i.e. removing the actual value and replacing it with a predicted one), and M is the actual output vector using the original value, then the percentage error in output would be:

$$\xi = \frac{\sum_{i=1}^n |M'_{i} - M_{i}|}{\sum_{i=1}^n M_{i}} \times 100$$

Eqn. (5.43)

Where $i=1$ to n , n represents the number of available training test cases.

The above formula should be calculated for all inputs, so that we have an error estimator for each missing output based on prediction.

The result of APIC would be two vectors: a Formula Vector (FV), associating a formula with each missing input and an Error Vector (EV) associating an error estimate with each predicted input.

5.4.3.3 Case study

In this section, we demonstrate the algorithm with a more realistic numerical example, and show the effect of using different regression orders on the accuracy of the result. We use ten different cases as an input.

The R-Squared coefficient shown in Table 5.3 is an indicator of the quality of the Regression process, so the higher the value of R-Squared, the better the polynomial would describe the given data set. This will lead in turn to a better prediction, when the function obtained is used to predict the missing outputs.

The first step is to check the Correlation of the input sets, and determine the best related two inputs. In this case, it is A1 and A2 with maximum correlation of $\text{Corr}(A1,A2)= 0.715626$. See Table 5.3.

The table also contains some other statistics like the Standard deviation and Mean.

This is confirmed by examining graphs 5.10 to 5.13 (the line represents the linear trend of the graph, just as a guide to show how scattered the points are).

The next step is to use multiple regression to find a formula that describe the relationship between A1 and A2. Figure 5.13 show the result using different orders for the regression polynomial. It is clear that the higher the order of the polynomial; the better the regression result is (R-squared is larger) [KLEI, 98]. In practice, second or third order polynomials usually yield acceptable results with reasonable computational overhead [DRAP, 98].

In real life examples, as we will demonstrate later in this chapter, first order or linear regression may be sufficient, as the improvement in the Determination Coefficient is negligible with higher orders.

Set	A1	A2	A3	A1*A2	A1*A3	A2*A3
Case 1:	0.1	0.2	0.1	0.02	0.01	0.02
Case 2:	0.2	0.1	0.5	0.02	0.1	0.05
Case 3:	0.3	0.2	0.7	0.06	0.21	0.14
Case 4:	0.4	0.3	0.4	0.12	0.16	0.12
Case 5:	0.5	0.6	0.8	0.3	0.4	0.48
Case 6:	0.6	0.4	0.1	0.24	0.06	0.04
Case 7:	0.7	0.7	0.3	0.49	0.21	0.21
Case 8:	0.8	0.6	0.2	0.48	0.16	0.12
Case 9:	0.9	0.4	0.8	0.36	0.72	0.32
Case 10:	1	0.5	0.5	0.5	0.5	0.25
E():	0.55	0.4	0.44	0.259	0.253	0.175
Var():	0.091667	0.04	0.071556	0.03841	0.049046	0.020672
StD():	0.302765	0.2	0.267499	0.195985	0.221462	0.143778
Corr():				0.715626	-0.02077	0.015684
Det():				0.512121	0.000431	0.000246

Table 5.3: A sample input set.

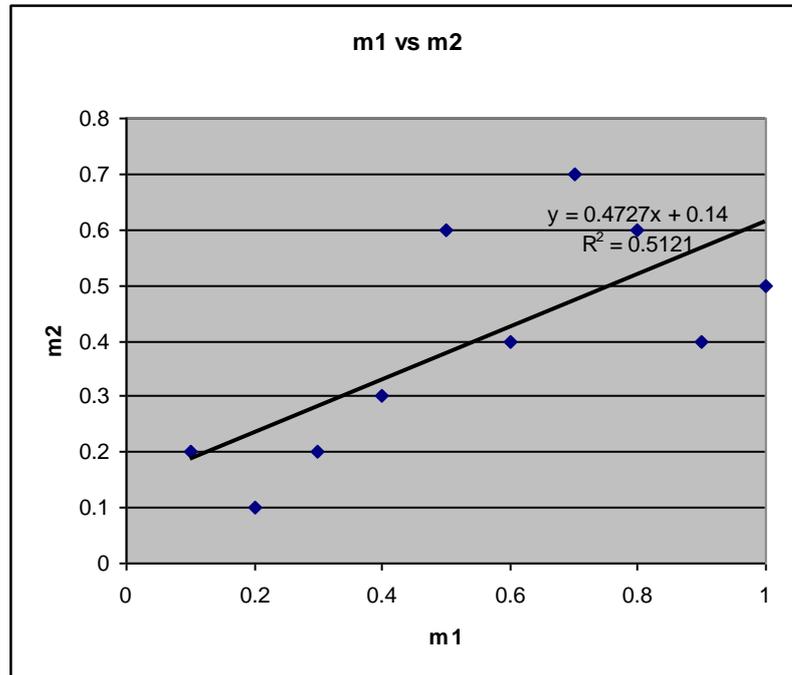


Figure 5.10: m1 versus m2

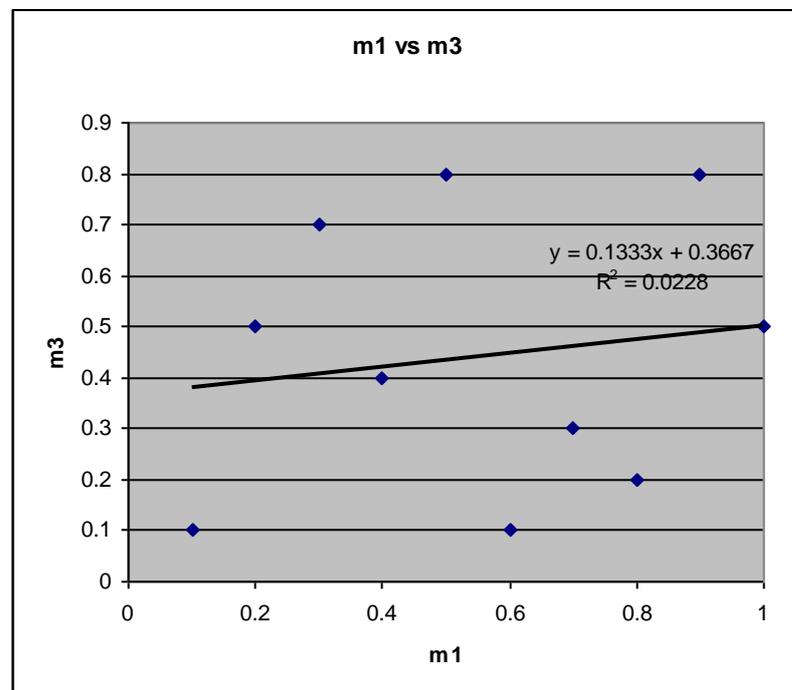


Figure 5.11: m1 versus m3

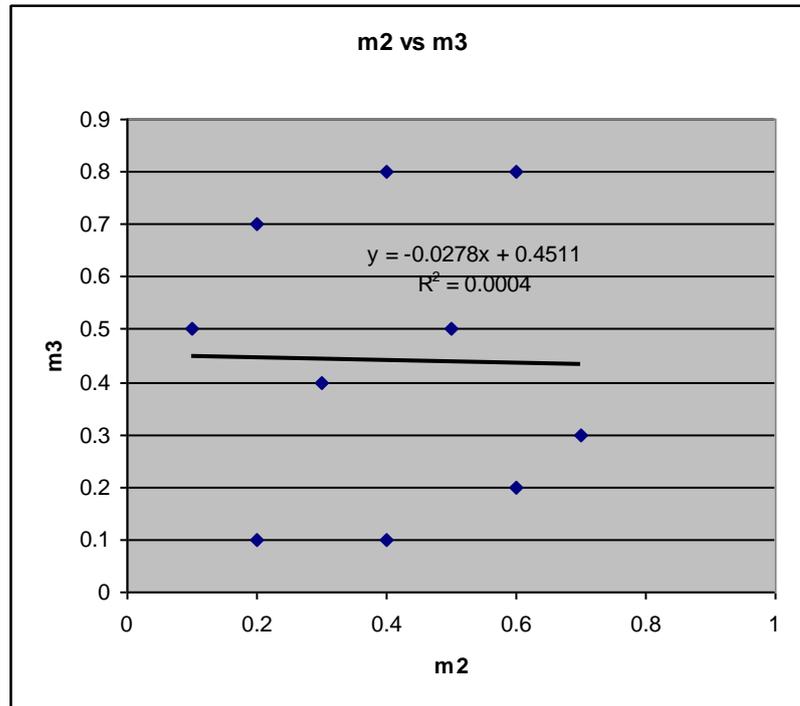


Figure 5.12: m1 versus m2

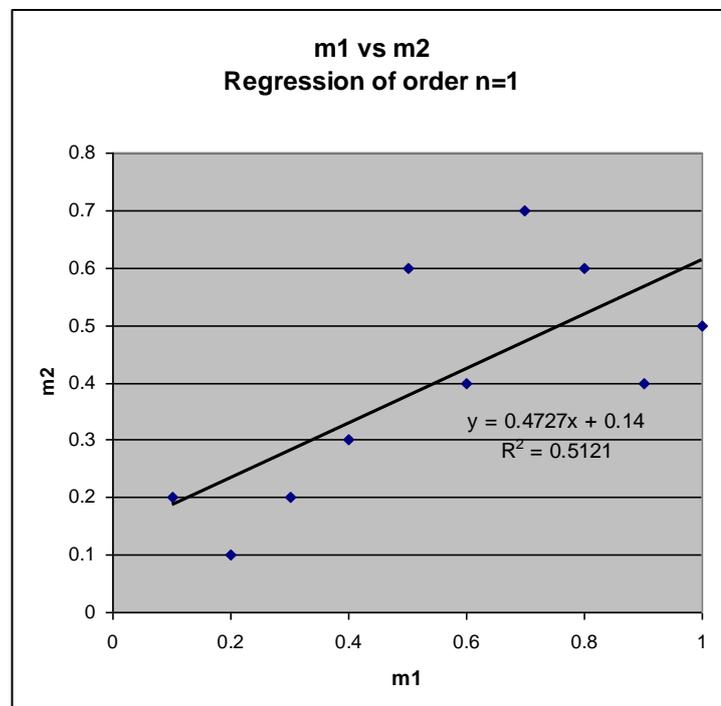


Figure 5.13a: m1 versus m2 using first order regression.

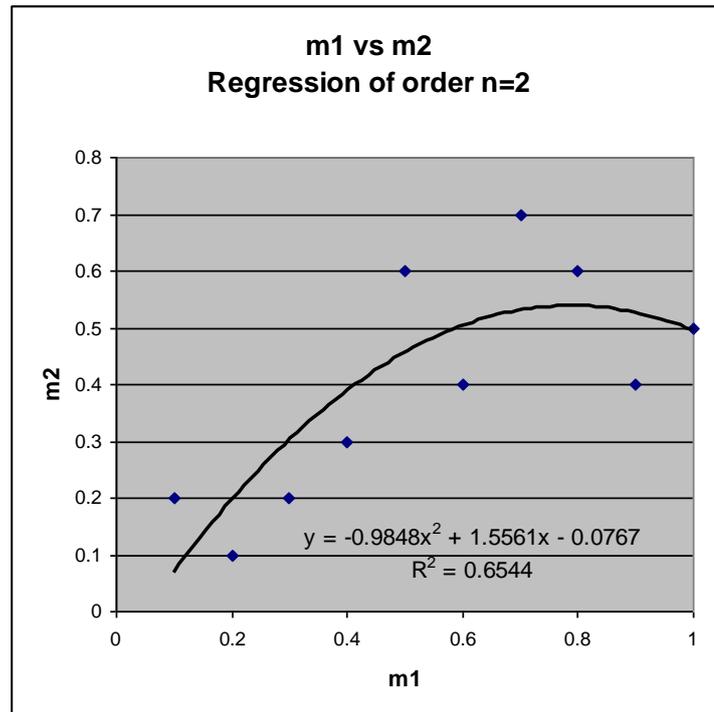


Figure 5.13b: m1 versus m2 using second order regression.

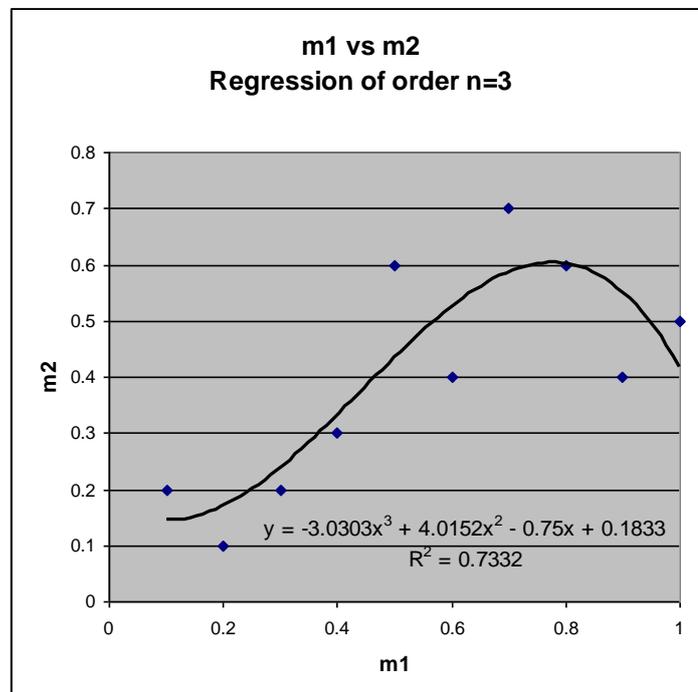


Figure 5.13c: m1 versus m2 using third order regression.

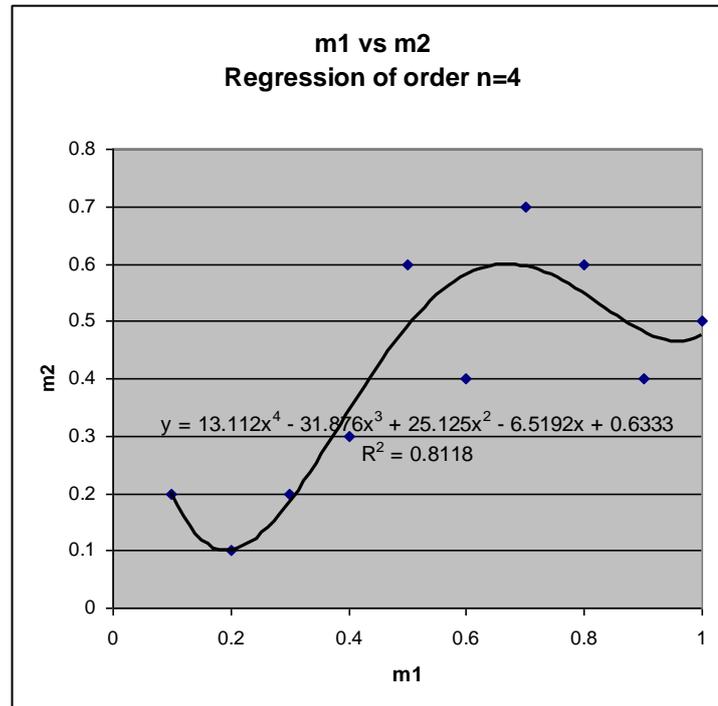


Figure 5.13d: m1 versus m2 using fourth order regression.

5.4.3.4 APIC and Analysis of the Tree

The above example showed us a very important aspect of the algorithm we have presented. Not only can the algorithm give an indication of the missing values, but it would be very useful in analyzing the tree structure itself.

Knowing, for example that m1 is linearly correlated to m2, we can actually reduce the tree inputs, by omitting m2 and using the correlation formula to generate m2 through the m1.

This could dramatically reduce the number of user inputs to the tree, thus reducing the overall error of the decision making process.

This would provide a good tool for analysing the tree and not only predicting missing inputs. It is hence useful both in the training stage as well as the decision support stage. Eventually this could lead to restructuring the tree based on the data, which could act as a feedback algorithm. This is beyond the scope of this work and can be addressed in future work.

4.4.3.5 APIC and iARRIVE

We have demonstrated how the APIC algorithm can be used to predict some of the missing inputs with a certain error margin. We can now extend the use of the algorithm, to be used in conjunction with iARRIVE to populate some of the missing data in the original training set. This means, using APIC in the pre-processing and training stage as well as the Decision support stage.

Assume we start with a basic training set, called the seed. This seed may not be complete itself, meaning some of the inputs may be missing across the samples. We can argue that although some of the inputs are missing, we can still obtain a starting point solution for the RI calculation using iARRIVE by using APIC to replace the missing inputs.

The initial RI values will contain errors, but over time, and by adding new data to the system incrementally (using iARRIVE), this error will eventually dilute and converge towards zero. This is an inherent property in multiple regression and curve fitting, as the far or erroneous points tend to lose their impact on the overall trend the larger the number of points used in the regression [DRAP, 98].

The APIC will need to be modified though, as we won't have the luxury of picking the most correlated pair out of all possible combination of the inputs in all the cases. Only the cases with complete inputs to the variable in question are paired. In other words, only the inputs that have corresponding values in the various test cases. An example is shown in table 5.4.

When calculating the correlation between Input1 and Input2 for example, only the cases with corresponding entries in both inputs are used (highlighted in Table 5.5). The same applies when calculating the correlating between Input1 and Input3 for example (Table 5.6).

Test Case	Input1	Input2	Input3	Input4	Input5
1	0.3	0.4		0.6	
2	0.2		0.6	0.3	0.1
3	0.5	0.2	0.4		0.3
4		0.5	0.2	0.6	0.4
5	0.6	0.8	0.3	0.4	
6	0.4	0.5			0.7
7	0.7	0.2	0.5	0.5	0.4
8			0.6		0.3
9	0.4	0.6		0.4	
10	0.5	0.3		0.2	
11		0.7	0.3	0.8	
12	0.6				0.5

Table 5.4 : A sample training set.

Test Case	Input1	Input2	Input3	Input4	Input5
1	0.3	0.4		0.6	
2	0.2		0.6	0.3	0.1
3	0.5	0.2	0.4		0.3
4		0.5	0.2	0.6	0.4
5	0.6	0.8	0.3	0.4	
6	0.4	0.5			0.7
7	0.7	0.2	0.5	0.5	0.4
8			0.6		0.3
9	0.4	0.6		0.4	
10	0.5	0.3		0.2	
11		0.7	0.3	0.8	
12	0.6				0.5

Table 5.5: The cases used for correlating Input1 and Input2.

This will inevitably mean that some correlations will be calculated using fewer and different test cases than others. In the case of Input1 and Input2, we have seven viable cases (1,3,5,6,7,9 and 10), whereas in case of Input1 and Input3 we only have four (2,3,5 and 7). (Note that in our real life training set, presented in the Results Chapter we have over 700 test cases).

Test Case	Input1	Input2	Input3	Input4	Input5
1	0.3	0.4		0.6	
2	0.2		0.6	0.3	0.1
3	0.5	0.2	0.4		0.3
4		0.5	0.2	0.6	0.4
5	0.6	0.8	0.3	0.4	
6	0.4	0.5			0.7
7	0.7	0.2	0.5	0.5	0.4
8			0.6		0.3
9	0.4	0.6		0.4	
10	0.5	0.3		0.2	
11		0.7	0.3	0.8	
12	0.6				0.5

Table 5.6: The cases used for correlating Input1 and Input3.

This does mean that errors will be present at the training stage, firstly due to the inconsistent correlation calculation and secondly because of using APIC itself. But in the absence of real values, this would be the only way forward. And because the regression process will tend to smooth the curves or average the noise, it means entries that are far off or inconsistent will tend to be diluted and their effect will be minimized. The larger the number of records the less the error becomes. The overall error should eventually converge to zero as we add more training sets to iARRIVE.

Another argument to support this approach is that the original seed itself, even if complete, may contain errors and inaccurate data. As we have demonstrated in iARRIVE, the errors in the seed can even be exposed once the training of the initial RI calculations is complete (through the regression coefficient, R^2).

All in all, the various errors will dilute over time, and the more data we have; the less errors we will get.

5.4.3.6 Case Study

In the following example, we demonstrate the application of APIC with more than two variables.

In this example, we present a case study showing training data with several missing inputs at different input cases (rows) as in Table 5.7.

The missing values are shown in shaded cells. All calculations were performed using MS Excel 2007.

We will extend APIC here to more than two variables, so we will analyse the relationship between two, three and four variables. In order to be able to assess the strength of the relationship of more than two variables, we will be using the determination coefficient, r^2 , which works for two or more variables.

This means that if only one variable value is missing in a row, we could potentially predict its value from the other two. This will be possible as we will use multivariate Regression [KLEI, 98], [SHEL, 72] to find the formula that relates the missing variable to the other two.

If two values are missing, then we could only use the value of the remaining variable in the row to predict the missing value.

So, in effect, we need to generate a table with all the formulas connecting a variable (in this example A) with all other combinations of the one, two and three variables. We also need to judge the relationship strength using the r^2 coefficient.

We need to find: $A = f(B)$, $A = f(C)$, $A = f(D)$, $A = f(B,C)$, $A = f(C,D)$, $A = f(B,D)$, and $A = f(B,C,D)$.

The table can then be used to predict the missing A value, depending on which other values are present. It is clear from the table that the missing data is randomly chosen, so some rows (input cases) have one or two values.

In order to apply APIC, we need to use only complete rows for the regression analysis first: So if we are analysing the relationship between A and B, we will only use rows with A and B present. We use multiple linear Regression to generate the formulae that link A with B, C and D and all their combinations. We have explored higher order regression, but the improvement was not significant and did not justify the complexity of the process (as in Figures 5.14 to 5.16).

A	B	C	D
1	2	4	53
2	3	7	50
3	6	6	52
4	7	12	48
5	11	17	33
6	13	21	29
7	12	19	44
8	13	26	52
9	20	29	18
4	9	11	26
7	14	9	33
2	5	8	
8	17		37
5	13	15	43
7	12	19	51
	6		24
9		20	44
6	15	9	
4	9	8	20
1		4	48
	5	8	41
4	10	12	49
	15	6	
7	11	26	46
5	12		29
	5	10	30
6	11		
9	17	20	39
1	3		29

Table 5.7: Original inputs with missing data.

The order of regression will depend on the data on hand and can be used as a tuning parameter which gives more flexibility to the algorithm.

A	B
1	2
2	3
3	6
4	7
5	11
6	13
7	12
8	13
9	20
4	9
7	14
2	5
8	17
5	13
7	12
6	15
4	9
4	10
7	11
5	12
6	11
9	17
1	3

SUMMARY
OUTPUT

<i>Regression Statistics</i>	
Multiple R	0.933507
R Square	0.871436
Adjusted R Square	0.865314
Standard Error	0.877982
Observations	23

$$A = 0.4755 B + 0.1519$$

Figure 5.14a: Analysis of the relationship between A and B

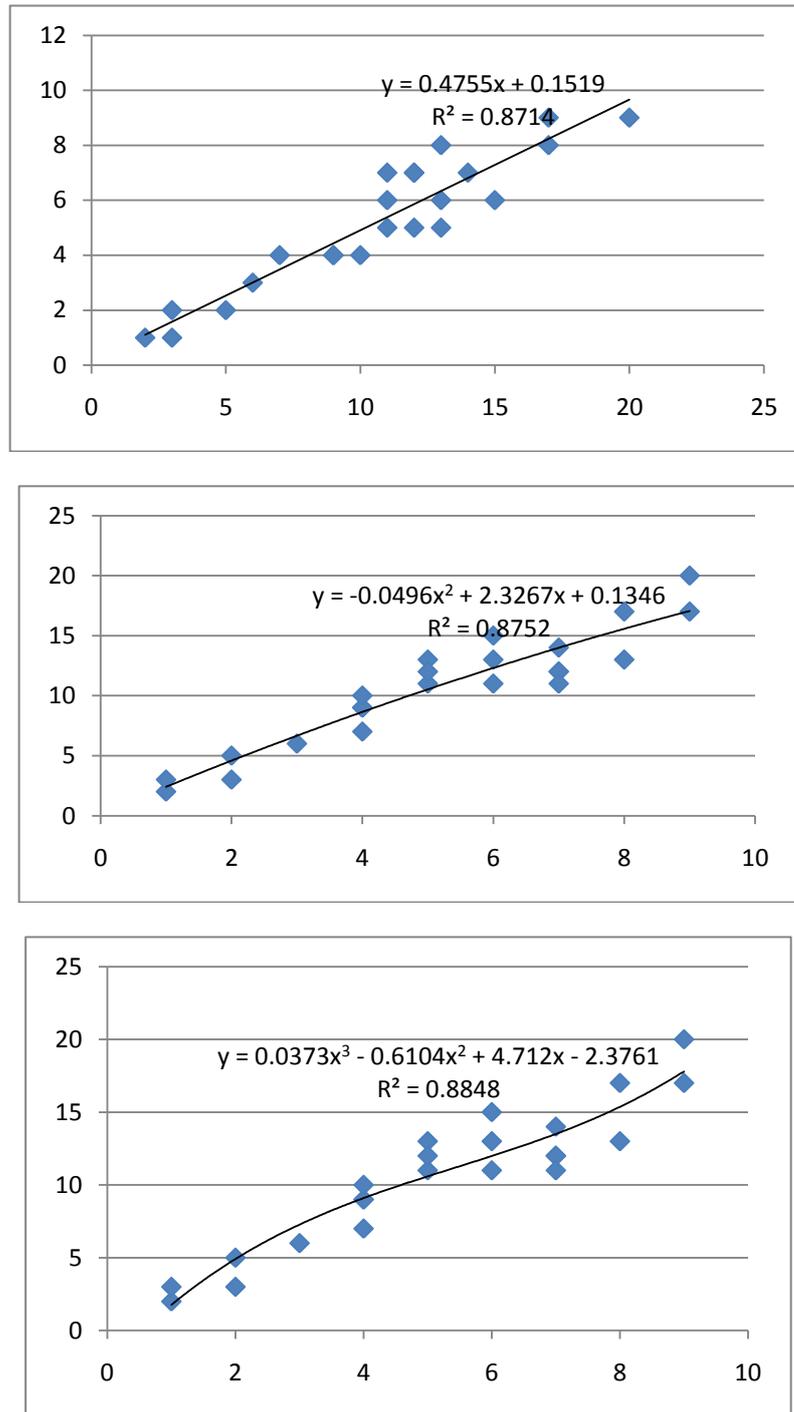


Figure 5.14b: Analysis of the relationship between A and B using APIC in order 1, 2 and 3.

A	C
1	4
2	7
3	6
4	12
5	17
6	21
7	19
8	26
9	29
4	11
7	9
2	8
5	15
7	19
9	20
6	9
4	8
1	4
4	12
7	26
9	20

SUMMARY OUTPUT

<i>Regression Statistics</i>	
Multiple R	0.84015
R Square	0.705853
Adjusted R Square	0.690371
Standard Error	1.428496
Observations	21

$$A = 0.2853 C + 1.1351$$

Figure 5.15a: Analysis of the relationship between A and C.

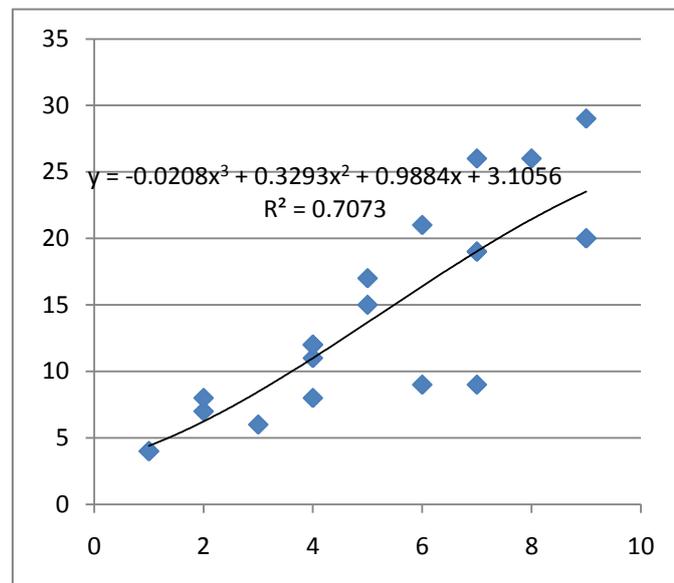
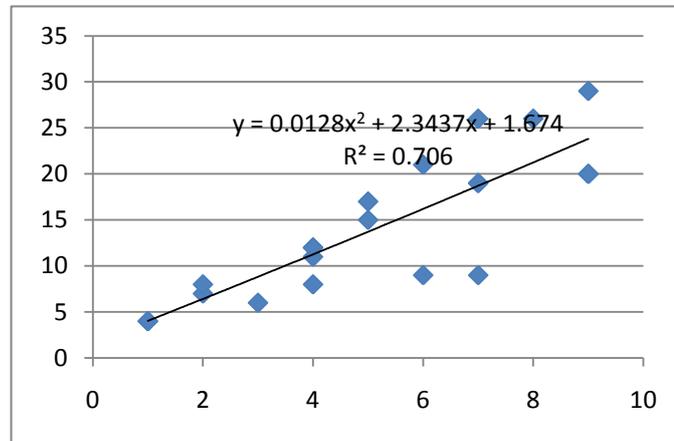
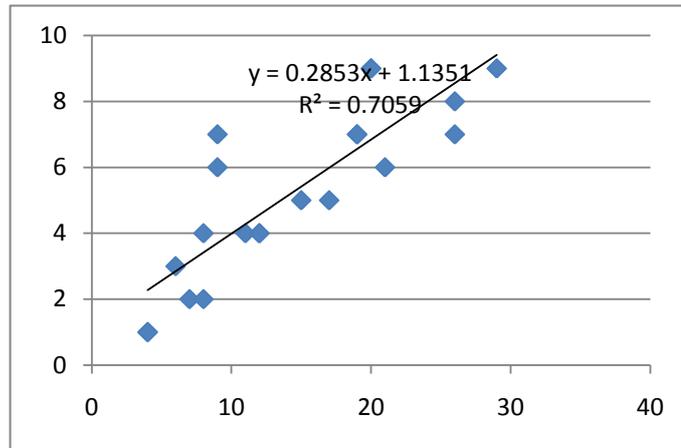


Figure 5.15b: Analysis of the relationship between A and C using APIC in order 1, 2 and 3.

A	D
1	53
2	50
3	52
4	48
5	33
6	29
7	44
8	52
9	18
4	26
7	33
8	37
5	43
7	51
9	44
4	20
1	48
4	49
7	46
5	29
9	39
1	29

SUMMARY OUTPUT

<i>Regression Statistics</i>	
Multiple R	0.176587
R Square	0.031183
Adjusted R Square	-0.01726
Standard Error	2.662701
Observations	22

$$A = -0.0426 D + 6.9639$$

Figure 5.16a: Analysis of the relationship between A and D.

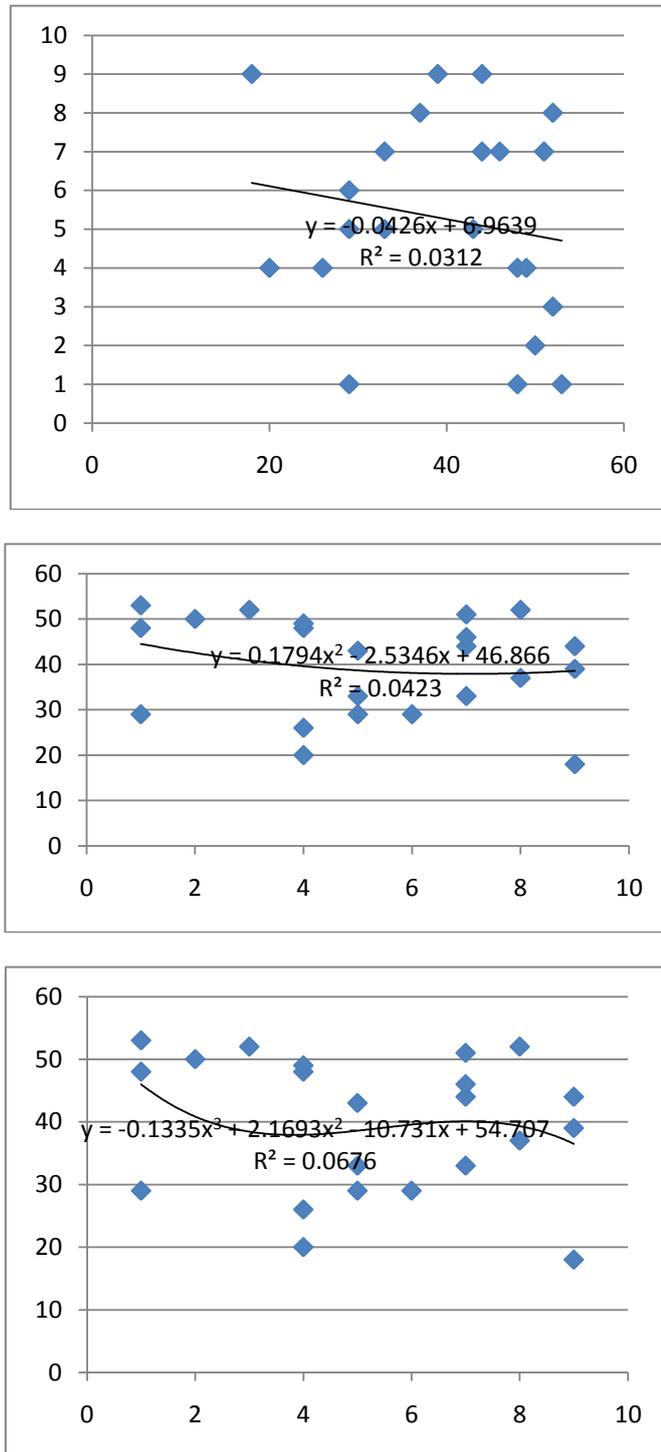


Figure 5.16b: Analysis of the relationship between A and D using APIC in order 1, 2 and 3.

A	B	C
1	2	4
2	3	7
3	6	6
4	7	12
5	11	17
6	13	21
7	12	19
8	13	26
9	20	29
4	9	11
7	14	9
2	5	8
5	13	15
7	12	19
6	15	9
4	9	8
4	10	12
7	11	26
9	17	20

<i>Regression Statistics</i>	
Multiple R	0.954692
R Square	0.911438
Adjusted R Square	0.900367
Standard Error	0.74281
Observations	19

$$A = -0.3469 B + 0.10777 C - 0.00242$$

Figure 5.17: Analysis of the relationship between A, B and C using APIC.

A	C	D
1	4	53
2	7	50
3	6	52
4	12	48
5	17	33
6	21	29
7	19	44
8	26	52
9	29	18
4	11	26
7	9	33
5	15	43
7	19	51
9	20	44
4	8	20
1	4	48
4	12	49
7	26	46
9	20	39

<i>Regression Statistics</i>	
Multiple R	0.865372495
R Square	0.748869555
Adjusted R Square	0.71747825
Standard Error	1.374804361
Observations	19

$$A=0.281247 X C - 0.0236 X D - 2.1163$$

Figure 5.18: Analysis of the relationship between A, C and D using APIC.

A	B	D
1	2	53
2	3	50
3	6	52
4	7	48
5	11	33
6	13	29
7	12	44
8	13	52
9	20	18
4	9	26
7	14	33
8	17	37
5	13	43
7	12	51
4	9	20
4	10	49
7	11	46
5	12	29
9	17	39
1	3	29

<i>Regression Statistics</i>	
Multiple R	0.957579049
R Square	0.916957635
Adjusted R Square	0.907187945
Standard Error	0.746893096
Observations	20

$$A = 0.5205 B + 0.0454 D - 2.0438$$

Figure 5.19: Analysis of the relationship between A, B and D using APIC.

A	B	C	D
1	2	4	53
2	3	7	50
3	6	6	52
4	7	12	48
5	11	17	33
6	13	21	29
7	12	19	44
8	13	26	52
9	20	29	18
4	9	11	26
7	14	9	33
5	13	15	43
7	12	19	51
4	9	8	20
4	10	12	49
7	11	26	46
9	17	20	39

<i>Regression Statistics</i>	
Multiple R	0.960498932
R Square	0.922558199
Adjusted R Square	0.904687014
Standard Error	0.724515184
Observations	17

$$A = 0.4191 B + 0.078349 C + 0.032946 D - 1.60792$$

Figure 5.20: Analysis of the relationship between A,, B, C and D using APIC.

It is clear from the results (Table 5.8) that the more variable values are available, the better the quality of the prediction (R2).

We call these functions: the Prediction Functions (PF) of A. We call table 5.8 the Prediction Table (PT).

R2	Function	Cases
0.92255819	A= 0.4191 B + 0.078349 C + 0.032946 D - 1.60792	17
0.91695	A= 0.5205 B + 0.0454 D - 2.0438	20
0.911438	A=0.3469 B + 0.10777 C - 0.00242	19
0.8714	A = 0.4755 B + 0.1519	23
0.748869	A=0.281247 C - 0.0236 D - 2.1163	19
0.7059	A = 0.2853 C + 1.1351	21
0.0312	A = -0.0426 D + 6.9639	22

Table 5.8: The Coefficient of Determination for various estimators of A.

In Table 5.9, we show the predicted values for A and the formulas used to generate them. We use the highest possible formula in the table (i.e., with the largest Determination Coefficient – we sorted the PT based on that), that contains the largest number of variables (i.e. values) present in the current row. This will ensure higher correlation and better prediction.

In the final test example, we predict one of the existing A values (original values are bold in the first row in the table below), as a test case. We use one, two and three variable to predict it, and it is clear that the larger number of known values means a better prediction and less error. This agrees with our finding in the A table and the R2 values associated with each formula.

This is largely due to the fact that the error spreads and in cases cancels or at least reduces each other. Also more variables mean more information.

The same process should be repeated for B, C and D, resulting in PF's for all variables. It is important to note that we need to conduct the analysis on B, C and D with the original data (i.e. with the missing values), otherwise we could influence the original data by our inferred values for A for example.

A	B	C	D	
1	2	4	53	
2	3	7	50	
3	6	6	52	
4	7	12	48	
5	11	17	33	
6	13	21	29	
7	12	19	44	
8	13	26	52	
9	20	29	18	
4	9	11	26	
7	14	9	33	
2	5	8		
8	17		37	
5	13	15	43	
7	12	19	51	
2.1688	6		24	$A = 0.5205 B + 0.0454 D - 2.0438$
9		20	44	
6	15	9		
4	9	8	20	
1		4	48	
2.4632	5	8	41	$A = 0.4191 B + 0.078349 C + 0.032946 D - 1.60792$
4	10	12	49	
5.8477	15	6		$A = 0.3469 B + 0.10777 C - 0.00242$
7	11	26	46	
5	12		29	
2.2581	5	10	30	$A = 0.4191 B + 0.078349 C + 0.032946 D - 1.60792$
6	11			
9	17	20	39	
1	3		29	

Table 5.9: The predicted values of A using APIC and the corresponding Prediction Functions (PF) used for each.

8	13	26	52	Original Data
7.5883	13	26	52	$A = 0.4191 B + 0.078349 C + 0.032946 D - 1.60792$
7.0835	13		52	$A = 0.5205 B + 0.0454 D - 2.0438$
6.3334	13			$A = 0.4755 B + 0.1519$
8.5535		26		$A = 0.2853 C + 1.1351$

Table 5.10: Verifying the APIC on existing data.

The output of the APIC would be n Prediction Tables (PTs) representing all the variables in the input set. These can then be used predict missing values in the initial training set, that is used to deduce the model, i.e. RI values.

These tables can then be used to predict missing values at the assessment stage too.

A simple flowchart of the process is presented in Figure 5.21.

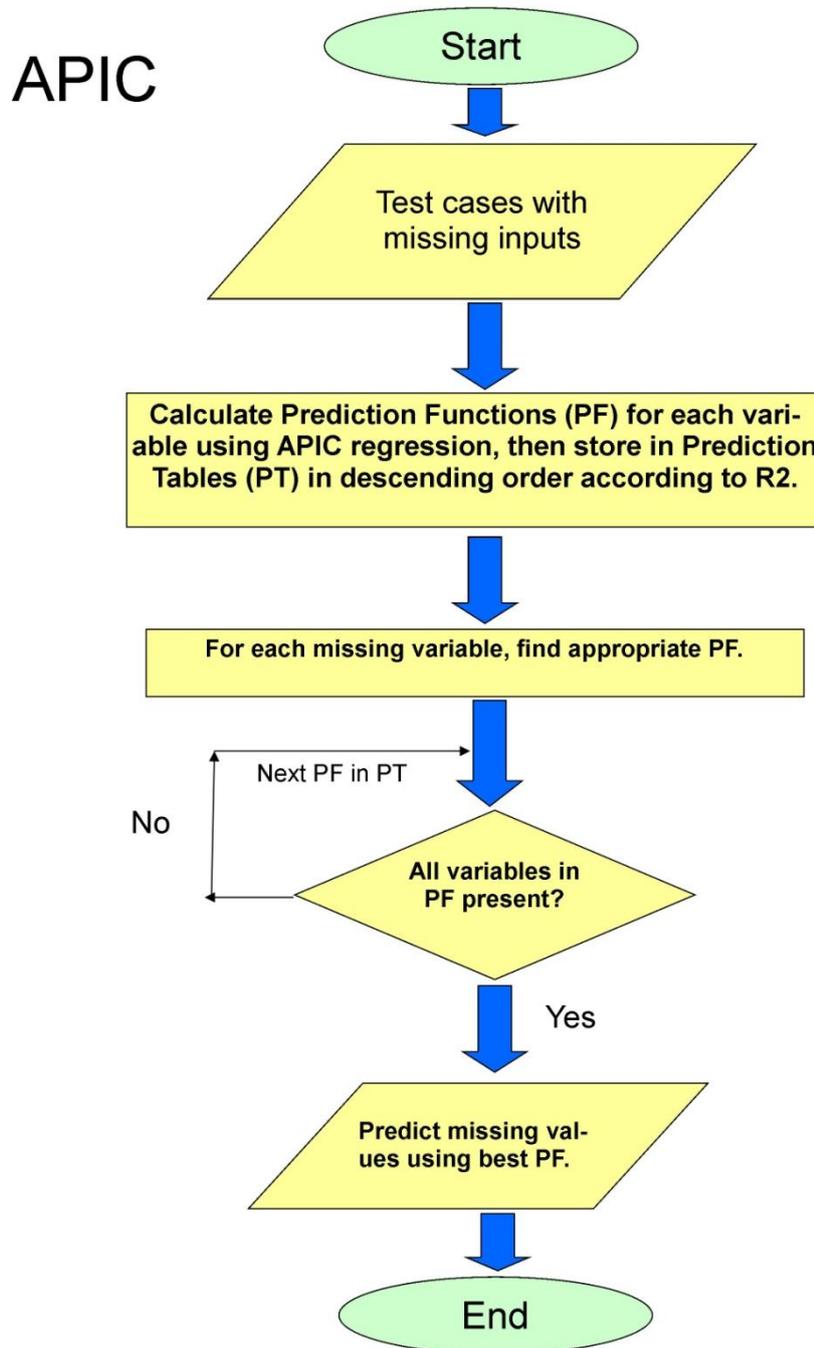


Figure 5.21: Complete APIC flowchart.

5.4.3.7 Analysis:

the number of variables in the case of GRiST is very large at over 200. This means the number of combinations is huge and the number of cases is likewise large (tens of thousands and increasing all the time). This begs the question of when to stop adding cases or variable combinations? To find the formula connecting a certain variable, A1 with the rest, say $A1 = f(A2, A3, \dots, A_n)$ requires a certain number of complete input rows.

The minimum number of complete rows needed to predict a formula using regression is equal to the number of variable used (n).

It is clear from the examples in the previous section, that the larger the number of variables, the less number of complete rows will be available.

This could be a challenge in real data where many cases are incomplete, especially in medical records. This means that some combinations may be impossible to predict. In this case, we will need to get the formula for the next available combination in the table, with fewer variables. E.g.: $A1 = f(A2, A3, \dots, A_{(n-1)})$. This is shown in Table 5.8 in the previous section, where variable A can be found using several formulas of combinations on B, C and D, depending on the available data in the other three variables.

The other issue would be the accuracy of the regression process: This depends highly on the number of complete training cases used. This means that one relationship or formula could be more accurate simply because of the higher number of cases involved in the regression process.

On the other hand, a formula with less number of cases but a higher Determination Coefficient, i.e., stronger relationship, would mean a better prediction.

Another factor would be the number of variables involved. As we showed in the previous section, the more variables we have in the relationship, the more accurate the prediction would be.

It would be interesting to study the effect of the number of complete cases on the determination coefficient. It would also be important to study the effect of the number of cases on the prediction and link to the number of variables. The complexity of the algorithm will also depend on this analysis. This is out of the scope of this work, and can be addressed in future work.

5.4.3.8 Error in results:

The following error analysis is performed on the previous case study (5.4.3.6) . Table 5.11 shows the actual A values, and corresponding predictions using each of the formulae in Table 5.8.

A	A= F(B)	A= F(C)	A=F(D)	A=f(B,C)	A=f(C,D)	A=f(B,D)	A=f(B,C,D)
1	1.1029	2.2763	4.7061	1.12246	1.990488	1.4034	1.289814
2	1.5784	3.1322	4.8339	1.79267	2.905029	1.7877	1.845123
3	3.0049	2.8469	4.7487	2.7256	2.576582	3.44	3.089966
4	3.4804	4.5587	4.9191	3.71912	4.358464	3.7789	3.847376
5	5.3824	5.9852	5.5581	5.64557	6.118699	5.1799	5.421331
6	6.3334	7.1264	5.7285	6.77045	7.338087	6.0393	6.441143
7	5.8579	6.5558	5.0895	6.20801	6.421593	6.1998	6.359535
8	6.3334	8.5529	4.7487	7.3093	8.201522	7.0835	7.590646
9	9.6619	9.4088	6.1971	10.06091	9.847663	9.1834	9.639229
4	4.4314	4.2734	5.8563	4.30515	4.596417	3.8211	3.882415
7	6.8089	3.7028	5.5581	5.82411	3.868723	6.7414	6.051839
5	6.3334	5.4146	5.1321	6.12383	5.320205	6.6749	6.432293
7	5.8579	6.5558	4.7913	6.20801	6.256393	6.5176	6.590157
4	4.4314	3.4175	6.1119	3.98184	3.894276	3.5487	3.449692
4	4.9069	4.5587	4.8765	4.75982	4.334864	5.3858	5.137622
7	5.3824	8.5529	5.0043	6.6155	8.343122	5.7701	6.55477
9	8.2354	6.8411	5.3025	8.05028	6.82084	8.5753	8.368654

Table 5.11: The actual values of A versus predicted values using APIC.

Note that in order to test the formulae we only used the complete rows (or inputs) in the original input set.

Table 5.12 shows the absolute errors between the corresponding sets. Table 5.13 then shows the calculated relative error in each cell, compared to the actual values of A. This is then used in each column (i.e. formula) to calculate the average relative error for each of the formulae, and the results are shown in Table 5.14.

It is clear that the results are consistent with the original Determination Coefficients in Table 5.8, with minor deviations. Which is logical, as the better the fit, the less the

error. The best result on average was using $A=f(B,C,D)$, with average relative error of 11%. This may sound high, but due to the small number of input parameters (only four) and the relatively small number of input test cases (in GRiST we have over 2000 test cases), this level of error is acceptable.

With larger number of data sets, the error should be reduced as more data will be available for the regression and thus spikes of inconsistent data will have less weight in the total process.

E1	E2	E3	E4	E5	E6	E7
0.1029	1.2763	3.7061	0.12246	0.990488	0.4034	0.289814
0.4216	1.1322	2.8339	0.20733	0.905029	0.2123	0.154877
0.0049	0.1531	1.7487	0.2744	0.423418	0.44	0.089966
0.5196	0.5587	0.9191	0.28088	0.358464	0.2211	0.152624
0.3824	0.9852	0.5581	0.64557	1.118699	0.1799	0.421331
0.3334	1.1264	0.2715	0.77045	1.338087	0.0393	0.441143
1.1421	0.4442	1.9105	0.79199	0.578407	0.8002	0.640465
1.6666	0.5529	3.2513	0.6907	0.201522	0.9165	0.409354
0.6619	0.4088	2.8029	1.06091	0.847663	0.1834	0.639229
0.4314	0.2734	1.8563	0.30515	0.596417	0.1789	0.117585
0.1911	3.2972	1.4419	1.17589	3.131277	0.2586	0.948161
1.3334	0.4146	0.1321	1.12383	0.320205	1.6749	1.432293
1.1421	0.4442	2.2087	0.79199	0.743607	0.4824	0.409843
0.4314	0.5825	2.1119	0.01816	0.105724	0.4513	0.550308
0.9069	0.5587	0.8765	0.75982	0.334864	1.3858	1.137622
1.6176	1.5529	1.9957	0.3845	1.343122	1.2299	0.44523
0.7646	2.1589	3.6975	0.94972	2.17916	0.4247	0.631346

Table 5.12: The absolute errors between actual and predicted values.

R1	R2	R3	R4	R5	R6	R7
0.1029	1.2763	3.7061	0.12246	0.990488	0.4034	0.289814
0.2108	0.5661	1.41695	0.103665	0.452515	0.10615	0.077439
0.001633	0.051033	0.5829	0.091467	0.141139	0.146667	0.029989
0.1299	0.139675	0.229775	0.07022	0.089616	0.055275	0.038156
0.07648	0.19704	0.11162	0.129114	0.22374	0.03598	0.084266
0.055567	0.187733	0.04525	0.128408	0.223015	0.00655	0.073524
0.163157	0.063457	0.272929	0.113141	0.08263	0.114314	0.091495
0.208325	0.069112	0.406413	0.086338	0.02519	0.114563	0.051169
0.073544	0.045422	0.311433	0.117879	0.094185	0.020378	0.071025
0.10785	0.06835	0.464075	0.076288	0.149104	0.044725	0.029396
0.0273	0.471029	0.205986	0.167984	0.447325	0.036943	0.135452
0.26668	0.08292	0.02642	0.224766	0.064041	0.33498	0.286459
0.163157	0.063457	0.315529	0.113141	0.10623	0.068914	0.058549
0.10785	0.145625	0.527975	0.00454	0.026431	0.112825	0.137577
0.226725	0.139675	0.219125	0.189955	0.083716	0.34645	0.284406
0.231086	0.221843	0.2851	0.054929	0.191875	0.1757	0.063604
0.084956	0.239878	0.410833	0.105524	0.242129	0.047189	0.07015

Table 5.13: The relative errors between actual and predicted values for A.

A= F(B)	A= F(C)	A=F(D)	A=f(B,C)	A=f(C,D)	A=f(B,D)	A=f(B,C,D)
0.131642	0.236979	0.561083	0.111754	0.213727	0.127706	0.110145

Table 5.14: Average relative errors for each prediction function.

5.5 Summary

In this Chapter, we have tackled the problem of missing data both at the training stage (induction) in iARRIVE and the decision stage. This is a very important part of any decision support system, as data may be incomplete, inconsistent or both.

We have introduced a method to predict missing inputs in a decision tree, APIC. We have used this algorithm and a variation of it using regression and the coefficient of determination to predict missing inputs both at the training stage and the decision support stage.

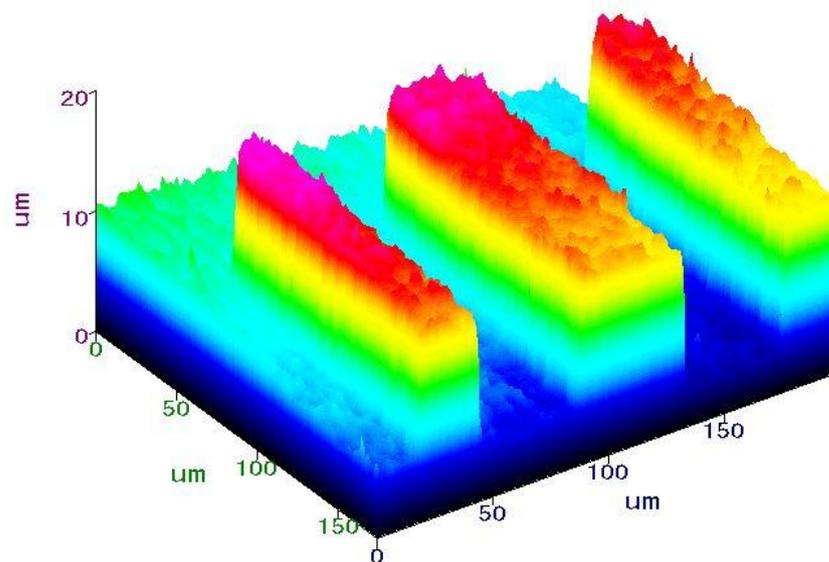
We have conducted error analysis at the training stage as well as the decision support stage. We have also calculated estimates for error due to missing inputs.

In the next chapter, we present the results obtained using real patient data, from our GRiST system. We applied the algorithms presented in the past few chapters on real data and presented an error analysis on the obtained results.



Chapter 6

Results



This Chapter covers the following:

- Introduction
- Using Real Data
- Data Description
- Result Analysis
- Error Distribution
- Summary

6.1 Introduction

In previous chapters, the methodology and the development of iARRIVE algorithm were discussed, as well as some of the practical issues that might face the implementation of iARRIVE. The Data Pre-processing Chapter covered some of the issues associated with missing values. In this chapter, we apply those algorithms to real data obtained from GRiST.

The System has been implemented in MatLab, using the clinicians inputs, in both the elicitation stage (Membership Grades (MG) parameter elicitation) as shown in Appendix B, and the actual patient data (the answers to the patient questionnaires).

This chapter shows the results obtained through the system. The implementation and code is presented in Appendix D.

6.2 Using Real Data

The data used for the trial came from live databases currently used by NHS Mental Health trusts deploying GRiST in the UK. All patient records were anonymous because no patient identification information was stored in the tables and neither were there any derived data that could somehow be used to trace back to the patient. From the research perspective, then, these are completely abstract rows of patient cues linked to clinical risk judgements that have no real-world connection. This is an important ethical feature and requirement by the trusts for data protection issues.

6.3 Data Description

The available database of over 10,000 records had to be prepared and cleaned before the iARRIVE algorithm could be applied. Data cleaning included discarding incomplete assessments and bad records. Missing data is a problem as the learning algorithms presented in Chapter three (iARRIVE) require a complete set of variables to work on.

Bad patient records include records with vital values missing (e.g. assessment date, patient's age) as well as missing Risk judgements (which are crucial for the learning algorithm because they represent the solution of the equation for the input cues). Techniques had to be used for some early screening through the database queries and tree structures to prune some of the bad records. In Chapter 5, the APIC algorithm was presented to deal with the missing values problem. An outline of the Data Cleaning process is presented in section 6.3.2 below.

After the data cleaning and pruning stage, to get to the best possible seed for iARRIVE algorithm, the outcome was that the data pool was reduced to 2000 records, and after elimination, 771 satisfactory patient records were used.

The Suicide Risk Tree was deployed for this trial. Each record contained 184 questions or leaf nodes, which represent the answers to the patient questionnaire. These are listed in table 6.1.

Table 6.1: The leaf nodes codes for Suicide Risk in the patient data sample.

suic-most-rec_answer	gen-distress_answer	gen-relat-detr_answer
suic-first-occ_answer	gen-jealous_answer	gen-relat-detr-chg_answer
suic-how-many_answer	gen-plans-future_answer	gen-move-freq_answer
suic-escalate_answer	gen-life-not-livng_answer	gen-home-type_answer
suic-planning_answer	grandiosity_answer	gen-isol-accom_answer
suic-note-prev_answer	worthlessness_answer	gen-neigrbrhd-rsky_answer
suic-discovery_answer	gen-motivation_answer	gen-accom-hm-care_answer
suic-lethality_answer	gen-voice-hal_answer	gen-accom-habitbl_answer
suic-ser-succd_answer	gen-paranoid-del_answer	gen-perc-debt-anx_answer
suic-regret_answer	gen-impaired-cog_answer	gen-poverty_answer
suic-leth-insght_answer	gen-rsk-behavr_answer	gen-job-chg-frq_answer
suic-plan-real_answer	gen-unint-risk-behavr_answer	gen-rec-bad-job-ch_answer
suic-plan-dtail_answer	gen-sleep-dist_answer	gen-rsk-behavr_answer
suic-steps-takn_answer	gen-diet-eating_answer	gen-unint-risk-behavr_answer
suic-prosp-leth_answer	gen-diet-weigt-ext_answer	gen-sleep-dist_answer
suic-int-inform_answer	gen-diet-weigt-chg_answer	gen-diet-eating_answer
suic-eol-prep_answer	gen-diet-drink_answer	gen-diet-weigt-ext_answer
suic-pot-trig_answer	gen-unusl-rec-bhvr_answer	gen-diet-weigt-chg_answer
suic-p-trig-mtch_answer	gen-chall-bhvr_answer	gen-diet-drink_answer
suic-id-control_answer	gen-day-struct_answer	gen-unusl-rec-bhvr_answer
suic-id-hi-risk_answer	gen-day-actvty-lev_answer	gen-chall-bhvr_answer
suic-id-freq_answer	gen-rapport_answer	gen-day-struct_answer
suic-id-strngth_answer	gen-responsve_answer	gen-day-actvty-lev_answer
insight-resp_answer	gen-gut-assmnt_answer	gen-alc-misuse_answer

<p>suic-rel-belief_answer sn-hair-clothes_answer sn-hygiene_answer sn-recnt-app-chnge_answer sn-skin_answer gen-sh-cuts_answer gen-rapport_answer gen-responsve_answer gen-gut-assmnt_answer gen-risk-aggrsv_answer gen-risk-upbeat_answer gen-coherence_answer gen-distrss-b-lang_answer gen-low-mood_answer gen-threat-move_answer gen-detached_answer gen-avoid-eye-contact_answer gen-eye-movement_answer gen-congruence_answer suic-s-h-behv_answer suic-fam-hist_answer gen-mood-swings_answer gen-negative-self_answer gen-angry-emotns_answer gen-anx-emotns_answer gen-helpless_answer gen-sad_answer gen-distress_answer gen-jealous_answer gen-plans-future_answer gen-life-not-livng_answer grandiosity_answer worthlessness_answer gen-mood-swings_answer gen-negative-self_answer gen-angry-emotns_answer gen-anx-emotns_answer gen-helpless_answer gen-sad_answer</p>	<p>gen-risk-aggrsv_answer gen-risk-upbeat_answer gen-coherence_answer gen-distrss-b-lang_answer gen-low-mood_answer gen-threat-move_answer gen-detached_answer gen-avoid-eye-contact_answer gen-eye-movement_answer gen-congruence_answer gen-phys-withd_answer gen-mental-withd_answer gen-dep-stage_answer gen-mentl-insght_answer gen-mania_answer gen-voice-dang-s_answer gen-voice-dang-o_answer gen-prob-act-voice_answer gen-paran-del-spec_answer gen-paran-del-pers_answer gen-prob-act-par-del_answer gen-cog-think-mem_answer gen-concentr_answer gen-learn-disab_answer gen-assertive_answer gen-empathy-abil_answer gen-dependence_answer gen-controlling_answer gen-coping-abil_answer gen-hostile_answer gen-impulse_answer gen-reliable_answer gen-phys-withd_answer gen-mental-withd_answer gen-motivation_answer</p>	<p>gen-drug-misuse_answer gen-insght-behvr_answer gen-resp-impct-oth_answer gen-nd-hlp-diff_answer gen-phys-hlth-deg-diag_answer gen-phys-hlth-pain_answer gen-phys-hlth-disa_answer gen-com-imp_answer gen-phys-hlth-det_answer gen-meds-concord_answer gen-serv-perc-supp_answer gen-serv-last-acc_answer gen-med-perc-benft_answer gen-relat-detr-chg_answer gen-rec-bad-job-ch_answer gen-phys-hlth-deg-diag_answer gen-sex-abse-last_answer gen-sex-abse-as-ch_answer gen-phys-abse-last_answer gen-phy-abse-as-ch_answer gen-emot-abse-last_answer gen-emo-abse-as-ch_answer gen-financial-abuse_answer gen-forensic-proc_answer gen-env-grew-up_answer gen-eating-dis_answer gen-educ-expr_answer gen-age_answer gen-gender_answer gen-marital-status_answer gen-accom-num-dep_answer gen-dep-ygnst-age_answer partner-share-acc_answer gen-accm-share-nd_answer gen-ethnicity_answer</p>
--	--	--

Mathworks MatLab [MATLAB] was used as the software tool to write functions to decode the data based on MG functions provided by the clinicians, and then concatenated using the MGM method to find the MG distributions (See Chapter 3: Methodology and Appendix B), and constructed MG decoding tables. (See Appendix D, Code).

The data was then processed through iARRIVE and results were mapped to the corresponding RI data nodes.

In total 264 Relative Influence (RI) values were calculated using iARRIVE (the required parameters for the GRIST tree), samples of which are shown in Table 6.2.

Table 6.2: Some RI codes and values as obtained through iARRIVE.

RI (Child Node Code)	Value	RI (Child Node Code)	Value
'sui-specific'	0.606124	'gen-risk-upbeat'	0.01168
'suic-past-att'	0.23874	'gen-coherence'	-0.11421
'suic-occur'	0.378099	'gen-body-face'	0.05991
'suic-most-rec'	0.723837	'gen-distrss-b-lang'	0.262224
'suic-patt-att'	0.276163	'gen-low-mood'	0.122345
'suic-first-occ'	0.030976	'gen-threat-move'	0.247395
'suic-how-many'	0	'gen-detached'	0.675741
'suic-escalate'	0.969024	'gen-eyes'	-0.3077
'suic-prep-serious-at'	0.403698	'gen-avoid-eye-contact'	1.158217
'suic-planning'	0.753293	'gen-eye-movement'	-0.15822
'suic-note-prev'	-0.1029	'gen-congruence'	0.707489
'suic-ser-method'	0.34961	'gen-eng-world'	0.091814
'suic-discovery'	0	'gen-phys-withd'	0.057365
'suic-lethality'	1	'gen-mental-withd'	0.942635
'suic-person-per'	0.218203	'gen-dep-stage'	0
'suic-thght-prev'	1.552644	'gen-ser-mentl-ill'	0.023203
'suic-ser-succd'	0.98832	'gen-mentl-insght'	1.943942
'suic-regret'	0.01168	'gen-mntl-cur-sympt'	-0.94394
'suic-leth-insght'	-0.55264	'gen-mania'	0.006366
'suic-curr-sit-behav'	0.576298	'gen-voice-hal'	0.797625
'suic-curr-int'	0.285192	'gen-voices-type'	0.030976
'suic-plans'	0.590171	'gen-voice-dang-s'	0.98832
'suic-plan-real'	0.723837	'gen-voice-dang-o'	0.01168
'suic-plan-dtail'	0.008555	'gen-prob-act-voice'	0.969024
'suic-steps-takn'	0	'gen-paranoid-del'	0.196009
'suic-prosp-leth'	0.267609	'gen-type-paranoid-del'	0.731781
'suic-int-inform'	0.474671	'gen-paran-del-spec'	0.98832
'suic-eol-prep'	-0.06484	'gen-paran-del-pers'	0.01168
'suic-int-p-trig'	0.271298	'gen-prob-act-par-del'	0.268219
'suic-pot-trig'	0.681865	'ment-fac'	0.060473
'suic-p-trig-mtch'	0.318135	'gen-impaired-cog'	-0.61372
'suic-ideation'	0.44351	'gen-cog-think-mem'	0.98832
'suic-id-control'	0.401986	'gen-concentr'	0.01168
'suic-id-hi-risk'	1.097995	'gen-learn-disab'	1.61372
'suic-id-freq'	-0.5339	'gen-personality'	-0.02257
'suic-id-strngth'	0.033918	'gen-assertive'	0.870153
'suic-bhvr-const'	0.332884	'gen-empathy-abil'	0.835195
'insight-resp'	0.568532	'gen-dependence'	0.097702
'suic-rel-belief'	0.431468	'gen-controlling'	1.853626
'suic-app-behvr'	-0.2576	'gen-coping-abil'	0.354964
'suic-phys-indic'	0.049272	'gen-hostile'	-2.32981
'sn-appearance'	0.554234	'gen-impulse'	-1.66238
'sn-hair-clothes'	0.723837	'gen-reliable'	0.980552
'sn-hygiene'	0.008555	'motive-eng'	-0.04439
'sn-recnt-app-chnge'	0	'gen-eng-world'	0.547588
'sn-skin'	0.267609	'gen-phys-withd'	0.98832
'gen-sh-cuts'	0.445766	'gen-mental-withd'	0.01168
'gen-presentation'	0.950728	'gen-motivation'	0

6.3.2 Data Cleaning

In an ideal world, all the patient records would be complete and the algorithm will use the data as it is to calculate the RI values. This however, is almost never the case, especially with pilot and test systems such as GRiST and in a complicated clinical environment. As part of the work, the following measures were used as guidelines for cleaning the data prior to the simulation process.

For a start, some of the patients records may be incomplete due to being unfinished, as the assessments could take several days or weeks. On the other hand, some records may be awaiting feedback or test results. These records can be called normal but incomplete. These will not be used in our sample pool.

The records that have data missing after the case has been closed can be referred to as incomplete. This causes a big problem, as the level of incompleteness varies from a few minor missing fields to vital data fields that will require eliminating the whole record from the test data pool. Two main types of incompleteness were identified in the data used in this trial: Major and minor.

In chapter 5, the APIC algorithm was presented as one way of dealing with missing data. This however, should not be applied on key data fields as the error margins could increase significantly and result in loss of accuracy. This is called *Major void*. These are fields that are crucial for the RI calculation process. If one of these fields is missing, the whole record is deemed invalid for this trial. An example of that would be the missing Risk Judgement, patient's date of birth and assessment date. These are all vital fields for the decoding process, and the iARRIVE, and if missing the record is considered invalid. Experts need to identify these fields and this information is then used in the cleaning process.

The number of missing fields in a record also represents a measure for how corrupt the record is. This is the second measure for corruption used, provided no Major corruption is present. It is referred to here as *Minor void*. This number is very subjective and will depend on the nature of the data and its sensitivity. This should be separately studied by the Clinical experts for each of the GRiST sub trees. This is out of the scope of this work. Basically, the more data missing, the less accurately you can predict them from the present data, and as a broad guide, any records with less than half of its data missing is considered invalid.

After these fields are identified, database queries had to be modified to return only the 'good' records. In this case, SQL was used. The data was then imported into MS Excel, and converted into the right CSV (Comma Separated Variables) format. The CSV file was used as the source for MatLab, representing the data matrix. Missing value algorithms (APIC) and Membership Grade (MG) defuzzification or decoding were also implemented in MatLab. This is presented in Appendix D: Code. The final stage, was the iARRIVE, which was implemented as part of the MatLab environment as well. MatLab [MATLAB] was chosen as the preferred mathematical tool, especially when dealing with complex matrix operations and regression, as in the case of this trial.

6.4 Results Analysis

In this section, analysis of the obtained results is presented, which was performed in spread sheet (MS Excel). We evaluate the performance of the iARRIVE algorithm on real data. This includes the error analysis of the Risk prediction system based on the newly obtained RI. For comparison the results are presented in terms of the actual judgement compared to the judgement obtained using the new RI parameters calculated by iARRIVE.

We have used some of the seed to test the results, i.e., using the calculated RI values to predict risk judgements for known cases from the seed. This is due to the fact that we need good or acceptable records for the testing. Being a hierarchical structure, any missing data affects the overall decision, and we need to use reasonable records for testing. The seed used for training was the best available, so it made sense to use it.

We used the ten fold cross validation on a data set of 700 records, divide into ten sets of 70 records each and cross validated. This was performed using MatLab software [MATLAB]. The results are shown in averages across the sets. The errors are then calculated and presented in graph formats.

The error is analysed as a Normal Distribution and some statistics are produced based on this assumption. This is a valid assumption and is widely used in statistical analysis of real life systems. [DIXO, 83]. This is used for illustration purposes and the actual distribution of the error could be the subject of future work.

Figure 6.1 shows the actual risk judgements on the X-axis plotted against the calculated judgements using iARRIVE on the Y-axis. Note that each actual risk has several calculated ones on the Y-axis. This is due to the fact that several actual cases could share the same risk value, but when estimated using the calculated RI parameters, they are bound to vary slightly due to errors and approximations.

The trend line on the graph shows how accurate is the overall representation of the GRiST system using the new calculated RI values in representing the original data, and thus the clinical expertise. The optimum would be at $y=x$, which would be at a 45 degree angle, with zero offset. This would only happen if the outcome exactly mirrors the inputs. In this trial, the trend line is trailing at $R^2=0.439$, which means the fit has a tendency towards linearity, and the predictions follow the original risks.

The errors in general are quite high, due to the small sample size and the fact that these test cases were assessed by more that one clinicians. This means that the original data itself is not consistent. However, even with these issues in mind, the errors are within acceptable limits as shown later.

Figure 6.2 shows the absolute error distribution in the sample. The X-axis represented the case numbers (patient records) and the Y-axis represents the absolute error in predicting that case, i.e., the difference between the predicted and the actual risk value. The horizontal trend line shows the average error overall, which stands at 1.178326. It is also clear from the graph that most errors are less than 2.

Figure 6.3 does the same as figure 6.2, except it uses Relative error instead. The average relative error sits at **0.420951**.

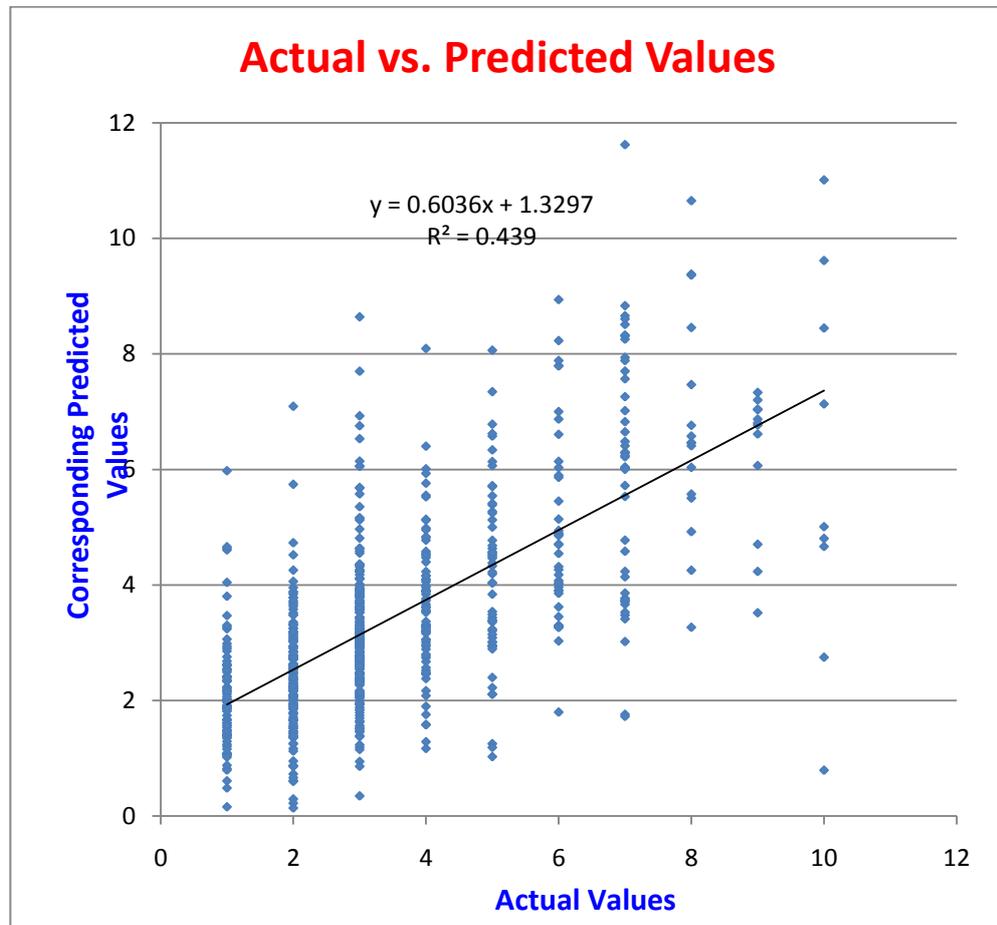


Figure 6.1 : Actual Risk Judgements vs calculated/ predicted judgements using iARRIVE.

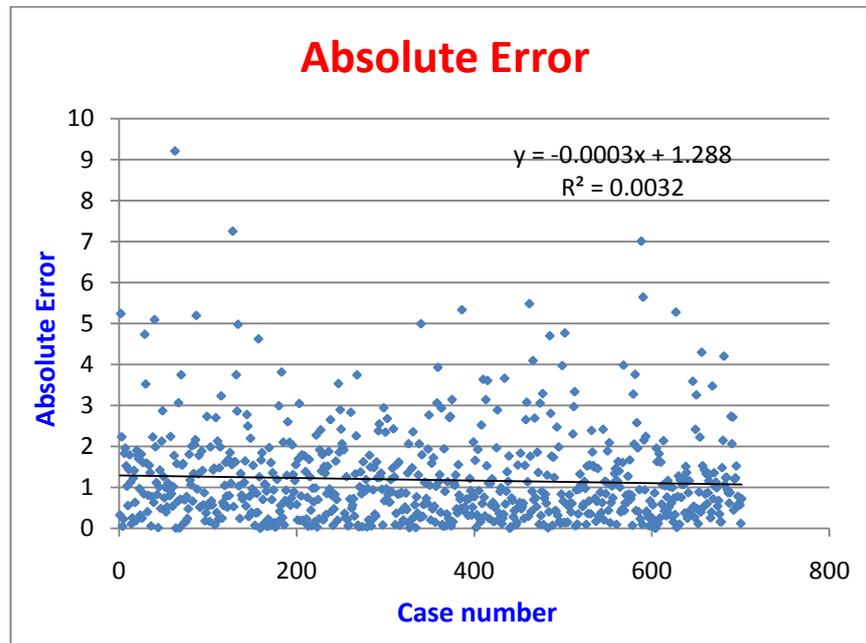


Figure 6.2: Absolute Error distribution across the training set. The line represents the average at 1.178326.

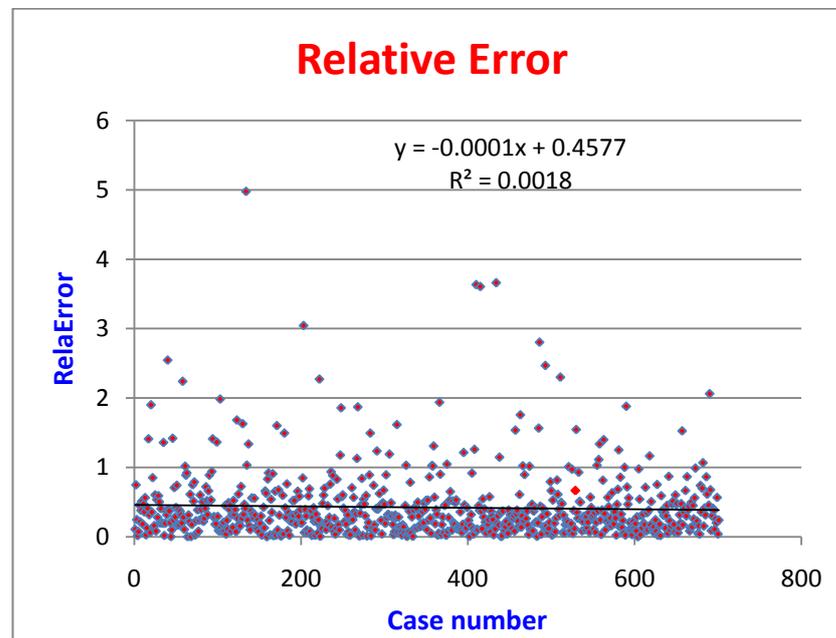


Figure 6.3: Relative Error distribution across the training set. The line represents the average at 0.420951

Figure 6.4 shows the absolute error distribution across the sample. For example, 27% of the predicted values have an absolute error between 0.5 and 1. In other words, they will deviate from the actual value by $\pm d$, where $0.5 < d < 1$.

Figure 6.5 summarizes figure 6.4, and shows that 84% of the predicted values have an absolute error of less than 2, and 16% have an absolute error of more than 2.

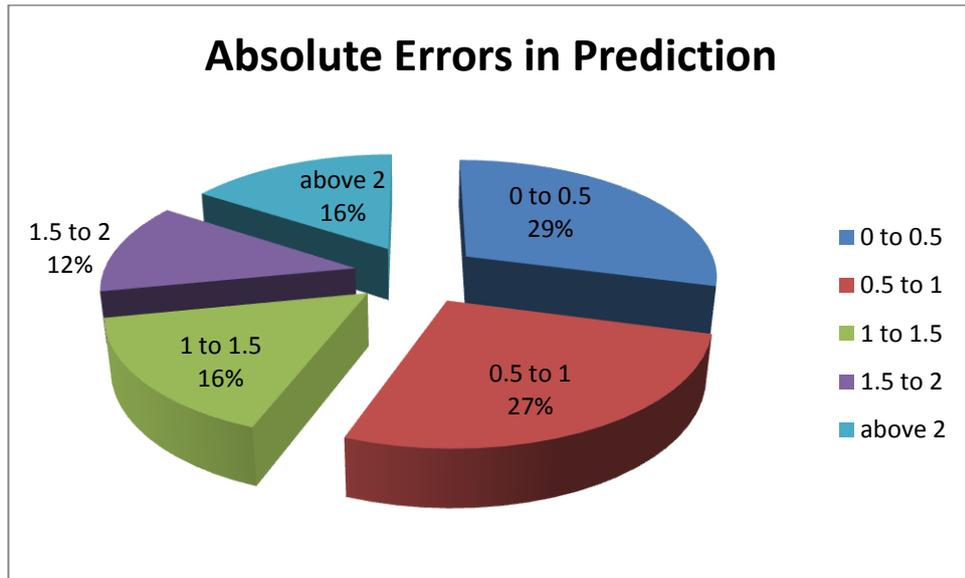


Figure 6.4: Absolute Error distribution across the test sample.

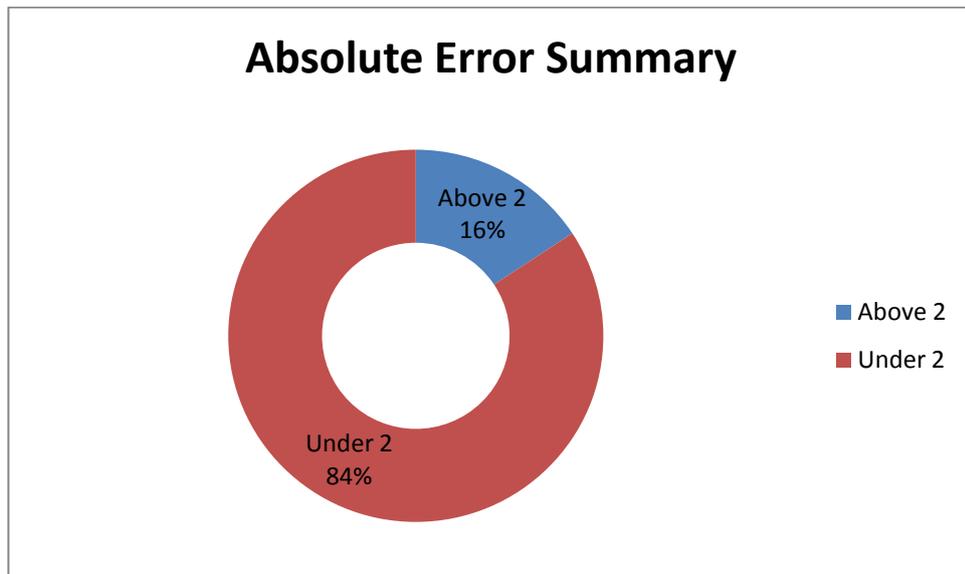


Figure 6.5: Distribution of absolute error in the training set, given in the % of the predicted values.

6.5 Error Distribution

As mentioned above, the Error is assumed to have a Normal Distribution. This is a valid assumption and is widely used in statistical analysis of real life systems. [DIXO,83]. In this case, the properties of a normal distribution can be used to analyse the results in a formal way. Figure 6.6 shows several Normal distributions with their associated parameters.

The function that describes any Normal Distribution is:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad (6.1)$$

where μ is the *mean* and σ^2 is the *variance*. The distribution with $\mu = 0$ and $\sigma^2 = 1$ is called the **standard normal**. [DIXO, 83]



Figure 6.6: Normal Distribution properties based on its parameters [WIKI, 11].

Figure 6.7 shows the actual distribution of error in this trial, based on the following parameters, which were obtained from the previous section based on analysing the results:

Parameter	Description	Value
μ	Average Error	1.178326
σ	Standard Deviation	1.114911
n	Sample size (Cross Validation, 10 fold)	700

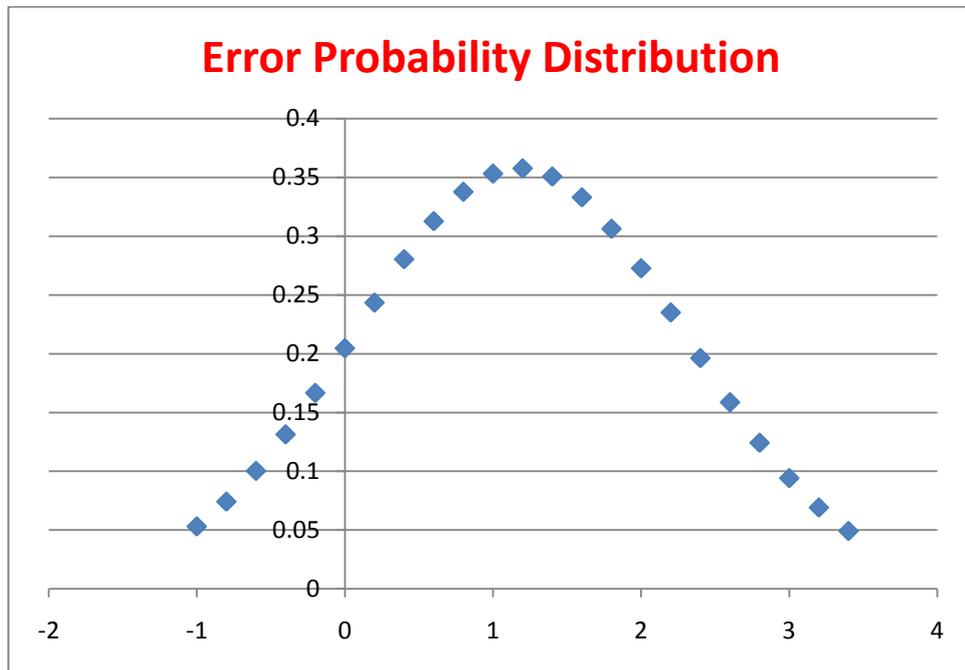


Figure 6.7: The Error represented in a Normal Distribution form.

Note that the average error is not zero, hence the graph in Figure 6.7 is shifted to the right. This is the standard graph for a normal distribution and therefore inherits all its properties.

From the properties of the Normal Distribution, about 68% of values drawn from a normal distribution are within one standard deviation σ away from the mean (Dark area in Figure 6.8, in the middle); about 95% of the values lie within two standard deviations (lighter area); and about 99.7% are within three standard deviations. This fact is known as the 68-95-99.7 rule, or the *empirical rule*, or the *3-sigma rule*. [EDWA, 74].



Figure 6.8: The probabilistic distribution of Data in a Normal Distribution [WIKI,11].

In this case, this means:

68.2 % Of the absolute errors will be between:
 (1.178326 + 1.114911) and (1.178326 - 1.114911)

Or:

2.293237 and 0.063414

And **95.4%** of the errors will be between: **3.408148 and -1.0515**

The above is illustrated in the charts in Figure 6.9.



Figure 6.9: Probable Percentage of errors in Confidence Intervals of 1*SD (right) and 2*SD (left). SD is Standard Deviation, in this case equals 1.114911.

This is acceptable in GRiST, as the classes of assessment judgement have a gap of 2 between them on the classification scale [BUCK,07], which starts at 0 (no risk) and ends at 10 (maximum risk). Risk is divided into five classes between 0 and 10, and each class occupies 2 partitions. See figure 6.10. This means that the new classifier is likely to produce a correct classification.

No Risk	Low Risk	Medium Risk	High Risk	Max Risk
0 to 2	2 to 4	4 to 6	6 to 8	8 to 10

Figure 6.10: The GRiST Risk classification scale.

Performance Analysis of GRiST:

It is important to put the results into perspective in the context of Clinical Decision making. As GRiST relies on classification of the risk into areas (as above), we need to assess the systems performance in terms of assessing the real life cases. The problem would arise if the clinician assesses a case as high risk, but GRiST produces a low or medium diagnosis. On the other hand, if the system gives a high risk diagnosis (value), and the clinician assesses the case as low risk, then there is considerable discrepancy. This section illustrates the system's performance on the test data out of this perspective. The problem arises when some cases produce an error greater than two and lie on the border between a high and medium risk, or low and medium (as in Figure 6.10). We call these critical or border line cases.

Total number of test cases: 700 Cross validated, 10-fold.

Mean Absolute Error: 1.178326

Total number of Cases with absolute Error above 2 : 110 (16%)

- Out of those:

A. Number of Cases Clinically above 7 but GRiST predicted under 7: 18
(2.6% of the total number of cases)

B. Number of Cases Clinically below 5 but GRiST predicted above 5: 21
(3% of total)

Cases A and B are demonstrated in Figure 6.11.

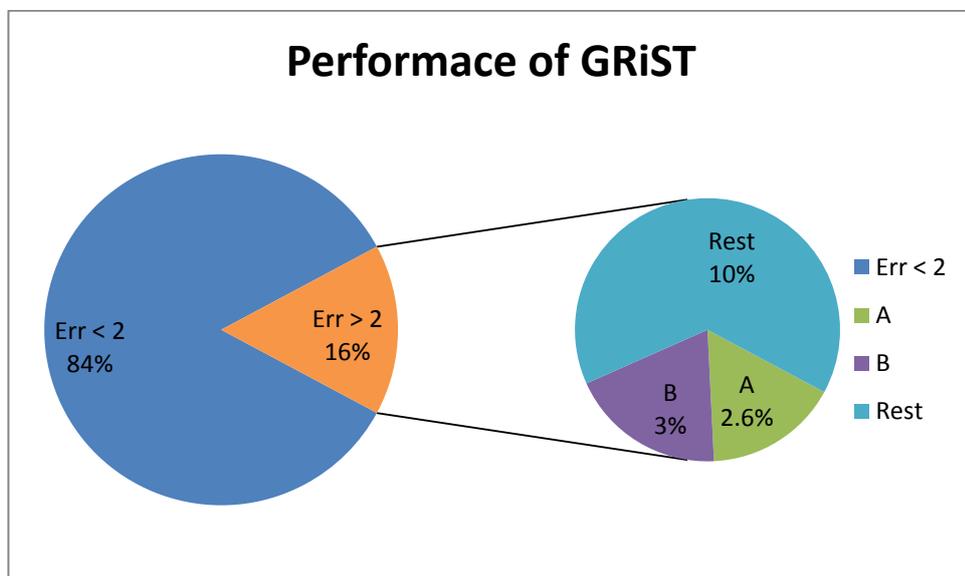


Figure 6.11: The system performance in real life situations, showing impact on clinicians decision.

6.6 Summary

This Chapter presented the results obtained from a system implementing our algorithms along with the error analysis associated with these results. The trial used 10,000 patient records that were pruned and cleaned to produce a satisfactory seed for the RI calculations. The results show that the classification error margins on the real patient data was acceptable and confirms that the iARRIVE method is a very promising solution for the parameter elicitation problem in a hierarchical knowledge structure such as the GRiST tree. The results were satisfactory despite the issues with the real data discussed in previous chapters, such as missing values, inaccurate inputs and the general inconsistency of the data. Analysis of the results in the case of GRiST shows the iARRIVE to be a suitable solution for the system. This is based on the classification requirements for GRiST. These thresholds and error margins may differ for other systems, but the algorithms we presented are flexible and adaptable enough to accommodate various adjustments as to suit the system requirements.

The incremental nature of the algorithm means new data can be easily used to adjust the parameters and reduce error margins on the long run.

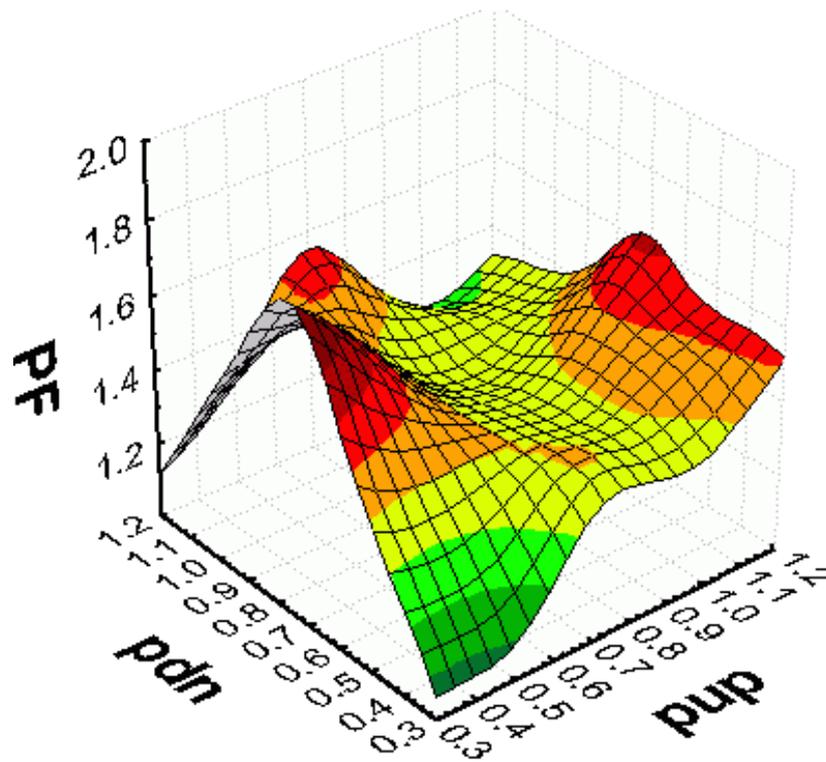
Our algorithms have been theoretically verified in Chapter 3, Methodology and in this chapter; we have tested them on real data obtained from the GRiST patient records. The system has also incorporated experts' opinions, as described in Chapter 4, Data Conditioning. This yields an extremely desirable and rare combination in Clinical Decision Support Systems (CDSS): expert opinions coupled with learning from the data. Our algorithms have achieved this.

The values of the errors have been discussed and were found within acceptable limits for our domain of application. It is however inevitable to have errors and in some cases medium to large, due to the fact that the data was supplied by more than one clinician, which makes the point of reference different for the different records, and as clinical judgement in mental health is a matter of opinion.



Chapter 7

Conclusions and Future Work



This Chapter covers the following:

- Conclusions
- Future Work

7.1 Conclusions

In this work, we have tackled the problem of parameter elicitation and knowledge representation in complex hierarchical structures. The application domain is Clinical Decision making, namely the GRiST Risk Screening Tool, which assesses the risk associated with mental-health problems. This is a very complex problem, in which the lack of data and information about the actual decision making process makes it subject to speculation. GRiST, however, is based on psychological analysis conducted over five years in collaboration with over 100 experts in the mental-health field. The actual data used for testing in this work, was supplied by 44 experts in our data input sessions.

GRiST models human expertise in a fuzzy-set system that has a multitude of parameters for weighting influences of branches in the tree. The research has explored ways of learning those weights from the data and explored the problems. The system combines automated parameter learning with human input to create the decision tree.

We have explored the problems associated with approximation, missing data and errors resulting from each scenario. This is inevitable in any real life system. It is particularly evident in the clinical decision making domain.

We have also applied some of our algorithms on real data, and we have developed our own software to gather and condition the data collected.

We have explored some of the challenges associated with real patient data, such as the quality of the data and the missing data problems. All the data is supplied by clinicians, which means human error is inevitable. We have explored ways of verifying the data and excluding the odd sets that are way off the trend. This means that the pre-processing stage is extremely important as the new more accurate data set produced from pre-processing will inevitably produce a better learning result than the original set provided by clinicians. We have presented methods and algorithms to do this. We call this the data cleaning stage.

The missing data problem was the second problem in the pre-processing stage. Many of the records provided by clinicians contained missing fields. This may have been due to errors in input, or unavailable data. This problem had to be dealt with as it

will severely affect the results of the RI calculations and could lead to an unsolvable problem. We have presented algorithms and methods to deal with this specific problem.

We also considered the advantages of exploring the results of RI elicitation in re-shaping the tree and streamlining the model. Some of the tree branches may become redundant or obsolete and others may rise in importance. This will provide an important insight into the tree structure and dynamically improve the representation of expertise using the actual data. This is very powerful as in the case of GRiST for example; the tree has over 250 different nodes. In our test data, we used a subtree with 184 leaf nodes.

We have published four IEEE papers based on this work and one journal paper. One of our papers [HEGA, 08] won the Best Paper award at the conference.

7.2 Future Work

In this work, we have presented a way of eliciting parameters based on the GRiST decision support system. The methods and algorithms presented here however, can be easily adopted for other knowledge engineering domains. Future work may include extended error analysis of the elicitation and consensus algorithms, exploring the possibility of combining probabilities into the system, and the effect of having more data on the overall convergence process and error margins. Using probabilities would extend the system's ability to model more complex data and thus may give a better understanding of the parameter elicitation process in real life.

Another possible field would be exploring the relationships within the tree and eliminating redundancy automatically. This could lead into self organizing GRiST trees and the introduction of formal optimization parameters that can be used to judge the quality of a certain tree or combination. For example, using the RI values, and the standard deviation of each or a group of values in a sub tree, could indicate the quality of the subtree. If the deviation is high, that could mean that that subtree requires further analysis or pruning. In some cases, a tree could end up as a branch, with one node, which eliminates the rest of the nodes due to their insignificance.

Some of the mathematical and statistical methods used in our algorithms and methods could be extended. This could mean further error analysis and optimization of the results. For example using higher order non-linear models for regression in our APIC algorithm could be one avenue. This might produce more accurate results if the original data was non-linear. To identify the type of model more suited for the data, more statistical analysis on the data need to be conducted.

A further possible area for research could be further exploring the semantic representation of the results and linking back to the original model and clinical expertise which will be a form of self-evaluation of GRiST. This would be evident in our MGM algorithm for obtaining meaningful consensus while adhering to the semantic restriction posed by the clinicians in terms of minima and maxima of the inputs.

The error calculations we presented in some areas of this work could be explored in more detail. Comprehensive error analysis would be very beneficial to assess not only the full potential of this work, but to determine the quality of the original data provided by clinicians. This could be used as an assessment tool to check the quality and consistency of the clinical data provided by clinicians. Any inconsistency can then be challenged and fed back to the clinicians and ultimately to the system. This would create a system life cycle, and the system will learn and feedback which makes it a dynamic and adaptable system.



References



References are provided in **[ABCD, YY]** format, where “ABCD” represents the first four letters of the main author’s surname, “YY” is the year of publication. For example, [BUCK, 08], represents a reference by “C.D.Buckingham” as the main author, published in 2008.

- [AHO, 74] Aho, Alfred V. (Alfred Vaino); The design and analysis of computer algorithms, *Reading, Mass. ; London (etc.) : Addison-Wesley, 1974.*
- [ANAH, 97] Anahory, Sam. Harlow; Data warehousing in the real world: a practical guide for building decision support systems, *Addison-Wesley, 1997.*
- [ARBO, 07] Ann Arbor, "Arden Syntax for Medical Logic Systems Standard Version 2.6". MI: Health Level 7; 2007.
- [ATKI, 89] Atkinson, Kendall A.; An Introduction to Numerical Analysis (2nd ed.), New York: John Wiley & Sons, ISBN 978-0-471-50023-0, 1989.
- [BAES, 03] Anke Meyer-Baese, *Pattern Recognition and Signal Analysis in Medical Imaging*, Academic Press; 1st edition , 2003, ISBN-10: 0124932908.
- [BARR, 12] Barros, Rodrigo C., Basgalupp, M. P., Carvalho, A. C. P. L. F., Freitas, Alex A. (2011). A Survey of Evolutionary Algorithms for Decision-Tree Induction. *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews*, vol. 42, n. 3, p. 291-312, May 2012.
- [BEMM, 97] van Bommel and Musen; *Handbook of Medical Informatics*, Springer; 1st ed. 1997. ISBN-10: 3540633510.
- [BERN, 07] Berner, Eta S.(2007). *Clinical Decision Support Systems*. New York, NY: Springer, 2007.
- [BERN, 99] S. Berner; *Clinical decision support systems: theory and practice*, New York ; London : Springer, c1999.
- [BOCA, 03] Boca Raton, Fla.; *Statistical data mining and knowledge discovery*, London : Chapman & Hall/CRC, 2003.
- [BODI, 85] Bodily, Samuel E.; *Modern decision making: a guide in modelling with decision support Systems*, New York ; London : McGraw-Hill, c1985
- [BRAT, 03] Ivan Bratko, Dorian Šuc. Learning qualitative models Published in *AI Magazine*, 2003, vol. 24, no. 4, pp. 107-119, © AAAI Press
- [BREI, 84] [12] Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software.
- [BREI, 84] Breiman, Leo; Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software. ISBN 978-0412048418.
- [BRUC, 72] Bruckheimer, Maxim. *Correlation*. London: Chatto and Windus, 1972.
- [BUCH, 82] B.G. Buchanan, "Research on Expert Systems," in J.E. Hayes, D. xichie, and Y.H. Pao (eds.), *Machine Intelligence 10*, :New York: John Wiley, 1982.

- [BUCH, 84] Buchanan, B.G. and Shortliffe, E.H. (eds). Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project. *Reading, Addison-Wesley, 1984.*
- [BUCK, 02] Buckingham, C.D. (2002). Psychological cue use and implications for a clinical decision support system. *Medical Informatics and the Internet in Medicine, 27(4), 237-251.*
- [BUCK, 07] Buckingham, C.D., Ahmed, A., & Adams, A.E. (2007). Using XML and XSLT for flexible elicitation of mental-health risk knowledge. *Medical Informatics and the Internet in Medicine, 32(1), 65-81.*
- [BUCK, 07B] [5] Buckingham, C.D. (2007). Improving mental health risk assessment using web-based decision support. *Health Care Risk Report, 13(3), 17-18.*
- [BUCK, 08] Buckingham, C. D., Adams, A.E. & Mace, C. (2008). Cues and knowledge structures used by mental-health professionals when making risk assessments. *Journal of Mental Health.*
- [BURK, 97] [18] Burke, L., Ignizio, JP. (1997). *A practical overview of neural networks, JOURNAL OF INTELLIGENT MANUFACTURING 8 (3): 157-165 JUN 1997.*
- [CAMO, 08] The Unscrambler® 9.7, <http://www.camo.com/rt/Products/Unscrambler/unscrambler.html>, 2008.
- [CAMP, 97] [1] Campbell-Kelly, Martin and Aspray, William; Computer: A History of the Information Machine by *Basic Books, 1997.*
- [CHAP, 00] Gretchen B. Chapman and Frank A. Sonnenberg; Decision making in health care : theory, psychology, and applications, *Cambridge : Cambridge University Press, 2000.*
- [CHAT, 77] Chatterjee, Samprit; Regression analysis by example, *New York ; London (etc.) : Wiley, 1977.*
- [CHAT, 98] Chatellier, G., Colombet, I., Degoulet, P. (1998). An overview of the effect of computer-assisted management of anticoagulant therapy on the quality of anticoagulation. *Int J Med Inf; 49: 311-20, 1998.*
- [CLAN, 84] Clancey, W.J. and Shortliffe, E.H. (eds). Readings in Medical Artificial Intelligence: The First Decade. *Reading, MA: Addison-Wesley, 1984.*
- [COIE, 03] E. Coiera. The Guide to Health Informatics (2nd Edition). *Arnold, London, October 2003.*
- [COIE, 94] E. Coiera, Question the Assumptions: Knowledge and Decisions in Health Telematics - The Next Decade, *IOS Press,Amsterdam, 1994, 61-66.*
- [COIE, 97] Coiera, E.; A Guide to medical informatics, the internet and Telemedicine. *London: Chapman and Hall, 1997.*

- [DIXO, 83] Dixon, W. J., Introduction to statistical analysis, New York ; London : McGraw-Hill, c1983. 4th ed.
- [DOMB, 93] F. T. De Dombal; Surgical Decision Making, *Butterworth-Heinemann* , ISBN: 0750607041, 1993.
- [DRAP, 98] Draper, N. R.; Applied regression analysis, *New York ; Chichester : Wiley, c1998, 3rd ed.*
- [DRUC, 96] Harris Drucker and Corinna Cortes. (1996). Boosting decision trees, In *Advances in Neural Information Processing Systems 8*, pages 479-485, 1996.
- [DYBO, 01] Richard Dybowski and Vanya Gant; Clinical applications of artificial neural networks, *Cambridge University Press, 2001.*
- [EBEL, 97] Ebell, M., Gaspar, D., Khurana, S. (1997). Family physicians' preferences for computerised decision-support hardware and software. *J Fam Pract; 45: 137-41, 1997.*
- [EDWA, 74] Edwards, Allen Louis; Statistical analysis, *New York ; London (etc.) : Holt, Rinehart and Winston, 1974, 4th ed.*
- [EDWA, 76] Edwards, Allen Louis. An introduction to linear regression and Correlation. *San Francisco : W. H. Freeman, c1976.*
- [EFRA, 08] Efraim Turban, Jay E. Aronson, Ting-Peng Liang; Decision Support Systems and Intelligent Systems. *p. 574, 2008.*
- [ELSO, 95] Elson, R., Connelly, D.; Computerised decision support systems in primary care. *Prim Care; 22: 365-84, 1995.*
- [FALK, 97] Falkenauer, Emanuel; Genetic Algorithms and Grouping Problems, *Chichester, England: John Wiley & Sons Ltd. ISBN 978-0-471-97150-4, 1997.*
- [FOX, 02] Fox, J., Thomson, R. (2002). Clinical decision support systems: a discussion of quality, safety and legal liability issues. *Proc AMIA Symp, 265-9.*
- [FRAS, 70] Fraser, Alex; Donald Burnell; Computer Models in Genetics. *New York: McGraw-Hill, 1970.*
- [GADEW, 10] Jyotirmay Gadewadikar, Ognjen Kuljaca, Kwabena Agyepong, Erol Sarigul, Yufeng Zheng and Ping Zhang, Exploring Bayesian networks for medical decision support in breast cancer detection , *African Journal of Mathematics and Computer Science Research, Accepted 13 September, 2010.*
- [GALU, 07] Galushkin, A. I.; Neural networks theory, *Berlin, New York : Springer, 2007.*
- [GARD, 98] Gardner RM, Pryor TA, Warner HR. (1998). The HELP hospital information system: update, 1998. *Int J Med Inf. 1999 Jun;54(3):169-82, 1998.*

[GIAQ, 03] Mariano Giaquinta and Giuseppe Modica, *Mathematical Analysis. Vol. 1, Functions of one variable: Boston, Mass.: Birkhäuser ; Berlin : Springer, 2003.*

[GIGE, 96] Gigerenzer, G. and Goldstein, D.; Reasoning the Fast and Frugal Way: Models of Bounded Rationality, *Psychological Review* Copyright 1996 by the American Psychological Association, Inc. , Vol. 103. No. 4, 650-669, 1996.

[GOLD, 89] Goldberg, David E; Genetic Algorithms in Search, Optimization and Machine Learning, *Kluwer Academic Publishers, Boston, MA, 1989.*

[GOVI, 06] N.K. Govil, "Frontiers in interpolation and approximation," *Chapman & Hall/CRC, London, 2006.*

[GRAY, 01] Gray, J.A. Muir; Evidence-based healthcare, *Edinburgh: Churchill Livingstone, 2001.*

[GREE, 07] Robert A. Greenes; Clinical decision support: the road ahead, *Amsterdam ; London : Elsevier, c2007.*

[GRIST,11] See www.galassify.org/grist , last visited May 2011.

[HAWL, 06] Hawley, C. J., Littlechild, B., Sivakumaran, T., Sender, H., Gale, T. M. & Wilson, K. J. (2006). Structure and content of risk assessment proformas in mental healthcare. *Journal of Mental Health, 15*, 437-448.

[HEBD, 98] Hebda, T., Czar, P., Mascava, C.; Handbook of informatics for nurses and health care professionals. *Menlo Park, CA: Addison-Wesley, 1988.*

[HEEK, 06] Heeks, R. (2006). Health Information Systems: failure, success and improvisation. *International Journal of Medical Informatics, 2006*, 125-137.

[HEGA, 08] Hegazy, S. E. and Buckingham C. D., "An Algorithm for Robust Relative Influence Values Elicitation (ARRIVE)," *Proceedings of the 3rd International Conference on Computing in the Global Information Technology (ICCGI 2008)*, , Athens, Greece, 2008, pp. 91- 96.

[HEGA, 09] Hegazy, S. E. and Buckingham C. D. (2009), "An Incremental Algorithm for Robust Relative Influence Values Elicitation (iARRIVE)," *Proceedings of eTelemed, IARIA, Cancun, Mexico, 2009*, pp. 239-244.

[HEIS, 75] Heise, David R.; Causal analysis, *New York ; London (etc.) : Wiley-Interscience, 1975.*

[HENK, 87] Henk G. Sol et al. (1985). Expert systems and artificial intelligence in decision support systems, *proceedings of the Second Mini Euroconference, Lunteren, The Netherlands, 17-20 November, 1985. Springer, 1987. ISBN 9027724377. p.1-2.*

[HARR, 72] Horrocks JC, McCann AP, Staniland JR, Leaper DJ, de Dombal F.T.; Computer-aided diagnosis: Description of an adaptable system and operational experience with 2,034 cases. *BMJ 1972.*

[JACK, 75] Jackson, M.A.; Principles of Program Design, *Academic Press, 1975.*

- [JANI, 96] C. Z. Janikow. (1996). "Exemplar learning in fuzzy decision trees," *Proc. FUZZIEEE*, pp. 1500-1505, 1996.
- [JENS, 07] Jensen, Finn V.; Bayesian networks and decision graphs. *Berlin: Springer, 2007. 2nd.*
- [JENS, 07] Jensen, Finn V; Nielsen, Thomas D. (June 6, 2007). *Bayesian Networks and Decision Graphs*. Information Science and Statistics series (2nd ed.). New York: Springer-Verlag. ISBN 978-0-387-68281-5.
- [JENS, 96] Jensen, Finn V.; An introduction to Bayesian networks. *London : UCL Press, 1996.*
- [KANA, 74] Kanal; Patterns in pattern recognition: 1968-1974, Information Theory, IEEE Transactions on *Issue Date: Nov 1974 Volume: 20 Issue: 6 On page(s): 697 - 722 ISSN: 0018-9448.*
- [KEEN, 78] Keen, P. G. W.; Decision support systems: an organizational perspective, *Reading, Mass., Addison-Wesley Pub. Co. ISBN 0-201-03667-3, 1978.*
- [KLEI, 93] G. Klein, J. Orasanu, Decision Making in Action: Models and methods, *Ablex Publishing Corp., Norwood. 1993.*
- [KLEI, 98] David G. Kleinbaum; "Applied regression analysis and other multivariable methods," *Pacific Grove : Duxbury Press, c1998, 3rd ed.*
- [KOLO, 93] Kolodner, Janet L.; Case-based reasoning, *San Mateo, Calif. : Morgan Kaufmann, 1993.*
- [KORV, 93] Korver, M., Lucas, P. (1993). Converting a rule-based system into a belief network. *Med Inform (Lond); 18: 219-41, 1993.*
- [LANC, 86] Peter Lancaster and Kestutis Salkauskas; Curve and surface fitting : an introduction, *London : Academic, 1986.*
- [LARO, 06] Larose, Daniel T.; Data mining methods and models, *Hoboken, NJ : Wiley-Interscience, c2006.*
- [LARO, 06] Larose, Daniel T., Data mining methods and models , *Hoboken, NJ : Wiley-Interscience, c2006.*
- [LEDL, 59] Ledley R.S., Lusted L.B. (1959) Reasoning foundation of medical diagnosis, *Science 130(3366) :9-21, 1959.*
- [LINN, 93] Linnarsson, R. (1993). Decision support for drug prescription integrated with computer-based patient records in primary care. *Med Inform (Lond); 18: 131-42, 1993.*
- [MATLAB] Matlab Software Suite, www.mathworks.com
- [MICH, 93] Michie, D. (1993) Knowledge, learning and machine intelligence. In: *Intelligent Systems* (ed. L. Sterling), pp. 2-19. New York

- [MILL, 89] Miller RA, Masarie FE Jr. (1989). Use of the Quick Medical Reference (QMR) program as a tool for medical education, *Methods, Inf Med.* 1989 Nov;28(4):340-5. PMID: 2695783.
- [MILL, 90] Miller, P., Sittig, D. (1990) The evaluation of clinical decision support systems: what is necessary versus what is interesting. *Med Inform (Lond)*; 15: 185-90, 1990.
- [MONT, 06] Montgomery, Douglas C.; Introduction to linear regression analysis, New York ; Chichester : Wiley-Interscience, 2006, 4th ed.
- [MORR,98] JG Morrison, Kelly, R.T., Moore, R. A., Hutchins, H. G. (1998) ; Implications of decision-making research for decision support and displays.
- [MUSE, 99] Musen MA. (1999). Stanford Medical Informatics: uncommon research, common goals. *MD Comput.* 1999, Jan-Feb;16(1):47-8, 50.
- [NADK, 07] Nadkarni P, Miller R.: "Service-oriented Architecture in Medical Software: Promises and Perils." *Journal of American Medical Informatics Association.* 2007 Mar-Apr;14(2).
- [NEAP, 04] Neapolitan, Richard E.; Learning Bayesian networks, Harlow : Prentice Hall, 2004.
- [NLREG] NLREG, <http://www.linearregression.com/>
- [OHNO, 98] Ohno-Machado L, Gennari JH, Murphy SN, et al.: "The guideline interchange format: a model for representing guidelines," *Journal of American Medical Informatics Association*, 1998 Jul-Aug;5(4):357-372.
- [OPEN, 09] "Decision support system", Feb. 2009. <http://www.openclinical.org/dss.html>.
- [OPEN, 85] Modelling with mathematics: an introduction, Milton Keynes: Open University Press, 1985.
- [PELE, 97] Peleska, J., Svejda, D., Zvarova, J. (1997). Computer supported decision making in therapy of arterial hypertension. *Int J Med Inf*; 45: 25-9, 1997.
- [PERRE, 99] Perreault L, Metzger J. A. (1999). pragmatic framework for understanding clinical decision support. *Journal of Healthcare Information Management*, 1999;13(2):5-21.
- [PLAC, 60] Plackett, R. L.; Principles of regression analysis, Clarendon P, 1960.
- [PRUV, 98] Purves, I. (1998). PRODIGY: implementing clinical guidance using computers. *Br J Gen Pract*; 48: 1552-3, 1998.
- [QUIN, 75] J. R. Quinlan (1975). *Machine Learning*, vol. 1.

- [QUIN, 86] J.R. Quinlan (1986). Induction of Decision Trees, *Machine Learning*, (1), 81-106
- [QUIN, 89] J. Ross Quinlan. (1989). Applications of expert systems, *Vol.2, the proceedings of the third and fourth Australian conferences 1989 Sydney ; Wokingham : Turing Institute Press in association with Addison-Wesley.*
- [QUIN,92] Quinlan; C4.5: Programs for Machine Learning, *Morgan Kaufmann, 1992.*
- [RAM, 04] Ram P, Berg D, Tu S, et al.: "Executing clinical practice guidelines using the SAGE execution engine." *Medinfo. 2004;11(Pt 1):251–255.*
- [RAMN, 03] Ramnarayan, P., Tomlinson, A., Rao, A., Coren, M., Winrow, A., & Britto, J. (1993). ISABEL: A web-based differential diagnostic aid for paediatrics: Results from an initial performance evaluation. *Archives of Disease in Childhood*, 88(5), 408-413, 2003.
- [SHEL, 72] Sheldon, James. "Applied multivariate analysis," New York ; London : *Holt, Rinehart and Winston, 1972.*
- [SHOR, 00] Shortliffe, E.H. and Perreault, L. (eds), Wiederhold, G., and Fagan, L.M. (assoc. eds.). *Medical Informatics: Computer Applications in Health Care and Biomedicine. New York: Springer-Verlag, 2000. (2nd edition of #4, new publisher and title).*
- [SHOR, 01] Edward H. Shortliffe, Leslie E. Perreault; *Medical informatics: computer applications in health care and biomedicine, New York, N.Y. ; London : Springer, c2001. 2nd ed.*
- [SHOR, 06] Shortliffe, E.H. (ed) and Cimino, J.J. (assoc. ed.). *Biomedical Informatics: Computer Applications in Health Care and Biomedicine. New York: Springer-Verlag, 2006. (3rd edition of #5).*
- [SHOR, 76] Shortliffe, E.H. *Computer-Based Medical Consultations: MYCIN, Elsevier/North Holland, New York, 1976.*
- [SHOR, 79] E.H. Shortliffe, B.C. Buchanan, and E.A. Feigenbaum. (1979). "Knowledge Engineering for Medical Decision Making: A Review of Computer-Based Clinical Decision Aids," *Proceedings of the IEEE*, 67, 1207-1224, 1979.
- [SHOR, 81] E. H. Shortliffe, A. C. Scott, M. B. Bischoff, A. B. Campbell, W. V. Melle, C. D. Jacobs. (1981). ONCOCIN: An expert system for oncology protocol management, *Seventh International Joint Conference on Artificial Intelligence, Vancouver, B.C.. Published in 1981.*
- [SHOR, 90] Shortliffe, E.H. and Perreault, L., (eds), Wiederhold, G., and Fagan, L.M. (assoc. eds.). *Medical Informatics: Computer Applications in Health Care. Reading, MA: Addison-Wesley, 1990.*
- [SIMO, 07] Simon Haykin , *Neural networks : a comprehensive foundation*, 3rd ed., Harlow : Prentice Hall, 2007.

- [SPRA, 93] Sprague, Ralph H.; Decision support systems: putting theory into practice , *Prentice-Hall, 1993. 3rd edition.*
- [TAN, 05] Tan, Pang-Ning; Introduction to data mining ,*Pearson, 2005.*
- [TANG, 07] Tang, Huajin; Neural networks: computational models and applications , *Berlin : Springer, c2007.*
- [THOM, 10] InfoButton Access™ from Thomson Reuters, *http://www.micromedex.com/products/infobutton/*
- [TURB, 05] Turban, Efraim; Aronson, Jay E. and Liang , Ting-Peng; Decision support systems and intelligent systems. *Upper Saddle River, N.J. : Prentice Hall, c2005. 7th edition.*
- [VADH, 97] Vadher, B., Patterson, D., Leaning, M. (1997). Evaluation of a decision support system for initiation and control of oral anticoagulation in a randomised trial. *BMJ; 314: 1252-6, 1997.*
- [WARN, 94] Warner HR, Bouhaddou O. (1994). "Innovation review: Iliad--a medical diagnostic support program.". *Top Health Inf Manage 14 (4): 51-8, 1994.*
- [WARN, 97] Warner, H., Sorenson, D., Bouhaddou, O.; Knowledge engineering in health informatics. *New York: Springer, 1997.*
- [WIKI, 11] www.Wikipedia.org, the online encyclopaedia, last viewed : August 2011.
- [WRIG, 08] Wright A, Sittig DF: "A four-phase model of the evolution of clinical decision support architectures", *International Journal for Medical Informatics, 2008 October; 77(10): 641-649.*
- [WRIG, 08b] Wright A, Sittig DF.: "SANDS: a service-oriented architecture for clinical decision support in a National Health Information Network", *Journal of Biomed Informatics 2008 Dec;41(6):962-81. Epub 2008 Mar 14.*
- [WYAT, 91] Wyatt J, Spiegelhalter D. (1991). Evaluating medical expert systems: what to test, and how? *Knowledge-based systems in medicine: methods, applications and evaluation. Heidelberg: Springer Verlag, 1991: 274-290.*
- [WYAT, 92] Wyatt J, Spiegelhalter D. And Evans D, Patel V. (1992). The evaluation of medical expert systems, *Advanced models of cognition for medical training and practice. MIT Press, 1992 (NATO ASI series F97): 101-120.*
- [ZADE, 65] Zadeh, L. A.: Fuzzy sets. *Information Control 1965;8:338-53.*



Appendix A

GRiST Tree Structure



In this section, we present parts of the detailed structure of the GRiST tree, [BUCK, 07]. The aim is to show the complicated semantics and structure that the system has

to elicit parameters for. This work is concerned with the abstract representation of the tree, hence no further details were included.

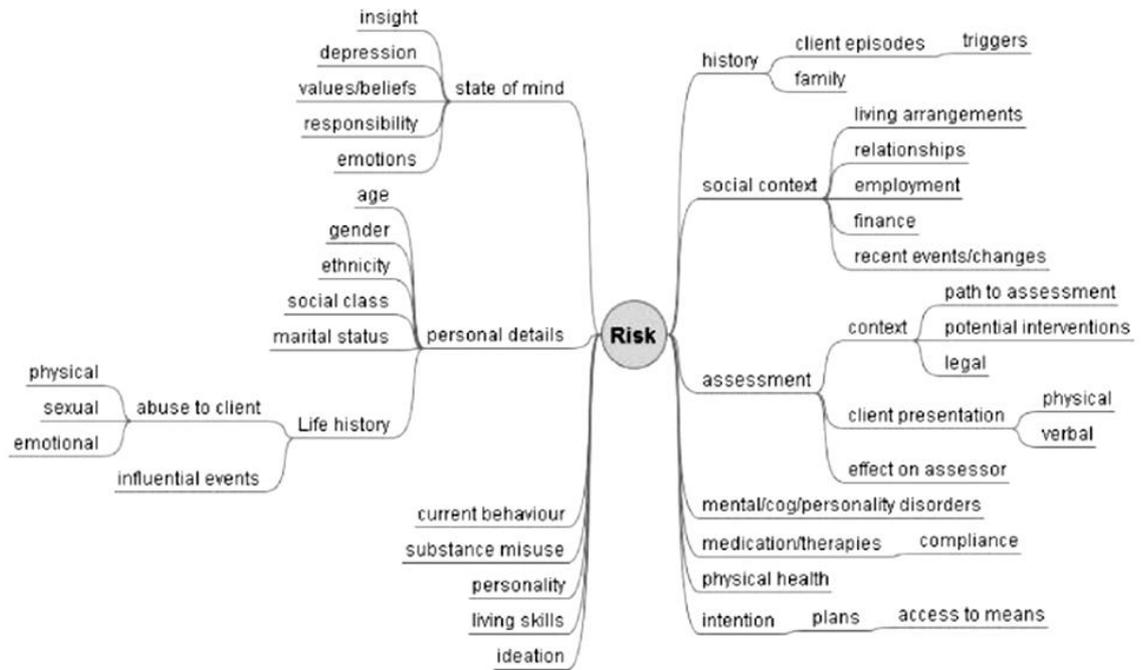


Figure 1 : Mind map framework used to code risk knowledge from individual interviews.



Figure 2. Part of the fully-expanded “pattern of episodes” concept within suicide risk of the combined map. Numbers after node names represent the identification numbers of different experts who mentioned the node.

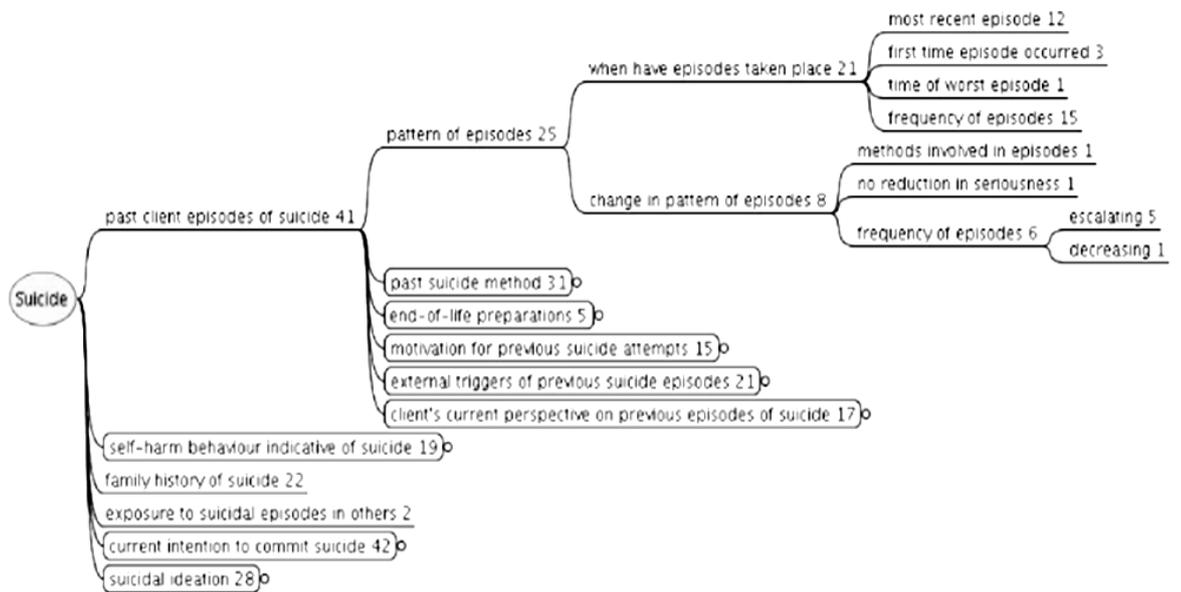


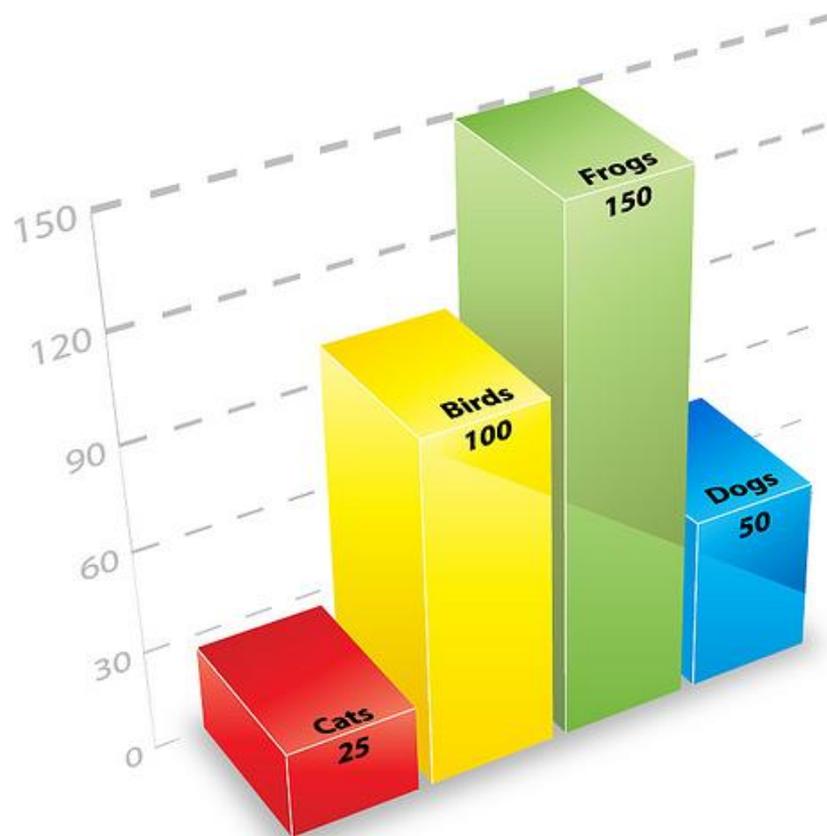
Figure 3. Part of the combined-coding mind map for suicide risk, where the numbers after the node names represent the total number of different experts who mentioned that node or any of its subcomponents (nodes within rounded rectangles are concepts without their internal structure displayed).

Immediate subcomponents of risks	Number of experts				SUM
	SUI	S-H	HTO	S-N	
Total experts	46	44	46	31	46
1 past episodes of risk	41	36	37	0	46
2 family history of risk	22	9	4	0	23
3 life history	25	21	18	1	34
3.1 abuse to the patient	14	14	8	0	24
4 current intention to effect risk	42	28	25	0	46
5 ideation about risk	28	13	15	0	33
6 hallucinations and delusions	10	2	20	4	26
6.1 auditory hallucinations	5	2	11	1	16
6.2 paranoid delusions	4	0	13	3	15
7 appearance and behaviour during assessment	36	19	25	26	42
8 living skills	1	0	0	10	11
9 current general behaviour	16	9	17	19	32
9.1 diet	1	1	0	18	19
9.2 inappropriate sexual, criminal, or abusive behaviour	1	1	8	0	8
10 substance misuse	34	22	35	9	43
11 constraints on effecting risk	18	4	8	0	25
11.1 responsibility for impact on others	15	2	1	0	17
12 feelings/emotions	36	22	19	2	42
12.1 angry emotions	2	1	15	0	16
12.2 hopelessness	27	11	1	0	29
13 depression	45	19	13	17	46
14 mental illness	45	25	31	14	45
14.1 insight into the illness	9	2	1	0	9
14.2 stage of illness	12	2	2	0	13
14.3 bipolar	15	7	14	2	22
14.3.1 manic/hypomanic phase	8	4	13	2	16
14.4 psychosis	30	7	23	3	35
14.5 schizophrenia	25	5	10	2	28
14.5.1 insight into schizophrenia	13	1	0	0	13
15 attitude	32	18	16	8	39
16 personality	19	13	21	3	29
16.1 impulsiveness	16	7	10	0	21
17 personality disorder	5	12	5	0	15
17.1 borderline personality disorder	2	7	1	0	7
18 cognitive faculties	10	3	7	3	15
19 social context	42	21	17	21	45
19.1 who client lives with	16	5	2	3	22
20 relationships	41	17	20	7	46
20.1 changes to relationships	27	11	1	1	31
21 physical health problems	25	5	0	9	29
21.1 pain	9	1	0	0	9
21.2 chronic health problems	21	2	0	3	22
22 medication/therapies	19	8	10	4	31
22.1 response	10	0	0	0	10
22.2 compliance	6	2	7	3	17
23 demographics	31	17	20	7	43

Table I. Numbers of experts citing top-level risk components.

Appendix B

Results



MGM Consensus Analysis

In this section, we list some of the results we obtained from the MGM algorithm, applied to all MG, based on data supplied by 44 Clinicians using the online elicitation tool.

We have opted for third order regression, as it best represented the set, although the analysis were conducted on the data using order one through five.

In some cases the resulting MGM seems like a straight line, but this is due to the small variations in the original data. We left it as it is to show that we have not approximated the MGM to a straight line, but it was the actual result of the algorithm.

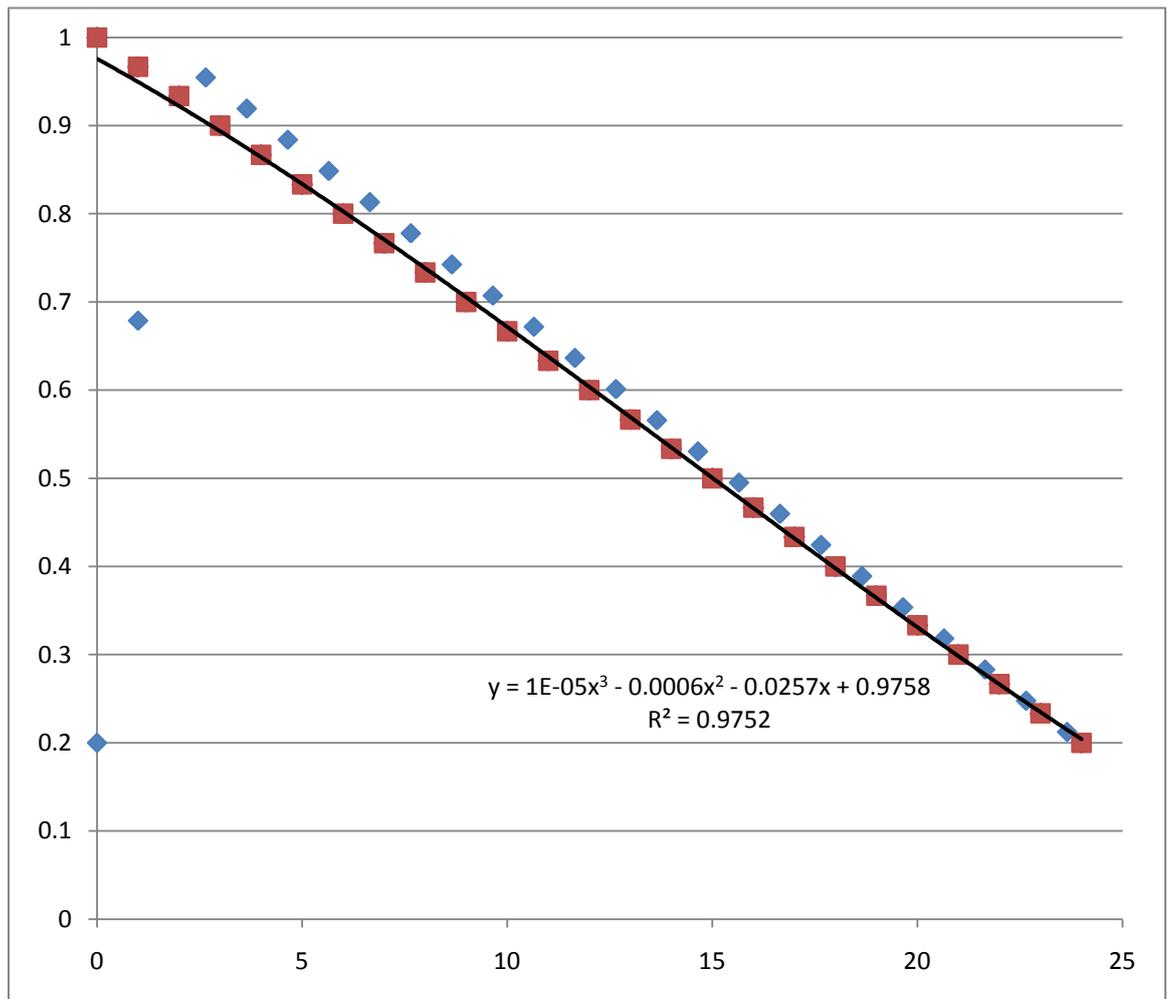
As the original graphs supplied by our online tool were saved as a collection of lines, this posed a problem for the regression operation. The lines were described only by their start and end points, with no actual equations. This meant that each expert graph could have different number of points depending on the number of lines they used to describe the MG graph. This makes it difficult to store in our database too. So the tool had to interpolate the lines and generated the line equations, then resample each graph at a set step (and in turn frequency) to get the same number of sampled points per expert for the same MG. This ensures that each expert has the same weight in the regression process and that the system is fair. This is because in order to perform the MGM and the regression; all the point from all experts for a certain MG have to be combined into one graph as one set. If one expert had more points on that graph, this would affect the regression and that expert will have more influence or pull on the MGM. This ensures that at each point X there will be n Y values, where n is the number of experts in that trial.

The functions are then concatenated using MGM, using an automated tool that we developed by performing regression on the overall set of points. The sampling step is one of the tuning parameters that can be adjusted for more accurate concatenation. It depends on the shape and the complexity of the original functions. In this trial, we have used a step of 0.5 which was sufficient.

The graphs were pre-processed and sampled using our tool before being concatenated using MGM in MS Excel. It is in third order polynomial regression, as second order was not sufficient to represent the dramatic change due to a small number of the experts disagreeing with the rest, which meant that the overall pool of

points has been split into what looks like two different graphs. Fourth order did not present much improvement over the third order.

MG Label: abuse to the person-most recent episode of physical abuse
MG Code: gen-phys-abse-last



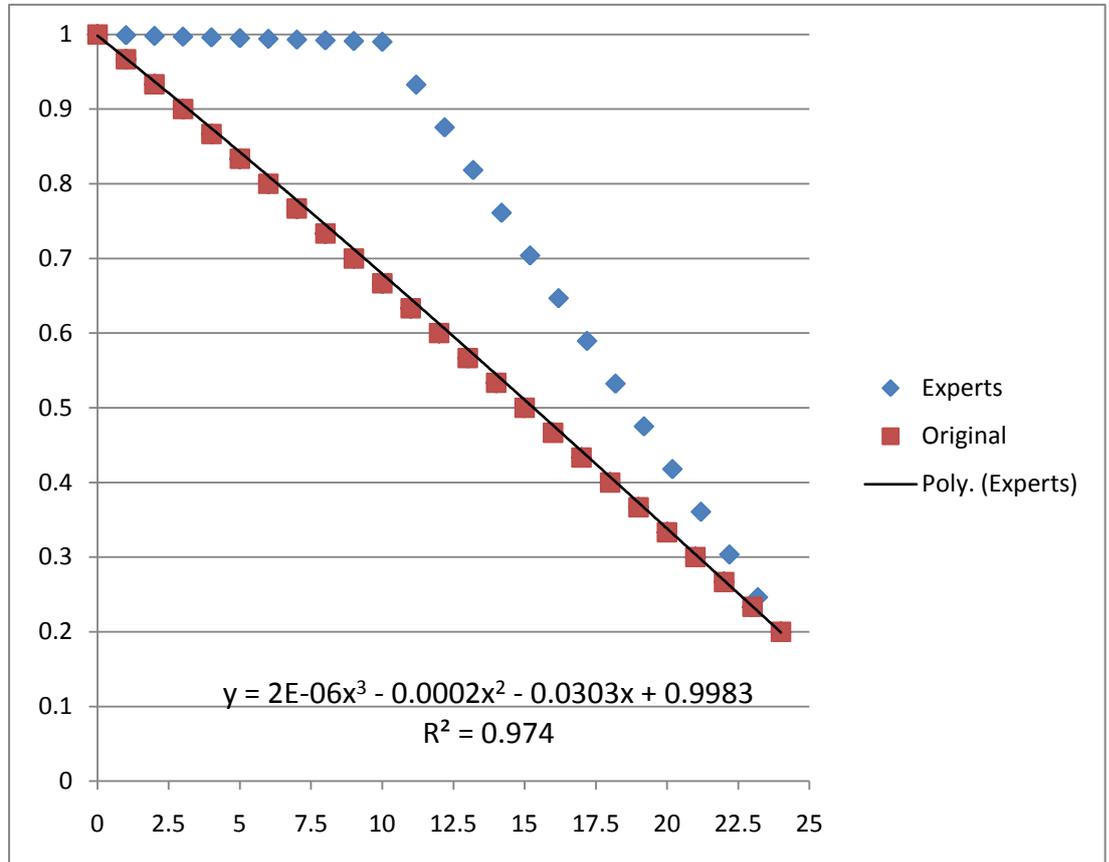
Start: (0,0.98)

End: (24,0.2)

Max.: (0, 0.98)

Comment	Clinician No.
0.5 at 24	14
0.8 at 24	16
0.2 at 24	18
0.6 at 24	20
1 at 0, 0 at 24	22
NOT ANSWERED`	30
people are so different and we can generalise, so question not answered	41
min 0.1, 0.7 at 24	5
0.4 at 24	9

MG Label: abuse to the person-most recent episode of emotional abuse
MG Code: gen-emot-abse-last



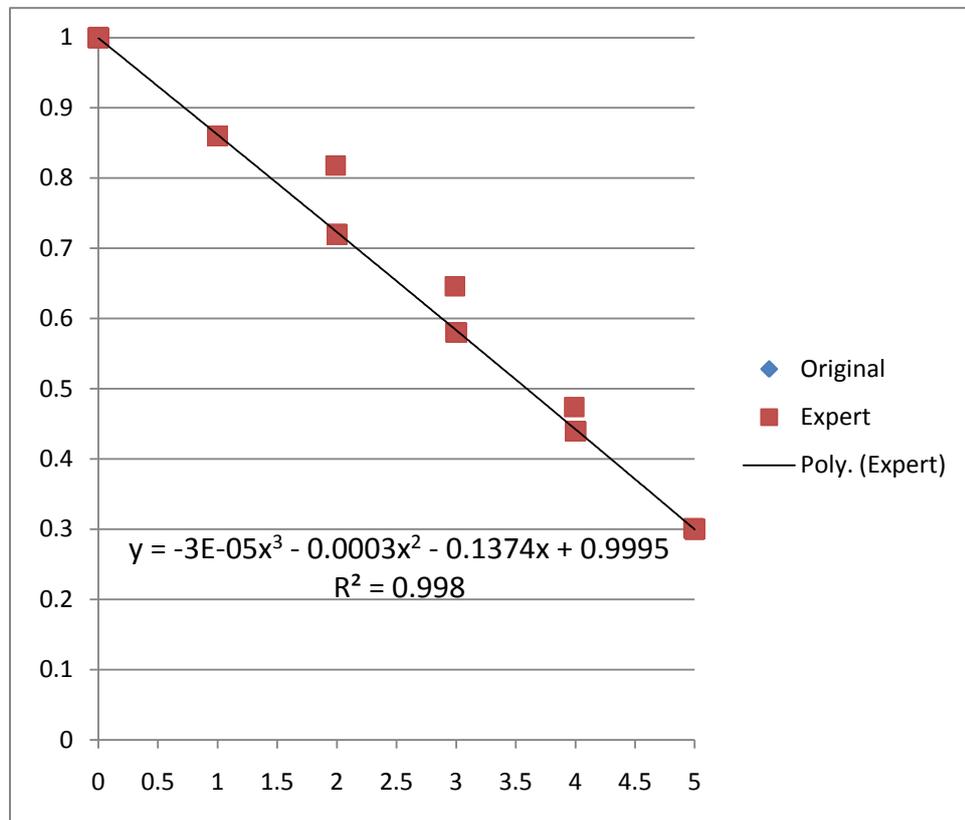
Start: (0,0.99)

End: (24,0.2)

Max.: (0, 0.99)

Comment	Clinician No.
0.7 at 1	14
0.8 at 24	16
0.5 at 24	18
More recent more risky	22
NOT ANSWERED	30
Recency (general) not able to isolate when something stops being recent - not done	4
people are so different and we can generalise, so question not answered	41
min 0.1, 0.7 at 24	5

MG Label: abuse to the person-most recent episode of sexual abuse
MG Code: gen-sex-abse-last



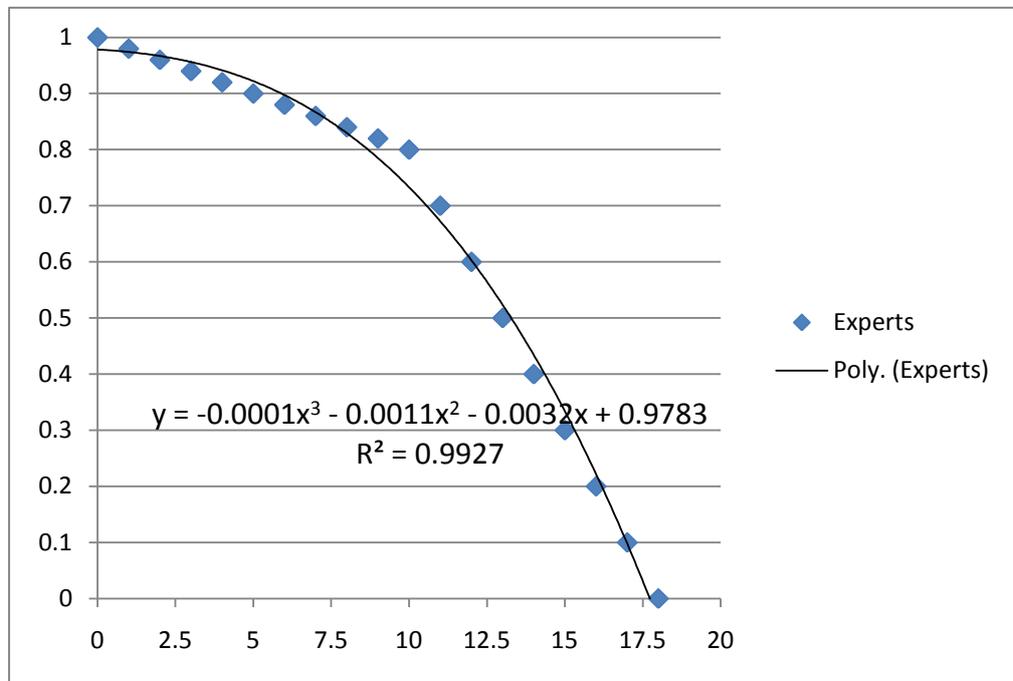
Start: (0,0.99)

End: (5,0.3)

Max.: (0, 0.99)

Comment	Clinician No.
0.8 at 5	14
0.6 at 5	16
0.4 at 5	18
Depends on person, not answered	20
1 at 0, 0 at 5	22
5YEARS WAS AT 0.7 BUT WAS SET AT ZERO	27
Scale should be 10 years	3
people are so different and we can generalise, so question not answered	41
min 0.1, 0.7 at 5	5
0.4 at 5	9

MG Label: age of youngest dependent
MG Code: gen-dep-ygnst-age



Start: (0,0.97)

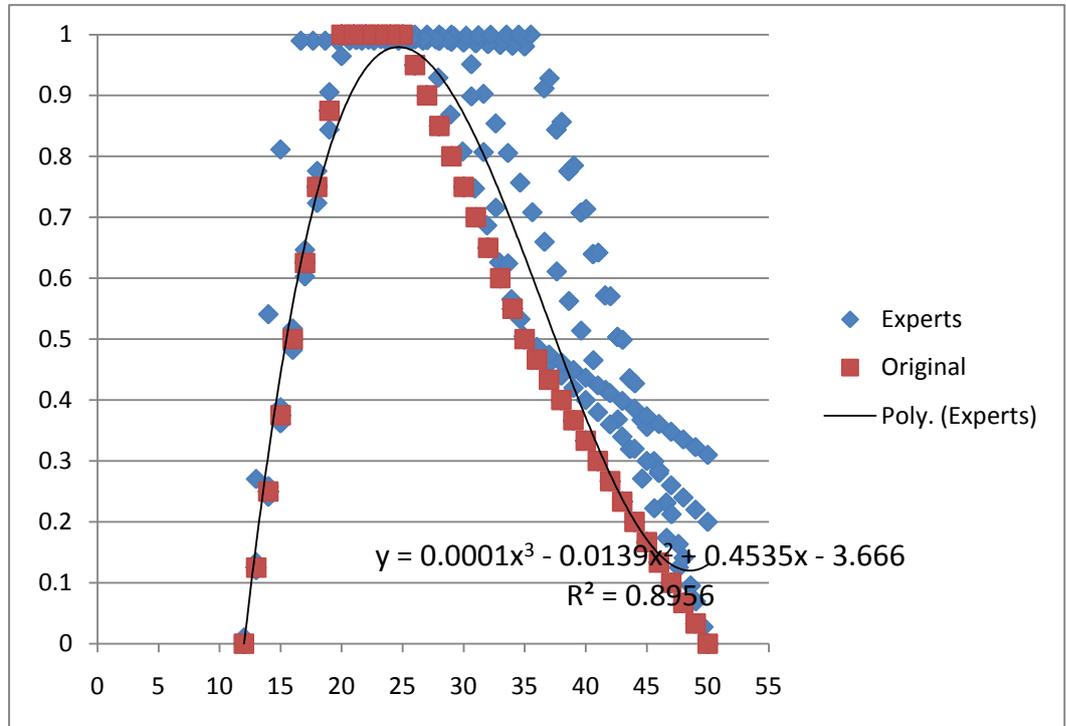
End: (18,0)

Max.: (0, 0.97)

Comment	Clinician No.

Note: No Datum was given,, in Expert 0. Not all experts elicited this one.

MG Label: age-harm to others or damage to property
MG Code: N/A



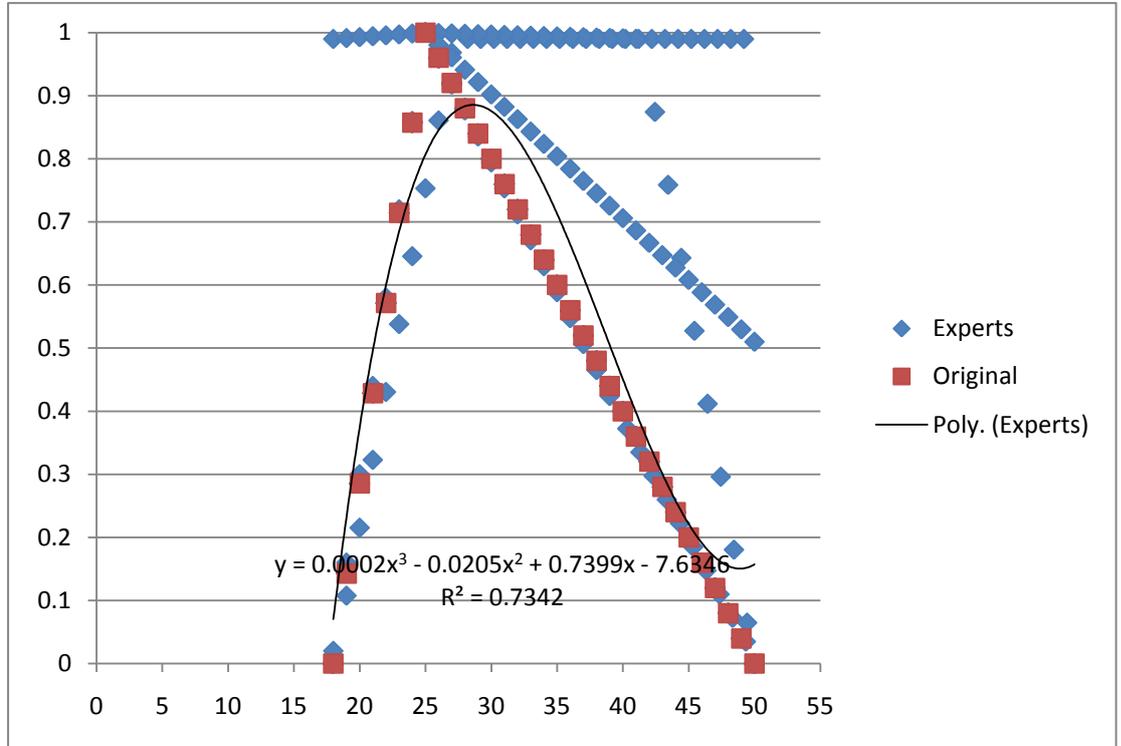
Start: (12,0)

End: (50,0.13)

Max.: (25, 0.98)

Comment	Clinician No.
0.3 at 12, 0.2 at 50	16
Query impact - not done	5

MG Label: age-risk to dependents
MG Code: N/A



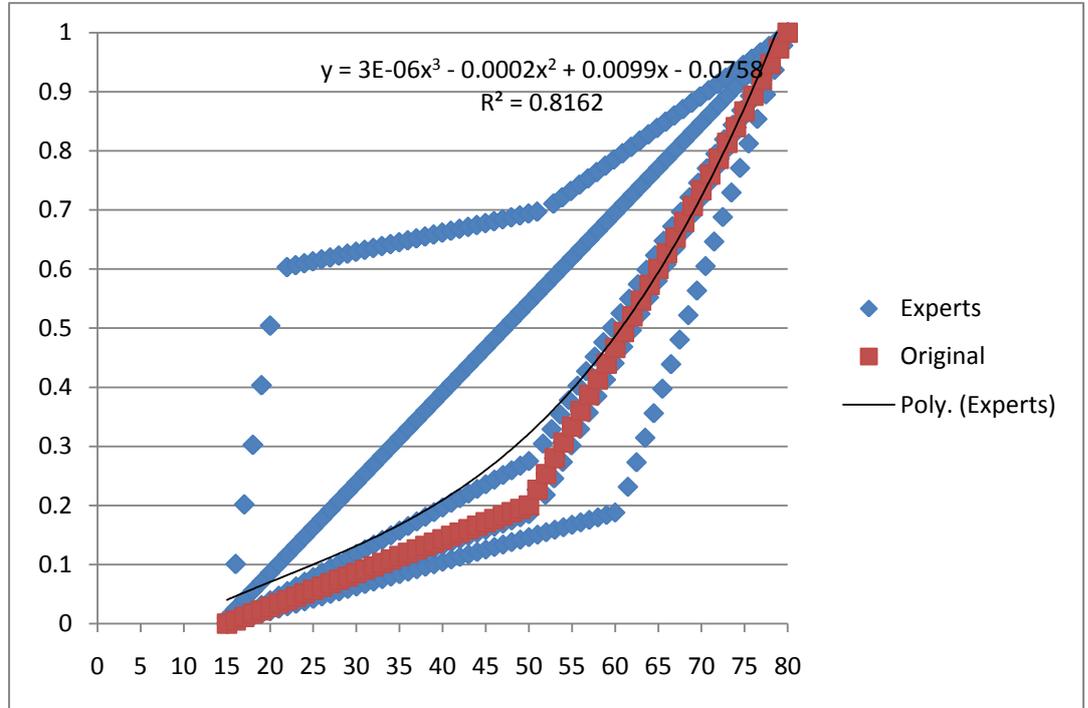
Start: (18,0.08)

End: (50,0.16)

Max.: (30,0.88)

Comment	Clinician No.
skipped - no experience	12
0.5 at 18, 0.2 at 50	16
not relevant	18
WOULD NOT ANSWER QUESTION AS COULD NOT ISCOLATE OTHER FACTORS	28
WOULD NOT ANSWER	31
Query impact - not done	5

MG Label: age-self neglect
MG Code: N/A



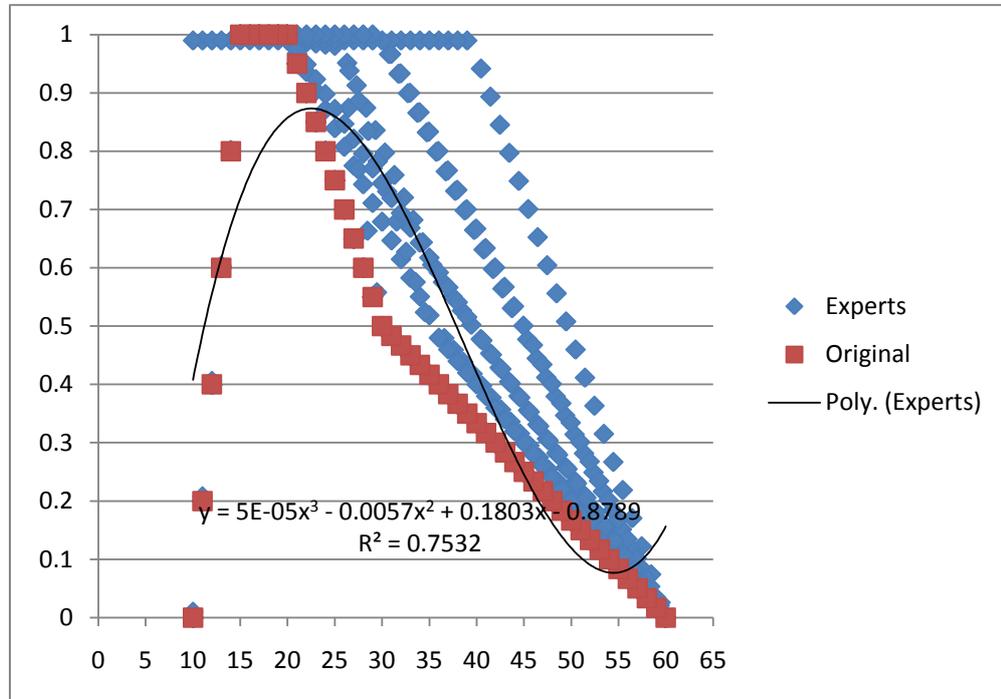
Start: (15,0.04)

End: (80,1)

Max.: (80,1)

Comment	Clinician No.
0.1 at 15	16
not relevant	18
WOULD NOT ANSWER	31
does not change with age so did not answer	41

MG Label: age-self-harm
MG Code: N/A



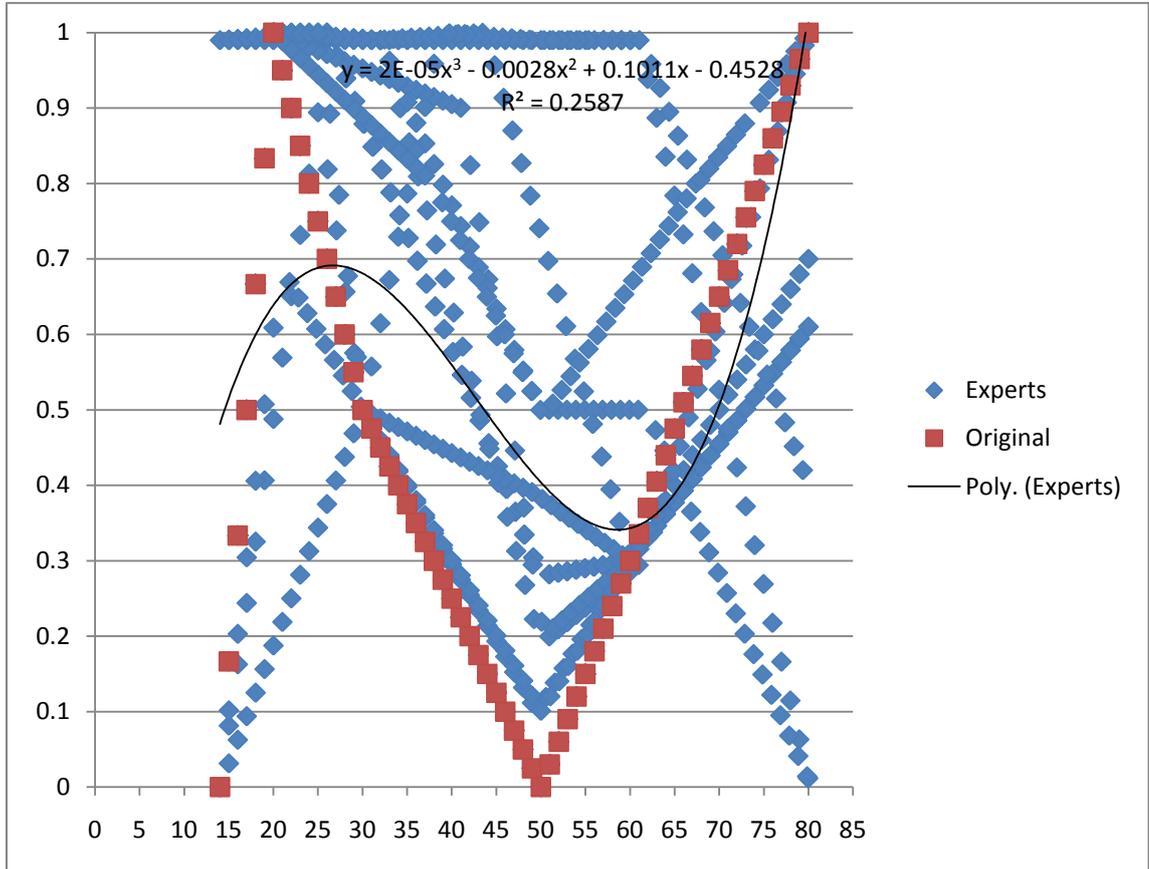
Start: (10,0.4)

End: (60,0.15)

Max.: (23,0.87)

Comment	Clinician No.
0.8 at 60	14
0.1 at 10, 0.1 at 60	16
WOULD NOT ANSWER	31

MG Label: age-suicide
MG Code: N/A



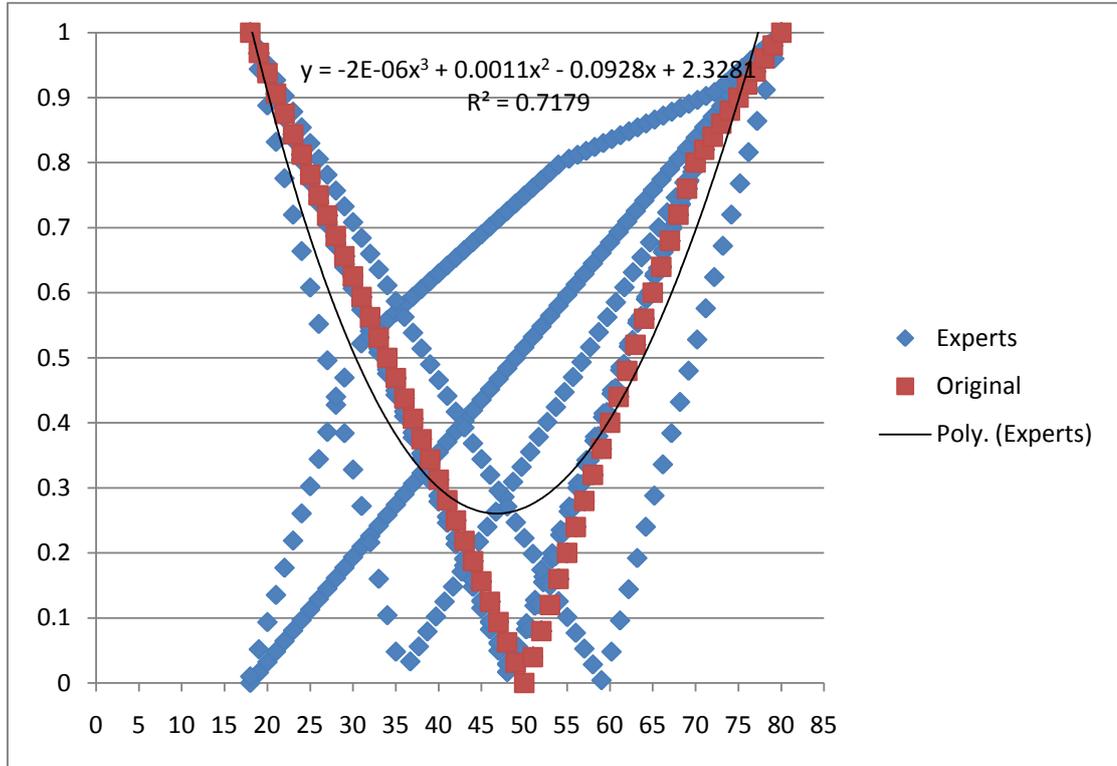
Start: (15,0.48)

End: (80,1)

Max.: (80,1)

Comment	Clinician No.
0.5 at 80	14
0.1 at 14	16
0.8 at 80	5

MG Label: age-vulnerability of service user
MG Code: N/A



Start: (18,1)

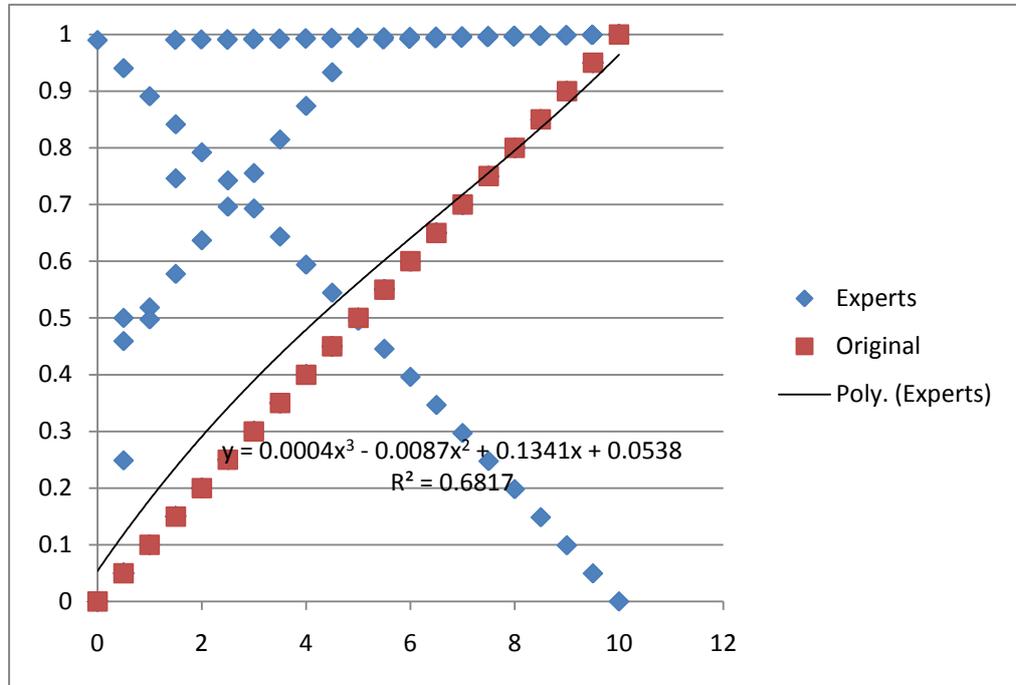
End: (78,1)

Max.: s+e

Comment	Clinician No.
0.5 at 48	12
0.2 at 48	16
not relevant	18
SACLW SHOULD START AT 16	28
no min 0.1 at 48	5

MG Label: current intention to commit suicide-level of detail and clarity of suicide plan

MG Code: suic-plan-dtail



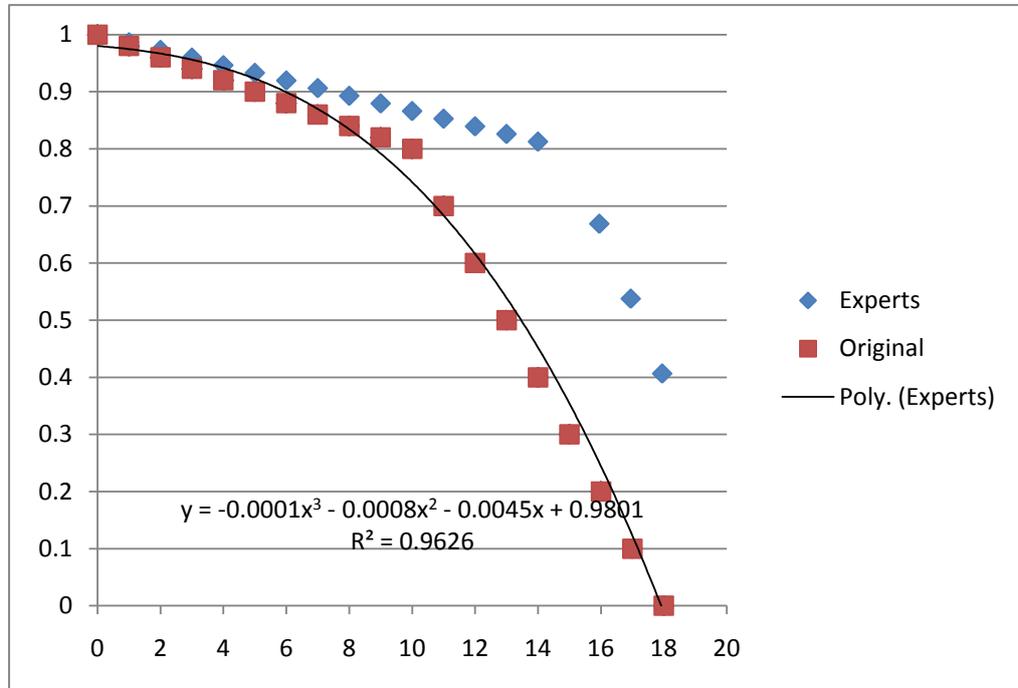
Start: (0,0.054)

End: (10,0.98)

Max.: (10,0.98)

Comment	Clinician No.
not relevant	17
NOT ANSWERED	29
not asked in error	38
Depends on individual. Some people are meticulous planners, some aren't. Not answered.	42
Not done couldn't say	6

MG Label: demographics-age of youngest dependent
MG Code: gen-dep-ygnst-age



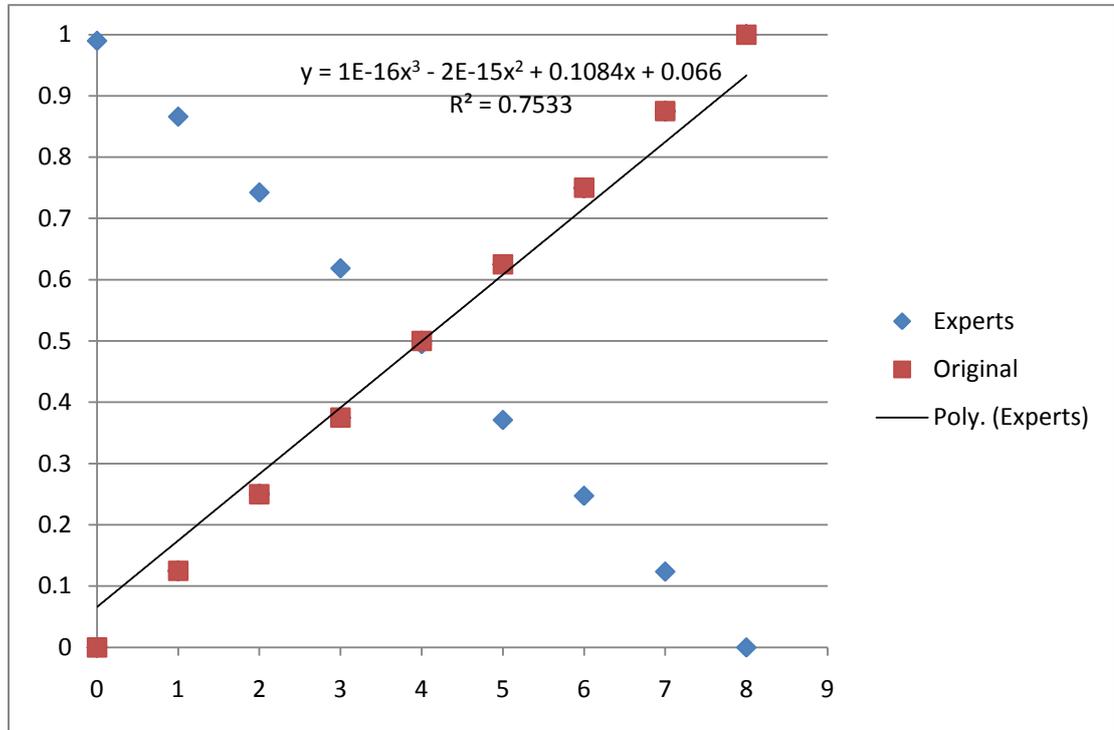
Start: (0,0.98)

End: (17.8,0)

Max.: (0,0.98)

Comment	Clinician No.
0.1 at 18	18

MG Label: demographics-number of dependents
MG Code: gen-accom-num-dep



Start: (0,0.066)

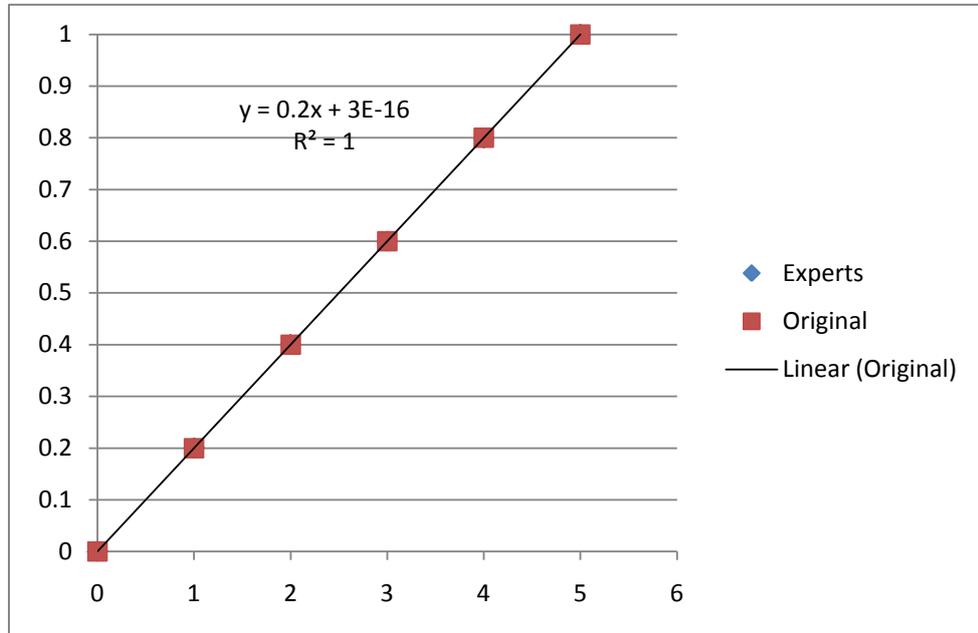
End: (8,0.93)

Max.: (8,0.93)

Comment	Clinician No.
skipped - depends on ages,support,resources	12
not relevant	18
can not isolate from other factors so would not answer	41
0=0.5 1=0.3 8=0.2 0.5 at 0	5

MG Label: demographics-number of non-dependents sharing accommodation

MG Code: gen-accm-share-nd



Start: (0,0)

End: (5,1)

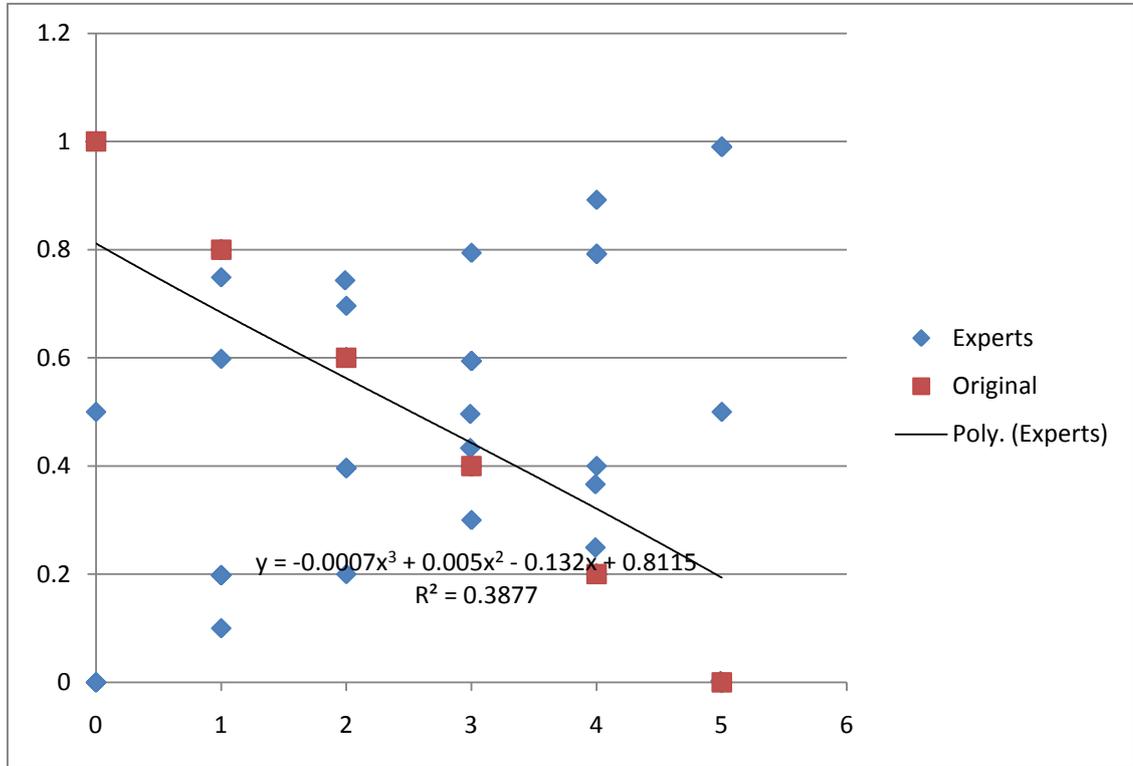
Max.: (5,1)

Comment	Clinician No.
not relevant END HERE	12
0.5 at 5 END HERE	14
not relevant END HERE	18
Not relevant, not answered End here	20
End here	22
THIS WAS THE LAST QUESTION	28
interview ended here	37
could not isolate from other factors so could not answer interview ended here	41
0=0.5 2=0.3	5

Note: Graphs overlap, as Experts agreed with original.

MG Label: harm or damage-first time a destructive act against property occurred?

MG Code: hto-first-time-destructive-ep



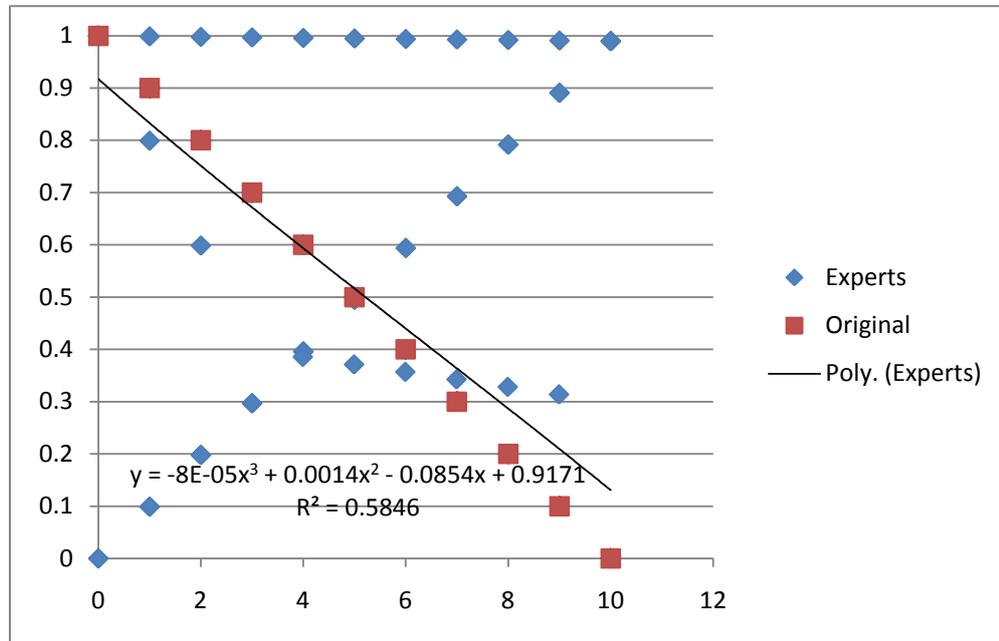
Start: (0,0.81)

End: (5,0.19)

Max.: (0,0.81)

Comment	Clinician No.
0.4 at 5	17
No more or less. If it happens often, it should have been addressed, so highlights fault in system. So not answered. END HERE	42
0.3 at 5	5
Makes no difference - not done	6

MG Label: harm or damage-first time animal abuse occurred
MG Code: hto-first-time-animal-ep



Start: (0,0.917)

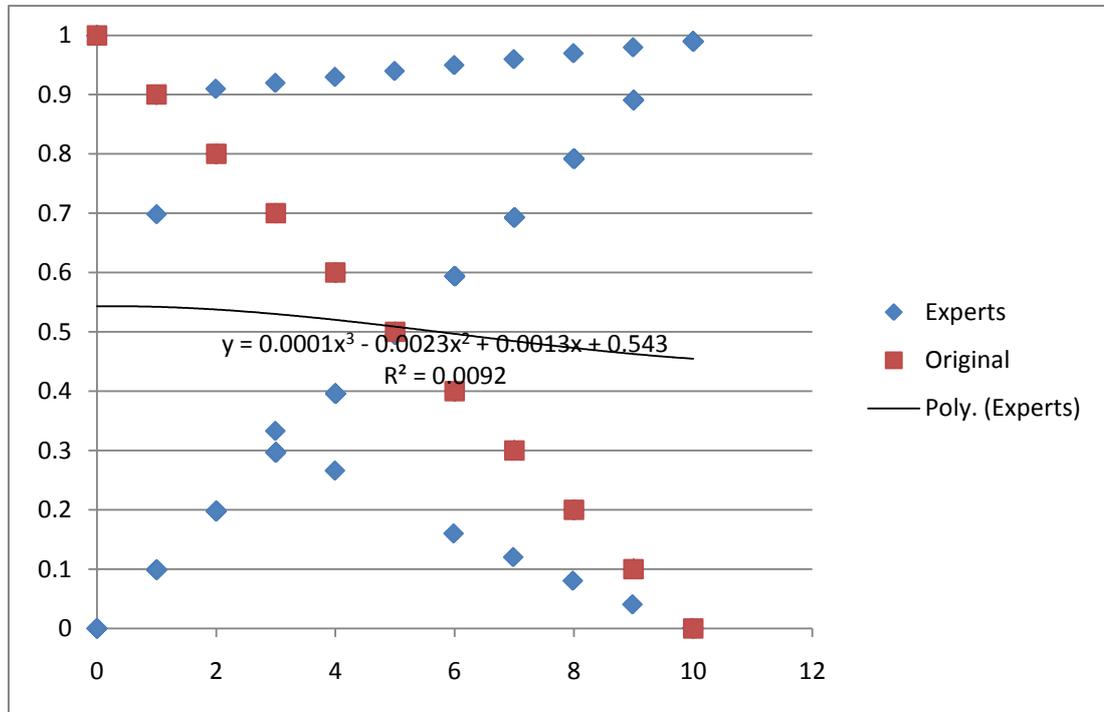
End: (10,0.13)

Max.: (0,0.917)

Comment	Clinician No.
Skipped - no experience	11
0.8 at 10	17
Need to know why, not answered	21
10 YEARS WAS PLACED AT 0.4 BUT WE HAD TO PLACE AT ZERO	26
not answered	40
0.3 at 10	5
No difference, all maximum	6
Equally concerned - not done	8

MG Label: harm or damage-first time emotional episode of harm to others occurred

MG Code: hto-first-time-emotional-ep



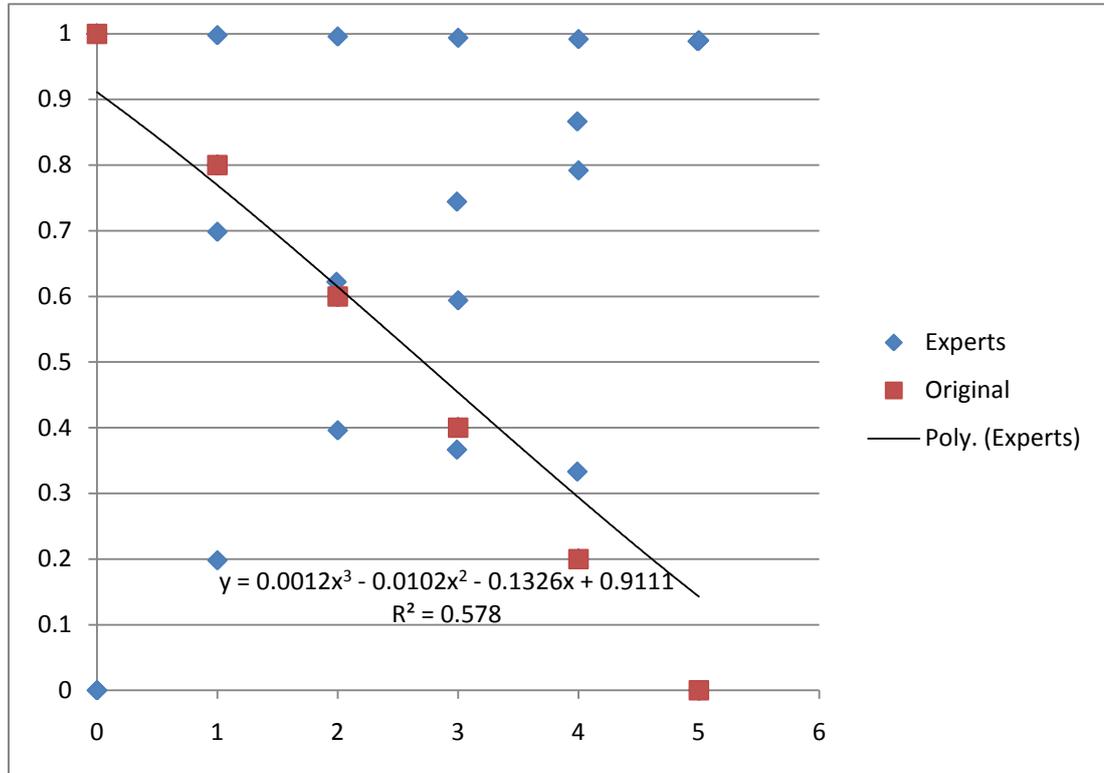
Start: (0,0.54)

End: (10,0.45)

Max.: (0,0.54)

Comment	Clinician No.
Equal	17
could not isolate from other factors so would not answer	38
Depends on mental health at the time of first episode. Harm can be indirect means of self-harming. Not answered.	42
0.1 at 10	5
0.2 at 10	6

MG Label: harm or damage-first time fire-setting episode occurred
MG Code: hto-first-time-fire-setting-ep



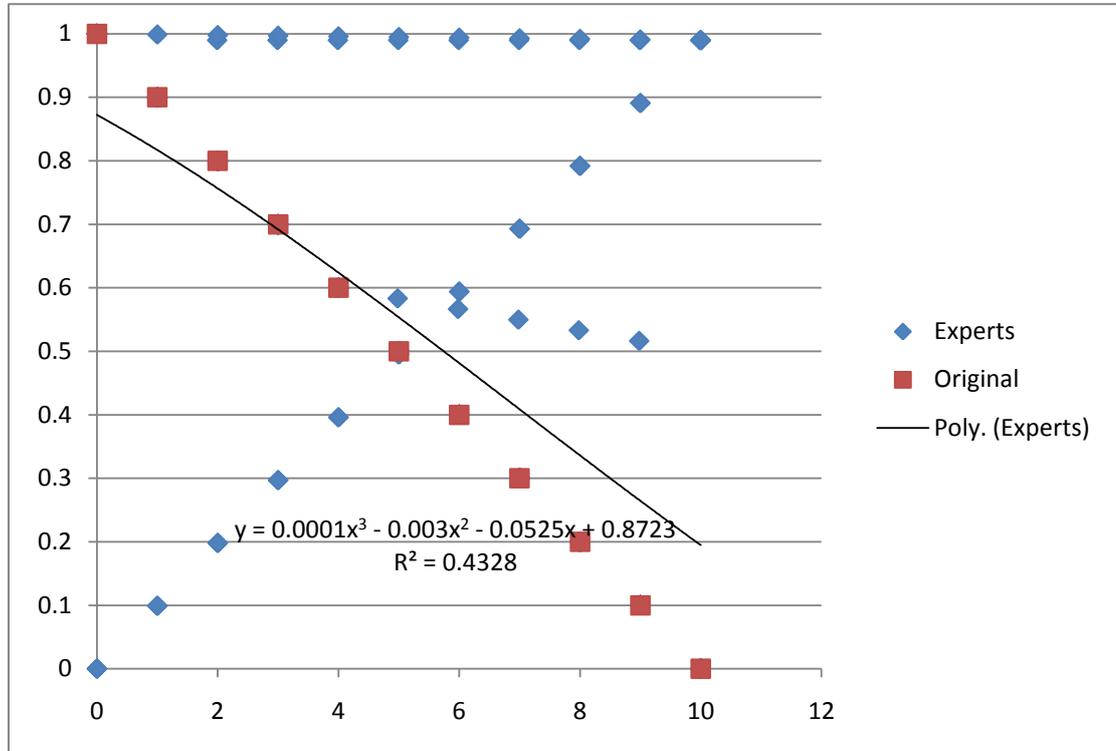
Start: (0,0.91)

End: (5,0.13)

Max.: (0,0.91)

Comment	Clinician No.
0.5 at 5	11
0.8 at 5	17
Not interested in ANY first time questions, not answered	19
5 YEARS SHOULD BE PLACED AT 0.2 BUT WE HAD TO SET AT ZERO	26
not answered	40
0.3 at 5	5
No difference, all maximum	6
Equally concerned - not done	8

MG Label: harm or damage-first time sexual assault occurred
MG Code: hto-first-time-sexual-ep



Start: (0,0.87)

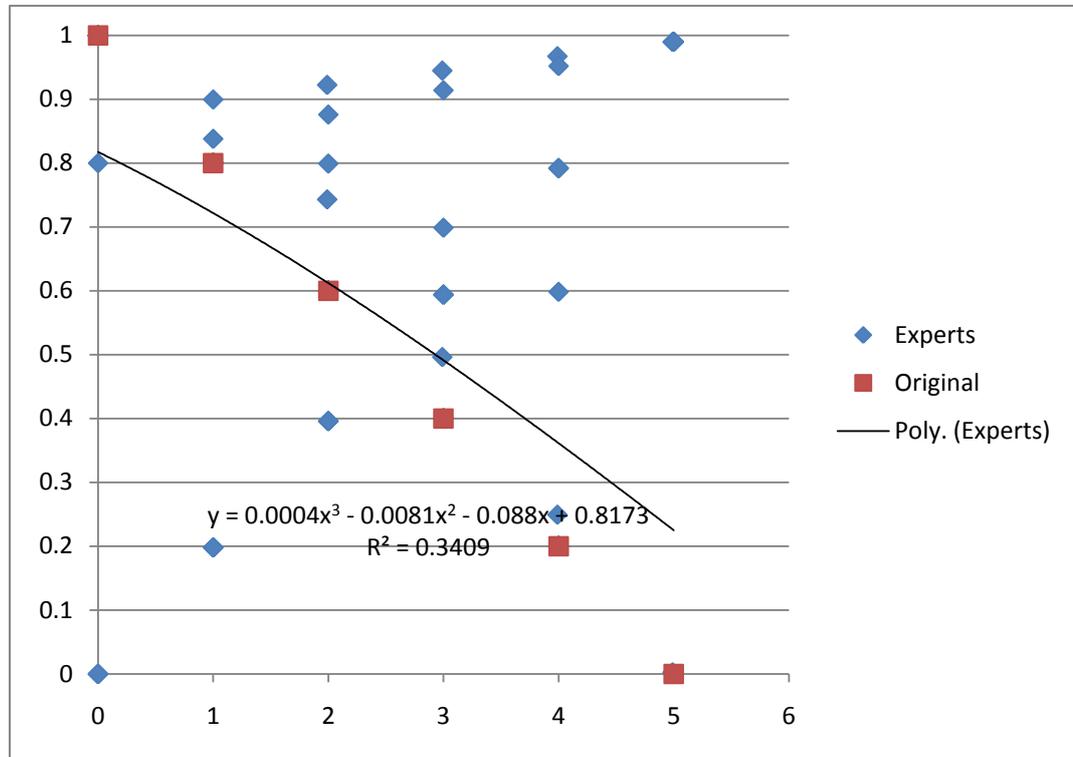
End: (10,0.2)

Max.: (0,0.87)

Comment	Clinician No.
0.5 at 10	17
0.3 at 10	23
SHOULD BE LINKED TO NUMBER OF OCCURENCES 10 YEARS SHOULD BE PLACED AT 0.4 BUT HAD TO BE PLACED AT ZERO	26
not answered as could not isolate other factors	40
0.5 at 10	5
Makes no difference, maximum at all points	6

MG Label: harm or damage-first time violent episode of harm to others occurred

MG Code: hto-first-time-violent-ep



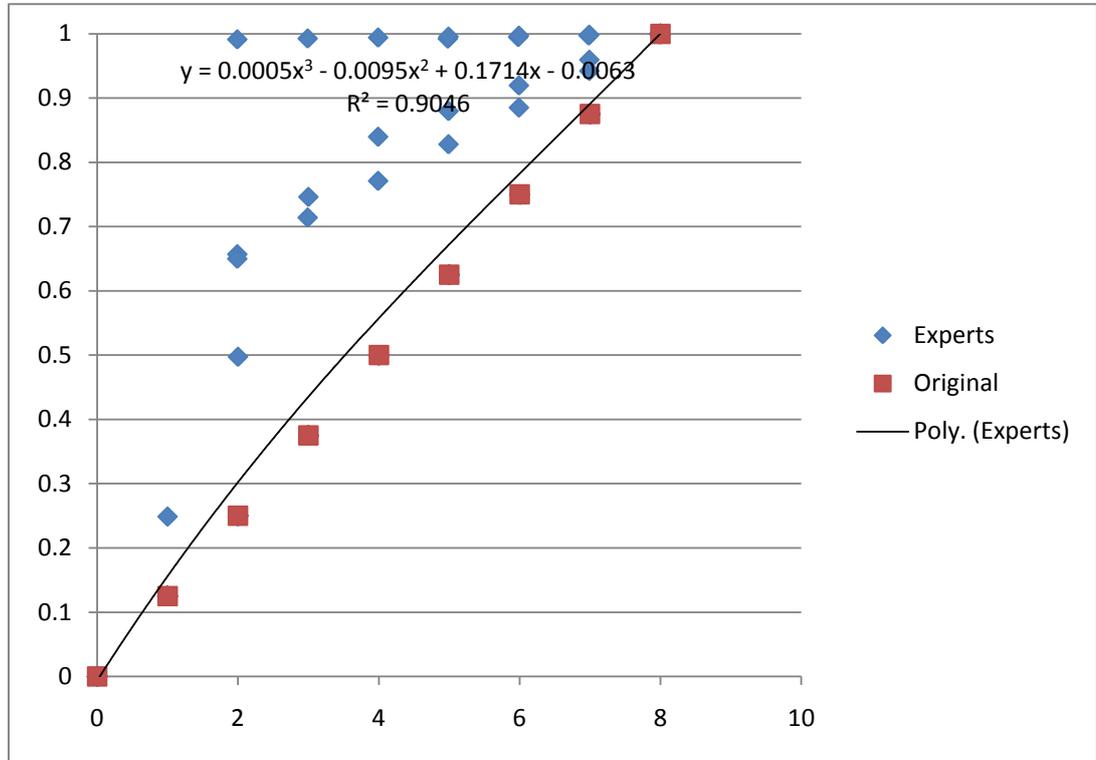
Start: (0,0.82)

End: (5,0.22)

Max.: (0,0.82)

Comment	Clinician No.
equal not relevant	17
5 YEARS SHOULD BE SET AT 0.2 BUT WE HAD TO SET AT ZERO	26
NOT ANSWERED, THIS WAS THE LAST QUESTION	29
clinician wasn't sure so did not want answer	38
If repeating harm to particular person, why? Command hallucination perhaps?	42
0.5 at 5	5

MG Label: harm or damage-how many harm or damage episodes
MG Code: hto-number



Start: (0,0)

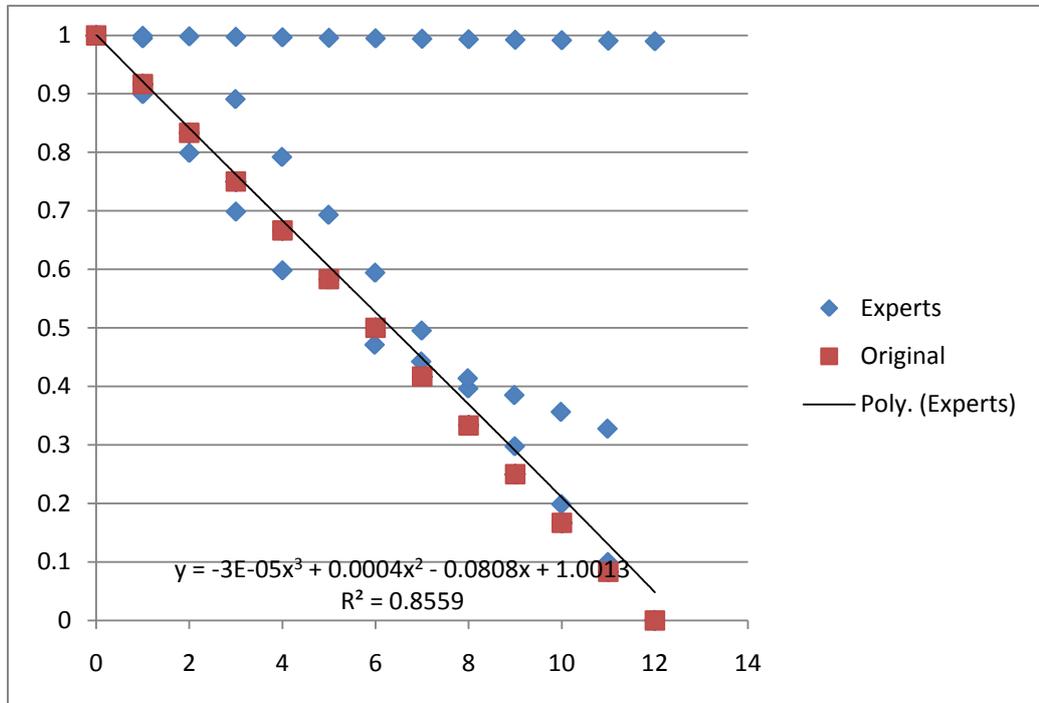
End: (8,1)

Max.: (8,1)

Comment	Clinician No.
END HERE	13

MG Label: harm or damage-most recent destructive act concerning property

MG Code: hto-recent-destructive-eps



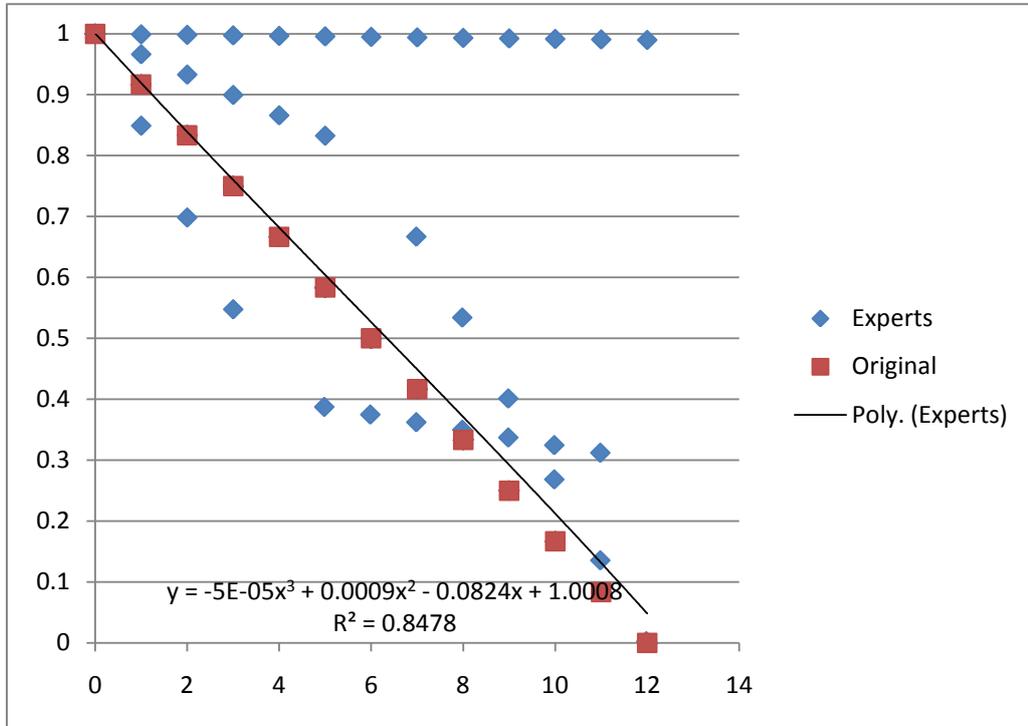
Start: (0,1)

End: (12,0.04)

Max.: (0,1)

Comment	Clinician No.
0.4 at 12	10
0.5 at 12	11
0.3 at 12	17
not answered	40
0.3 at 12	5
Scale should be a couple of years,0.5 at 12	8

MG Label: harm or damage-most recent episode of animal abuse
MG Code: hto-recent-animal-eps



Start: (0,1)

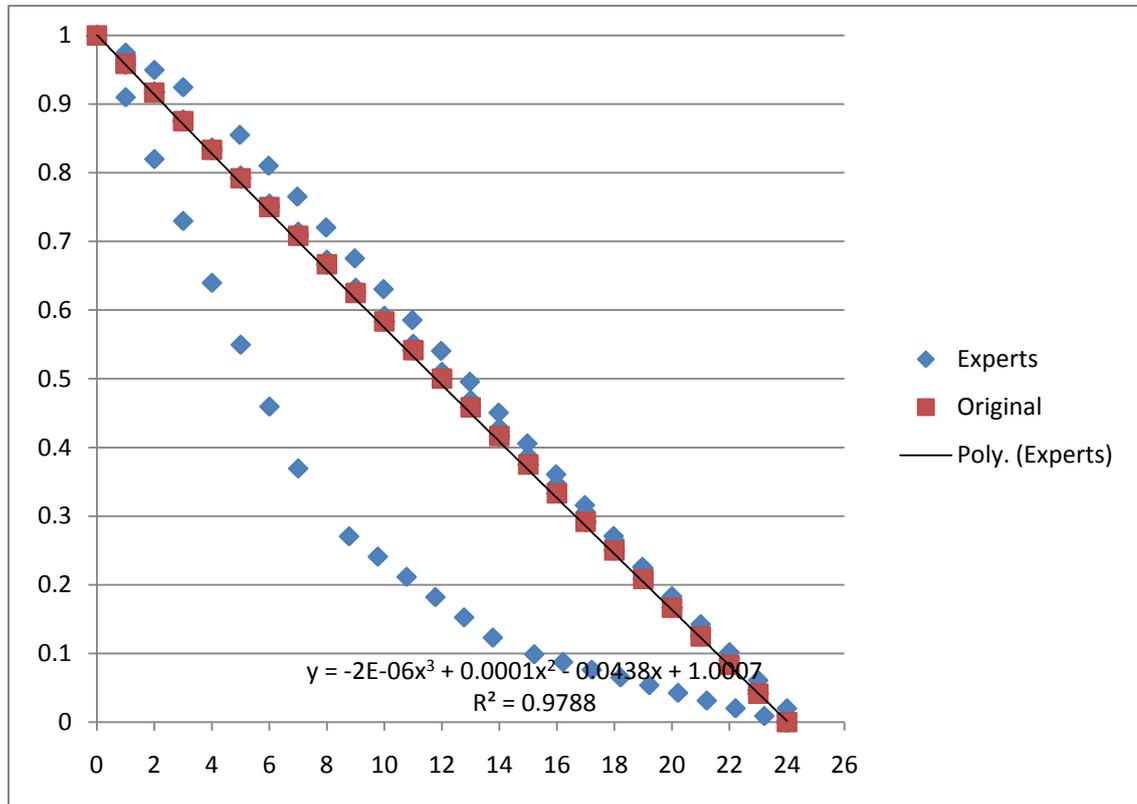
End: (12,0.03)

Max.: (0,1)

Comment	Clinician No.
0.4 at 12	10
0.6 at 12	11
0.5 at 12	17
0.7 at 12	19
TWO YEARS WAS SET AT 0.2 BUT WE HAD TO SET AT ZERO	26
not answered	40
0.3 at 12	5
No difference, all maximum	6
Equally concerned - not done	8

MG Label: harm or damage-most recent episode of emotional harm to others

MG Code: hto-recent-emotional-eps



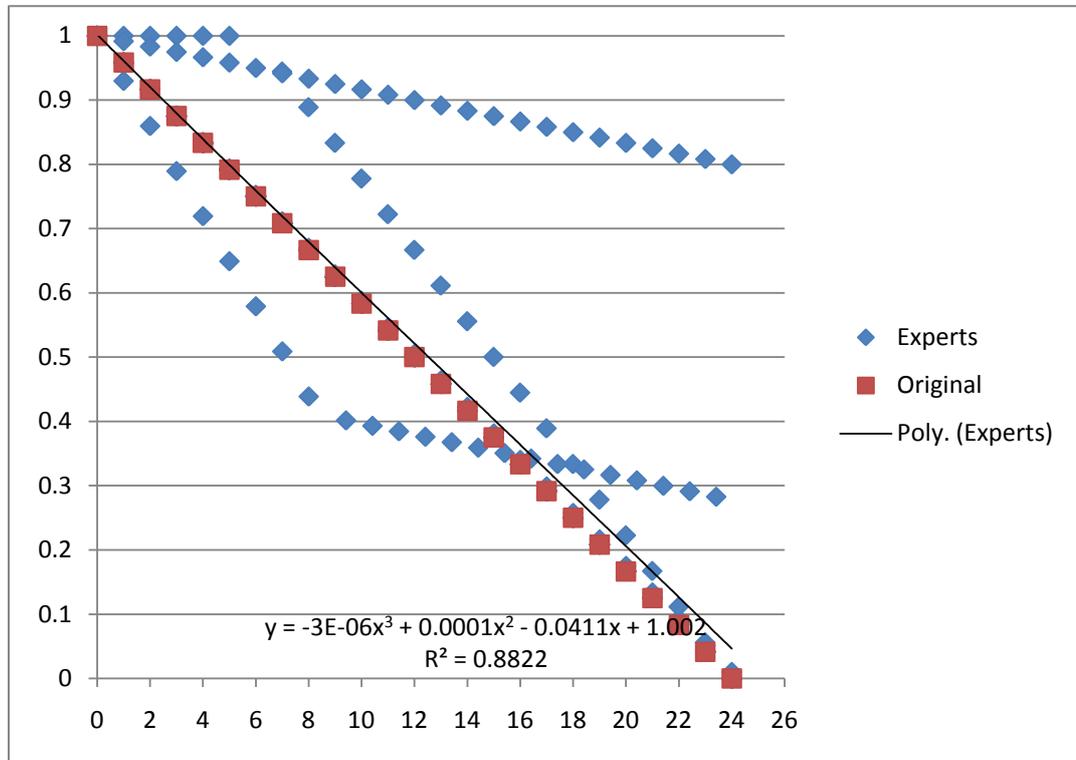
Start: (0,1)

End: (24,0)

Max.: (0,1)

Comment	Clinician No.
0.3 at 24	10
0.2 at 24	17
LAST QUESTION INTERVIEW ENDED HERE	32
0.8 at 24	34
could not isolate from other factors so would not answer	38
couldn't answer	40
Not asked.	42
0.1.at 10	5
Scale should be 5 years, 0.5 at 24	6

MG Label: harm or damage-most recent episode of fire setting
MG Code: hto-recent-fire-setting-eps



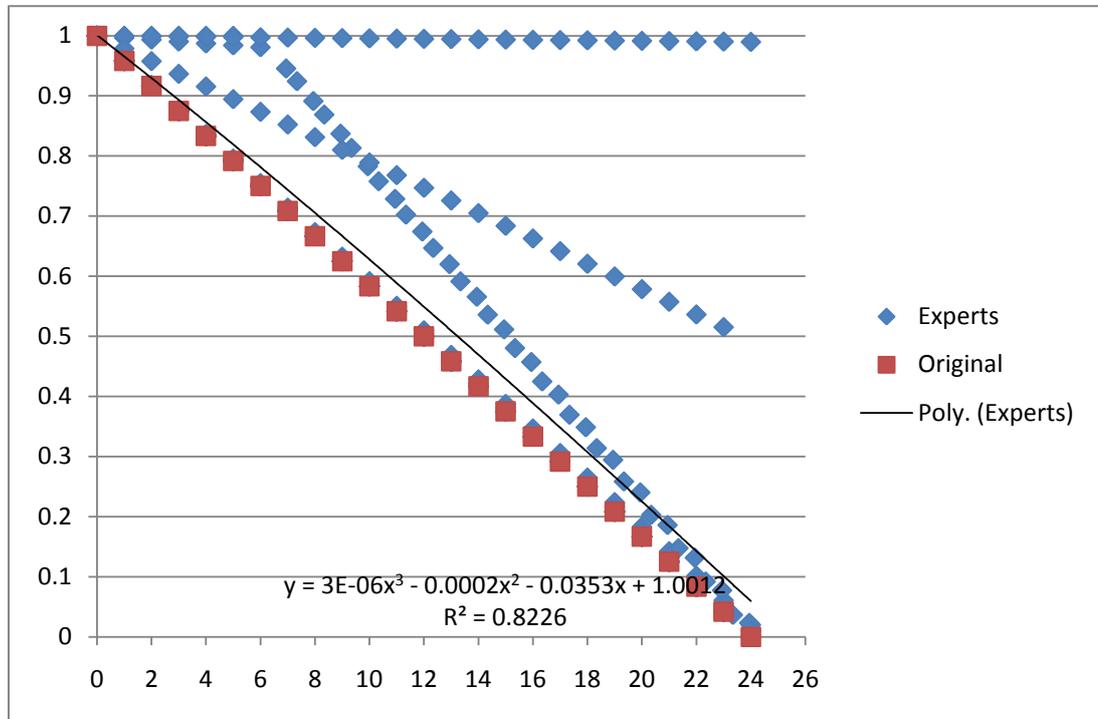
Start: (0,1)

End: (24,0.05)

Max.: (0,1)

Comment	Clinician No.
0.5 at 24	10
0.5 at 24	11
0.5 at 24	17
2 YEARS WAS PLACED AT 0.2 BUT WE HAD TO PLACE AT ZERO.	26
not answered	40
0.3 at 24	5
No difference, all maximum	6
Equally concerned - not done	8

MG Label: harm or damage-most recent episode of sexual assault
MG Code: hto-recent-sexual-eps



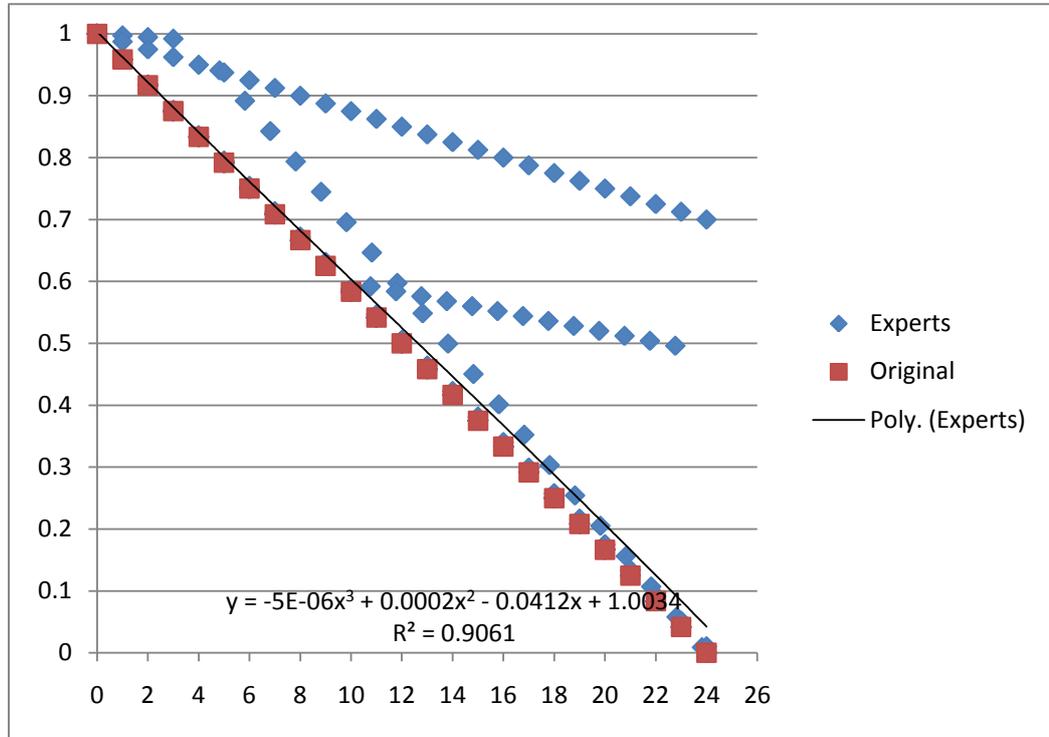
Start: (0,1)

End: (24,0.06)

Max.: (0,1)

Comment	Clinician No.
0.3 at 24	10
0.6 at 24	24
0.4 at 24	17
0.5 at 24	23
TIMESCALE SHOULD BE 24 MONTHS, 2 MONTHS AT 0.4 BUT WAS PLACED AT ZERO	26
Recency lasts until the episode has passed, and an episode can last any amount of time (note: clinician's definition of episode seems to be of mental instability than of sexual assault). So not answered. Drugs and other factors may have impact. We must be	42
0.5 at 24	5
Should be 5 years, 0.8 at 24	6

MG Label: harm or damage-most recent episode of violent harm to others
MG Code: hto-recent-violent-eps



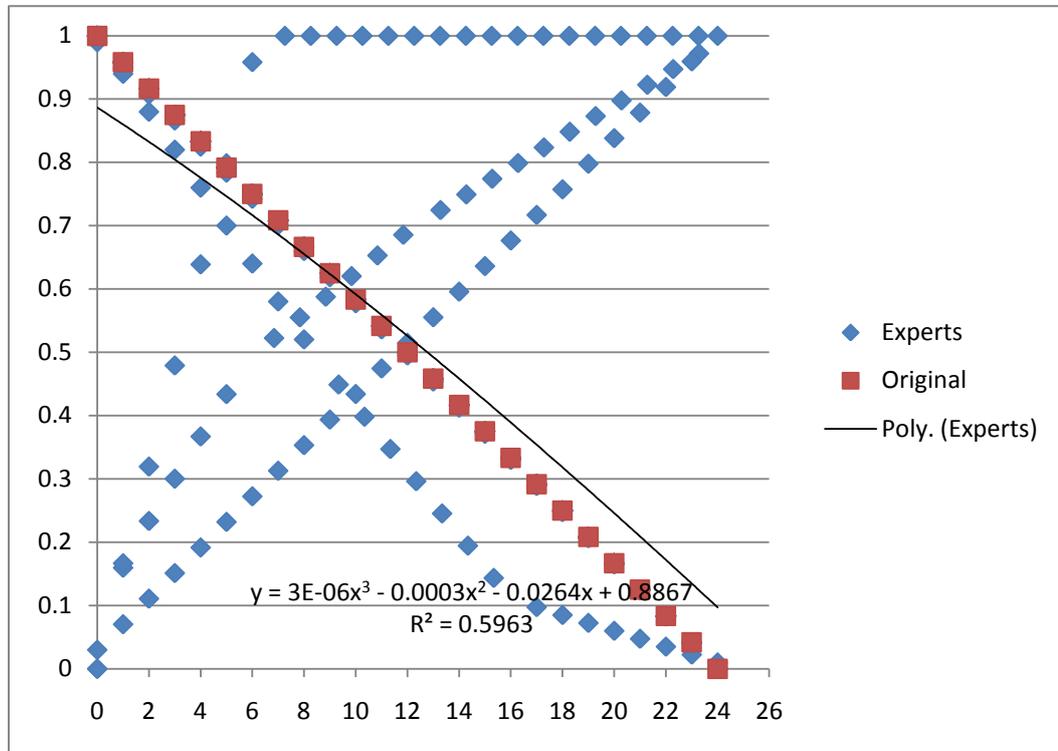
Start: (0,1)

End: (24,0.04)

Max.: (0,1)

Comment	Clinician No.
0.3 at 24	11
0.1 at 24	17
24 MONTHS SHOULD BE PLACED 0.2 BUT WE HAD TO SET AT ZERO	26
interview ended here	38
need to look at the time frame again so not answered	40
Not asked.	42
0.5 at 24	5
Difficult because mental health could have improved over time, 0.5 at 5	6

MG Label: self harm-first time self-harm episode occurred
MG Code: sh-first-time-ep



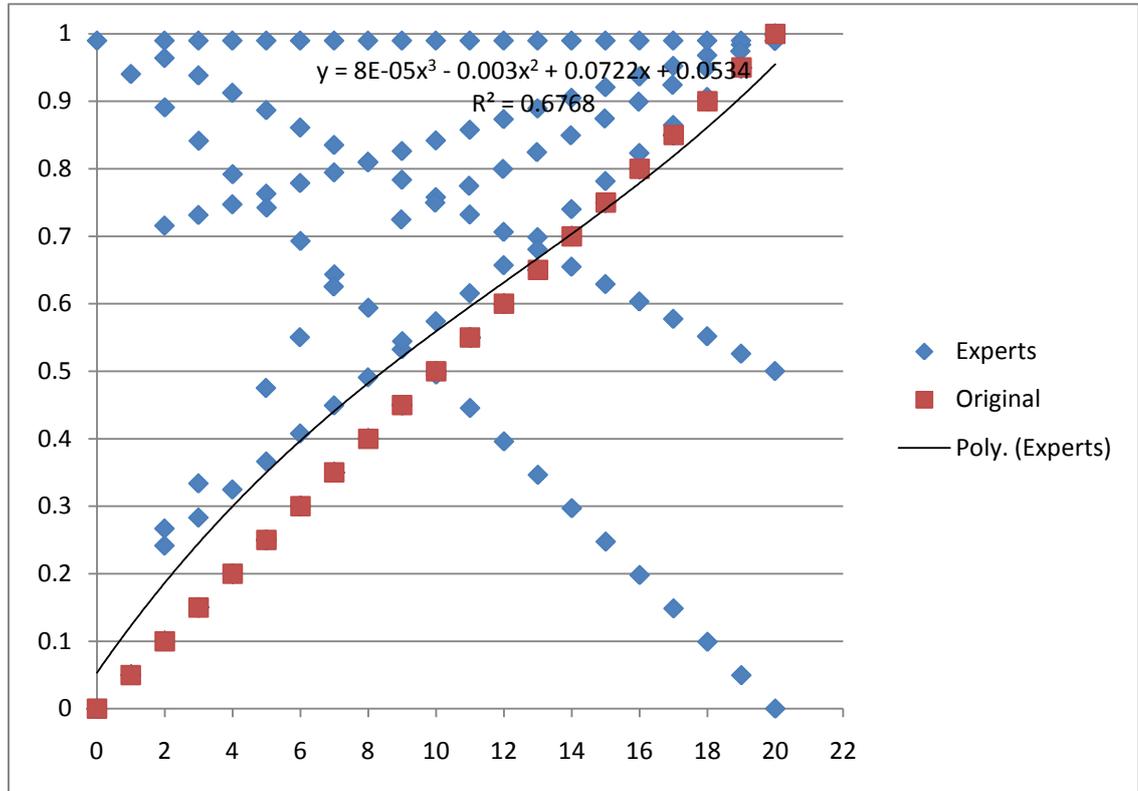
Start: (0,0.89)

End: (24,0.1)

Max.: (0,0.89)

Comment	Clinician No.
0.5 at 24	17
Not answered	19
COULD NOT ISCOLATE OTHER FACTORS SO DID NOT ANSWER	29
could not isolate from other factors so would not answer	38
Not asked.	42
0.1 at 30	5

MG Label: self harm-how many self-harm episodes
MG Code: sh-number-eps



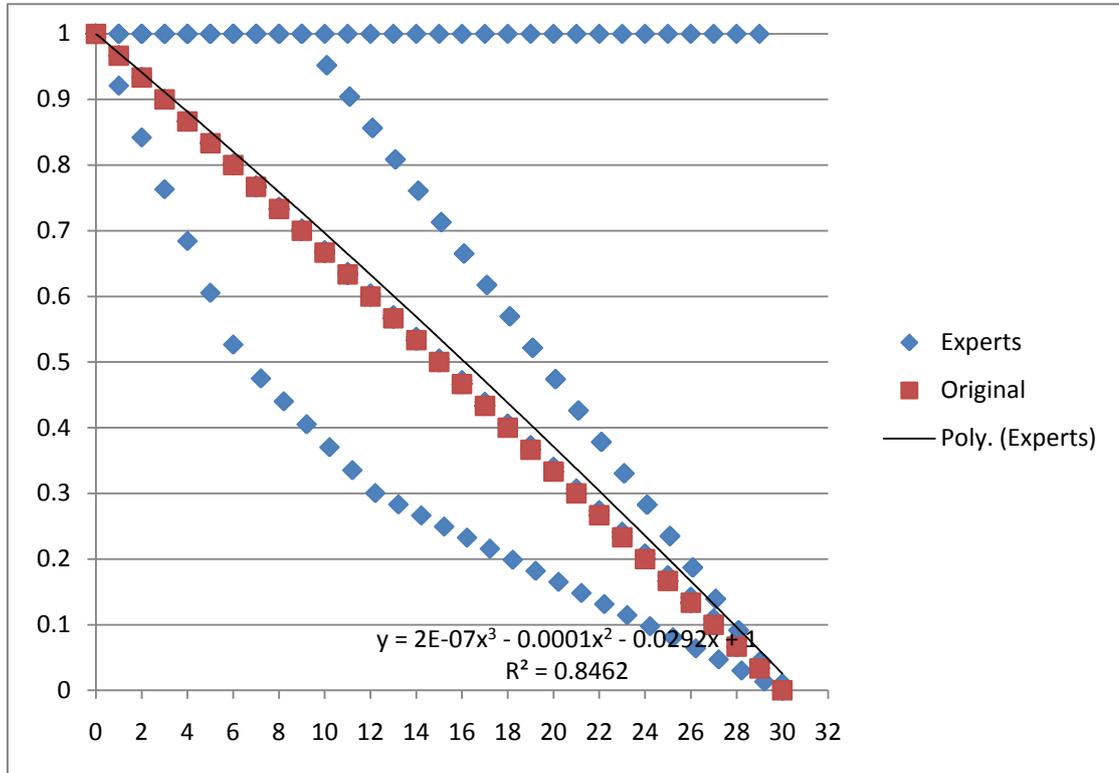
Start: (0,0.53)

End: (20,0.95)

Max.: (20,0.95)

Comment	Clinician No.
0.8 at 20	17
COULD NOT ISCOLATE OTHER FACTORS SO DID NOT ANSWER	29
NOT ANSWERED	32
0.9 at 0	34
any self harm attempt could be very risky, so number of attempts not relevant - would not answer	38
must be looked at in conjunction with time frame so not answered	40
Same amount of risk either way. If clinician knows the reason for single self-harm episode, then could be prevented if treated well. Depends on seriousness of self-harm. So not answered.	42
No difference-not done	8

MG Label: self harm-most recent self-harm episode
MG Code: sh-most-rec-ep



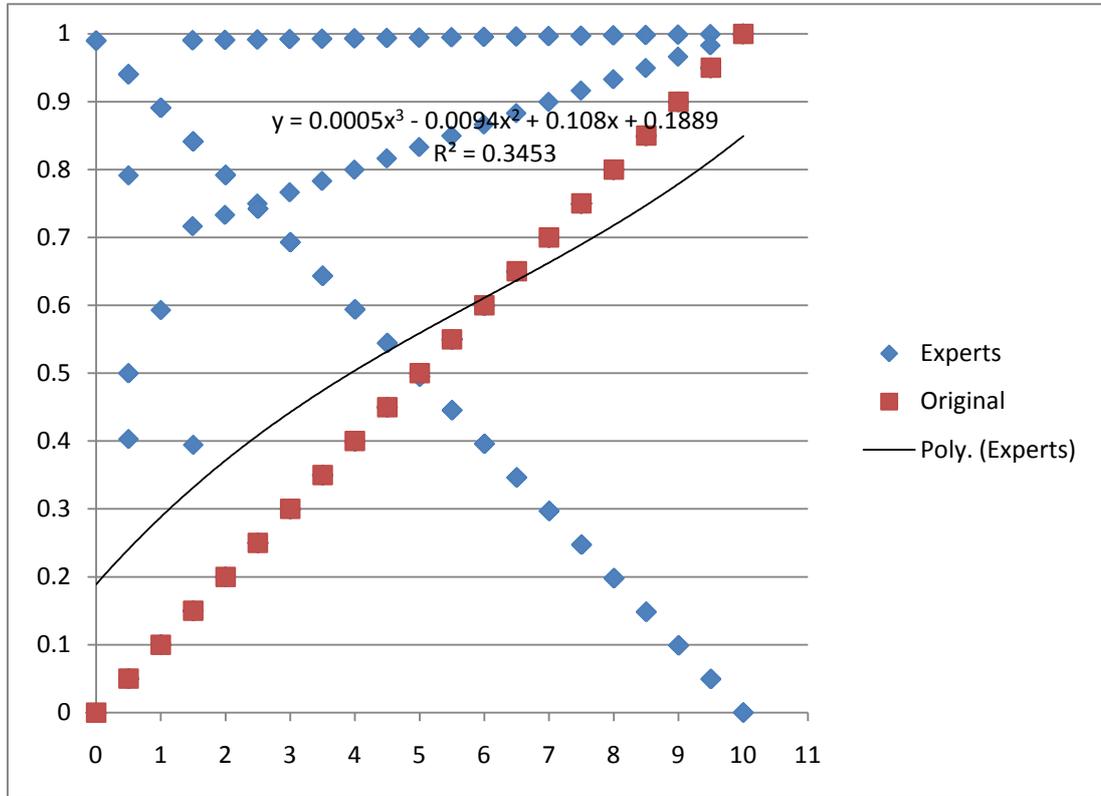
Start: (0,1)

End: (30,0.02)

Max.: (0,1)

Comment	Clinician No.
0.2 at 30 scale up to 6 months	10
Month is long enough for recency	11
Not sure, not answered	19
0.7 at 30	34
would need to know why self harm occurred, so would not answer	38
could not put a time scale on it so not answered	40
Hard question to answer. Too dependent on other factors. After all, change in behaviour could be a seasonal thing. So not answered.	42
0.1 at 30	5
Scale should go to 6 months, 1 at 30	6
Scale should be a couple of months, 0.5 at 6 months	8

MG Label: self harm-planning involved in self-harm episodes
MG Code: sh-planning



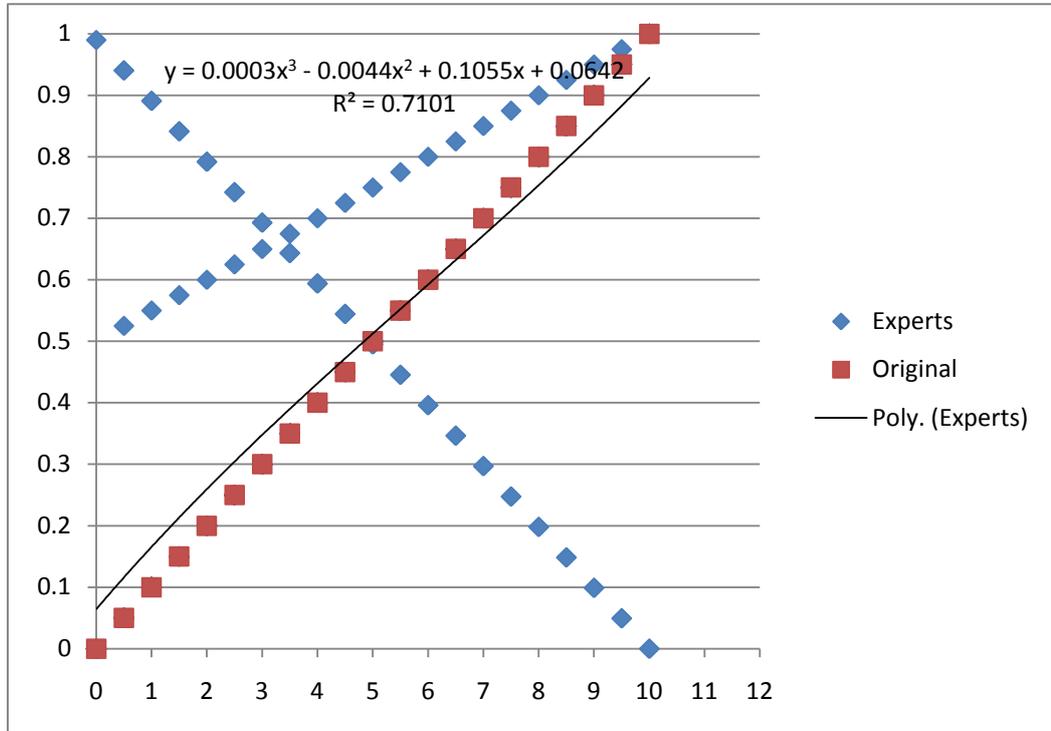
Start: (0,0.19)

End: (10,0.84)

Max.: (10,0.84)

Comment	Clinician No.
Self harm generally less planned than suicide. Problem with 0 as covers all rest of population who will not self harm	5

MG Label: self harm-self-harming to get help with difficulties
MG Code: sh-for-hlp-diff



Start: (0,0.64)

End: (10,0.92)

Max.: (10,0.92)

Comment	Clinician No.
How do we know this - not answered	19
NOT ANSWERED	29
NOT ANSWERED	32
Not possible to say what is a cry for help and what isn't - not answered	34
Any cry for help would be indicator of a fault in the system rather than of risk. So not relevant to risk. Not answered.	42
problem judging this - not done	5

Appendix C

Software



GRiST Elicitation and Assessment Tools

In this section, we give an overview of the various software components in the GRiST system.

- GRiST decision support system
- Data visualisation
- Tuning expertise
- Reports

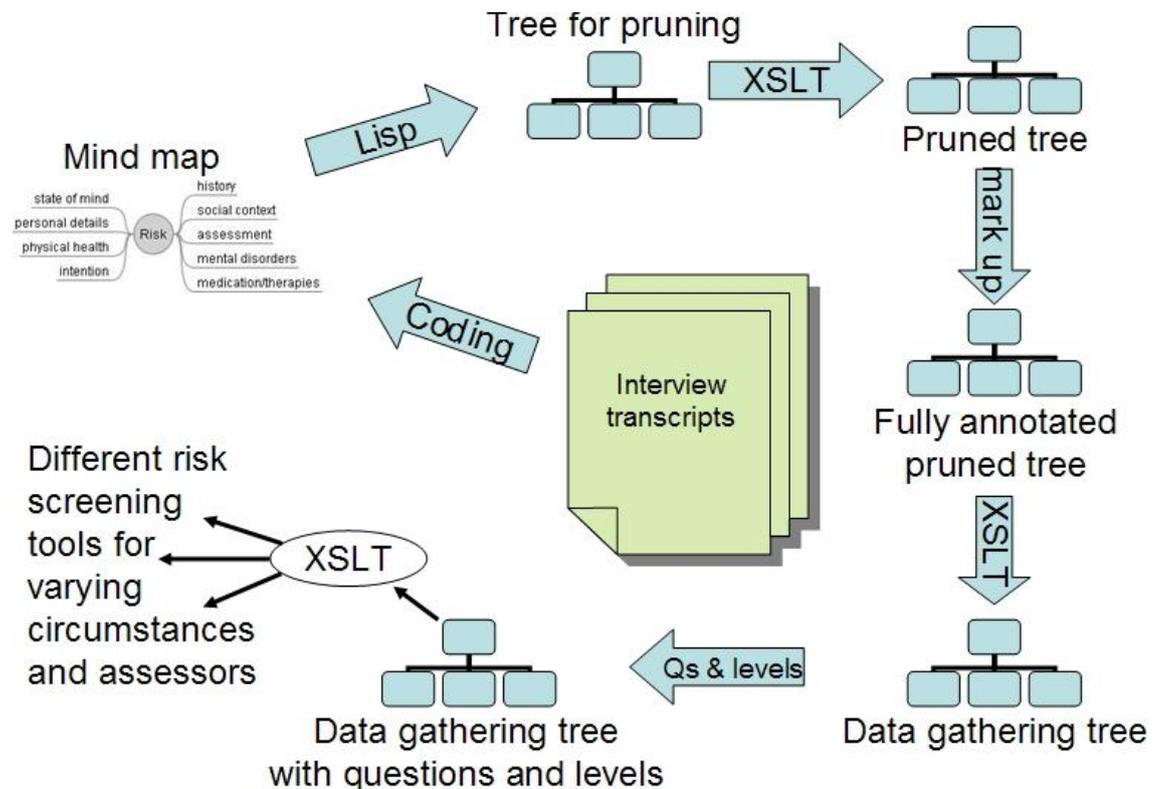


Figure 1: The conceptual model for the Online GRiST system, including the various tools and their interaction.

Elicitation Tools:

These are the tools that allow experts to tune the system, specify various parameters, modify the questions and even restructure the tree, based on the application domain. We have designed these tools as part of this work using Macromedia Flash. They are Web-based and available for clinicians to use. For more information, visit: www.eGRiST.org

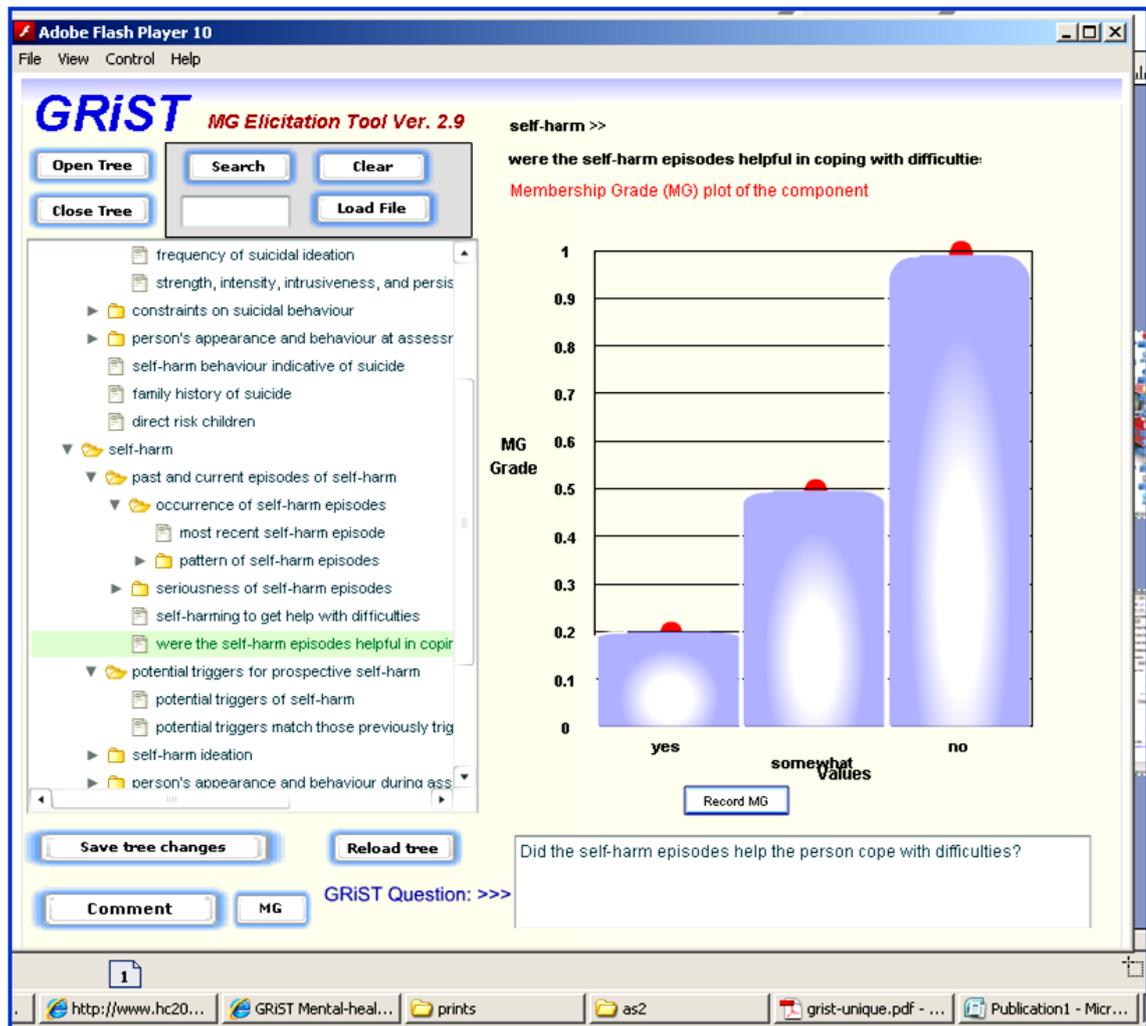


Figure 2: The Flash MG Elicitation tool, here showing the Membership Grade of one of the Self-Harm components being entered.

These tools provide an intuitive visual interface for clinicians to use, and the inputs are then translated into the tree parameters. It is a very important part of the project, as data gathering is usually one of the challenges in any expert system and the Human Computer Interface (HCI) could affect the success of the trials.

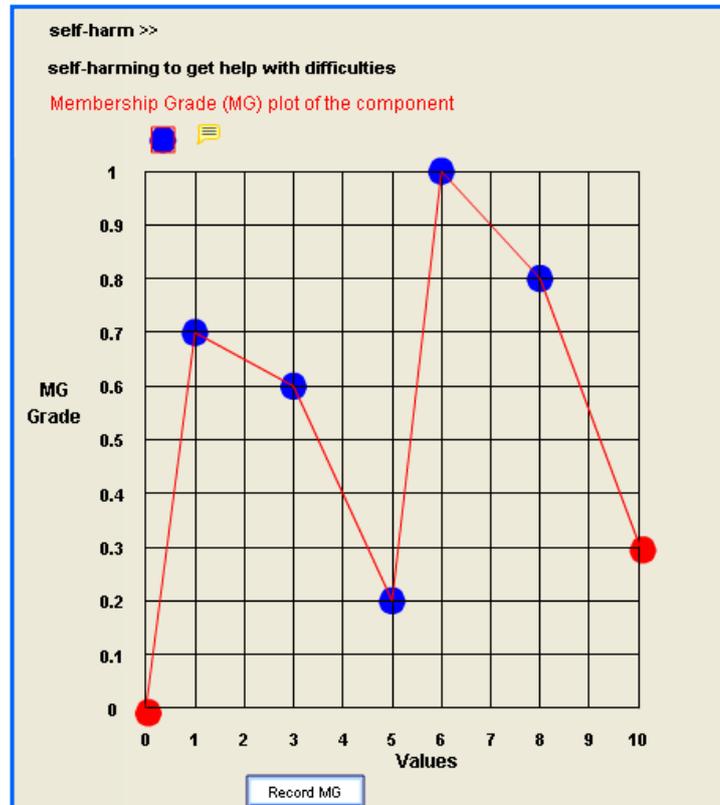


Figure 3: Another type of MG Graphs supported by the tool, the Scale, where users can drag the points around, add or delete new points in order to create the best representation of the MG function.

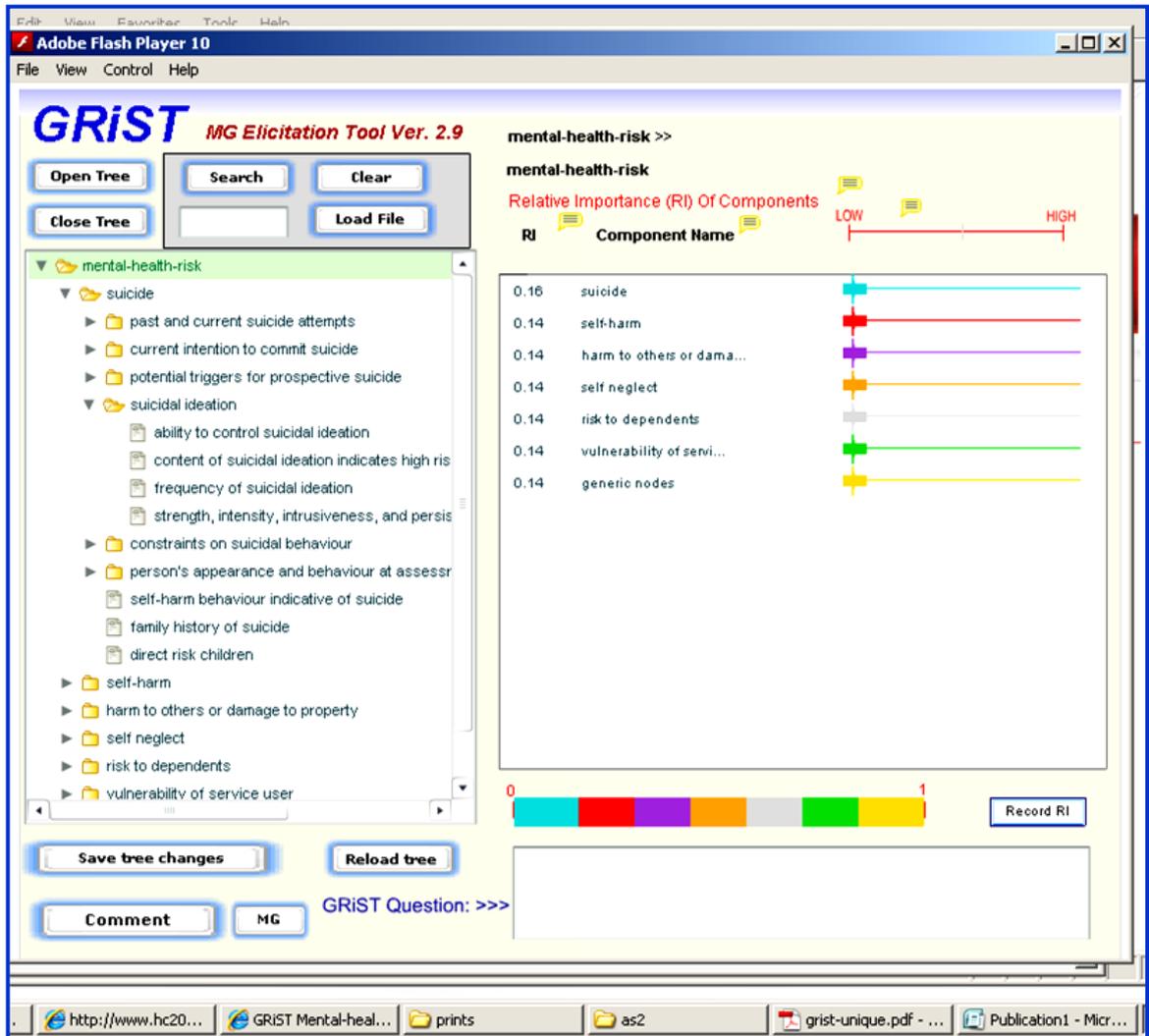


Figure 4: The Flash Elicitation tool, here used to elicit Relative Influence (RI) Values manually.

Data Gathering Tools:

These are the tools used by clinicians to actually enter patients, interview questions, and maintain the eHealth records. These are all online and web-based, which makes it very accessible. They also allow users to complete the questionnaires in more than one session, which is more natural. The inputs are saved and processed, and initial risks are displayed based on the data entered so far.

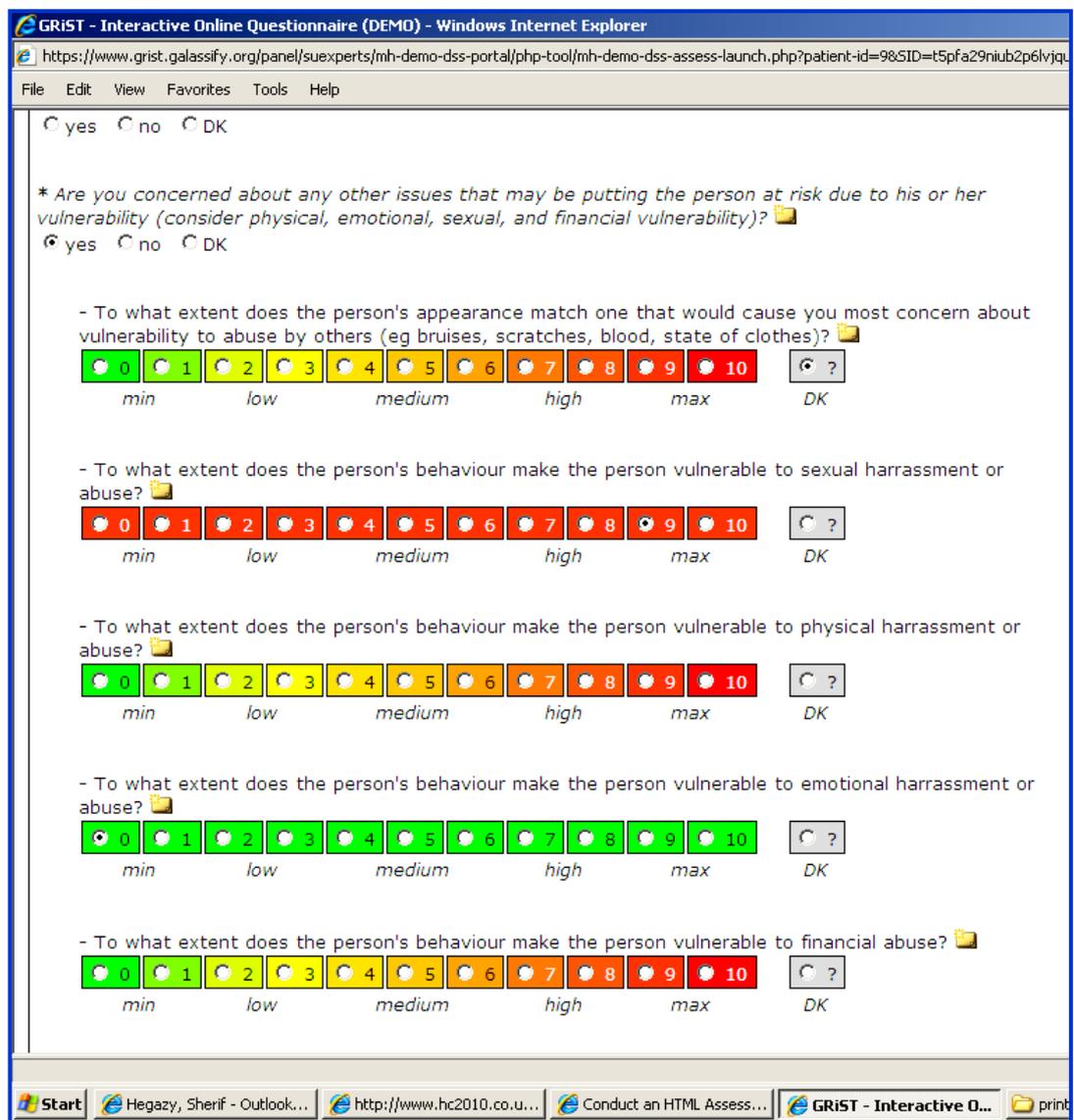


Figure 5: An example of the online (live) questionnaire or interview.

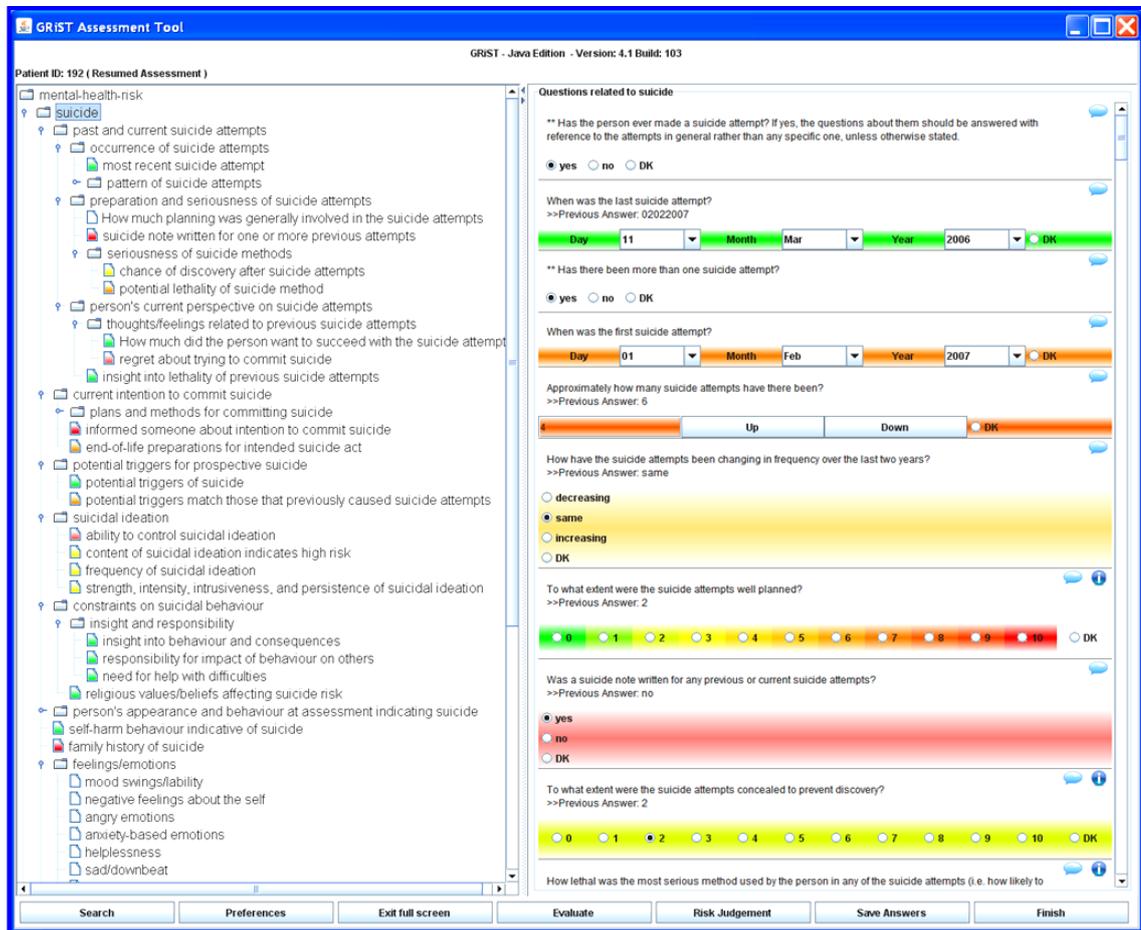
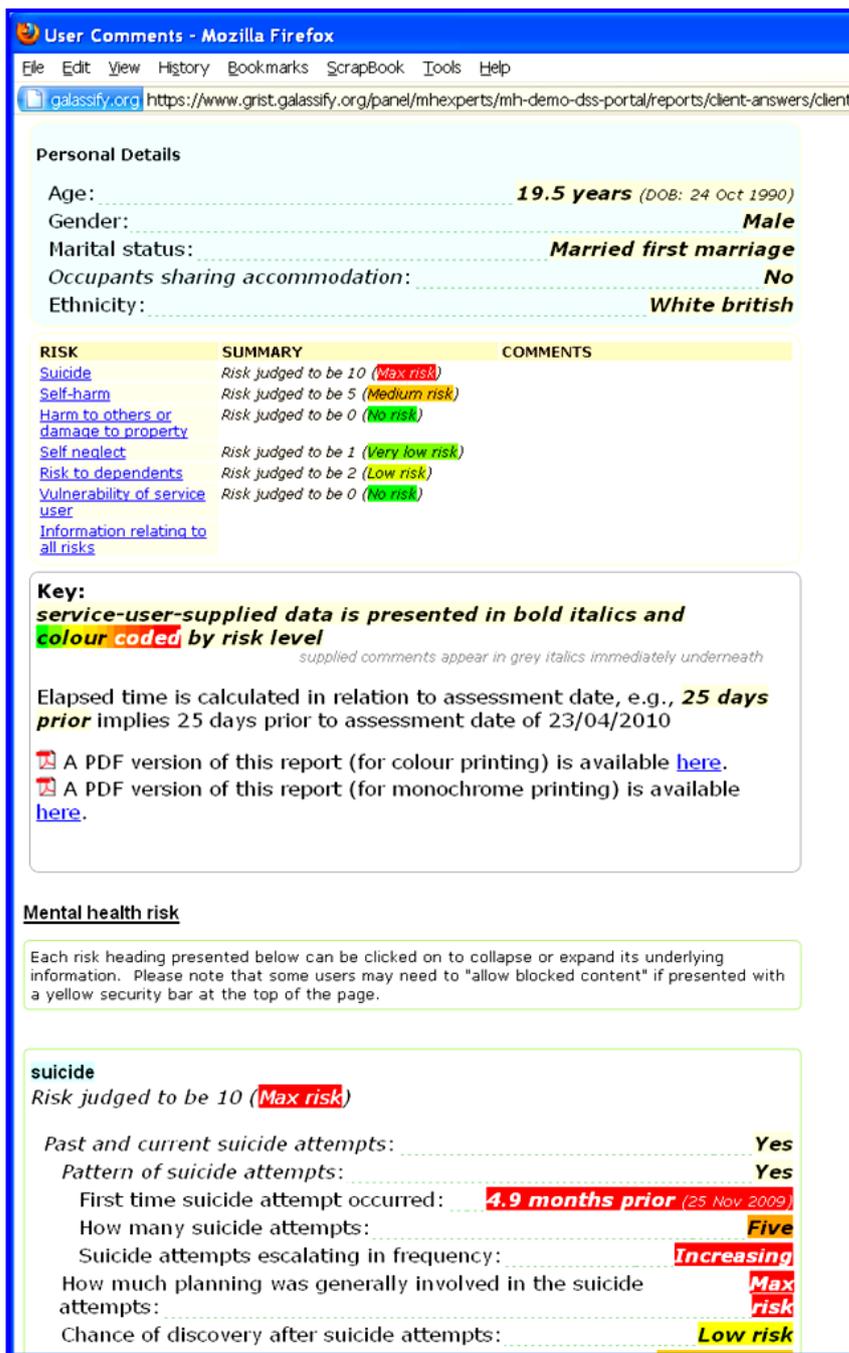


Figure 6: A screen shot from the Java tool, showing the GRIST Tree (left) and the questions associated with one of the nodes (Suicide in this case).

Reporting Tools:

These generate a variety of reports and summaries on a patient, and include risk assessments and results.



User Comments - Mozilla Firefox
 File Edit View History Bookmarks ScrapBook Tools Help
 galassify.org https://www.grist.galassify.org/panel/mhexperts/mh-demo-dss-portal/reports/client-answers/client-

Personal Details

Age: **19.5 years** (DOB: 24 Oct 1990)
 Gender: **Male**
 Marital status: **Married first marriage**
 Occupants sharing accommodation: **No**
 Ethnicity: **White british**

RISK	SUMMARY	COMMENTS
Suicide	Risk judged to be 10 (Max risk)	
Self-harm	Risk judged to be 5 (Medium risk)	
Harm to others or damage to property	Risk judged to be 0 (No risk)	
Self neglect	Risk judged to be 1 (Very low risk)	
Risk to dependents	Risk judged to be 2 (Low risk)	
Vulnerability of service user	Risk judged to be 0 (No risk)	
Information relating to all risks		

Key:
service-user-supplied data is presented in bold italics and colour coded by risk level
supplied comments appear in grey italics immediately underneath

Elapsed time is calculated in relation to assessment date, e.g., **25 days prior** implies 25 days prior to assessment date of 23/04/2010

A PDF version of this report (for colour printing) is available [here](#).
 A PDF version of this report (for monochrome printing) is available [here](#).

Mental health risk

Each risk heading presented below can be clicked on to collapse or expand its underlying information. Please note that some users may need to "allow blocked content" if presented with a yellow security bar at the top of the page.

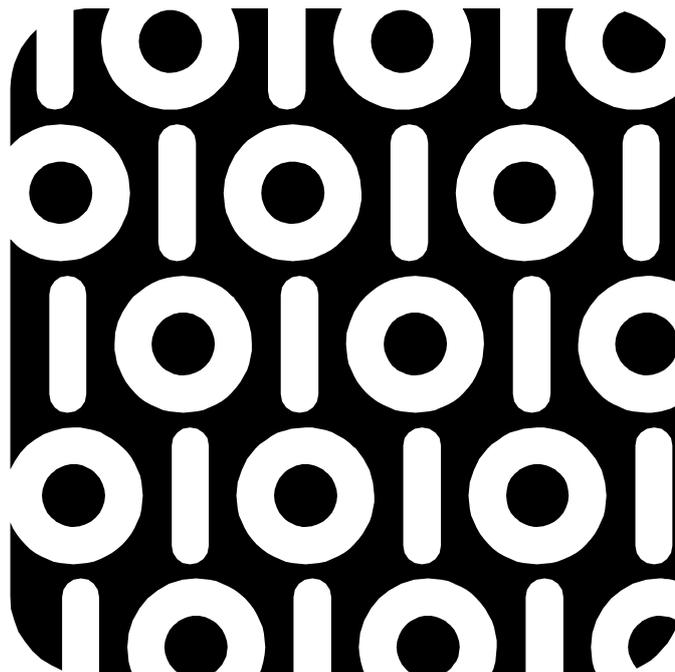
suicide
 Risk judged to be 10 (**Max risk**)

Past and current suicide attempts: **Yes**
 Pattern of suicide attempts: **Yes**
 First time suicide attempt occurred: **4.9 months prior** (25 Nov 2009)
 How many suicide attempts: **Five**
 Suicide attempts escalating in frequency: **Increasing**
 How much planning was generally involved in the suicide attempts: **Max risk**
 Chance of discovery after suicide attempts: **Low risk**

Figure 7: A sample Patient report, which shows the various risk ratings in colour codes scale (Red means high risk, Green means no risk).. This report can be generated online and saved as Pdf version as well.

Appendix D

Code



GRiST Functions in MATLAB.

This appendix lists some of the functions we wrote in MATLAB to implement the algorithms and methods presented in this work.

In total 54 functions were coded.

The main tasks performed using these function are:

1. Reading the data from the database.
2. Preparing and conditioning the data.
3. Substituting missing values (APIC Algorithm).
4. Calculating the parameters (iARRIVE Algorithm).
5. 10-Fold Cross validation on the data set.

Handling missing Data:

The problem of missing data is inherent in any application that involves questionnaires or large amounts of data entered by the users. A wider problem would be the accuracy of the input data and how much noise it contains. In the specific case of GRiST [5], with in excess of 200 questions, it is inevitable that some of the data will be missing (i.e. not entered/ answered) and this is evident in the data currently being collected during clinical use. This causes a difficulty with the model we previously developed for learning RI values (ARRIVE [6] and iARRIVE [7]) because the algorithms depend on complete data sets.

In the case of GRiST , there are two type of missing values in the data set (see Fig. 1):

1. Missing values due to input filtration (in Red).
2. Missing values due to human error or incomplete assessment (in Blue).

The first type occurs naturally, due to what we call Filter Questions. In the GRiST system [4], the knowledge is represented as a tree, and at some nodes (called Filter nodes), a decision is made whether to branch to a sub tree or not. If the decision was not to branch, then all the values in the subtree connected to the no answer will be blank.

These values have to be treated as zeros and not as missing values. To overcome this problem, we have developed a series of functions to traverse the subtrees of nodes that are connected to Filter nodes and zero the values if the Filter node contains a 'no' for example.

It is type 2 of missing values that poses the bigger problem. Since there is no way of predicting them from the semantics of the tree directly as in type 1, we need to develop a method to be able to substitute these values with viable alternatives, that can be justified.

The work examines the practical considerations we faced when implementing an algorithm we developed to handle missing data so that parameter learning can be affected and also shows how the method can help with analysing errors in output.

Data Structures and Functions

In this section we analyse the required functions and data structures to implement APIC in MatLab [12] or any other programming language. We opted for MatLab due to the flexibility and the support for matrix operations, which we require intensively in our application domain.

The pseudo code is listed in Listing 1.

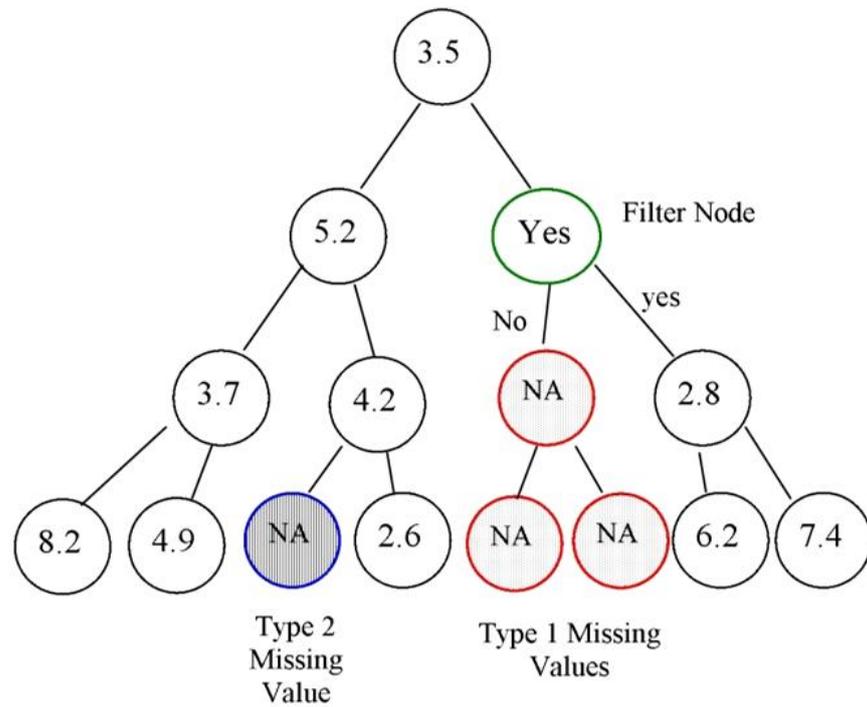


Figure 1: Types of missing inputs.

Data Structures

Matrices: $M()$, $m()$, $m'()$, $C()$: these contain the input Matrix with all the cases in rows, sub-matrix with complete (non-missing) values of the variable in question, sub matrix with only missing values of the variable in question, and a sub-matrix with all complete rows only, respectively.

These are illustrated in Figures 3 to 5 for variable A based on input matrix $M()$ in Figure 2.

A	B	C	D
1	2	4	53
2	3	7	50
3	6	6	52
4	7	12	48
5	11	17	33
6	13	21	29
7	12	19	44
8	13	26	52
9	20	29	18
4	9	11	26
7	14	9	33
2	5	8	
8	17		37
5	13	15	43
7	12	19	51
	6		24
9		20	44
6	15	9	
4	9	8	20
1		4	48
	5	8	41
4	10	12	49
	15	6	
7	11	26	46
5	12		29
	5	10	30
6	11		
9	17	20	39
1	3		29

Figure 2: Original inputs with missing data, M().

A	B	C	D
9	20	29	18
4	9	11	26
7	14	9	33
2	5	8	
8	17		37
5	13	15	43
7	12	19	51
9		20	44
6	15	9	
4	9	8	20
1		4	48
4	10	12	49
7	11	26	46
5	12		29
6	11		
9	17	20	39
1	3		29

Figure 3: Matrix m(A)

The data was extracted from the MySQL [] database in a CSV (Comma Separated Values) file, which was then read into MatLab, and pre-processed to convert to the correct data types.

We had to used several type conversion functions in MatLab, such as CELLSTR, EVAL, to convert between string and numerical or FLOAT data types. This is due to the way the data was stored in the MySQL data base from the interface originally [13].

MatLab allows for dynamic matrix resizing, which allows us to slice or shrink the matrices as required.

This was extremely important for creating $C()$, $m()$, $m'()$ from $M()$ as in the pseudo-code in Listing 1.

A	B	C	D
	6		24
	5	8	41
	15	6	
	5	10	30

Figure 4: Matirx $m'(A)$

A	B	C	D
9	20	29	18
4	9	11	26
7	14	9	33
2	5	8	
5	13	15	43
7	12	19	51
6	15	9	
4	9	8	20
4	10	12	49
7	11	26	46
9	17	20	39

Figure 5: Matrix $C(B,C)$ for variable A.

Matrix PT (Prediction Table)

This will contain the Prediction functions for each variable. This can be represented as a 3D matrix, with rows represented different variable, and columns representing different combinations.

The number of rows is equal to the number of question nodes (inputs), so is the depth (Z-axis). The number of columns will equal the maximum number of combinations. We can set this to 1000, as not all combinations will be stored, only the ones needed for prediction. A sample table for variable A is shown in Table 3. This contains the Coefficients of the regression processes, i.e. the functions that describe A in terms of B, C, and/or D.

The problem with this approach is that there will be quite a bit of waste in memory, due to the difference in the number of combinations from one variable to another.

A better solution would be a 2-D matrix, with number of columns equal the number of leaf nodes and the rows are dynamic. Each row will represent PF (prediction function) coefficients for a certain input (leaf node). The input leaf node number for the PF can be stored in the first cell of a row, e.g. PT(i,0).

This way, each variable, will have several PF. So PT(l,0) is not unique and a simple search function should be available to search PF for a certain variable in PT. Note that PF for a certain variable will usually be consequent, as PT is populated through the Pseudo code above, per variable.

A sample table for variable A is shown in Table 1. We used the CELL structure in Matlab to be able to fit various data types in the table. And allow dynamic resizing at the same time.

We also had to used type conversion functions, CELLSTR, EVAL, to convert between string and numerical or FLOAT data types. This is due to the way the data was stored in the MySQL data base.

Functions

In practical terms, we will use shortcuts in generating the prediction table, in order to avoid exponential explosion of the number of combinations.

The algorithm will start with the required combinations first. These are the combinations of other variables that are available in a certain case where the variable in question is missing. This is done using the function **Find_Combinations(m,i), Sub_Missing (m, i)**.

We call these the Blue Prints. These Blue Prints are then tested to make sure there are enough complete Blue Prints to generate a Prediction Function (PF) for the variable in question to add to the Prediction Table(PT). The function that does that is **Sub_Complete(m,i)**

If a PF for certain combination could not be obtained, due to not enough complete training cases for that combination, the next best possible combination is checked (i.e. , with less variables), until a suitable PF is found above the threshold (R^2 Determination Coefficient) .

PT(I,0)	PT(I,1)	PT(I,2)	PT(I, 3)	PT(I,4)	PT(I,5)
	Coeff(A)	Coeff(B)	Coeff(C)	Coeff(D)	Const
A	0	0.419	0.078	0.033	-1.607
A	0	0.521	0	0.045	-2.043
A	0	0.346	0.107	0	-0.002
A	0	0.475	0	0	0.152
A	0	0	0.285	0	1.135
A	0	0	0	-0.042	6.964

Table 1: The Prediction Table(PT), for variable A, PT(A).

APIC

Input Matrix $\mathbf{M}(x=1 \text{ to } k, y=1 \text{ to } n)$

Start

For each missing input $x(i)$ *Thus only dealing with missing variables columns*

Find sub matrix with complete column (i)

$\mathbf{m}(i) = \text{Sub_Complete}(i)$ *Matrix containing only rows of $M(\)$ with I present*

$\mathbf{m}'(i) = \text{Sub_Missing}(i)$ *Matrix containing only rows of $M(\)$ with I missing*

For each variable combination with missing value (i) in \mathbf{m}' *(i.e. only few combinations)*

Find a complete set with that combination and I in $\mathbf{m}(i)$.

For $t=1$ to $k-1$ *(t denotes the combination variables for incomplete \mathbf{m}')*

$\mathbf{C}(\) = \text{Sub_Complete}(t)$ applied to $\mathbf{m}(i)$

After all iterations, this matrix $\mathbf{C}(\)$ will only contain complete rows with I and all other variables in this combination. $\mathbf{C}(\)$ is a sub matrix of $\mathbf{m}(i)$

If $\text{Row_Count}(\mathbf{C}) \geq$ number of variables in combination

Find another PF (Prediction Function) and add to PT
(Prediction Table of i)

PF = Regression (C)

If R^2 (PF) > predefined threshold Then

{Determination Coefficient}

PT(i)+= PF *(Add this function to the table of i)*

Find missing value of I in $\mathbf{m}'(i)$ using new PF for

that combination.

If $\text{Sub_Complete}(\mathbf{m}'(i)) \neq \mathbf{m}'(i)$ *i.e. some combinations were not above threshold*

Iterate on missing I values using best possible PF in PT(i)

Listing 1: Pseudo-code for APIC

MatLab Code:

```
function [RI_Names, RI]= GRiST(O,MM,xml_file,filter_nodes)

[M,nulls]=decode_Data(MM,filter_nodes,xml_file);

'Input Data Decoded successfully.....'

[m,p,pc,pb]=APIC(M);

'Missing Data replaced successfully.....'

R =RI_BluePrint(xml_file);

'RI Blue Print generated successfully...'

RI=iarrive(m,O,R);

'RI Values successfully calculated...'

RI_Names=RI_Codes(xml_file);

'RI Codes order generated successfully.....'
```

```
function [o, data_matrix, nulls]= decode_Data(CSV_matrix, xml_file)

% m is a matrix containing the leaf node data. First Column contains
% assessment date which is required to decode MG for date values.
% filter_nodes: is a matrix containing Filter nodes data.
% xml_file: is the name of the CAT xml file which includes MG and MG
value
% attributes.
% This function will do two things:
% 1. Replace -1 (missing values ) with 0 if their filter node was 0.
% 2. Replace leaf nodes with decoded MG values.

% nulls returns the number of missing values in the data set for
debugging.

% to get the Filter nodes from data base in the correct order, use:
% getFilterCodes(xml)

% to get Leaf nodes from DB in the correct order, use:
% getLeafCodes(xml)

% the following two functions parse the input data, as MATLAB didn't
quite
% understand the CSV files!!

[o,m,filter_nodes]=readCSV(CSV_matrix);

'CSV matrix read successfully.....'

mg= cell(1,3);

tree=parseXML(xml_file);
'XML Tree parsed, ok.....'

mg=getMG(tree,mg);

'MG Lookup table generated.....'

fp=[0 0 0];

fp=getFilterPattern(tree,fp,0);

'Filter Node Patterns constructed successfully....'

% This section Zeros all children of Filter nodes that equal 0.

nulls=0 ;

'Puning subtrees based on Filter node values.. Counter shows case no.'
```

```

for i=1:size(m,1)
    i
    for k=1:size(filter_nodes,2)
        if strcmp(filter_nodes(i,k),'no')==1
            for t=2:size(fp,2) % this should equal size(m,2)+1
                if fp(k,t)=='1'
                    m(i,t)='0'; %Filter Matrix starts at 2 (First
row 0's)
                end;
            end;
        end;
    end;
end;

'Filter child nodes successfully pruned.....'

% This section decodes all leaf values based on the MG tree.

'Decoding Data based on MG functions... Counter shows case number.'
for i=1:size(m,1);
    i
    % assessment_date = m{i,1};
    for t=2:size(m,2) % MG Table starts at 2, first row is 0's.

        if strcmp(m(i,t),'\N')==1 || strcmp(m(i,t),'DK')==1
            nulls=nulls+1; % just counting the number of missing
values.
            m(i,t)=cellstr('-1');
            end;
            if strcmp(m(i,t),'-1')~=1

                m{i,t}=decode_MG(mg,t,m{i,t},Convert_date(m{i,1})); % to
be replaced with assessment date
            end;
        end;
    end;
end;
'All data values decoded successfully using MG table.....'

data_matrix=m;
data_matrix=Convert_Data(m); % Returns numeric matrix
o=Convert_Data(o);

```

```
function [Filter_matrix] = getFilterPattern(tree, Filter_matrix,
offset)

% This function creates the patterns of Filter node children (i.e.
which
% children are connected to each filter node. This is used to
eliminate missing data due to Filter nodes.

padding=0;

filter=0;
children_count= size(tree.Children, 2);

for yy=1:size(tree.Attributes,2)
    if strcmp(tree.Attributes(1,yy).Name, 'filter-q')== 1
        filter=1;
    end;
end;

x= getLeaf(tree,0);

if filter==1
    Filter_matrix= add_Filter(padding, x, Filter_matrix, offset);
end;

if children_count>0
    for y=1:children_count

        if strcmp(tree.Children(1,y).Name, 'node') == 1

            Filter_matrix=getFilterPattern(tree.Children(1,y),
Filter_matrix, padding+offset);
            x= getLeaf(tree.Children(1,y),0);
            padding=padding+ x ;

        end;

    end;
end;
```

```
function [MG] = add_MG(mg, MG_Type, MG_String)

% this function adds a new MG function to the decoding table MG.
% note : the first row is empty, so indexing starts at 2.
% this corresponds to input Data matrix, as the first column will
include
% the date of the assessment, which need not be decoded.

MG=mg;

current_row= size(MG,1)+1;

MG{current_row,2}= MG_Type;

    curr_col=3;
    curr_val='';
    for i=1:size(MG_String,2)
        if MG_String(1,i)~='(' && MG_String(1,i)~=' ' &&
MG_String(1,i)~=')' '
            curr_val=strcat(curr_val,MG_String(1,i));
        else
            if MG_String(1,i)==' ' || MG_String(1,i)==' '

                MG{current_row,curr_col}=curr_val;

                curr_val='';
                curr_col= curr_col+1;
            end;
        end;
    end;

% store the number of pairs in MG, in the first column of the
matrix,

MG{current_row,1}= (curr_col-4)/2;
```

```

function [mg_value] = decode_MG(mg_table, row, input, date)

% This function substitutes a leaf node value with the decoded value
based
% on the MG table (like Fuzzy logic).

row=row+1; %MG table has an empty row at start,

MG_Type= mg_table{row,2};

mg_value=0;

total_pairs = mg_table{row,1};

if strcmp(mg_table{row,2}, 'nominal')==1
    for i=3:2:(2+ total_pairs*2)
        if strcmp(mg_table{row,i},input)==1
            mg_value= mg_table{row,i+1};
        end;
    end;
end;

if strcmp(mg_table{row,2}, 'scale')==1
    for i=3:2:(total_pairs*2)
        if (eval(mg_table{row,i})<= eval(input)) &&
(eval(mg_table{row,i+2})> eval(input))
            mg_value= Curve(eval(input),mg_table{row,i},
mg_table{row,i+1}, mg_table{row,i+2}, mg_table{row,i+3});
        end;
    end;
end;

if strcmp(mg_table{row,2}, 'date-year')==1
    input=correct_Date(input);
    input=Calc_years(input, date);
    for i=3:2:(total_pairs*2)

        if eval(mg_table{row,i})<= input &&
eval(mg_table{row,i+2})> input

            mg_value= Curve(input,mg_table{row,i},
mg_table{row,i+1}, mg_table{row,i+2}, mg_table{row,i+3});
        end;
    end;
end;

if strcmp(mg_table{row,2}, 'date-week')==1
    input=correct_Date(input);
    input=Calc_weeks(input, date);
    for i=3:2:(total_pairs*2)

```

```
        if eval(mg_table{row,i})<= input &&
eval(mg_table{row,i+2})> input

            mg_value= Curve(input,mg_table{row,i},
mg_table{row,i+1}, mg_table{row,i+2}, mg_table{row,i+3});

            end;
        end;
    end;

    if strcmp(mg_table{row,2}, 'date-month')==1
        input=correct_Date(input);
        input=Calc_months(input, date);
        for i=3:2:(total_pairs*2)

            if eval(mg_table{row,i})<= input &&
eval(mg_table{row,i+2})> input

                mg_value= Curve(input,mg_table{row,i},
mg_table{row,i+1}, mg_table{row,i+2}, mg_table{row,i+3});

                end;
            end;
        end;
```

```

function [MM,PT,PTC,PTB] = APIC(M)

% The main APIC algorithm to replace all missing values in M.
% missing values are represented as -1 in M.
% all inputs are positive or zero in the GRiST tree otherwise.
% Inputs mean questionnaire answers, and each of them is a Column in M.

MM=M;

PT= zeros(1,3);
PTC= zeros(1, size(MM,2));
PTB= zeros(1, size(MM,2));

for i=1:size(MM,2)
    t=missing(MM,i);
    if t==1 % there are missing values at Column i.
        m= Sub_complete(MM,i);
        mn= Sub_missing(MM,i);
        BPT= Find_Combinations(mn); % find all required input
combinations corresponding to missing values of i.

        if sum(m(1,:))>0 % Check is there is a complete
matrix for that input, i.
            [PT, PTC, PTB]= Create_PT(m,BPT,i,PT, PTC, PTB);
        end;
    end;
end;

% Check if PT not empty, i.e., predictions were found and there was
% sufficient data for regression, this is an extreme case.
% all Prediction functions were found, now replace missing inputs
using PT,

if sum(PTB(1,:))>0
    for t=1:size(MM,2)
        MM= Replace_missing(MM,t,PT,PTC, PTB);
        if size(Sub_complete(MM,t),1)~= size(MM,1) % there are
still missing values in Column t,
            % go to the second best ,
            MM=Replace_missing_alternative(MM,t,PT, PTC, PTB);
        end;
    end;
end;
end;

```

```
function [BPT] = Find_Combinations(mn)

% Find all the different Blue Ptint Combinations in an incomplete sub
matrix mn,

comb_no= 1;
% eml. varsize(BPT);

BPT= zeros(1, size(mn,2));

for h=1:size(mn,1)
    BP= Find_BluePrint(mn(h, :));
    found=0;
    if comb_no>1
        found=Check_BPT(BPT,BP);
    end;
    if found==0
        BPT(comb_no,:)=BP;
        comb_no= comb_no+1;
    end;
end;

function [BP]= Find_BluePrint(current_row)

% This function returns the Blue Print vector (BP) of a row in a
matrix, i.e. a binary
% representation of the row, with 1 representing a present value and 0
% representing a missing value. This will then be used in comparison
and
% finding the required variable combinations for prediction.

BP = current_row;
for i=1:size(current_row,2)
    if BP(i)==-1
        BP(i)=0;
    else BP(i)=1;
    end;
end;

function [PT, PTC, PTB]= Create_PT(m,BPT,i,PT, PTC,PTB)
```

```

% This function adds new Prediction functions of the current i to the
% existing Prediction Table PT.
% PTC is the Coefficients table, holding regression results.
% PTB is the Blue Print Table, holding blue print BP corresponding to
Coefficients.
% BPT is the Blue Print Table, holding all Blue Prints for missing
values
% for current i. These will form the regression equations.

    min_threshold = 0;    % Threshold for quality of fit, set by user,
R2.

    if sum(PTB(1,:)) == 0    % if this is the first iteration, adjust
index to fill first row.
        PF_Count=0;        % as PT tables are initialized with zeroes.
    else
        PF_Count= size(PT,1);
    end;

    for f=1:size(BPT,1)
        BP=BPT(f,:);
        C = Complete_BP(m,BP);
        if size(C,1) > sum(BP) && sum(C(1,:)) > 0    % check if
number of rows > number of variables, i.e. regression possible.
            [a,b,R2]= Calc_Regression(C,i,BP);
            if abs(R2) >= min_threshold

                PF_Count= PF_Count+1;

                PT(PF_Count, 1)= i;
                PT(PF_Count, 2)= b;
                PT(PF_Count, 3)= R2;

                PTC(PF_Count,:)=a;
                PTB(PF_Count,:)=BP;
            end;
        end;
    end;
end;

```

```
function [a,b]= Find_PF(i,BP,PT, PTC, PTB)

% Find a Prediction Function PF, for input i, in Prediction Table PT,
with Blue Print BP

a=zeros(1,size(BP,2));
b=0;

for j=1:size(PT,1)
    if PT(j,1)==i
        if BP==PTB(j,:)
            a=PTC(j,:);
            b=PT(j,2);
        end;
    end;
end;
```

```
function [found]= Check_BPT(BPT,BP)

% Check if Blue Print BP exists as a row in the Blue Print Table BPT.

found=0;

for k=1:size(BPT,1)
    if BP == BPT(k, :)
        found=1;
    end;
end;
```

```

function [M] = Replace_missing(m,i,PT, PTC, PTB)

% replace all missing values in i-th Column (input)

M=m;

a=zeros(1,size(m,2));
b=0;

for t=1:size(m,1)
    if m(t,i)==-1
        BP=Find_BluePrint(m(t,:));
        [a,b]= Find_PF(i,BP,PT,PTC,PTB);
        if sum(a)~=0
            M(t,i)= (a* m(t, :))' + b ;
        end;
    end;
end;

function [M] = Replace_missing_alternative(m,i,PT, PTC, PTB)

% replace all missing values in i-th Column (input)

M=m;

a=zeros(1,size(m,2));
b=0;
for t=1:size(m,1)
    if m(t,i)==-1
        BP=Find_BluePrint(m(t,:));
        [a,b]= Find_Alternative_PF(i,BP,PT, PTC,PTB,m);
        if sum(a)>0
            p=m(t,:)' ;
            g= (a*p) + b ;
            M(t,i)=g;
        end;
    end;
end;

% Check if all missing values were replaced, if not delete rows with
% missing values from the matrix.

for t=1:size(M,2)
    M= Sub_complete(M,t);
end;

function [mm]= Sub_complete(m,i)

```

```
% This function returns a sub matrix mm, of m, where all values at the  
i th  
% Column are missing.
```

```
mm=zeros(1,size(m,2));
```

```
Row_count = 1;  
for k=1:size(m,1)  
    if m(k,i)~= -1  
        mm(Row_count, :)= m(k, :);  
        Row_count= Row_count + 1;  
    end;  
end;
```

```
function [mm]= Sub_missing(m,i)
```

```
% This function returns a sub matrix mm, of m, where all values at the  
i th  
% Column are missing.
```

```
mm=zeros(1,size(m,2));
```

```
Row_count = 1;  
for k=1:size(m,1)  
    if m(k,i)== -1  
        mm(Row_count, :)= m(k, :);  
        Row_count= Row_count + 1;  
    end;  
end;
```

```
function [C]= Complete_BP(m, BP)
```

```
% returns a matrix C , with complete columns corresponding to 1's in  
BP.
```

```
% This will be used for Regression.
```

```
C=m;
```

```
for q=1:size(BP,2)  
    if BP(q)==1  
        C=Sub_complete(C,q);  
    end;  
end;
```

```
function [MG] = getMG(tree, MG_matrix)

% #This function creates the MG lookup tables, by reading all the MG
values
% at Leaf nodes.

total_leafs = getLeaf(tree,0);

padding=0;
MG_Type='';
MG_String='';

MG=MG_matrix;

children_count= size(tree.Children, 2);

if children_count>0
    for y=1:children_count
        if strcmp(tree.Children(1,y).Name, 'node') == 1
            MG=getMG(tree.Children(1,y), MG);
        end;
    end;
else
    % No children, this is a leaf node, get the MG attributes and add
to MG
    % look up table.

    %MG= add_MG(padding, x, total_leafs, MG);
    for t=1:size(tree.Attributes,2)
        if strcmp(tree.Attributes(1,t).Name, 'values')==1
            MG_Type=tree.Attributes(1,t).Value;
        end;
        if strcmp(tree.Attributes(1,t).Name, 'value-mg')==1
            MG_String=tree.Attributes(1,t).Value;
        end;
    end;

    MG=add_MG(MG,MG_Type,MG_String);
end;
```

```
function [R] = RI_BluePrint(ss)

% This function returns the required tree description for the GRiST
% function. It represents the relation ship between RIs and the leaf
% nodes connected to them in a binary
% format.

Tree=parseXML(ss);

RI= zeros(1,getLeaf(Tree,0));

R=getRI(Tree,RI,0);

function [RI] = getRI(tree, RI_matrix, offset)

total_leafs = getLeaf(tree,0);

padding=0;
RI=RI_matrix;

children_count= size(tree.Children, 2);

for y=1:children_count
    if strcmp(tree.Children(1,y).Name, 'node') == 1

        x= getLeaf(tree.Children(1,y),0);
        RI= add_RI(padding, x, total_leafs, RI, offset);

        RI=getRI(tree.Children(1,y), RI, padding);
        padding=padding+ x ;

    end;
end;
```

```
function [RI]= iarrive(M,O,R)

% function to calculate RI values using the inputs matrix M and
% outputs or
% risk decisions O
% R matrix contains the tree structure represented in binary, for each
% child subtree, all leaf nodes are listed, followed by the subtree of
% that
% child's node.
% It is binary, i.e. 0 means not connected, 1 connected, and the R
% array is
% 2D.

A= inv(M' *M)* M' * O;

for i=1:2:size(R,1)-1
    denom=0;
    nom=0;
    for k=1:size(R,2)
        if R(i,k)== 1
            nom=nom +A(k);
        end;
        if R(i+1,k) == 1
            denom=denom +A(k);
        end;
    end;

    RI((i+1)/2)= nom/ denom;

end;
```

```
function [RI] = getRI_Codes(tree, RI_matrix)

% Generate a list of RI codes to be used for printing the results
% (i.e. RI
% values).

RI=RI_matrix;

children_count= size(tree.Children, 2);

for y=1:children_count
    if strcmp(tree.Children(1,y).Name, 'node') == 1

        for k=1:size(tree.Children(1,y).Attributes,2)
            if strcmp(tree.Children(1,y).Attributes(1,k).Name,
'code')== 1
                string = tree.Children(1,y).Attributes(1,k).Value;
                RI= add_RI_Code(string, RI);

            end;
        end;

        RI=getRI_Codes(tree.Children(1,y), RI);

    end;
end;
```

```
function [O,E]= GRiSTCross(data,text, xml_file)

[O,MM]=decode_Data(data, text, xml_file);

'This function performs 10-fold cross validation on the GRiST data.'

E=0;

R =RI_BluePrint(xml_file);

for i=1:70:700
    test= MM([i:i+70],:);
    train= cat(1, MM([1:i],:), MM([i+70:700],:));
    Ot= cat(1, O([1:i],:), O([i+70:700],:));

    [RI, error,A]=iarrive(train,Ot,R);

    K= test*A;

    E([i:i+70],:)= K;

i
end;
```

The Suicide Tree Description in XML

```

- <node label="suicide" code="suic" values="scale" value-mg="((0 0)(10 1))">
- <node label="suicide specific questions" code="sui-specific">
- <node label="past and current suicide attempts" code="suic-past-att" layer="0"
  filter-q="" ERROR="This node is persistent; however the following ancestor
  nodes are not: sui-specific">
- <node label="occurrence of suicide attempts" code="suic-occur">
  <node label="most recent suicide attempt" code="suic-most-rec" values="date-
  week" value-mg="((0 1)(26 0))" ERROR="This node is persistent; however
  the following ancestor nodes are not: sui-specific" />
- <node label="pattern of suicide attempts" code="suic-patt-att" values="scale"
  value-mg="((0 0)(10 1))" filter-q="">
  <node label="first time suicide attempt occurred" code="suic-first-occ"
  values="date-year" value-mg="((0 1)(10 0))" ERROR="This node is
  persistent; however the following ancestor nodes are not: suic-patt-att" />
  <node label="how many suicide attempts" code="suic-how-many"
  values="integer" value-mg="((0 0)(2 1)(8 0.2))" />
  <node label="suicide attempts escalating in frequency" code="suic-escalate"
  values="nominal" value-mg="((DECREASING 0)(SAME 0.5)(INCREASING
  1))" />
  </node>
  </node>
- <node label="preparation and seriousness of suicide attempts" code="suic-
  prep-serious-at" values="scale" value-mg="((0 0)(10 1))">
  <node label="How much planning was generally involved in the suicide
  attempts" code="suic-planning" values="scale" value-mg="((0 0)(10 1))"
  ERROR="This node is persistent; however the following ancestor nodes are
  not: sui-specific" />
  <node label="suicide note written for one or more previous attempts"
  code="suic-note-prev" values="nominal" value-mg="((YES 1)(NO 0))"
  ERROR="This node is persistent; however the following ancestor nodes are
  not: sui-specific" />
- <node label="seriousness of suicide methods" code="suic-ser-method">
  <node label="chance of discovery after suicide attempts" code="suic-
  discovery" values="scale" value-mg="((0 0)(10 1))" ERROR="This node is
  persistent; however the following ancestor nodes are not: sui-specific" />
  <node label="potential lethality of suicide method" code="suic-lethality"
  values="scale" value-mg="((0 0)(10 1))" ERROR="This node is persistent;
  however the following ancestor nodes are not: sui-specific" />
  </node>
  </node>
- <node label="person's current perspective on suicide attempts" code="suic-
  person-per" values="scale" value-mg="((0 0)(10 1))">
- <node label="thoughts/feelings related to previous suicide attempts"
  code="suic-thght-prev">
  <node label="How much did the person want to succeed with the suicide
  attempts" code="suic-ser-succd" values="scale" value-mg="((0 0)(10 1))"
  ERROR="This node is persistent; however the following ancestor nodes are
  not: sui-specific" />

```

```

<node label="regret about trying to commit suicide" code="suic-regret"
  values="scale" value-mg="((0 0)(10 1))" />
</node>
<node label="insight into lethality of previous suicide attempts" code="suic-
leth-insght" values="scale" value-mg="((0 0)(10 1))" />
</node>
</node>
= <node label="current suicidal situation and behaviour" code="suic-curr-sit-
behav">
= <node label="current intention to commit suicide" code="suic-curr-int"
  layer="0">
= <node label="plans and methods for committing suicide" code="suic-plans"
  values="scale" value-mg="((0 0)(10 1))" filter-q="">
<node label="realism of suicide plan" code="suic-plan-real" values="scale"
  value-mg="((0 0)(10 1))" />
<node label="level of detail and clarity of suicide plan" code="suic-plan-dtail"
  values="scale" value-mg="((0 0)(10 1))" />
<node label="physical steps taken to implement suicide plan" code="suic-
steps-takn" values="scale" value-mg="((0 0)(10 1))" />
<node label="potential lethality of prospective suicide method" code="suic-
prosp-leth" values="scale" value-mg="((0 0)(10 1))" />
</node>
<node label="informed someone about intention to commit suicide"
  code="suic-int-inform" values="nominal" value-mg="((YES 1)(NO 0))" />
<node label="end-of-life preparations for intended suicide act" code="suic-eol-
prep" values="scale" value-mg="((0 0)(10 1))" />
</node>
= <node label="potential triggers for prospective suicide" code="suic-int-p-trig"
  values="scale" value-mg="((0 0)(10 1))" layer="0">
<node label="potential triggers of suicide" code="suic-pot-trig" values="scale"
  value-mg="((0 0)(10 1))" />
<node label="potential triggers match those that previously caused suicide
attempts" code="suic-p-trig-mtch" values="scale" value-mg="((0 0)(10 1))"
/>
</node>
= <node label="suicidal ideation" code="suic-ideation" values="scale" value-
mg="((0 0)(10 1))" layer="0" filter-q="">
<node label="ability to control suicidal ideation" code="suic-id-control"
  values="scale" value-mg="((0 0)(10 1))" />
<node label="content of suicidal ideation indicates high risk" code="suic-id-hi-
risk" values="scale" value-mg="((0 0)(10 1))" />
<node label="frequency of suicidal ideation" code="suic-id-freq"
  values="nominal" value-mg="((DAILY 1)(WEEKLY 0.5)(MONTHLY 0.2)(LESS-
THAN-MONTHLY 0))" />
<node label="strength, intensity, intrusiveness, and persistence of suicidal
ideation" code="suic-id-strngth" values="scale" value-mg="((0 0)(10 1))" />
</node>
</node>
= <node label="constraints on suicidal behaviour" code="suic-bhvr-const"
  values="scale" value-mg="((0 0)(10 1))">

```

```

<node label="insight and responsibility" code="insight-resp" generic="generic
  issues >> direct risk children >> insight and responsibility" ERROR="PATH
  INVALID: generic issues >> direct risk children >> insight and
  responsibility" />
<node label="religious values/beliefs affecting suicide risk" code="suic-rel-
  belief" values="nominal" value-mg="((STRONGLY-PROTECT 0)(PROTECT
  0)(NO-EFFECT 0)(INCREASE 0.5)(STRONGLY-INCREASE 1))"
  header="General suicide questions" ERROR="This node is persistent;
  however the following ancestor nodes are not: sui-specific" />
</node>
- <node label="person's appearance and behaviour at assessment indicating
  suicide" code="suic-app-behvr" values="scale" value-mg="((0 0)(10 1))">
- <node label="physical indicators of suicide" code="suic-phys-indic">
- <node label="appearance indicators of self neglect" code="sn-appearance"
  layer="0" generic="g">
  <node label="hair and clothing indicative of self neglect" code="sn-hair-
  clothes" values="scale" value-mg="((0 0)(10 1))" />
  <node label="personal hygiene" code="sn-hygiene" values="scale" value-
  mg="((0 0)(10 1))" generic-type="g" />
  <node label="recent change in appearance of self neglect" code="sn-recnt-app-
  chnge" values="scale" value-mg="((0 0)(10 1))" />
  <node label="skin" code="sn-skin" values="scale" value-mg="((0 0)(10 1))" />
  </node>
  <node label="self-harming cuts" code="gen-sh-cuts" generic-datum="self-harm
  >> person's appearance and behaviour during assessment indicating self-
  harm >> self-harming cuts" ERROR="PATH INVALID: self-harm >>
  person's appearance and behaviour during assessment indicating self-
  harm >> self-harming cuts" />
  </node>
- <node label="person's behavioural presentation during assessment"
  code="gen-presentation" layer="0" values="scale" value-mg="((0 0)(10 1))"
  generic="gd">
- <node label="person's engagement with assessor" code="gen-engagement"
  filter-q="" generic-type="gd">
  <node label="rapport/empathy" code="gen-rapport" values="scale" value-
  mg="((0 0)(10 1))" />
  <node label="person's responsiveness" code="gen-responsve" values="scale"
  value-mg="((0 0)(10 1))" />
  <node label="assessor's uneasiness about the person" code="gen-gut-assmnt"
  values="scale" value-mg="((0 0)(10 1))" />
  </node>
- <node label="verbal indicators of risk" code="gen-risk-verbal" filter-q=""
  generic-type="gd">
- <node label="tone" code="gen-risk-tone">
  <node label="degree of aggression/hostility" code="gen-risk-aggrsv"
  values="scale" value-mg="((0 0)(10 1))" />
  <node label="how upbeat or downbeat/depressed" code="gen-risk-upbeat"
  values="scale" value-mg="((0 0)(10 1))" />
  </node>
  <node label="degree to which the person is making sense" code="gen-
  coherence" values="scale" value-mg="((0 0)(10 1))" />

```

```

</node>
= <node label="body language and expression" code="gen-body-face" filter-q=""
  generic-type="gd">
  <node label="body language indicating distress" code="gen-distrss-b-lang"
    values="scale" value-mg="((0 0)(10 1))" />
  <node label="movements, posture, facial expression indicating low mood"
    code="gen-low-mood" values="scale" value-mg="((0 0)(10 1))" />
  <node label="aggressive/threatening movements, posture, or expression"
    code="gen-threat-move" values="scale" value-mg="((0 0)(10 1))" />
  <node label="preoccupied/detached demeanour" code="gen-detached"
    values="scale" value-mg="((0 0)(10 1))" />
= <node label="eyes" code="gen-eyes" generic-type="gd">
  <node label="avoid eye contact" code="gen-avoid-eye-contact" values="scale"
    value-mg="((0 0)(10 1))" />
  <node label="eye movement" code="gen-eye-movement" values="nominal"
    value-mg="((UNRESPONSIVE-GLAZED 1)(APPROPRIATE 0)(FIXED-STARING
    1)(DARTING 1))" />
  </node>
  </node>
  <node label="congruence of physical, verbal, and emotional presentation"
    code="gen-congruence" values="scale" value-mg="((0 0)(10 1))" />
  </node>
  </node>
  <node label="self-harm behaviour indicative of suicide" code="suic-s-h-behv"
    values="scale" value-mg="((0 0)(10 1))" ERROR="This node is persistent;
    however the following ancestor nodes are not: sui-specific" />
  <node label="family history of suicide" code="suic-fam-hist" values="nominal"
    value-mg="((YES 1)(NO 0))" ERROR="This node is persistent; however the
    following ancestor nodes are not: sui-specific" />
  </node>
= <node label="feelings/emotions" code="gen-feel-emot" layer="0" generic-
  type="gd">
  <node label="mood swings/lability" code="gen-mood-swings" values="scale"
    value-mg="((0 0)(10 1))" />
  <node label="negative feelings about the self" code="gen-negative-self"
    values="scale" value-mg="((0 0)(10 1))" />
  <node label="angry emotions" code="gen-angry-emotns" values="scale" value-
    mg="((0 0)(10 1))" />
  <node label="anxiety-based emotions" code="gen-anx-emotns" values="scale"
    value-mg="((0 0)(10 1))" />
  <node label="helplessness" code="gen-helpless" values="scale" value-mg="((0
    0)(10 1))" />
  <node label="sad/downbeat" code="gen-sad" values="scale" value-mg="((0
    0)(10 1))" />
  <node label="distress" code="gen-distress" values="scale" value-mg="((0 0)(10
    1))" />
  <node label="jealousy" code="gen-jealous" values="scale" value-mg="((0 0)(10
    1))" />
= <node label="hopelessness" code="gen-hopeless" values="scale" value-mg="((0
  0)(10 1))" header="Questions on hopelessness" generic-type="g">

```

```

<node label="plans for the future" code="gen-plans-future" values="scale"
  value-mg="((0 0)(10 1))" />
<node label="life not worth living" code="gen-life-not-livng" values="scale"
  value-mg="((0 0)(10 1))" />
</node>
</node>
- <node label="person's perspective of self worth" code="gen-self-worth-p"
  layer="0" generic-type="gd">
  <node label="grandiosity" code="grandiosity" values="scale" value-mg="((0
    0)(10 1))" />
  <node label="worthlessness" code="worthlessness" values="scale" value-
    mg="((0 0)(10 1))" />
  </node>
- <node label="mental health problems" code="mental-health" layer="0" filter-
  q="">
- <node label="depression" code="gen-depression" filter-q="" generic-type="g">
- <node label="Seriousness of current depression" code="serious-depression"
  values="nominal" value-mg="((none 0)(mild 0.3)(moderate 0.65)(severe
    10))">
- <node label="feelings/emotions" code="gen-feel-emot" layer="0" generic="gd">
  <node label="mood swings/lability" code="gen-mood-swings" values="scale"
    value-mg="((0 0)(10 1))" />
  <node label="negative feelings about the self" code="gen-negative-self"
    values="scale" value-mg="((0 0)(10 1))" />
  <node label="angry emotions" code="gen-angry-emotns" values="scale" value-
    mg="((0 0)(10 1))" />
  <node label="anxiety-based emotions" code="gen-anx-emotns" values="scale"
    value-mg="((0 0)(10 1))" />
  <node label="helplessness" code="gen-helpless" values="scale" value-mg="((0
    0)(10 1))" />
  <node label="sad/downbeat" code="gen-sad" values="scale" value-mg="((0
    0)(10 1))" />
  <node label="distress" code="gen-distress" values="scale" value-mg="((0 0)(10
    1))" />
  <node label="jealousy" code="gen-jealous" values="scale" value-mg="((0 0)(10
    1))" />
- <node label="hopelessness" code="gen-hopeless" values="scale" value-mg="((0
    0)(10 1))" header="Questions on hopelessness" generic-type="g">
  <node label="plans for the future" code="gen-plans-future" values="scale"
    value-mg="((0 0)(10 1))" />
  <node label="life not worth living" code="gen-life-not-livng" values="scale"
    value-mg="((0 0)(10 1))" />
  </node>
</node>
- <node label="person's perspective of self worth" code="gen-self-worth-p"
  layer="0" generic="gd">
  <node label="grandiosity" code="grandiosity" values="scale" value-mg="((0
    0)(10 1))" />
  <node label="worthlessness" code="worthlessness" values="scale" value-
    mg="((0 0)(10 1))" />
  </node>

```

```

<node label="general motivation in life" code="gen-motivation" generic-
  datum="g" values="scale" value-mg="((0 0)(10 1))" />
<node label="voice hallucinations" code="gen-voice-hal" generic="generic
  issues >> direct risk children >> mental health problems >> serious
  mental illness >> current symptoms of severe mental illness >> voice
  hallucinations" ERROR="PATH INVALID: generic issues >> direct risk
  children >> mental health problems >> serious mental illness >> current
  symptoms of severe mental illness >> voice hallucinations" />
<node label="paranoid delusions" code="gen-paranoid-del" generic="generic
  issues >> direct risk children >> mental health problems >> serious
  mental illness >> current symptoms of severe mental illness >> paranoid
  delusions" ERROR="PATH INVALID: generic issues >> direct risk children
  >> mental health problems >> serious mental illness >> current
  symptoms of severe mental illness >> paranoid delusions" />
<node label="impaired cognitive function" code="gen-impaired-cog"
  generic="generic issues >> direct risk children >> mental
  faculties/cognitive capacity >> impaired cognitive function" ERROR="PATH
  INVALID: generic issues >> direct risk children >> mental
  faculties/cognitive capacity >> impaired cognitive function" />
= <node label="general current behaviour" code="gen-currnt-bhvr" layer="0"
  generic="gd">
<node label="reckless risk taking" code="gen-rsk-behavr" values="scale" value-
  mg="((0 0)(10 1))" />
<node label="unintentional risk making" code="gen-unint-risk-behavr"
  values="scale" value-mg="((0 0)(10 1))" />
<node label="sleep disturbance" code="gen-sleep-dist" values="scale" value-
  mg="((0 0)(10 1))" />
= <node label="appropriateness of diet" code="gen-app-diet" values="scale"
  value-mg="((0 0)(10 1))" layer="0" generic="g">
<node label="eating" code="gen-diet-eating" values="scale" value-mg="((0
  0)(10 1))" />
= <node label="weight" code="gen-diet-weight">
<node label="person's weight" code="gen-diet-weigt-ext" values="nominal"
  value-mg="((EXTREME-UNDERWEIGHT 1)(UNDERWEIGHT 0.5)(WEIGHT-OK
  0)(OVERWEIGHT 0.5)(EXTREME-OVERWEIGHT 1))" />
<node label="extreme weight change" code="gen-diet-weigt-chg"
  values="scale" value-mg="((0 0)(10 1))" />
</node>
<node label="drinking" code="gen-diet-drink" values="scale" value-mg="((0
  0)(10 1))" />
</node>
<node label="uncharacteristic recent change in behaviour" code="gen-unusl-
  rec-bhvr" values="scale" value-mg="((0 0)(10 1))" />
<node label="challenging behaviour" code="gen-chall-bhvr" values="scale"
  value-mg="((0 0)(10 1))" />
= <node label="daily activity" code="gen-day-actvty" values="scale" value-
  mg="((0 0)(10 1))">
<node label="structure of day" code="gen-day-struct" values="scale" value-
  mg="((0 0)(10 1))" />
<node label="general level of activity during the day" code="gen-day-actvty-
  lev" values="nominal" value-mg="((PASSIVE-INERT 0)(UNDERACTIVE
  0)(NORMAL 0)(OVERACTIVE 0.5)(HYPERACTIVE 1))" />

```

```

</node>
</node>
= <node label="person's behavioural presentation during assessment"
  code="gen-presentation" layer="0" generic="gd">
= <node label="person's engagement with assessor" code="gen-engagement"
  filter-q="" generic-type="gd">
  <node label="rapport/empathy" code="gen-rapport" values="scale" value-
    mg="((0 0)(10 1))" />
  <node label="person's responsiveness" code="gen-responsve" values="scale"
    value-mg="((0 0)(10 1))" />
  <node label="assessor's uneasiness about the person" code="gen-gut-assmnt"
    values="scale" value-mg="((0 0)(10 1))" />
  </node>
= <node label="verbal indicators of risk" code="gen-risk-verbal" filter-q=""
  generic-type="gd">
= <node label="tone" code="gen-risk-tone">
  <node label="degree of aggression/hostility" code="gen-risk-aggrsv"
    values="scale" value-mg="((0 0)(10 1))" />
  <node label="how upbeat or downbeat/depressed" code="gen-risk-upbeat"
    values="scale" value-mg="((0 0)(10 1))" />
  </node>
  <node label="degree to which the person is making sense" code="gen-
    coherence" values="scale" value-mg="((0 0)(10 1))" />
  </node>
= <node label="body language and expression" code="gen-body-face" filter-q=""
  generic-type="gd">
  <node label="body language indicating distress" code="gen-distrss-b-lang"
    values="scale" value-mg="((0 0)(10 1))" />
  <node label="movements, posture, facial expression indicating low mood"
    code="gen-low-mood" values="scale" value-mg="((0 0)(10 1))" />
  <node label="aggressive/threatening movements, posture, or expression"
    code="gen-threat-move" values="scale" value-mg="((0 0)(10 1))" />
  <node label="preoccupied/detached demeanour" code="gen-detached"
    values="scale" value-mg="((0 0)(10 1))" />
= <node label="eyes" code="gen-eyes" generic-type="gd">
  <node label="avoid eye contact" code="gen-avoid-eye-contact" values="scale"
    value-mg="((0 0)(10 1))" />
  <node label="eye movement" code="gen-eye-movement" values="nominal"
    value-mg="((UNRESPONSIVE-GLAZED 1)(APPROPRIATE 0)(FIXED-STARING
    1)(DARTING 1))" />
  </node>
  </node>
  <node label="congruence of physical, verbal, and emotional presentation"
    code="gen-congruence" values="scale" value-mg="((0 0)(10 1))" />
  </node>
= <node label="engagement with world" code="gen-eng-world" values="scale"
  value-mg="((0 0)(10 1))" generic="gd">
  <node label="physical withdrawal from world" code="gen-phys-withd"
    values="scale" value-mg="((0 0)(10 1))" />
  <node label="mental withdrawal" code="gen-mental-withd" values="scale"
    value-mg="((0 0)(10 1))" />

```

```

</node>
</node>
<node label="stage of depression" code="gen-dep-stage" values="nominal"
value-mg="((FIRST-DIAGNOSIS 1)(RECOVERY-SINGLE-EPISODE
0)(RECOVERY-REPEAT-EPISODES 0.4)(RELAPSE 1))" />
</node>
- <node label="serious mental illness" code="gen-ser-mentl-ill" filter-q="" generic-
type="gd">
<node label="insight into mental-health problems" code="gen-mentl-insght"
values="scale" value-mg="((0 0)(10 1))" />
- <node label="current symptoms of severe mental illness" code="gen-mntl-cur-
sympt" filter-q="">
<node label="mania/hypomania" code="gen-mania" values="scale" value-
mg="((0 0)(10 1))" />
- <node label="voice hallucinations" code="gen-voice-hal" filter-q=""
header="Questions on voice hallucinations" generic-type="g">
- <node label="type of voices" code="gen-voices-type">
<node label="danger of voices to self" code="gen-voice-dang-s" values="scale"
value-mg="((0 0)(10 1))" />
<node label="danger of voices to others" code="gen-voice-dang-o"
values="scale" value-mg="((0 0)(10 1))" />
</node>
<node label="likelihood of acting on the voices" code="gen-prob-act-voice"
values="scale" value-mg="((0 0)(10 1))" />
</node>
- <node label="paranoid delusions" code="gen-paranoid-del" filter-q=""
header="Questions on paranoid delusions" generic-type="g">
- <node label="type of paranoid delusions" code="gen-type-paranoid-del">
<node label="about specific individuals" code="gen-paran-del-spec"
values="scale" value-mg="((0 0)(10 1))" />
<node label="being harmed, killed, or persecuted" code="gen-paran-del-pers"
values="scale" value-mg="((0 0)(10 1))" />
</node>
<node label="likelihood of acting on delusions" code="gen-prob-act-par-del"
values="scale" value-mg="((0 0)(10 1))" />
</node>
</node>
</node>
</node>
- <node label="mental faculties/cognitive capacity" code="ment-fac" layer="0">
- <node label="impaired cognitive function" code="gen-impaired-cog" filter-q=""
generic-type="g">
<node label="thinking processes and memory" code="gen-cog-think-mem"
values="scale" value-mg="((0 0)(10 1))" />
<node label="concentration" code="gen-concentr" values="scale" value-mg="((0
0)(10 1))" />
</node>
<node label="learning disabilities" code="gen-learn-disab" values="scale" value-
mg="((0 0)(10 1))" />
</node>
- <node label="personality" code="gen-personality" layer="0" generic-type="gd">

```

```

<node label="assertiveness" code="gen-assertive" values="nominal" value-
mg="((NOT-ASSERTIVE 1)(SOMEWHAT-ASSERTIVE 0.3)(NORMALLY-
ASSERTIVE 0)(VERY-ASSERTIVE 0)(EXCESSIVELY-ASSERTIVE 0))" />
<node label="ability to empathise" code="gen-empathy-abil" values="scale"
value-mg="((0 0)(10 1))" />
<node label="dependence" code="gen-dependence" values="scale" value-
mg="((0 0)(10 1))" />
<node label="controlling/organisational approach" code="gen-controlling"
values="nominal" value-mg="((chaotic 0.5)(disorganised 0.2)(normal
0)(very-organised 0.5)(obsessional-perfectionist 1))" />
<node label="capacity to cope with major life stresses" code="gen-coping-abil"
values="scale" value-mg="((0 0)(10 1))" />
<node label="hostility" code="gen-hostile" values="scale" value-mg="((0 0)(10
1))" />
<node label="impulsiveness" code="gen-impulse" values="scale" value-mg="((0
0)(10 1))" />
<node label="reliability" code="gen-reliable" values="scale" value-mg="((0
0)(10 1))" />
</node>
= <node label="motivation and engagement with world" code="motive-eng"
layer="0">
= <node label="engagement with world" code="gen-eng-world" values="scale"
value-mg="((0 0)(10 1))" generic-type="gd">
<node label="physical withdrawal from world" code="gen-phys-withd"
values="scale" value-mg="((0 0)(10 1))" />
<node label="mental withdrawal" code="gen-mental-withd" values="scale"
value-mg="((0 0)(10 1))" />
</node>
<node label="general motivation in life" code="gen-motivation" values="scale"
value-mg="((0 0)(10 1))" />
<node label="listless, no energy, slowed down, loss of drives" code="gen-
listless" values="scale" value-mg="((0 0)(10 1))" />
</node>
= <node label="social context" code="gen-soc-contxt" layer="0" generic-
type="gd">
= <node label="relationships" code="gen-relatnshps" header="Questions on
current relationships" generic-type="gd">
<node label="external network of relationships" code="gen-net-relat"
values="scale" value-mg="((0 0)(10 1))" />
= <node label="nature of relationships" code="gen-relat-nature">
<node label="supportive relationships" code="gen-relat-supp" values="scale"
value-mg="((0 0)(10 1))" />
<node label="detrimental relationships" code="gen-relat-detr" values="scale"
value-mg="((0 0)(10 1))" />
</node>
<node label="detrimental changes to relationships" code="gen-relat-detr-chg"
values="scale" value-mg="((0 0)(10 1))" />
</node>
= <node label="living arrangements" code="gen-living-arr" header="Questions on
living arrangements" generic-type="gd">

```

```

<node label="frequency of moving accommodation" code="gen-move-freq"
  values="nominal" value-mg="((LESS-THAN-EVERY-YEAR 0)(ONCE-EVERY-
  YEAR 0.5)(SEVERAL-TIMES-YEAR 0.8)(ONCE-A-MONTH-OR-MORE 1))" />
<node label="type of home" code="gen-home-type" values="nominal" value-
  mg="((HOMELESS 1)(HOSTEL 0.8)(institution-fully-supervised 0)(daily-
  support 0.1)(limited-support 0.2)(no-support 0.5))" />
= <node label="neighbourhood" code="gen-neighbrhd">
  <node label="isolated accommodation" code="gen-isol-accom" values="scale"
    value-mg="((0 0)(10 1))" />
  <node label="risky neighbourhood" code="gen-neighbrhd-rsky" values="scale"
    value-mg="((0 0)(10 1))" />
  </node>
= <node label="state of accommodation" code="gen-accom-state">
  <node label="care of home" code="gen-accom-hm-care" values="scale" value-
    mg="((0 0)(10 1))" />
  <node label="habitable accommodation" code="gen-accom-habitbl"
    values="scale" value-mg="((0 0)(10 1))" />
  </node>
  </node>
= <node label="financial problems" code="gen-finance-prob" filter-q=""
  header="Questions on financial problems">
  <node label="anxiety about perceived level of debts" code="gen-perc-debt-
    anx" values="scale" value-mg="((0 0)(10 1))" />
  <node label="chronic poverty" code="gen-poverty" values="scale" value-
    mg="((0 0)(10 1))" />
  </node>
= <node label="employment" code="gen-employment" filter-q=""
  header="Questions on employment">
  <node label="frequency of changing jobs" code="gen-job-chg-frq"
    values="scale" value-mg="((0 0)(10 1))" />
  <node label="recent or potential detrimental change to employment"
    code="gen-rec-bad-job-ch" values="scale" value-mg="((0 0)(10 1))" />
  </node>
  </node>
= <node label="general current behaviour" code="gen-currnt-bhvr" layer="0"
  generic-type="gd">
  <node label="reckless risk taking" code="gen-rsk-behavr" values="scale" value-
    mg="((0 0)(10 1))" />
  <node label="unintentional risk making" code="gen-unint-risk-behavr"
    values="scale" value-mg="((0 0)(10 1))" />
  <node label="sleep disturbance" code="gen-sleep-dist" values="scale" value-
    mg="((0 0)(10 1))" />
= <node label="appropriateness of diet" code="gen-app-diet" values="scale"
  value-mg="((0 0)(10 1))" layer="0" generic="g">
  <node label="eating" code="gen-diet-eating" values="scale" value-mg="((0
    0)(10 1))" />
= <node label="weight" code="gen-diet-weight">
  <node label="person's weight" code="gen-diet-weigt-ext" values="nominal"
    value-mg="((EXTREME-UNDERWEIGHT 1)(UNDERWEIGHT 0.5)(WEIGHT-OK
    0)(OVERWEIGHT 0.5)(EXTREME-OVERWEIGHT 1))" />

```

```

<node label="extreme weight change" code="gen-diet-weigt-chg"
  values="scale" value-mg="((0 0)(10 1))" />
</node>
<node label="drinking" code="gen-diet-drink" values="scale" value-mg="((0
  0)(10 1))" />
</node>
<node label="uncharacteristic recent change in behaviour" code="gen-unusl-
  rec-bhvr" values="scale" value-mg="((0 0)(10 1))" />
<node label="challenging behaviour" code="gen-chall-bhvr" values="scale"
  value-mg="((0 0)(10 1))" />
= <node label="daily activity" code="gen-day-actvty" values="scale" value-
  mg="((0 0)(10 1))">
  <node label="structure of day" code="gen-day-struct" values="scale" value-
    mg="((0 0)(10 1))" />
  <node label="general level of activity during the day" code="gen-day-actvty-
    lev" values="nominal" value-mg="((PASSIVE-INERT 0)(UNDERACTIVE
    0)(NORMAL 0)(OVERACTIVE 0.5)(HYPERACTIVE 1))" />
  </node>
  </node>
= <node label="substance misuse" code="gen-subs-misuse" layer="0" filter-q=""
  generic-type="gd">
  <node label="detrimental effects of alcohol misuse" code="gen-alc-misuse"
    values="scale" value-mg="((0 0)(10 1))" />
  <node label="detrimental effects of drugs misuse" code="gen-drug-misuse"
    values="scale" value-mg="((0 0)(10 1))" />
  </node>
= <node label="insight and responsibility" code="insight-resp" values="scale"
  value-mg="((0 0)(10 1))" layer="0" generic-type="g">
  <node label="insight into behaviour and consequences" code="gen-insght-
    bhvr" values="scale" value-mg="((0 0)(10 1))" />
  <node label="responsibility for impact of behaviour on others" code="gen-
    resp-impct-oth" values="scale" value-mg="((0 0)(10 1))" />
  <node label="need for help with difficulties" code="gen-nd-hlp-diff"
    values="scale" value-mg="((0 0)(10 1))" />
  </node>
= <node label="physical health problems" code="gen-phys-hlth-prb"
  values="scale" layer="0" generic-type="gd">
  <node label="when life-threatening or degenerative illness first diagnosed"
    code="gen-phys-hlth-deg-diag" values="date-month" value-mg="((3 1)(36
    0))" />
  <node label="pain" code="gen-phys-hlth-pain" values="scale" value-mg="((0
    0)(10 1))" />
  <node label="disability" code="gen-phys-hlth-disa" values="scale" value-
    mg="((0 0)(10 1))" />
  <node label="communication impairment" code="gen-com-imp" values="scale"
    value-mg="((0 0)(10 1))" />
  <node label="deterioration in physical health" code="gen-phys-hlth-det"
    values="scale" value-mg="((0 0)(10 1))" />
  </node>
= <node label="concordance with health services/medication/therapies"
  code="gen-meds-therpy" layer="0" generic-type="g">

```

```

<node label="concordance" code="gen-meds-concord" values="scale" value-
mg="((0 0)(10 1))" />
<node label="person's perception of the supportiveness of service received"
code="gen-serv-perc-supp" values="scale" value-mg="((0 0)(10 1))" />
<node label="time since person accessed services" code="gen-serv-last-acc"
values="scale" value-mg="((0 0)(10 1))" />
<node label="perceived benefit of medication/therapies" code="gen-med-perc-
benft" values="scale" value-mg="((0 0)(10 1))" />
</node>
= <node label="adverse life events" code="adv-life-event" layer="0" filter-q=""
generic-type="gd">
= <node label="traumatic experiences" code="gen-life-trauma" generic-
type="gd">
= <node label="recent traumatic life changes" code="gen-recent-life-trauma"
generic-type="g">
<node label="detrimental changes to relationships" code="gen-relat-detr-chg"
generic-datum="g" values="scale" value-mg="((0 0)(10 1))" />
<node label="recent or potential detrimental change to employment"
code="gen-rec-bad-job-ch" generic-datum="g" values="scale" value-mg="((0
0)(10 1))" />
<node label="when life-threatening or degenerative illness first diagnosed"
code="gen-phys-hlth-deg-diag" generic-datum="g" values="date-month"
value-mg="((3 1)(36 0))" />
</node>
= <node label="abuse to person" code="gen-life-abuse" filter-q="" generic-
type="g">
= <node label="sexual abuse" code="gen-life-sex-abuse" values="scale" value-
mg="((0 0)(10 1))" filter-q="">
<node label="most recent episode of sexual abuse" code="gen-sex-abse-last"
values="date-year" value-mg="((0 1)(5 0.3))" ERROR="This node is
persistent; however the following ancestor nodes are not: gen-life-sex-
abuse" />
<node label="sexual abuse during childhood (0 to 16)" code="gen-sex-abse-
as-ch" values="nominal" value-mg="((YES 1)(NO 0))" ERROR="This node is
persistent; however the following ancestor nodes are not: gen-life-sex-
abuse" />
</node>
= <node label="physical abuse" code="gen-phys-abse" values="scale" value-
mg="((0 0)(10 1))" filter-q="">
<node label="most recent episode of physical abuse" code="gen-phys-abse-
last" values="date-month" value-mg="((0 1)(24 0.2))" ERROR="This node is
persistent; however the following ancestor nodes are not: gen-phys-abse"
/>
<node label="physical abuse during childhood (0 to 16)" code="gen-phy-abse-
as-ch" values="nominal" value-mg="((YES 1)(NO 0))" ERROR="This node is
persistent; however the following ancestor nodes are not: gen-phys-abse"
/>
</node>
= <node label="emotional abuse" code="gen-emot-abse" values="scale" value-
mg="((0 0)(10 1))" filter-q="">

```

```

<node label="most recent episode of emotional abuse" code="gen-emot-abse-
last" values="date-month" value-mg="((0 1)(24 0.2))" ERROR="This node is
persistent; however the following ancestor nodes are not: gen-emot-abse"
/>
<node label="emotional abuse during childhood (0 to 16)" code="gen-emo-
abse-as-ch" values="nominal" value-mg="((YES 1)(NO 0))" ERROR="This
node is persistent; however the following ancestor nodes are not: gen-
emot-abse" />
</node>
<node label="financial abuse" code="gen-financial-abuse" values="nominal"
value-mg="((YES 1)(NO 0))" />
</node>
<node label="forensic/criminal proceedings" code="gen-forensic-proc"
values="nominal" value-mg="((YES 1)(NO 0))" />
</node>
<node label="environment person grew up in" code="gen-env-grew-up"
values="scale" value-mg="((0 0)(10 1))" />
<node label="eating disorders" code="gen-eating-dis" values="scale" value-
mg="((0 0)(10 1))" />
<node label="educational experience" code="gen-educ-expr" values="scale"
value-mg="((0 0)(10 1))" />
</node>
= <node label="demographics" code="gen-demog" generic-type="gd">
<node label="age" code="gen-age" values="date-year" value-mg="((14 0)(20
1)(30 0.5)(50 0)(60 0.3)(80 1))" />
<node label="gender" code="gen-gender" values="nominal" value-mg="((MALE
1)(FEMALE 0))" />
<node label="relationship status" code="gen-marital-status" values="nominal"
value-mg="((SINGLE 0.8)(WITH-PARTNER 0)(SEPARATED-FROM-PARTNER
1)(PARTNER-DIED 1))" />
= <node label="occupants sharing accommodation" code="gen-accom-share"
filter-q="">
= <node label="Dependents" code="gen-accom-depdnts" filter-q="">
<node label="number of dependents" code="gen-accom-num-dep"
values="integer" value-mg="((0 0)(8 1))" />
<node label="age of youngest dependent" code="gen-dep-ygnst-age"
values="integer" value-mg="((0 1)(10 0.8)(18 0))" />
</node>
<node label="partner sharing" code="partner-share-acc" values="nominal"
value-mg="((no 1)(yes 0))" />
<node label="number of non-dependents sharing accommodation" code="gen-
accm-share-nd" values="integer" value-mg="((0 0)(5 1))" />
</node>
<node label="ethnicity" code="gen-ethnicity" values="nominal" value-
mg="((WHITE-BRITISH 0)(WHITE-IRISH 0)(OTHER-WHITE 0)(WHITE-
BLACK-CARIBBEAN 0)(WHITE-BLACK-AFRICAN 0) (WHITE-ASIAN
0)(OTHER-MIXED 0)(INDIAN 0)(PAKISTANI 0)(BANGLADESHI 0)(OTHER-
ASIAN 0)(BLACK-CARIBBEAN 0) (BLACK-AFRICAN 0)(OTHER-BLACK
0)(CHINESE 0)(OTHER-ETHNIC 0))" />
</node>
</node>

```