ECONOMIC MODELLING USING COMPUTATIONAL INTELLIGENCE TECHNIQUES

Msizi Smiso Khoza

FACTOR OF ENGINEERING & THE BUILT ENVIRONMENT

UNIVERSITY OF JOHANNESBURG

MARCH 2012

DISSERTATION SUBMITTED IN FULFILMENT OF THE REQUIREMENTS OF

Magister Ingeneriae
Declaration

I hereby declare that this work is my own unaided work unless otherwise stated in the form of a numbered reference. It is being submitted for the degree: Magister Ingeneriae at the University of Johannesburg (South Africa). This dissertation has never been submitted for any degree or examination at any other university.
Abstract

Economic modelling tools have gained popularity in recent years due to the increasing need for greater knowledge to assist policy makers and economists. A number of computational intelligence approaches have been proposed for economic modelling. Most of these approaches focus on the accuracy of prediction and not much research has been allocated to investigate the interpretability of the decisions derived from these systems.

This work proposes the use of computational intelligence techniques (Rough set theory (RST) and the Multi-layer perceptron (MLP) model) to model the South African economy. RST is a rule-based technique suitable for analysing vague, uncertain and imprecise data. RST extracts rules from the data to model the system. These rules are used for prediction and interpreting the decision process.

The lesser the number of rules, the easier it is to interpret the model. The performance of the RST is dependent on the discretization technique employed. An equal frequency bin (EFB), Boolean reasoning (BR), entropy partition (EP) and the Naive algorithm (NA) are used to develop an RST model. The model trained using EFB data performs better than the models trained using BR and EP.

RST was used to model South Africa’s financial sector. Here, accuracy of 86.8%, 57.7%, 64.5% and 43% were achieved for EFB, BR, EP and NA respectively.

This work also proposes an ensemble of rough set theory and the multi-layer perceptron model to model the South African economy wherein, a prediction of the direction of the gross domestic product is presented.
This work also proposes the use of an auto-associative Neural Network to impute missing economic data. The auto-associative neural network imputed the ten variables or attributes that were used in the prediction model.

These variables were: Construction contractors rating lack of skilled labour as constraint, Tertiary economic sector contribution to GDP, Income velocity of circulation of money, Total manufacturing production volume, Manufacturing firms rating lack of skilled labour as constraint, Total asset value of banking industry, Nominal unit labour cost, Total mass of Platinum Group Metals (PGMs) mined, Total revenue from sale of PGMs and the Gross Domestic Expenditure (GDE).

The level of imputation accuracy achieved varied with the attribute. The accuracy ranged from 85.9% to 98.7%.
To my mothers,

For all your prayers, support and guidance

Throughout my academic career.
Acknowledgments

I wish to thank my academic supervisor and mentor, Professor Tshilidzi Marwala, for his support, encouragement and insightful guidance, without which I would not have been able to complete this work. His passion and enthusiasm has proved invaluable in the completion of this work.

Many thanks also go out to my family for their continued emotional support during the period at which this research was conducted.

It would be mischievous of me not to mention Mr Lindokuhle Mpanza and Dr Pretesh Patel who have proved reliable counsel and advisors.
Chapter 1

Introduction

1.1 Background & Motivation

Mastering of the practice of economic modeling has, for a long time, attracted the attention and interests of economists, government bureaucrats, political theoreticians and scientists alike. In today’s global socio-political environment, economics has become an important and central feature of the determinants that shape the policies, outlook and character of modern nation states.

Today an economic system extends far beyond its traditional formulation and determines relations between nations, international alliances and politics. All of these underpin the importance of the development of methods and tools that can be used for the accurate modeling of an economic system. As the Scottish philosopher and “father of modern economics”, Adam Smith, proffered in his work, “An inquiry into the nature and causes of the wealth of nations”, there are many factors that influence the economy of any country.

It is reasonable to assume that one can define these factors as belonging to one of two broad categories: Hard and Soft. Those factors belonging to the former category are those which can be subjected to analytical methods of analysis whilst those that belong to the latter are ones which are related to calculating “intrinsic value” of an economy and are much harder to define as functions of generalized formulae [1].

Hard factors would be such factors as the inflation index, the percentage contribution to the gross domestic product of various sectors of the economy, the output and value of certain sectors of an economy, the gross national product, the current account deficit etc.

Soft factors would be such factors as the perceived level of geo-political stability, the character and nature of macro-economic policy, relations with principal international trading partners, social cohesion, commitment to the rule of law, levels of democratic engagement, political tolerance and freedom of speech etc.
Soft factors are obviously much harder to predict and subject to statistical analysis. Accordingly we can draw parallels and similarities between these factors and the concept of Fundamental Analysis in stock market price prediction much like we can draw similarities between Hard factors and Technical Analysis in stock market price prediction.

It is generally accepted that the diversity of factors (both hard and soft) that influence the direction of any country’s economy makes it difficult to find a single solution that encompasses the entire solution space. Accordingly, this makes the development of an accurate model of any nation’s economy an interesting research and engineering problem.

1.2 Literature Review

Around the world economics has defined the character, nature and outlook of individuals, governments and private companies. Thus the practice of economic modelling has long attracted the interests of government bureaucrats, political theoreticians and scientists alike. For a long time statistical techniques such as Bayesian models, regression and some econometric techniques have dominated research activities in prediction [1].

One of the biggest problems with the use of regression methods is that they fail to give satisfactory forecasting results for some series’ because of their linear structure and other inherent limitations [13,15,16].

The emergence of computational intelligence techniques as a viable alternative to the “traditional” statistical models that have dominated this area since the 1930’s [2,3] has given impetus to the increasing usage of these techniques in fields such as economics and finance [3,4,5].

This led to the extensive use of artificial intelligence (AI) techniques particularly neural networks because of their natural ability to learn and adapt to complex non-linear mappings of various statistical distributions [5,11,18,19].

Since the year of its introduction by Pawlak in 1982 [10], rough set theory has been extensively used as an effective data mining and knowledge discovery technique in numerous applications in the finance, investment and banking fields. Data mining is a discipline in computational intelligence that deals with knowledge discovery, data analysis, and full and semi-autonomous decision making [13]. It entails the analysis of data sets such that unsuspected relationships among data objects are found.

Whilst RST has been used successfully in the past in prediction and modelling applications, its combination with the canonical form of the artificial neural network to give effect to rough neural computing or neuro-rough model has some advantages over the traditional RST model when applied exclusively [26,28,29,33-36].
1.3 Research Focus, Approach & Methodology

This work attempts to provide a generic model of the South African economy using a rough neural network. However, acknowledging the immensely important role that the financial markets play in the success of any economic system, I felt it incumbent upon me to pay particular attention to that aspect of the economy - the financial markets induced global economic recession of 2008/2009 is but a case in point.

What is meant by “particular attention” is the development of a generic RST based model of the financial markets (Chapter 2).

Next a generic rough neural network of the South African economy is developed using the back bone of the RST model developed for the financial markets model (Chapter 2) in combination with the canonical form of the artificial neural network (ANN). Furthermore, a missing data imputation algorithm is implemented for missing economic data using an ANN and genetic algorithm (GA) (Chapter 4).

Because of the specific history of the evolution of the South African economy and society, the fundamental hypothesis is that the model takes no regard for the effect that South Africa’s political transition might have had on the economic system. That is to say, the model is developed ceteris paribus from the date at which the data set starts to the date at which it ends.

The general approach used is as follows:

- The first step entailed the collection and definition of the data,
- Based on basic economics and finance theory, a decision was taken on the definition of independent variables,
- The data is then normalised to the correct format, that is to say that there were some variables whose data was organised monthly and others whose data was organised to reflect changes per every quarter of a year (3 calendar months),
- Next the data was discretized through the employ of multiple data discretization algorithms. The algorithms are benchmarked against each other using the accuracy metric to decide which is the best discretization algorithm to be used,
- A RST model is built,
- An ANN model is also developed using the particle swarm optimization algorithm for finding the optimal number of hidden layers and nodes,
- Lastly, an ensemble of the RST model and ANN is created to create a holistic and robust model of the South African economic system.
1.4 Outline of Dissertation

The remaining parts of this dissertation are organized in the following manner:

Chapter 2: This chapter encompasses the design of the RST model developed for purposes of modelling the financial market. The data used is from the Johannesburg Stock Exchange (JSE) and stretches a five year period beginning April 2006 and ending April 2011.

Chapter 3: The chapter presents the design of the ANN as well as the RST using the backbone of the model developed in Chapter 2. In this chapter the ensemble of both models is presented to provide a holistic model of the South African economic system.

Chapter 4: This chapter presents an ANN missing economic data imputation model that uses a GA as an optimisation algorithm.

Chapter 5: The fifth chapter of this dissertation serves as its last and concludes the work by giving an analysis of the findings and gives suggestions for future work.

Although the contents of Chapter 4 are, strictly speaking independent of the main thrust of this work, the reader is encouraged to read it notwithstanding this fact as it provides a good appendage to this work.

1.5 Published Works

In the course of this research our findings have been documented in the form of academic papers. The following papers have been published.


• M Khoza and T Marwala, “Computational Intelligence Techniques for Modelling an Economic System”, 2012 IEEE International Joint Conference on Neural Networks, World Congress on Computational Intelligence, Brisbane, Australia, 10-15 June 2012
Chapter 2

A Rough Set Theory Based Model of the Financial Markets

This chapter encompasses the design of a RST model developed for purposes of modelling the financial market. The data used is from the Johannesburg Stock Exchange (JSE) and stretches a five year period beginning April 2006 and ending April 2011.

2.1 Theoretical Foundations of Rough Set Theory

Rough set theory (RST) was introduced by Pawlak in 1982. The theory can be regarded as a mathematical tool used for imperfect data analysis [10]. Thus RST has proved useful in applications spanning the engineering, financial and decision support domains to mention but a few. It is based on the assumption that “with every object in the universe of discourse, some information (data or knowledge) is associated” [10].

In practical applications, the “universe of discourse” described in [10] is usually a table called the decision table in which the rows are objects or data elements and the columns are attributes and the entries are called the attribute values [3].

The objects or data elements described by the same attributes are said to be indiscernible (indistinguishable) by the attribute set. Any set of indiscernible data elements forms a granule or atom of knowledge about the entire “universe of discourse” (information system framework) [10].
A union of these elementary sets (granules) is said to be a precise or crisp set, otherwise the set is said to be rough [1,2,7,10]. Every rough set will have boundary cases i.e. data objects which cannot certainly be classified as belonging to the set or its complement when using the available information [10]. Associated with every rough set is a pair of sets called the lower and upper approximation of the rough set.

The lower approximation consists of those objects which one can definitively say belong to the target set. The upper approximation consists of those objects which possibly belong to the target set. The difference between the two sets is the boundary region.

The decision rule derived specifies an outcome based on certain conditions. Where the derived rule uniquely identifies outcomes based on some conditions the rule is said to be certain else it is uncertain. Every decision rule has a pair of probabilities associated with it, the certainty and coverage coefficients [3].

These conditional probabilities also satisfy Bayes’ theorem [7,10]. The certainty coefficient is the conditional probability that an object that belongs to the decision class outlined by the rule given that it satisfies the conditions of the rule.

The coverage coefficient on other hand expresses the conditional probability of reasons given some decision [10]. Clearly RST can be seen to overlap with many other theories in the realm of imperfect knowledge analysis such as evidence theory, Bayesian inference, fuzzy sets etc [1, 3, 4, 10, 11, 12].

To define rough sets mathematically, we begin by defining an information system \( S = (U,A) \), where \( U \) and \( A \) are finite and non-empty sets that represent the data objects and attributes respectively. Every attribute \( a \in A \) has a set of possible values \( V_a \). \( V_a \) is called the domain of \( a \). A subset of \( A \) say \( B \) will determine a binary relation \( I(B) \) on \( U \), which is called the indiscernibility relation.

The relation is defined as follows: \((x,y) \in I(B)\) if and only if \( a(x) = a(y) \) for every \( a \) in \( B \), where \( a(x) \) denotes the value of attribute \( a \) for data object \( x \) [10]. \( I(B) \) is an equivalence relation. All equivalence classes of \( I(B) \) as \( U/I(B) \). An equivalence class of \( I(B) \) containing \( x \) is denoted as \( B(x) \). If \((x,y) \) belong to \( I(B) \) they are said to be indiscernible with respect to \( B \). All equivalence classes of the indiscernibility relation, \( I(B) \), are referred to as \( B \)-granules or \( B \)-elementary sets [10].

In the information system defined above, we define as in [10]:

\[
X \subseteq U
\]  

(2.1)

And,
We now define the two operators assigned to every (1) two sets called the upper and lower approximation of \( X \). The two sets are defined as follows [10]:

\[
B_l(X) = \bigcup_{x \in X} \{ B(x) : B(x) \subseteq X \} \tag{2.3}
\]

And,

\[
B_u(X) = \bigcup_{x \in X} \{ B(x) : B(x) \cap X \neq \emptyset \} \tag{2.4}
\]

Thus, the lower approximation is the union of all \( B \)-elementary sets that are included in the target set, whilst the upper approximation is the union of all \( B \)-elementary sets that have a non-empty intersection with the target set. The difference between the two sets is called the boundary of region (BR) of \( X \) and can be written as follows [10].

\[
BR = B_u(X) - B_l(X) \tag{2.5}
\]

If the boundary region is an empty set then \( X \) is crisp with respect to \( B \), if however the boundary region is non-empty then \( X \) is rough with respect to \( B \). Accordingly, the set is said to be rough if it cannot be defined exactly from the available data. The set of attributes that is sufficient to represent the entire equivalence class structure is called the reduct.

The reduct of the information system is not unique. There are potentially many subsets of attributes which preserve the equivalence class structure. The set of attributes common to all reducts is called the core. The core can be regarded as the indispensable attribute of the information system. However, in practical applications where the information system contains thousands or possible tens of thousands of objects it is seldom that a core exists as shown in [1].
2.2 Design of the Rough Set Predictive Model

The design of the predictive model is based on rough set theory. The model comprises a set trading rules extracted from a set of *reducts* generated from discretized training data. The measure of the accuracy of the model is based on the percentage correctness of its predictions. The model comprises 6 stages, as can be seen in Figure 2.1 below.

![Diagram](https://via.placeholder.com/150)

Figure 2.1. The Outline of the Rough Set Predictive Model

2.2.1 Data Pre-processing

This phase of the design process includes activities such as additional attributes computation, filtering and data completion. There is a difference between time series with and without real time constraints. In the case of the latter, the only primary factor is the chronological order of events. As with stock price data and the model we sought to build the interval between events is a major contributing factor. To construct rough set objects from real time data we adopted a hybrid method that amalgamates the “columnizing method” the “mobile moving window” model proposed in [13].

The decision table is constructed by having columns as various technical indicators and the rows representing trading data at each point in time, whilst the window presents a “snapshot” of the state of the market in that period.
The data was randomly split in 75/25 ratio. The former part of the data is used for training and the remaining used for validation. There were a total of 10 attributes used. These are shown Table 2.1.

### Table 2.1 Table of Attributes

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>The opening price</td>
</tr>
<tr>
<td>High</td>
<td>The highest registered price on the day</td>
</tr>
<tr>
<td>Low</td>
<td>The lowest registered price on the day</td>
</tr>
<tr>
<td>Close</td>
<td>The closing price</td>
</tr>
<tr>
<td>Adjusted Close</td>
<td>The adjusted closing price</td>
</tr>
<tr>
<td>Moving Average</td>
<td>Moving average over 5 days</td>
</tr>
<tr>
<td>Momentum</td>
<td>( \frac{P_t - P_{t-4}}{P_t} \times 100 )</td>
</tr>
<tr>
<td>Rate of Change (ROC)</td>
<td>( \frac{P_i}{Moving\ Average} \times 100 )</td>
</tr>
<tr>
<td>Disparity</td>
<td>( \frac{P_i}{Moving\ Average} \times 100 )</td>
</tr>
<tr>
<td>Decision (D)</td>
<td>Decision Attribute</td>
</tr>
</tbody>
</table>

The Decision attribute, \( D \), is the signal indicating whether we buy or sell. The decision attribute indicates the future direction of the index. The attribute was calculated using the following [2]:

\[
D = \frac{\sum_{i=1}^{n}(n+1-i) \times |Close(i) - Close(0)|}{\sum_{i=1}^{n}i} \tag{2.6}
\]
The value resulting from (6) above is normalized to restrict it to a value between -1 and +1. A value of +1 indicates that every day for the next \( n \) days into the future the price closes higher than today and similarly a value of -1 indicates that every day for the next \( n \) days into the future the price closes lower than today.

### 2.2.2 Data Discretization

In classification problems, the collective attributes of the objects may have varying degrees of importance. To quantitatively map this variance and thus distinguish those attributes with higher and lower importance, weights are accorded to the attributes.

This of course requires some auxiliary knowledge of the problem domain [2, 3, 12, 14, 15]. Since rough set theory makes no such assumption and only does computations on data included in the information table itself, it is therefore prudent to discretize continuous value attributes. It was shown in [16] that the quality of the learning algorithm and ultimately the accuracy of the prediction model is heavily dependent on the type of discretization algorithm used.

Four discretization algorithms were tested at this stage of the model. The four methods are: EFB, Boolean Reasoning, Entropy and the Naïve Algorithm. The method selected in this regard is the equal frequency binning (EFB) because it showed the highest accuracy. The values of the continuous value attributes are cut into a number of segments or intervals and values lying within the same interval are mapped onto a single value.

This has the advantage that the rules generated are not too specific and therefore the number of rules generated is decreased. Whilst a decrease in the number of rules is desirable, there is a trade-off between the number of rules generated and the accuracy of the model.

### 2.2.3 Redundant Attribute Elimination

The information table contains attributes that offer little or no new information about the objects. These attributes are said to be redundant [17]. These attributes can be eliminated from the information table without adversely affecting the degree of dependency between the remaining attributes and the decision [3, 5, 6, 10, 11, 17].
2.2.4 Reduct Generation

The minimal subset of attributes which can effectively maintain the dependency relation is called the reduct [4, 5, 7, 10, 13-15, 17]. A reduct is not unique, that is a table may have more than one reduct. However, finding all reducts is an NP-complete problem [2]. Chen and Wang [7] found that it is not necessary to find all reducts of the information system.

The reducts are found using the genetic algorithm (GA). Genetic algorithms were first introduced in the 1960’s by Holland [19]. The basic genetic algorithm is thus: a population of possible solutions is maintained; the better two solutions are selected for recombination; use their offspring (combination) to replace poorer solutions [18, 19]. In general, GA’s have been applied to large spectra of problems [19].

In this project the reduct generation was done through the rough set analysis tool, ROSETTA. In reduct generation, the end goal is that given a set or input patterns (or attributes) in \( k \)-dimensional space to find a transformed pattern in the \( n \)-dimensional space where \( n < k \) such that the optimization is met. The transformed patterns are evaluated using dimensionality as well as the class separation or classification accuracy.

In ROSETTA, the GA maintains a population of competing feature matrices. To assess the accuracy of each matrix, the input patterns are multiplied by the matrix. A set of transformed patterns is produced. These are sent to the classifier which measures the hit ratio or accuracy of the transformation matrix and in representing the data set. The accuracy is fed back into the GA. Thus the GA is modelled as a parallel iterative optimizer.

2.2.5 Rule Generation and Prediction

From the training set a set of corresponding rules are generated. Naturally there exists an engineering trade-off between the number of rules and the prediction accuracy. Whilst we envisage minimising the number of rules as this has implications for processing time responses to changing input sizes. We will return to this later.

The prediction is done on the validation data set. The validation data set consists of 312 objects. The prediction method is described pictorially in Figure 2.
To gain a better understanding of the trade-off between the number of rules generated and the inherent implication on resource requirements and the accuracy, the method used to assess this relationship is the *Big O Notation* which is a member of a larger family of notations called the *Bachmann-Landau Notations* [17].

The chapter introduced the theoretical foundations of Rough Set Theory (RST) and presented a RST based model of the South African financial market by predicting the direction in which the JSE’s All Share Index will turn. The model showed very high accuracy rates of 80.4%.
2.3 Results and Discussion

In the modelling of the Johannesburg Stock Exchange’s All Share Index (ALSI) over the past 5 years (2006 – 2011), there was a total of 182 reducts and 1004 rules were generated. Table 2.2 presents the dominance of each attribute, measured as an appearance percentage of the total number of reducts generated.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Count</th>
<th>Percentage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>54</td>
<td>29.35</td>
</tr>
<tr>
<td>High</td>
<td>73</td>
<td>39.67</td>
</tr>
<tr>
<td>Low</td>
<td>81</td>
<td>44</td>
</tr>
<tr>
<td>Close</td>
<td>71</td>
<td>38.58</td>
</tr>
<tr>
<td>Adj Close</td>
<td>54</td>
<td>29.34</td>
</tr>
<tr>
<td>MAV</td>
<td>68</td>
<td>36.95</td>
</tr>
<tr>
<td>Momentum</td>
<td>72</td>
<td>39.14</td>
</tr>
<tr>
<td>ROC</td>
<td>69</td>
<td>37.5</td>
</tr>
<tr>
<td>Disparity</td>
<td>89</td>
<td>48.37</td>
</tr>
</tbody>
</table>

The 6 most important attributes then are:

{High, Low, Close, Momentum, ROC, Disparity}

In order to be able to provide comparison, a set of rules was generated using the 182 reducts generated in the first phase of the previous stage and then a second set of rules was generated using only the core reduct above. With 182 reducts a total of 1004 rules were generated. With the core reduct above, a total of 246 rules were constructed.
There are a number of parameters that influence the accuracy of the model [16]. Chief among these are the data split ratio, discretization algorithm and the classifier method. To construct the most robust model the correct combination of these parameters was sought.

Since it was found in [16] that the quality of the model and hence its accuracy depends heavily on the discretization algorithm, a number of discretization algorithms were used so as to assess which gave the best results. These were: EFB, the Boolean Reasoning (BR) algorithm, Entropy/MDL algorithm and the Naïve algorithm. The results are tabulated in Table 2.3.

<table>
<thead>
<tr>
<th>Discretization Algorithm</th>
<th>Evaluation Criteria</th>
<th>Reducts</th>
<th>Rules</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>EFB (with 4 Data Cuts)</td>
<td></td>
<td>182</td>
<td>1004</td>
<td>86.8 %</td>
</tr>
<tr>
<td>BR</td>
<td></td>
<td>2</td>
<td>1510</td>
<td>57.7 %</td>
</tr>
<tr>
<td>Entropy</td>
<td></td>
<td>2</td>
<td>484</td>
<td>64.5 %</td>
</tr>
<tr>
<td>Naïve Algorithm</td>
<td></td>
<td>32</td>
<td>31188</td>
<td>43 %</td>
</tr>
</tbody>
</table>

From Table 2.3 it is clear that the EFB gave the best results and that all other parameter manipulations from here on would be based on the premise that the EFB is likely to give the best results. This selection of EFB as the discretization algorithm prompted the addition of an additional parameter, that of the number of data cuts. The relationship between this and accuracy is shown in Table 2.4.

<table>
<thead>
<tr>
<th>EFB Cuts</th>
<th>Evaluation Criteria</th>
<th>Reducts</th>
<th>Rules</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td>171</td>
<td>943</td>
<td>66.0 %</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>182</td>
<td>1004</td>
<td>86.8 %</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>190</td>
<td>1856</td>
<td>77.3 %</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>197</td>
<td>2450</td>
<td>76.1 %</td>
</tr>
</tbody>
</table>
As shown in Table 2.4, it was found that 4 data cuts gave the best results, from which other manipulations on parameters can be based. In this regard, it was found that the Standard Voting classifier gave the best accuracy. The other classifier methods used was the Standard Voting with Object Tracking method [71].

Using the standard voting classifier, the difference in accuracy and rules generated from the EFB (4 data cuts) with the normal reducts and the core reducts are tabulated in Table 2.5.

<table>
<thead>
<tr>
<th>Reducts</th>
<th>Rules</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>1004</td>
<td>86.8 %</td>
</tr>
<tr>
<td>Core Reduct</td>
<td>246</td>
<td>80.4 %</td>
</tr>
</tbody>
</table>

As evidenced by Table 2.5 above there exists a tradeoff between the number of rules generated and the accuracy of the model. Whilst we endeavor to make the prediction model as compact as possible (i.e with as few rules as possible), the consideration needs to be given to the accuracy of the model.

The Rough Set Model’s confusion matrix is shown in Figure 2.6 for the core reduct based system.

<table>
<thead>
<tr>
<th>Actual</th>
<th>Predicted</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>147</td>
<td>14</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>43</td>
<td>87</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.7736</td>
<td>0.8613</td>
</tr>
</tbody>
</table>
2.4 Conclusion

This chapter has presented a rough set theory based predictive model for stock prices. The data is randomly split in a 75/25 ratio into two data sets, the training and validation data sets. To increase the efficiency of the model, the data set is discretised using Equal Frequency Binning with 4 data cuts.

Next, the reducts are extracted using the Genetic Algorithm and finally the set of dependency rules are generated from the set of reducts. A rough set confusion matrix is used to assess the accuracy of the model. The model gave a prediction accuracy of 80.4% using the Standard Voting classifier.

Based on data obtained from the daily movements of the Johannesburg stock Exchange’s All Share Index the model showed a high degree of accuracy.
Chapter 3

An Ensemble of Rough Set Theory & Artificial Neural Networks: The Rough Neural Network

This chapter introduces the rough neural network - a hybrid model that encompasses RST and the canonical form of the artificial neural network (ANN). The RST component of the model is developed using the methods described in Chapter 2.
3.1 The Artificial Neural Network

The development of artificial neural networks (ANNs) constituted some of the earliest work in artificial intelligence [23]. There are fundamentally two approaches that are used in modelling, namely the universal approach and the local approach. Alternatively one can interpret this as modelling in the time domain as opposed to the feature/trajectory domain [13].

The canonical form of the ANN is considered an instance of the former approach. More specifically multi-layer perceptron neural networks can be regarded as a non-linear generalization of the autoregressive moving average (ARMA) [13]. The ARMA has long been used in economics and finance because it is generally considered a good way to model a system which is a function of unobserved and sporadic shocks.

In most cases the network consists of two layers of adaptive weights and with full connectivity between inputs and hidden units and between hidden units and outputs. In general, the multi-layer perceptron model is capable of universal approximation.

That is to say, it can approximate to arbitrary accuracy any continuous function provided the weights and biases are chosen appropriately. In practice this means that provided that there is enough data an MLP can model any smooth function.

The simple mathematical model of the neuron was proposed in [17]. Generally speaking each neuron “fires” when a linear combination of its inputs exceeds a given threshold [23]. The multi-layer perceptron model (MLP) contains multiple layers of a network of these neurons.

In general, the ANN is an attractive modeling tool because it possesses the following qualities:

- Learning ability – because each synaptic weight to each neuron can be adjusted, either manually or using an optimization algorithm, the ANN displays learning abilities,
- Universal approximation – its structure allows the ANN to inherently approximate any arbitrary function,
- Parallelism – each individual neuron does its computation independently of any other but all neurons that constitute the network work towards a common goal.
As the basic component of ANN, the neuron, is described first. Let an input vector to a neuron be $x$, a neuron is, therefore, a function $f(x)$ that transforms $x$ into $y$. The basic input-output function of a neuron is as shown in equation (3.1).

The input of the $j^{th}$ hidden unit is a weighted linear combination of the $d$ input values.

\[ in_j = \sum_{i=0}^{d} w_{j,i}^{(1)} x_i \]  

(3.1)

Where, $w_{j,i}$ is the weight in the 1st layer going from input $i$ to hidden input $j$.

The value $w_{j,0}$ considered to be the bias. The bias is included in order to prevent a zero output summation. The output of each unit is obtained by transforming the inputs using an activation function $g$ – we will return to the aspect of activation functions later [17].
In a two layer model the activation function \( g \) is referred to as an inner function. The output \( a_j \) is used as an input to the next layer. As with the hidden layer, each neuron in the output layer will sum all its inputs. The input of the \( m^{th} \) output neuron is [17]:

\[
  i_{n_m} = \sum_{j=0}^{k} w_{m,j} a_j
\]  

(3.3)

The \( m^{th} \) neuron also requires its own activation function, say \( g_2 \). So the output, \( y \), of each neuron in the output layer then becomes [17]:

\[
y = g_2 \left( \sum_{m=0}^{k} (i_{n_m}) \right)
\]  

(3.4)

Using (3.3) and (3.2) we get [17],

\[
y = g_2 \left( \sum_{m=0}^{k} w_{m,j} g_1 \left( \sum_{i=0}^{d} w_{j,i} x_i \right) \right)
\]  

(3.5)

We will now briefly discuss an aspect mentioned earlier in this chapter, that of the choice of the activation function \( g \).

**Activation Function**

There are many possible choices for the non-linear activation functions in a multi-layered network, and the choice of activation functions for the hidden units may often be different from that for the output units. This is because hidden and output units perform different roles [41, 44, 60].

\[
a_j = g_1 (i_{n_j}) = g_1 \left( \sum_{i=0}^{d} w_{j,i} x_i \right)
\]  

(3.2)
The fundamental problem is to find an activation function that can represent the different mappings between smooth variables. The use of the sigmoid function (3.6) in the output of a neuron has the advantage of ensuring a probabilistic interpretation of each output [22, 27, 41, 50].

The sigmoid function is [59]:

\[ g(a) = \frac{1}{1+e^{-a}} \]  \hspace{1cm} (3.6)

Equation (3.6) is often used as an activation function for the hidden layer(s) in a MLP model [36, 40, 43-46, 55, 60]. However, there are some minor but practical advantages of using the hyperbolic tangential function as an activation function.

The hyperbolic tangential function is given by [59]:

\[ g (a) = \tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}} \]  \hspace{1cm} (3.7)

It is important to note that the activation function in (3.6) and (3.7) only differ in through a linear transformation of the inputs and outputs. That is to say, \( g (\alpha) = \tanh(\alpha) \) is equivalent and equal to \( g(\alpha) = 1/(1 + e^{-\alpha}) \) if we perform the following linear transformations at the inputs and outputs respectively [59]:

\[ \alpha = \frac{a}{2} \]  \hspace{1cm} (3.8)

And,

\[ g(\alpha) = 2g(a) - 1 \]  \hspace{1cm} (3.9)

Accordingly the result is that two ANNs with the hyperbolic tangent function and the sigmoid function will be equivalent but with different synaptic weights and biases. Many scholars have found empirically that the use of the hyperbolic tangential function gives rise to faster convergence of training algorithms [1, 17, 22, 27, 30, 33, 40, 51-53, 60].
Bishop in [59] found that the MLP can generate an arbitrarily complex decision boundary and thus enabling it to model any “arbitrarily complex continuous function” [59].

It is this property of the MLP and its inherent ability to adapt to non-linear and complex mappings that makes it attractive to many scholars as a prediction tool [5, 10, 16]. The Universal Approximation Theorem as proposed by Haykin (1999) proved that a two-layered architecture suffices for the MLP model [20].

Accordingly, the ANN architecture chosen in this work is the two layered MLP model. The general model is shown in Figure 3.3.
3.2 ANN Training

The purpose of training in neural networks is not necessarily to learn an exact representation of the training data but to use the training data and statistics to build a model which generates that data. This is very important in the effectiveness of the network in approximating any given function [60].

Given some n-dimensional data vector \( x_n \) and a corresponding target vector \( t_n \), the process of training is to adjust the synaptic weights of the network such that:

\[
y_i = f(x_i, w)
\]  

(3.10)

Where \( x_i \) is the \( i^{th} \) instance of unseen data and \( w \) is the set of weights learnt from examples. This is an example of supervised learning since it is trained with \( x_n \rightarrow t_n \). The general supervised learning methodology is illustrated in Figure 3.3.

![Figure 3.3. General Supervised Learning Illustration](image-url)
When training an ANN the danger exists that the network might be over trained and “over fit” the data. That is to say, that the network might know the training data so well that it confirms to noise that might exist in the training data [7, 10, 15, 16, 44, 58].

So in training the network when the validation error is its smallest, since for most algorithms the error is monotonically decreasing function of the iteration index (the number of training epochs undergone).

**Back Error Propagation**

MLP uses error back propagation for the supervised training process. The objective function for training an MLP is given by (3.11) [59].

\[
E(w) = \frac{1}{2} \sum_{i=1}^{N} \| y_i - t_i \| \tag{3.11}
\]

The error calculated in (3.11) is back propagated into the MLP until it is within an acceptable range or other training criteria are met. An optimization method, which uses the derivative of the sum-of-square of error, \( E(w) \), to tune the weight matrix, is used for this purpose.

Back propagation tunes the weight matrix by employing (3.12) [59].

\[
w_{new} = w_{old} - \eta \cdot \frac{\partial w_{old}}{\partial w} \tag{3.12}
\]

Where \( \eta \) is the learning rate and,

Where \( w_{new} \) and \( w_{old} \) are the new and old weight matrices respectively.

Whilst the term \( \frac{\partial w_{old}}{\partial w} \) represents the error gradient of the old weight matrix used to optimize the new weight matrix. The weight optimization method used in this work is the scaled conjugate gradient (SCG) due to its computational efficiency.
3.3 The Rough Neural Network

The paradigm of rough neuro-computing draws its inspiration from RST. Historically rough neuro-computing rests on three main threads, namely: the production of a training set description, calculus of granules and interval analysis. Since its introduction by Lingras in the mid-1990’s it has attracted much interest in academia [40-44, 55, 58-60, 65]. The focus of development in this area has been on the development of a training set description and inductive learning based on knowledge reduction algorithms.

Another area in rough neuro-computing is the granule construction in a distributed system of intelligent agents (neural networks) [15]. Zadeh introduced the concept of the intuitive formulation of information granulation where intelligent agents (neurons) acquire knowledge by granulating and approximating inputs from other neurons.

The adaptive calculus of granulisation has provided a means through which granules (objects which have been assembled together by virtue of indiscernability or more generally their resultant functionality) can be interpreted by neural networks. Lingras in [15] inferred that in the context of a universal information system this granulisation of data objects could be done using rough set theory.

Lingras subsequently coined the term “rough neuron” [15]. Investigations into this field have led to the discovery of many subtle variations to the use of the “rough neuron” [15], namely the interval based rough neuron, the approximate neuron and the decider neuron.

Interval based rough neurons are useful in applications of the characterization of each object is based on a finite set of features, in which each feature would be defined by a upper and lower bounds (rough sets) as shown in [16, 17, 21].
Each neuron would use these values in each computation and aggregate into a single result the results obtained using both values.

Peters, in [25], explains that the approximate neuron measures the degree to which the approximation of the finite set of neuron inputs overlaps with the equivalence class containing measurements derived from known objects. The decider neuron implements a collection of decision rules by:

- creating an experimental conditional vector $C_{exp}$ from inputs of a rough membership function [38,39],

- Derive a set of decision rules with $C_i \implies D_i$ based on a condition vector that closely matches the experimental condition vector the most and,

- The result is logically ANDed with the relative error term.

The decider neuron can be seen in Figure 3.4.

Where the upper input comes from an approximation neuron, as a set of measurements in response to a stimulus provided by a new object, $f$, requiring classification. This is used to create $C_{exp}$. The lower input is a set of rules that have been derived from a decision table using RST.

Recent applications of rough neuro-computing have included speech and signal analysis, classification of power system faults, transformer bushings fault detection and the control of autonomous vehicles.
A rough mereological approach to rough neural networks springs from an interest in knowledge synthesized (induced) from successive granule approximations performed by neurons (cooperating agents) [44].

The distributed agent model for a neural network leads naturally to non-layered neural network architectures, that is, it is possible for an agent (neuron) to communicate granules of knowledge to other agents (neurons) in its neighbourhood rather than following the usual restricted model of a movement of granules “forward” from neurons in one layer to neurons in the next layer.

For this reason, the distributed agent model for rough neuro-computing is reminiscent of the Wiener model for message passing between neurons in the human nervous system [68] and, more recently, the swarm intelligence model.

The focus of this work has been on a rough neuro-computing model based on adaptive calculus of granules.

The fundamental feature of a granulation system is the exchange of information granules of knowledge between agents by transfer functions induced by rough connectives extracted from information systems. A calculus of granules has been introduced to provide a foundation for the design of information granulation systems.

The keystone in such systems is the granularity of knowledge for approximate reasoning by agents [42]. Approximate reasoning on information granules is not only caused by inexactness of information that we have but also by the fact that we can gain efficiency in reasoning if it is enough to deliver approximate solutions, sufficiently close to ideal solutions.

In Chapter 2, we provided the theoretical foundations of Rough Set Theory (RST). We also defined a all equivalence classes of the indiscernability relation, $I(B)$, as $B$-granules or $B$-elementary sets. We can express this mathematically through (3.13) [71].
\[ Ind_A(B) = \forall a \in B, \ a(x) = a(\hat{x}) \]  

(3.13)

We now introduce \([x]_B\) which denotes the equivalence class of \(Ind_A(B)\) defined by \(x\). In this section, a set function form of the traditional rough membership function introduced.

Let \(S = (U, A)\) be an information system, \(B \subseteq A\) and let \([u]_B\) be an equivalence class of an object \(u \in U\) of \(Ind_A(B)\). A set function \(\mu_u^B : \emptyset(U) \rightarrow [0, 1]\) defined by (3.14):

\[
\mu_u^B(X) = \frac{\text{card}(X \cap [u]_B)}{\text{card}([u]_B)}
\]  

(3.14)

is called the rough membership function.

A rough membership function provides a classification measure in as much as it tests the degree of overlap between the set \(X\) and the equivalence class.

The rough neuro-computing model was developed to model the South African economy using quarterly economic data from 1980 to 2010. This thesis analyses economic indicators and designed a model based on ten of these. These attributes or indicators are shown in Table 3.1 below.

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construction contractors rating lack of skilled labour as constraint</td>
<td>%</td>
</tr>
<tr>
<td>Tertiary economic sector contribution to GDP</td>
<td>Millions of SA Rands</td>
</tr>
<tr>
<td>Income velocity of circulation of money</td>
<td>Rands/year</td>
</tr>
<tr>
<td>Total manufacturing production volume</td>
<td>Kg</td>
</tr>
<tr>
<td>Manufacturing firms rating lack of skilled labour as constraint</td>
<td>%</td>
</tr>
<tr>
<td>Total asset value of banking industry</td>
<td>Millions of South African Rands</td>
</tr>
<tr>
<td>Nominal unit labour cost</td>
<td>South African rands</td>
</tr>
<tr>
<td>Total mass of PGMs mined</td>
<td>Kilograms</td>
</tr>
</tbody>
</table>
As with most modelling problems it was inherent in this thesis to prepare the raw data for use in the model. Accordingly, the data preparation phase took shape in a four stage process that includes: Data pre-processing, data discretization and for the rough set model redundant attribute elimination, reduct generation and rule generation.

The neural network component of this design was implemented with the MATLAB neural networking toolbox, Netlab [72]. The rough neuron is a pair \((r_l, r_u)\) where \(r_l\) is the lower neuron and \(r_u\) the upper neuron. In effect, the rough neuron stores the upper and lower bounds of the input value of a an attribute and uses it in its computation. The output of the rough upper neuron and rough lower neuron will be computed as in (3.15) and (3.16) respectively.

\[
Out_{r_l} = \min \left( f(in_{r_l}), f(in_{r_u}) \right) \tag{3.15}
\]

\[
Out_{r_u} = \max \left( f(in_{r_l}), f(in_{r_u}) \right) \tag{3.16}
\]

The basic process undergone in the design of this model as in Figure 3.5:
3.4 The Rough Neuro-Computing Model

As we have described the theoretical foundations of the rough neuro-computing model we will now give a description of the internal machinations that inform the model developed for this work. That having been said, it is important for one to recall that the primary objective of developing a classification model is to achieve the best possible classification accuracy [60, 63, 68, 70].

It is generally accepted that there is no classifier that can be considered the single best solution to a given data set [51, 53, 60]. One of the main advantages to be gained from using a classifier ensemble resides in its ability to retain model diversity (complexity) whilst simultaneously increasing accuracy.

The act of combining classifiers is done primarily using two techniques. The first one uses the posterior probability by employing a different feature set - Bagging [65], employing different training sets - Boosting [66] and using outputs from different classifier as inputs to a new classifier [63]. The disadvantage of these techniques is that they are computationally expensive.

The second and popular technique uses the output class labels form the classifier to form an ensemble. Kittler et al. [62] implemented the product rule, sum rule, max rule, min rule, median rule and the majority voting techniques and proved that the majority voting is one of the best ensemble methods. Hence, this work only focuses on the majority voting scheme [67][68].

One of the principal reasons behind the combination of classifiers in an ensemble is to improve generalization and classification accuracy by harnessing the abilities of diverse classifiers. Methods for measuring diversity are
classified into two categories. Namely: structural based and outcome based [70]. Mpanza et al. [68] proposed a list of ten measures of diversity: six structural and four outcomes based.

Table 3.2 shows a two classifier diversity confusion matrix where \( C_j \) and \( C_i \) are the two classifiers in question.

<table>
<thead>
<tr>
<th></th>
<th>( C_i ) correct</th>
<th>( C_i ) incorrect</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_j ) correct</td>
<td>( X_{both\text{,correct}} )</td>
<td>( X_{Cj} )</td>
</tr>
<tr>
<td>( C_j ) incorrect</td>
<td>( X_{Ci} )</td>
<td>( X_{both\text{,incorrect}} )</td>
</tr>
</tbody>
</table>

From Table 3.2 it follows that variables: \( X_{both\text{,correct}} \), \( X_{both\text{,incorrect}} \), \( X_{Cj} \), and \( X_{Ci} \) represent the frequency of both classifiers being correct, both classifiers being incorrect, only classifier \( C_j \) being correct and only classifier \( C_i \) being correct respectively.

We will now briefly look at the four type of diversity measures that can be deduced from Table 3.2. These are: the Q-statistic, Correlation, Disagreement and Double Fault.

**The Q-Statistic**

Q statistic is the measure of the degree of similarity or dissimilarity of two classifiers. It is described by [60, 65, 67]:

---

\[ Q = \frac{N - \sum X_{both\text{,correct}} - X_{both\text{,incorrect}} - (N - X_{Cj} - X_{Ci})}{\sqrt{X_{Cj} X_{Ci}}} \]
\[
\frac{x_{\text{both correct}} \cdot x_{\text{both incorrect}} - x_{C_i} x_{C_j}}{x_{\text{both correct}} \cdot x_{\text{both incorrect}} + x_{C_i} x_{C_j}}
\]  
(3.17)

It follows from (3.17) that the maximum diversity is zero.

**Correlation**

Correlation is the measure of correlation of outputs of two classifiers and is defined as [60, 65, 67]:

\[
\frac{x_{\text{both correct}} \cdot x_{\text{both incorrect}} - x_{C_i} x_{C_j}}{\sqrt{(x_{\text{both correct}} + x_{C_i})(x_{\text{both incorrect}} + x_{C_j})(x_{\text{both incorrect}} + x_{C_j})(x_{\text{both correct}} + x_{C_j})}}
\]  
(3.18)

From (3.18) it is clear that the maximum correlation would have to be zero.

**Disagreement**

Disagreement is the frequency of disagreement between two classifiers. Disagreement is defined as [60, 65, 67]:

\[
\frac{x_{C_i} + x_{C_j}}{x_{\text{both correct}} + x_{\text{both incorrect}} + x_{C_i} + x_{C_j}}
\]  
(3.19)

**Double Fault**

Double fault is the frequency of both classifiers classifying a class incorrectly. It is defined by (3.20) below [60, 65, 67]:

\[
\frac{x_{C_i} + x_{C_j}}{x_{\text{both correct}} + x_{\text{both incorrect}} + x_{C_i} + x_{C_j}}
\]  
(3.20)
\frac{x_{both\ incorrect}}{x_{both\ correct} + x_{both\ incorrect} + x_C + x_{Cj}}
The multi-layer perceptron is trained in the Matlab development environment using Netlab. The number of input units is set to nine, based on the data described in Table 3.1, and one output unit (Gross Domestic Product (GDP) growth rate as a percentage).

A two layer network is developed. This model is optimized using the particle swarm optimization (PSO) algorithm / method. The MLP classifier confusion matrix is shown in Table 3.3.

<table>
<thead>
<tr>
<th>Predicted Positive</th>
<th>Predicted Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual Positive</td>
<td>20</td>
</tr>
<tr>
<td>Actual Negative</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 3.4 shows an example of the rules that were generated.

Rules

BEGIN,

1) IncomeVelocity([12, *]) AND Manufacturing([39, *]) AND ManufacturingTotal([51, *]) AND PGMSalesValue([4.45334005, *]) AND GDE([9, *]) => Decision(2) 1 1 1.0 0.011494 0.05 1.0 5 1

2) IncomeVelocity([12, *]) AND Manufacturing([39, *]) AND ManufacturingTotal([51, *]) AND PGMSalesValue([1.64997995, 4.45334005]) AND GDE([9, *]) => Decision(2) 1 1 1.0 0.011494 0.05 1.0 5 1

3) IncomeVelocity([11, 12]) AND Manufacturing([39, *]) AND ManufacturingTotal([51, *]) AND PGMSalesValue([*, -8.86100006]) AND GDE([6, 9]) => Decision(1) 1 1 1.0 0.011494 0.02 1.0 5 1

4) IncomeVelocity([11, 12]) AND Manufacturing([39, *]) AND ManufacturingTotal([51, *]) AND PGMSalesValue([1.64997995, 4.45334005]) AND GDE([6, 9]) => Decision(2) 1 1 1.0 0.011494 0.05 1.0 5 1

5) IncomeVelocity([11, 12]) AND Manufacturing([39, *]) AND ManufacturingTotal([51, *]) AND PGMSalesValue([4.45334005, *]) AND GDE([9, *]) => Decision(2) 1 1 1.0 0.011494 0.05 1.0 5 1
6) IncomeVelocity([11, 12]) AND Manufacturing([39, *]) AND ManufacturingTotal([51, *]) AND PGMSalesValue([*, -8.86100006]) AND GDE([-4, 2]) => Decision(1) 1 1 1.0 0.011494 0.02 1.0 5 1
7) IncomeVelocity([12, *]) AND Manufacturing([-16, 3]) AND ManufacturingTotal([45, 51]) AND PGMSalesValue([*, -8.86100006]) AND GDE([*, -4]) => Decision(0) 1 1 1.0 0.011494 0.058824 1.0 5 1
8) IncomeVelocity([12, *]) AND Manufacturing([*, -16]) AND ManufacturingTotal([41, 45]) AND PGMSalesValue([4.45334005, *]) AND GDE([*, -4]) => Decision(0) 1 1 1.0 0.011494 0.058824 1.0 5 1
9) IncomeVelocity([12, *]) AND Manufacturing([*, -16]) AND ManufacturingTotal([34, 41]) AND PGMSalesValue([4.45334005, *]) AND GDE([*, -4]) => Decision(0) 2 2 1.0 0.022989 0.117647 1.0 5 1
10) IncomeVelocity([12, *]) AND Manufacturing([-16, 3]) AND ManufacturingTotal([34, 41]) AND PGMSalesValue([-8.86100006, -6.21966982]) AND GDE([6, 9]) => Decision(1) 1 1 1.0 0.011494 0.02 1.0 5 1

This chapter has presented a generic rough neuro-computing model that can be used to predict the direction in which the GDP of the South African economy will turn dependant on ten (10) parameters. Though the results of the rough neuro-computing model are documented in detail in Chapter 5, the model has shown good accuracy and this work can be used by other researchers in this field and compliment other tools of analysis used by economic policy analysts.
Despite many advances in the use of computational intelligence techniques for missing data imputation over the last three decades, this problem remains largely unsolved. Many techniques have emerged in the literature as candidate solutions, including the Expectation Maximization (EM), and the combination of auto-associative neural networks and genetic algorithms (NN-GA).

The scope of this chapter is limited to research on the effectiveness of both the EM and NN-GA algorithms in estimating missing economic data.
4.1 Introduction

Decision making processes are highly dependent on the availability of data, from which information can be extracted. All scientific, business and economic decisions are in a way related to and dependent on the information available to them at the time of making such decisions.

Even advances in decentralized decision making theory have been unable to disprove the importance of data availability in decision making, as it follows that each decision maker (processor) will make a specific decision based on available data or data available to other decision makers (processors). There are many situations where, the data which must inform a particular decision is, at best, partially corrupt or, at worst, incomplete [55, 70, 71, 75].

For this reason, it is often essential to make the decision based on available data. Most decision making tools such as the commonly used neural networks, support vector machines and many other computational intelligence techniques cannot be used for decision making if data are not complete.

In such cases, the decision output should still be maintained despite the missing data. Accordingly the first step in cases where data are incomplete is to complete the missing values. Once missing values have been estimated, pattern recognition tools for decision making can thereafter be used [68].

In most applications, solving the problem of missing data is a cumbersome and extraneous exercise that is not the main focus in a decision making task or application. This therefore imposes a quick and efficient technique to handle the problem of missing data. This necessarily raises conceptual, computational and practical challenges [22, 30, 73, 80].

Various heuristics of missing data imputation such as mean substitution, which is the substitution of the missing variable by the mean of the observed data for that data field and hot deck imputation may be used, but these in turn depend on and require prior knowledge of how data points become missing.

There are several reasons why data might be missing, and these missing data may follow an observable pattern.

There are primarily three common ways in which the problem of missing data is solved. These are [64]:
• “Listwise deletion” – this method simply deletes instances of data with missing values,

• Finding estimates of the missing values – in this method, estimates of the missing data are found and instances with missing data are attributed to these estimates. Various statistical estimates may be used such as means, zeros, averages etc.

• The third general technique assumes some model for the prediction of the missing values and uses the maximum likelihood approach to estimate the missing values.

Figure 4.1 above shows missing data patterns where $X_p$ and $Y_p$ are attributes in a database with $N$ rows or data points.

![Figure 4.4. Patterns of Missing Data (a) Univariate Pattern (b) Monotone Pattern and (c) Arbitrary Pattern [61]](image)

In Figure 4.1 (a) we see the univariate pattern in which the data is missing from a single variable or attribute. Figure 4.1(b) shows data missing from a number of variables or attributes but missing in a particular and distinguishable fashion whilst Figure 4.1(c) shows data missing in an arbitrary and random pattern.
In [60] described three (3) missing data mechanism or definitions. Namely: “missing at random” (MAR), “missing completely at random” (MCAR) and “missing not at random” (MNAR) [70].

In [64], they also observed that the missing data can, in general, be described by the equation:

\[ Y_{\text{complete}} = f(Y, X_{\text{observed}}) \]  

(4.1)

Where \( Y_{\text{complete}} \) is the complete data is set and \( X_{\text{observed}} \) are the observed values.

**4.1.1 Missing at Random (MAR)**

For data to be classified as missing at random necessarily requires or assumes that the cause of missing data is unrelated to the missing values themselves [68]. However the cause may be related to other variables in the data base [68].

More simply put, the cause of missing data is due to external influence and not to the variable in question. Suppose there are two sensors, \( S \) and \( T \), the probability of some datum \( d \) from sensor \( S \) to be missing is dependent on other measured variables from the same database.

**4.1.2 Missing Completely at Random (MCAR)**
This missing data mechanism relates is the condition when the probability of missing data for some sensor $S$ is not related to the measured values of any other sensor. That is to say, in this missing data mechanism, cases with complete data are indistinguishable from cases with incomplete data [68].

Thus the probability of sensor $S$ values missing is independent of observed values neither is it dependent on the previous state of the sensor.

### 4.1.3 Missing Not at Random (MNAR)

This missing data mechanism requires that it be related to the missing values. That is to say, in this condition, why some data are missing can be explained. However, this explanation is only dependent on the same variables that are missing and cannot be explained in terms of any other variables in the database [68].

An example of this is if, say some sensor fails after it reads a value above or below a certain threshold. So if that sensor measures some variable $K$ then the probability of missing values of $K$ is dependent on $K$ itself.

In [68] we find the definition of this case non-ignorable as the missing data as the missing observation is dependent on the outcome of interest. In this case, the readings from $S$ might be missing merely because sensor $T$ is not working.
4.2 Neural Network – Genetic Algorithm (NN-GA) Approach

Nelwamondo [70] found that, in solving the problem of missing data, it is important to understand why data is missing but further conceded that often such information cannot be known [70] and that in such cases it is important to employ imputation algorithms [70].

In this regard such algorithms as Expectation Maximization (EM) and auto-associative Neural Networks combined with Genetic Algorithms (NN-GA) have emerged as candidate solutions. For instance Nelwamondo and Marwala [69] used neural networks together with Genetic Algorithms (GA) to approximate missing data.

Pukel, Muhr and Lick [26] used neural networks and Particle Swarm Optimization (PSO) to keep track of the dynamics of a power plant in the presence of missing data. Mpanza and Marwala [53] have used Evolutionary computing in condition monitoring of high voltage (HV) bushings in the presence of missing data.

In their study, auto-associative neural networks were used together with GA or PSO to predict the missing values and also to optimize the prediction. Computational intelligence and maximum likelihood techniques can capture the interrelationships between observed and missing data and are therefore important for the problem of missing data imputation [70].

The focus of this chapter is to implement the NN-GA as described by Nelwamondo and Marwala [69] and compare its effectiveness in imputing missing data with the “traditional” Maximum Likelihood (ML) through the Expectation Maximization (EM) algorithm.

We will begin by introducing the auto-associative NN-GA and then go on to describe the ML and EM algorithms.
Auto-associative neural networks, also known as auto-encoders, are neural networks which can recall their input space [22, 44, 55, 70]. Auto-encoders have a remarkable ability to learn certain linear and non-linear interrelationships such as correlation and covariance inherent in the input space.

Auto-associative neural networks project the input onto some smaller set by intensively squashing it into smaller details. Thompson [60] found that the optimal number of the hidden nodes of the auto-encoder, although dependent on the application, must be smaller than that of the input layer.

In this work, the auto-associative neural network is constructed through the use of the multi-layer perceptron model (MLP). The MLP was trained using the back-propagation algorithm. As we have described in Chapter 3, MLPs are feed-forward neural networks with an architecture composed of the input layer, the hidden layer and the output layer.

Each layer is formed from small units known as neurons. Neurons in the input layer receive the input signals $\vec{X}$ and distribute them forward to the rest of the network. In the next layers, each neuron receives a signal, which is a weighted sum of the outputs of the nodes in the previous layer.

Inside each neuron, an activation function is used to control the input. Such a network determines a non-linear mapping from an input vector to the output vector, parameterized by a set of network weights, which are referred to as the vector of weights, $\vec{W}$.

The structure of the auto-encoder used in this work is shown in Figure 4.2.
Since this chapter follows from the chapter we have thought it to be futile to re-derive, from first principles, how each output neuron, represented in Figure 4.2, is composed.

However, it is worth mentioning that the first step in approximating the weight parameters of the model is finding the appropriate architecture of the MLP, where the architecture is characterized by the number of hidden units, the type of activation function, as well as the number of input and output variables. The second step estimates the weight parameters using the training set [68-70].
Genetic Algorithms

Most real world problems require the optimization of many and sometimes competing criteria. Traditionally, this problem has been overcome by combining these criteria into a single criterion to be optimized according to some utility function [13].

There are multiple optimization techniques that are aimed at optimizing some attribute to conform to some target function. In some instances these methods converge at local optimal solutions instead of the required global optimal solutions. Although stochastic in nature, genetic algorithms (GAs) often converge to a global optimal solution [10, 16, 17, 29, 33, 70, 73].

The main thrust upon which GAs are built is embedded in the concept of survival of the fittest over consecutive generations to solve optimization problems. As in biological evolution, the fitness of each population member in a generation is evaluated to determine whether it will be used in the breeding of the next generation.

Evolutionary biological techniques such as inheritance, mutation, natural selection, and recombination are employed in the process of creating the next generation. The GA algorithm implemented in this chapter uses a population of string chromosomes, which represent a point in the search space.

The optimum level of operation of the GA heavily depends on the following parameters: crossover rate, mutation rate, population size, chromosome type and generational gap. The population approach and multiple sampling make GAs less susceptible or vulnerable to becoming trapped at local minima than traditional direct optimization approaches such as constrained conjugate gradient [61].

One of the advantages of using a GA is that it has the inherent ability to navigate large solution spaces with an efficient number of samples. Although it is not guaranteed to provide the globally optimal solution, GAs have proved to be very efficient at reaching a near optimal solution in a computationally efficient manner. More details on GAs can be found in [61] and [62]. The general structure of a GA is shown below.
Algorithm 1: The Genetic Algorithm

BEGIN,

1) Create Initial Population beginning at an initial generation, \( g = 0 \);

2) Repeat Until Terminated;

3) Evaluate each individual's fitness using some fitness function, \( f \);

4) Prune population (eliminate the weakest individuals);

5) Select pairs to mate from the fittest individuals in the population;

6) Replenish population using crossover operator or mutation operator;

7) Check for the termination criteria;

8) If (not terminated)
   go back to 2);

   Else
   End;

The method used in this work combines both auto-associative neural networks and genetic algorithms. In this architecture, an auto-encoder is trained from complete data. GAs are then used in this work, to approximate missing data. A similar method was developed to approximate missing data by Nelwamondo and Marwala [69].

Other researchers have investigated the use of algorithms other than GAs to solve this problem. For instance, Dhlamini [60] found GAs to be better than Simulated Annealing and Particle Swarm Optimization (PSO) in terms of speed of convergence.

Due to this reason, in the comparative study of this chapter, a genetic algorithm is used to estimate the missing values by optimizing an objective function.

Figure 4.3 shows a pictorial description of this system.
Cognizant of the fact that the method uses an auto-encoder, it is warranted for one to expect the input to be very similar to the output if we have chosen architecture of the auto-encoder well [33, 70]. This is, however, only expected on a dataset similar to the problem space from which the inter-correlations have been captured [70].

The difference between the target and the actual output is used as the error, $E$. This error is defined by (4.2).
\[ E = \vec{x}_i - \vec{x}_o \]  \hspace{1cm} (4.2)

Where \( \vec{x}_i \) and \( \vec{x}_o \) are the input and output vectors respectively.

We can further expand (4.2) as the output vector, \( \vec{x}_o \), is essentially a function of the weight vector, \( \vec{W} \), and the input vector \( \vec{x}_i \). Accordingly, we can (4.2) as:

\[ E = \vec{x}_i - f(\vec{W}, \vec{x}_i) \]  \hspace{1cm} (4.3)

In order to be certain the error function is always positive we square it. Thus we have:

\[ E = (\vec{x}_i - f(\vec{W}, \vec{x}_i))^2 \]  \hspace{1cm} (4.4)

Since the input vector consist of the known, \( X_k \), and unknown, \( X_u \), entries, the error function, \( E \), can be written:

\[ E = \left( \begin{bmatrix} X_k \\ X_u \end{bmatrix} - f(\vec{W}, \begin{bmatrix} X_k \\ X_u \end{bmatrix}) \right)^2 \]  \hspace{1cm} (4.5)

Equation (4.5) is used as the objective function that is minimized using the GA.
4.3 Results and Discussion

The algorithm was tested on South African economic data statistics from January 1960 to December 2012. The data was transformed using a min-max normalisation to [0,1]. This is done to ensure that the data are within the active range of the activation function of the neural network.

The effectiveness of the missing data system is evaluated using the correlation coefficient and the relative prediction accuracy. The correlation coefficient was used as a measure of similarity between the prediction and the actual data whilst the relative prediction accuracy can be interpreted as a measure of how many of the missing values are predicted within the tolerance and the tolerance can be made to be any value depending on the sensitivity of the application.

In this case the tolerance was made to be 15% but this figure can be changed depending on the required accuracy of the application.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Prediction Correlation Coefficient</th>
<th>Relative Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construction contractors rating lack of skilled labour as constraint</td>
<td>0.941</td>
<td>95.6</td>
</tr>
<tr>
<td>Tertiary economic sector contribution to GDP</td>
<td>0.821</td>
<td>98.7</td>
</tr>
<tr>
<td>Income velocity of circulation of money</td>
<td>0.791</td>
<td>88.0</td>
</tr>
<tr>
<td>Total manufacturing production volume</td>
<td>0.901</td>
<td>94.0</td>
</tr>
<tr>
<td>Manufacturing firms rating lack of skilled labour as constraint</td>
<td>0.831</td>
<td>90.0</td>
</tr>
<tr>
<td>Total asset value of banking industry</td>
<td>0.776</td>
<td>88.7</td>
</tr>
<tr>
<td>Nominal unit labour cost</td>
<td>0.891</td>
<td>95.8</td>
</tr>
<tr>
<td>Total mass of PGMs mined</td>
<td>0.888</td>
<td>97.1</td>
</tr>
</tbody>
</table>
4.4 Conclusion

This chapter has presented and discussed an auto-associative Neural Network combined with Genetic Algorithms (NN-GA) in its application to missing economic data imputation. The algorithm was used on data that contains at least ten economic variables. The results were compared with other missing data imputation algorithms developed by other scholars (Nelwamondo in [70]). The results, indicate that the NN-GA method is effective in imputing data where there exists a lot of dependency between variables.
Chapter 5

Conclusion and suggested future works
5.1 Summary of findings

The goal of this thesis was to investigate and develop computational intelligence techniques that might be used to model an economic system. Firstly a rough set theory based model of an important aspect of any economic system (the financial and capital markets) is presented.

The skeleton of this model is then used in the construction of a rough neuro-computing model of the South African economic system. In addition to this, in order to circumvent the challenge of missing data in our database we investigated the auto-associative neural network combined with genetic algorithms (NN-GA) as a tool for imputing missing economic data on a database.

Rough set theory (RST) is investigated because of its ability to deal with uncertain and vague data. The break of data collection tools and storage capacity has resulted in data analysis difficulty. RST is used to learn the patterns from the data to be able to predict new data cases. However, many scholars have proved that the effectiveness or any RST algorithm is heavily dependent on the type of discretization algorithm used.

To optimise the RST model, four discretization algorithms were tested at this stage of the model. The four methods are: EFB, Boolean Reasoning, Entropy and the Naïve Algorithm. The method selected in this regard is the equal frequency binning (EFB) as it showed the highest accuracy. What makes rough set an attractive tool is its rule-based nature which enables it to interpret its decisions. The lesser the number of rules, the easier it is to interpret.

The relationship between the accuracy and the interpretability was investigated. The results show that this relationship is inversely proportional. A certain increase in the interpretability can be achieved without significant loss of accuracy. Care must be taken to avoid significant loss of accuracy.

The RST model was later combined with the multi-layer perceptron model (MLP) to form an rough neuro-computing model. The ensemble functions as a stand-alone application and uses a majority voting scheme.

This investigation gave insight into the reason why the ensemble performs better than the
individual models. The performance is attributed to the fact that the ensemble inherits the generalization ability of both models.

5.2 Direction for Future Work

The dependency of the rough set model of the discretization algorithm leaves room for further investigation of discretization tools to find a better method. The accuracy versus transparency experiment carried out in this work can be improved by testing different data models and a variety of discretization tools.

An ensemble of more diverse classifiers can be implemented to validate the claim that more diversity will produce high generalization. The ability of computational intelligence methods to model an economic system is not sufficient to define a holistic and over-arching decision system.
References


[25] Jiang-Hong Man: An improved fuzzy discretization way for decision,


[38] A. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition, 30:1145-1159, 1997.


[70] FV Nelwamondo, Computational intelligence techniques for missing data imputation, PhD thesis, University of the Witwatersrand, Johannesburg, South Africa


Appendix A

Rough Set Theory Algorithm

This section presents the algorithm used to develop the rough set model.
Algorithm 1: Rough set theory algorithm

**Input:** Training and testing data in a form, \( A = (U, B \cup \{d\}) \)

**Begin**

1) Data processing

2) Data discreetization

3) for \( b \leftarrow 1 \) to \( \text{sizeof} \ U \) do

   - Compute the lower approximation, \( BX \)
   - Extract the rule from \( BX \);
   - Compute the upper approximation \( \tilde{BX} \);
   - Extract the rule from \( \tilde{BX} \);
   - Calculate the plausibility;
   - Calculate the support and coverage;

   end

4) Test the model

5) produce the confusion matrix

6) Produce the ROC curve

7) Compute the AUC

**end**
Appendix B

Published works

This section presents work that was the result of this study published to the 12th IEEE International Symposium on Computational Intelligence and Informatics (Budapest, Hungary) and the 2012 IEEE International Joint Conference on Neural Networks, World Congress on Computational Intelligence (Brisbane, Australia).
A Rough Set Model for Stock Price Prediction

Msizi Khoza
Faculty of Engineering and the Built Environment
University of Johannesburg
Johannesburg, South Africa
khozams@gmail.com

Tshilidzi Marwala
Faculty of Engineering and the Built Environment
University of Johannesburg
Johannesburg, South Africa
tmarwala@uj.ac.za

Abstract— Attempting to successfully and accurately predict the financial market has long attracted the interests and attention of economists, bankers, mathematicians and scientists alike. The financial markets form the bedrock of any economy. There are a large number of factors and parameters that influence the direction, volume, price and flow of traded stocks. This coupled with the markets’ vulnerability to external and non-finance related factors and the resulting intrinsic volatility makes the development of a robust and accurate financial market prediction model an interesting research and engineering problem. In an attempt to solve this engineering problem, the authors of this paper present a rough set theory based predictive model for the financial markets. Rough set theory has, as its base, imperfect data analysis and approximation. The theory is used to extract a set of reducts and a set of trading rules based on trading data of the Johannesburg Stock Exchange (JSE) for the period 1 April 2006 to 1 April 2011. To increase the efficiency of the model four of discretization algorithms were used on the data set, namely (Equal Frequency Binning (EFB), Boolean Reasoning, Entropy and the Naïve Algorithm. The EFB algorithm gives the least number of rules and highest accuracy. Next, the reducts are extracted using the Genetic Algorithm and finally the set of dependency rules are generated from the set of reducts. A rough set confusion matrix is used to assess the accuracy of the model. The model gave a prediction accuracy of 80.4% using the Standard Voting classifier.

Keywords—rough set theory; financial market modelling, neural networks, discretization, classification.

INTRODUCTION

Around the world, trading in the stock market has gained enormous popularity as a means through which one can reap huge profits. Attempting to successfully and accurately predict the financial market has long attracted the interests and attention of economists, bankers, mathematicians and scientists alike. Thus, stock price movement prediction has long been a cherished desire of investors, speculators and industries [1].

For a long time statistical techniques such as Bayesian models, regression and some econometric techniques have dominated research activities in prediction [2]. The primary approach to financial forecasting has been the identification of a stock price trend and continuation of the investment strategy until evidence suggests that the trend has reversed [3]. One of the biggest problems with the use of regression methods is that they fail to give satisfactory forecasting results for some series’ because of their linear structure and other inherent limitations [8, 9].

However the emergence of computational intelligence techniques as a viable alternative to the “traditional” statistical models that have dominated this area since the 1930’s [2,3] has given impetus to the increasing usage of these techniques in fields such as economics and finance [3]. Apart from these, there have been many other successful applications of intelligent systems to decision support and complex automation tasks [4 - 6]. Since the year of its inception in 1982, rough set theory has been extensively used as an effective data mining and knowledge discovery technique in numerous applications in the finance, investment and banking fields [4, 7, 11, 12, 20].

Data mining is a discipline in computational intelligence that deals with knowledge discovery, data analysis, and full and semi-autonomous decision making [7]. It entails the analysis of data sets such that unsuspected relationships among data objects are found. Predictive modeling is the practice of deriving future inferences based on these relationships. The financial markets form the bedrock of any economy. There are a large number of factors and parameters that influence the direction, volume, price and flow of traded stocks. This coupled with the markets’ vulnerability to external and non-finance related factors and the resulting intrinsic volatility makes the development of a robust and accurate financial market prediction model an interesting research and engineering problem. This paper presents a generic stock price prediction model based on rough set theory. The model is derived on data from the daily movements of the Johannesburg Stock Exchange’s All Share Index. The data used was collected over a five year period from 1 April 2006 to 1 April 2011.

The methodology used in this paper is as follows: data preprocessing, data splitting, data discretization, redundant attribute elimination, reduct generation, rule generation and prediction.
The rest of this paper is organized as follows: Section II covers the theoretical foundations of rough set theory. Section III gives the design of the proposed prediction model while an analysis of the results and conclusions are presented in Sections IV and V respectively.

**Theoretical Foundations of Rough Sets**

Rough set theory (RST) was introduced by Pawlak in 1982. The theory can be regarded as a mathematical tool used for imperfect data analysis [10]. Thus RST has proved useful in applications spanning the engineering, financial and decision support domains to mention but a few. It is based on the assumption that "with every object in the universe of discourse, some information (data or knowledge) is associated" [10]. In practical applications, the "universe of discourse" described in [10] is usually a table called the decision table in which the rows are objects or data elements and the columns are attributes and the entries are called the attribute values [3].

The objects or data elements described by the same attributes are said to be indiscernible (indistinguishable) by the attribute set. Any set of indiscernible data elements forms a granule or atom of knowledge about the entire “universe of discourse” (information system framework) [10]. A union of these elementary sets (granules) is said to be a precise or crisp set, otherwise the set is said to be rough [1,2,7,10]. Every rough set will have boundary cases i.e data objects which cannot certainly be classified as belonging to the set or its complement when using the available information [10].

Associated with every rough set is a pair of sets called the lower and upper approximation of the rough set. The lower approximation consists of those objects which one can definitively say belong to the target set. The upper approximation consists of those objects which possibly belong to the target set. The difference between the two sets is the boundary region. The decision rule derived specifies an outcome based on certain conditions. Where the derived rule uniquely identifies outcomes based on some conditions the rule is said to be certain else it is uncertain. Every decision rule has a pair of probabilities associated with it, the certainty and coverage coefficients [3].

These conditional probabilities also satisfy Bayes’ theorem [7,10]. The certainty coefficient is the conditional probability that an object that belongs to the decision class outlined by the rule given that it satisfies the conditions of the rule. The coverage coefficient on the other hand expresses the conditional probability of reasons given some decision [10]. Clearly RST can be seen to overlap with many other theories in the realm of imperfect knowledge analysis such as evidence theory, Bayesian inference, fuzzy sets etc [1, 3, 4, 10, 11, 12].

To define rough sets mathematically, we begin by defining an information system $S = (U, A)$, where $U$ and $A$ are finite and non-empty sets that represent the data objects and attributes respectively. Every attribute $a \in A$ has a set of possible values $V_a$. $V_a$ is called the domain of $a$. A subset of $A$ say $B$ will determine a binary relation $I(B)$ on $U$, which is called the indiscernibility relation. The relation is defined as follows: $(x, y) \in I(B)$ if and only if $a(x) = a(y)$ for every $a$ in $B$, where $a(x)$ denotes the value of attribute $a$ for data object $x$ [10]. $I(B)$ is an equivalence relation. All equivalence classes of $I(B)$ as $U/I(B)$. An equivalence class of $I(B)$ containing $x$ is denoted as $B(x)$. If $(x,y)$ belong to $I(B)$ they are said to be indiscernible with respect to $B$. All equivalence classes of the indiscernibility relation, $I(B)$, are referred to as $B$-granules or $B$-elementary sets [10].

In the information system defined above, we define as in [10]:

$$X \subseteq U$$

And,

$$B \subseteq A$$

We now define the two operators assigned to every (1) two sets called the upper and lower approximation of $X$. The two sets are defined as follows [10]:

$$B_u(X) = \bigcup_{x \in U} \{ B(X) : B(X) \subseteq X \}$$

And,

$$B_l(X) = \bigcup_{x \in U} \{ B(X) : B(X) \cap X \neq \emptyset \}$$

Thus, the lower approximation is the union of all B-elementary sets that are included in the target set, whilst the upper approximation is the union of all B-elementary sets that have a non-empty intersection with the target set. The difference between the two sets is called the boundary region of $X$.

$$BR = B_u(X) - B_l(X)$$

If the boundary region is an empty set then $X$ is crisp with respect to $B$, if however the boundary region is non-empty then $X$ is rough with respect to $B$. Accordingly, the set is said to be rough if it cannot be defined exactly from the available data. The set of attributes that is sufficient to represent the entire equivalence class structure is called the reduct.

The reduct of the information system is not unique. There are potentially many subsets of attributes which preserve the equivalence class structure. The set of attributes common to all reducts is called the core. The core can be regarded as the indispensable attribute of the information system. However, in practical applications where the information system contains thousands or possible tens of thousands of objects it is seldom that a core exists as shown in [1].
The design of the predictive model is based on rough set theory. The model comprises a set trading rules extracted from a set of reducts generated from discretized training data. The measure of the accuracy of the model is based on the percentage correctness of its predictions. The model comprises 6 components, as can be seen in Fig. 1 below.

**Data Pre-processing.**

This phase of the design includes activities such as additional attributes computation, filtering and data completion. There is a difference between time series with and without real time constraints. In the case of the latter, the only primary factor is the chronological order of events. As with stock price data and the model we sought to build the interval between events is a major contributing factor. To construct rough set objects from real time data we adopted a hybrid method that amalgamates the “columnizing method” the “mobile moving window” model proposed in [13]. The decision table is constructed by having columns as various technical indicators and the rows representing trading data at each point in time, whilst the window presents a “snapshot” of the state of the market in that period.

The data was randomly split in 75/25 ratio. The former part of the data is used for training and the remaining used for validation. There were a total of 10 attributes used. These are shown in Table 1 below.

**Table 1. Attributes**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>The opening price</td>
</tr>
<tr>
<td>High</td>
<td>The highest registered price on the day</td>
</tr>
<tr>
<td>Low</td>
<td>The lowest registered price on the day</td>
</tr>
<tr>
<td>Close</td>
<td>The closing price</td>
</tr>
<tr>
<td>Adjusted Close</td>
<td>The adjusted closing price</td>
</tr>
<tr>
<td>Moving Average</td>
<td>Moving average over 5 days</td>
</tr>
<tr>
<td>Momentum</td>
<td>( P_l - P_{l-4} )</td>
</tr>
<tr>
<td>Rate of Change (ROC)</td>
<td>( \frac{Momentum}{P_l} \times 100 )</td>
</tr>
<tr>
<td>Disparity</td>
<td>( \frac{P_l}{Moving \ Average} \times 100 )</td>
</tr>
<tr>
<td>Decision (D)</td>
<td>Decision Attribute</td>
</tr>
</tbody>
</table>

The Decision attribute, D, is the signal indicating whether we buy or sell. The decision attribute indicates the future direction of the index. The attribute was calculated using the following [2]:

\[
D = \frac{\sum_{i=1}^{n}(i + 1) \times (\text{Close}(i) - \text{Close}(0))}{\sum_{i=1}^{n} i}
\]  

The value resulting from (6) above is normalized to restrict it to a value between -1 and +1. A value of +1 indicates that every day for the next \( n \) days into the future the price closes higher than today and similarly a value of -1 indicates that every day for the next \( n \) days into the future the price closes lower than today.

**Data Discretization**

In classification problems, the collective attributes of the objects may have varying degrees of importance. To quantitatively map this variance and thus distinguish those attributes with higher and lower importance, weights are accorded to the attributes. This of course requires some auxiliary knowledge of the problem domain [2, 3, 12, 14, 15]. Since rough set theory makes no such assumption and only does computations on data included in the information table itself, it is therefore prudent to discretize continuous value attributes. It was shown in [16] that the quality of the learning
algorithm and ultimately the accuracy of the prediction model is heavily dependent on the type of discretization algorithm used.

Four discretization algorithms were tested at this stage of the model. The four methods are: EFB, Boolean Reasoning, Entropy and the Naïve Algorithm. The method selected in this regard is the equal frequency binning (EFB). The method gives the best accuracy as will be shown in Section IV below. The values of the continuous value attributes are cut into a number of segments or intervals and values lying within the same interval are mapped onto a single value. This has the advantage that the rules generated are not too specific and therefore the number of rules generated is decreased. Whilst a decrease in the number of rules is desirable, there is a tradeoff between the number of rules generated and the accuracy of the model as evidenced in Section IV below.

Redundant Attribute Elimination

The information table contains attributes that offer little or no new information about the objects. These attributes are said to be redundant [17]. These attributes can be eliminated from the information table without adversely affecting the degree of dependency between the remaining attributes and the decision [3, 5, 6, 10, 11, 17].

Reduct Generation

The minimal subset of attributes which can effectively maintain the dependency relation is called the reduct [4, 5, 7, 10, 13-15, 17]. A reduct is not unique, that is a table may have more than one reduct. However, finding all reducts is an NP-complete problem [2]. Chen and Wang [7] found that it is not necessary to find all reducts of the information system. The reducts are found using the genetic algorithm (GA). Genetic algorithms were first introduced in the 1960’s by Holland [19]. The basic genetic algorithm is thus: a population of possible solutions is maintained; the better two solutions are selected for recombination; use their offspring (combination) to replace poorer solutions [18, 19]. In general, GA’s have been applied to a large spectra of problems [19].

In this project the reduct generation was done through the rough set analysis tool, ROSETTA. In reduct generation, the end goal is that given a set of input patterns (or attributes) in $k$-dimensional space to find a transformed pattern in the $n$-dimensional space where ($n < k$) such that the optimization is met. The transformed patterns are evaluated using dimensionality as well as the class separation or classification accuracy. In ROSETTA, the GA maintains a population of competing feature matrices. To assess the accuracy of each matrix, the input patterns are multiplied by the matrix. A set of transformed patterns is produced. These are sent to the classifier which measures the hit ratio or accuracy of the transformation matrix and in representing the data set. The accuracy is fed back into the GA. Thus the GA is modeled as a parallel iterative optimizer.

There are 182 generated reducts and 1004 rules were generated. Table 2 below presents the dominance of each attribute, measured as an appearance percentage of the total number of reducts generated.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Count</th>
<th>Percentage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>54</td>
<td>29.35</td>
</tr>
<tr>
<td>High</td>
<td>73</td>
<td>39.67</td>
</tr>
<tr>
<td>Low</td>
<td>81</td>
<td>44</td>
</tr>
<tr>
<td>Close</td>
<td>71</td>
<td>38.58</td>
</tr>
<tr>
<td>Adj Close</td>
<td>54</td>
<td>29.34</td>
</tr>
<tr>
<td>MAV</td>
<td>68</td>
<td>36.95</td>
</tr>
<tr>
<td>Momentum</td>
<td>72</td>
<td>39.14</td>
</tr>
<tr>
<td>ROC</td>
<td>69</td>
<td>37.5</td>
</tr>
<tr>
<td>Disparity</td>
<td>89</td>
<td>48.37</td>
</tr>
</tbody>
</table>

Thus the highest 6 most important attributes were selected as forming the core reduct. The core reduct is:

{High, Low, Close, Momentum, ROC, Disparity}

Rule Generation.

In order to be able to provide comparison, a set of rules was generated using the 182 reducts generated in the first phase of the previous stage and then a second set of rules was generated using only the core reduct above. With 182 reducts a total of 1004 rules were generated. With the core reduct above, a total of 246 rules were constructed. A detailed account of accuracy is presented in Section IV below.

Prediction.

The prediction is done on the validation data set. The validation data set consists of 312 objects. The prediction method is per Fig 2 below.
RESULTS AND DISCUSSION

There are a number of parameters that influence the accuracy of the model [16]. Chief among these are the data split ratio, discretization algorithm and the classifier method. To construct the most robust model the correct combination of these parameters was sought. Since it was found in [16] that the quality of the model and hence its accuracy depends heavily on the discretization algorithm, a number of discretization algorithms were used so as to assess which gave the best results. These were: EFB, the Boolean Reasoning (BR) algorithm, Entropy/MDL algorithm and the Naïve algorithm. The results are tabulated below.

<table>
<thead>
<tr>
<th>Discretization Algorithm</th>
<th>Evaluation Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>EFB (with 4 Data Cuts)</td>
<td>Reducts: 182, Rules: 1004, Accuracy: 86.8 %</td>
</tr>
<tr>
<td>BR</td>
<td>Reducts: 2, Rules: 1510, Accuracy: 57.7 %</td>
</tr>
<tr>
<td>Entropy</td>
<td>Reducts: 2, Rules: 484, Accuracy: 64.5 %</td>
</tr>
<tr>
<td>Naïve Algorithm</td>
<td>Reducts: 32, Rules: 31188, Accuracy: 43 %</td>
</tr>
</tbody>
</table>

From Table 3 above it is clear that the EFB gave the best results and that all other parameter manipulations from here on would be based on the premise that the EFB is likely to give the best results. This selection of EFB as the discretization algorithm prompted the addition of an additional parameter, that of the number of data cuts. The relationship between this and accuracy is shown in Table 4 below.

<table>
<thead>
<tr>
<th>EFB Cuts</th>
<th>Evaluation Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Reducts: 171, Rules: 943, Accuracy: 66.0 %</td>
</tr>
<tr>
<td>4</td>
<td>Reducts: 182, Rules: 1004, Accuracy: 86.8 %</td>
</tr>
<tr>
<td>5</td>
<td>Reducts: 190, Rules: 1856, Accuracy: 77.3 %</td>
</tr>
<tr>
<td>6</td>
<td>Reducts: 197, Rules: 2450, Accuracy: 76.1 %</td>
</tr>
</tbody>
</table>

As shown in Table 4 above, it was found that 4 data cuts gave the best results, from which other manipulations on parameters can be based. In this regard, it was found that the Standard Voting classifier gave the best accuracy. The other classifier methods used was the Standard Voting with Object Tracking method.

Using the standard voting classifier, the difference in accuracy and rules generated from the EFB (4 data cuts) with the normal reducts and the core reducts are tabulated in Table 5 below.

<table>
<thead>
<tr>
<th>Reducts</th>
<th>Rules</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>1004</td>
<td>86.8 %</td>
</tr>
<tr>
<td>Core</td>
<td>246</td>
<td>80.4 %</td>
</tr>
</tbody>
</table>

As evidenced by Table 5 above there exists a tradeoff between the number of rules generated and the accuracy of the model. Whilst we endeavor to make the prediction model as compact as possible (i.e. with as few rules as possible), the consideration needs to be given to the accuracy of the model.

The Rough Set Confusion matrices are shown below for the core reduct based system.

<table>
<thead>
<tr>
<th>Actual</th>
<th>Predicted</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>147</td>
<td>14</td>
<td>0.9130</td>
</tr>
<tr>
<td>1</td>
<td>43</td>
<td>87</td>
<td>0.6692</td>
</tr>
<tr>
<td></td>
<td>0.7736</td>
<td>0.8613</td>
<td><strong>0.8041</strong></td>
</tr>
</tbody>
</table>

CONCLUSION

This paper has presented a rough set theory based predictive model for stock prices. The data is randomly split in a 75/25 ratio into two data sets, the training and validation data sets. To increase the efficiency of the model, the data set is
discretized using Equal Frequency Binning with 4 data cuts. Next, the reducts are extracted using the Genetic Algorithm and finally the set of dependency rules are generated from the set of reducts. A rough set confusion matrix is used to assess the accuracy of the model. The model gave a prediction accuracy of 80.4% using the Standard Voting classifier. Based on data obtained from the daily movements of the Johannesburg stock Exchange’s All Share Index the model showed a high degree of accuracy.

REFERENCES

K Kim, J Huh and I Han, “Trading rule extraction in stock market using the rough set approach”, IEEE Int. Conference on Transactions using Neural Networks, 2009, pp 175-180


Computational Intelligence Techniques for Modelling an Economic System

Msizi Khoza
Faculty of Engineering & the Built Environment
University of Johannesburg
Johannesburg, South Africa
khozams@gmail.com

Tshilidzi Marwala
Faculty of Engineering & the Built Environment
University of Johannesburg
Johannesburg, South Africa
tmarwala@uj.ac.za

Abstract—Mastery of the practice of economic modeling has long attracted the interests of economists, government bureaucrats, political theoreticians and scientists alike. In today’s global socio-political environment, economics has become an important and central feature of the determinants that shape the policies, outlook and character of modern nation states. Economics extends far beyond its traditional formulation and determines even international relations and politics. All of these underpin the importance of tools that can be used for modeling an economic system. This makes the development of an accurate model of any nation’s economy an interesting research and engineering problem. The authors of this paper present an ensemble of the results of two computational intelligence techniques in an attempt to solve this engineering problem. The techniques used are the Multi-layer perceptron (MLP) model and Rough set theory. Outputs of each method are combined to give a singular output. Rough set theory has, as its base, imperfect data analysis and approximation. The theory is used to extract a set of reducts and a set of rules based on relationships deduced from 10 attributes that influence the direction of the percentage change in the gross domestic product of the South African economy. The data used spans from 1980 to the year 2010. The MLP model developed consists of a single hidden layer and several hidden units. The optimal selection of the number of hidden layers, number of hidden units and values of weights is determined by the particle swarm optimization algorithm. The model gave a prediction accuracy of 86.8 %.

Keywords—neural networks; rough set theory; modelling; optimization.

INTRODUCTION

Around the world economics has defined the character, nature and outlook of individuals, governments and private companies. Thus the practice of economic modeling has long attracted the interests of government bureaucrats, political theoreticians and scientists alike. For a long time statistical techniques such as Bayesian models, regression and some econometric techniques have dominated research activities in prediction [1]. However the emergence of computational intelligence techniques as a viable alternative to the “traditional” statistical models that have dominated this area since the 1930’s [2,3] has given impetus to the increasing usage of these techniques in fields such as economics and finance [3,4,5]. Since the year of its inception in 1982, rough set theory has been extensively used as an effective data mining and knowledge discovery technique in numerous applications in the finance, investment and banking fields. Data mining is a discipline in computational intelligence that deals with knowledge discovery, data analysis, and full and semi-autonomous decision making [13]. It entails the analysis of data sets such that unsuspected relationships among data objects are found.

The primary approach to forecasting has been the identification of a trend and continuation of a strategy until evidence suggests that the trend has reversed [15]. One of the biggest problems with the use of regression methods is that they fail to give satisfactory forecasting results for some series’ because of their linear structure and other inherent limitations [13,15,16]. This led to the extensive use of artificial intelligence (AI) techniques particularly neural networks because their natural ability to learn and adapt to complex non-linear mappings of various statistical distributions [5,11,18,19]. Rough neuro-computing has its roots in rough set theory. Historically this paradigm consists of three main components, namely: training set description, calculus of granules and interval analysis [6, 15]. A formal treatment of the hierarchy of relations of being a part to a degree (also known as approximate rough mereology) was introduced by Polkowski and Skowron in the mid to late 1990s. Approximate rough mereology provides a basis for an agent-based, adaptive calculus of granules. This calculus serves as a guide in designing rough neuro-computing systems. The studies of neural networks in the context of rough sets and granular computing are extensive. Recent applications that make use of rough neuro-computing have included speech analysis, classification of waveforms of
power system faults, signal analysis, software quality and control of unmanned autonomous vehicles.

The rest of this paper is organized as follows: Section II discusses the theoretical foundations of rough sets, Section III provides a description of the design of the model whilst an analysis of the results and conclusions are presented in Sections IV and V respectively.

THEORETICAL FOUNDATIONS

Rough Set Theory

Rough set theory (RST) was introduced by Pawlak in 1982. The theory can be regarded as a mathematical tool used for imperfect data analysis [10]. Thus RST has proved useful in applications spanning the engineering, financial and decision support domains to mention but a few. It is based on the assumption that “with every object in the universe of discourse, some information (data or knowledge) is associated” [10]. In practical applications, the “universe of discourse” described in [10] is usually a table called the decision table in which the rows are objects or data elements and the columns are attributes and the entries are called the attribute values [3]. The objects or data elements described by the same attributes are said to be indistinguishable (indistinguishable) by the attribute set. Any set of indiscernible data elements forms a granule or atom of knowledge about the entire “universe of discourse” (information system framework) [10]. A union of these elementary sets (granules) is said to be a precise or crisp set, otherwise the set is said to be rough [1,2,7,10]. Associated with every rough set is a pair of sets called the lower and upper approximation of the rough set. The lower approximation consists of those objects which one can definitively say belong to the target set. The upper approximation consists of those objects which possibly belong to the target set. The difference between the two sets is the boundary region.

To define rough sets mathematically, we begin by defining an information system \( S = (U, A) \), where \( U \) and \( A \) are finite and non-empty sets that represent the data objects and attributes respectively. Every attribute \( a \) as a set of possible values \( V_a \), \( V_a \) is called the domain of \( a \). A subset of \( A \) say \( B \) will determine a binary relation \( I(B) \) on \( U \), which is called the indiscernibility relation. The relation is defined as follows: \((x, y) \in I(B)\) if and only if \( a(x) = a(y) \) for every \( a \) in \( B \), where \( a(x) \) denotes the value of attribute \( a \) for data object \( x \) [10]. \( I(B) \) is an equivalence relation. All equivalence classes of \( I(B) \) as \( U / I(B) \). An equivalence class of \( I(B) \) containing \( x \) is denoted as \( B(x) \). If \((x,y)\) belong to \( I(B) \) they are said to be indiscernible with respect to \( B \). All equivalence classes of the indiscernibility relation, \( I(B) \), are referred to as \( B \)-granules or \( B \)-elementary sets [10].

In the information system defined above, we define as in [10]:

\[
X \subseteq U
\]  \hspace{1cm} (1)

And,

\[
B \subseteq A
\]  \hspace{1cm} (2)

We now define the two operators assigned to every (1) two sets called the upper and lower approximation of \( X \). The two sets are defined as follows [10]:

\[
B_1(X) = \bigcup_{x \in U} \{ B(x) : B(X) \subseteq X \}
\]  \hspace{1cm} (3)

And,

\[
B_u(X) = \bigcup_{x \in U} \{ B(x) : B(X) \cap X \neq \emptyset \}
\]  \hspace{1cm} (4)

Thus, the lower approximation is the union of all \( B \)-elementary sets that are included in the target set, whilst the upper approximation is the union of all \( B \)-elementary sets that have a non-empty intersection with the target set. The difference between the two sets is called the boundary of region of \( X \). If the boundary region is an empty set then \( X \) is crisp with respect to \( B \), if however the boundary region is non-empty then \( X \) is rough with respect to \( B \). Accordingly, the set is said to be rough if it cannot be defined exactly from the available data. The set of attributes that is sufficient to represent the entire equivalence class structure is called the reduct.

Multi-layer Perceptron

The development of artificial neural networks (ANNs) constituted some of the earliest work in artificial intelligence [23]. There are fundamentally two approaches that are used in modeling, namely the universal approach and the local approach. Alternatively one can interpret this as modeling in the time domain as opposed to the feature/trajectory domain [13]. The canonical ANN is considered an instance of this paradigm. More specifically multi-layer perceptron neural networks can be regarded as a non-linear generalization of the autoregressive moving average (ARMA) [13].

The ARMA has long been used in economics and finance because it is considered a good way to model a system which is a function of unobserved and sporadic shocks. In most cases the network consists of two layers of adaptive weights and with full connectivity between inputs and hidden units and between hidden units and outputs. In general, the multi-layer perceptron model is capable of universal approximation, that is, that it can approximate to arbitrary accuracy any continuous function provided the weights and biases are chosen appropriately. In practice this means that provided that there is enough data an MLP can model any smooth function.

The simple mathematical model of the neuron was proposed by McCulloch and Pitts (1943). Generally speaking each neuron “fires” when a linear combination of its inputs exceeds a given threshold [23]. The multi-layer perceptron model (MLP) contains multiple layers of a network of these neurons. The output of the \( j^{th} \) hidden unit is a weighted linear combination of the \( d \) input values.

\[
in_{ij} = \sum_{i=0}^{d} w_{ij}^{(1)} x_i
\]  \hspace{1cm} (5)

Where \( w_{ij} \) is the weight in the 1\textsuperscript{st} layer going from input \( i \) to hidden input \( j \). The value \( w_{x0} \) will be the bias. The output of
each unit is obtained by transforming the inputs using an activation function $g$.  

$$a_j = g(in_j) = g(\sum_{i=0}^{d} w_{j,i}^{(1)} x_i) \tag{6}$$

In a two layer model the activation function $g$ is referred to as an inner function. The output $a_j$ is used as an input to the next layer. The MLP model can approximate any continuous function to a relative degree of accuracy if the number of hidden layers is large enough [16, 20]. It is this property of the MLP and its inherent ability to adapt to non-linear and complex mappings that makes it attractive to many scholars as a prediction tool [5, 10, 16]. The Universal Approximation Theorem as proposed by Haykin (1999) proved that a two-layered architecture suffices for the MLP model [20].

**Rough Neurocomputing**

The paradigm of rough neurocomputing draws its inspiration from RST. Since its introduction by Lingras in the mid-1990’s it has attracted much interest in academia. The focus of development in this area has been on the development of a training set description and inductive learning based on knowledge reduction algorithms. Another area in rough neurocomputing is the granule construction in a distributed system of intelligent agents (neural networks) [15]. Zadeh at al introduced the concept of the intuitive formulation of information granulation where intelligent agents (neurons) acquire knowledge by granulating and approximating inputs from other neurons. The adaptive calculus of granulisation has provided a means through which granules (objects which have been assembled together by virtue of indescernibility or more generally their resultant functionality) can be interpreted by neural networks. Lingras in [15] inferred that in the context of a universal information system this granulisation of data objects could be done using rough set theory. Lingras subsequently coined the term “rough neuron” [15]. Investigations into this field have led to the discovery of many subtle variations to the use of the “rough neuron” [15], namely the interval based rough neuron, the approximate neuron and the decider neuron.  

Interval based rough neurons are useful in applications the characterization of each object is based on a finite set of features, in which each feature would be defined by a upper and lower bounds (rough sets) as shown in [16, 17, 21]. Each neuron would use these values in each computation and aggregate into a single result the results obtained using both values. Peters in [25] explains that the approximate neuron measures the degree to which the approximation of the finite set of neuron inputs overlaps with the equivalence class containing measurements derived from known objects. The decider neuron implements a collection of decision rules by (i) creating an experimental conditional vector $C_{exp}$ from inputs of a rough membership function [38,39], (ii) Derive a set of decision rules with $C_i => D_i$ based on a condition vector that closely matches the experimental condition vector the most, (iii) the result is logically ANDes with the relative error term. The flow diagram of the decider neuron can be seen in Figure 1 below.

**Figure 9. Decider Neuron**

The upper input comes from an approximation neuron, as a set of measurements in response to a stimulus provided by a new object $f$ requiring classification. This is used to create $C_{exp}$. The lower input is a set of rules that have been derived from a decision table using RST [13].

**DESIGN OF MODEL**

The rough neuro-computing model was developed to model the South African economy using quarterly economic data from 1980 to 2010. The authors of this paper analyzed economic indicators and designed a model based on ten of these. These attributes or indicators are shown in Table 1 below.

**Table 1. List of Economic Indicators used as Attributes**

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construction contractors rating lack of skilled labour as constraint</td>
<td>%</td>
</tr>
<tr>
<td>Tertiary economic sector contribution to GDP</td>
<td>Millions of SA Rands</td>
</tr>
<tr>
<td>Income velocity of circulation of money</td>
<td>Rands/year</td>
</tr>
<tr>
<td>Total manufacturing production volume</td>
<td></td>
</tr>
<tr>
<td>Manufacturing firms rating lack of skilled labour as constraint</td>
<td>%</td>
</tr>
<tr>
<td>Total asset value of banking industry</td>
<td>Millions of South African Rands</td>
</tr>
<tr>
<td>Nominal unit labour cost</td>
<td>South African rands</td>
</tr>
<tr>
<td>Total mass of PGMs mined</td>
<td>Killograms</td>
</tr>
<tr>
<td>Total revenue from sale of PGMs</td>
<td>Millions of South African Rands</td>
</tr>
<tr>
<td>Gross Domestic Expenditure</td>
<td>%</td>
</tr>
</tbody>
</table>
As with most modeling problems it was inherent upon the authors of this paper to prepare the raw data for use in the model. Accordingly, the data preparation phase took shape in a four stage process that includes: Data pre-processing, data discretization and for the rough set model redundant attribute elimination, reduct generation and rule generation.

**Data Discretization**

In classification problems, the collective attributes of the objects may have varying degrees of importance. To quantitatively map this variance and thus distinguish those attributes with higher and lower importance, weights are accorded to the attributes. It is therefore prudent to discretize continuous value attributes. It was shown in [16] that the quality of the learning algorithm and ultimately the accuracy of the prediction model is heavily dependent on the type of discretization algorithm used.

Four discretization algorithms were tested at this stage of the model. The four methods are: EFB, Boolean Reasoning, Entropy and the Naive Algorithm. The method selected in this regard is the equal frequency binning (EFB) algorithm. The values of the continuous value attributes are cut into a number of segments or intervals and values lying within the same interval are mapped onto a single value. This has the advantage that the rules generated are not too specific and therefore the number of rules generated is decreased. Whilst a decrease in the number of rules is desirable, there exists a tradeoff between the number of rules generated and the accuracy of the rules produced.

**Redundant Attribute Elimination**

The information table contains attributes that offer little or no new information about the objects. These attributes are said to be redundant [17]. These attributes can be eliminated from the information table without adversely affecting the degree of dependency between the remaining attributes and the decision.

**Reduct Generation**

The minimal subset of attributes which can effectively maintain the dependency relation is called the reduct [4, 5, 7, 10, 13-15, 17]. A reduct is not unique, that is a table may have more than one reduct. However, finding all reducts is an NP-complete problem [2]. The reducts are found using the genetic algorithm (GA). Genetic algorithms were first introduced in the 1960’s by Holland [19].

The basic genetic algorithm is thus: a population of possible solutions is maintained; the better two solutions are selected for recombination; use their offspring (combination) to replace poorer solutions [18, 19]. In this project the reduct generation was done through the rough set analysis tool, ROSETTA. In reduct generation, the end goal is that given a set or input patterns (or attributes) in k-dimensional space to find a transformed pattern in the n-dimensional space where (n < k) such that the optimization is met. The transformed patterns are evaluated using dimensionality as well as the class separation or classification accuracy. In ROSETTA, the GA maintains a population of competing feature matrices. To assess the accuracy of each matrix, the input patterns are multiplied by the matrix. A set of transformed patterns is produced. These are sent to the classifier which measures the hit ratio or accuracy of the transformation matrix and in representing the data set. The accuracy is fed back into the GA. Thus the GA can be seen as a parallel iterative optimizer.

The neural network component of this design was implemented with the MATLAB neural networking toolbox, Netlab. The rough neuron is a pair \((r_l, r_u)\) where \(r_l\) is the lower neuron and \(r_u\) the upper neuron. In effect, the rough neuron stores the upper and lower bounds of the input value of a an attribute and uses it in its computation. As with (6) above, the output of the rough upper neuron and rough lower neuron will be computed as in (7) and (8) respectively.

\[
Out_{r_l} = \min \left( f(in_{r_l}), f(in_{r_u}) \right)
\]

\[
Out_{r_u} = \max \left( f(in_{r_l}), f(in_{r_u}) \right)
\]

The function, \(f()\), is the activation function and in this project we have used the sigmoid function. That is to say, the function, \(f()\), takes the form:

\[
f(x) = \frac{a}{1 + e^{px+y}}
\]

**RESULTS AND DISCUSSION**

There are a number of parameters that influence the accuracy of the model [16]. From the rough set theory side, the among these are the data split ratio, discretization algorithm and the classifier method. To construct the most robust model the correct combination of these parameters was sought. Since it was found in [16] that the quality of the model and hence its accuracy depends highly on the discretization algorithm. Based on previous work [our reference] we have done in this field, we chose the equal frequency binning algorithm. We also found that the standard voting classifier gave the best accuracy. The confusion matrix presented as Table 2 below shows the accuracy of the rough set model.

<table>
<thead>
<tr>
<th>Table 2. Rough Set Model Confusion Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted</td>
</tr>
<tr>
<td>Actual</td>
</tr>
</tbody>
</table>
The discussed rough set based neural model was developed to increase the efficiency of the conventional neural network model. The optimal learning constant ($\alpha$), the number of hidden units, the number of hidden layers and the values of weights were determined by the particle swarm optimization method. Table 3 below shows the classification accuracy of the rough neural network for different rough set discretization algorithms used with the number of training epochs remaining the same.

<table>
<thead>
<tr>
<th>Discretization Algorithm</th>
<th>Evaluation Criteria</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>EFB (with 4 Data Cuts)</td>
<td>Reducts 182</td>
<td>Epochs 100</td>
</tr>
<tr>
<td>BR</td>
<td>2</td>
<td>100</td>
</tr>
<tr>
<td>Entropy</td>
<td>2</td>
<td>100</td>
</tr>
<tr>
<td>Naïve Algorithm</td>
<td>32</td>
<td>100</td>
</tr>
</tbody>
</table>

As shown by Table 3 above, the data was discretized using the EFB algorithm as it showed to have the greatest positive effect on the accuracy of the model.

CONCLUSION

This paper has presented a generic rough neuro-computing model that can be used to predict the direction in which the GDP of the South African economy will turn dependant on ten (10) parameters. The data was discretized using the EFB algorithm and split in a 75/25 ratio for training and validation respectively. The optimal number of hidden rough neurons and layers was determined through the employ of the particle swarm optimization method. The integrated model gave a prediction accuracy of 86.8%. We have reason to believe that this area can be further researched in order to provide greater value to governments and economic policy analysts and researchers.

REFERENCES

K Kim, J Huh and I Han, “Trading rule extraction in stock market using the rough set approach”, IEEE Int. Conference on Transactions using Nural Networks, 2009, pp 175-180

M Khoza, T Marwala, “A rought set theory based predictive model for stock prices”, 12th IEEE International Conference on Computational Intelligence and Informatics, Nov. 2011
Hongsheng Su, Quanzhi Li: Fuzzy Neural Classifier for Fault Diagnosis of Transformer Based on Rough Sets Theory: IEEE, CS, pp. 2223-2227.
Jiang-Hong Man: An improved fuzzy discretization way for decision, Conference on Machine Learning and Cybernetics, Hong Kong, August 2007.