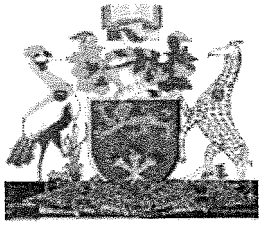
a)



UNIVERSITY OF NAIROBI

SCHOOL OF COMPUTING AND INFORMATICS (SCI)

APPLICATION OF ARTIFICIAL NEURAL NETWORKS IN TIME SERIES FORECASTING

- A CASE STUDY OF STUDENTS ENROLMENT IN A COLLEGE

by

Kandiri, John *Mugo(P56IPI783412000)*

SUPERVISORS

MR. P. Waiganjo Wagacha

Mr. Eric Ayienga

A Project submitted to the School of Computing and Informatics (SCI) of

The University of Nairobi

in partial fulfillment of the requirements
for the degree

MASTER OF SCIENCE

In

INFORMATION SYSTEMS (MSc. IS.)

@September, 2003

DECLARATION

This project, as presented in this report, is my original work and has not been presented/or any other University award.

Kandiri, John Mugo *(P56IPI783412000)* ..... *i.Date* 30.09.2003

Signed by (Supervisors)

Supervisor 1. *Mr. Waiganjo Wagacha* ..... *Date* .....

Supervisor 2: *Mr. Eric Ayienga* . ..... *Date* 3/11/2003

School of Computing & I~
University of NAIROBI
P. O. Box 30197
NAIROBI

DEDICATION

To my wife Mary for her encouragement and support and to our son, you are a source of encouragement - You gave the family a new meaning to life.

ACKNOWLEDGEMENT

This project completion would not have been succesful were it not for the contribution from a number of persons. These are people who provided important guidance, comments and motivation in course of the study.

First, to my ***Heavenly Father*** for the good health and provision of resources - money, time and people, that necessitated my taking this course and seeing me to completion.

I am particularly grateful to my supervisors ***Mr. Peter*** *Waiganjo Wagacha* and ***Mr. Eric*** *Ayienga* who not only offered comments as part of supervision but also offered moral support. Their guidelines during the specifications and drafting of the report cannot go unmentioned.

To ***Mr.*** *Owino,* my MSc. Colleague, your indebt knowledge and love for programming deserve acknowledgement and appreciation. Without your sacrifice, C++ programming and specifically object oriented implementation would not have come out this clear!

To ***Mr. Abade*** (undergraduate student). Your willingness to perform system testing is well appreciated.

To ***Mr.*** *Misati and* ***Mr. Juma*** (colleagues at Vision Institute and lecturers in *Statistics)* for proof reading the chapter 1.

To the Board of Directors - ***Vision Institute of Professionals,*** I greatly appreciate your acceptance to my request to use college data for this research. This was great trust on me from you!

More so, many thanks to my colleagues ***Cyprian*** and ***Violet*** whose moral support boosted my moral.

Glossary

ANN: Abbreviation for Artificial Neural Network. Is a computerized system which mimics human learning (Please see the Literature Review).

Activation function: In neural networks, an activation function is the function that describes the output behaviour of a neuron.

Epoch: In training a neural net, an epoch refers to a complete pass through all of the training patterns. The weights in the neural net may be updated after each pattern is presented to the net, or they may be updated just once at the end ofthe epoch. Frequently used as a measure of speed of learning - for example "training was complete after n epochs".

Hidden layer: Is the layer(s) between the input and output layers. The input layer is composed of the input units while the output layer is composed ofthe output units.

Learning Rate: It is a value that denotes the effect of errors on weight adjustment during neural network learning. It influences the network stability and convergence rate, that is, the time or number of recursions it will take before the computed error value goes below a set minimum error known as threshold. Coming up with this value is more of experimental as shown in tables 1 and 2.

Minimum Error: Also called acceptance error. It is a value that is set during neural network training and help to determine when training should stop. When the computed error goes below this value, the neural network weight as taken to be the "best" weights that can be used in neural network application.

Overfitting in Neural Network: This is also called 'overtraining' or 'overlearning'. It is the phenomenon that in most cases a network gets worse instead of better after a certain point during
training when it is trained to as low errors as possible. This is because such long training may make the network 'memorize' the training patterns, including all of their peculiarities

Supervised learning algorithm: A back propagation network uses a supervised learning algorithm when an input pattern is presented to the network and then an output pattern is computed. This output pattern is compared to a target output pattern resulting in an error value. The error value is propagated backwards through

the network, (the network inherits its name from this methodology) and the values of the connections between the layers of units are adjusted in a way that the next time the output pattern is computed, it will be more similar to the target output pattern. This process is repeated until output pattern and target output pattern are (almost) equal.

Training pattern: This term is used to refer to a set of input values and the corresponding set of desired or target output values for use in training an ANN.

Unsupervised learning: Unsupervised learning signifies a mode of machine learning where the system is not told the "right answer" - for example, it is not trained on pairs consisting of an input and the desired output but if left to get the right output.

ABSTRACT

Estimation and prediction of students enrolment in a college provides, besides straightforward profit opportunities, indications to various important data and information for example information on customer fluctuation over period, thus helping in decision making especially in resource allocation. Forecasting the number of students expected to enroll in a college is therefore of prime importance to the management for both tactical and strategic management. This can be based on the available historical data and use of time series prediction.

Students' enrolment time series can be predicted with a certain degree of confidence. This is from analyzed data. Future activities can be determined from the past performances. In this project, a short forecast will be used. This is because a shorter forecast gives a more accurate result with a higher degree of confidence.

So far, the primary means of detecting trends and patterns has involved statistical methods such as statistical clustering and regression analysis. The mathematical models associated with these methods for economical forecasting, however, are linear and may fail to forecast the turning points in economic cycles because in many cases the data they model maybe highly nonlinear.

In the contemporary generation in computing, new methodologies, including neural networks, knowledge-based systems and genetic algorithms, has attracted attention for analysis of trends and patterns. In particular, neural networks are being used extensively for financial forecasting with stockmarkets, foreign exchange trading, and commodity future trading and bond yields.

Stockmarket prediction is an area of financial forecasting which attracts a great deal of attention

This research paper therefore presents a scheme for time series forecasting with a neural network.

To help evaluate the performance of the Neural Net, a benchmark Autoregression model will be developed using regression analysis. A statistical package SPSS will be used to come up with the model. A theoretical comparison of the methods (ANN and autoRegression) is provided in the conclusion.

LIST OF TABLES

LIST **OF** FIGURES

TABLE OF CONTENTS

1.0 INTRODUCTION

1.1 Prediction and Training Colleges

Prediction/Forecasting is necessary to every decision or policy position. Essentially, the policy or decision one advocates assumes forecasts of what will happen as a result of the action being recommended. Consequently forecasting as an activity permeates all aspects of good management and policy-making practices.

Many method of forecasting can be used. Some forecasts are purely based on intuition and human judgment, others require complex mathematical (Wonnacott 1994[14]) and computer based models (Kingdon 1997[8]) . Some forecasts rely heavily on social and physical science theory while others do not.

Computer training colleges deals with resource allocation (either lecturers or computer) in day-today life. The importance of prediction in a college is therefore a fundamental activity as this can make college get a competitive activity over its competitors.

This research is based Vision Institute of Professionals. This is a commercial training college that trains in computer packages among other courses.

Below is a brief description of Vision Institute of Professionals (VIP). *1.2 A*

brief history of VIP

VIP is a training institution that offers training in information technology (IT) and accountancy, The college was established in 1997 and started training in accountancy, later training diversified to IT training. During its expansion, VIP has realized a growth in the student population from less than a hundred in 1997 to more than 2500 now.

In the IT department, the college offers three diploma courses and several computer package certificates (computer literacy classes). The diploma courses enrolment takes place semiannually, that is, classes start every January and July. Enrolment therefore takes place before the classes commence. For the computer packages certificate courses, classes commence every

Monday. Enrolment takes place either before or on the material day the student is to begin his/her class (Monday).

As noted above, the computer package training is Sh011 term and therefore prior planning for the resources is required. The college has of late embarked on expanding its computer lab to cater for the computer literacy classes (packages classes).

To cater for this growing student population, the management needs to have the right number of resources, which include computers and human personnel. This requires forecasting as the college will be making a big mistake to arrange for the resources when the students are already there.
This therefore calls for a reliable prediction method. Currently, the management uses intuition and human judgment to predict students' enrollment, This method is subjective and error prone. Alternative methods that can be applied include use of statistical methods (Wonnacott 1994(14]) and computer based models. Computer based models can be developed by use of machine learning techniques such as artificial neural networks (MITCHELL 1997[10]).

This research is based on application of artificial neural networks (ANN) in forecasting student's enrolment. It therefore uses **VIP** one of the computer packages training institute as the case study. The performance of the neural network model developed is compared with a regression model developed for the purpose of validation. The neural network can be used in other training
institutions for predicting purposes.

1.3 Research Problem

Estimation and prediction of students enrolment in a college provides, besides straightforward profit opportunities, indications to various important data and information for example information on customer fluctuation over period, thus helping in decision making especially in resource allocation. Forecasting the number of students expected to enroll in a college is therefore of prime importance to the management for both tactical and strategic management. This can be based on the available historical data and use of time series prediction.

So far, the primary means of detecting trends and patterns whish can help in prediction has involved statistical methods such as statistical clustering and regression analysis. Also heuristic method where managers use heuristics to predict future enrolment. The mathematical models associated with these methods for economical forecasting, however, are linear and may fail to forecast the turning points in economic cycles because in many cases the data they model may be highly nonlinear. The heuristic methods are also more of probabilistic and based on the managers' experience.

In the contemporary generation in computing, new methodologies, including neural networks, knowledge-based systems and genetic algorithms, has attracted attention for analysis of trends and patterns. In particular, neural networks are being used extensively for financial forecasting with stockmarkets, foreign exchange trading, and commodity future trading and bond yields. Stockmarket prediction is an area of financial forecasting which attracts a great deal of attention

This research proposal therefore uses this backdrop to demonstrate the application of ANN in decision making process. ANN is used here as a management tool and a decision support (forecasting) mechanism, which a training institution can use to make it more competitive. ANN here is demonstrated as a way of improving (and replacing) domain experts' decision-making as well as a means of improving the efficiency and consistency of the decision process. This can be seen as a significant step beyond the type of automation that has been experienced 111 manufacturing, as been pointed out for intelligent systems in finance (Kingdon 1997[8])

The system will therefore facilitate in the improvement in the prediction of students enrolment at different times of the year.

The project after completion can be used as reference by other researchers in the area of AI and ANN. It can also be used as a prototype by other training institutions-colleges and universities in the same class and affected by similar environmental conditions. What will be required is just customization to meet the specific user requirements.

1.4 Hypothesis

The existing prediction method based on human knowledge and statistical methods is inadequate in meeting the training colleges' resource management.

1.5 Aims & Objectives

The project aims at coming up with a system for forecasting the number of students to enroll in a college at a given time. The project also aims at utilizing time series as a benchmark to predict the number of students to enroll at any given time.

The objectives of the project are:

d) To explain the principals of using multilayer perceptrons, and to show the effects of the different parameters on a neural network.
e) To demonstrate the suitability of ANN as a forecasting tool by its ability to learn and generalise;
c) To evaluate the performance of ANN as a forecasting tool visa vis with the conventional (statistical) methods.
d) To come up with an ANN prototype that can be used for forecasting;

One goal of the research was to determine the application of artificial neural network as a functional approximator in time series forecasting

1.6 Scope and Limitations of the study

The study will cover forecasting the number of students at a given time. System will use data produced after students enroll and will not cover the enrollment module.

1. 7 ***Chapters Summary***

Chapter 2 introduces time series forecasting, goals of time series forecasting and areas of application of time series forecasting which justified this research. Several difficulties associated with time series forecasting are be identified

Later in the chapter, Machine Learning techniques are discussed. The chapter then zeros on ***Artificial neural Networks (ANN).*** In trying to understand neural networks, the chapter gives a comparison between biological neuron and the artificial neuron. Also presented is the ANN learning and feed-forward neural networks and backpropagation training, which was used as the primary time series forecasting technique in this research work. The decisions to adopt ANN as the machine learning method and also to develop a test bed neural network is explained in this chapter. The chapter concludes by looking at related works on machine learning and specifically ANN application.

Chapter 3 looks at the project methodology. In its introduction, a developed neural net *CEFS* ***College Enrolment Forecasting System*** is introduced. The project methodology is detailed.

In chapter 4, the empirical results are given. Here the way the network architecture was arrived at is shown. Also the learning rate and momentum factor chosen. In the same chapter, an evaluation of the neural network is done against a benchmark Autoregression model.

Chapter 5 presents a discussion of the results.

Finally, a conclusion is given from the empirical result.

2.0 LITERATURE REVIEW

2.10 Time Series Forecasting

Predicting future events is called *forecasting:* an attempt to foresee events by examining the past. Forecasting techniques may be based on:

a) Experiences;
b) Judgments;
c) Opinions of "experts";
d) Or on mathematical models-

Almost all forecasts are based on the assumption that the item to be forecast is affected by another variable (e.g., time) or by several variables (such as those that measure the overall economic conditions of a region). One of the mathematical models is based on time series data.

A time series is a sequence of measurements made at regular times, for example, daily students enrolment in a college, daily temperature like in the morning, and number of births in a given month of the year among others.

Time Series Analysis (TSA) is a method that is used for estimating the process that underlies some output, or for forecasting from some observed behaviour over time. Time series forecasting, or time series prediction, takes an existing series of data and forecasts the data values.

If the set is continuous, the time series is said to be continuous, and if the set is discrete, the time series is said to be discrete. The observations from a discrete time series, made at some fixed interval h, at times $\tau_1, \tau_2, \ldots, \tau_N$ may be denoted by $Z(\tau_1), Z(\tau_2), \ldots, Z(\tau_N)$

By sampling a continuous time series;
b) By accumulating a variable over a period of time

2.11 Goal of Time series

There are two main goals of time series analysis:

a) Identifying the nature of the phenomenon represented by the sequence of observations, and;
b) Forecasting (predicting future values of the time series variable).

Both of these goals require that the pattern of observed time series data is identified and more or less formally described. Once the pattern is established, we can interpret and integrate it with other data (i.e., use it in our theory of the investigated phenomenon, e.g., seasonal commodity prices

Most time series forecasting methods assume that a time series has the following characteristics:

a) The time periods at which each measurement, or data value, is recorded are of equal length;
b) The data values are always arranged in order and there is a data value for each time period; i.e., there are no missing values;
c) The process or activity and the method of measuring remain consistent over time.

Models for time series and dynamic systems can be used in three important areas of application:

a) Forecasting of future values of a time series from current and past values;
b) The determination of a transfer function of a system: the determination of a dynamic input output model that can show the effect on the output of a system subject to inertia, of any given series of inputs;
c) The design of simple feed-forward and feedback control schemes by means of which potential deviations of the system output from a desired target may be compensated, so far as possible.

Examples of data series include financial data series (stocks, indices, rates, etc.), physically observed data series (weather, flood etc.), and mathematical data series (integrals of differential

equations, etc.). The phrase "time series" generically refers to any data series, whether or not the data are dependent on a certain time increment.

Characteristics of time series include:

a) Time periods are of equal length;
b) No missing values.

2.12 Areas of Time Series application

a) Forecasting;
b) Determination of a transfer factions of a system;
c) Design of simple feed-forward and feedback control schemes.

In Forecasting, applications include:

a) Economic and business planning;
b) Inventory and production control;
c) Control and optimization of industrial processes.

2.13 Time series patterns

Most time series exhibit some consistent patterns of behavior. To forecast a time series, then, we must be able to identify what these patterns are and extend them into the future. This is the basic philosophy behind all self-projecting forecasting methods. Some of the more common patterns encountered are:

a) The overall trend pattern is the overall movement or direction in the time series from. Beginning to end. While the trend will be simply rising, falling, or horizontal, it is possible to consider more complex overall trends;
b) The seasonal pattern is the repetition of the same basic pattern at regular intervals. The same interval is called the season, and the time periods within the season are called the periods per season;
c) A cyclic pattern is a more or less consistently rising and falling pattern over an extended period of time, however, the distance between peaks and valleys are generally not equal;

d) A statistical pattern is a consistent relationship among time series values (e.g., an increase in value from one period to the next may be followed by a decrease in value in the next period). Recognition of statistical patterns forms the basis of the Box-Jenkins forecasting method. *The ARlMA model is discussed below under AutoRegression.*

The first three patterns are visual patterns that can generally be recognized when a time series is displayed in graphical form. Some time series may contain only one of these patterns, while others may contain all of them combined. Most traditional self-projecting forecasting methods are based on the attempt to generate these types of patterns or combinations of patterns. It is possible, however, that patterns that are not visually perceptible might exist in time series data. Without the proper tools it would be very difficult to identify such patterns. Such patterns are often called statistical patterns, since it is with statistical tools that they can be identified.

Once we have identified the patterns in a time series, we extend these patterns into the future by developing a mathematical model. Intuitively, we know that no mathematical formula will exactly reproduce each and every value in a time series and there will always be a discrepancy (error or residual) between a formula-generated value and the actual time series value at a given period even if the process we are interested in forecasting could be precisely described by some mathematical formula. One or more factors may contribute to this error, and among them, is the inexact nature of the measuring device. Therefore, any time series is assumed to have two major components: a pattern component and a non pattern, or random error, component; i.e.

Time series = patterns + random errors

$Z_t = F, + a_t$ 1.11

By its very nature the random error component cannot be forecasted, and the best we can do is provide some estimate of how big it might be in future time series values. Therefore, more accurate forecasts would be achieved with the greater the presence of patterns in a series.

In general, the process of building a time series model involves the following steps:

a) Identify the correct model form;
b) Estimate the values for the parameters in the model so that the generated F, are as close as possible to the original z_t, i.e., so that *at* is minimized in some sense. If the model form is the correct one, the residuals should not exhibit any patterns, since they are supposed to be urn-elated random errors. Depending on the model form, one of a number of known mathematical procedures can be used to estimate the parameter values;
c) Forecast the time series and provide a measure of the uncertainty in these forecasts;
d) Monitor the accuracy of the forecasts when new, actual time series values are obtained. Forecasts that are off the mark more than expected may suggest a change in the historical patterns (and therefore the model form) or a change in the model parameter values, which must therefore be re-estimated. In this sense building a forecast model is a dynamic process.

The diagrams below illustrate how time series can be used in prediction. In the first figure *(figure* 1), the yearly data displays a seasonal pattern so strong that we discern its main features at a glance. Expenditure at every 4th quarter of the year tends to be high followed by a big drop next quarter. Seasonal variation might occur due to:

a) A holiday e.g. Christmas;

b) Seasons may affect economic activity.

If Y_t represents sales and Ψ_t least squares fit

Q], Q2, Q3 Q4 represents 4 quarters in an year,

Then Ψ = trend + seasonal pattern + residual

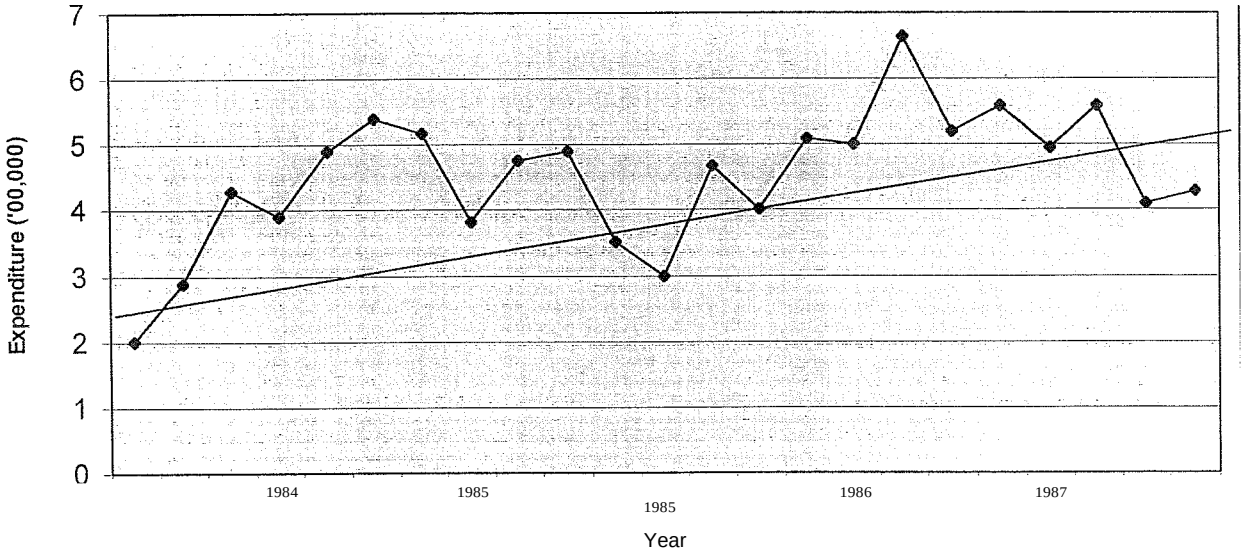


***FIGURE* 1 :** *Seasonal variation in a time series. (Adaptedfrom WONNACOTT* 1994 *[14J)*

Figure 2 below exemplifies serial correlation in Time series. When the Previous observations are low, the next tends to be low. Therefore in the above figure, the line is expected to go to B not A

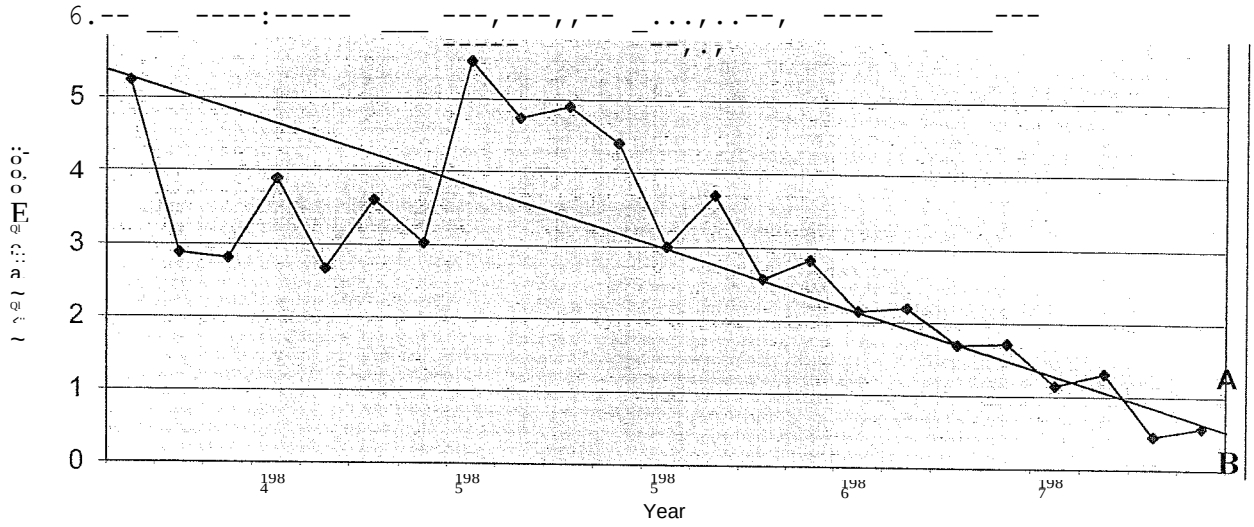


***FIGURE* 2.** *Serial correlation in Time series (Adaptedfrom WONNACOTT* 1994 [14])

2.14 Brief Notes On Auto regression

Regression is the study of relationships among variables, a principal purpose of which is to predict, or estimate the value of one variable from known or assumed values of other variables related to it.

Predictors (Variable): To make predictions or estimates we must identify the effective predictors or the variables of interest: which variables are important indicators and can be measured at the least cost, which carry only a little information, and which are redundant. **Predicting:** Predicting a change over time or extrapolating from present conditions to future conditions is not the function of regression analysis. To make estimates of the future, use time series analysis.

Experiment: Begin with a hypothesis about how several variables might be related to another variable and the form of the relationship.

Types of Analysis

- Simple Linear Regression: A regression using only one predictor IS called a simple regression,
- Multiple Regression: Where there are two or more predictors, multiple regression analysis is employed.

Data: Since it is usually unrealistic to obtain information on an entire population, a sample which is a subset of the population is usually selected. The sample may be either randomly selected for a researcher may chose the x-values based on the capability of the equipment utilized in the experiment or the experiment design. Where the x-values are preselected, usually only limited inferences can be drawn depending upon the particular values chosen. When both x and yare randomly drawn, inferences can generally be drawn over the range of values in the sample.

The Model: If we have determined there is a linear relationship between t and y we want a linear equation stating y as a function of x in the form $Y = a + bt + e$ where a is the intercept, b is the slope and e is the error term accounting for variables that affect y but are not included as predictors, and/or otherwise unpredictable and uncontrollable factors.

Methods:

Exponential smoothing

Seasonal decomposition

Auto regression

Auto regression

Auto regression analysis has been often used in time series forecasting using statistical approaches such as ARMA models. This analysis is mainly used in detecting the autocorrelations between successive observations of time series, and used in the well-known ARIMA models with Box-Jenkins methods that are very efficient in forecasting linear time series

The standard modeling and forecasting procedures used in identifying patterns in time series data involve knowledge about the mathematical model of the process. However, in real-life research

and practice, patterns of the data are unclear, individual observations involve considerable error, and we still need not only to uncover the hidden patterns in the data but also generate forecasts.

The auto regression procedure estimates a linear regression model with first order autoregressive errors (i.e. uses the simple regression model for forecasting time series models)

It provides for independent and multiple dependent variables. Therefore it is preferred to the exponential smoothing which requires identification of the correct exponential smoothing models (with the attendant limitations for each, i.e. simple model, Holt, Winters, and Custom models).

Methods available for Autoregression:

a) *Exact maximum likelihood;*
b) *Cochrane-Orcutt;*
c) *Prais-Winsten model;*
d) *ARIMA.*

Exact maximum likelihood

This technique finds those parameter estimates that are most likely to have produced the observed data. The method can handle missing data within the series. It can also be used when one of the independent variables is the lagged dependent variable.

Cochrane-Orcutt

This is a simple and widely used estimation procedure for estimating a regression equation whose errors follow a first order autoregressive process.

It has a weakness in that it cannot be used when a series contains imbedded missing values.

ARIMA

The ARIMA methodology developed by Box and Jenkins has gained enormous popularity 111 many areas and research practice confirms its power and flexibility. However, because of its power and flexibility, ARIMA is a complex technique; it is not easy to use, it requires a great deal of experience, and although it often produces satisfactory results, those results depend on the researcher's level of expertise, This calls for a simpler but reliable forecasting method especially for this study.

Prais- Winsten ***model***

This method uses a generalised least squares method for estimating a regression equation whose errors follow a first order autoregressive process. It cannot be used when a series contains imbedded missing values.

Generally, this method is preferable to the Cochran-Orcutt method.

As a result, it has been used to fit the predictive equation for the time series data for College Enrolment Forecasts.

It is good to note that all forecasting models have either an implicit or explicit error structure, where error is defined as the difference etween the model prediction and the "true" value. Additionally, many data snooping methodologies within the field of statistics need to be applied to data supplied to a forecasting model. Also, diagnostic checking, as defined within the field of statistics, is required for any model which uses data.

Difficulties inherent in time series forecasting and the importance of time series forecasting are presented next.

2.15 Difficulties Inherent in Time Series Forecasting

Difficulties of Time Series Prediction include:

a) Limited quantity of data;
b) Noise;
c) No stationary data;
d) Forecasting technique selection

Several difficulties can arise when performing time series forecasting. Depending on the type of data series, a particular difficulty mayor may not exist. A first difficulty is a *limited quantity of data.* With data series that are observed, limited data may be the foremost difficulty. For example, an institution's data reflecting students' enrolment in an year may not be adequate and may be very limited data for use in neural network forecasting.

A second difficulty is *noise.* Two types of noisy data are (1) erroneous data points and (2) components that obscure the underlying form of the data series. Two examples of erroneous data are measurement errors and a change in measurement methods or metrics. In this paper, we will not be concerned about erroneous data points. An example of a component that obscures the underlying form of the data series is an additive high-frequency component. The technique used in this paper to reduce or remove this type of noise is the moving average. The data series becomes after taking a moving average with an interval i . Taking a moving average reduces the number of data points in the series.

A third difficulty is *nonstationarity,* data that do not have the same statistical properties (e.g., mean and variance) at each point in time. A simple example of a nonstationary series is the Fibonacci sequence: at every step the sequence takes on a new, higher mean value. The technique used in this paper to make a series stationary in the mean is first-differencing. The data series becomes after taking the first-difference. This usually makes a data series stationary in the mean. If not, the second-difference of the series can be taken. Taking the first-difference reduces the number of data points in the series by one.

A fourth difficulty is *forecasting technique selection.* From statistics to artificial intelligence, there are a variety of techniques that have been employed. One of the simplest techniques is to search a data series for similar past events and use the matches to make a forecast. One of the most complex techniques is to train a model on the series and use the model to make a forecast. Neural networks are examples of the second technique.

2.16 Importance of Time Series Forecasting

Time series forecasting has several important applications. One application is preventing undesirable events by forecasting the event, identifying the circumstances preceding the event, and taking corrective action so the event can be avoided.

Another application is forecasting undesirable, yet unavoidable, events to preemptively lessen their impact. Finally, many people, primarily in the financial markets, would like to profit from time series forecasting. Whether this is viable is most likely a never-to-be-resolved question. During this research, a lot of literature was available on application of ANN in financial forecasting.

Forecasting has therefore been a broad application domain of AI and particularly ANN. Several studies have been made in financial forecasting (Kingdon 1997[8]) , floods and River flow Forecasting. Forecasting can therefore be seen as a managerial responsibility where a good prediction is a desirable quality in business operations.

Forecasting is not an easy task as it is highly "noisy" application because a system receives thousands of inputs, which interact in a complex nonlinear fashion *[see figures* 2 *and* 3] with the inputs that are fed to the system, the system behavior is estimated and extrapolated into the future. But, while performing predictions, some correlation is noted in successive events or inputs that affect the time series *[Figures* 2 *and* 3**].** This time series pattern can therefore give a hint of the future.

2.20 Artificial Intelligence and Machine Learning

Neural computing is a branch of Artificial Intelligence (AI) or machine learning where computers are made to mimic human brain. Before we look at Neural computing therefore, a brief overview of AI and machine learning will suffice.

Artificial Intelligence (AI) is a term that refers to the ability of an artifact to perform the same kinds of functions that characterize human thought. AI can also be defined as the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable. It is good to note that this intelligence is artificial as any machine intelligence comes from humans. This is unlike natural intelligence found in animals.

The possibility of developing some such artifact has intrigued human beings since ancient times. In modern applications, AI programs are meant to perform more complex that straight forward programming for business systems. AI has therefore been used in pattern recognition, computer games, medical diagnosis (this can be in Radiology where a radiologist is able to isolate for example a cancerous cell from a non cancerous cell in human beings). AI has also been used in manufacturing and control of other devices. Another application is in prediction. This research paper looks at the application of AI (specifically ANN) in prediction.

Artificial Intelligent (AI) has been applied in real world problem solving for sometime now. This is especially in manufacturing (Robotics) and forecasting. There is thus a rising ambition to produce more intelligent machines, which are gaining popularity in scientific research, business, finance and industry. AI techniques such as Neural Networks, Genetic Algorithms and Fuzzy Logic have been applied in Financial Forecasting, River flow Forecasting, Credit Rating and Medicine (Radiology)

Machine Learning

Mitchell [10] defines machine learning as;

> a computer program is said to learn from experience with respect to some class of tasks and performance <P> if its performance at task <T> as measured by <P>, improves with experience <E>.

Like human beings, machine learning involves acquiring general concepts from specific training examples. The concepts can be a Boolean valued function defined in a larger set. This concept is learning, which can be seen as in family with a Boolean valued function from the examples of its inputs and outputs.

Mitchell gives the following scenario to explain machine learning:

> A target concept is sought e.g. when one enjoys cycling.
> Here, you have to give the attributes that describe a day e.g. days weather (dry or wet), wind (strong or not), among others.
> You specify a single required value for each attribute.
> If some instances X satisfies all the constraints of hypothesis by then h classifies X as a positive example h(x)=1
> Instances the set of items over which the concept is defined. I can for example denote set of all the target concept and the concept to be learned.

To realise the full potential of AI, researchers have developed many machine learning techniques. The diagram below shows the various machine learning techniques.

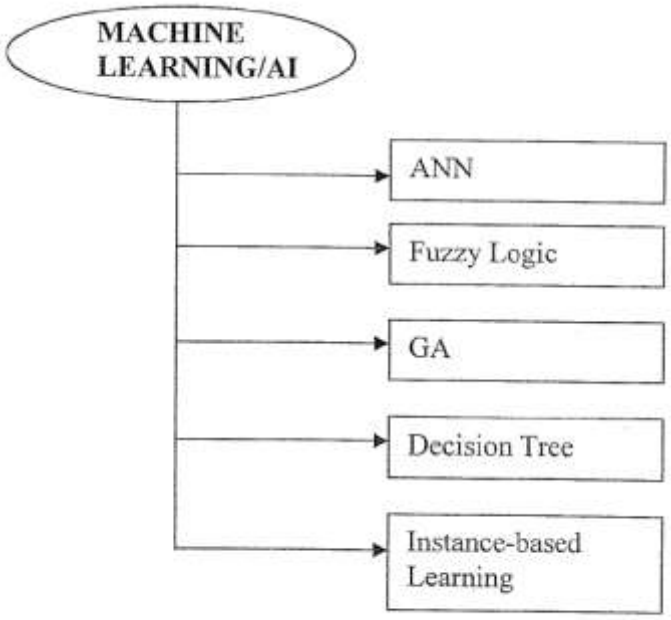


Figure 3: *Machine Learning Methods*

The above machine learning techniques have been implemented as forecasting techniques. This has been evident in the available literature review. This research paper will focus on the use of Neural Networks (ANN).The research therefore attempts to fill a gap in the abundant neural network time series forecasting literature, where testing arbitrary neural networks on arbitrarily complex data series is common, but not very enlightening. This paper thoroughly analyzes the responses of specific neural network configurations to a real data series collected from a training college. Each data series has a specific characteristic. Neural network-based learning was adopted because from the available literature, NN has been applied effectively for forecasting. NN have also been noted to generalize well in cases that have previously not been encountered. Also, because the inputs and outputs are numerically coded and neuron-like, they are able to manage multivalued input attributes. The literature provides an insight on what is ANN and compares this area of machine learning to natural learning techniques. The literature starts by

defining what ANN is, this is followed by a discussing the biological (natural) neuron, then artificial neuron before giving a comparison between the two neurons.

2.21 Artificial Neural Networks

Neural computing is an approach that attempts to mimic the manner in which our brain works. It applies machine learning where machine can learn from experience just like human beings. This is use of historical case.

By mimicking human learning, an artificial Neural Network (ANN) is able to perform parallel processing, have fast retrieval of large amounts of data/information and have the ability to recognize pattern based on historical data. Given a training set of data, the ANN can learn the data with a learning algorithm. In this project, the most common algorithm, back propagation will be used, using back propagation the ANN forms a mapping between inputs and desired output from the training set by altering weighted connection with the network.

ANN has been escalated by need for brain-like information processing, advances in computer technology and progress in neuroscience (Turban 1998 [13]) - *Neur-oscience is the study of the human nervous system, the brain, and the biological basis of consciousness, perception, memor-y,*

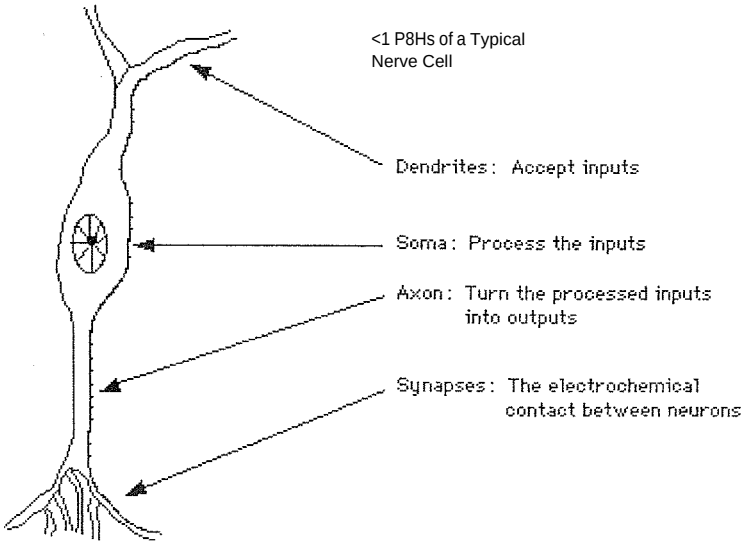


Figure 4: *The Biological Neuron, (WebSite [4J)*

and learning. ANNs are therefore an attempt at modeling the information processing capabilities of a nervous system. The paragraph that follows briefly discusses the biological neuron.

The most basic element of the human brain is a specific type of cell, which provides us with the abilities to remember, think, and apply previous experiences to our every action. These cells are known as neurons, each of these neurons can connect with up to 200000 other neurons. The power of the brain comes from the numbers of these basic components and the multiple connections between them.

Neurons are linked to each other through weighted communication channels. While a neuron does very little on its own, when one neuron is combined with hundreds, thousands or millions of other neurons it can solve very complex problems. The human brain is composed of billions of such neurons

2.21.a The Neuron

Although it has been proposed that there is anything between 50 and 500 different types of neurons in our brain, they are mostly just specialized cells based upon the basic neuron.
It is estimated that the number of neurons in the human brain are in order of 100 billions. This forms a very complex network especially taking that some neurons have about 100,000 interconnections. In contrast the largest ANN have about 10 million neurons (AI Website [6]) .

The basic neuron consists of synapses, the soma, the axon and dendrites. Synapses are connections between neurons - they are not physical connections, but miniscule gaps that allow electric signals to jump across from neuron to neuron. These electrical signals are then passed across to the soma, which performs some operation and sends out its own electrical signal to the axon. The axon then distributes this signal to dendrites. Dendrites carry the signals out to the various synapses, and the cycle repeats.

Just as there is a basic biological neuron, there is basic artificial neuron. Each neuron has a certain number of inputs, each of which have a *weight* assigned to them. The weights simply are an indication of how 'important' the incoming signal for that input is. The *net value* of the neuron is then calculated - the *net* is simply the weighted sum, the sum of all the inputs multiplied by their specific weight. Each neuron has its own unique threshold value, and if the *net* is greater than the threshold, the neuron fires (or outputs a 1), otherwise it stays quiet (outputs a 0). The output is then fed into all the neurons it is connected to.

Neurons are the basic processing elements in an ANN . They are interconnected to form a network. The artificial neurons receive inputs that are analogous to the electrochemical signals that natural neurons receive from other neurons. By changing weight therefore, the artificial neuron learns just like the natural neuron.

The diagram below shows a comparison between a biological neuron and an artificial neuron.

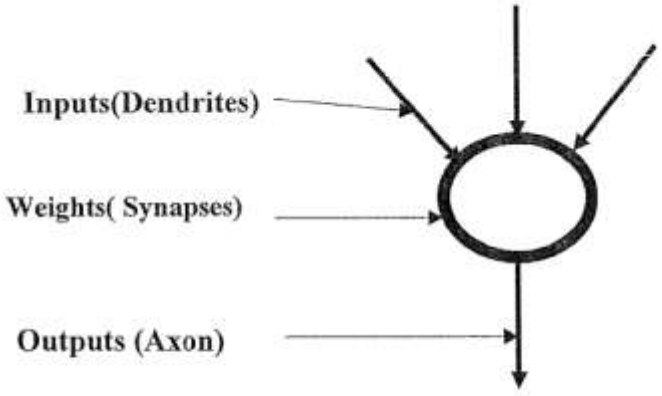


FIGURE 5: *A comparison between the biological neuron with a typical artificial neuron.*

Note the similarities with figure 3 above oth ta e inputs , use weights and generate an output.

A comparison between artificial and natural neurons inspired and motivated the development of the

ANN technology. The ANN therefore attempts to simulate the information processing and the propagation among natural networks. The weight changes in the ANN simulate the way a formal neuron treats each incoming signal as an input and the way the synapses emit signals to other cells helping in information propagation.

In an ANN therefore, finding the weight factor appropriate to a problem is essential for correct employment of the net and the quality of its solution.

2.21.b Learning

A question might be posed, how are the ANN values set? Several network training methods exist as there are the types of network. Some of the more popular ones include:

a) The back-propagation;

b) The delta rule and;

c)Kohonen learning.

As architectures vary, so do the learning rules, but most rules can be @at~gQri~~d into two areas - ***supervised and unsupervised.*** Supervised learning rules require a 'teacher' to tell them what the desired output is given an input. The learning rules then adjust all the necessary weights (this can be very complicated in networks), and the whole process starts again until the network can correctly analyze the data. Supervised learning rules include back-propagation and the delta rule. Unsupervised rules do not require teachers because they produce their own output that is then further evaluated.

h)

2.21.c Architecture

Architecture is the topological organization of a collection of neurons. Neurons are connected in a Feed_forward fashion. Most ANNs have unidirectional connections;

Mode of operation refers to what activities are going on when the network computes.

Style of learning is where some systems correct possible errors in their performance with respect to the provided training data.

The way information is processed and the intelligence stored depends on the architecture and the algorithms of neural net.

i)

There is, though, a standard network architecture.

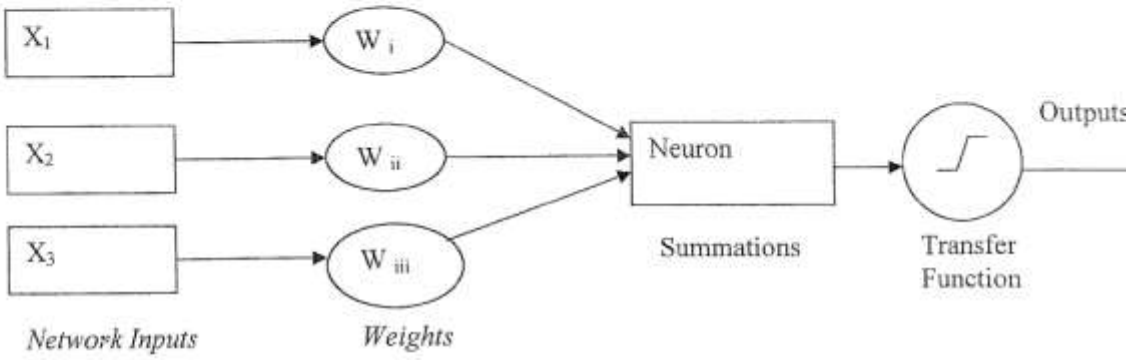


Figure 6. *Processing Information in an ANN (Adapted from Turban 1998 [13])*

The network consists of several "layers" of neurons, an input layer, hidden layers, and output layers. Input layers take the input and distribute it to the hidden layers (so-called *hidden* because the user cannot see the inputs or outputs for those layers). These hidden layers do all the necessary computation and output the results to the output layer, which (surprisingly) outputs the data to the user.

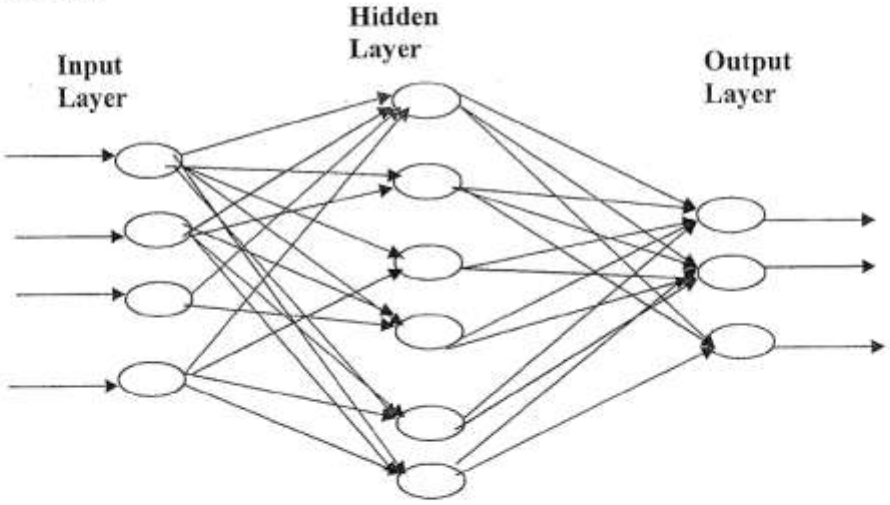


FIGURE 7: *A fully connected neuro_network architecture.*

The diagram below shows and ANN architecture

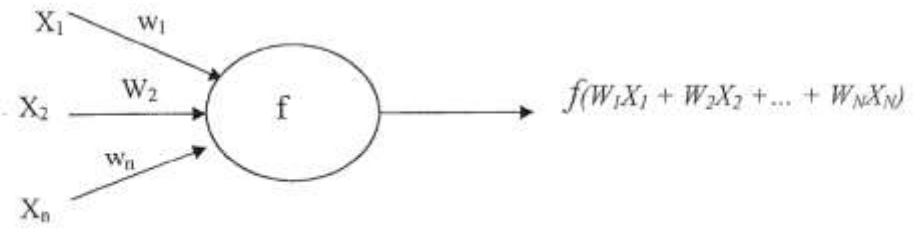


FIGURE 8: *An abstract neuron (Adapted from Turban 1998 [13])*

Growth of Neuro Computing

The re-emergence of neural networks as a paradigm from building intelligent systems gained feet around 1980's. According to Achin Hoffman 1998 [7] the five most important reasons for the popularity in neural networks are:

a) Development in microelectronics. This is especially advancement in very large scale integration making easier to build massively parallel machines. There was an urge to exploit the power in parallelism in machines;
b) Philosophical objectives to symbolic AI. There was a philosophical observation of human thought just as human cognition and understanding is based on social interaction intelligence systems were also seen to be able to interpret embedded symbols and to adopt to change. But unlike human beings whose real representation is implicit, computers cannot behave like human beings and therefore represent values and goals as well of the "view" of real world explicitly by symbols;
c) Practical experiences in the symbolic paradigm. There was need to equip AI systems with a good amount and detail of "knowledge" so as to deal with a complex domain.
d) The philosophical metaphor. Human brain works on a massively parallel basis. This is the result of the astonishing performance of the brain. With parallelism the vision of ANN was becoming a reality.

2.2J.d Backpropagation Rule

Backpropagation also *Generalized Delta Rule* is a general learning rule. It allows finding in a multi-layer perception (MLP). The operation of back-propagation is based on presented patterns and iteratively adjusting the weights and threshold of each neuron depending on whether the produced output is a desired one or not. It therefore compares the value of output signal to desired value and evaluates the difference in order to modify the weights and threshold of the output unit.

Based on the desired output values of intermediate layer units, behavior of intermediate units is modified as well to approach the respectively desired output value. Back-propagation rule modifies the weights in direct proportion to the error in the units to which it is connected.

The following formulae can be used to get the output function in each node

$$f_i[I]=\sum_{i=0}^{nj}(w_{ij}i_{ji}) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 1.21$$

Where

i - Input Vector

w_{ij} – Weight of Unit from output of node i to the input of node j.

2.21.e Neural Computing as a "panacea" to Challenges in Students Enrolment Predication Among the challenges that are to be encountered in the system and which the artificial neural network system will address include:

a) Incomplete and noisy input information. Some information for example castrophies cannot be determined. Also it cannot be easily got from past data. If some data is incorrectly recorded, this introduces noise in the systems. The system should therefore manage incomplete data and be noise-tolerant;
b) Multivalued input attributes. Several conditions might exist making the input challenging;
c) Multiple decisions outcome;
d) Data processing, converting non-numeric data to numeric input. The best input is numeric data.

The solution to the above problems will be to develop a Backpropagation network. The back-propagation network will deal effectively with noisy, partial and potentially conflicting data. Situations that have not been previously encountered.

Numerically coded input and output will mean the system can manage the multi-valued decision outcomes. To cater for the relationship between large number of input and the associated decision outcome, a multi-layered neural network will be preferred. It will also be;

a) Fully Connected;
b) Feed-Forward;
c) Train Backward.

2.21.f ***Neural Networks and Time Series Forecasting***

Recently information technologies and optimization algorithms are aggressively applied to various problems in the area of logistics and supply chain management. Accurate forecasting of demand change often improves the quality of solution to such resource planning problems as production planning and replenishment planning.

Artificial neural network is typically applied when there *ape no available solutions.* There are only many observed (input, output) data pairs. And the input to output mappings could not be or too hard to be analyzed in some mathematical models. For example, handwriting recognition, no simple system could model people's handwriting in any way. This is the case with forecasting, that is, there is no available solution. But when the input-output set are fed and trained, the system is able to learn and come up with a prediction. This therefore made the researcher adopt ANN for prediction of students' enrolment other than the other machine learning techniques.

A neural network is a computational model that is loosely based on the neuron cell structure of the biological nervous system. Given a training set of data, the neural network can learn the data with a learning algorithm; in this research, the most common algorithm, backpropagation, is used. Through backpropagation, the neural network forms a mapping between inputs and desired outputs from the training set by altering weighted connections within the network.

Therefore an ANN functions in the same way like any other forecasting method. That is, input data is provided, coefficients are fitted to model the dominant patterns in the data. A residual error is got. But unlike other methods like ARlMA models, ANN has much greater complexity. And like ARIMA where many types exist, there are many types of ANN networks.
The versatility of application of ANN allows it to model complex mathematical and logical functions.

Unlike the conventional computer programming which use an algorithmic approach where a defined set of instructions is followed to solve a problem, ANN use complex method to learn. That means no specific method is to be followed to come up with a solution making them appropriate for scenarios where the problem is neither well understood nor the solution well known. This makes ANNs suitable for time series forecasting where the solution is not well known.

The above does not mean that ANN and conventional algorithmic computers are in competition but rather they complement each other, in fact, languages used are same. It also means that neural networks needs a software engineering approach to its development just like the conventional systems! In this research project for example, a development methodology was adopted (see figure 10) and C++ programming used to develop the application.

*2.22 Fuzzy **logic***

This area of AI has mainly been used in control systems. This is because it deals with imprecision, that is by use of fuzzy sets. This means we do not have a predefined rule in making a decision. In our daily lives, fuzzy logic is applied for example if a road is blocked, we make a decision on taking another route, but note, one can also decide to wait until the road is clear. Deciding whether to wait or get an alternative route is a fuzzy decision.

2.23 Genetic algorithms

Genetic Algorithms (GAs) are adaptive heuristic search algorithm premised on the evolutionary ideas of natural selection and genetic. GAs simulate the evolution theory as stipulated by Charles Darwin's principle of survival of the fittest. Using GA, random solutions are developed which might not be very good to start with. Through evolution, the "population" is improved. The solutions are coded in binary form in a way the computer can easily understand. These O's and l's form the "chromosomes" which mutate. The solution of the mutation is better than the previous solution. The off springs that are superior are added to the population while the bad ones are discarded, thus the Darwin's principle of "survival for the fittest". This process is repeated for the better off springs thus ensuring realization of better and better solutions.

GA does not therefore involve calculation but selecting the best solution. What is required is therefore to evaluate the quality of the solution to meet our requirement.

Has been applied in traveling salesman problem to determine the most efficient route.

When GAs applied to very large problems, they fail in two aspects:

a) They scale rather poorly (in terms of time complexity) as the number of cities increases. ;

b) The solution quality degrades rapidly.

GA has also been used to determine file allocation for a distributed system. The objective is to maximize the programs' abilities to reference the files located on remote nodes.

Mutation makes GA a rather brutal approach, requiring large amounts of processing power and thus was not the preferred machine learning method for this research.

2.24 Instance-based Learning Methods

This is a simpler machine learning method than ANN because instead, the data series is searched for situations similar to the current one each time a forecast needs to be made. It is classified under the *Instance Learning Methods* which are straight forward approaches to approximating real-valued or discrete -valued target function.

Instance learning methods delay processing until a new instance must be classified hence "lazy learning" methods can estimate the target function locally and differently for each new instance to be classified.

2.24.a k-Nearest ***Neighbour Learning***

Is the most popular instance learning method. It assumes that all instances correspond to points in the n-dimensional space. The target function being learnt may be discrete or real-valued.

It is easily adapted to approximating continuous valued target functions. This is done by calculating the mean of the k-nearest training example rather than the most common value target function. An algorithm is used to calculate the mean value of the k nearest training examples rather than calculate their most common value.

k-Nearest neighbour algorithm, which is distance weighted according to Mitchell, is highly effective inductive inference method for many practical problems. It's robust to noisy training data and quite effective when it is provided with a sufficiently large set of training data.

Note: Unlike the decision tree learning which selects only a subset of the instance attribute forming the hypothesis, k-Nearest neighbour algorithm calculates the distance between instances based on all attributes of the instance.

k-Nearest Neighbor is prone to the problem of curse of dimensionality where only a few attributes of instance are relevant and the further apart neighbors will be dominated by a large number of irrelevant attributes.

Due to the algorithm delays, this has an impact on memory indexing.

2.24.b Case-Based Reasoning (CBR)

Is based on the first two properties of k-nearest neighbour and locally weighted regression. Instances in CBR as represented using more rich symbolic description and the methods used to retrieve similar instances are corresponding more elaborate.

Applications

Has been applied to problems such as

a) Conceptual design of mechanical devices that can be stored of previous designs
b) Reasoning about new legal cases based on previous rulings
c) Solving planning and scheduling problems by reusing and combining portions of previous solutions to similar problems

A major problem with CBR is that the syntactic similarity measures provide only an approximate indication of the relevance of a particular case to a particular problem. Therefore, when CBR attempts to re-use the retrieved cases, it may uncover difficulties that were not captured by this syntactical measure.

When problems of incompatibility occurs in CBR, the CBR system may backtrack and lead to search for additional cases, adapt existing cases, or resort to other problem solving methods . Such difficulties when detected provide training data for improving the similarity metric or equivalently the indexing structure for the case library.

2.25 Decision Tree

Decision trees are used approximation of functions with discrete values (known value e.g. a true or false condition). In *machine learning* (Mitchel 1997 [10]), decision tree learning can be applied to problems such as learning to classify medical patients by their diseases, equipment malfunction by their cause and loan applications by their likelihood of difficulty on payments.

The tasks above will involve classify examples into discrete sets of possible categories. Instances in decision tree learning are classified by sorting down the tree from the root to some leaf node. This provides the classification of the instance.

A node in the tree will specify a test of some attribute of instance while each branch descending from the node corresponds to one of the possible values for this attribute.

"An instance is classified by starting at the root node of the tree, testing the attribute. Specified by this node, then moving down the tree branch corresponding to the value of the attribute in the given example."

Mitchel points that decision tree learning is suited for problems with the following characteristics

a) Instances are represented by attributes -value pairs. Instances with attributes described by a small number of disjoint values, for example, student career-student or working
b) The target function has discrete output values. This can be exemplified by a Boolean Yes or No value
c) The training data may contain errors. This is because decision tree learning is immune to errors. This can be realized during classification of training data.
d) Training data may contain missing attribute values .Where training data has unknown values, decision tree learning can be used.

2.3 Estimating Students Enrollment

Training managers in colleges use past data to predict the number of clients in a given season. Typically, the manager start by reading the information provided about previous month and season experience. The information is on a computer printout showing the student, course enrolled, start time, end time, amount paid and trainer. From this orientation, the training manager arrives at a reasoning by trainer

From this orientation, the training manager arrive at a reason by first generating a hypothesis. The manager the attempts to validate the hypothesis by comparing the current situation with the previous observation by reviewing the existing information.

He therefore goes through several past and differing experiences in order to try and validate his/her hypothesis. Information has therefore to be generated not from one past experience but from several experiences.

The manager will therefore generate a report of findings and provide an appropriate recommendation for his budgetary allocation.

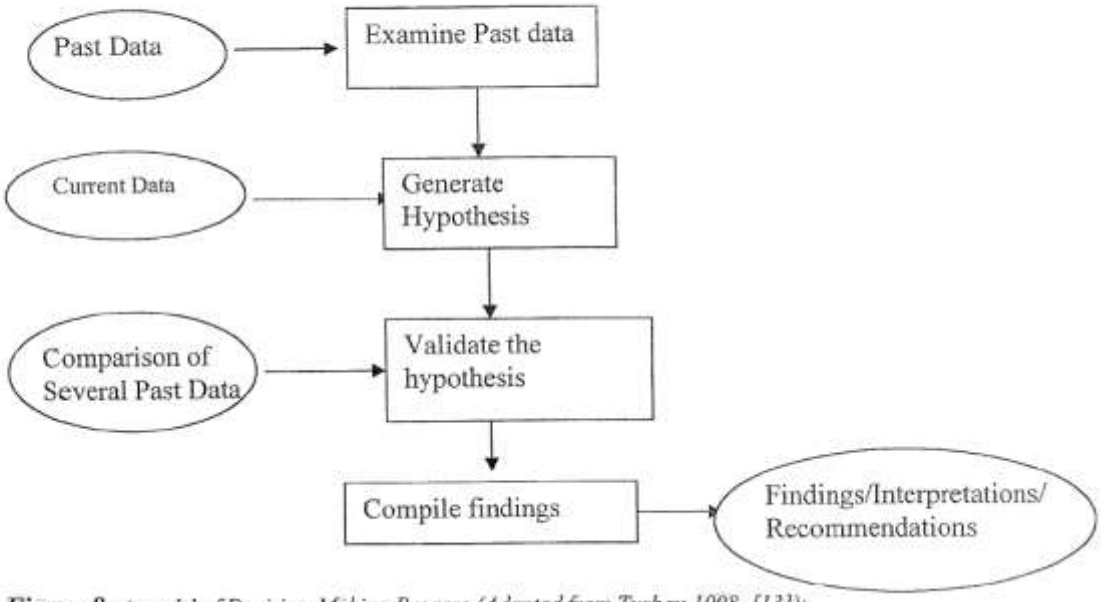


Figure 9: *A model of Decision-Making Process (Adapted from Turban 1998 [13]):*

2.24 Why ANN was preferd to other machine learning techniques predict students enrolment

Artificial Neural Network was chosen as a learning technique because:

a) From the available literature, ANN has been applied in a variety of applications that involve forecasting;
b) ANN, reportedly can deal with noisy, partial and potentially conflicting data;
c) Because their inputs and output nodes are numerically coded and neuron like, they can manage multivalued input attributes and multiple decision outcomes;
d) Prediction/Forecasting does not involve precise numerical values but solution may change over time;
e) The "human like learning" interested the researcher who opted to pick ANN as the machine learning method.

Throughout the research, several ANN "shells" existed which could be used for this research, but the researcher developed a test bed neural network for the following reasons:

a) To best understand and test the various working of the neural network as the researcher believed that this was the best way to learn and understand the intricacies in ANN;
b) The available "shells" could not have catered well for the variables used in this study.

In this research project therefore, ANN is tested as a function approximator for a time series prediction. A real data series is used to analyze the response of a specific ANN configuration.
This research paper presents another neural network forecasting application - forecasting the number of students to enroll in a college. From the correlation indicated above, it is hoped that using ANN and the college time series data will help predict number of students expected to enroll in a college and hence help in proper planning of resources. Also the forecast will help in resource allocation.

During prediction, the training manager exercises a series of judgment to determine what resources to allocate at a given time. Prediction here is based on personal experience. The judgment is complex, subjective to the estimator's capability and fluid in the they may change over time. The problem of students forecasting is of great importance to training colleges both public and private. Taking the fuzzy flow of students in a college, ANN will be appropriate to develop a system to help in forecasting the number of students expected to enroll. Using time series, the data collected from college can be used to forecast future enrolment. The current data series will be observed to enable future unknown data values to be forecasted accurately.

2.5 Related Works

Sheng R. O. (et al) 1998 [11] in their article *Neural Net Learning for Intelligent Patient-Image Retrieval* uses real data to compare use of Backpropagation Neural Network and engineered knowledge base and also real world image pre-fetching practice in radiology. The BPN was noted to tolerate noisy and incompatible data. Their study result *"Demonstrated that AI-based learning techniques (read BPN) can provide intelligent information at significant cost and resources than knowledge engineering"*. The researchers used prior "200 cases whose

clinical readings had been performed by radiologists…" This therefore makes raw data for machine learning using ANN.

One of the most explicit applications of neural networks and genetic algorithms using Time Series Forecasting is by Kingdon (1997[8]). He presents an application of BPN time series forecasting in financial forecasting. In this book, he has presented and evaluated an automated intelligent system for financial forecasting that utilizes neural networks and genetic algorithms. The author starts by describing neural network and genetic algorithms and how they can be used individually before explaining how genetic algorithms is used in the financial forecasting.

Corne, Simon, (et al) from the School of Geography, University of Leeds paper on *"The Use And Evaluation Of Artificial Neural Networks In Flood Forecasting".* Theirs was an investigation at to " .. Whether ANN can provide levels of performance which are sufficient, good and robust so as to provide a basis for practical short term flood forecasting" They compared ANN with among other methods the ARMA time series forecasting model. Over the 15 years of research, they observed that ANN could be applied in flood forecasting.

Other areas worth noting where ANN and time series forecasting have been used include [111] airline passengers, and stock market forecasting.

3.0 METHODOLOGY

To best test the research objectives the researcher base the research on a training institution which provided real data collected. Also, to best understand and test the various working of a neural network, the researcher went a step further and developed a test bed neural network named ***CEFS*** **-** *College Enrolment Forecasting System.*

CEFS *is presented 'with a simple menu interface.* A simple menu driven interface was adopted for this project. This is because:

a) A menu interface saves on the storage space.
b) Users can easily interact with the system with minimal typing. This is by selecting a number from the availed options.

CEFS was written **in** C++ language using the Object Oriented approach. Several classes were used as can be seen in appendix A. Object-Oriented Programming (OOP) approach has the following qualities that made the approach be preferred by the researcher:

a) Emphasis on the objects involved **in** the task, not on the procedure;

b) An object encapsulates a data set with the code that is used to operate on it;

c) Standardized programming modules can be reused;

d) Applications can be rapidly developed with appropriate objects from an object library.

The following are some of the merits of OOP If

used properly:

a) *Code reusability* - similar tasks can be handled by the same code. In this neural net code for example, the code for *Training* was reused by the *Validation* function.
b) *Modularity* - modules in code may be replaced without affecting other pieces of code. Class headers were used for all classes to realize modularity in the program. *Software*

c) *reusability* - old programs may be accustomed to modified tasks. In the case of *CEFS*, the program use the text file as the database for the research data and also Random weights.

The *CEFS* can therefore be used by other researchers in this area of neural computing.

3.1 The Model

The model below was adopted in development of neural network. It represents the phases after feasibility study.

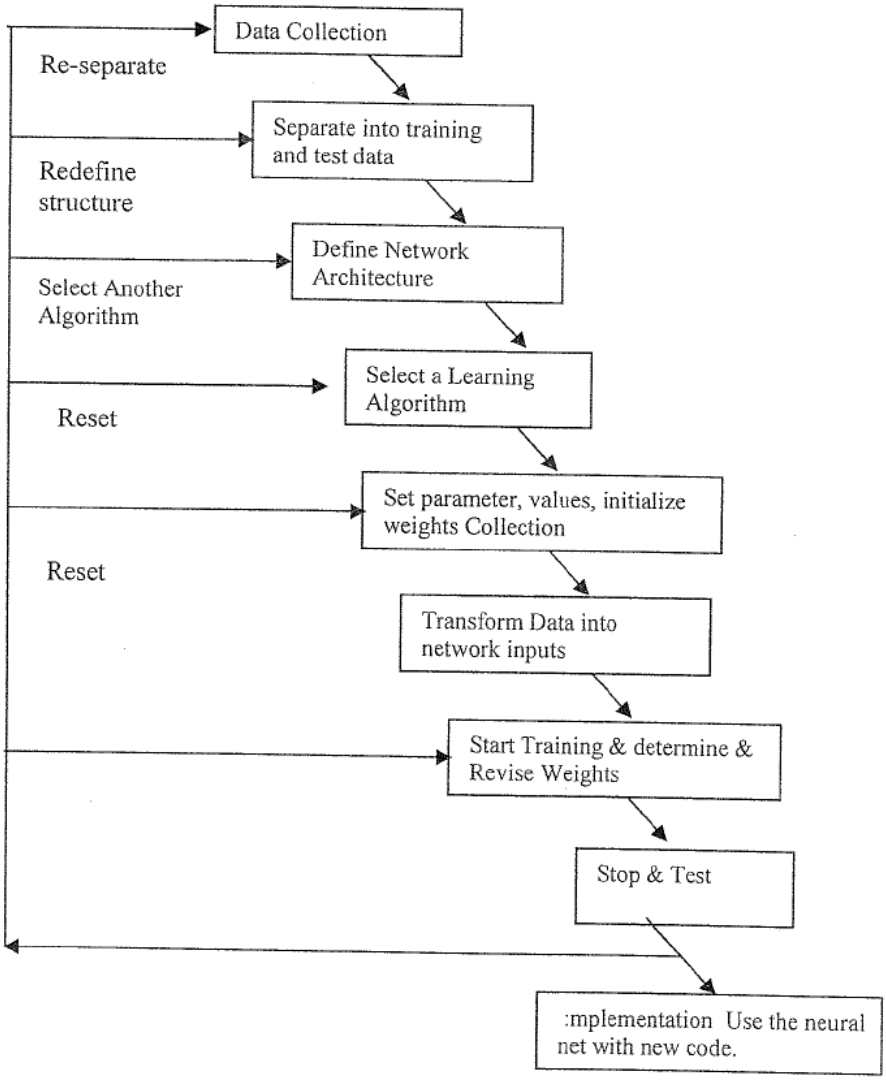


Figure 10. *A Model of Development of an ANN (Adapted from TURBAN 1998 [13])*

3.11 Data collection

The neural net uses real data from a training college - *Vision Institute of Professionals*, computer packages training section of information technology (IT) department. This is used as the raw data for the network. The data is from June 1997 to June 2003. The data was collected daily for the whole period except on Sundays (College closes on Sunday)

3.12 Data preprocessing

The data inputs and outputs were numerically coded. In the inputs for example, a week that fell during the *exam period* or *just after a holiday* was written as 1, but if the week did not fall either in an exam period or after holiday was coded as O. This helped in working with arrays in the ANN. The data is therefore made to be homogeneous. This facilitates training especially where array data structure was used in the algorithm (Note that an array is a homogeneous data structure). The number of records initially was 2600 record sets. This was grouped to enrolment per week in month and reduced the number of record sets for the network to 360. The large number of data set though it increases the processing time improves network training accuracy and prevents over fitting during training. The 360 data set was divided into:

a) Training set (70%);
b) Validation set and
c) Testing set (30%).

Training data set

This is the data was using during training and adjustment was done on the weights to come up with a desired algorithm for the network. Backpropagation (discussed later) was used where the delta rule was applied. The data set was supplied as the inputs to the network and subjected to the weights. Weights were adjusted to come up with the best topology and weights. The data file was

ResearchData Training. txt

For the data to exhibit the time series nature, the data in the text file was sequential. This means that the variable, for example *previous enrolment* could better be taken care of in this manner. For example for a record in week X the variable previous enrolment should be the enrolment in week (X-l)

Validation data set

After training the network using the training data set, the validation set is used to test whether the same result from training set could be obtained with a different data set. Unlike the training set, here, no adjustment of the weights is done but rather the final weights from the training set are used.

The data used here is out of sample data - that is, is not part of the training sample. If the out_ of , sample mean error is consistent with the training set error, the model appears to be valid and can be used for forecasting.

Test Data set

When the system was noted to perform well under the training and validation set, the testing set was used to test how well the network can generalise. This is again another record set that was not used during training. The file containing the source data was *Research.Testing.txt.* Like the case of the data in the training set, the data in the text file used for training was also sequential.

3.13 *Network architecture*

A neural network can be single layered or multilayered. In a single layered neural network, the input layer is connected to the output layer through some weights. In multilayered architecture one or more hidden layers exists between the input and output layers. A multilayered has been discovered to perform well where there are several inputs to produce the desired outcome. A multilayered back propagation neural network was preferred in this study. The problem with multiplayer backpropagation neural network is that it is difficult, if not impossible, to select *a priori* the weights "near" the final values calculated during training. Instead, networks must be initialized randomly, and as a result, training can be quite computationally intensive.

Inputs (Predictors) to the network were:

a. Years(Y)

b. Month(M)

c. Week(W)

d. Exam week(Ew)

e. Whether After Holiday(Ah)

f. Previous enrolment(Last enrolment)(En)

Outputs will include

Number of students to be enrolled (TEn)

The neural network structure chosen was a three layer feed-forward network consisting of an input layer, a hidden layer and an output layer as shown *in figure* 11 below. Throughout literature, results demonstrated that three-layer feedforward ANNs can be used to model real world functional relationships that may be of unknown or poorly defined form and complexity. However, many problems still remain unsolved. The first problem involves specification of the number of nodes required in the hidden layer; the more complex the mapping, the larger the number of hidden nodes required. The second problem involves finding the "optimal" values for the connection weights. The structure can be described as $6 X n X 1$. The $6 X n X 1$ means the neural net will comprise of six inputs, n (definite number to be established after training) neurodes, and one output. To solve the first problem (number of nodes required - *n for this case)* several structures were experimented and recorded (see figure 16). This was done by starting with a small number of nodes and then gradually increase the network size until the desired modeling accuracy was achieved. This dependeOnt on the ability to find an "optimal" weight (please see 3.14 *Training weights).* Therefore, the second problem was solved as can be documented in 3.14 below.

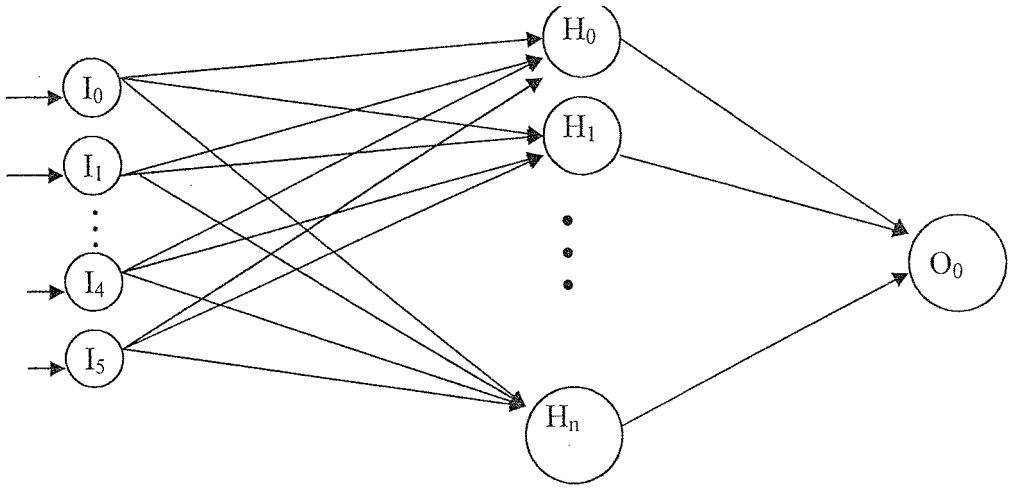


***Figure* 11:** *Fully Connected NetWork Architecture*

The scaled inputs (input was scaled between 0 and 1 by use of sigmoid function - see formulae 2.3) are distributed to the hidden node (neurons). Each neurode receives all the scaled input variables. This result in parallel processing of all inputs at multiple nodes.

Besides the network being multilayered, it is also :

a) *Fully connected:* Where each node in one layer is connected to every other node in the next layer;
b) *Feed Jorward:* Data activation takes place from input to output;
c) *Train _Backvvard:* during training, the weights are adjusted using the deviation of the expected output from the computed output.

The following pseudo-code gives an illustration of the feed-forward and back-propagation multi-layer neural network:

```
begin
        create a multilayer network
        initialize all weights w to small random numbers
        while terminate_condition not meet do
        begin
                for each (input, output) pair in training_data do
                begin
                        calculate output for each unit: sig(sum(wx))
                        calculate overall error E
                        for each unit, calculate its error term E
                        update each weight: w = w + f(E)
                end
        end
end
```

3.14 Training weights

The network's initial data weight was randomly generated. This facilitated commg up with enough data and in short time for all the weight requirements. The random data generator library function *(randtl)* in Microsoft Visual C++ was used. The inputs were "seeded" to facilitate generation of different data set incase during learning the network was not able to generalise after one epoch.

To achieve the best learning performance, a learning rate and momentum factor were chosen.
The user of the *CEFS* is prompted to enter the desired learning rate as shown below:

Please select the **desired LEARNING RATE:**

a. 0.05
b. 0.10
c. 0.20
d. 0.25
e. 0.30
f. Exit

Euter **your choice (a-f)**

These were to control the network's learning process.
"The *learning rate* denotes the effect of errors on weight adjustments and therefore influences the networks stability and convergence rate [8]". The learning rate also known as the learning coefficient is set so that the change in the weights is not too rapid or too slow.

The learning Rate chosen is used throughout the learning epochs.

"The *momentum factor* determines the effect of the previous weight adjustments on the current one [8]" The momentum factor accelerates convergence and minimises potential oscillations. This may result from the system going to a local minima rather than a global mimima. To offset the reduction in steepness that occurs at local minima, a momentum term is incorporated. Again a fixed momentum factor was adopted for this research.

3.15 Learning Algorithm

Turban [l3] states that there are more than 100 types of learning algorithms. In this project, *supervised learning* was adopted. In supervised learning, the desired outputs are known. Specifically, the backpropagation approach is used where the difference between the desired and the computed outputs are used to adjust the weights during learning.

All the inputs will be converted to numeric for equivalence.

The correctness of the output will be determined and weights adjusted so as to achieve correct results for a given input. Six input patterns will be fed to the system. Computer on each iteration adjusts weights. The steps will be repeated until the weights give a uniform set of values.

Formulae:

$$\sum_{J=0}^{6}(A[i][j] \times w[j][r]) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 2.1$$

$$\sum_{i=0}^{M}\sum_{r=0}^{N}(A[i][j] \times w[j][r]) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 2.2$$

M designated the number of record rows (Number of rows in the program are counted during run time)

N designates number of nodes (are dynamic depending on number chosen during running of the program)

3.16 Using Backpropagation

3.161 Backpropagation Training

The method of propagating information about errors at the output layers back to hidden layers is known as *Backpropagation* or *Backprop.* Learning in a neural network involves adjustment of values to come up with the values (of weights, learning rate and momentum) that the network can use to achieve an optimal generalisation. To make meaningful forecasts, the neural network has to be trained on an appropriate data series and weights. The examples in this case which were in the form of <input, output> pairs were extracted from a real data series, where input and output are vectors equal in size to the number of network inputs and outputs, respectively. For every example, backpropagation training consisted of three steps:

a) Present an example's input vector to the network inputs and run the network: compute activation functions sequentially forward from the first hidden layer to the output layer;
b) Compute the difference between the real output and the actual network computed output;

c) Propagate the error sequentially backward from the output layer to the first hidden layer (the error here, known as delta, was multiplied with momentum value and learning rate);
d) For every connection, change the weight modifying that connection in proportion to the error;
e) Repeat the above steps.

The steps were performed for every example in the network, which constituted an epoch. Training usually lasts thousands of epochs, possibly until a predetermined maximum number of epochs *(epochs limit)* is reached or the network output error *(error limit)* falls below an acceptable threshold. The threshold in this project was varied to come up with the best network.

Two methods were used during training to control when to stop training:

a) In the first method, the maximum number of epochs was set to 10,000. This means that the network could iterate for 10,000 times upon which if the mean error had not gone below a given threshold, training would stop.
b) The other method was by setting a threshold, which was chosen at the preliminary stage by selecting one of the Menu options as shown in table below:

Please select the desired MINIMUM ERROR:

a. 0.15
h. 0.20
c. 0;25
d. 0.30
e. 0.35
f. Exit

Enter your choice (a-f)

This means when a mean train error reaches the minimum specified or less, training stops and the weights are saved.

Note, if the network iterates 10,000 times without achieving the set threshold, are not saved but new weights are set and training is repeated. This controlled the number of iterations the network perform and gave the trainer an interactive working with the system.

In the first step, an input values were presented to the network, then for each layer starting with the first hidden layer and for each unit in that layer, compute the output of the unit's activation function (the sigmoid function is used- equation 2.3)

$$1/(1+e^{-n}) \quad \text{..... 2.3}$$

If the output (or activation) is far from the desired one which often is 1,0 (-1) the derivative is relatively high because we are somewhere in the middle of the activation function, which means weight changing is "forced". The function ensured that each output produced was between minus one(-l) and one (+1). The hidden layer outputs were subjected too to output layer function and also subjected the sigmoid function shown above. The output value computed is between mmus one (-1) and one(+ 1) .. This is forward propagation.

Eventually, the network will propagate values through all units to the network output(s).

(Final output)$O = 1/(1 + e^{-n})$

In the second step, for each layer starting with the output layer and for each unit in that layer, an *error term* is computed.

Scaled Error(e) = Desired output(D) - Computed value (0)

$$ei = D - 0 \quad \text{..... 2.4}$$

where i denotes the observation at training set

Based on the errors in the step above, the weights throughout the network are adjusted so as to move to minimisation of the mean error. Note that training used here was batch oriented as shown in the algorithm below:

Initialise the weights

Repeat the following steps if scaled error is greater than threshold/minimum error):

Process all the training data

Compute the scaled error

Update the weights

New Weight = old weight + (Calculated Error) X *learning Rate* X *Momentum value* The new weights are now used in forward propagation.

The epoch scaled error is a mean error computed as shown:

MSE=-i ((2:(ei)2)/nt) 2.5

Where:

MSE = Approximately the Mean standard error

ej = computed scaled error in the last epoch

n, = Number of observations in the epoch/Training set

In this study, the training data set MSE was compared with the validation data set MSE. If the two values were consisted, the model appeared to be valid and could be used for forecasting. The figures appeared consistent from 100 neurons and thus the 110 neurons were picked to represent the number of neurons for the network. This number was reached at after several training.

After backpropagation therefore, the final network topology becomes a 6 - 110-l.

That is:

1. 6 input sets;
11. 110 neurons and 111.
1 output.

But the user has a choice on the preferred number of nodes as shown in the table of menu below:

Please select the desired number of nodes for the network:

a. 95
b. 100
c. 105
d. 110
e. 120
f. 130
g. Exit

Enter your choice (a-.g)

3.17 Testing the Network capability

As noted earlier, this project involved the use of real data series. The *Test data set* form the initial data set is used to test the suitability ofthe model for forecasting. This will mainly deal with first evaluating the network capability in relation to the benchmark real world applications. The power and efficiency of the system will therefore be a main consideration during the evaluation. After the *Train Data set* and the *Validation data sets* are noted to produce consistent RMSE, the *test data set* comes in handy .. B seeding the network inputs with the last data points, the network is making *out-of-sample* forecasts. This is done by using the test data set. The training set and validation set are considered *in-sample,* which means the network was trained on them. The training set is clearly in-sample.

4.0 EMPIRICAL RESULTS

In course of this study, the researcher came across some developed neural network. One of the characteristics of these systems was that the data series was a simulated data series. In one of the projects Microsoft Excel randomly generated the training data. In this study, the training data was a real data.

The observations and conclusion in this study were therefore empirical and based on a real data series. This however does not mean that he data is more accurate than others as the data series is not generally applicable. But the weights were randomly generated.

To effectively train a neural network, the errors must be monitored during training. Each error value has a specific relation to the training procedure. Total squared error is given in Equation 1.6 and is a common metric for comparing how well different networks learned the same training data set. The Total Squared Error was displayed in course of training to help monitor the progress.

To test the effect of the available data series on training, the input data was varied. It was noted that with less data availed to the neural net, the Total Squared Error less and network seemed to generalise faster. Unsealed error was also used to help give the real perception of the error magnitude. This was error computed was from the actual "unsigrnoid" figure.

It was observed that arriving at the proper architectural design and parameter values requires considerable experimentation. In this research, the number of hidden nodes needs to be chosen carefully. This is because an inappropriate number can make the network *underfit* or *over fit* during training. It was observed that the training error decreases as the number of nodes increases (see figure 14 below]. This means the network becomes more flexible in fitting the data. The average error eventually almost levels out. At this point, the number of nodes for the network were taken (Any value between 95 and 110 was found to be a good representative). As shown in figure 4, variation in the mean squared error values becomes minimal between 85 and 160 nodes.

The systems was seen to generalize well when number of hidden nodes were about 110 as shown below:

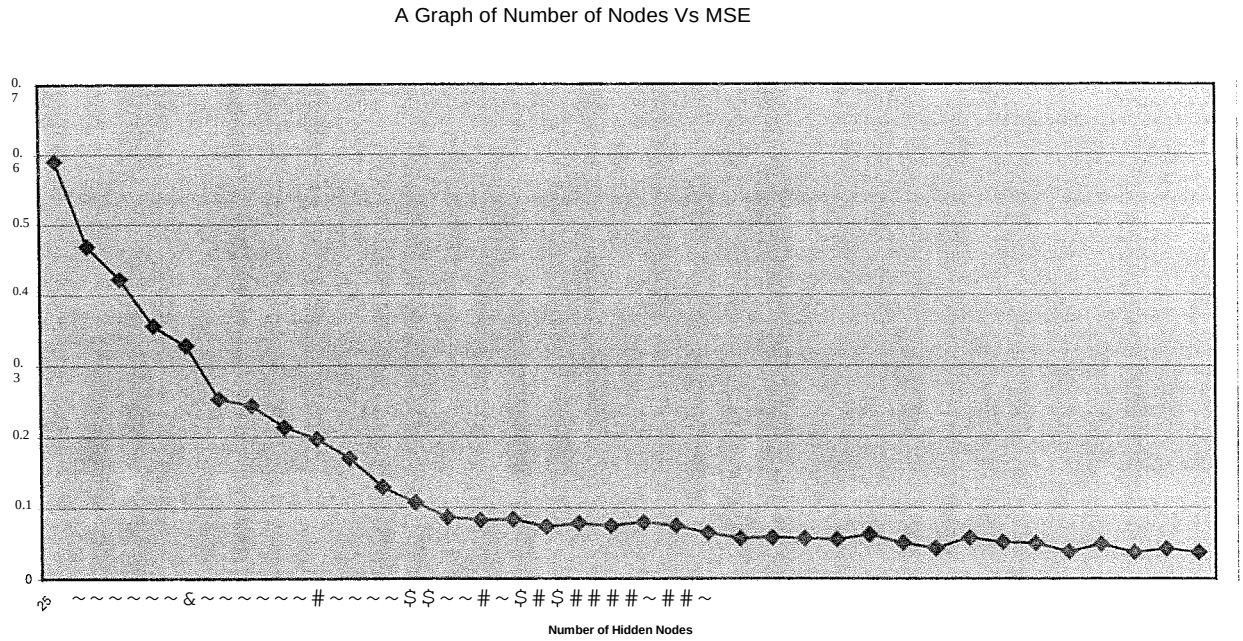


***Figure* 12:** *A Graph of Number of Nodes Vs MSE*

Parameter tuning was concerned with selecting the learning rate and momentum factor for the best performance of the neural network. This was done experimentally. In the first case, learning rate was set to various values between 0.05 and 0.3 while momentum was set to 0.7 [see table 1 below]. Network learning was timed under this values and column two filled. Next, the momentum was set to 0.9 and the learning rate again varied [see table 2 below]. Again the tie taken for learning was recorded.

Learning rate	Time(Sec)	Moment
0.05	6.23	0.7
0.1	0.02	0.7
0.15	1.2	0.7
0.05	3.1	0.7
0.25	1.0	0.7
0.35	0.44	0.7
0.2	0.5	0.7
0.3	0.4	0.7

Table 1: Time Taken to Train A NN at different learning rates and Momentum of 0.7

Learning Rate	Time (Sec)	Moment
0.05	02.3	0.9
0.1	01.03	0.9
0.15	00.63	0.9
0.2	00.55	0.9
0.25	00.53	0.9
0.3	00.44	0.9
0.35	00.39	0.9
0.4	00.31	0.9
0.45	00.28	0.9
0.5	00.2	0.9

Table 2: Time Taken to Train A NN at different learning rates and Momentum of 0.9

From the above experiments, it was noted that the learning rate chosen has an effect of errors on weights adjustment thus influencing the network convergence. A low learning rate meant the

network took longer to train. This has a danger of making the network trapped to a local minimum making the network take longer to train (a condition where the mean square error becomes constant). On the other hand, a high learning rate accelerates convergence . This required a consensus and therefore, a learning rate of 0.15 was preferred.

The next challenge was to find an appropriate momentum factor. From previous research, it has been noted that the momentum factor "determines the effects of the previous weight adjustments on the current one. A properly chosen factor accelerates network convergence" [11]. The learning rate was therefore set at 0.15 and the momentum factor adjusted. A consensus momentum factor of 0.9 was adopted- see table 3 below.

Learning rate	Time(Sec)	Moment
0.15	03.1	0.0
0.15	00.8	0.9
0.15	00.7	0.8
0.15	01.2	0.7
0.15	00.8	0.6
0.15	00.9	0.5
0.15	01.4	0.4
0.15	02.60	0.3
0.15	02.7	0.2
0.15	02.8	0.1

Table 3: Time Taken to Train the ANN at different Momentum and earning rate set at s and Momentum 0.15

As can be noted, trial and error method was used to arrive at reasonable value of parameters (Learning rate, Momentum factor and also number of nodes).

5.0 DISCUSSION

In this research, forecasting procedures is described and a test bed backpropagation neural network was developed which learns forecasting of student enrolment in a college. The learning performance of this neural network is compared with a benchmark developed based on statistical method. The results are discussed next.

5.1 Evaluation of the Neural Network

An experiment was done to evaluate the Neural Network. The neural network performance was evaluated relative to a bench marked statistical (Auto Regression) model developed. The Autoregression model was developed using the *Prais- Winsten* autoregression method and model developed using SPSS statistical package.

5.2 Learning Performance

The effectiveness of a time series forecasting network can be measured by its accuracy in meeting forecasting needs. In this research, *Recall Rate* was used in evaluation of the network. *Recall Rate* is the percentage of the accurately forecasted figures from the testing data set after model is trained. It therefore demonstrates the reliability of the system in prediction

5.3 Procedure

The Neural Network was trained using a training set and tested using the testing set. A total of 260 records were used during the training and testing. The same training and testing data set was used to generate and test the auto regression model. Because multiple forecasts could be made, a threshold was set to decide whether a particular forecast could be used. Below are the results *5.4 Results*

Model	Recall Rate (Hit Rate)	Average Error
Artificial neural Network	71.3%	0.2
AutoRegression model	62%	1.5821

Table 4: *Learning performance with an Autoregression model and ANN forecasting*

For the results of the predictive model for AutoRegression, see appendix D.

It also took four days to develop the autoregression model using SPSS package, while to train the neural network too about 9 hours. This shows the efficiency of a neural net especially when it come to time.

From table 4 above, it is clear that the ANN performance was better than the AutoRegresion model. The main benefits of the ANN over the Auto regression model can therefore be seen as:

a) The ANN will be faster than the Auto regression model in producing the outcome. This can be confirmed by the time taken to come up with the models. For the Autoregression model, it took 4 days while the neural network training took only 9 hours;
b) The ANN is more flexible. Note that the Auto regression model is static. That is, the generated model is used in all forecasting. It therefore assumes that the model got is the best. This is not the case with ANN as it brings in flexibility.

5.5 CONCLUSION

One of the objectives of this research was to come up a neural network prototype to demonstrate the application of ANN in time series forecasting. From the developed test bed and in comparison with the regression model developed, the following conclusions can be made:

This research has investigated the application of a learning system in forecasting. The results are significant as they demonstrate that machine learning techniques such as artificial neural networks can provide ***intelligent information systems*** at a significantly low cost in time and resources than statistical methods. This can be demonstrated by the time required to train the system and the performance results. Specifically, they demonstrate the applicability of the A1'-<~ as a function approximator.

But, it is good to note that the above conclusion - ANN performs better than Regression method, is made from the empirical observation in this research and on the available data set. The observation should therefore not be construed as the case in any data and in all circumstances. In

fact, through out literature, some researchers observed that Regression method performed better than Artificial Neural Networks!

The neural net was found to train and produce an error below a given threshold. But it was noted that similar trained neural networks could produce drastically varying forecasts. This could be explained as resulting from the fact that at each time the *CEFS* was run, the random weights varied. This however does not mean that the system is erroneous but rather this can be seen as one of the demerits of neural network to deal with complex situation. Through weight agjustment through training, the neural net was found was found to generalise well regardless of the variation in the computed output. The forecast had a high percentage of accuracy as the Total Squared Error was below the set threshold after training. This demonstrates the applicability of ANN in time series forecasting.

The research also show that arriving at the proper architectural design and parameter values usually requires considerable experimentation *(see figure 16 and Table 1,2 and S above).* This means parameter tuning is paramount so as to come up with the best network architecture for the intelligent system. As one neural computing researcher noted, the above observations add up to one thing: *feed-forward neural networks are extremely sensitive to architecture and parameter choices, and making such choices is currently more art than science, more trial-and-error than absolute, more practice than theory.*

It can also be noted that though the neural net performs well on the available learning data, a small data set (only 235 training records) put the network to a disadvantage. However, this performance is expected to improve as more data is added to the existing set. Note that the *CEFS* uses dynamic arrays and therefore more records will conveniently be accommodated

Finally, the field of neural networks is very diverse and opportunities for future research exist in many aspects, including data preprocessing and representation, architecture selection, and application. The future of neural network time series forecasting seems to be in more complex network types that merge other technologies with neural networks. Nevertheless, a theoretic

6.0 APPENDIX A: REFERENCES

1" http:/www.pd.astro.it/TNG/TechRep/re67/rep67.html

2. http:www.shet.ac.uk!psychology/gurney/notesI11/section3-4.html

3. http;!/www.immex.ucla.edu/profDevelopment/ caseStudy/frank 1.html

4. http://www.thaiengineering.com/aliicle/ ArtificialNeuralNetworks.html

5. DELURGING, S.A. (1998) Forecasting Principles and Applications, McGraw-Hill, 5th ed.

6. "Artificial Intelligence," Microsoft® Encarta® Online Encyclopedia 2003 http://encarta.msn.com

7.HOFFMAN (1998) Paradigms of Artificial Intelligence a Methodological and Computational Analysis, Springer, Singapore.

8. KINGDON, Jason (1997) Intelligent Systems and Financial Forecasting: Perspectives in Neural Computing, Springer, London.

9. LAUDON, K and Laudon, J. (2000) Management Information Systems: Organisation and Technology, Prentice_Hall Inc. 4th Ed.

10. MITCHELL, T.M. (1997) Machine Learning, McGraw-Hill,NY.

11. SHENG, O.R.L et al (1998) *Neural Net Learning for Patient_Image Retrieval* IEEE Intelligent Systems and Their application, Jan *lFeb* , p 49-57

12. SIRANDULA, MA. *Use of Artificial Intelligence Systems in Analysis of Biological Data.* M.Sc. (IS) Project 2001

13. TURBAN, E. and ARONSON, J. (1998) Decision support Systems and Intelligent Systems ,U Prentice Hall Incl.

14. WONNACOTT, T.H. and WONNACOTT, R.J. (1994) Introductory Statistics For Business and Economics, John Wiley & Sons, NY.

15. PERRY,G(1993) Teach Yourself Object Oriented Programming with Turbo C++ in 21 days, Sams Publishing, Indianapolis, 1 st ed.

16. SCHILDT, H. (1998) Teach Yourself C++, Mcflrawl-lill.Berkley.S"' ed.

7.0 APPENDIX B: Class List

Coding will be done in Visual C++.

Following function are necessary for realisation of this program.

Table 4: Functions

FUNCTION	PURPOSE
Username ()	Is the function that allows the user to enter his/her user name
Password ()	Is the function that allows the user to enter his/her pass word.
AccessWord ()	This is a member function to the class *pass* that is used to authenticate the *Username* and *Password* entered.
RowOrColNumbers ()	Initializing a constructor. The number of rows in the raw data is got
GetRows ()	This is a member function to the class *RowOrColNumbers*. It returns the number of records/rows from the text file containing the data set for training or testing purpose.
GetCols ()	This is a member function to the class *RowOrColNumbers*. It returns the number of columns (also used as neurons in the neural net) from the text file containing the data set for training or testing purpose.
ResData ()	Is the constructor to the class ResData. It is used to read data from the *Training* or *Testing* data set and flood the data into the dynamic array for use during propagation.
~ResData ()	Is a destructor for the class ResData.
DataRow ()	Is a member function to the class *ResData* Function outputs the current data row.
WeiData ()	Initializing a constructor. The instantiation of this class helps get the random weights for the neural net. Data is got from the *RandWeights* text file.
~WeiData ()	Is the destructor for the WeiData class

WeightCol ()	Is a member function to the class *WeiData*. Outputs the current data column
RandWeights()	Is a member function to the class *WeiData*. It is purpose is to generated random data to be used as weights in the neural net and save it to a text file.
SaveWeights ()	Is a member function to the class *WeiData*. Saves the weight matrix after training to be used in the learning set
Propagate ()	The constructor to the class *Propagate*. It is used to initialize the minimum error for the network.
~Propagate ()	The destructor to the class *Propagate*.
Propagatting ()	This function if used in the forward and back propagation of the network. It is used for both training and validation purposes and therefore produces the training error and the validation error respectively
Testing ()	This function is used for testing the neural network so as to determine its suitability as a function approximator.
RandNumber ()	This is the function for random data generation
Sigmoid ()	This is the transfer function that coverts an output to between –1 and 1
Main ()	Is the main function
SetMinError ()	Function to returns the threshold (*Minimum error)* to be used during training. The threshold guides the training in the network.
GetMeanTrainError ()	Outputs the computed training error. This is the MSE after training
GetMeanTestError ()	Outputs the testing error to be compared with the training error. It is the MSE got from testing data set.

m)

Table 5: Classes	
CLASS	**PURPOSE**
Pass	This class has the function that calls the password function. It is instantiated by the main() function
BasicRowOrCol	This class is an abstract class. It is used to describe the variables to be used in the system. It is instantiated by Propagate It is instantiated by the main() function
RowOrColNumbers	This class has the function used to compute the number or records (Rows) from the data file.
ResData	Initializing a Constructor. The data (input data) is feed into he dynamic array . It is instantiated by the main() function
Propagate	This the forwardpropagation and backpropagation class. It is instantiated by the main() function
WeiData	Initializing a constructor. The instantiation of this class helps get the random weights for the network. It is instantiated by Propagate

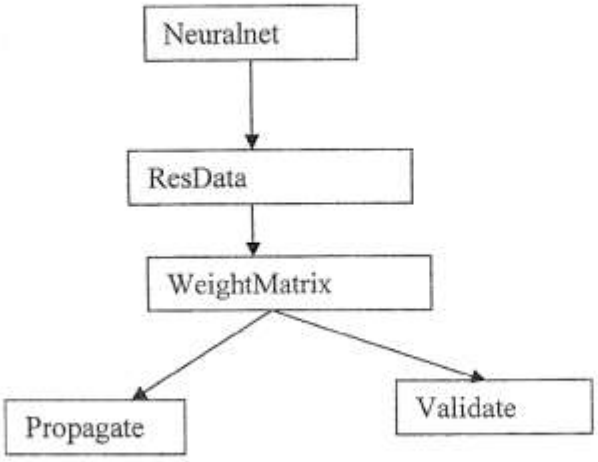


Figure 13: *Neural network structure*

8.0 APPENDIX C: Environmental requirements

4.1 Software Requirements

1. Operating systems- Windows NT server or UNIX or Win 98
11. Programming Languages C++

4.2 Hardware Requirements

1. At least one IBM server
   - 10GB Hard Disk
   - 192 MB RAM
   - 100 MB Zip Disk
   - Floppy Disk Drive
   - Zip Drive
11. IBM Workstations each with at least
   - 4GB Hard Disk
   - 32MB RAM
   - Floppy Disk Drive

9.0 APPENDIX D: PREDICTIVE MODEL FOR AUTOREGRESSION FOR CEFS

The following is the result when SPSS (Statistical Package for Social Services) package was used to come up with a prediction model using Prais- Winsten model

```
MODEL:  MOD_1
Model Description:
Variable:    TOTALENR
Regressors: YEAR
            MONTH
            WEEK
            AFTERHOL
            EXAMWEEK

95.00 percent confidence intervals will be generated.

Split group number: 1  Series length: 72
No missing data.

Termination criteria:
Parameter epsilon: .001
Maximum number of iterations: 10

FINAL PARAMETERS:

Estimate of Autocorrelation Coefficient

            Analysis of Variance:

               DF   Sum of Squares      Mean Square

Regression      5         943.2386         188.64772
Residuals      65        1885.2818          29.00433
```

Variables in **the** Equation:

	B	SEB	BETA	T	SIG T
YEAR	1.59724	.35253	.46439318	4.5307	.00002578
MONTH	-.21285	.19026	-.12537203	-1.1187	.26738203
WEEK	.13068	.45834	.02970994	.2851	.77 64 6656
AFTERHOL	-1. 60667	1. 78940	-.09554332	-.8978	.37256333
EXAMWEEK	4.23696	2.70594	.17202916	1.5658	.12225105
CONSTANT	-3185.76842	705.14150		-4.5179	.00002700

The following new variables are being created:

Name	Label
FIT **1**	Fit for TOTALENR from AREG, MOD_l
ERR **1**	Error for TOTALENR from AREG, MOD_l
LCL **1**	95% LCL for TOTALENR from AREG, MOD 1 95% DCL
DCL **1**	for TOTALENR from AREG, MOD 1 SE of fit for
SEP 1	TOTALENR from AREG, MOD **1**

The predictive model is given below:

Total Enrollment=-3185.768+1.597Y+O.1307W-O.2128M-l.60667A+4.23696E

Where:

Y= Year of enrolment

W= Week of enrolment

M= Month

A=Afterholiday

E= Exam week

RESULTS PREDICTIVE MODEL FOR AUTOREGRESSION FOR CEFS

At 95% confidence interval; the following was observed:

Standard deviation on % error=51.5

Absolute mean(Median) on % error =46.7

Average error = 2.5821

Hit Ratio (Recall Rate) =62%

Therefore, with an average error of 2.58 at 95% confidence interval, the model has a very strong prediction.

The medium was chosen as the most representative average for the sample data because:

a) Half of all values are equal to or less than the median and the other half are greater or equal to median. This means that as the middle value in the distribution, it is the most representative;
b) Because of the extreme values in this distribution, the usual arithmetic mean cannot be used as it gives distorted values (a major limitation because it uses all the values in the distribution)
c) The median is the best average because it can be used even when data is not complete, that is , there are missing values.

p)

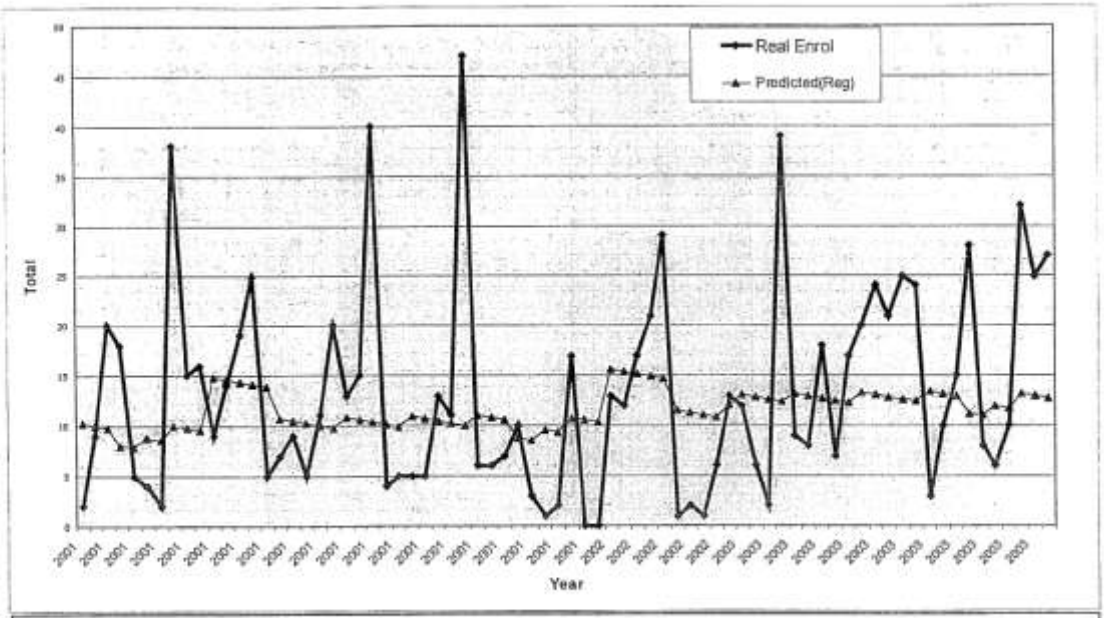


Figure 14: *A Graph of Real Output Versus Computed output using the conventional method*

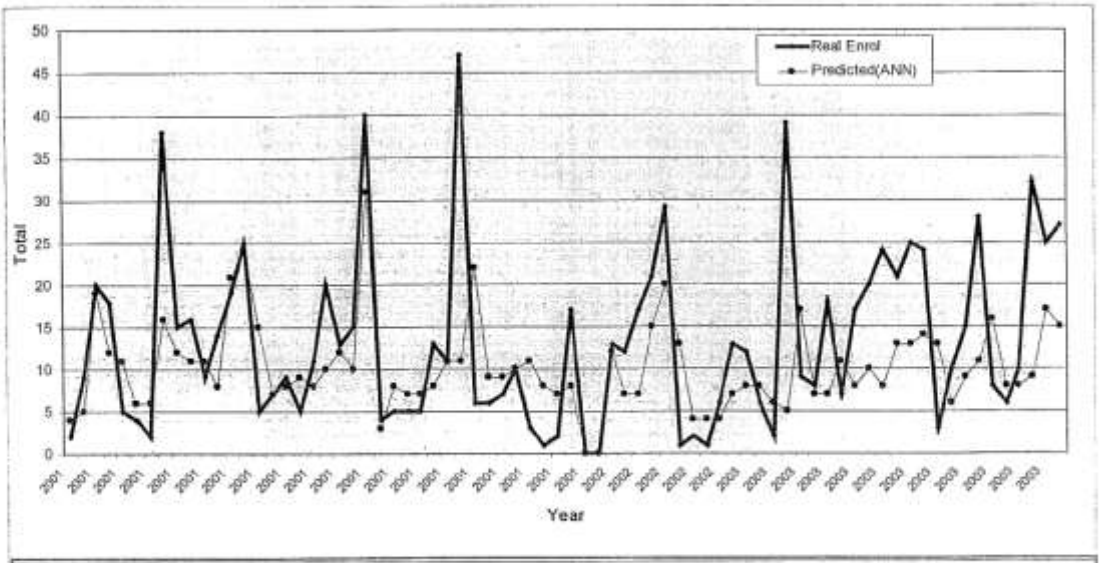


Figure 15: *A Graph of Real Output Versus Computed output using the Neural Network*

10.0 APPENDIX F: USER MANUAL

Introduction

This user manual is aimed to aid the user in applying the software for prediction. Steps 1-12 should be done by the system administrator or head of IT department who has undergone some training to understand the terminologies such as *Minimum Error, Learning Rate* and *Number of Nodes.* The final user should use the screen provided in step 13.

To use the system:

1, Go to *start* button;

2. Select *Run:*

3. Open the CD ROM by typing D: in the window that results;

4. Click the *MAIN* icon

5. The window below appears

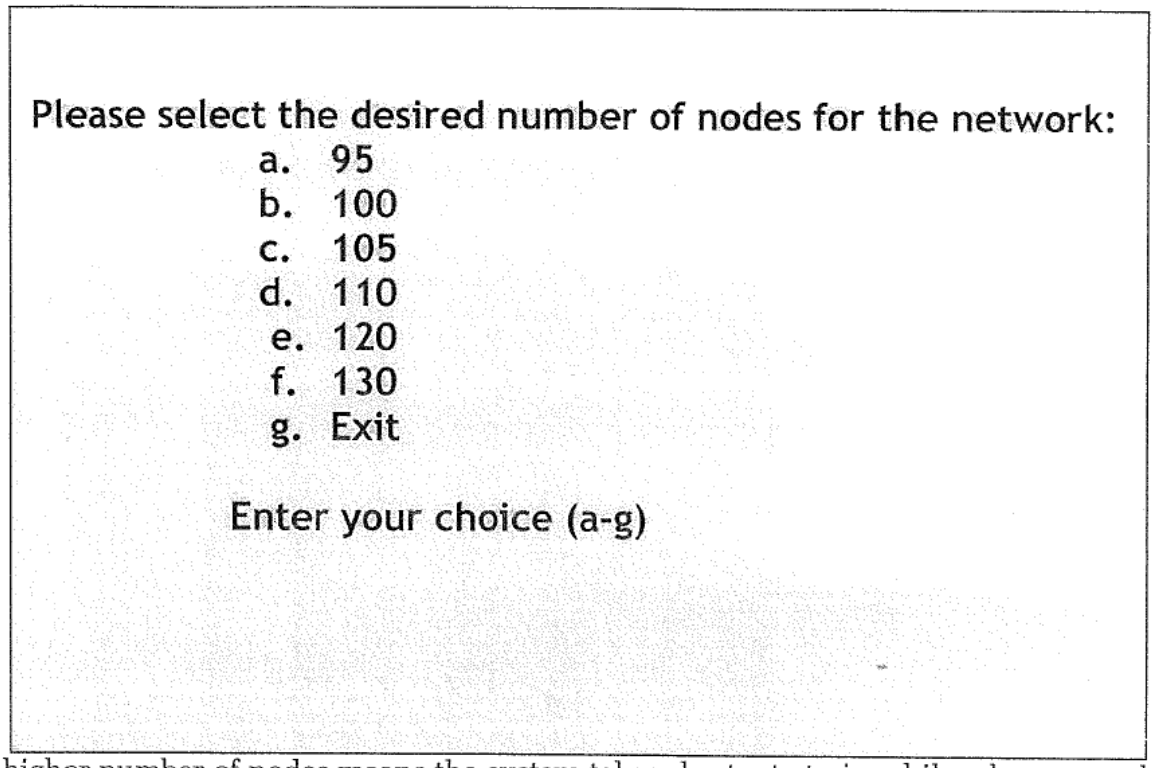
<< CEFS (COLLEGE ENROLMENT FORECASTNG SYSTEM) »

<< THIS IS A TIME SERIES FORECASTING NEURAL NETWORK>>

System Authentication code ...Welcome ...

Enter your username >>

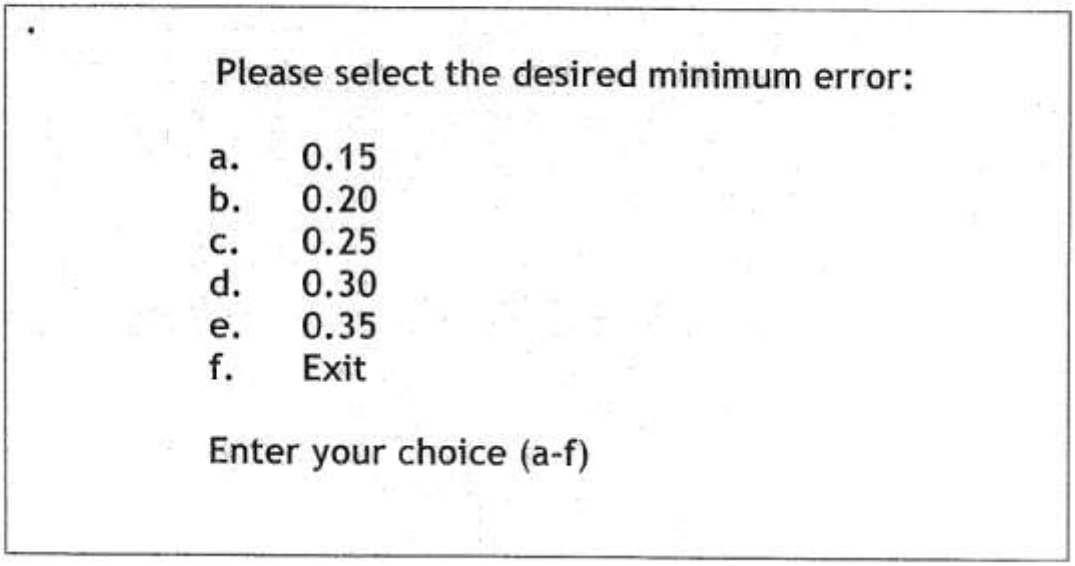
6. Enter the *USERNAME* and the *PASSWORD.* (The system allows you to attempt three times otherwise, the system exits: Note, the password is *PASSWORD* and username *USERNAME)* On entering the correct username and password, the screen below appears:



A higher number of nodes means the system takes shorter to train while a lower number of nodes means the system takes less time

7. On entering the number of nodes, the screen below appears;

A low minimum error means the system takes longer to train than a high one.



8. The screen below then appears:

Please select the desired LEARNING ·RATE:

a. ***D.05***
b. ***D.10***
c. ***D.2.0***
d. ***D.25***
e. ***D.30***
f. **Exit**

Enter yoarcheice (a-f)

Again, a high learning rate accelerates learning. A low learning rate is preferred.

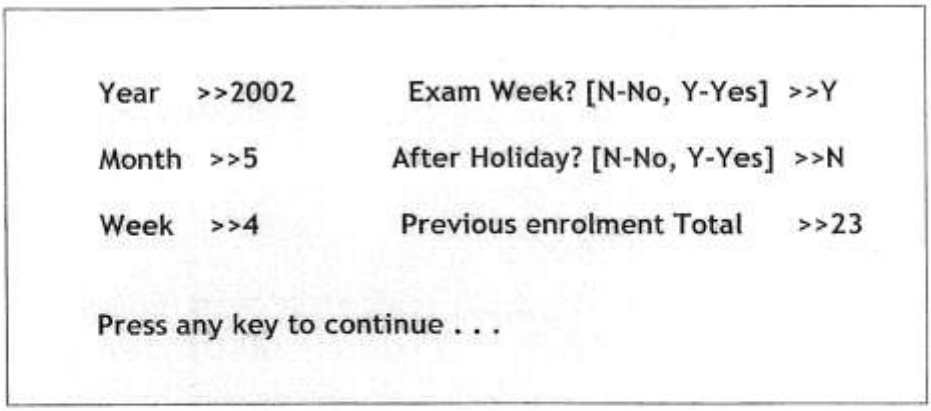
9. After entering the ***Learning Rate,*** the system starts to learn and prompts to wait. During learning, the system adjusts some parameters so as to come up with the optimum after learning from your data. The system gives you the output and gives you the minimum error rate computed. It then prompts to "press any key to continue"

10. The system prompts you now to enter the parameters which you want to use for forecasting. This is as shown in the screen below:

WELCOME TO TESTING MQBWLE ...

Enter the **year** > >

11. After entering all parameters; the following screen appears:



This confirms the parameters entered. Note,

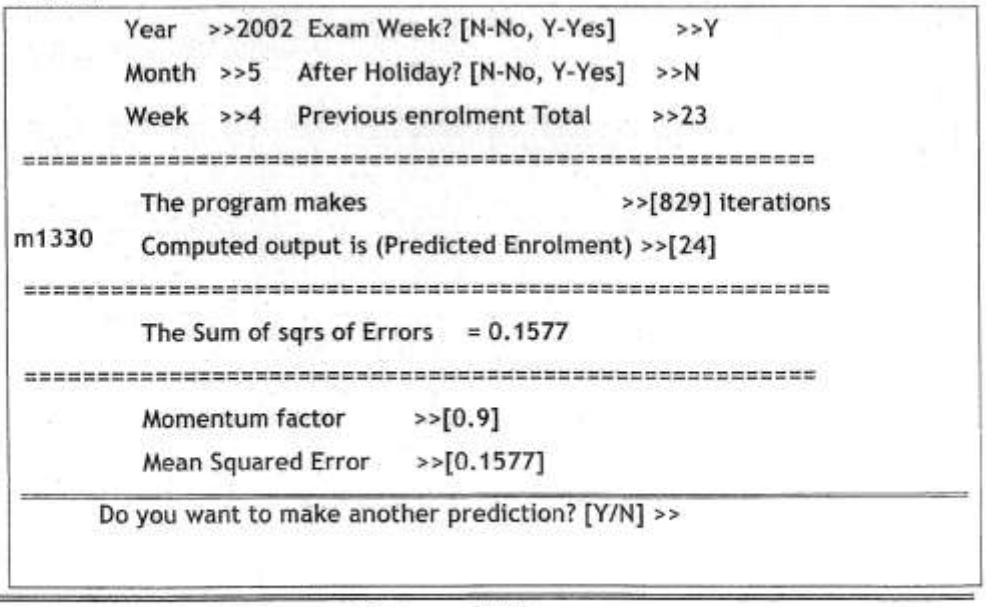
Year denotes the year you want to make the prediction;

Month Is the exact month you want to make the prediction on;

Week Is the week of the month that the prediction is to be made

Previous Enrolment Is the total number of students enrolled in the previous week

12. Below is the window that appears showing the prediction from the parameters you entered:



Page 71/87

11.0 APPENDIX F: SAMPLE PROGRAM SOURCE CODE

```
//Program Name : CEFS - College Enrolment Forecasting System
//Purpose      : A Backpropagation Neural Network for Time Series Forecasting
//Author       : Kandiri, Mugo John (P56/P/7834/2000)
//Last Modified : 14th September , 2003
```

```
//Program     :Password2.h
//Called by   :Password2.cpp, Main.cpp
//Purpose     :Header File for Password

#ifndef PASS_FILE
#define PASS_FILE
#include <iostream>
#include <conio.h>
#include <stdlib.h>
//#include <fstream>
#include <string>
#include <time.h>

using namespace std;
class pass
{
public:
        void AccessWord();
};

#endif

//-------------------------------------------------------------------------------------------
//Program Name :password2.cpp
//Calls        :password2.h
//Purpose      :Implementation for password function
//Called by    :None

//#include "password2.h"

#include <iostream>
#include <conio.h>
#include <stdlib.h>
#include <fstream>
#include <string>
#include <time.h>

using namespace std;
class pass
{
public:
```

```
        void AccessWord();
};

bool PSChk()
{
                char ch;
                string password="PASSWORD";
                cout<<endl<<"\t\tEnter your password >>";
                string temp;
                int count=0;
                while ((ch = getch())!= 13) // end do-while
                {
                                cout <<"*" ;
                }
                if (password == temp) cout <<endl<< "\t\t"; else cout <<"\t\tFalse";
                cout <<endl;
                return (password == temp);
}
bool PSChk2()
{
                char ch2;
                string username="USERNAME";

                string temp2;
                int count=0;
                cout<<endl<<endl<<"\t\tSystem Authentication code ...Welcome ... \n\n";
                cout<<endl<<"\t\tEnter your username >>";
                while ((ch2 = getche())!= 13) // end do-while
                {
                        temp2 +=toupper(ch2) ;
                }
                if (username == temp2) cout <<endl<<"\t\t"; else cout <<"\t\tFalse";
                cout <<endl;
        return (username == temp2);
}
void pass::AccessWord()
{
        int Retry=0;
        do
        {
                Retry++;
                if (PSChk2() && PSChk() )
        {
                cout <<endl<<endl<< "\t\t Code Correct!...Welcome >> \n\n" ; // end if
           break;
        }
        cout<<"\n\n"; system("cls");cout<<endl<<"\t>>Wrong username or password, you have ["<<(3- Retry)<<"]
attempts"<<endl;
        if (Retry == 3)
        {
        cout<<endl <<"\tAccess Denied ....<<Bye>>\n"<<endl;
```

```
        cout<<"\t\t";exit(1);
        }

}while(Retry < 3);

}

//-------------------------------------------------------------------------------------------------
// Program Name         :BasicRowOrCol.h
//Calls                 :None
//Called By             : BasicRowOrCol.cpp,

#ifndef BRD
#define BRD
#include <fstream>
#include <iostream>

using namespace std;

//-------------------------------------------------------------------------------------------------
class RowOrColNumbers
{
        int _DataRows,_Cols;//_TestDataRows,

public:
        RowOrColNumbers();
        void RecordCount(ifstream *MyTrainData);//Name of text file is passed
                                                        from main() function
        int GetRows();                  //Name of text file is passed from main()
                                                function
        int GetCols();
};
//-------------------------------------------------------------------------------------------------
class BasicRowOrCol
{
public:
        float Year; //This this field holds data on year
        float Month; //This field holds data on current month
        float Week; //This field holds data on current week
        float ExamWeek; //This field holds on whether the week is an exam week
        float AfterHoliday; //This field holds on whether the week is after a holiday
        float Total;
        float BiasWeigh;
};
#endif
```

```
//-------------------------------------------------------------------------------------------------
//Program Name : BasicRowOrCol.cpp
//Calls                 :BasicRowOrCol.h
//Called By             :None

#include "BasicRowOrCol.h"
#include <list>
#include <string>

//-------------------------------------------------------------------------------------------------
using namespace std;
#include <stdlib.h>
#include <conio.h>

//-------------------------------------------------------------------------------------------------
RowOrColNumbers::RowOrColNumbers()
{
        {
                char NodeChoice;
                cout<<endl<<"\tPlease select the desired number of nodes for the network: ";
                        cout << endl << "\t\ta. 95" << endl;
                        cout << "\t\tb. 100" << endl;
                        cout << "\t\tc. 105" << endl;
                        cout << "\t\td. 110" << endl;
                        cout << "\t\te. 120" << endl;
                        cout << "\t\tf. 130" << endl;
                        cout << "\t\tg. exit" << endl << endl;
                        cout << "\t\tEnter your choice (a-g)";
                do
                {
                        NodeChoice = getch();
                        NodeChoice=tolower(NodeChoice);
                } while (NodeChoice != 'a' && NodeChoice != 'b' && NodeChoice != 'c' && NodeChoice != 'd' &&
                NodeChoice != 'e' && NodeChoice != 'f' && NodeChoice != 'g');
                switch(NodeChoice)
                {
                        case 'a':_Cols=100;     break;
                        case 'b':_Cols=100;     break;
                        case 'c':_Cols=105; break;
                        case 'd':_Cols=110; break;
                        case 'e':_Cols=120; break;
                        case 'f':_Cols=130; break;
                        case 'g':cout<<endl<<endl<<"\t\t<<Program execution aborted... Bye >>\n\n";
                        cout<<"\t\t";exit(1);break;
                };
        }
}
//-------------------------------------------------------------------------------------------------
void RowOrColNumbers::RecordCount(ifstream *MyTrainData) //The name of the text file is passed here from
main() fumction.
//This becomes the source file where number of rows are counted by this function
```

```
{
                //fstream MyTrainData("RESEARCHDATAtraining.txt");
                string line;
                _DataRows=0;
                while(getline(*MyTrainData,line))
                {
                        _DataRows++;
                }
                cout<<endl<< " \t\tNodes in your network >>"<<_Cols<<" and Total records  >>"<<_DataRows<<endl;
                cout<<"\n\n"<<"\t\t"; system("pause");
        }
int RowOrColNumbers::GetRows()
{
        return _DataRows;
}

//-------------------------------------------------------------------------------------------------------
int RowOrColNumbers::GetCols()
{
        return _Cols;
}

//-------------------------------------------------------------------------------------------------------
//Program Name : ResData.h
//Purpose               :Header File for ResData.cpp
//Calls                 :BasicRowOrCol.h
//Called by             :ResData.cpp, Main.cpp

#ifndef MYDATA
#define MYDATA
//-------------------------------------------------------------------------------------------------------
//#include <fstream.h>
#include "BasicRowOrCol.h"
//-------------------------------------------------------------------------------------------------------
class ResData
{
private:
        short _CurrCol;
        int _CurrRow;
        BasicRowOrCol *_Rows;
public:
        ResData(int RecRows,ifstream &DataFile);
        ~ResData();
        BasicRowOrCol DataRow(ifstream DataFile);
        void SetCurrRow(int);
        BasicRowOrCol* GetRows(ifstream* DataFile) {return _Rows;}
};
#endif

//-------------------------------------------------------------------------------------------------------
//Program Name : ResData.cpp
```

```
//Purpose                 :Reads data from text file for the neural network
//Calls                   :ResDataI.h
//Called by               :None
//-------------------------------------------------------------------------------------------

#include "ResData.h"
//-------------------------------------------------------------------------------------------
ResData::ResData(int RecRows, ifstream &DataFile)// Name of text file whichat first is
RESEARCHDATAtraining.txt for training and then
{
        _CurrRow = _CurrCol = 0;
        _Rows = new BasicRowOrCol[RecRows];
        cout<<"\n\n"<<"\t\tNumber of rows in the source data "<<RecRows<<endl;
        for (int aRow = 0; aRow < RecRows; aRow++)
        {
                DataFile >> _Rows[aRow].Year;
                DataFile >> _Rows[aRow].Month;
                DataFile >> _Rows[aRow].Week;
                DataFile >> _Rows[aRow].ExamWeek;
                DataFile >> _Rows[aRow].AfterHoliday;
                DataFile >> _Rows[aRow].Total;
        }

}

//-------------------------------------------------------------------------------------------
ResData::~ResData()
{
        delete _Rows;
}
//-------------------------------------------------------------------------------------------
BasicRowOrCol  ResData::DataRow(ifstream neurodata)
{
        return _Rows[_CurrRow];
}
//-------------------------------------------------------------------------------------------
void  ResData::SetCurrRow(int x)
{
        _CurrRow = x;
}
//-------------------------------------------------------------------------------------------
#ifndef RAND
#define RAND
```

```
//-------------------------------------------------------------------------------------------------
//Program Name :RandWeights.h
//Purpose                :Header file for RandWeights.cpp
//Calls                  : BasicRowOrCol.h
//Called by              : RandWeights.cpp, Main.cpp

#include "BasicRowOrCol.h"
#include <fstream>
#include <iomanip>

using namespace std;
//-------------------------------------------------------------------------------------------------
class WeiData
{
private:
        short _CurrRow;
        int _CurrCol;
        BasicRowOrCol* _Cols;//using a pointer why? number of columns not constant
        float* _MWeights;
public:
        WeiData(int);
        ~WeiData();
        BasicRowOrCol  WeightCol();
        void  SetCurrCol(int);
        BasicRowOrCol* GetCols() {return _Cols;}
        void RandWeights(int);
        void SaveWeights(int, double);
};
#endif
//-------------------------------------------------------------------------------------------------

//Program Name : RandWeights.cpp
//Purpose                :Randomly generates data for the weights and feeds to text file
//Calls                  : RandWeights.h
//Called by              :None

#include "Propagator.h"
#include "RandWeights.h"
//-------------------------------------------------------------------------------------------------
WeiData::WeiData(int TheCols)
{
        _CurrRow = _CurrCol = 0;
        _Cols = new BasicRowOrCol[TheCols];
        fstream RandWeights("RandWeights.txt");
        for (int aCol = 0; aCol < TheCols; aCol++)
        {
                RandWeights >> _Cols[aCol].Year;
                RandWeights >> _Cols[aCol].Month;
                RandWeights >> _Cols[aCol].Week;
                RandWeights >> _Cols[aCol].ExamWeek;
                RandWeights >> _Cols[aCol].AfterHoliday;
```

```
                RandWeights >> _Cols[aCol].Total;
                RandWeights >> _Cols[aCol].BiasWeigh;
        }
}
//---------------------------------------------------------------------------------------------
WeiData::~WeiData()
{
        delete _Cols;//deallocates the memory allocated in the constructor
}
//---------------------------------------------------------------------------------------------BasicRowOrCol
WeiData::WeightCol()
{
        return _Cols[_CurrCol];
}
//---------------------------------------------------------------------------------------------void
WeiData::SetCurrCol(int x)
{
        _CurrCol = x;
}
//---------------------------------------------------------------------------------------------void
WeiData::RandWeights(int TheCols)
{
        cout<<"\n\n"<<"\t\t<<Saving Weights to text file >>\n\n ";
   _Cols = new BasicRowOrCol[TheCols];//allocates/reserving memory space
        fstream RandomWeights("RandWeights.txt");

        for(int aCol = 0; aCol < TheCols; aCol++)
        {
                srand((unsigned)(time(NULL)));
                _Cols[aCol].Year = fabs(float(rand())/(32767/2)-1);//Function RandNumber is called from
                Propagator.cpp where it is generated
                _Cols[aCol].Month = fabs(float(rand())/(32767/2)-1);
                _Cols[aCol].Week = fabs(float(rand())/(32767/2)-1);
                _Cols[aCol].ExamWeek = fabs(float(rand())/(32767/2)-1);
                _Cols[aCol].AfterHoliday = fabs(float(rand())/(32767/2)-1);
                _Cols[aCol].Total = fabs(float(rand())/(32767/2)-1);
                _Cols[aCol].BiasWeigh = fabs(float(rand())/(32767/2)-1);

        }

        cout<<"\t\tFile created.  Number of Nodes >> "<<TheCols<<endl;
        cout<<"\n\n"<<"\t\t"; system("pause");
        for (aCol = 0; aCol < TheCols; aCol++)
        {
                RandomWeights<<setprecision(3)<< _Cols[aCol].Year<<"\t";
                RandomWeights<<setprecision(3)<< _Cols[aCol].Month<<"\t";
                RandomWeights<<setprecision(3)<< _Cols[aCol].Week<<"\t";
                RandomWeights<<setprecision(3)<< _Cols[aCol].ExamWeek<<"\t";
                RandomWeights<<setprecision(3)<< _Cols[aCol].AfterHoliday<<"\t";
                RandomWeights<<setprecision(3)<< _Cols[aCol].Total<<"\t";
                RandomWeights<<setprecision(3)<< _Cols[aCol].BiasWeigh<<endl;
```

```
        }
}

/*In this function, the normalised weights realised during
learning are saved to be used in the testing set*/

void WeiData::SaveWeights(int TheCols, double ErrorByMoment)
{
        cout<<"\t\t<<Saving Weights to text file >>\n\n ";
        fstream RandomWeights("RandWeights.txt");
        cout<<"\t\tFile created \n";
                for (int aCol = 0; aCol < TheCols; aCol++)
        {
                RandomWeights<<setprecision(3)<< _Cols[aCol].Year+ErrorByMoment<<"\t";
                RandomWeights<<setprecision(3)<< _Cols[aCol].Month+ErrorByMoment<<"\t";
                RandomWeights<<setprecision(3)<< _Cols[aCol].Week+ErrorByMoment<<"\t";
                RandomWeights<<setprecision(3)<< _Cols[aCol].ExamWeek+ErrorByMoment<<"\t";
                RandomWeights<<setprecision(3)<< _Cols[aCol].AfterHoliday+ErrorByMoment<<"\t";
                RandomWeights<<setprecision(3)<< _Cols[aCol].Total+ErrorByMoment<<"\t";
                RandomWeights<<setprecision(3)<< _Cols[aCol].BiasWeigh+ErrorByMoment<<endl;


        }
}
//-------------------------------------------------------------------------------------------
//Program Name : Propagator.h
//Purpose                :Header File for Backpropagation Code
//Calls                  : RandWeights.h, ResData.h
//Called by              : Propagator.cpp, Main.cpp
//-------------------------------------------------------------------------------------------
#include "ResData.h"
#include "RandWeights.h"
//#include <stdlib.h>
#include <time.h>
#include <iomanip>
#include <math.h>
using namespace std;
#include <conio.h>
//-------------------------------------------------------------------------------------------
class Propagatte
{
                double MeanTrainError;
                double MeanValError;
                double MinError;
                double MeanTestError;
public:
        Propagate();
        ~Propagate();
        double Sigmoid(float num);
```

```
        double Propagatting(int TrainDataRows, BasicRowOrCol* Data, int WeightCols, int Cols, double,
        Initial, double End,int loop);
        double Testing(int TestDataRows, BasicRowOrCol* Data, int WeightCols, int Cols, double, int);
        double GetMeanTrainError();
        double SetMinError();
        double GetMeanTestError();
};
//-------------------------------------------------------------------------------------------//Program Name
Propagator.cpp
//Purpose                 :Backpropagation Code
//Calls                   : Propagator.h
//Called by               :None
//-------------------------------------------------------------------------------------------

#include "Propagator.h"

//-------------------------------------------------------------------------------------------
Propagate::Propagate()
{
        double MeanError=0;

        {
                char ErrChoice;
                cout<<endl<<"\t\tPlease select the desired minimum error: ";
                cout <<endl<<"\t\ta. 0.15" << endl;
                cout << "\t\tb. 0.20" << endl;
                cout << "\t\tc. 0.25" << endl;
                cout << "\t\td. 0.30" << endl;
                cout << "\t\te. 0.35" << endl;
                cout << "\t\tf. exit" << endl<<endl;


                do
                {

                                cout<<"\t\t"; ErrChoice = getch();
                                ErrChoice=tolower(ErrChoice);
                } while (ErrChoice != 'a' && ErrChoice != 'b' && ErrChoice != 'c' && ErrChoice != 'd' &&
ErrChoice != 'e'&& ErrChoice != 'f' );
                switch(ErrChoice)
                {
                        case 'a': MinError=0.20;break;
                        case 'b': MinError=0.20;break;
                        case 'c': MinError=0.25;break;
                        case 'd': MinError=0.30;break;
                        case 'e': MinError=0.35;break;
                case 'f': cout<<endl<<endl<<"\t\tProgram Execution Aborted ... <<Bye>>"; exit(1); break;
                };

                cout<<"\t\tMinimum Error >>:"<<MinError<<endl;
        }
```

```
}
//-------------------------------------------------------------------------------------------------
Propagate::~Propagate()
{
}
//-------------------------------------------------------------------------------------------------float RandNumber()
//This function provides random data generation
{
        srand((unsigned)(time(NULL)));
        return fabs(float(rand())/(32767/2)-1);//You can put te fabs function  before float to make all data
        +ve
}
//-------------------------------------------------------------------------------------------------
double Propagate::Sigmoid(float num)
{
        return  1.0/(1.0 + exp(-num));

}
//-------------------------------------------------------------------------------------------------
double Propagate::SetMinError()
{
        return MinError;
}
//-------------------------------------------------------------------------------------------------
double Propagate::Propagatting(int DataRows, BasicRowOrCol* Data, int WeightCols,
                                int Cols, double ErrorByMoment, double Initial=0, double End=0.7, int
                                loop=1 )
{
        WeiData TheCols(Cols);
        BasicRowOrCol* Weights = TheCols.GetCols();
        cout.precision(4);
        cout<<"\t\tPlease wait as program learns\n";
        {
        double SqrTrainErrors = 0;
        for (int CurrRow = Initial; CurrRow <int(End* DataRows); CurrRow++)
        {
                cout.precision(4);
                double TrainError = 0;
                double DataByWeightSum = 0;
                double PrevDataTotal;
                if (CurrRow) PrevDataTotal = Data[CurrRow-1].Total;
                        else PrevDataTotal = 0;
                for (int CurrCol = 0; CurrCol < WeightCols; CurrCol++)
                {
                        cout.precision(4);
                        DataByWeightSum = Sigmoid(Data[CurrRow].Year-1996) * (Weights[CurrCol].Year +
                        ErrorByMoment);
                        DataByWeightSum +=Sigmoid(Data[CurrRow].Month) * (Weights[CurrCol].Month +
                        ErrorByMoment);
                        DataByWeightSum +=Sigmoid(Data[CurrRow].Week)  * (Weights[CurrCol].Week +
                        ErrorByMoment);
```

```
                    DataByWeightSum +=Sigmoid(Data[CurrRow].ExamWeek) *
                    (Weights[CurrCol].ExamWeek + ErrorByMoment);
                    DataByWeightSum +=Sigmoid(Data[CurrRow].AfterHoliday) *
                    (Weights[CurrCol].AfterHoliday + ErrorByMoment);
                    DataByWeightSum +=Sigmoid(PrevDataTotal) * (Weights[CurrCol].Total +
                    ErrorByMoment);
                    DataByWeightSum=Sigmoid(DataByWeightSum);
            }

        TrainError = Sigmoid(Data[CurrRow].Total) - (DataByWeightSum) *(Weights[6].BiasWeigh +
        ErrorByMoment);
        SqrTrainErrors += (TrainError * TrainError);
        cout<<setprecision(6)<<"Row = "<<CurrRow+1<<"compute "<<Sigmoid(DataByWeightSum)
        *(Weights[6].BiasWeigh + ErrorByMoment)<<" ErrorByMoment
        }

        MeanTrainError=SqrTrainErrors/WeightCols;
        return MeanTrainError;
}
}
//-------------------------------------------------------------------------------------------double
Propagate::GetMeanTrainError()
{
        return MeanTrainError;
}
//-------------------------------------------------------------------------------------------double
Propagate::Testing(int TestDataRows, BasicRowOrCol* Data, int WeightCols,
                                int Cols, double ErrorByMoment, int counter )
{
        WeiData TheCols(Cols);
        BasicRowOrCol* Weights = TheCols.GetCols();
        cout.precision(4);
        cout<<"\n\n"<<"\t\t"; system("cls");
        cout<<"\n\n=============================================================
        cout<<"\n\n"<<"\t\tWELCOME TO TESTING MODULE ...\n\n";

        int AboveCount=0;
        int TestRow;
        do
        {
                cout<<endl;
                cout<<"\t\tPlease enter the record to test >>";
                cin>>TestRow;
                if(TestRow>TestDataRows)
                        cout<<"\t\tNo such record exists .. Please enter a valid number\n";
        }while(TestRow>TestDataRows);



        {



        {
```

```
double SqrTestErrors = 0;

for (int CurrRow = TestRow-1; CurrRow < TestRow; CurrRow++)
{
        cout.precision(4);
        double TestError = 0;
        double DataByWeightSum = 0;
        double BiasWeight=RandNumber();
        double PrevDataTotal;
        if (CurrRow) PrevDataTotal = Data[CurrRow-1].Total;
                else PrevDataTotal = 0;

                for (int CurrCol = 0; CurrCol < WeightCols; CurrCol++)
                {
                        cout.precision(4);
                        DataByWeightSum = (Data[CurrRow].Year-1996) * (Weights[CurrCol].Year);
                        DataByWeightSum +=(Data[CurrRow].Month) * (Weights[CurrCol].Month);
                        DataByWeightSum +=(Data[CurrRow].Week)  * (Weights[CurrCol].Week);
                        DataByWeightSum +=(Data[CurrRow].ExamWeek) *
                        (Weights[CurrCol].ExamWeek);
                        DataByWeightSum +=(Data[CurrRow].AfterHoliday) *
                        (Weights[CurrCol].AfterHoliday);
                        DataByWeightSum +=(PrevDataTotal) * (Weights[CurrCol].Total);

                }

        TestError = (Data[TestRow-1].Total) - (DataByWeightSum *(Weights[6].BiasWeigh));

        cout<<" "<<"\t\t";
        SqrTestErrors += (TestError * TestError);

                        double PercentError;


PercentError=fabs((Data[TestRow-1].Total-(DataByWeightSum
*Weights[6].BiasWeigh))/(DataByWeightSum *Weights[6].BiasWeigh))*100;

cout<<"\n================================================================\n\n";
cout<<endl<<"\t\tThe program makes "<<"\t>>["<<counter <<"] epochs "<<endl<<endl;
cout<<"\t\tReal Output is "<<"\t\t>>["<<Data[TestRow-1].Total<<"]"<<endl<<endl;
cout<<"\t\tComputed output is " <<"\t>>["<<DataByWeightSum *Weights[6].BiasWeigh<<"]"<<endl<<endl;
cout<<"\t\tThe absolute Error is " <<"\t>>["<<fabs(Data[TestRow-1].Total - DataByWeightSum
*Weights[6].BiasWeigh)<<"]"<<endl<<endl;
cout<<"\t\tThe Percentage error is "<<">>["<<PercentError<<"%]"<< endl<<endl;

cout<<"================================================================\n\n";
cout<<"\t\t"; system("pause");

fstream FinalTestResult("TestOutPut.txt",ios::app);
```

```
        //cout<<"\t\tFile <<FinalTestResult>> created \n";
        FinalTestResult<<setprecision(5)<<TestRow<<"\t"<<Data[TestRow-1].Total<<"\t"<<DataByWeightSum
        *Weights[6].BiasWeigh<<"\t"<<PercentError<<"\t"<<endl;
}

MeanTestError=SqrTestErrors/WeightCols;
return MeanTestError;
}
}

double Propagate::GetMeanTestError()
{
        return MeanTestError;
}
//-------------------------------------------------------------------------------------------------
//Program Name :Main.cpp
//Purpose                 : Contains the Main Function
//Calls                   : Propagator.h , RandWeights.h, ResData.h,Password2.h
//Called by               :None

#include "Propagator.h"
#include "RandWeights.h"
#include "ResData.h"
#include "Password2.h"
#include <conio.h>

void main()
{
//-------------------------------------------------------------------------------------------------
cout<<endl<<endl<<"\t\t<<  CEFS (COLLEGE ENROLMENT FORECASTNG SYSTEM) >>\n\n";
        cout<<"\t  <<  THIS IS A TIME SERIES FORECASTING NEURAL NETWORK  >>\n\n";

        pass ps;//Instantiates the constructor for the password function
        ps.AccessWord();//Calls the password function


        RowOrColNumbers TheRowsCols;//instantiating RowOrColNumbers therefore calls the default
        constructor

        WeiData Weigh(TheRowsCols.GetCols());

        Propagate Learn;
        cout.precision(4);//Allows data to 4 decimal places

        double LearningRate;
        double Momentum;
        {
                char LearnRateOption;
                cout<<endl<<"\t\tPlease select the desired LEARNING RATE: ";
                cout <<endl<<"\t\ta. 0.05" << endl;
                cout << "\t\tb. 0.10" << endl;
```

```
            cout << "\t\tc. 0.20" << endl;
            cout << "\t\td. 0.25" << endl;
            cout << "\t\te. 0.30" << endl;
            cout << "\t\tf. exit" << endl << endl;
            cout << "\t\tEnter your choice (a-f)"<<endl;
            do
            {
                    cout<<"\t\t\t"; LearnRateOption = getch();
                    LearnRateOption=tolower(LearnRateOption);
            } while (LearnRateOption != 'a' && LearnRateOption != 'b' && LearnRateOption != 'c' &&
            LearnRateOption != 'd' && LearnRateOption != 'e'&& LearnRateOption != 'f' );
            switch(LearnRateOption)
            {
                    case 'a': LearningRate=0.20;break;
                    case 'b': LearningRate=0.20;break;
                    case 'c': LearningRate=0.20;break;
                    case 'd': LearningRate=0.25;break;
                    case 'e': LearningRate=0.30;break;
                    case 'f': cout<<endl<<endl<<"\t\tProgram Execution Aborted ... <<Bye>>"; exit(1); break;
            };

    }

    Momentum=0.9;
    cout<<"\n\n"<<"\t\t"; system("cls");
    cout<<endl<<"\t\tSystem learns now ...";

    ifstream neurodata("RESEARCHDATA.txt",ios::in |ios::out);

    TheRowsCols.RecordCount(&neurodata2);

    Weigh.RandWeights(TheRowsCols.GetCols());//The random weights file is created
    double ErrorByMoment=0;

    GlobalRows=TheRowsCols.GetRows();

    ResData* TheRows = new ResData(GlobalRows,
    neurodata);//TheRowsCols.GetRows(&neurodata,GlobalRows),neurodata);//creates a anonymous object
    of type ResData and makes TheRows point to it

    WeiData* TheCols = new WeiData(TheRowsCols.GetCols());//Here you are passing a member of
    TheRowsCols object to the constructor of WeiData

    Learn.Propagatting(GlobalRows,TheRows>GetRows(&neurodata),TheRowsCols.GetCols(),TheRowsCols.Get
Cols(),ErrorByMoment,0,0.7,0 );

    int counter;
    counter=0;//Counts the number of iterations the program makes for a given number of neurons

    for (int x=0;x<10000;x++)
    {
```

```
            ++counter;


            if(Learn.GetMeanTrainError()>Learn.SetMinError())
            {
            ErrorByMoment=Learn.GetMeanTrainError()*LearningRate* Momentum;

            Learn.Propagatting(GlobalRows,TheRows>GetRows(&neurodata),TheRowsCols.GetCols(),TheRow
      sCols.GetCols(),ErrorByMoment,0,0.7,counter);//Main is passing the values inside the bracket to
      propagatting function
            }
            else
            {
            cout<<"\t\tWait ... Validation begins... "<<endl<<endl<<endl;
            Weigh.SaveWeights(TheRowsCols.GetCols(),ErrorByMoment);//Saves the weights to be used
            in testing

      Learn.Propagatting(GlobalRows,TheRows>GetRows(&neurodata),TheRowsCols.GetCols(),TheRowsCols.Get
Cols(),ErrorByMoment,0.7,1,counter);//Main is passing the values inside the bracket to propagatting function
                        {
                              ifstream testdata("RESEARCHDATAtesting.txt",ios::in |ios::out);

                              ifstream testdata2("RESEARCHDATAtesting.txt",ios::in |ios::out);


                              TheRowsCols.RecordCount(&testdata2);
                              GlobalRows=TheRowsCols.GetRows();

                              ResData* TheRows = new ResData(GlobalRows,testdata);//creates a
                              anonymous object of type ResData and makes TheRows point to it

                              Learn.Testing(GlobalRows, TheRows-
                              >GetRows(&testdata),TheRowsCols.GetCols(),TheRowsCols.GetCols(),ErrorBy
                              Moment,counter);//Main is passing the values inside the bracket to
                              propagatting function


      cout<<"\n\n"<<"================================================="<<endl<<endl;
            cout<<endl<<"\t\tMomentum factor "<<"\t>>["<<Momentum<<"] "<<endl;
            cout<<endl<<"\t\tMean Squared Error  "<<"\t>>["<<Learn.GetMeanTrainError()<<"]"<<endl;

            fstream FinalData("TrainVsValError.xls",ios::app);
                  cout<<"\n\n"<<"================================================="<<endl;
                        cout<<"\t\tWant to make another prediction? [y/n] >>";
                        char Ans;
                        cin>>Ans;
                        Ans=toupper(Ans);

                        if(Ans!='Y')
                              return;
      }}}}//-------------------------------------------------------------------------------------------------
```