NAVAL
POSTGRADUATE
SCHOOL

MONTEREY, CALIFORNIA

THESIS

APPLICATION OF NEURAL NETWORKS TO PREDICT UH-60L ELECTRICAL GENERATOR CONDITION USING (IMD-HUMS) DATA

by

Evangelos Tourvalis

December 2006

Thesis Advisor:	Lyn R. Whitaker
Second Reader:	Samuel E. Buttrey

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2006	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Application of Neural Networkss to Predict UH-60L Electrical Generator Condition using (IMD-HUMS) data			**5. FUNDING NUMBERS**
6. AUTHOR(S) Tourvalis Evangelos			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

In 2003, the US Army began using the Integrated Mechanical Diagnostics Health and Usage Management System (IMD-HUMS), an integrated airborne and ground-based system developed by Goodrich Corporation, to support maintenance of the UH-60L. IMD-HUMS is responsible for collecting, processing, analyzing, and storing an enormous amount of vibratory and flight regime data obtained from sensors located throughout the aircraft.

The purpose of this research is to predict failures of the UH-60L's electrical generators, applying Artificial Neural Networks (ANN) on the IMD-HUMS-produced data. Artificial NNs are data based vice rule based, thereby possessing the potential capability to operate where analytical solutions are inadequate. They are reputed to be robust and highly tolerant of noisy data. Software tools such as Clementine 10.0, S-Plus 7.0, and Excel are used to establish these predictions.

This research has verified that ANNs have a position in machinery condition monitoring and diagnostics. However, the limited nature of these results indicates that ANNs will not solve all machinery condition monitoring and diagnostics problems by themselves. They certainly will not completely replace conventional rule-based expert systems. Ultimately, it is anticipated that a symbiotic combination of these two technologies will provide the optimal solution to the machinery condition monitoring and diagnostics problem.

14. SUBJECT TERMS Condition Based Maintenance, IMD-HUMS, ANNs, Backpropagation, Learning Process			**15. NUMBER OF PAGES** 99
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

APPLICATION OF NEURAL NETWORKS TO PREDICT UH-60L ELECTRICAL GENERATOR CONDITION USING (IMD-HUMS) DATA

Evangelos Tourvalis
Major, Hellenic Air Force

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
December 2006

Author:	Evangelos Tourvalis
Approved by:	Lyn R. Whitaker Thesis Advisor
	Samuel E. Buttrey Second Reader
	James N. Eagle Chairman, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

In 2003, the US Army began using the Integrated Mechanical Diagnostics Health and Usage Management System (IMD-HUMS), an integrated airborne and ground-based system developed by Goodrich Corporation, to support maintenance of the UH-60L. IMD-HUMS is responsible for collecting, processing, analyzing, and storing an enormous amount of vibratory and flight regime data obtained from sensors located throughout the aircraft.

The purpose of this research is to predict failures of the UH-60L's electrical generators, applying Artificial Neural Networks (ANN) on the IMD-HUMS-produced data. Artificial NNs are data based vice rule based, thereby possessing the potential capability to operate where analytical solutions are inadequate. They are reputed to be robust and highly tolerant of noisy data. Software tools such as Clementine 10.0, S-Plus 7.0, and Excel are used to establish these predictions.

This research has verified that ANNs have a position in machinery condition monitoring and diagnostics. However, the limited nature of these results indicates that ANNs will not solve all machinery condition monitoring and diagnostics problems by themselves. They certainly will not completely replace conventional rule-based expert systems. Ultimately, it is anticipated that a symbiotic combination of these two technologies will provide the optimal solution to the machinery condition monitoring and diagnostics problem.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ANN(s)	Artificial Neural Network(s)
BP	Back Propagation
BL	Batch Learning
CBM	Condition Based Maintenance
GSS	Ground Station System
FFN	Feed Forward Networks
IL	Incremental Learning
IMD-HUMS	Integrated Mechanical Diagnostics Health and Usage Maintenance System
LMS	Least-Mean-Squares
MLN	Multi Layer Networks
OBS	On Board System
PM	Predetermined Maintenance
RBFN	Radial Basis Function Networks
SLN	Single Layer Networks

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to acknowledge the help of my excellent thesis advisor, Professor Lyn R. Whitaker, and my second reader, Samuel E. Buttrey, for their direction and assistance as this research was developed.

Also, I would like to mention that this thesis could not have been completed without the presence, support and encouragement of my wife Elena and my son Vasilis.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Readiness is a key factor for military forces to stay effective and reliable in a continuously growing and demanding environment. Increased readiness can be achieved by increasing availability through performing efficient maintenance, performing less corrective maintenance actions, and identifying more accurate preventive maintenance periods. Today, the United States and allied forces spend billions of dollars for time or phased maintenance periods that overlook several facts and realities of operational use. Important savings can be gained by using hardware and software to evaluate component health and the conditions of systems based on operational usage and performing maintenance in relation to statistical and engineering analyses that predict availability and readiness.

Nowadays, the majority of maintenance processes are accomplished by either the predetermined preventive or the corrective approach. The former approach has fixed maintenance intervals; the latter is performed after the fault of the component. Because both approaches are costly, some industries have started to perform maintenance action in a predictive manner, Condition Based Maintenance (CBM), where the condition is the key parameter to set the maintenance intervals and appropriate maintenance tasks.

Condition Based Maintenance (CBM) is a technology weapon that tries hard to recognize initial faults before they develop into critical failures, which permits more precise scheduling of the preventive maintenance. The causes that have motivated a boost in the action of CBM include the need for reduced maintenance and logistics costs, protection against failure of mission-important equipment, and upgraded equipment availability.

In 2003, the US Army began using the Integrated Mechanical Diagnostics Health and Usage Management System (IMD-HUMS), an integrated airborne and ground-based system developed by Goodrich Corporation, to support maintenance of the UH-60L. IMD-HUMS is responsible for collecting, processing, analyzing, and storing an enormous amount of data obtained from sensors located throughout the aircraft. The IMD-HUMS improves aircraft availability for operators by identifying potential

problems early so that maintenance can be performed before it becomes an issue that could impact flight operations. The system also provides operators with accurate flight parameter data, monitored automatically on each flight, allowing them to better schedule routine maintenance and, in some cases, avoid unnecessary early repair and overhaul.

Neural networks are used in numerous fields, including medical diagnostics. In this thesis neural networks are used for machinery diagnostics and specifically for diagnosing the UH-60L helicopter's electrical generator. In order to accomplish this, a database collected from IMD-HUMS is used. The emphasis in this thesis is to develop a neural network that would utilize the collected data from IMD-HUMS, manufactured by Goodrich Corporation, in order to discover patterns that would predict a potential failure of a UH-60L helicopter generator. Many different neural networks are evaluated for their success rate for this faulting diagnosis.

As in any prediction/forecasting model, the selection of appropriate model inputs is extremely important. However, in most ANN Artifiacial Neural Network) applications, less attention is given to this task. The main reason for this is that ANNs belong to the class of data-driven approaches, whereas conventional statistical methods are model driven. In the latter, the structure of the model has to be determined first, which is done with the aid of empirical or analytical approaches, before the unknown model parameters can be estimated. Data-driven approaches, on the other hand, have the ability to determine which model inputs are critical, so there is less need for "...a priori rationalization about relationships between variables..." However, presenting a large number of inputs to ANN models and relying on the network to determine the critical model inputs usually increases network size. This has a number of disadvantages, such as decreasing processing speed, increasing the amount of data required to estimate the connection weights efficiently and degrading performance of the AAN. This is particularly true for complex problems, where the number of potential inputs is large and where no a priori knowledge is available to suggest which inputs to include.

Clementine which is the software used in this research, incorporates several features to avoid some of the common pitfalls of ANNs, including sensitivity analysis, network accuracy, and feedback graph. With these options selected, a sensitivity analysis will provide information on which input fields are most important in predicting the output field, a network accuracy will provide the percentage of records for which the prediction of the model matches the observed value in the data, and the feedback graph will depict the accuracy of the network over time as it learns.

In practice, building an ANN forecasting model involves a lot of trial and error. Consequently, the objective of this thesis is to provide a practical, non-technical introduction to structure an ANN forecasting model using real operating data of UH-60L helicopters. The success of ANN applications for an individual researcher depends on three key factors. First, the researcher must have the time, patience, and resources to experiment. Second, the ANN software must allow automated routines, such as walk-forward testing, optimization of hidden neurons, and testing of input variable combinations—either through direct programming or the use of batch/script files. Third, the researcher must maintain a good set of records that lists all parameters for each network tested.

This research has verified that ANNs have a position in machinery condition monitoring and diagnostics. However, the limited nature of these results indicates that ANNs will not solve all machinery condition monitoring and diagnostics problems by themselves. They certainly will not completely replace conventional rule-based expert systems. Ultimately, it is anticipated that a symbiotic combination of these two technologies will provide the optimal solution to the machinery condition monitoring and diagnostics problem.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Readiness is a key factor in enabling military forces to stay effective and reliable in a continuously growing and demanding environment. Increased readiness can be achieved by increasing availability through performing efficient maintenance, performing fewer corrective maintenance actions, and identifying more accurate preventive maintenance periods. Today, the United States and allied forces spend billions of dollars on time or phased-maintenance approaches that overlook several facts and realities of operational use. Important savings can be gained by using hardware and software to evaluate component health and the conditions of systems based on operational usage and performing maintenance in relation to statistical and engineering analyses that predict availability and readiness.

The emphasis in this thesis is to develop a neural network that utilizes data collected from IMD-HUMS, manufactured by Goodrich Corporation, in order to discover patterns that can predict a failure of a UH-60L helicopter generator. Many different neural networks will be evaluated for their success rate on this faulting diagnosis.

A. CONDITION BASED MAINTENANCE

Maintenance is usually carried out in either time-based scheduled periods (so-called preventive maintenance) or by corrective maintenance. Preventive maintenance aims to avoid system or component failure by performing repair, service, or replacement within the fixed time intervals. On the other hand, corrective maintenance is performed after the failure or when an apparent fault has taken place (Davis, A., 1998). For several types of equipment or systems the maintenance action must be done without delay, but for many others it can be delayed depending on the equipment's function. In many cases the preventive maintenance can be divided into two groups: Condition-Based Maintenance (CBM) and Predetermined Maintenance (PM). PM is scheduled in time, while CBM mostly has dynamic or on-request intervals (Figure 1).

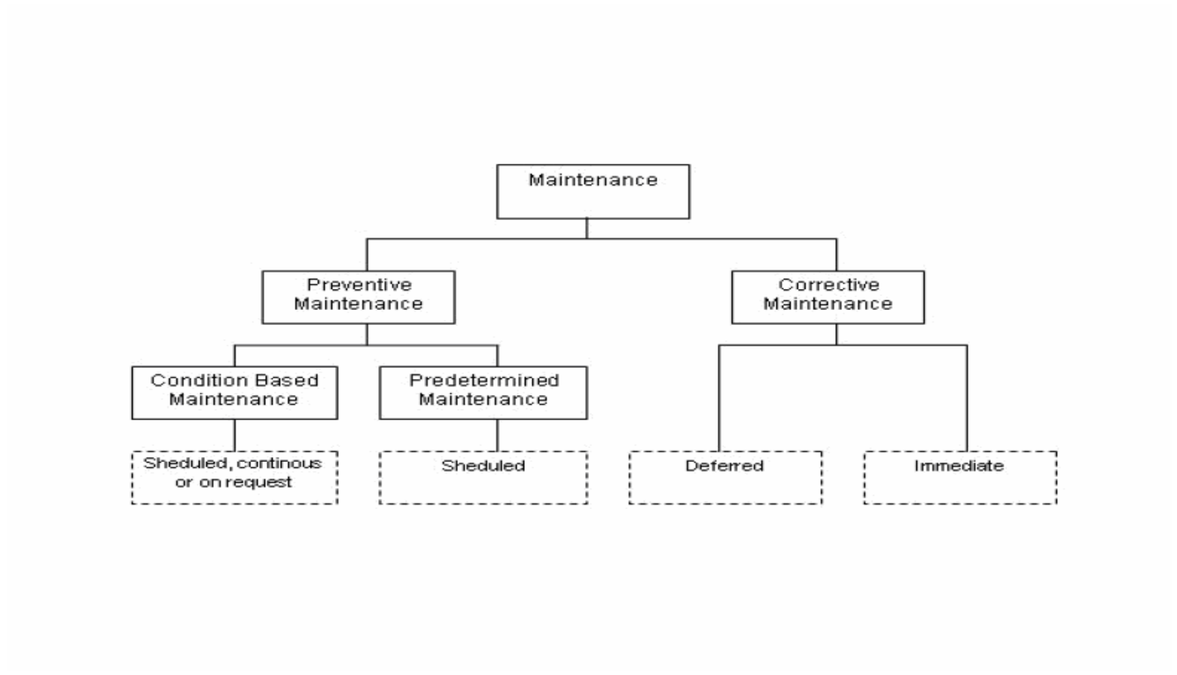

Maintenance
Preventive Maintenance
Corrective Maintenance
Condition Based Maintenance
Predetermined Maintenance
Sheduled, continous or on request
Sheduled
Deferred
Immediate

Figure 1. Overview of Maintenance Terminology

Entire CBM schemes involve a number of efficient capabilities, like sensing and data acquisition, signal processing, condition and health estimation, prognostics, and decision assistance. Moreover, in order for the user to have access to the system, a Human System Interface (HSI) development is necessary. Generally, the integration of various hardware and software components is needed to implement a CBM system.

A complete architecture for CBM systems should cover the range of functions from data collection through the recommendation of specific maintenance actions. The major tasks that assist CBM consist of (http://www.osacbm.org):

- Sensing and data acquisition
- Signal processing and feature extraction
- Production of alarms or alerts
- Fault or failure diagnosis and health evaluation
- Prognostics: projection of health profiles to future health or estimation of remaining useful life
- Decision aiding: maintenance recommendations, or evaluation of asset readiness for a particular operational setting
- Management and control of data flows or test sequences

- Management of historical data storage and historical data access
- System configuration management
- Human system interface.

CBM makes use of information collected on equipment through monitoring devices. As equipment becomes more complex, more manufacturers are providing these monitoring devices to assist companies or organizations handle and maintain their equipment (Tsang, A., 1995). CBM uses this online data to compare equipment conditions to predefined operating thresholds. Data that happen to fall outside these thresholds generates a maintenance alert by the software that signals a problem or area of concern.

B. IMD-HUMS

In 2003, the US Army began using the Integrated Mechanical Diagnostics Health and Usage Management System (IMD-HUMS), an integrated airborne and ground-based system developed by Goodrich Corporation, to support maintenance of the UH-60L. IMD-HUMS is responsible for collecting, processing, analyzing, and storing an enormous amount of data obtained from sensors located throughout the aircraft. The IMD-HUMS improves aircraft availability for operators by identifying potential problems early so that maintenance can be performed before it becomes an issue that could impact flight operations. The system also provides operators with accurate flight parameter data, monitored automatically on each flight, allowing them to better schedule routine maintenance and, in some cases, avoid unnecessary early repair and overhaul. The IMD-HUMS consists of two main subsystems: the On-Board System (OBS) and the Ground Station System (GSS) (System Users Manual For IMD-HUMS, 1995).

1. On-Board System (OBS)

The OBS is comprised of the following components (Figure 2):

- Cockpit display unit (CDU)
- Data transfer unit (DTU)
- Remote data concentrator (RDC)
- Main processor unit (MPU)
- 2 junction boxes (JB1/JB2)
- 20 drive train and gearbox accelerometers

- 4 engine accelerometers
- 5 trim and balance accelerometers
- 1 4g body accelerometer for regime recognition
- Main and tail rotor magnetic RPM sensors
- Main rotor blade tracker
- Engine output shaft optical tachometers.

The heart of the IMD-HUMS OBS is the Main Processing Unit (MPU). The MPU collects the data from the accelerometers, analyzes the inputs, and records the data, seeking for vibration exceedances and events. It calculates time spent in various flight regimes, performs various diagnostic algorithms, and stores the data to an onboard data cartridge. The OBS also provides for crew interaction through a Cockpit Display Unit (CDU) in order to support prompted procedure actions related to power assurance checks, power train analyses, and rotor track and balance data acquisitions. Besides prompted actions, the OBS uses regimes information to automatically store power train and rotor vibration data.

2. Ground Station System (GSS)

The GSS is the major user interface for the IMD-HUMS. It performs after-flight debrief and is designed to analyze, process, and compile flight data into useful information for the maintenance crew, logistics teams, the operations department, and engineering support. The IMD-HUMS GSS functions include:

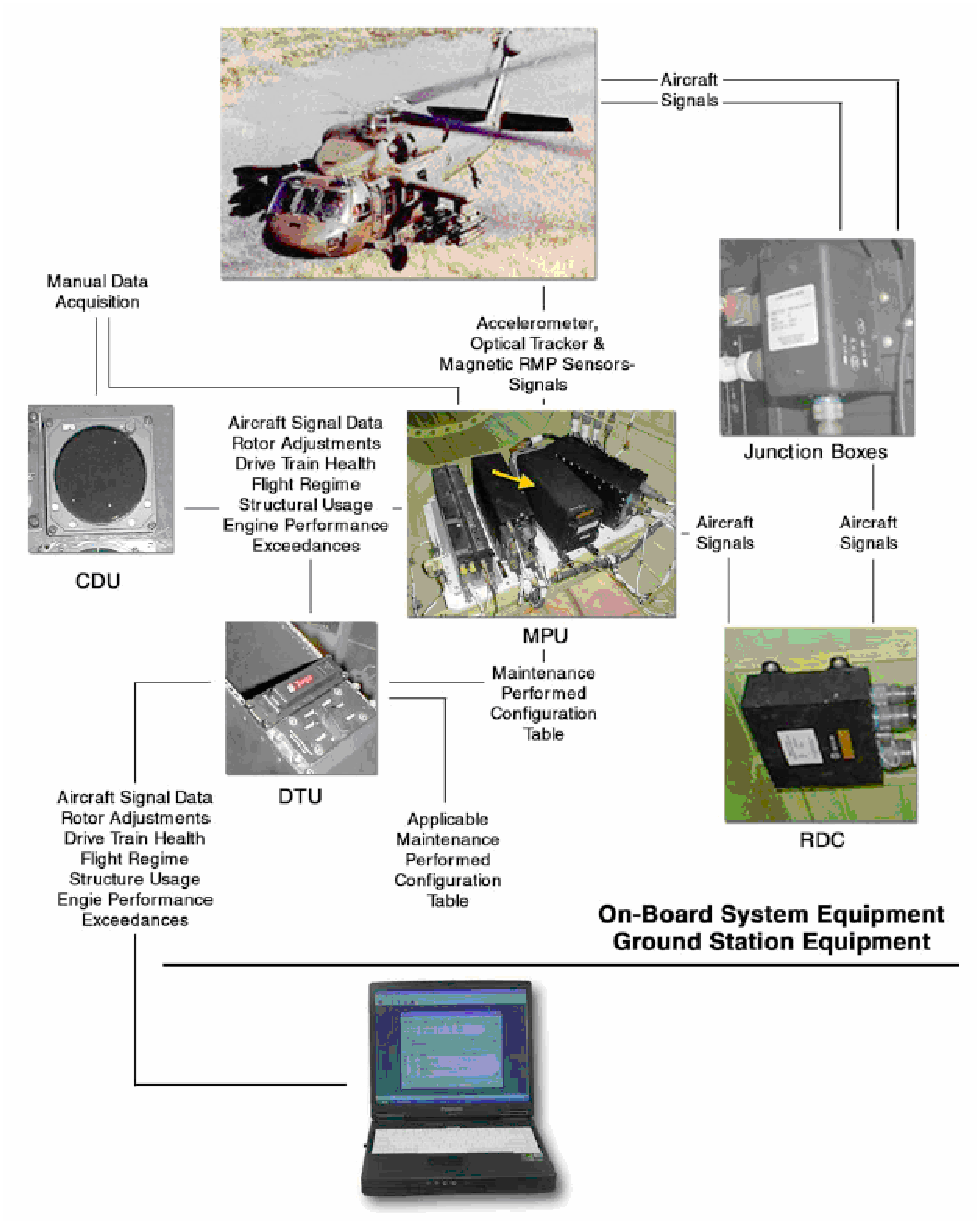


Figure 2. OBS & GSS (From: IMD-HUMS User Manual, 2005)

- Rotor Time and Balance
- Strip Charts of Aircraft Data
- Engine Performance
- Trending
- Usage Computation and Tracking
- Regime Identification and Processing
- Flight Operations Management
- Fault/BIT Display
- Maintenance Management

C. PREVIOUS WORK

Willard and Klesch in their 2005 thesis, used 36,742 observations from monitored components of 30 UH-60L helicopter's generators. The data was collected during the two-year period where the IMD-HUMS were installed. Each IMD-HUMS acquisition concerning the shaft, spur gear, and bearing of generators results in 170 variables. Each generator is assigned a binary value 1 or 0 to classify its known state. The value of one was given to the generators that were removed for fault, hence referred as bad generators. The value of zero was given to the generators that were not removed, referred to as good generators. To accomplish this generator classification, maintenance records and photographs from the 101st AVN Division were used.

Principal components and other techniques were applied to reduce the 170 initial predictors to only 10. A logistic regression model and random forest classifiers were used on each generator, and the plotted probabilities of being bad were smoothed and used to predict the current functional condition of generators in the test set. Only Condition Indicators (CI) computed in the last 20 observations of each generator were used in the predictive models because generators classified as bad were not necessary bad through their entire two-year history. Due to the highly variable nature of the predictor values, the model had lower success predicting states with just one acquisition. One the other hand, some surprising cases of generators which were wrongly presumed to be bad and, conversely, another generator which was wrongly assumed to be good, were classified correctly by this study's approach.

D. AREA OF RESEARCH AND APPROACH

ANNs have a number of traits that make them an attractive alternative to conventionally configured expert systems. First, many are capable of discriminating non-linear relationships. Second, they are capable of functioning with a certain degree of background noise and erroneous information with minimal degradation of their pattern recognition abilities. Third, they have the ability to generalize, having the ability to classify previously unseen vector patterns into existing and, in some cases, new output categories. They are also capable of identifying multiple faults. These are all areas where traditional expert systems typically fall short. Moreover, ANNs are data-based rather than rule-based. This means that they may be capable of correctly discriminating relationships previously hidden from the best of “experts”.

ANNs are not without their disadvantages. They, like all computer algorithms, are capable only of manipulating numbers and require an engineer to discern the intelligence of their output. Their success is largely limited to the quality of the data that they are provided. If the input vectors provided are inadequate to describe the decision space fully, then their likelihood for success is small. Again, they require an engineer to provide the proper inputs. Finally, they may be able to distinguish new relationships, but the relationships themselves remain hidden; all that is seen external to the network are the input and the output vectors. It is generally believed that the relationships are somehow hidden in the connection weights and the hidden layers but meaningful extraction of this information has yet to occur.

ANNs appeared to have potential in numerous fields, including machinery diagnostics. The question might be asked whether an ANN should theoretically be capable of recognizing patterns in vibration signatures. It is the scope of this research to determine whether this potential can be realized in the region of machinery diagnostics and specifically for the UH-60L helicopter’s electrical generator. In order to accomplish this, a database collected from IMD-HUMS is be used. Pattern recognition is an essential component of rotating machinery condition forecasting; therefore, examining and training different model structures and shapes in trying to identify patterns that are “storing” the “weights” of the networks is being researched.

E. STATISTICAL TOOLS

ANN software prices start at a few hundred dollars and can go to hundreds of thousands, or even more. The most expensive ones are generally packaged with more complete data mining products, which contain ANNs as one of the capabilities offered. This research utilizes one of those statistical packages, Clementine 10.0 produced by SPSS. This product is designed to function on servers and networks and has the ability to handle massive databases. The software also provides some handy features useful in evaluating the function fit by the NN, such as a computation of sensitivities. Through this software, the researcher had the opportunity to apply and, at the same time, train a number of different kinds of ANNs, in an effort to find the most suitable for the database of interest.

In addition, S-PLUS 7.0 from Insightful and Excel from Microsoft, two inexpensive and well-spread tools were used to assist the models in this research.

F. ORGANIZATION OF STUDY

This thesis begins with Chapter I, which briefly introduces the reader to the modern concept of CBM and the tools available to support it, like IMD-HUMS manufactured by Goodrich Corporation. A summary of previous relative work is provided, along with the author's area of interest and the tools utilized to achieve the objective of this research.

Chapter II is dedicated to the traditional explanation of ANNs. Theory and pictures are used concurrently trying to clarify what is commonly known to ANNs as a "black box" solution. Data enter the "black box" and a prediction comes out of it.

Chapter III describes the database, procedures that are followed to clean and choose the final data in use, and presents the steps of the methodology that leads to the result of this research.

Chapter IV briefly presents and discusses the results and outputs of the models that were trained in supporting the goal of this research. Most of the structures of these models are summarized in Appendix B.

Finally, Chapter V summarizes the conclusions and recommendations for future research ideas associated with the database and used techniques.

II. ARTIFICIAL NEURAL NETWORKS OVERVIEW

An Artificial Neural Network (ANN) is an information processing system that has certain performance characteristics in common with biological ANNs. They are parallel in nature and the fundamental idea behind them is that, if it works in nature, it must be able to work in computers. ANNs are data-based vice rule-based, so they possess the potential of being able to operate where analytical solutions are inadequate. They are reputed to be robust and extremely tolerant of noisy data.

A. HISTORY

The ANN concept was first introduced in 1943 by W. McCullock and W. Pitts, who, while trying to describe how the brain's neurons might work, modeled an ANN using electrical circuits. In 1949, D. Hebb introduced the training of the ANNs in his book "The Organization of Behavior," in which he argued that if two nerves fire at the same time, their connection is strengthened and thereby it is also possible that the same two nerves will fire again. By the 1950's, while computers became more advanced, researchers were eventually able to simulate such a hypothetical ANN. N. Rochester of IBM laboratories was an early pioneer in this field but, unfortunately, his effort failed. During 1959 and 1962, B. Widrow and M. Hoff developed two models (ADALINE and MADALINE) that recognized binary patterns, introduced a new learning algorithm applying the Least-Mean-Squares (LMS) learning rule, and used it to train adaptive ANNs.

For almost for two decades (1960-1980), interest in ANNs faded because of a lack of new ideas and computational power. In 1972, T. Kohonen and J. Anderson developed a similar network independently of one another, with both resulting in a collection of analog ADALINE circuits. The first multi-layered network was developed in 1975, an unsupervised network.

Since 1980, many researchers have boosted the idea and today ANNs are extremely popular as prediction and forecasting tools in a number of areas. The future of ANNs, though, lies in the development of hardware.

B. BIOLOGICAL NEURON

In principal, the brain is composed of almost 10 billion neurons; each is attached to about 10,000 other neurons. The main segment of the cell is called the soma or cell body (Figure 3). Neurons have a large number of extensions called dendrites. Each neuron receives electrochemical signals from other neurons connected throughout different axons. At the synapses—between the dendrite and axons—electrical signals are modulated in various amounts. If the sum of these electrical signals is sufficiently influential to activate the neuron, it produced an electrochemical output along the axon, and passes this signal to the other neurons, whose dendrites are attached at any of the axon terminals (Norgaardm M., Ravn O., Poulsen N., Hansen L., 2000). These attached neurons may then fire. It is essential to point out that a neuron fires only if the entire signal received at the cell body surpasses a certain level. The neuron either fires or it doesn't; there aren't different levels of firing.

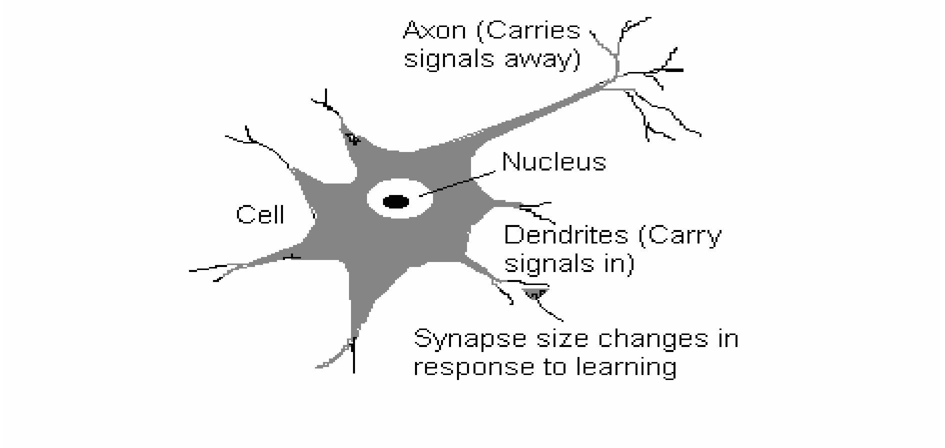

Axon (Carries signals away)
Nucleus
Cell
Dendrites (Carry signals in)
Synapse size changes in response to learning

Figure 3. A Biological Neuron (From: Lawrence, J., 1993)

The human brain is composed of these interrelated, electrochemical, broadcasting neurons. From a huge number of extremely naive processing units (each carrying out a weighted sum of its inputs, and then firing a binary output if the total input exceeds a certain level) the brain controls very complex tasks. This is the model on which artificial ANNs are based. Although ANNs haven't even approximated modeling the complexity of the human brain, they have appeared to be excellent at problems that are easy for a human but difficult for a conventional computer, such as image recognition and forecast based on earlier knowledge.

C. ARTIFICIAL NEURON

The artificial neuron is meant to mimic the major characteristics of the biological neuron. The three crucial mechanisms of the artificial neuron are:

(1) The synapses or connecting links that give weights, w_j to the input values, x_j for j=1, 2,…, m (Figure 4).

(2) A summer that adds all the weighted input values to calculate the input to the activation function $v = w_0 + \sum_{j=1}^{m} w_j x_j$, where w_0 called the bias (not to be confused with statistical bias in prediction or estimation) which is an arithmetic value associated with the neuron. It is suitable to consider the bias as the weight for an input x_0 whose value is constantly equivalent to 1, so that $v = \sum_{j=0}^{m} w_j x_j$.

(3) An activation function g that maps v to g (v), the output value of the neuron. This function is a monotone function.

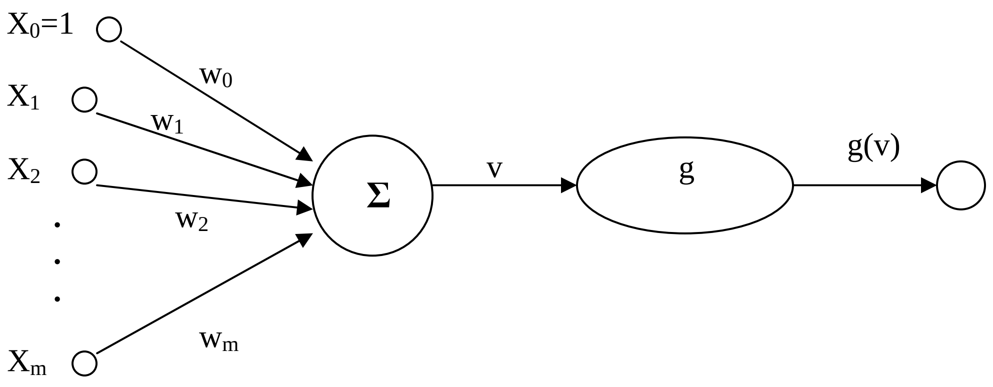


Figure 4. An Artificial Neuron

The artificial neuron has two stages of operation; the training stage and the using stage (Fausett, L., 1994). During the former, the neuron can be trained to fire or not, for specific input patterns or vector. In the latter, when a taught input pattern is identified at the input, its relayed output updates the current output. If the input pattern does not belong in the taught list of input vector or patterns, the firing rule is used to determine whether to fire or not. Is important to state that the firing rule is related to all the input patterns, not only the ones on which the neuron was trained.

D. ARCHITECTURE OF NEURAL NETWORKS

1. Single Layer Networks (SLN)

The structure of an artificial ANN consists of the 'input layer' connected to the 'hidden layer', which is connected to the 'output layer.' Any action of the input units corresponds to the raw data that is fed into the network, while the action of each hidden unit is determined by the behavior of the input units and the weights on the links between the input and the hidden units (Ripley, B., 1996). The status of the output units depends on the activity of the hidden units and the weights between the hidden and output units (Figure 5).

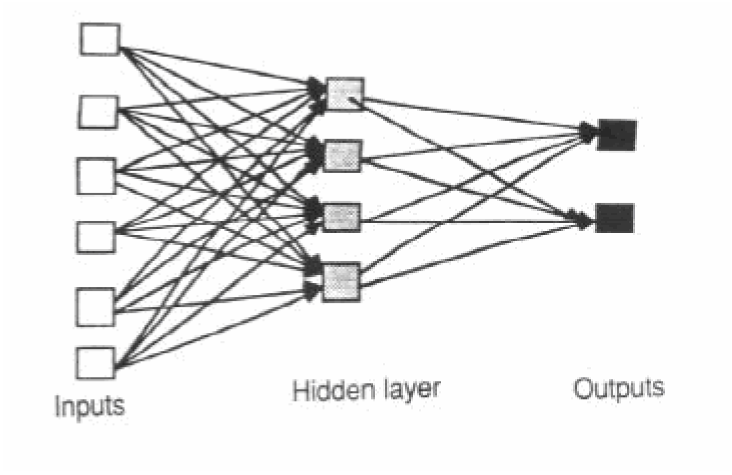


Figure 5. Single Layer Network

This simple class of network is especially interesting because the hidden units are free to build their own versions of the input. The weights between the input and hidden units decide when each hidden unit is active, and so by adjusting these weights, a hidden unit can choose what it represents.

2. Multi Layer Networks (MLN)

The most frequent ANN model is the multilayer perceptron (MLP). This family of ANN is known as a supervised network because it needs a preferred output for learning purposes (Kartalopoulos, S., 1996). The goal of this network is to generate a model that correctly matches the input to the output using chronological data so that the model can then be used to create the output when the desired output is not known. A graphical demonstration of an MLP is shown below (Figure 6).

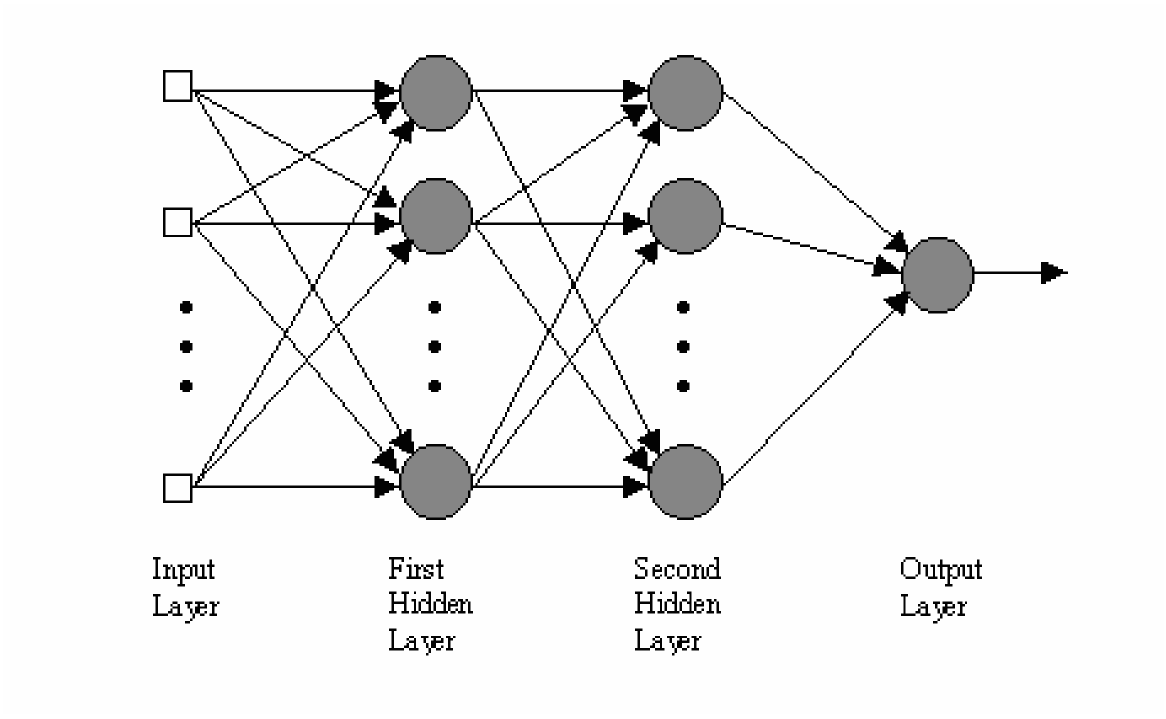


Figure 6. Multi Layer Network

3. Feed -Forward Networks (FFN)

Feed-forward ANNs are the most accepted and extensively used models in several realistic applications. FFNs permit signals to pass through one way only: from input to output. There is no feedback or loop and they tend to be straight-forward networks that connect inputs with outputs. They are broadly used in pattern recognition. This structure of organization is also referred to as bottom-up or top-down. (Figure 7)

4. Radial Basis Function Networks (RBFN)

The input of a Radial Basis Function (RBF) network is nonlinear while the output is linear. Because of their nonlinear approximation properties, RBF networks are capable of modeling complex mappings, which perceptron ANNs can only model by using multiple intermediary layers (Bishop, C., 1995). To use an RBF network we need to state the hidden unit activation function, the quantity of processing units, a criterion for modeling a given task, and a training algorithm for finding the parameters of the network.

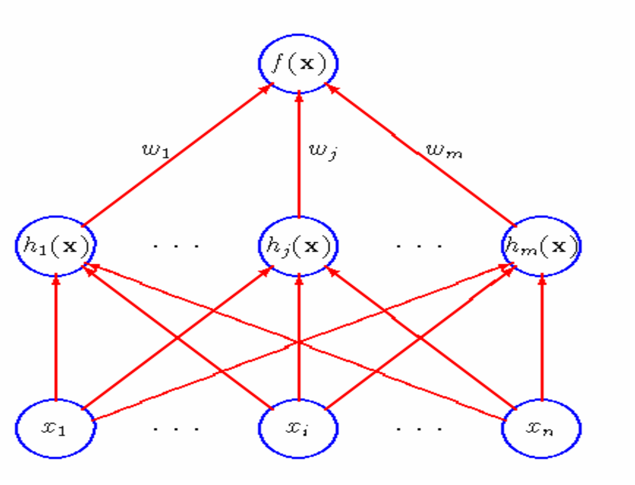


Figure 7. Feed Forward and RBF Network Representation

E. LEARNING PROCESS

One of the most essential phases of ANN is the learning process. Learning can be done in a supervised or unsupervised way.

1. Supervised Learning

In supervised learning, both the inputs and the outputs are known. During the training, the net runs the inputs and compares its resulting outputs against the known outputs. The differences of this comparison (the errors) are calculated, and the system adjusts the weights that manage the network. The aim is to establish a set of weights that minimizes the error. One famous method, which is frequent to many learning procedures, is the Least Mean Square (LMS) convergence (Duda, R., 2000). In this case, the network learns “offline” because the learning and the operation stages are different. Supervised learning can be subdivided into the following three general types:

a. Hebbian Learning

Hebbian learning is based on the premise that those connections that receive the most signal energy should in turn be strengthened. In this type of ANN, connection weights increase in a manner proportional to the magnitude of the signals provided that both the input through the path and the desired output are high. While historically important and neurologically accurate, it is not widely used in neural computing applications.

b. Delta Rule Learning

Today, the most frequent form of learning in use is the delta rule. Here, weights are adjusted based on a direct comparison between the actual and desired outputs. Back propagation is one learning rule based on the generalized delta rule:

$$W_{ij} = C_1 E_{ij} + C_2 M_{ij} + C_3 X_{ij} \tag{2.1}$$

where W_{ij} is the weight of the connection from the i^{th} element in the current layer to the j^{th} element of the previous layer; $C_1, C_2, \text{and}\, C_3$ are coefficients varying from 0 to 1; E_{ij} is the error proportional to the difference between the actual and desired output of the network; M_{ij} is the momentum term based on the difference between the previous weight of the given connection and the weight immediately prior to that; and X_{ij} is the activation energy associated with that particular connection (Ripley, B., 1996).

c. Competitive Learning

Competitive learning is where the output of processing elements is weighted according to the magnitude of its response relative to those of other processing elements. The "winning" processing element weighting is then modified according to comparison between actual and desired outputs. Thus only the strongest activation energies are adjusted; weak signals get progressively weaker unless the magnitudes of their response become comparable to those of the "winners".

2. Unsupervised Learning

In unsupervised training, the network is provided only with inputs and not with preferred outputs. In the training phase, an input pattern is applied to the input layer and the net is permitted to achieve equilibrium ("winner"). Thereby, weight changes are made according to some instructions. The model itself should then decide what features it will use to cluster the input data. This type of organization is also known self-organized or adoption. Here the network learns "online," because it learns and operates at the same time.

3. Activation Functions

The activation function is basically used to introduce nonlinearity to the net. The activation function $g(\bullet)$ transforms the presented input of an artificial neuron during its activation, and determines how influential should be the output from the neuron, based on the sum of the inputs. If the artificial neuron must mimic a biological neuron, the activation function $g(\bullet)$ has to be a simple threshold function returning binary values. However, this is not always the approach that artificial neurons implement. Sometimes it is more powerful and efficient to have a smooth differentiable activation function. The output from this group of activation functions lies within the ranges of [0,1] or [-1,1], depending on which activation function is applied. Some cases, where the identity function is used as the activation function, do not have these restrictions. On while, inputs and weights have no boundaries and take values within the R range $(-\infty, +\infty)$, in practice they often have small values centered around zero or are rescaled to have such small values.

As pointed out before, there are many different activation functions, but the most frequently used are the identity function (Figure 8), the sigmoid (Figure 9), and the hyperbolic tangent (Figure 10). It is obvious that all the functions should be differentiable because the back propagation (BP) algorithm requires this property in order to work out the data process within the network (Bishop, C., 1995).

An identity function, also known as an identity map or an identity transformation, is a function which does not have any effect. It always returns the same value that was used as its argument. For ANNs, that is reflected by the absence of hidden layers (perceptron).

$$\boxed{g(x) = x}$$

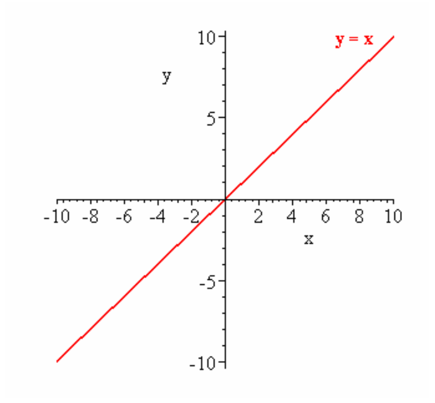


Figure 8. Identity Function

A sigmoid function, also known as logistic, is an S-shaped curve that maps all input to the range [0, 1]. It has a limit of 0 as x approaches negative infinity, and 1 as x approaches infinity.

$$\boxed{g(x) = \frac{1}{1+e^{-x}}}$$

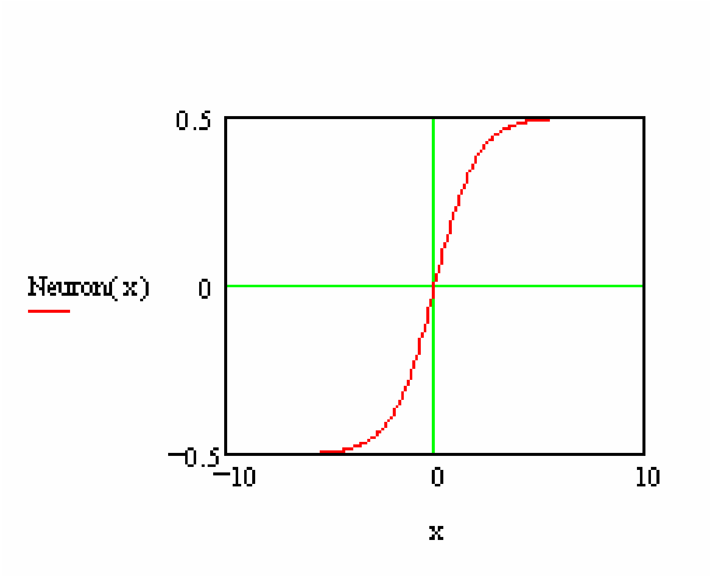


Figure 9. Sigmoid Function

A hyperbolic tangent function is analogous to a sigmoid, but it maps all of its input to the range [-1, 1]. It has a limit of -1 as x approaches negative infinity and 1 as x approaches infinity. The constants a and b define the possible output range (a) as well as the slope (b). The neuron transforms the stimulus in a nonlinear way, an essential precondition for solving a variety of highly complex problems.

$$\boxed{g(x) == \frac{e^{2x}-1}{e^{2x}+1}}$$

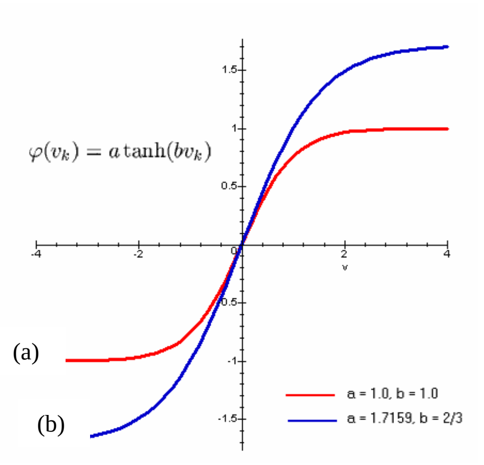


Figure 10. Hyperbolic Tangent Function

4. Gradient Descent

As clarified previously, in training, a function estimator frequently decreases to result in a value of $\vec{w}$ that minimizes a scalar error function $E(\vec{w})$. This is a typical optimization issue and various methods have been developed to answer it. The most frequent one is gradient descent. Gradient descent consists in thinking that E is the height of a landscape on the weight space: to locate a minimum, beginning from an arbitrary point, march descending until a minimum is reached (Figure 11) (Sobajic D., 1993). From the figure we can see that gradient descent will not at all times converge to an absolute minimum of E, but only to a local minimum. In the majority of these cases, this local minimum is good enough, given a realistic initial value of $\vec{w}$.

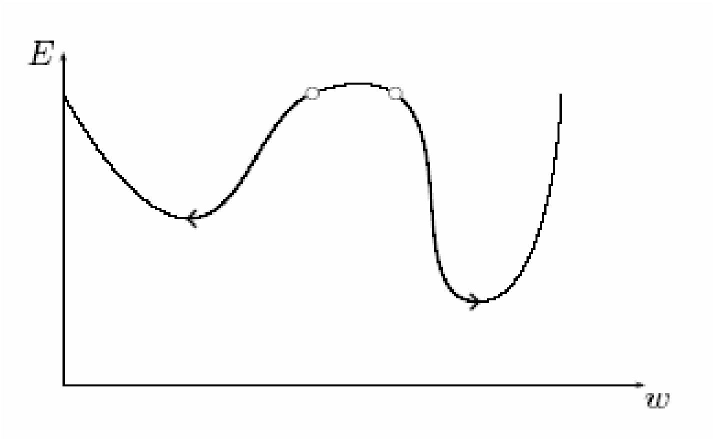


Figure 11. Gradient Descent.

5. Back propagation Algorithm

Back propagation (BP) is one of the earliest training algorithms, first developed by P. Werbos and is widely used for training supervised networks. The goal of BP is to minimize the square error of the predictions over all the observations (Fausett, L., 1994). Based on this algorithm, the output error is assumed to be collectively contributed by all connection weights. Basically, at the center of the algorithm, an application of the chain rule for ordered partial derivatives takes place to compute the sensitivity that a cost function has with respect to the environment and weights of the net.

Initial weights are usually chosen as small random values, so that each neuron will adapt a different set of weights. The network's input z_j to a node j is resolved by summing the weights of its inputs

$$z_j = \sum_i w_{i,j} x_i \ \forall\, j \tag{2.2}$$

where x_i designates the input in one node, and $w_{i,j}$ the weight from node i to node j. Next, the node's threshold value θ (bias) is added to net's input z_j and the calculated value is filtered through an activation function, usually a sigmoid function (Figure 12):

$$\mathrm{F}(\mathrm{z_j}) = \frac{1}{(1 + \mathrm{e}^{-\mathrm{z_j} + \theta_j})} \tag{2.3}$$

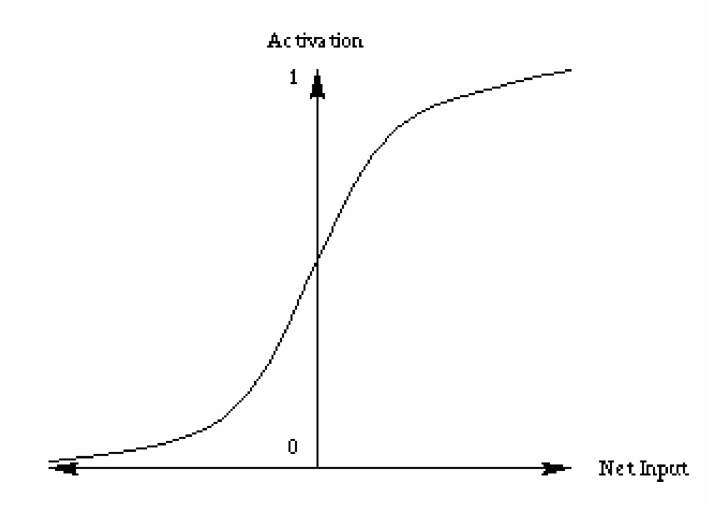


Figure 12. Sigmoid Function

The sigmoid function is also known as a ”squashing” function because it maps its inputs on a preset range number between 0 and 1.

The learning procedure in a BP net has two stages. At the first stage, each input pattern I_p is provided to the net sequentially, and propagated forward until the output. In the second stage, a technique called “gradient descent” is applied to minimize the total error on the input patterns within the training set. During this technique, weights are altered in proportion to the negative of an error derivative with respect to each weight

$$\Delta w_{j,i} = -\varepsilon[\frac{\partial E}{\partial w_{j,i}}] \tag{2.4}$$

Then, the weights moving toward the steepest descent of the error surface, which is defined by the total error

$$E = \frac{1}{2}\Sigma_p \Sigma_j (t_{p,j} - o_{p,j})^2 \tag{2.5}$$

where $o_{p,j}$ denotes as the responding output of node j to pattern p, and $t_{p,j}$ is the target output for node j. After the error on each pattern is calculated (1.2), all the weights are readjusted in proportion to this error, and back-propagated from the outputs to the inputs, applying the gradient descent method. The new calculated weights decrease the overall error in the net. The idea of gradient descent using only a single weight is presented in the following picture.

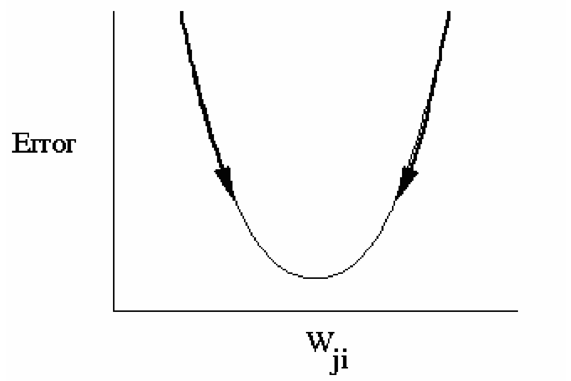


Figure 13. Gradient Descent Using One Weight

An application of the chain rule is used to develop the BP learning rule, while reworking the error gradient for each pattern as the product of two partial derivatives. The first partial derivative represents the change in error as a function of the network input $\partial E_p / \partial w_{j,i}$, while the second partial derivative represents the effect of a weight change on a change in the network input $\partial z_{p,j} / \partial w_{j,i}$. Modifying the error gradient turns into

$$\frac{\partial E_p}{\partial w_{j,i}} = [\frac{\partial E_p}{\partial z_{p,j}}] * [\frac{\partial z_{p,j}}{\partial w_{j,i}}] \tag{2.6}$$

Using the equation (2.2) for the net input z_j to a node j, we can solve directly for the second partial derivative and derive the network's output $o_{p,i}$ for the pattern p to the node i:

$$\frac{\partial z_{p,j}}{\partial w_{j,i}} = [\frac{\partial(\sum_k w_{i,k} o_{i,k})}{\partial w_{j,i}}] = o_{p,i} \tag{2.7}$$

Naming the negative of the first partial derivative as the error signal:

$$d_{p,j} = -\frac{\partial E_p}{\partial z_{p,j}} \tag{2.8}$$

the corresponding change in the weight $w_{i,j}$ with respect to the error E_p becomes

$$\Delta_p w_{j,i} = \eta * d_{p,j} * o_{p,j} \tag{2.9}$$

where η is a parameter describing the learning rate. The speed and accuracy of the learning process during the iterations to update the weights also depend on the learning rate η. A low learning rate can guarantee more stable convergence, but a high learning rate can accelerate convergence in several cases.

The next step in the BP algorithm is to calculate the $d_{p,j}$ for each node in the net. The equation (1.8) can be rewritten as:

$$d_{p,j} = -[\frac{\partial E_p}{\partial o_{p,j}}]*[\frac{\partial o_{p,j}}{\partial z_{p,j}}] \tag{2.10}$$

To compute the first partial derivative there are two cases to examine:

a. First Case

Assume that j is an output node of the net; then, from equation (1.5) it follows that:

$$\frac{\partial E_p}{\partial o_{p,j}} = 2(t_{p,j} - o_{p,j}) \tag{2.11}$$

substitute equation (1.11) to equation (1.8) becomes:

$$d_{p,j} = 2(t_{p,j} - o_{p,j}) * f(z_{p,j}) \tag{2.12}$$

b. Second Case

Assume that is not an j output node of the net, then applying again the chain rule, we obtain:

$$\begin{aligned} \frac{\partial E_p}{\partial o_{p,j}} &= \sum_k [\frac{\partial E_p}{\partial z_{p,k}}][\frac{\partial z_{p,k}}{\partial o_{p,j}}] = \\ &= \sum_k [\frac{\partial E_p}{\partial z_{p,k}}][\frac{\partial(\sum_i w_{k,i} o_{p,i})}{\partial o_{p,j}}] = \\ &= \sum_k [\frac{\partial E_p}{\partial z_{p,k}}] w_{k,j} \\ &= \sum_k d_{p,k} w_{k,j} \end{aligned} \tag{2.13}$$

Combining now the above cases, we form an iterated process for calculating the signal error $d_{p,j}$ for all nodes in the net. These errors can then be used to update its weights.

As BP uses a gradient descent method, the corresponding net tracks the contour of an error surface with weight updates moving in the direction of the steepest descent. Assuming a plain net without hidden layers, it is easy to minimize the error using gradient descent because the error surface is bowlshaped (Figure 13). The net will always locate an optimal solution at the base of the bowl. Such optimal solutions are called global minima. On the other hand, cases which are more complex require the existence of an extra hidden layer to carry out the solution to such difficult problems. Here, error surfaces become also complex, containing possibly many minima. Because some minima are deeper than others, it is possible that gradient descent will not locate a global minimum, and the network may be trapped in a local minimum, which is a suboptimal solution.

It is clear that we want to avoid local minima while training a BP net. Although in some case this may be difficult to do, in practice it is essential to try to find how often and under what circumstances local minima occur. Moreover we have to study possible approaches for avoiding them. It is known in the ANNs theory that the more hidden layers you have in a net, the less possible you meet a local minimum during training. Although additional hidden nodes amplify the complexity of the error surface, the extra dimensionality increases the number of possible flee paths.

The BP algorithm analyzed in this chapter only involves only weight changes that are proportional to the derivative of the error. As we mentioned before at equation (1.9), the increment of the learning rate contributes η to an increment of the weight changes on each iteration, and the faster the net learns (although the magnitude of the learning rate can also control whether the net reaches a stable solution). If the learning rate gets too large, then the weight changes no longer approximate a gradient descent procedure and that often results in oscillation of the weights. Obviously, we want to get the largest learning rate without causing oscillation, achieving the best learning speed while minimizing the training time for the net. One technique that has been used is a small modification of the BP algorithm that contains a momentum term.

The idea of momentum is that previous changes in the weights should control the current direction of movement in the weight space. This idea is implemented by the adjusted weight update rule:

$$\Delta w_{j,i}(\eta+1) = \varepsilon * d_{p,j} a_{p,j} + \alpha \Delta w_{j,i}(\eta) \tag{2.14}$$

where η is the learning rate. With momentum, if the weights begin to move in a specific direction in their space, they tend to keep on moving in that direction. Momentum can aid the net to overcome a local minimum, in addition to speeding learning, particularly along extensive flat error surfaces.

In Clementine, the default learning rate is 0.25 and the default momentum parameter is 0.9. When using BP for a series of problems, much smaller values than these are often used. For especially complex problems, a learning rate of 0.01 is very common.

6. Efficient Algorithms

Selecting the proper value for the learning rate η is not an easy concept. If η has a small value, then the learning procedure will be too slow. On the other hand, if η is assigned a large value, the learning procedure may diverge. An acceptable value of η can be established by trial and error, but this is a quite boring and wasteful process (Bishop, C., 1995). In order to deal with this problem, a large range of efficient learning methods has been developed. Here, we briefly present the most essential theoretical ideas underlying them. One of the most basic ideas for speeding up the learning procedure is to use the second-order information regarding the error function. Assuming a quadratic error in one dimension, the best learning rate is the inverse of the second order derivative (Figure 14), which can aid in designing capable learning methods.

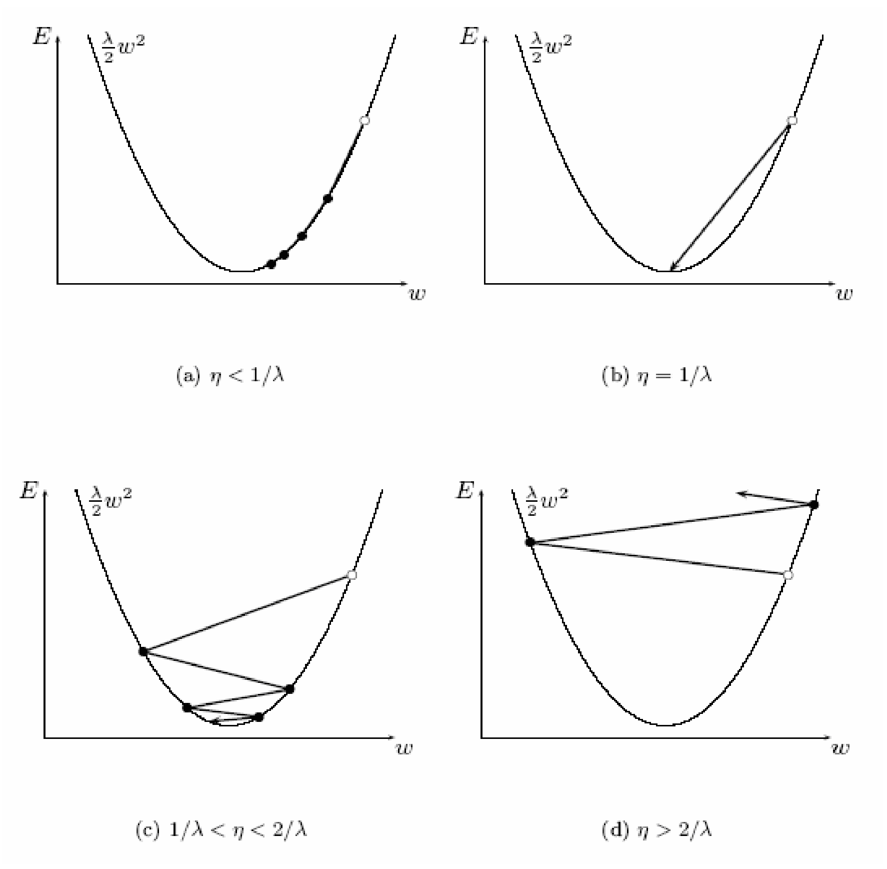


Figure 14. Learning Rate effect on Gradient Descent (From: Fausett, L., 1994)

If it is possible to calculate this second-order derivative, then it is feasible to achieve a good learning rate. Unhappily, the error function might not be quadratic at all. Therefore, setting the learning coefficient to the inverse of the second-order derivative only works near the optimum, in regions where the quadratic approximation is valid. But if the second-order derivative is negative, this does not work. For handling such circumstances, particular care must be taken. Moreover, a few other issues come up when the dimension of the weight space is bigger than one, which is the most common case in practice. The second order derivative is not a single number anymore, but a matrix called the Hessian and defined as:

$$H = \frac{\partial^2 E}{\partial \vec{w}^2} = \begin{pmatrix} \frac{\partial^2 E}{\partial \vec{w_1}^2} & \frac{\partial^2 E}{\partial \vec{w_1} \partial \vec{w_2}} & \cdots & \frac{\partial^2 E}{\partial \vec{w_1} \partial \vec{w_n}} \\ \frac{\partial^2 E}{\partial \vec{w_2} \partial \vec{w_1}} & \frac{\partial^2 E}{\partial \vec{w_2}^2} & \cdots & \frac{\partial^2 E}{\partial \vec{w_2} \partial \vec{w_n}} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ \frac{\partial^2 E}{\partial \vec{w_n} \partial \vec{w_1}} & \frac{\partial^2 E}{\partial \vec{w_n} \partial \vec{w_2}} & \cdots & \frac{\partial^2 E}{\partial \vec{w_n}^2} \end{pmatrix} \tag{2.15}$$

Sometimes it is probable to have dissimilar curvatures in different directions (Figure 15). This can generate a major problem if there is, for instance, a second derivative of one hundred (100) in one direction, and a second derivative of one (1) in another. In this case, the learning coefficient must be less than 0.002 in order to avoid divergence. This means that convergence will be very slow in the direction where the second derivative is one (1). This phenomenon is called 'ill conditioning.' Efficient algorithms frequently try to change the weight space in order to have the same curvatures in all directions. This has to be done cautiously so that instances where the curvature is negative work as well. Some of the best algorithms are QuickProp, Levenberg Marquardt, and Conjugate Gradient.

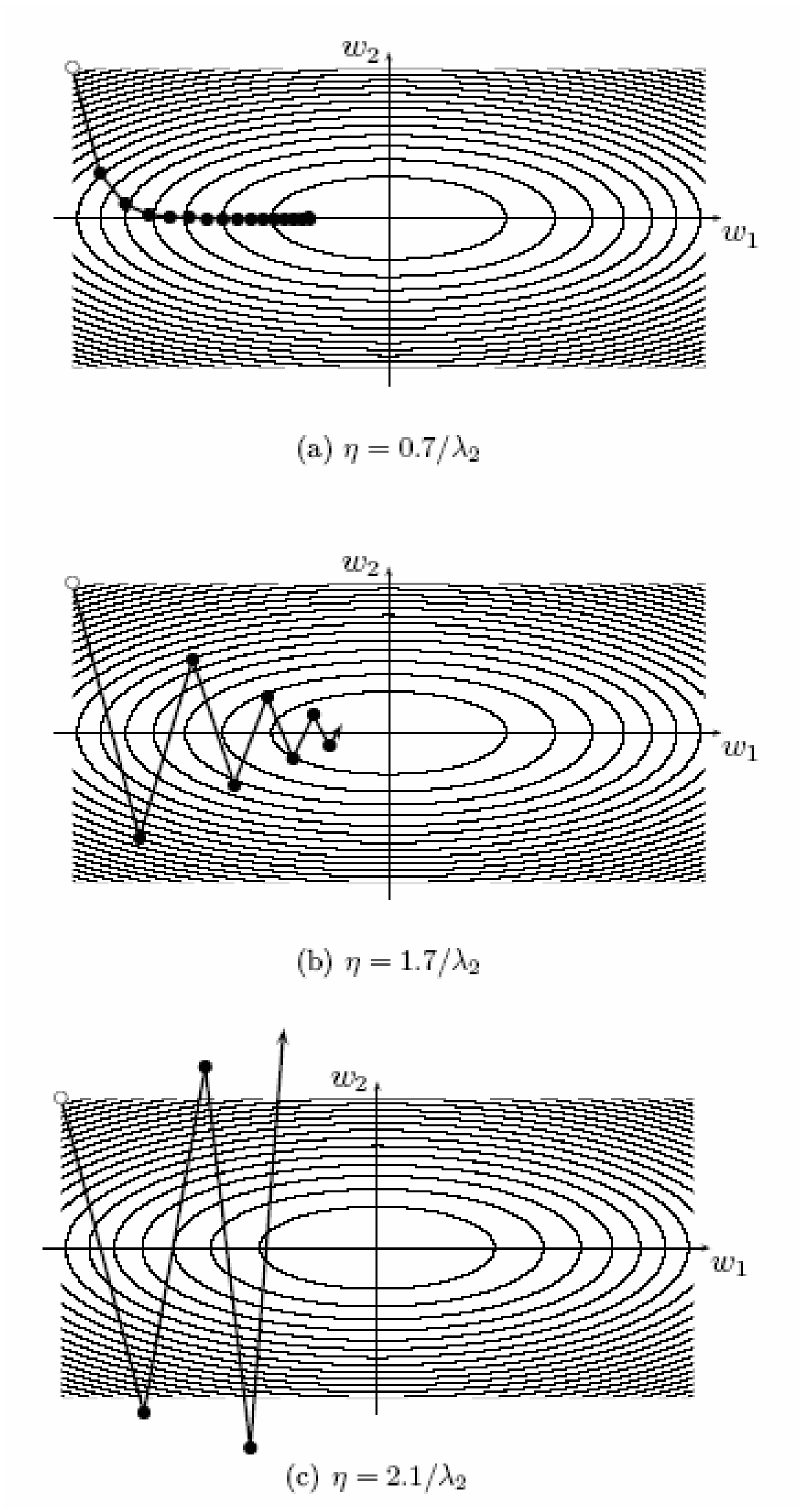


Figure 15. Ill Conditioning. (From: Bishop, C., 1995)

7. Batch Vs Incremental Learning

In unsupervised learning, as described earlier, the error function is frequently defined as a sum of error terms over a finite amount of training samples that consist of pair vectors (input, output). Again, the error function is:

$$E = \sum_{i}^{p} E_i \tag{2.16}$$

with

$$E_i = \frac{1}{2}(f_{\vec{w}}(\overrightarrow{x_i}) - \overrightarrow{y_i})^2 \tag{2.17}$$

Applying the steepest descent on E is known as "batch learning" (BL) for the reason that the gradient of the error has to be calculated on the full training set previous to weights modification. A different method to modify weights aiming to minimize E is "incremental learning" (IL). Within this method, the gradient descent steps are applied on E_i instead of E. Some other known names for IL are online or stochastic learning. Now the obvious question that arises is which of these methods is the best. The answer is not so simple and depends always on the particular problem to be solved. Here are a few of the tips to consider (Simpson P., 1996):

a. Advantages of Incremental Learning (IL)

- IL is usually faster, particularly when the training set is redundant. In situations where the training set has input and output patterns that are similar, BL wastes time calculating and adding similar gradients before setting one weight update.
- IL often results in better outcomes. This happens because the randomness of IL generates noise in the weight updates. This noise aids weights to jump out of bad local optima.
- IL is capable of tracking changes. As an example, consider a learning model of the dynamics of a mechanical system. While this system gets older, its properties might slowly evolve and IL can track this type of drift.

b. Advantages of Batch Learning (BL)

- In IL, noise causes the weights to continuously oscillate around a local optimum, and they never converge to a constant stable value. This is not the case in BL, making it easier to analyze.

- Various acceleration methods can operate only in BL, such as some of the algorithms mentioned earlier (QuickProp, Conjugate Gradient).
- Another benefit related to the absence of noise in BL is that the theoretical analysis of the weight dynamics and convergence rates are very simple.

THIS PAGE INTENTIONALLY LEFT BLANK

III. DATA DESCRIPTION AND METHODOLOGY

Vibration analysis is among the most powerful tools available for the detection and isolation of incipient faults in mechanical systems. Among the methods of vibration analysis in use today and under continuous study are broadband vibration monitoring, time-domain analysis, and frequency analysis. All have varying degrees of utility in machinery condition monitoring and diagnostics and all have characteristics that lend themselves particularly well to specific applications. Since the effectiveness of ANN is directly related to how effectively the chosen inputs define a particular decision space, the selection of the optimum vibration parameters for inputs to the ANNs is critical. Thus, a good understanding of elementary machinery diagnostics techniques is essential.

A. SOURCES OF VIBRATION

In mechanical systems, any mechanical component which periodically comes in contact with a second component to transmit an axial, radial, or torsional load is a potential source of mechanical vibration. In machines with a gear train, the principal components involved with load transfer will be its torsional power source, such as a motor; the gear meshes; the bearings; and those items that interconnect them, the shafts. Additionally, because vibrational isolation is seldom complete, additional extraneous sources of vibration will also be present. The diagnostician is generally interested in extracting the vibrations created by specific machinery components and ignoring the other sources as extraneous noise. In this study, we are particularly interested in the vibrations generated by the rotating machinery's gears, bearings, and shafts. As such, the discussion will be limited to these sources of vibration.

1. Gear Vibration

In a gear train, the gear mesh is the dominating source of mechanical vibration. This vibration primarily stems from the non-uniformity in the transmission of angular motion from one gear to its mate. The non-uniformity of the angular motion occurs due to geometric deviations of the contact surfaces from the ideal involutes shape and the elastic deformation that any mechanical system undergoes when transmitting a load (Mark, W.D., 1998). Moreover, torque fluctuations and deflections of the gearbox can also be sources of vibration in gears. Clearly, any damage that occurs to the gear contact surface, as well as other mechanical linkages to the gear mesh, will also have an effect on the gear's vibration (Mattew, J., and Alfredson, R.J., 1987).

2. Bearings

Bearing vibrations occur for much the same reasons as gears. However, because bearings are not situated directly along the power transmission train and support largely static loads, they characteristically generate a small vibration signal until the damage inflicted upon them reaches advanced stages. Because of the low magnitude of these signals, they are often masked by much stronger gear-related signals. Partially because of this belated detection of trouble, antifriction bearings are among the most common causes of machinery failure in moderately sized machines. The frequencies associated with the bearing-related signals generally depend on the location of the damage, the dimensions of the bearings, and the shaft rotation speed.

3. Shafts

Shafts generally produce vibration signals at their rotational frequency and its harmonics. Shafts are also prone to a number of different faults, all of which register at the shaft rotative frequency. In the case of bent shafts and shafts misalignments, the second harmonic is the dominant frequency in 90 percent of the cases (Collacott, R.A., 1979). Imbalances in the shaft or load characteristically generate a dominant signal at the shaft rotative frequency but there tends to be a phase shift as well. Mechanical looseness can also introduce increases in the shaft rotational frequency but also characteristically involves higher harmonics as well (Hewlett Packard, 1983).

B. DATA COLLECTION

The database used in this research was provided by Goodrich Corporation and collected through the IMD-HUMS installed on 30 UH-60L helicopters. The period of data collection starts at 9/22/2003 and stops at 7/31/2005. The total number of observations utilized was 36,742 and each observation consists of 169 fields. For the database's structure and reference ease, each generator was assigned a number starting from 1 up to 66. Appendix A summarizes the allocation of generators among the helicopters and the number of the recorded observations for each generator.

C. SELECTING VARIABLES

As in any prediction or forecasting model, the selection of appropriate model inputs is extremely important. However, in most ANN applications, less attention is given to this task. The main reason for this is that ANNs belong to the class of data-driven approaches, whereas conventional statistical methods are model-driven. In the latter, the structure of the model has to be determined first, which is done with the aid of empirical or analytical approaches, before the unknown model parameters can be estimated. Data-driven approaches, on the other hand, have the ability to determine which model inputs are critical, so there is less need for "...a priori rationalization about relationships between variables..." However, presenting a large number of inputs to ANN models and relying on the network to determine the critical model inputs usually increases network size. This has a number of disadvantages, such as decreasing processing speed; increasing the amount of data required to estimate the connection weights efficiently, and degrading the AAN performance. This is particularly true for complex problems, where the number of potential inputs is large and where no a priori knowledge is available to suggest which inputs to include.

1. Input Vector

According the vibration theory mentioned before and after an assiduous study of the entire data recorded by the IMD-HUMS, the variables which potentially could form a well-informed input vector—all of them or subsets—to the training process of this research model, are shown and briefly explained below (Goodrich Corporation, 1998):

a. Torque

Torque is a measure of how much force acting on an object causes that object to rotate.

b. SO_1 (Shaft Order 1)

SO_1 is the once-per-revolution energy in the signal average and is used to detect shaft imbalance.

c. SO_2 (Shaft Order 2)

SO_2 is the twice-per-revolution energy in the signal average and is used to detect shaft misalignment.

d. SO_3

SO_3 is the thrice-per-revolution energy in the signal average and is used to detect shaft disparity.

e. Signal Average RMS

Frequencies that are integer multiples of the basic shaft frequency will be enhanced by the averaging process and other frequencies will be relatively attenuated. Depending on the mechanical environment, the signal average generally requires about 100 revolutions to converge to a usable waveform.

f. Residual Kurtosis

The residual analysis first removes all the strong tones from the signal average to produce a residual signal so as to minimize the interference of these strong tones. Here the process handles kurtosis, which measures the thickness of the tails of the distribution of bearing vibrations after the background signal has been removed (Harris, 2002).

g. Residual RMS

The residual process deals with the Root Mean Square, which is the overall energy level of the vibration data.

h. Side Band Modulation_1

This analysis is designed to reveal any sideband activities that may be the result of certain gear faults, such as eccentricity, misalignment, or looseness. The indicator characterizes the degree of sideband modulation for the first sideband.

i. Gear Distributed Fault

This attribute is an effective detector for distributed gear faults, like wear and multiple tooth cracks. It is a dimensionless measurement calculated from the ratio of explained and unexplained variances of a vibration generated at the meshing of gears.

j. G2_1

G2_1 is an algorithm developed by Goodrich Corporation to compute the ratio of the signal average peak to peak and the gear meshing energies.

k. Residual Peak to Peak

The residual process deals with the algebraic difference between the extremes of the vibration quantity.

l. Gear Misalignment_1

Gear Misalignment_1 is a dimensionless measurement resulting from the ratio of the energies of the vibrations produced when gears mesh (Harris, 2002).

m. Ball Energy

Ball Energy is the total energy associated with the bearing ball spin defect frequency and its harmonics.

n. Cage Energy

Cage Energy is the total energy associated with the bearing cage defect frequency and its harmonics. Usually it is detectable only at the later stage of a bearing failure, but some studies show that this indicator may increase before the others.

o. Inner Race Energy

Inner Race Energy is the total energy associated with the bearing inner race defect frequency and its harmonics.

p. Outer Race Energy

Outer Race Energy is the total energy associated with the bearing outer race defect frequency and its harmonics.

q. Envelope RMS

The main purpose of envelope analysis is to sum and normalize the six multiples of frequencies above or below the Root Mean Square value of the vibration.

Table 1 enumerates the above potential predictors for later reference purposes.

Predictors			
1	Torque	10	G2_1
2	SO1	11	Residual Peak to Peak
3	SO2	12	Gear Misalignment_1
4	SO3	13	Ball Energy
5	Signal Average RMS	14	Cage Energy
6	Residual Kurtosis	15	Inner Race Energy
7	Residual RMS	16	Inner Race Energy
8	Side Band Modulation_1	17	Envelope RMS
9	Gear Distributed Fault		

Table 1. Potential Model Predictors

2. Output Vector

In this research's models, the output vector consists only of one variable, which is the operating condition of the generator. Each generator is assigned a binary value of 1 or 0 to classify its known state. The value of 1 corresponds to generators removed for fault while the value of 0 corresponds to good generators. The fact that each generator is assigned a state of 0 (good) or 1 (bad) does not mean these generators are actually in the assigned state. The given state of 0 (good) or 1 (bad) is based only upon whether a generator was removed for fault or not, according the maintenance records. A generator with a hidden fault would be assigned a state of 0 (good). Similarly, a generator which was taken out for a malfunction and given a state of 1 (bad) could have been mechanically good (Willard, L., Klesch, G., 2005). Unlikely the previous work of Willard and Klesch (2005), the entire history of "good" and "bad" generators is used here. Thus most "bad" generators should have an initial period of "good" following by "bad" as they fail.

The following table shows the generators that were confirmed as bad, either after maintenance or based on IMD-HUMS data, and assigned the value of 1 (Table 2). The failure of two of the generators, numbers 9 and 33, were detected during operation by a generator warning light. Faults in the remaining four generators, numbers 22, 31, 53, and 56, did not trigger the generator warning light. However each of the four generators had unusually high SO1 readings upon removal. Three of these generators, numbers 22, 31, and 53, showed evidence of fault or wear. The removal of generator number 56 resulted from the case of an identifiable buzz (Willard, L., Klesch, G., 2005).

Bad Generators—Reasons for Replacement	
Generator	**Comments**
9	Generator failed during shutdown upon APU generator coming on during start.
22	SO1 near 2 ips. After Spline Adapter Coupler replacement, SO1 returned to 0.05 ips.
31	SO1 at 3 ips. After Spline Adapter Coupler replacement, SO1 remained high and so generator was replaced
33	Generator bad
53	SO1 at 3 ips. After Spline Adapter Coupler replacement SO1 returned below 0.05 ips.
56	SO1 over 4 ips. Generator and Spline Adapter Coupler were replaced.
Note:	Replacement of generators 9 & 22 were made due to maintenance records while the rest due to Johnny Wright and Ground Station Team, IMD-HUMS Fault Detections, Goodrich Corporation. Draft 5/25/2005 (Ver 117)

Table 2. Bad Generators—Reasons for Replacement

D. DATA PREPROCESSING

Data preprocessing is frequently used to analyze and transform the input and output variables to minimize noise, emphasize essential relationships, identify trends, and flatten the distribution of the variables to aid the ANN in learning the relevant patterns. Since ANNs are pattern matchers, the representation of the data is important in designing a successful network. In most datasets there is a large variability in the scale of range fields. To balance this effect of scale, range fields are transformed so that they all have the same scale. In Clementine, range fields are rescaled to have values between 0 and 1. The transformation used is:

$$x_i' = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} \tag{2.18}$$

where x_i' is the rescaled value of input field x for record *i*, x_i is the original value of x for record *i*, $x_{\min}$ is the minimum value of x for all records, and $x_{\max}$ is the maximum value of x for all records.

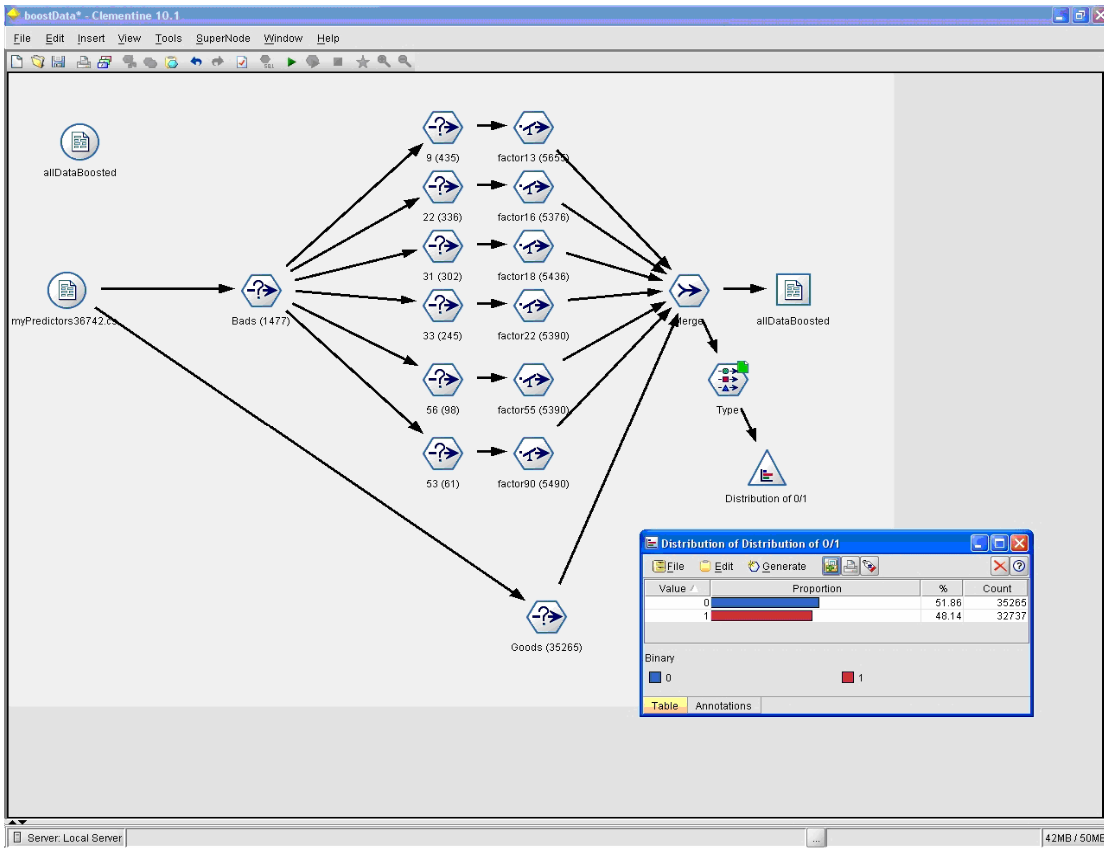
An additional problem for network representation of the utilized database was that, out of 36,742 total observations, only 1,477 cases referred to the bad generators. This fact directly affects the learning procedure of the network by creating a tendency to predict only good generators. For example, Table 3 gives results for a AAN using the default settings in Clementine where the classification is perfect for “good” generators while the classification for the “bad” generators is that actually they are “good” too. Changing the ANN architecture consistently gives similar results; most observations are predicted as “good.” To correct this situation, the records of bad generators were replicated so that the ratio of good to bad generators was close to 1. We can consider this scheme as a weighting technique to emphasize the input vector of the bad generators and, concurrently, the information they might convey. The multiplication factors differed for each generator. They chose so that each bad generator had about the same number of total observations and so that the number of bad generators was about equal to the number of good generators. Table 4 describes the latter procedure of data manipulation, and Figure 16 presents the setup of Clementine to create the new normalized database.

ORIGINAL SET	ALL DATA W/O GEN	TRAINING ACCURACY	PREDICTION ACCURACY	
			BAD	GOOD
36742 observations	9	98.479 %	0 %	100 %
	22	98.173 %		
	31	97.111 %		
	33	97.657 %		
	53	97.151 %		
	56	96.928 %		

Table 3. Training Set using only Original Data

Generator		Observations	Factor	Result	Total	Ratio
Bad Gen	9	435	13	5655	32737	0.93
	22	336	16	5376		
	31	302	18	5436		
	33	245	22	5390		
	53	61	55	5390		
	56	98	90	5490		
Good Gen		All			35265	

Table 4. Data Multiplication of Bad Observations



Value	Proportion	%	Count
0		51.86	35265
1		48.14	32737

Figure 16. Clementine Preprocessing Data

E. DATA SETS

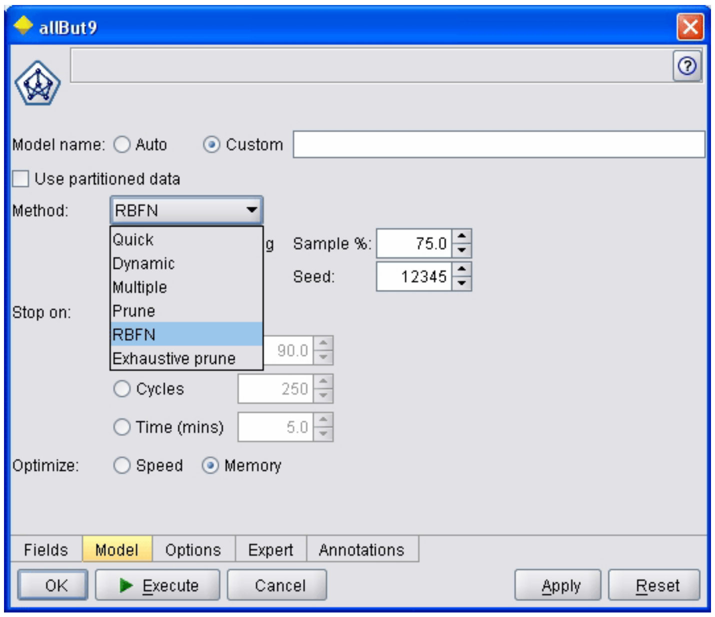
In ANNs, it is a common practice to partition the database into three separate sets called the training, test, and validation sets.

1. Training Set

The training set is the largest set and is used by the ANN to learn the patterns that exist in the data. The training set used for this research consists of the observations of five (5) bad and fifty-six (56) good generators, for a total of about 60,000 records.

2. Test Sets

The test sets, varying in size from 10% to 30% of the training set, are used to evaluate the ability of a trained ANN to generalize to a new set of data. The researcher fits the parameters and the topology of the network that achieves the best results on a test set. Using the features of the software, the size of a test set was always taken to be at 25% of the training set. Periodically during the process of training, new test sets are selected from the training set. Although this sample was selected randomly from the training set, the same seed of (12345) was used, in order to duplicate results for different model fits. Figure 17 shows the format chosen for the training and testing sets.



allBut9

Model name: Auto [✓ Custom]

Use partitioned data

Method: RBFN

Sample %: 75.0

Seed: 12345

Stop on: 90.0

Cycles 250

Time (mins) 5.0

Optimize: Speed [✓ Memory]

Figure 17. Model Architecture GUI

For the purposes of training, each record is considered as an observation. Thus it is likely that the test set contains records from each of the 61 generators.

3. Validation Set

The validation set is used last as a final check on the performance of the trained network. The size of the validation set should keep a balance between obtaining a sufficient sample size to evaluate a trained network and having enough remaining observations for both training and testing. The validation set consists of one (1) bad and four (4) good generators. The records from these five generators are not used in training the network. They are only used for validation.

F. NETWORK ARCHITECTURE AND EVALUATION CRITERIA

As a baseline, this analysis uses a model architecture consisting of one input layer containing the predictors for the model, one output layer giving the estimate of the probability that the record is bad, and one hidden layer. Clementine provides six training methods for building ANN models but for this research utilized only the following two: (Figure 17)

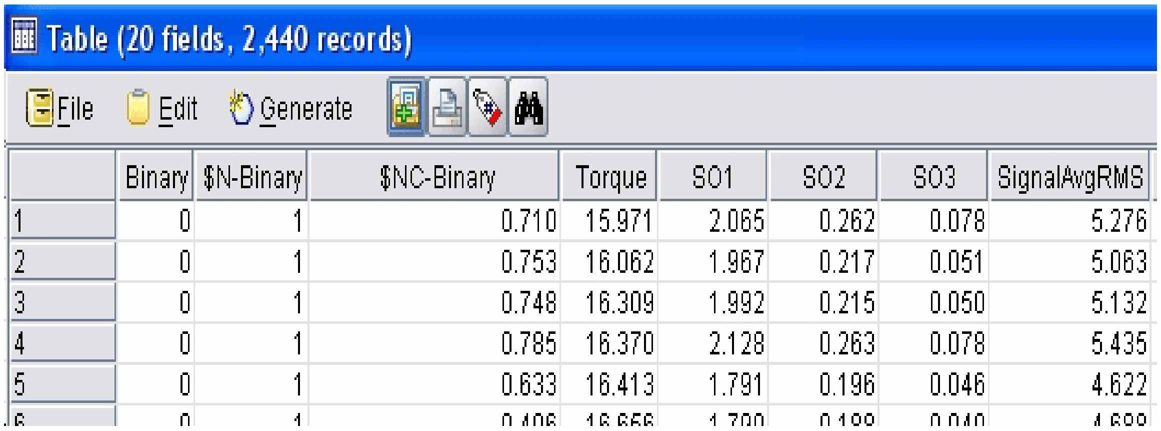
- **Prune:** This method starts with a large network and removes (prunes) the weakest units in the hidden and input layers as training proceeds. This method is usually slow, but it often yields better results than other methods.
- **RBFN:** The radial basis function network uses a technique similar to k-means clustering to partition the data based on values of the target field.

Clementine incorporates several features to avoid some of the common pitfalls of ANNs, including sensitivity analysis, network accuracy, and feedback graph. With these options selected, a sensitivity analysis will provide information on which input fields are most important in predicting the output field, network accuracy will provide the percentage of records for which the prediction of the model matches the observed value in the data, and the feedback graph will depict the accuracy of the network over time as it learns (Clementine 10.0 Node Reference, 1999). Moreover, a Confidence level ($N-Binary) is provided for each observation after the prediction, which, in this model with a flag (0 or 1) output, is calculated by using the formula:

$$\$\text{N-Binary} = 2\,|\,0.5 - \text{Raw Output}\,| \qquad (2.19)$$

where the RawOutput is the output unit value scaled so that it is between o and 1.

If the output unit value is below 0.5, the observation is predicted as 0 (false), and if it is 0.5 or above, the observation is predicted as 1 (true). For example, if the ANN prediction value is 0.72, the prediction is displayed as "true," and the confidence will be $2|0.5-0.72| = 0.44$. A part of some output results is presented in Figure 18.



	Binary	$N-Binary	$NC-Binary	Torque	SO1	SO2	SO3	SignalAvgRMS
1	0	1	0.710	15.971	2.065	0.262	0.078	5.276
2	0	1	0.753	16.062	1.967	0.217	0.051	5.063
3	0	1	0.748	16.309	1.992	0.215	0.050	5.132
4	0	1	0.785	16.370	2.128	0.263	0.078	5.435
5	0	1	0.633	16.413	1.791	0.196	0.046	4.622
6	0	1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Figure 18. Clementine Prediction Table

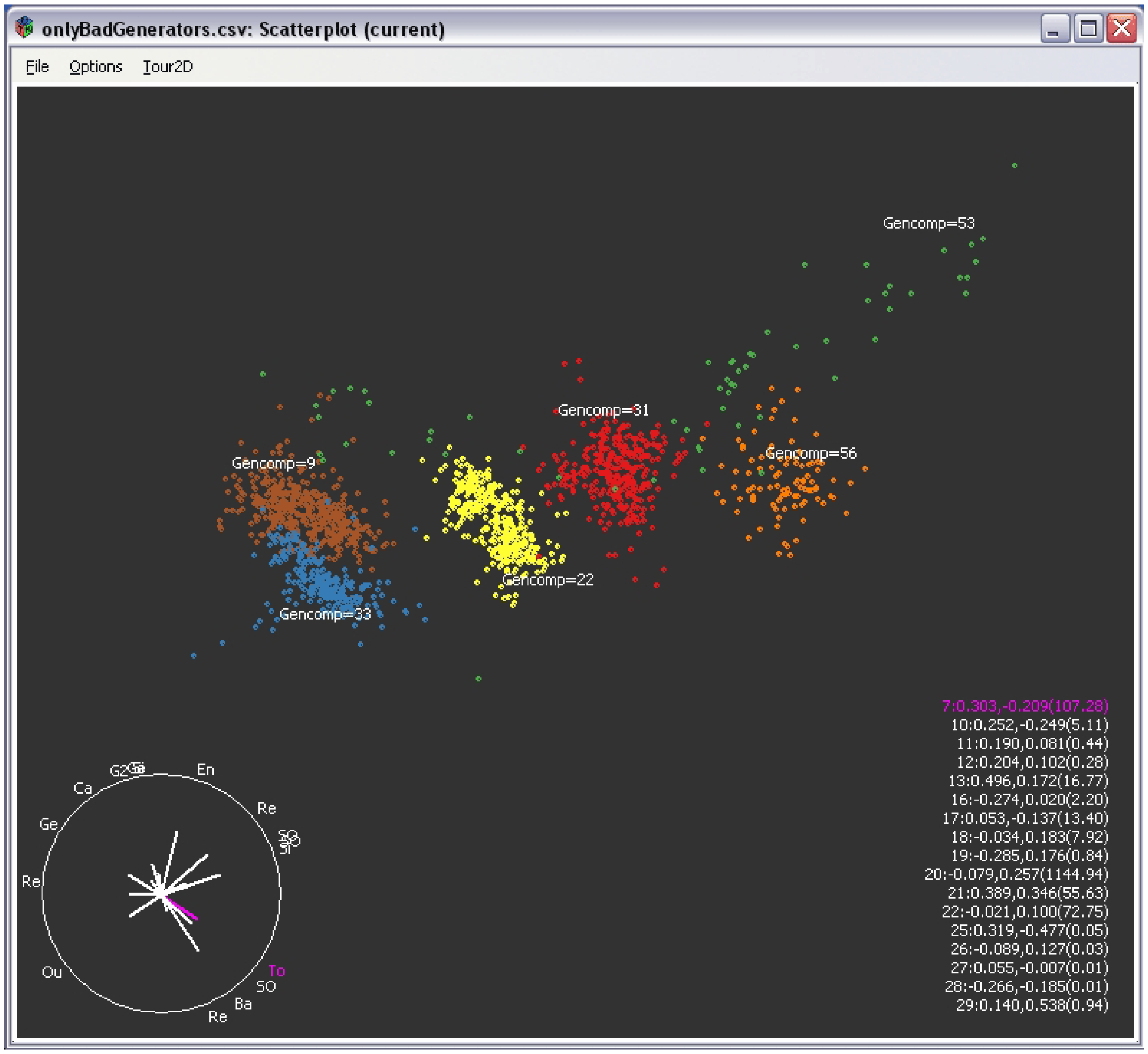
IV. RESULTS AND DISCUSSION

A. MODEL WITH ALL PREDICTORS

The first attempt of this experiment was to predict the state of one (1) bad and four (4) good generators. The good generators were chosen randomly and form a constant group for the rest of the experiment. The bad generators were sequentially chosen, one at a time, in order to observe how the model reacts and how this might affect the learning procedure. Initially, all the predictors were used to form the input vector for the training procedure of the network (Table 1).

Appendix B provides the structure of the models in Tables 4, 5 and 6 along with their sensitivity tables. The training sets used for those models consist of the observations of five (5) bad and fifty-six (56) good generators. The size of the test sets was always taken to be at 25% of the training set while the validation set consists of one (1) bad and four (4) good generators.

It is obvious from Table 4 that all the models behave well through the learning process and their training accuracy, computed on the test set taken from the training set, is very high (above 99%). Moreover, the prediction accuracy of the models, for the group of the good generators in the validation set, is also high (above 86%). On the other hand, it is noticeable that four (4) models could not predict the bad generator in the validation set at all, or to be more specific, they predicted that the generator was good. These models are even unable to predict generators 9 and 33, which are certainly bad because they had been replaced due to total failure and not preventively. The most likely explanation is that the patterns for the bad generators are too close to those to the good ones. It is also likely that the patterns of predictors for bad generators differ among the bad generators. There are a variety of potential causes for these differences. It is plausible that variability among aircrafts, among generators, or even among placement of IMD-HUMS acidometers could cause differences in readings from generator to generator. We also expect that bad generators with different failure causes will have different vibration patterns. Projection pursuit implemented by the statistical software Ggobi, used to gain a visual perspective of the relationship among the variables. Ggobi plots two-dimensional projections of multi-dimensional data and Figure 19 depicts some of the data leading to the assumption of a unique failure type for each “bad” generator.



```
  7:0.303,-0.209(107.28)
   10:0.252,-0.249(5.11)
    11:0.190,0.081(0.44)
    12:0.204,0.102(0.28)
   13:0.496,0.172(16.77)
   16:-0.274,0.020(2.20)
  17:0.053,-0.137(13.40)
   18:-0.034,0.183(7.92)
   19:-0.285,0.176(0.84)
20:-0.079,0.257(1144.94)
   21:0.389,0.346(55.63)
  22:-0.021,0.100(72.75)
   25:0.319,-0.477(0.05)
   26:-0.089,0.127(0.03)
   27:0.055,-0.007(0.01)
  28:-0.266,-0.185(0.01)
    29:0.140,0.538(0.94)
```

Figure 19. Ggobi screen for “bad” generators

Attempting to explore and understand further the models’ behavior, we tested five (5) new models to predict the condition of the two generators, 31 and 53, which were well predicted from models 3 and 5 of Table 5. Each model includes the whole database excluding those data of the validation set which are the “bad” generators in question and the “good” generators 4, 26, 42 and 66. For these models, we exclude from the training set the data of the pair referenced in table.

The prediction accuracy for generator 53 drops from 78.69 % in the single model 5, to a range of 77.05 % - 59.02 % in the paired models. This seems to indicate that all the bad generators contribute to the prediction (Table 6).

In contrast, for generator 31, the accuracy of models 8 and 9 drops less than 10% and for model 10 is almost 0%. This might indicate that generator 31 is more closely related to those generators that form the corresponding pairs (Table 7). Once again, we notice that the previous assumptions of close bad and good generator patterns and the possibility of unique reasons for generator failures are valid. Additionally, we observe that any change made on the models didn’t affect the accuracy of good generators.

MODEL	VALIDATION SET		TRAINING ACCURACY (%)	PREDICTION ACCURACY (%)
1	BAD	9	99.913	0
	GOOD	4		90.73
		26		96.85
		42		100
		66		100
2	BAD	22	99.721	0
	GOOD	4		100
		26		95.40
		42		100
		66		100
3	BAD	31	99.708	89.74
	GOOD	4		100
		26		95.66
		42		100
		66		100
4	BAD	33	99.821	0
	GOOD	4		100
		26		99.74
		42		100
		66		100
5	BAD	53	99.448	78.69
	GOOD	4		100
		26		99.77
		42		100
		66		100
6	BAD	56	99.376	0
	GOOD	4		100
		26		86.20
		42		100
		66		100

Table 5. Models Predicting Single Generator

MODEL	VALIDATION SET		TRAINING ACCURACY (%)	FIRST PREDICTION (%)	SECOND PREDICTION (%)
7	BAD	53	89.976	78.69	77.05
		9		99.31	0
	GOOD	4		100	100
		26		97.77	100
		42		100	100
		66		100	100
8	BAD	53	99.707	78.69	72.13
		22		100	0
	GOOD	4		100	100
		26		97.77	99.47
		42		100	100
		66		100	100
9	BAD	53	99.478	78.69	59.02
		31		100	0.33
	GOOD	4		100	100
		26		97.77	95.66
		42		100	100
		66		100	100
10	BAD	53	99.728	78.69	78.69
		33		99.18	0
	GOOD	4		100	100
		26		97.77	99.87
		42		100	100
		66		100	100
11	BAD	53	99.640	78.69	70.49
		56		100	6.12
	GOOD	4		100	100
		26		97.77	98.69
		42		100	100
		66		100	100

Table 6. Models Predicting Pair of Generators Including 53

MODEL	VALIDATION SET		TRAINING ACCURACY (%)	FIRST PREDICTION (%)	SECOND PREDICTION (%)
12	BAD	31	99.830	89.74	97.68
		9		99.54	0
	GOOD	4		100	100
		26		95.66	96.85
		42		100	100
		66		100	100
13	BAD	31	99.795	89.74	7.62
		22		100	0
	GOOD	4		100	100
		26		95.66	96.98
		42		100	100
		66		100	100
14	BAD	31	99.780	89.74	1.99
		33		100	0
	GOOD	4		100	100
		26		95.66	99.87
		42		100	10
		66		100	100
15	BAD	31	99.478	89.74	0.33
		53		100	59.02
	GOOD	4		100	100
		26		95.66	95.66
		42		100	100
		66		100	100
16	BAD	31	99.509	89.74	82.12
		56		100	24.49
	GOOD	4		100	100
		26		95.66	98.29
		42		100	100
		66		100	100

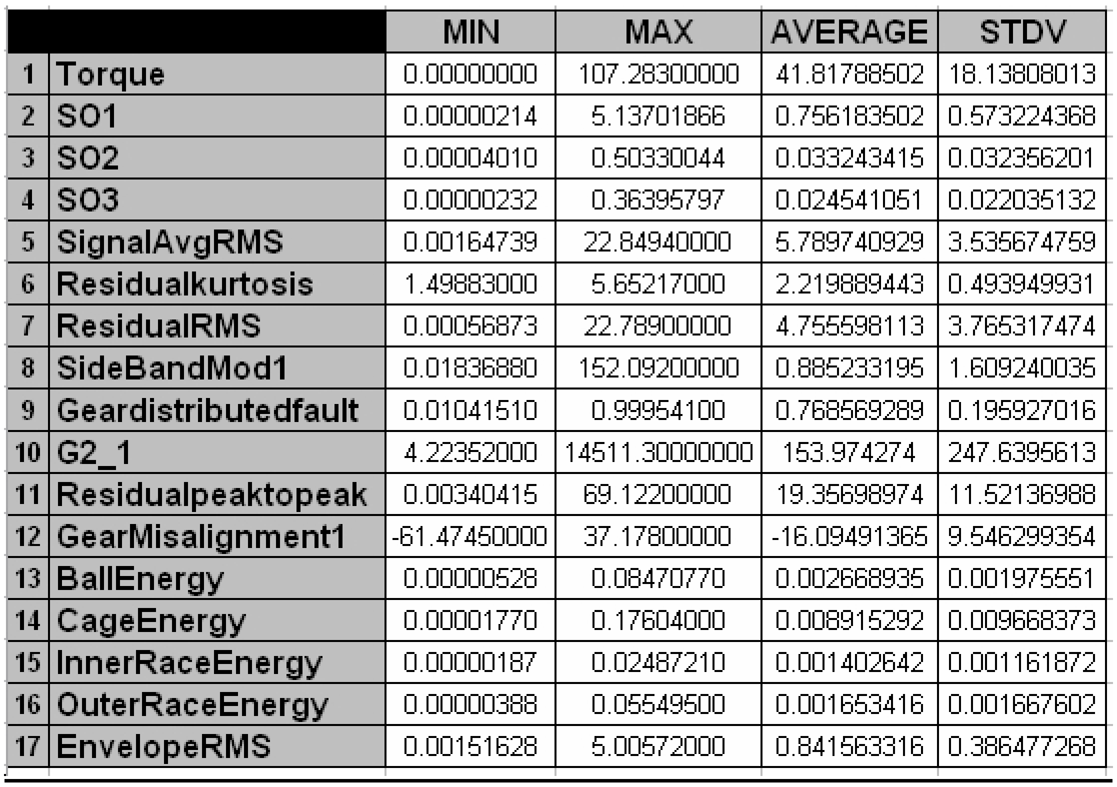
Table 7. Models Predicting Pair of Generators Including 31

B. ARTIFICIAL TRAINING SETS

Leaving out certain bad generators degrades the performance of the network when fitting the ANN. If including patterns of predictors for those bad generators is important for classification, then there may be other patterns of predictors for bad generators that have not yet been observed and are not included in the data set.

In this section an attempt is made to predict patterns of bad generators not included in the training set. First assume that a “bad” generator is any pattern which is not like the “good” generators in the data set. With the large number of good generators in the data set and the fact that the predictors of the good generators seem to clump together in two dimensions, this seems like a reasonable assumption. The ANN classifies as “bad” any pattern which is not like the good generators of the training set. To accomplish this, artificial records of “bad” generators are constructed and included in the training set. These artificial records are constructed by simulating values of their prediction variables using uniform distributions with lower and higher limits taken to be respectively the minimum and maximum value for that predicted variable. Table 8 gives these values for each predictor.

Uniform random numbers were generated in Excel. With Excel’s limited memory, only 65,000 uniform random numbers could be generated at one time. With seventeen (17) predictors, limiting the number of these artificial records to 65,000 makes for a sparse set of “not good” records.



		MIN	MAX	AVERAGE	STDV
1	Torque	0.00000000	107.28300000	41.81788502	18.13808013
2	SO1	0.00000214	5.13701866	0.756183502	0.573224368
3	SO2	0.00004010	0.50330044	0.033243415	0.032356201
4	SO3	0.00000232	0.36395797	0.024541051	0.022035132
5	SignalAvgRMS	0.00164739	22.84940000	5.789740929	3.535674759
6	Residualkurtosis	1.49883000	5.65217000	2.219889443	0.493949931
7	ResidualRMS	0.00056873	22.78900000	4.755598113	3.765317474
8	SideBandMod1	0.01836880	152.09200000	0.885233195	1.609240035
9	Geardistributedfault	0.01041510	0.99954100	0.768569289	0.195927016
10	G2_1	4.22352000	14511.30000000	153.974274	247.6395613
11	Residualpeaktopeak	0.00340415	69.12200000	19.35698974	11.52136988
12	GearMisalignment1	-61.47450000	37.17800000	-16.09491365	9.546299354
13	BallEnergy	0.00000528	0.08470770	0.002668935	0.001975551
14	CageEnergy	0.00001770	0.17604000	0.008915292	0.009668373
15	InnerRaceEnergy	0.00000187	0.02487210	0.001402642	0.001161872
16	OuterRaceEnergy	0.00000388	0.05549500	0.001653416	0.001667602
17	EnvelopeRMS	0.00151628	5.00572000	0.841563316	0.386477268

Table 8. Summary Statistics for each Predictor variable including the Minimum, Maximum, Average and Standard Deviation.

The following table (Table 9) presents the results of forecasting generator 9 using the above idea of training a uniformly distributed artificial set, combined along with the rest of generators regardless of their condition. We can see that propagating the artificial training set, even up to 1.3 million (1.3M) observations, the model still has no ability to predict the real condition of the bad generator in question. It does while still having excellent prediction accuracy on the group of good generators. The model starts to have encouraging prediction accuracy (11.72 %) after the artificial set reaches 2.6M observations but, at this point, it becomes less effective on the group of good generators. As the artificial set reaches a size of 2.75 M observations, it has excellent prediction accuracy (100 %) on the generator in question but now the predictions for the good group become very poor. This suggests once again that the patterns of good and bad generators are closely related. We can also conjecture that, because of the large number of predictor variables, we are seeing the so called "curse of dimensionality." This means that the complexity of the model grows exponentially with the dimension, rapidly outstripping the computational and memory storage capabilities of computers. For this data training more than 2.75M observations become infeasible.

The same method is applied to generator 33. The results are analogous and summarized in Table 10, while the structures of the networks are provided in Appendix C.

ARTIFICIAL SET	ALL DATA W/O GEN	MODEL ACCURACY	PREDICTION ACCURACY	
			BAD	GOOD
65,000	9	93.676 %	0%	Excellent
650,000		98.140 %	0%	
1.30 M		99.101 %	0%	
2.60 M		99.137 %	11.72%	Average
2.75 M		99.243	100%	Poor

Table 9. Artificial Training Set to Predict Gen 9

ARTIFICIAL SET	ALL DATA W/O GEN	MODEL ACCURACY	PREDICTION ACCURACY	
			FOR BAD	FOR GOOD
65,000	33	91.983%	0%	Excellent
650,000		98.053 %	0%	
1.30 M		98.915 %	0%	
2.60 M		99.209 %	0.41%	
2.75 M		99.537 %	0.40%	

Table 10. Artificial Training Set to Predict Gen 33

C. STEPWISE PREDICTORS USAGE

A simpler, but sometimes very effective, way of dealing with high-dimensional data is to reduce the number of dimensions. At this phase of the research we pursued a stepwise approach, similar to that of forward selection used for regression models. First were trained separate networks for each input variable. The network achieving the best training accuracy is then preserved. The effect of adding each of the rest of the inputs to this model in sequence is evaluated. This procedure is repeated for one, two, three, etc., predictors until the addition of extra predictors does not result in a significant improvement in model performance. Like any process, this approach has several disadvantages. The biggest are that it requires substantial computation and that it is also unable to capture the importance of certain combinations of predictors that might be insignificant on their own.

This phase of the research was the most time-consuming and many single and combined predictors were evaluated, aiming to find the model that better classified all the generators of the validation set. In the following table (Table 11), for space and time saving reasons, we present only one model as representative of those with the best ability to classify correctly.

MODEL TO	PREDICTORS	TRAINING ACCURACY	PREDICTION ACCURACY	
			BAD	GOOD (4,26,42,66)
Predict Gen9	ResidualRMS SideBandMod1 ResidualPeakToPeak BallEnergy InnerRaceEnergy	86.029 %	39.08%	> 85 %
Predict Gen22		84.036 %	26.49 %	> 83 %
Predict Gen31		87.345 %	5.36 %	> 72 %
Predict Gen33		85.377 %	40.82 %	> 90 %

MODEL TO	PREDICTORS	TRAINING ACCURACY	PREDICTION ACCURACY	
			BAD	GOOD (4,26,42,66)
Predict Gen53		87.321 %	36.07 %	> 73 %
Predict Gen56		86.386%	2.04 %	> 69 %

Table 11. Stepwise Good Generated Model

This last phase comes to verify that no matter which strategy was applied, the patterns of good and bad generators are so closely related that no model of the structure that we trained could distinguish the different pre-assigned condition of those generators. The time that the one model was a good classifier for the bad generator, it failed to predict the good ones and via versa; if the model performed well on the good generators, it failed to recognize the bad ones. The model summarized in Table 11 achieved the best overall performance for the database of this research. Even though the overall prediction accuracy appears low for the records of "bad" generators, the pattern of predicted probability that a record is "bad" $\hat{p}$, as a function of time is very different for "bad" generators than for "good" generators. As an example, Figures 20 and 21 give plots of $\hat{p}$ versus time for a "bad" and "good" generator respectively. The "good" generator has value of $\hat{p}$ close to zero, where as the "bad" generator $\hat{p}$ is vary considerably and show an increasing trend in time. Appendix D contains plots of $\hat{p}$ versus time for the rest of the generators in the validation set and sensitivity analysis of this model.

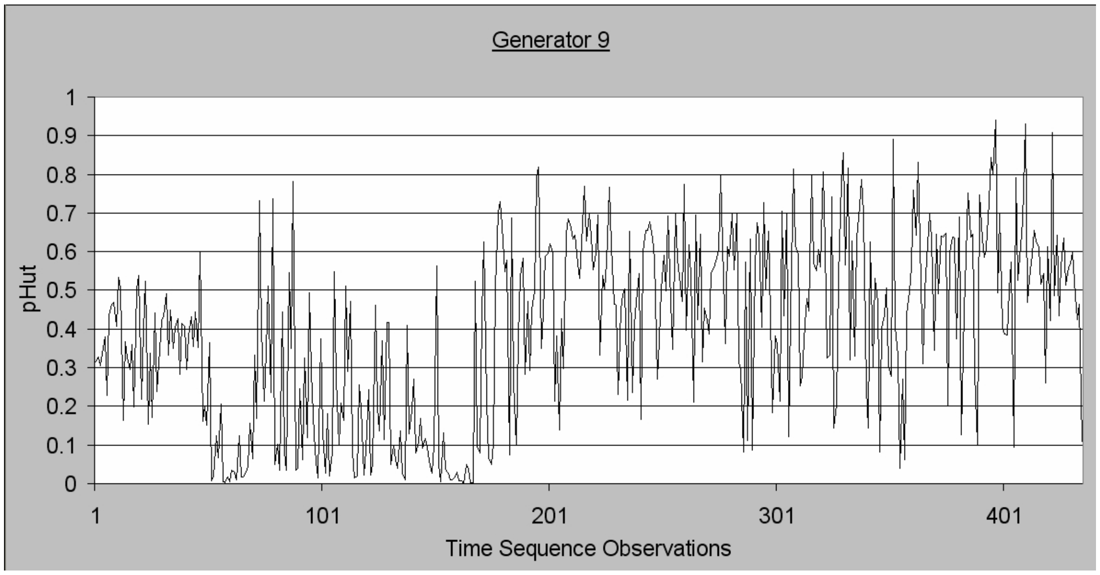


Figure 20. $\hat{p}$ from “bad” Generator 9

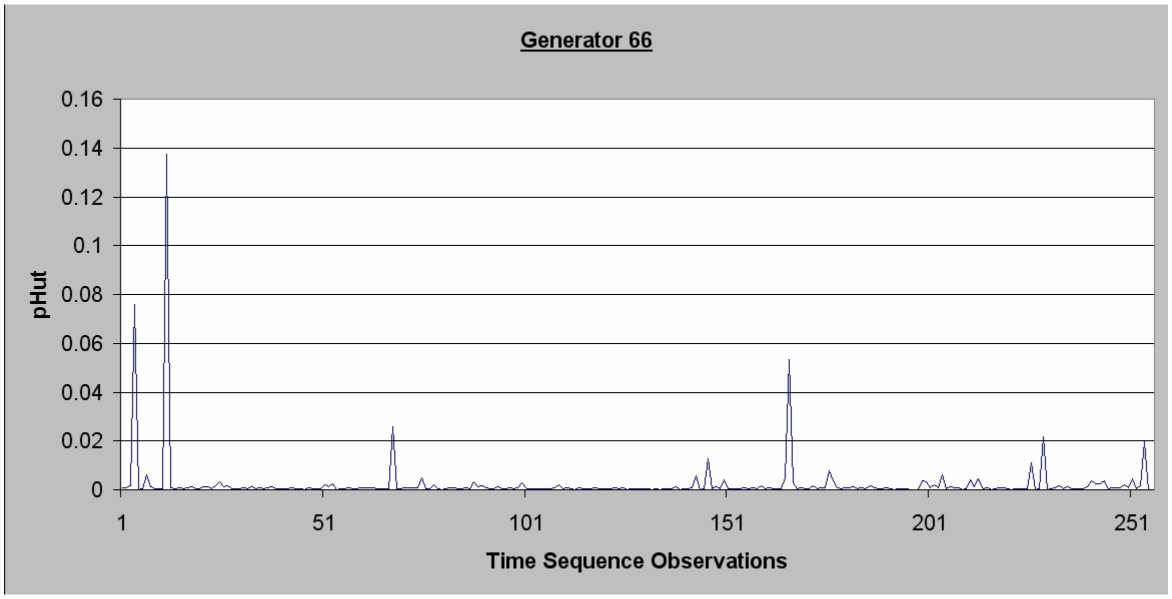


Figure 21. $\hat{p}$ from “good” Generator 66

V. CONCLUSIONS AND RECOMMENDATIONS

The emphasis in this thesis was to develop an ANN that would utilize the collected data from IMD-HUMS, manufactured by Goodrich Corporation, in order to discover patterns that would predict a potential failure of a UH-60L helicopter generator. Many different ANNs were evaluated for their success rate on this faulting diagnosis.

The first models that this research shaped were trained, tested, and evaluated using only the data collected by IMD-HUMS. The whole database was normalized and populated accordingly. One (1) bad and four (4) good generators, which were left out of the training and test sets, were used for validation purposes for those models. The method was applied sequentially to each bad generator, always maintaining the same group of good generators. Two of the six models were considered as good classifiers: with accuracy above 78.69% and 89.74%, respectively. The other four models had zero ability to classify the bad generators, although they all predicted very well the group of good generators.

The next phase of the experiment was to generate a uniformly distributed artificial data set, using it along with the original database to form a training/testing set for further classifications. Artificial sets up to 2.6M were created in an effort to capture the bad generator patterns. Because the learning phase was time -and resource- consuming, the researcher was limited to testing two generators, the ones originally replaced for failure (9 and 33). One model started to show some promising results by the time the researcher reached the computer’s limits. Obviously, the “curse of dimensionality” comes into play at this point of this research.

During the last portion of this experiment, the researcher tried to deal with the multi-dimensional problem and, at the same time, shape a model with good generalization behavior. A stepwise strategy was followed and many models were trained with various combinations of predictors. Once again, the produced classifiers were not generally successful in generator prediction.

Concluding, the researcher realized that in each phase of this experiment, the bad and good generators patterns are very closely related. This affects the learning procedure of the network by blocking its ability to build a model capable of classifying the good and bad generators concurrently. Additionally, it is possible that the reason of failure for each bad generator is unique, so that the size of this database and the structure of those models are not capable of capturing those patterns in a generalized form. On the other hand, this research exploits many paths, identifies various issues about the classification of the UH-60L helicopter generators, and finally comes up with models capable of classifying a big portion of the database in question.

ANNs are a category of artificial intelligence technology that mimics the human brain's skill at identifying patterns. In theory, ANNs are capable of approximating any continuous function. Such flexibility makes for a potentially powerful forecasting tool but the big number of parameters that must be selected makes the design process difficult. In practice, building an ANN forecasting model involves a lot of trial and error. Consequently, the objective of this thesis was to provide a practical, non-technical introduction to structure an ANN forecasting model using real operating data of UH-60L helicopters. The success of ANN applications for an individual researcher depends on three key factors. First, the researcher must have the time, patience, and resources to experiment. Second, the ANN software must allow automated routines, such as walk-forward testing, optimization of hidden neurons, and testing of input variable combinations—either through direct programming or the use of batch/script files. Third, the researcher must maintain a good set of records that lists all parameters for each network tested.

This research has verified that ANNs have a position in machinery condition monitoring and diagnostics. However, the limited nature of these results indicates that ANNs will not solve all machinery condition monitoring and diagnostics problems by themselves. They certainly will not completely replace conventional rule-based expert systems. Ultimately, it is anticipated that a symbiotic combination of these two technologies will provide the optimal solution to the machinery condition monitoring and diagnostics problem.

Further future work can be conducted on building models with more than one hidden layer which can explain more complex data, but which requires experience in manipulating the parameters of the utilized software and good a priori knowledge of the database itself. Furthermore, enhancing the current database with new real data, accompanied with proper maintenance records, will benefit and improve any future effort on predicting the generators’ conditions, regardless of the techniques that will be used by the researcher.

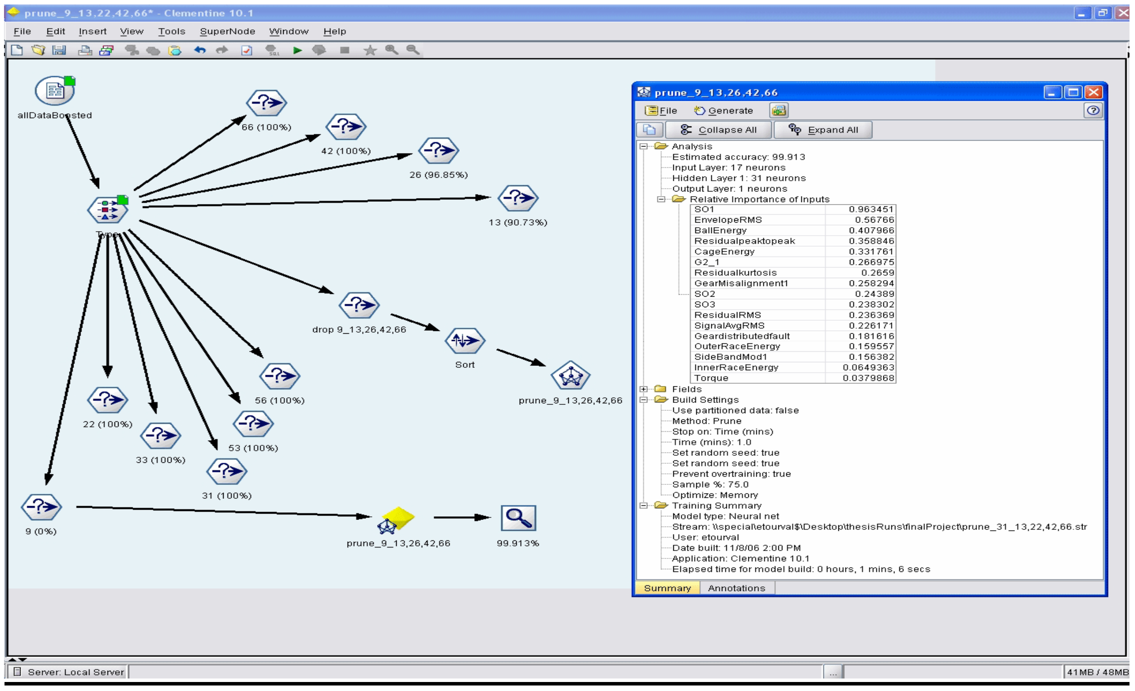
THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A

TAIL	#GENERATOR	OBSERVATIONS	FINAL CONDITION
9126351	1	526	Good
	35	526	
9226432	2	980	
	36	980	
9226435	3	313	
	37	313	
9226438	4	439	
	38	439	
9226439	5	282	
	39	282	
9226441	6	651	
	40	651	
9226443	7	476	
	41	476	
9226446	8	647	
	42	647	
9226450	9	435	Bad
	10	488	Good
	43	923	
9226453	11	825	
	44	825	
9226455	12	545	
	45	545	
9326477	13	917	
	46	917	
9326485	14	617	
	47	617	
9326493	15	487	
	48	487	
9326500	16	873	
	49	873	

TAIL	#GENERATOR	OBSERVATIONS	FINAL CONDITION
9326506	17	588	Good
	50	588	
9326507	18	423	
	51	423	
9326509	19	895	
	52	895	
9326515	20	648	
	53	61	Bad
	54	587	Good
9326516	21	553	
	55	553	
9326518	22	336	Bad
	23	12	Good
	56	98	Bad
	57	250	Good
9326519	24	621	
	58	621	
9326524	25	457	
	59	457	
9426530	26	761	
	60	761	
9426533	27	1077	
	61	1077	
9426534	28	298	
	62	298	
9426537	29	1102	
	63	1102	
9426545	30	254	
	64	254	
9426549	31	302	Bad
	32	287	Good
	65	589	
9926829	33	245	Bad
	34	11	Good
	66	256	

APPENDIX B

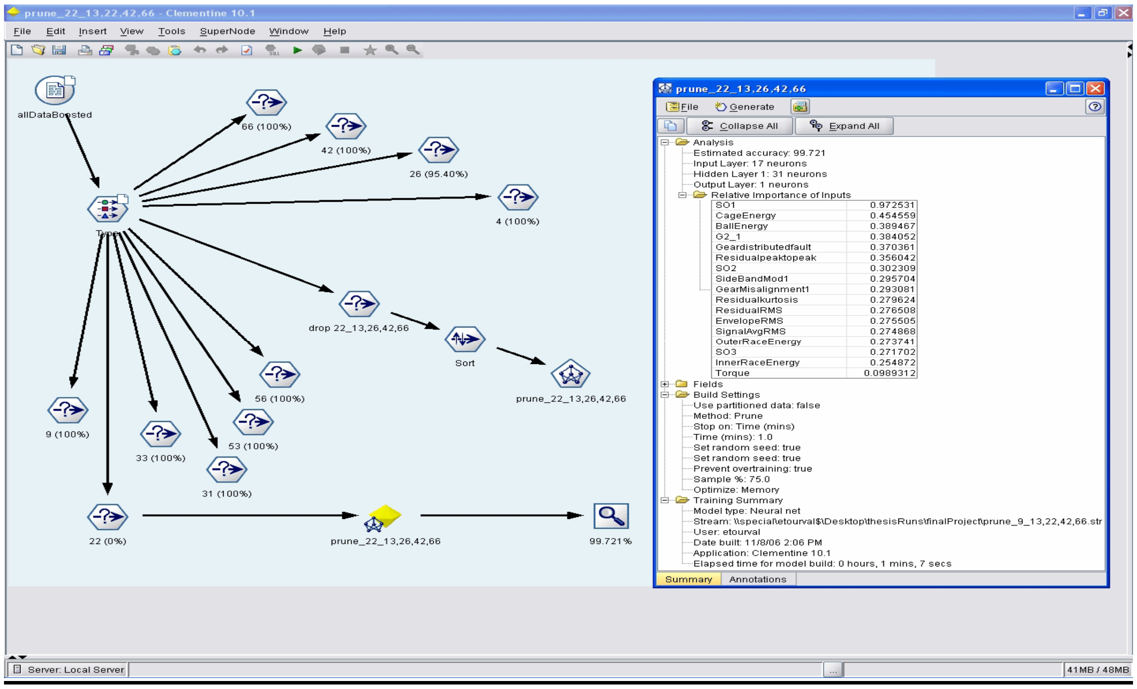


Analysis
Estimated accuracy: 99.913
Input Layer: 17 neurons
Hidden Layer 1: 31 neurons
Output Layer: 1 neurons
Relative Importance of Inputs

SO1	0.963451
EnvelopeRMS	0.56766
BallEnergy	0.407966
Residualpeaktopeak	0.358846
CageEnergy	0.331761
G2_1	0.266975
Residualkurtosis	0.2659
GearMisalignment1	0.258294
SO2	0.24389
SO3	0.238302
ResidualRMS	0.236369
SignalAvgRMS	0.226171
Geardistributedfault	0.181616
OuterRaceEnergy	0.159557
SideBandMod1	0.156382
InnerRaceEnergy	0.0649363
Torque	0.0379868

Fields
Build Settings
Use partitioned data: false
Method: Prune
Stop on: Time (mins)
Time (mins): 1.0
Set random seed: true
Set random seed: true
Prevent overtraining: true
Sample %: 75.0
Optimize: Memory
Training Summary
Model type: Neural net
Stream: \\special\etourval$\Desktop\thesisRuns\finalProject\prune_31_13,22,42,66.str
User: etourval
Date built: 11/8/06 2:00 PM
Application: Clementine 10.1
Elapsed time for model build: 0 hours, 1 mins, 6 secs

Figure 22. Model 1 (Predict Bad 09 and Good 4, 26, 42, 66)

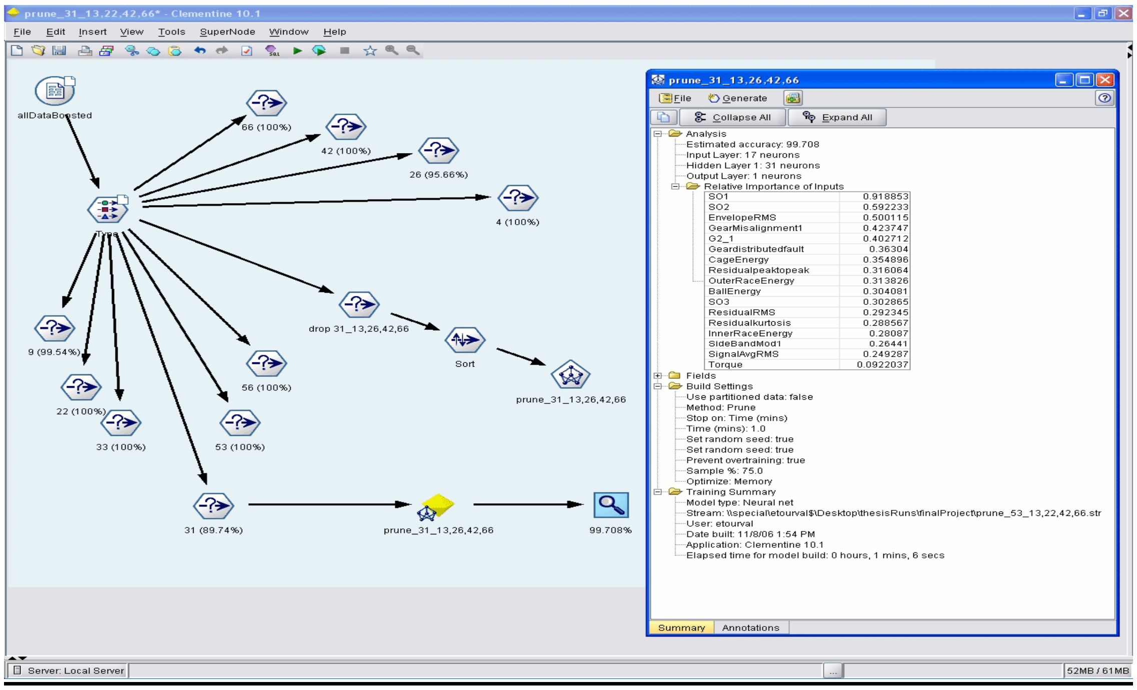


Analysis
Estimated accuracy: 99.721
Input Layer: 17 neurons
Hidden Layer 1: 31 neurons
Output Layer: 1 neurons
Relative Importance of Inputs

SO1	0.972531
CageEnergy	0.454559
BallEnergy	0.389467
G2_1	0.384052
Geardistributedfault	0.370361
Residualpeaktopeak	0.356042
SO2	0.302309
SideBandMod1	0.295704
GearMisalignment1	0.293081
Residualkurtosis	0.279624
ResidualRMS	0.276508
EnvelopeRMS	0.275505
SignalAvgRMS	0.274868
OuterRaceEnergy	0.273741
SO3	0.271702
InnerRaceEnergy	0.254872
Torque	0.0989312

Fields
Build Settings
Use partitioned data: false
Method: Prune
Stop on: Time (mins)
Time (mins): 1.0
Set random seed: true
Set random seed: true
Prevent overtraining: true
Sample %: 75.0
Optimize: Memory
Training Summary
Model type: Neural net
Stream: \\special\etourval$\Desktop\thesisRuns\finalProject\prune_9_13,22,42,66.str
User: etourval
Date built: 11/8/06 2:06 PM
Application: Clementine 10.1
Elapsed time for model build: 0 hours, 1 mins, 7 secs

Figure 23. Model 2 (Predict Bad 22 and Good 4, 26, 42, 66)

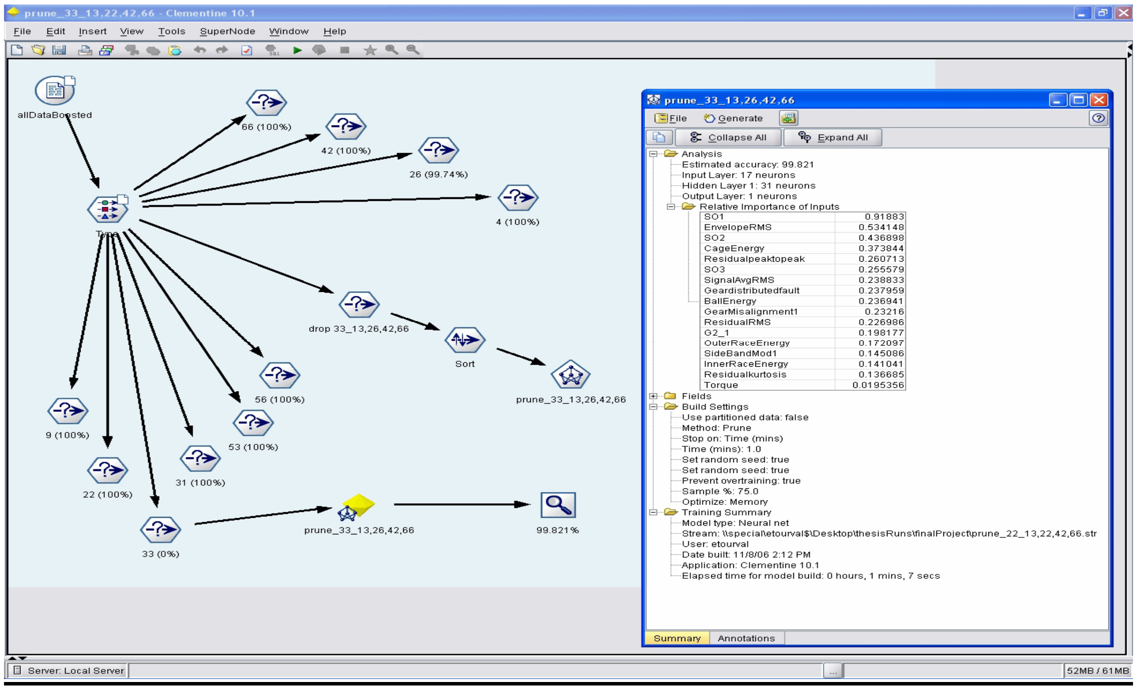


```
Analysis
    Estimated accuracy: 99.708
    Input Layer: 17 neurons
    Hidden Layer 1: 31 neurons
    Output Layer: 1 neurons
    Relative Importance of Inputs
```

SO1	0.918853
SO2	0.592233
EnvelopeRMS	0.500115
GearMisalignment1	0.423747
G2_1	0.402712
Geardistributedfault	0.36304
CageEnergy	0.354896
Residualpeaktopeak	0.316064
OuterRaceEnergy	0.313826
BallEnergy	0.304081
SO3	0.302865
ResidualRMS	0.292345
Residualkurtosis	0.288567
InnerRaceEnergy	0.28087
SideBandMod1	0.26441
SignalAvgRMS	0.249287
Torque	0.0922037

```
Fields
Build Settings
    Use partitioned data: false
    Method: Prune
    Stop on: Time (mins)
    Time (mins): 1.0
    Set random seed: true
    Set random seed: true
    Prevent overtraining: true
    Sample %: 75.0
    Optimize: Memory
Training Summary
    Model type: Neural net
    Stream: \\special\etourval$\Desktop\thesisRuns\finalProject\prune_53_13,22,42,66.str
    User: etourval
    Date built: 11/8/06 1:54 PM
    Application: Clementine 10.1
    Elapsed time for model build: 0 hours, 1 mins, 6 secs
```

Figure 24. Model 3 (Predict Bad 31 and Good 4, 26, 42, 66)

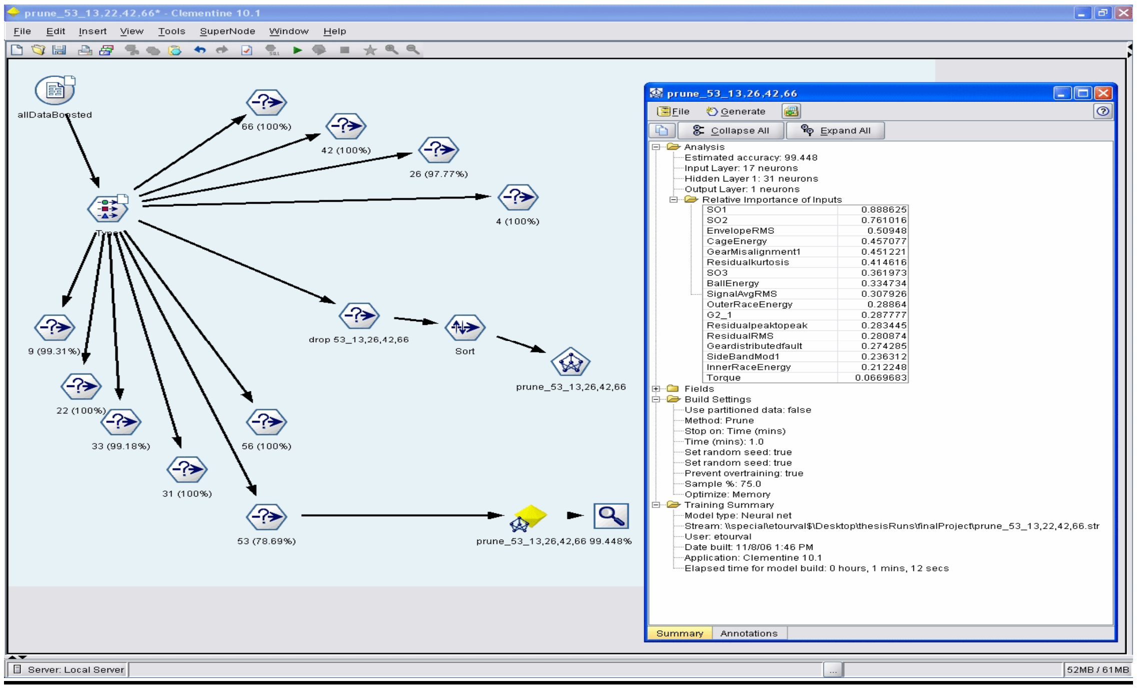


```
Analysis
    Estimated accuracy: 99.821
    Input Layer: 17 neurons
    Hidden Layer 1: 31 neurons
    Output Layer: 1 neurons
    Relative Importance of Inputs
```

SO1	0.91883
EnvelopeRMS	0.534148
SO2	0.436898
CageEnergy	0.373844
Residualpeaktopeak	0.260713
SO3	0.255579
SignalAvgRMS	0.238833
Geardistributedfault	0.237959
BallEnergy	0.236941
GearMisalignment1	0.23216
ResidualRMS	0.226986
G2_1	0.198177
OuterRaceEnergy	0.172097
SideBandMod1	0.145086
InnerRaceEnergy	0.141041
Residualkurtosis	0.136685
Torque	0.0195356

```
Fields
Build Settings
    Use partitioned data: false
    Method: Prune
    Stop on: Time (mins)
    Time (mins): 1.0
    Set random seed: true
    Set random seed: true
    Prevent overtraining: true
    Sample %: 75.0
    Optimize: Memory
Training Summary
    Model type: Neural net
    Stream: \\special\etourval$\Desktop\thesisRuns\finalProject\prune_22_13,22,42,66.str
    User: etourval
    Date built: 11/8/06 2:12 PM
    Application: Clementine 10.1
    Elapsed time for model build: 0 hours, 1 mins, 7 secs
```

Figure 25. Model 4 (Predict Bad 33 and Good 4, 26, 42, 66)

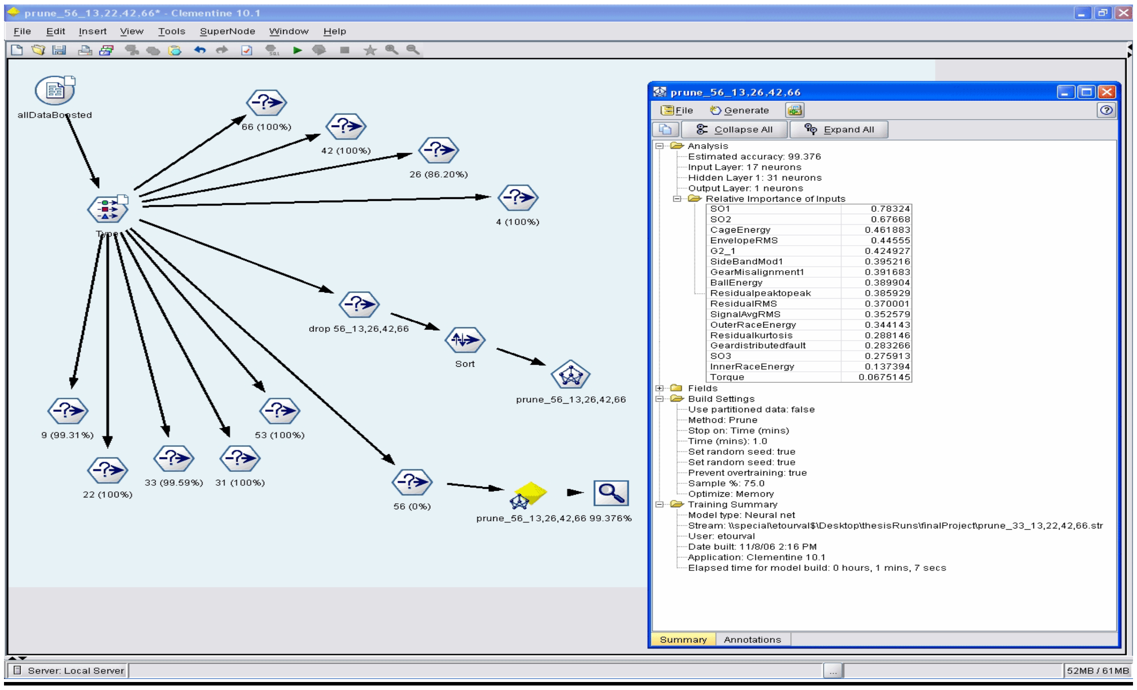


Analysis
Estimated accuracy: 99.448
Input Layer: 17 neurons
Hidden Layer 1: 31 neurons
Output Layer: 1 neurons
Relative Importance of Inputs

SO1	0.888625
SO2	0.761016
EnvelopeRMS	0.50948
CageEnergy	0.457077
GearMisalignment1	0.451221
Residualkurtosis	0.414616
SO3	0.361973
BallEnergy	0.334734
SignalAvgRMS	0.307926
OuterRaceEnergy	0.28864
G2_1	0.287777
Residualpeaktopeak	0.283445
ResidualRMS	0.280874
Geardistributedfault	0.274285
SideBandMod1	0.236312
InnerRaceEnergy	0.212248
Torque	0.0669683

Fields
Build Settings
Use partitioned data: false
Method: Prune
Stop on: Time (mins)
Time (mins): 1.0
Set random seed: true
Set random seed: true
Prevent overtraining: true
Sample %: 75.0
Optimize: Memory
Training Summary
Model type: Neural net
Stream: \\special\etourval$\Desktop\thesisRuns\finalProject\prune_53_13,22,42,66.str
User: etourval
Date built: 11/8/06 1:46 PM
Application: Clementine 10.1
Elapsed time for model build: 0 hours, 1 mins, 12 secs

Figure 26. Model 5 (Predict Bad 53 and Good 4, 26, 42, 66)

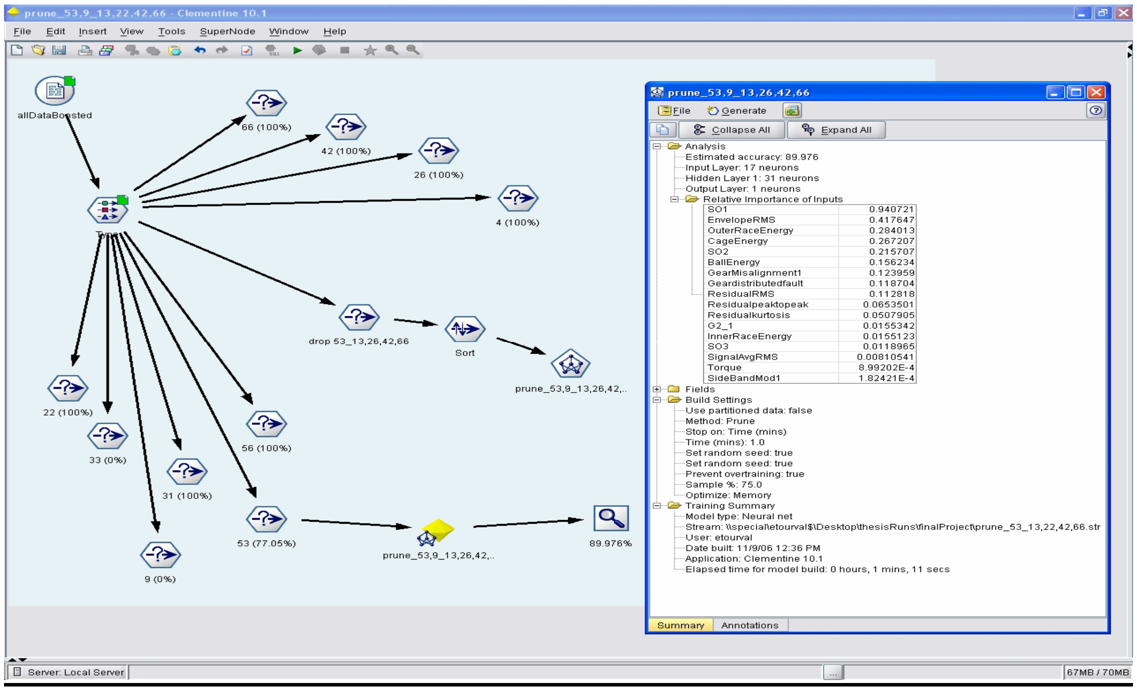


Analysis
Estimated accuracy: 99.376
Input Layer: 17 neurons
Hidden Layer 1: 31 neurons
Output Layer: 1 neurons
Relative Importance of Inputs

SO1	0.78324
SO2	0.67668
CageEnergy	0.461883
EnvelopeRMS	0.44555
G2_1	0.424927
SideBandMod1	0.395216
GearMisalignment1	0.391683
BallEnergy	0.389904
Residualpeaktopeak	0.385929
ResidualRMS	0.370001
SignalAvgRMS	0.352579
OuterRaceEnergy	0.344143
Residualkurtosis	0.288146
Geardistributedfault	0.283266
SO3	0.275913
InnerRaceEnergy	0.137394
Torque	0.0675145

Fields
Build Settings
Use partitioned data: false
Method: Prune
Stop on: Time (mins)
Time (mins): 1.0
Set random seed: true
Set random seed: true
Prevent overtraining: true
Sample %: 75.0
Optimize: Memory
Training Summary
Model type: Neural net
Stream: \\special\etourval$\Desktop\thesisRuns\finalProject\prune_33_13,22,42,66.str
User: etourval
Date built: 11/8/06 2:16 PM
Application: Clementine 10.1
Elapsed time for model build: 0 hours, 1 mins, 7 secs

Figure 27. Model 6 (Predict Bad 56 and Good 4, 26, 42, 66)

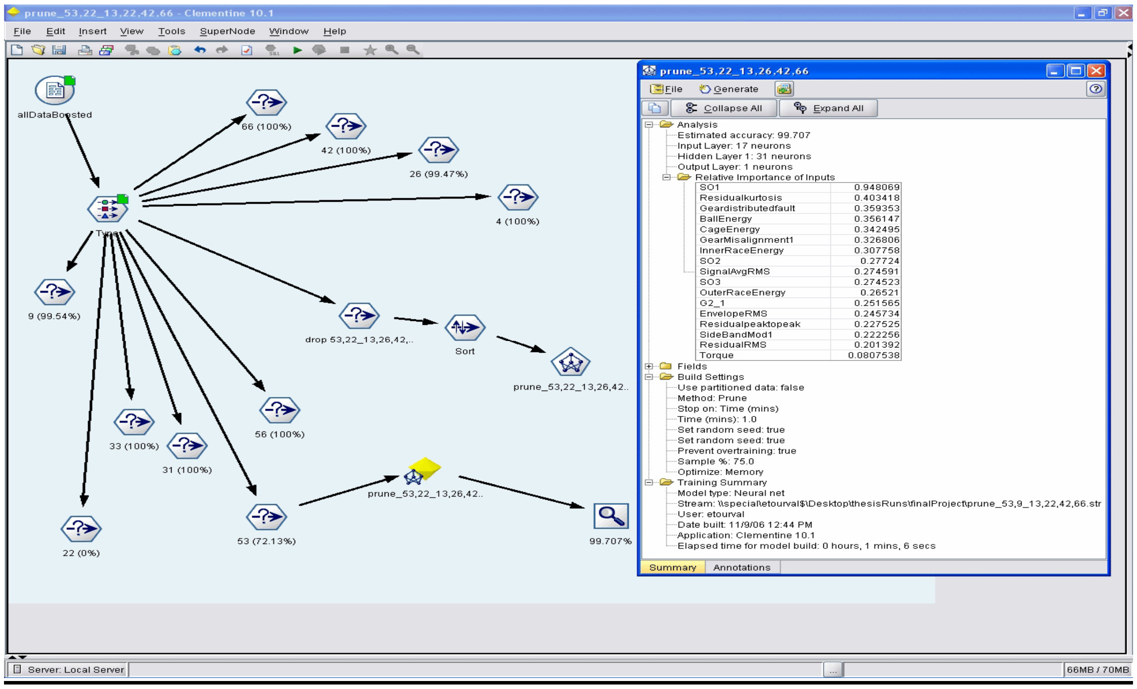


```
Analysis
  Estimated accuracy: 89.976
  Input Layer: 17 neurons
  Hidden Layer 1: 31 neurons
  Output Layer: 1 neurons
  Relative Importance of Inputs
```

SO1	0.940721
EnvelopeRMS	0.417647
OuterRaceEnergy	0.284013
CageEnergy	0.267207
SO2	0.215707
BallEnergy	0.156234
GearMisalignment1	0.123959
Geardistributedfault	0.118704
ResidualRMS	0.112818
Residualpeaktopeak	0.0653501
Residualkurtosis	0.0507905
G2_1	0.0155342
InnerRaceEnergy	0.0155123
SO3	0.0118965
SignalAvgRMS	0.00810541
Torque	8.99202E-4
SideBandMod1	1.82421E-4

```
Fields
Build Settings
  Use partitioned data: false
  Method: Prune
  Stop on: Time (mins)
  Time (mins): 1.0
  Set random seed: true
  Set random seed: true
  Prevent overtraining: true
  Sample %: 75.0
  Optimize: Memory
Training Summary
  Model type: Neural net
  Stream: \\special\etourval$\Desktop\thesisRuns\finalProject\prune_53_13,22,42,66.str
  User: etourval
  Date built: 11/9/06 12:36 PM
  Application: Clementine 10.1
  Elapsed time for model build: 0 hours, 1 mins, 11 secs
```

Figure 28. Model 7 (Predict Bad 53, 9 and Good 4, 26, 42, 66)

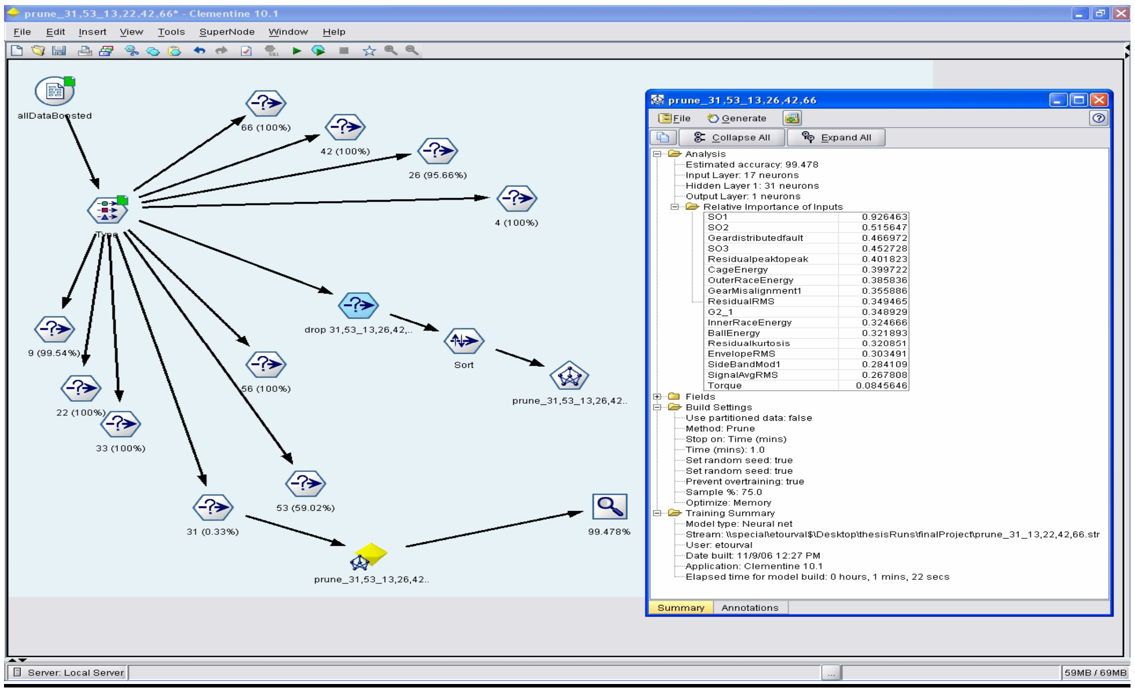


```
Analysis
  Estimated accuracy: 99.707
  Input Layer: 17 neurons
  Hidden Layer 1: 31 neurons
  Output Layer: 1 neurons
  Relative Importance of Inputs
```

SO1	0.948069
Residualkurtosis	0.403418
Geardistributedfault	0.359353
BallEnergy	0.356147
CageEnergy	0.342495
GearMisalignment1	0.326806
InnerRaceEnergy	0.307758
SO2	0.27724
SignalAvgRMS	0.274591
SO3	0.274523
OuterRaceEnergy	0.26521
G2_1	0.251565
EnvelopeRMS	0.245734
Residualpeaktopeak	0.227525
SideBandMod1	0.222256
ResidualRMS	0.201392
Torque	0.0807538

```
Fields
Build Settings
  Use partitioned data: false
  Method: Prune
  Stop on: Time (mins)
  Time (mins): 1.0
  Set random seed: true
  Set random seed: true
  Prevent overtraining: true
  Sample %: 75.0
  Optimize: Memory
Training Summary
  Model type: Neural net
  Stream: \\special\etourval$\Desktop\thesisRuns\finalProject\prune_53,9_13,22,42,66.str
  User: etourval
  Date built: 11/9/06 12:44 PM
  Application: Clementine 10.1
  Elapsed time for model build: 0 hours, 1 mins, 6 secs
```

Figure 29. Model 8 (Predict Bad 53, 22 and Good 4, 26, 42, 66)

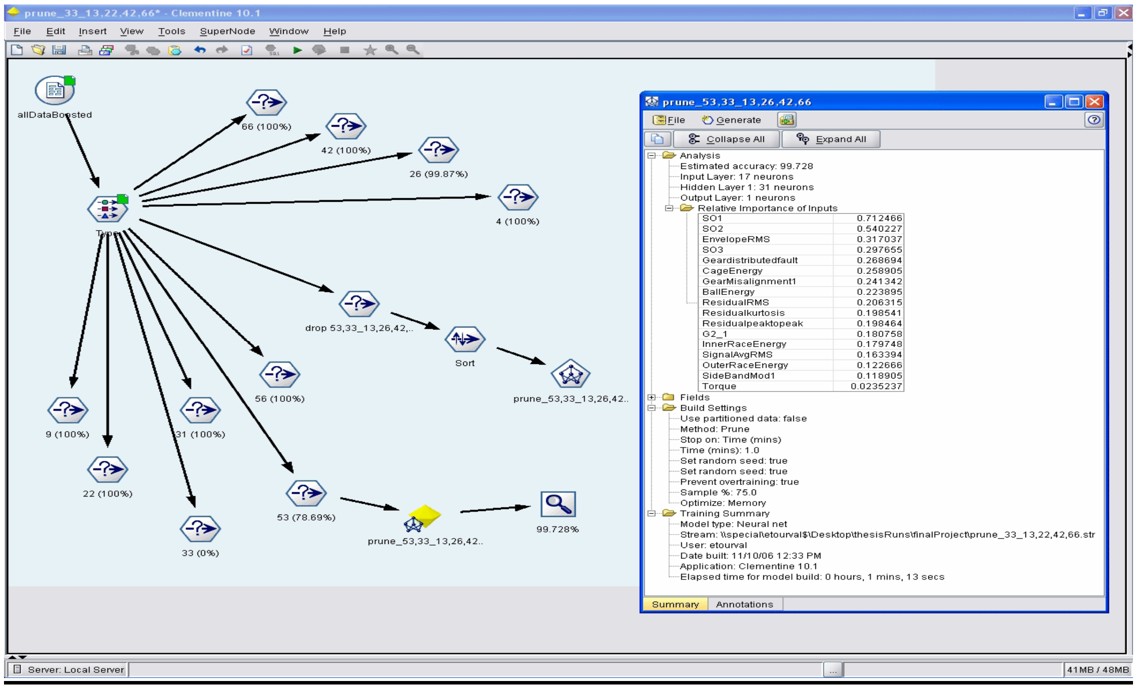


```
Analysis
  Estimated accuracy: 99.478
  Input Layer: 17 neurons
  Hidden Layer 1: 31 neurons
  Output Layer: 1 neurons
  Relative Importance of Inputs
```

SO1	0.926463
SO2	0.515647
Geardistributedfault	0.466972
SO3	0.452728
Residualpeaktopeak	0.401823
CageEnergy	0.399722
OuterRaceEnergy	0.385836
GearMisalignment1	0.355886
ResidualRMS	0.349465
G2_1	0.348929
InnerRaceEnergy	0.324666
BallEnergy	0.321893
Residualkurtosis	0.320851
EnvelopeRMS	0.303491
SideBandMod1	0.284109
SignalAvgRMS	0.267808
Torque	0.0845646

```
Fields
Build Settings
  Use partitioned data: false
  Method: Prune
  Stop on: Time (mins)
  Time (mins): 1.0
  Set random seed: true
  Set random seed: true
  Prevent overtraining: true
  Sample %: 75.0
  Optimize: Memory
Training Summary
  Model type: Neural net
  Stream: \\special\etourval$\Desktop\thesisRuns\finalProject\prune_31_13,22,42,66.str
  User: etourval
  Date built: 11/9/06 12:27 PM
  Application: Clementine 10.1
  Elapsed time for model build: 0 hours, 1 mins, 22 secs
```

Figure 30. Model 9 (Predict Bad 53, 31 and Good 4, 26, 42, 66)

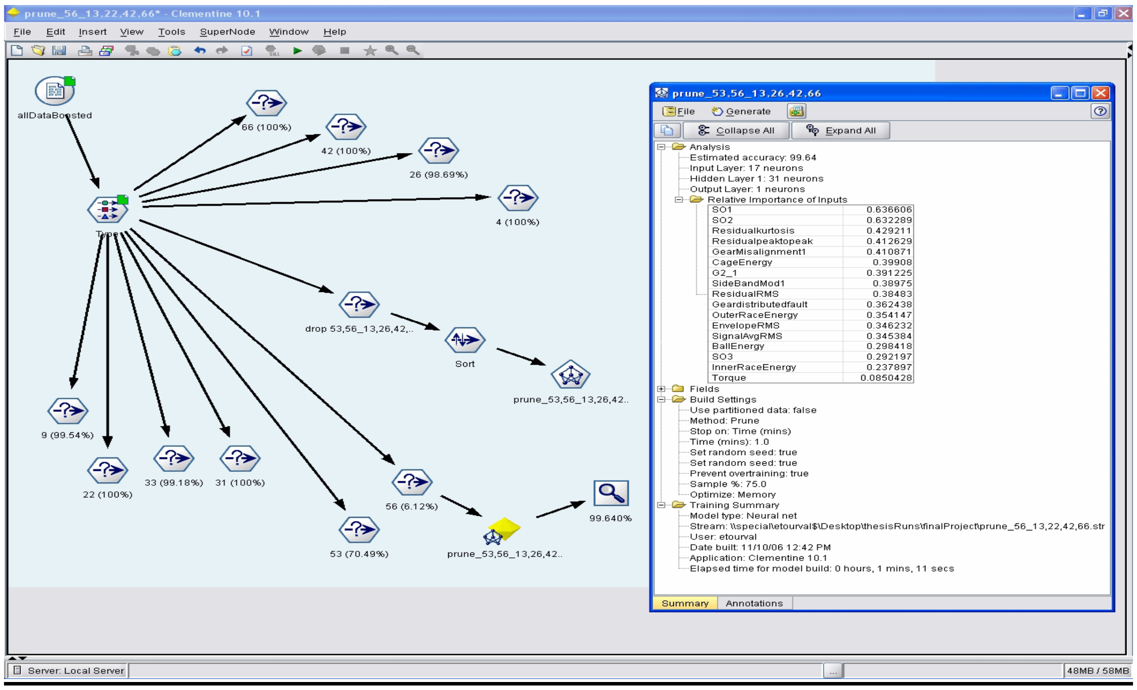


```
Analysis
  Estimated accuracy: 99.728
  Input Layer: 17 neurons
  Hidden Layer 1: 31 neurons
  Output Layer: 1 neurons
  Relative Importance of Inputs
```

SO1	0.712466
SO2	0.540227
EnvelopeRMS	0.317037
SO3	0.297655
Geardistributedfault	0.268694
CageEnergy	0.258905
GearMisalignment1	0.241342
BallEnergy	0.223895
ResidualRMS	0.206315
Residualkurtosis	0.198541
Residualpeaktopeak	0.198464
G2_1	0.180758
InnerRaceEnergy	0.179748
SignalAvgRMS	0.163394
OuterRaceEnergy	0.122666
SideBandMod1	0.118905
Torque	0.0235237

```
Fields
Build Settings
  Use partitioned data: false
  Method: Prune
  Stop on: Time (mins)
  Time (mins): 1.0
  Set random seed: true
  Set random seed: true
  Prevent overtraining: true
  Sample %: 75.0
  Optimize: Memory
Training Summary
  Model type: Neural net
  Stream: \\special\etourval$\Desktop\thesisRuns\finalProject\prune_33_13,22,42,66.str
  User: etourval
  Date built: 11/10/06 12:33 PM
  Application: Clementine 10.1
  Elapsed time for model build: 0 hours, 1 mins, 13 secs
```

Figure 31. Model 10 (Predict Bad 53, 33 and Good 4, 26, 42, 66)



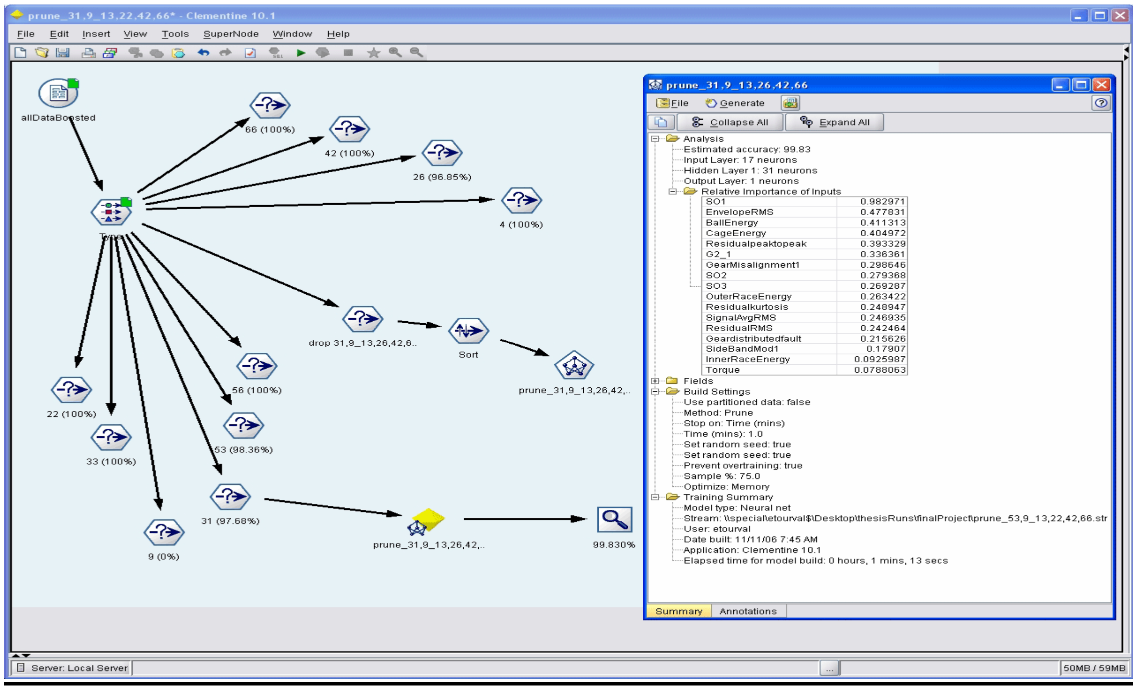
prune_53,56_13,26,42,66

```
Analysis
  Estimated accuracy: 99.64
  Input Layer: 17 neurons
  Hidden Layer 1: 31 neurons
  Output Layer: 1 neurons
  Relative Importance of Inputs
```

SO1	0.636606
SO2	0.632289
Residualkurtosis	0.429211
Residualpeaktopeak	0.412629
GearMisalignment1	0.410871
CageEnergy	0.39908
G2_1	0.391225
SideBandMod1	0.38975
ResidualRMS	0.38483
Geardistributedfault	0.362438
OuterRaceEnergy	0.354147
EnvelopeRMS	0.346232
SignalAvgRMS	0.345384
BallEnergy	0.298418
SO3	0.292197
InnerRaceEnergy	0.237897
Torque	0.0850428

```
Fields
Build Settings
  Use partitioned data: false
  Method: Prune
  Stop on: Time (mins)
  Time (mins): 1.0
  Set random seed: true
  Set random seed: true
  Prevent overtraining: true
  Sample %: 75.0
  Optimize: Memory
Training Summary
  Model type: Neural net
  Stream: \\special\etourval$\Desktop\thesisRuns\finalProject\prune_56_13,22,42,66.str
  User: etourval
  Date built: 11/10/06 12:42 PM
  Application: Clementine 10.1
  Elapsed time for model build: 0 hours, 1 mins, 11 secs
```

Figure 32. Model 11 (Predict Bad 53, 56 and Good 4, 26, 42, 66)



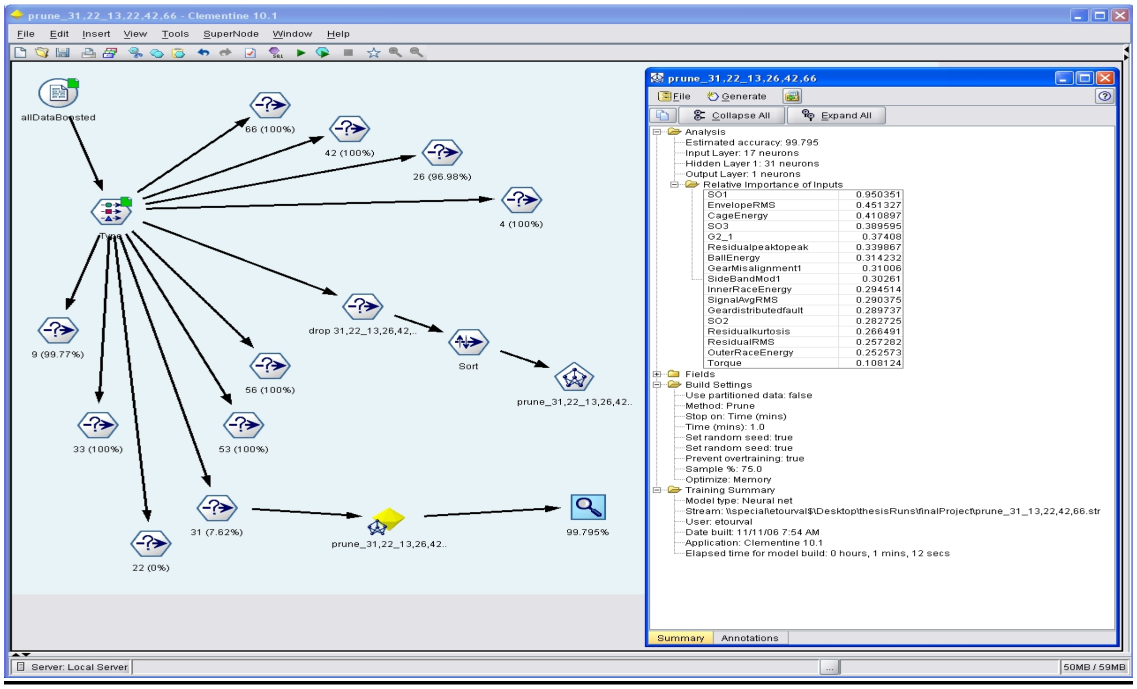
prune_31,9_13,26,42,66

```
Analysis
  Estimated accuracy: 99.83
  Input Layer: 17 neurons
  Hidden Layer 1: 31 neurons
  Output Layer: 1 neurons
  Relative Importance of Inputs
```

SO1	0.982971
EnvelopeRMS	0.477831
BallEnergy	0.411313
CageEnergy	0.404972
Residualpeaktopeak	0.393329
G2_1	0.336361
GearMisalignment1	0.298646
SO2	0.279368
SO3	0.269287
OuterRaceEnergy	0.263422
Residualkurtosis	0.248947
SignalAvgRMS	0.246935
ResidualRMS	0.242464
Geardistributedfault	0.215626
SideBandMod1	0.17907
InnerRaceEnergy	0.0925987
Torque	0.0788063

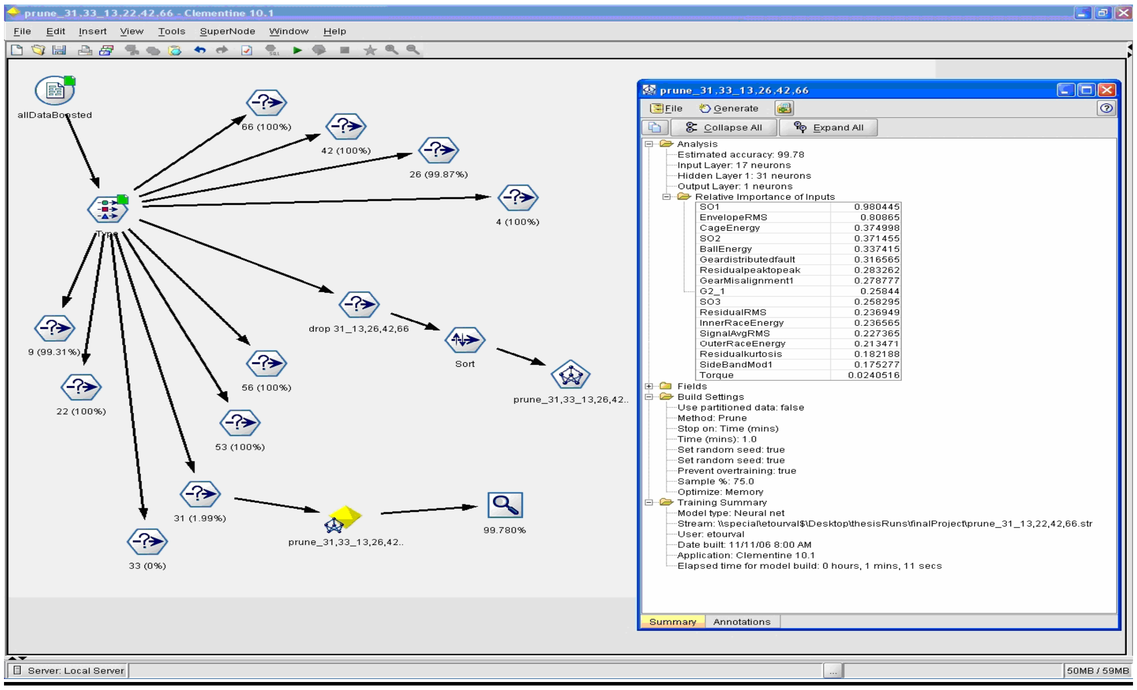
```
Fields
Build Settings
  Use partitioned data: false
  Method: Prune
  Stop on: Time (mins)
  Time (mins): 1.0
  Set random seed: true
  Set random seed: true
  Prevent overtraining: true
  Sample %: 75.0
  Optimize: Memory
Training Summary
  Model type: Neural net
  Stream: \\special\etourval$\Desktop\thesisRuns\finalProject\prune_53,9_13,22,42,66.str
  User: etourval
  Date built: 11/11/06 7:45 AM
  Application: Clementine 10.1
  Elapsed time for model build: 0 hours, 1 mins, 13 secs
```

Figure 33. Model 12 (Predict Bad 31, 9 and Good 4 ,26, 42, 66)



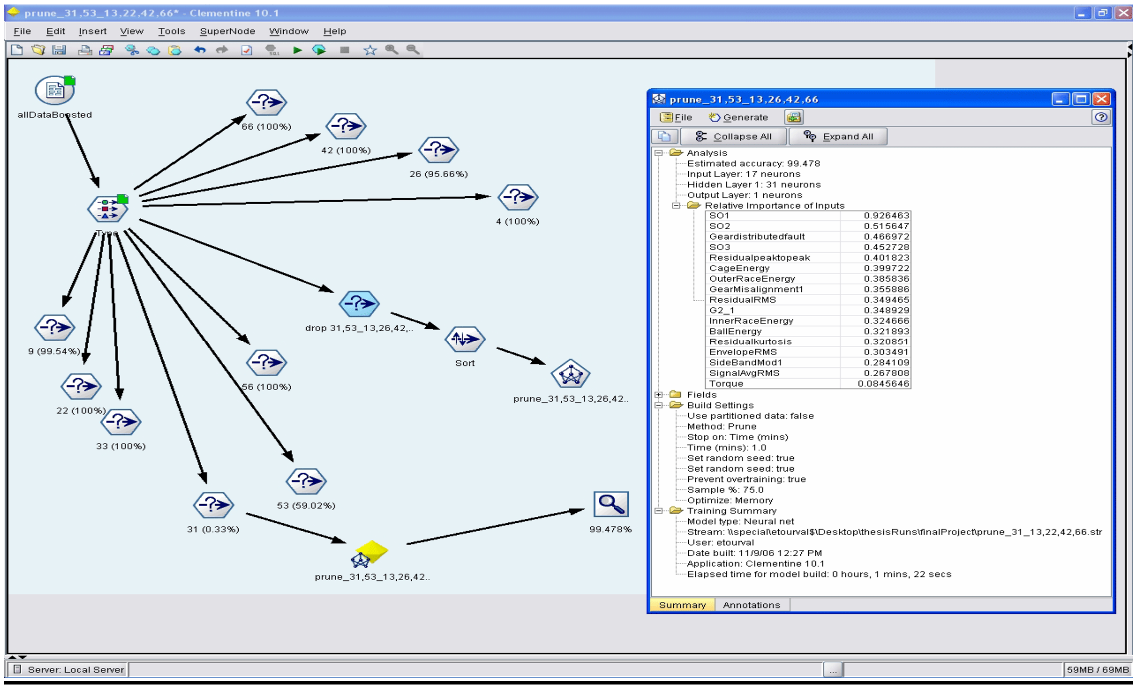
```
prune_31,22_13,26,42,66
Analysis
    Estimated accuracy: 99.795
    Input Layer: 17 neurons
    Hidden Layer 1: 31 neurons
    Output Layer: 1 neurons
    Relative Importance of Inputs
        SO1                   0.950351
        EnvelopeRMS           0.451327
        CageEnergy            0.410897
        SO3                   0.389595
        G2_1                   0.37408
        Residualpeaktopeak    0.339867
        BallEnergy            0.314232
        GearMisalignment1      0.31006
        SideBandMod1           0.30261
        InnerRaceEnergy       0.294514
        SignalAvgRMS          0.290375
        Geardistributedfault  0.289737
        SO2                   0.282725
        Residualkurtosis      0.266491
        ResidualRMS           0.257282
        OuterRaceEnergy       0.252573
        Torque                0.108124
Fields
Build Settings
    Use partitioned data: false
    Method: Prune
    Stop on: Time (mins)
    Time (mins): 1.0
    Set random seed: true
    Set random seed: true
    Prevent overtraining: true
    Sample %: 75.0
    Optimize: Memory
Training Summary
    Model type: Neural net
    Stream: \\special\etourval$\Desktop\thesisRuns\finalProject\prune_31_13,22,42,66.str
    User: etourval
    Date built: 11/11/06 7:54 AM
    Application: Clementine 10.1
    Elapsed time for model build: 0 hours, 1 mins, 12 secs
```

Figure 34. Model 13 (Predict Bad 31, 22 and Good 4, 26, 42, 66)



```
prune_31,33_13,26,42,66
Analysis
    Estimated accuracy: 99.78
    Input Layer: 17 neurons
    Hidden Layer 1: 31 neurons
    Output Layer: 1 neurons
    Relative Importance of Inputs
        SO1                   0.980445
        EnvelopeRMS            0.80865
        CageEnergy            0.374998
        SO2                   0.371455
        BallEnergy            0.337415
        Geardistributedfault  0.316565
        Residualpeaktopeak    0.283262
        GearMisalignment1     0.278777
        G2_1                   0.25844
        SO3                   0.258295
        ResidualRMS           0.236949
        InnerRaceEnergy       0.236565
        SignalAvgRMS          0.227365
        OuterRaceEnergy       0.213471
        Residualkurtosis      0.182188
        SideBandMod1          0.175277
        Torque               0.0240516
Fields
Build Settings
    Use partitioned data: false
    Method: Prune
    Stop on: Time (mins)
    Time (mins): 1.0
    Set random seed: true
    Set random seed: true
    Prevent overtraining: true
    Sample %: 75.0
    Optimize: Memory
Training Summary
    Model type: Neural net
    Stream: \\special\etourval$\Desktop\thesisRuns\finalProject\prune_31_13,22,42,66.str
    User: etourval
    Date built: 11/11/06 8:00 AM
    Application: Clementine 10.1
    Elapsed time for model build: 0 hours, 1 mins, 11 secs
```

Figure 35. Model 14 (Predict Bad 31, 33 and Good 4, 26, 42, 66)

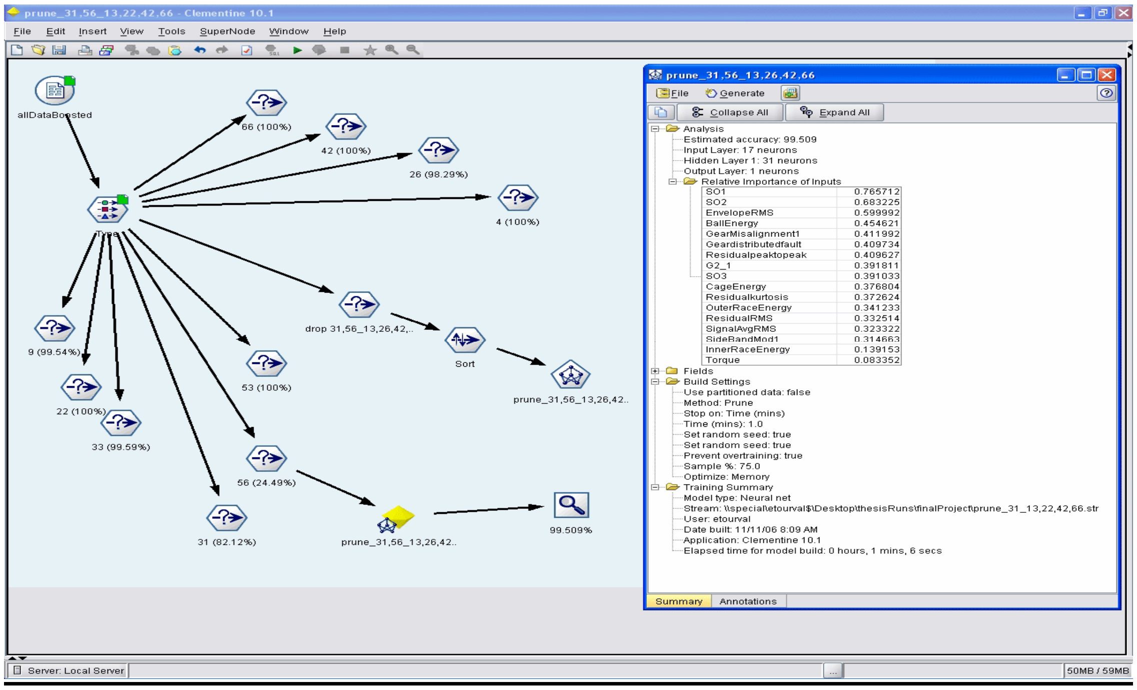


```
Analysis
    Estimated accuracy: 99.478
    Input Layer: 17 neurons
    Hidden Layer 1: 31 neurons
    Output Layer: 1 neurons
    Relative Importance of Inputs
```

SO1	0.926463
SO2	0.515647
Geardistributedfault	0.466972
SO3	0.452728
Residualpeaktopeak	0.401823
CageEnergy	0.399722
OuterRaceEnergy	0.385836
GearMisalignment1	0.355886
ResidualRMS	0.349465
G2_1	0.348929
InnerRaceEnergy	0.324666
BallEnergy	0.321893
Residualkurtosis	0.320851
EnvelopeRMS	0.303491
SideBandMod1	0.284109
SignalAvgRMS	0.267808
Torque	0.0845646

```
Fields
Build Settings
    Use partitioned data: false
    Method: Prune
    Stop on: Time (mins)
    Time (mins): 1.0
    Set random seed: true
    Set random seed: true
    Prevent overtraining: true
    Sample %: 75.0
    Optimize: Memory
Training Summary
    Model type: Neural net
    Stream: \\special\etourval$\Desktop\thesisRuns\finalProject\prune_31_13,22,42,66.str
    User: etourval
    Date built: 11/9/06 12:27 PM
    Application: Clementine 10.1
    Elapsed time for model build: 0 hours, 1 mins, 22 secs
```

Figure 36. Model 15 (Predict Bad 31, 53 and Good 4, 26, 42, 66)



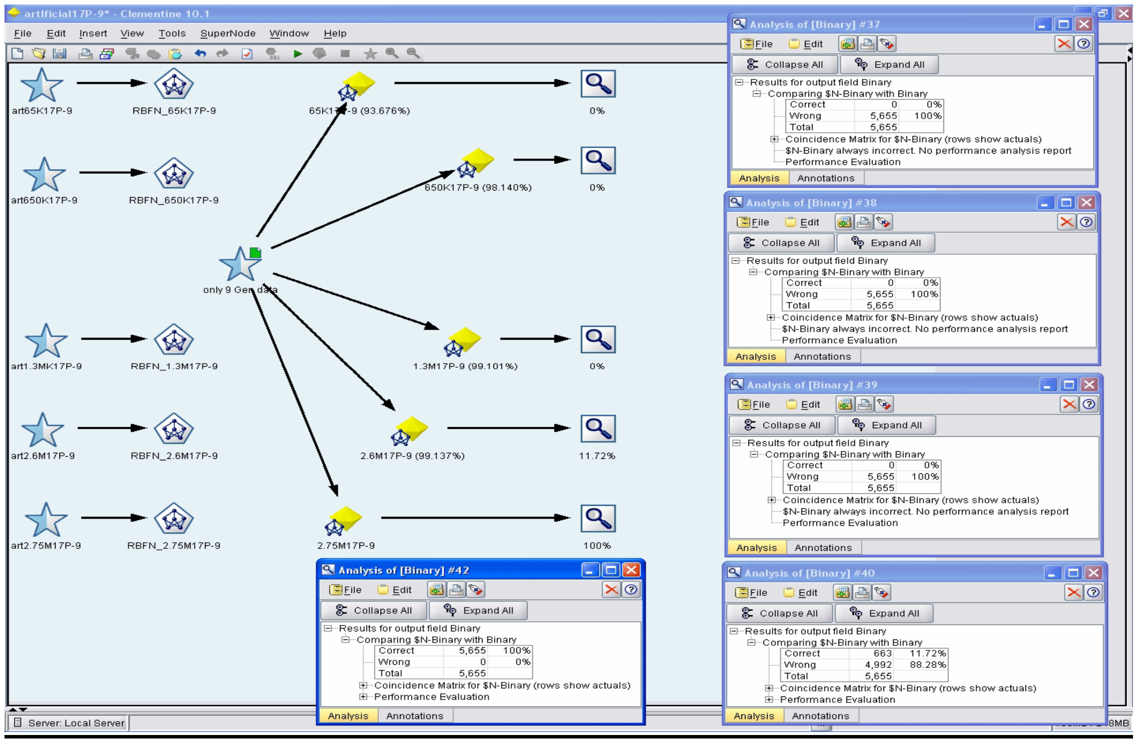
```
Analysis
    Estimated accuracy: 99.509
    Input Layer: 17 neurons
    Hidden Layer 1: 31 neurons
    Output Layer: 1 neurons
    Relative Importance of Inputs
```

SO1	0.765712
SO2	0.683225
EnvelopeRMS	0.599992
BallEnergy	0.454621
GearMisalignment1	0.411992
Geardistributedfault	0.409734
Residualpeaktopeak	0.409627
G2_1	0.391811
SO3	0.391033
CageEnergy	0.376804
Residualkurtosis	0.372624
OuterRaceEnergy	0.341233
ResidualRMS	0.332514
SignalAvgRMS	0.323322
SideBandMod1	0.314663
InnerRaceEnergy	0.139153
Torque	0.083352

```
Fields
Build Settings
    Use partitioned data: false
    Method: Prune
    Stop on: Time (mins)
    Time (mins): 1.0
    Set random seed: true
    Set random seed: true
    Prevent overtraining: true
    Sample %: 75.0
    Optimize: Memory
Training Summary
    Model type: Neural net
    Stream: \\special\etourval$\Desktop\thesisRuns\finalProject\prune_31_13,22,42,66.str
    User: etourval
    Date built: 11/11/06 8:09 AM
    Application: Clementine 10.1
    Elapsed time for model build: 0 hours, 1 mins, 6 secs
```

Figure 37. Model 16 (Predict Bad 31, 56 and Good 4, 26, 42, 66)

APPENDIX C



```
Analysis of [Binary] #37
File  Edit
Collapse All   Expand All
Results for output field Binary
  Comparing $N-Binary with Binary
    Correct        0      0%
    Wrong      5,655    100%
    Total      5,655
  Coincidence Matrix for $N-Binary (rows show actuals)
  $N-Binary always incorrect. No performance analysis report
  Performance Evaluation
Analysis  Annotations
```

```
Analysis of [Binary] #38
File  Edit
Collapse All   Expand All
Results for output field Binary
  Comparing $N-Binary with Binary
    Correct        0      0%
    Wrong      5,655    100%
    Total      5,655
  Coincidence Matrix for $N-Binary (rows show actuals)
  $N-Binary always incorrect. No performance analysis report
  Performance Evaluation
Analysis  Annotations
```

```
Analysis of [Binary] #39
File  Edit
Collapse All   Expand All
Results for output field Binary
  Comparing $N-Binary with Binary
    Correct        0      0%
    Wrong      5,655    100%
    Total      5,655
  Coincidence Matrix for $N-Binary (rows show actuals)
  $N-Binary always incorrect. No performance analysis report
  Performance Evaluation
Analysis  Annotations
```

```
Analysis of [Binary] #42
File  Edit
Collapse All   Expand All
Results for output field Binary
  Comparing $N-Binary with Binary
    Correct    5,655    100%
    Wrong          0      0%
    Total      5,655
  Coincidence Matrix for $N-Binary (rows show actuals)
  Performance Evaluation
Analysis  Annotations
```

```
Analysis of [Binary] #40
File  Edit
Collapse All   Expand All
Results for output field Binary
  Comparing $N-Binary with Binary
    Correct      663   11.72%
    Wrong      4,992   88.28%
    Total      5,655
  Coincidence Matrix for $N-Binary (rows show actuals)
  Performance Evaluation
Analysis  Annotations
```

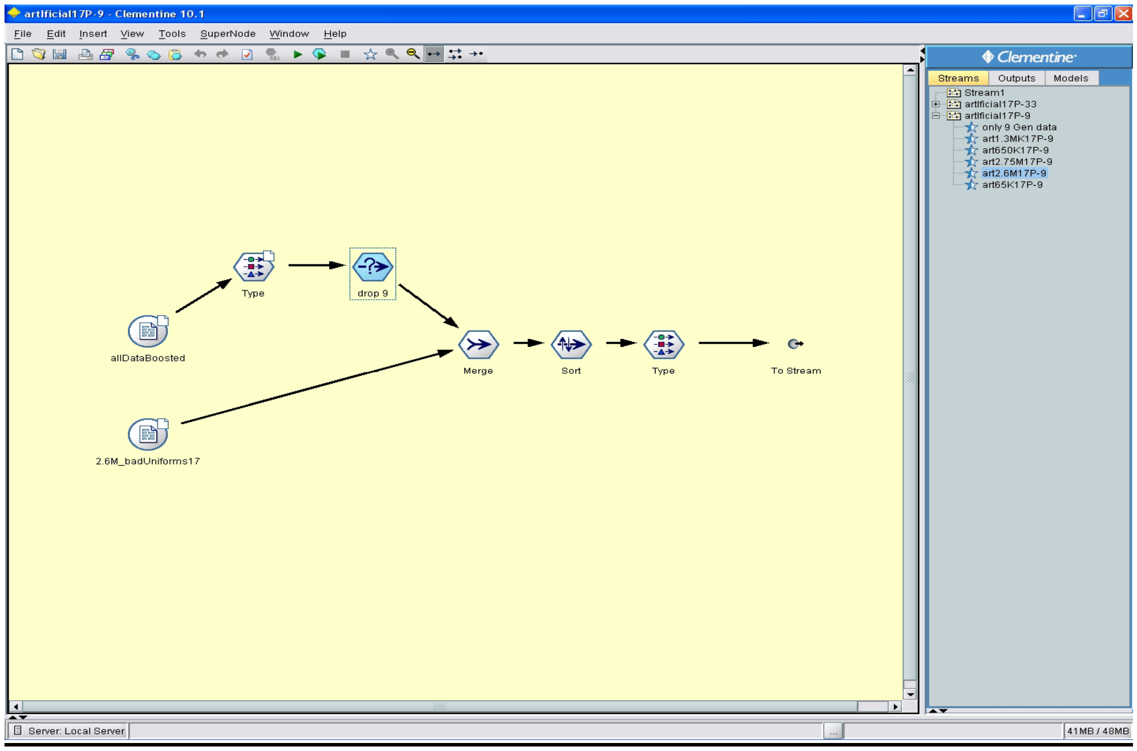
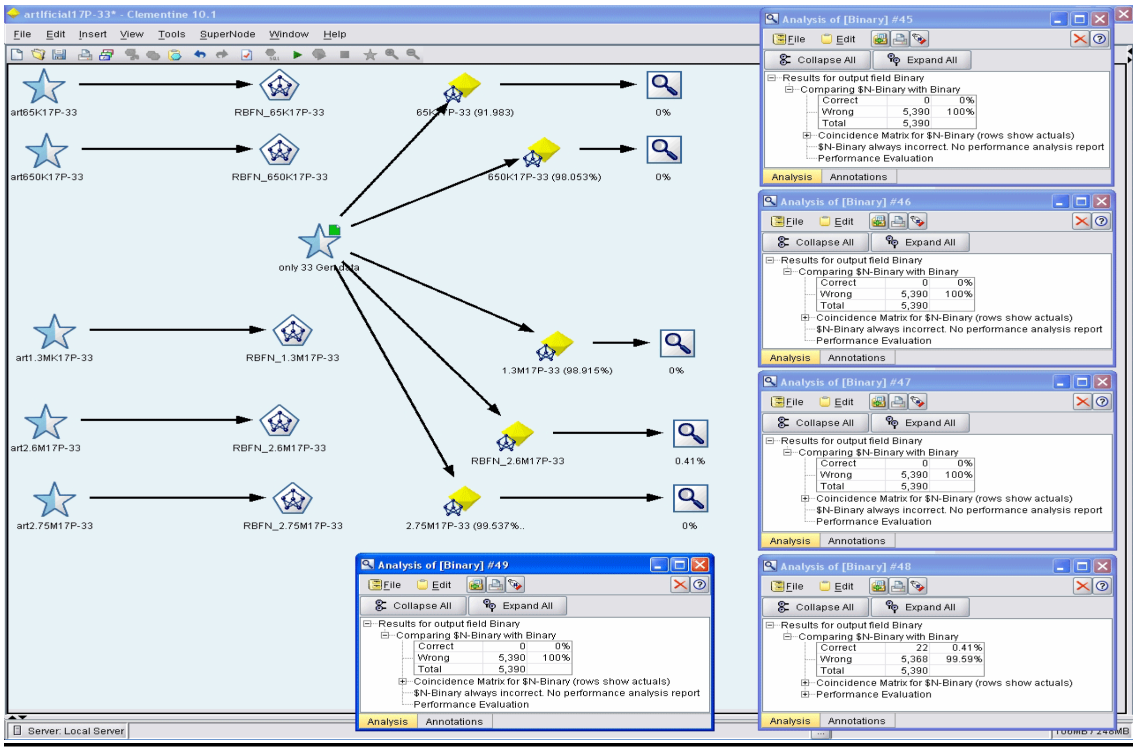


Figure 38. Model 17 (Predict Bad 9 Using Artificial Sets)



Analysis of [Binary] #45

```
Results for output field Binary
  Comparing $N-Binary with Binary
    Correct       0      0%
    Wrong     5,390    100%
    Total     5,390
  Coincidence Matrix for $N-Binary (rows show actuals)
  $N-Binary always incorrect. No performance analysis report
  Performance Evaluation
```

Analysis of [Binary] #46

```
Results for output field Binary
  Comparing $N-Binary with Binary
    Correct       0      0%
    Wrong     5,390    100%
    Total     5,390
  Coincidence Matrix for $N-Binary (rows show actuals)
  $N-Binary always incorrect. No performance analysis report
  Performance Evaluation
```

Analysis of [Binary] #47

```
Results for output field Binary
  Comparing $N-Binary with Binary
    Correct       0      0%
    Wrong     5,390    100%
    Total     5,390
  Coincidence Matrix for $N-Binary (rows show actuals)
  $N-Binary always incorrect. No performance analysis report
  Performance Evaluation
```

Analysis of [Binary] #48

```
Results for output field Binary
  Comparing $N-Binary with Binary
    Correct      22    0.41%
    Wrong     5,368   99.59%
    Total     5,390
  Coincidence Matrix for $N-Binary (rows show actuals)
  Performance Evaluation
```

Analysis of [Binary] #49

```
Results for output field Binary
  Comparing $N-Binary with Binary
    Correct       0      0%
    Wrong     5,390    100%
    Total     5,390
  Coincidence Matrix for $N-Binary (rows show actuals)
  $N-Binary always incorrect. No performance analysis report
  Performance Evaluation
```

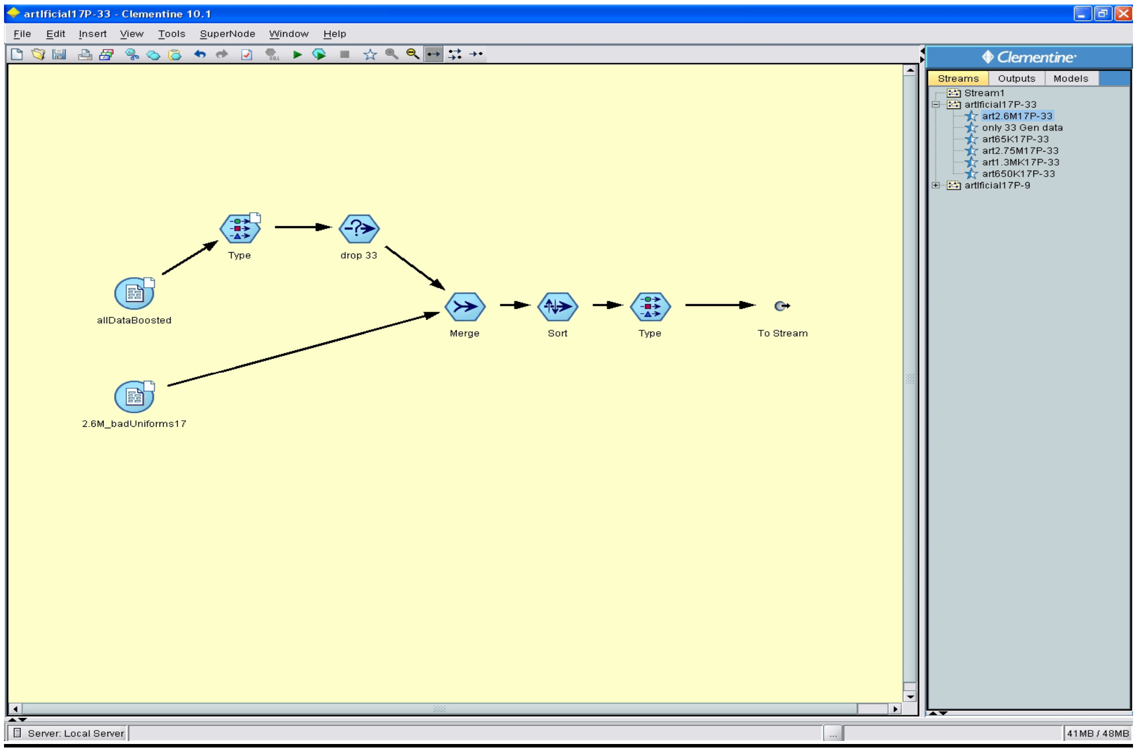
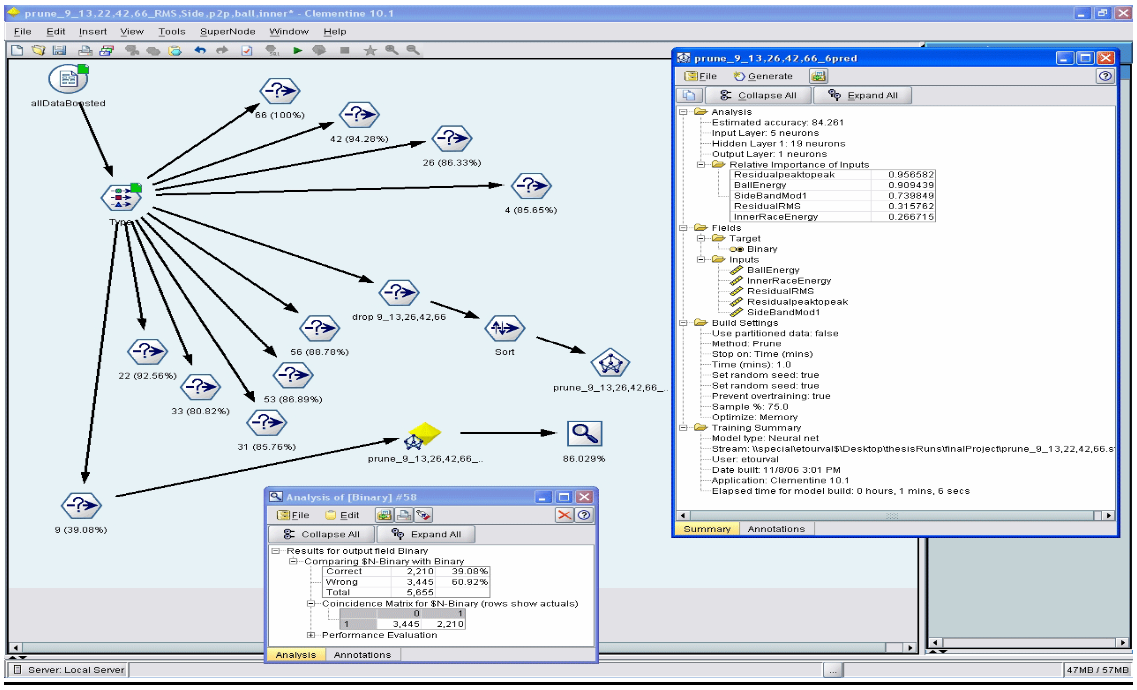


Figure 39. Model 18 (Predict Bad 33 Using Artificial Sets)

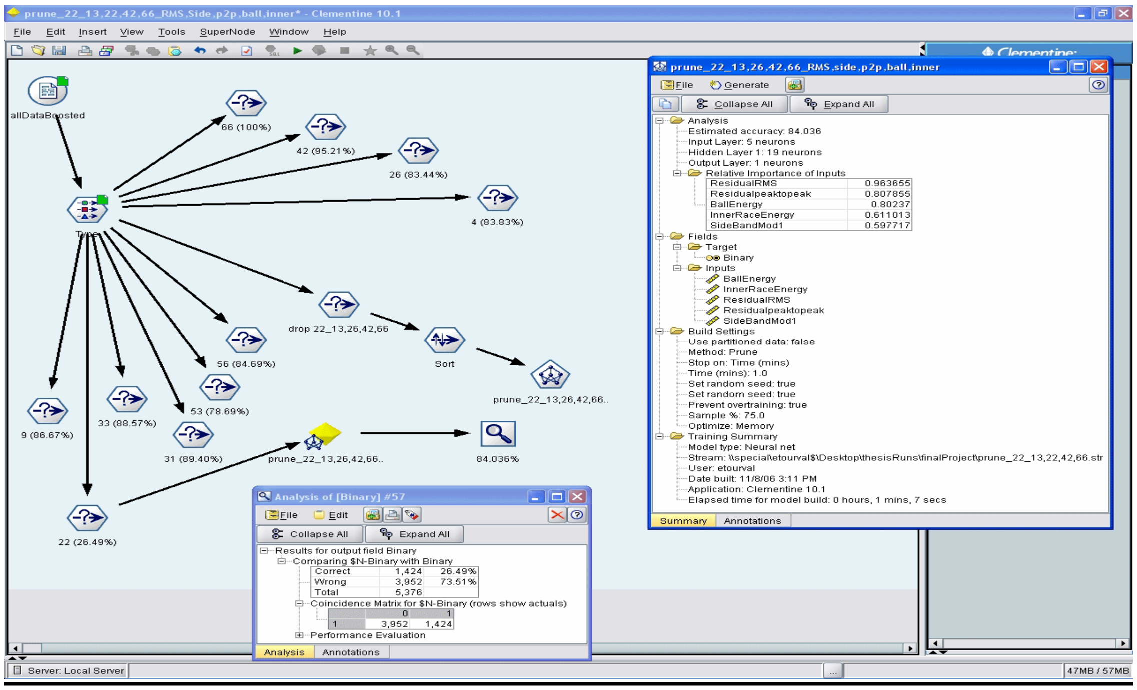
APPENDIX D



```
prune_9_13,26,42,66_6pred
Analysis
  Estimated accuracy: 84.261
  Input Layer: 5 neurons
  Hidden Layer 1: 19 neurons
  Output Layer: 1 neurons
  Relative Importance of Inputs
    Residualpeaktopeak      0.956582
    BallEnergy              0.909439
    SideBandMod1            0.739849
    ResidualRMS             0.315762
    InnerRaceEnergy         0.266715
Fields
  Target
    Binary
  Inputs
    BallEnergy
    InnerRaceEnergy
    ResidualRMS
    Residualpeaktopeak
    SideBandMod1
Build Settings
  Use partitioned data: false
  Method: Prune
  Stop on: Time (mins)
  Time (mins): 1.0
  Set random seed: true
  Set random seed: true
  Prevent overtraining: true
  Sample %: 75.0
  Optimize: Memory
Training Summary
  Model type: Neural net
  Stream: \\special\etourval$\Desktop\thesisRuns\finalProject\prune_9_13,22,42,66.s
  User: etourval
  Date built: 11/8/06 3:01 PM
  Application: Clementine 10.1
  Elapsed time for model build: 0 hours, 1 mins, 6 secs
```

```
Analysis of [Binary] #58
Results for output field Binary
  Comparing $N-Binary with Binary
    Correct     2,210    39.08%
    Wrong       3,445    60.92%
    Total       5,655
  Coincidence Matrix for $N-Binary (rows show actuals)
                0        1
    1       3,445    2,210
  Performance Evaluation
```

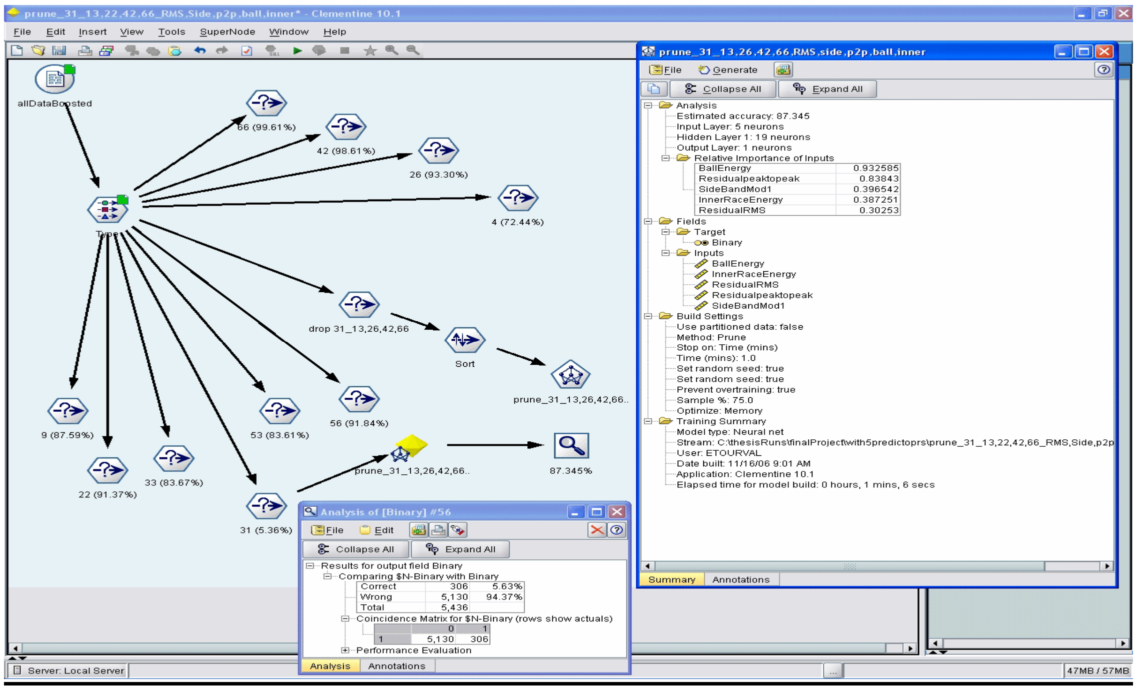
Figure 40. Stepwise Model Using 5 Predictors (Predict Bad 9)



```
prune_22_13,26,42,66_RMS,side,p2p,ball,inner
Analysis
  Estimated accuracy: 84.036
  Input Layer: 5 neurons
  Hidden Layer 1: 19 neurons
  Output Layer: 1 neurons
  Relative Importance of Inputs
    ResidualRMS             0.963655
    Residualpeaktopeak      0.807855
    BallEnergy               0.80237
    InnerRaceEnergy         0.611013
    SideBandMod1            0.597717
Fields
  Target
    Binary
  Inputs
    BallEnergy
    InnerRaceEnergy
    ResidualRMS
    Residualpeaktopeak
    SideBandMod1
Build Settings
  Use partitioned data: false
  Method: Prune
  Stop on: Time (mins)
  Time (mins): 1.0
  Set random seed: true
  Set random seed: true
  Prevent overtraining: true
  Sample %: 75.0
  Optimize: Memory
Training Summary
  Model type: Neural net
  Stream: \\special\etourval$\Desktop\thesisRuns\finalProject\prune_22_13,22,42,66.str
  User: etourval
  Date built: 11/8/06 3:11 PM
  Application: Clementine 10.1
  Elapsed time for model build: 0 hours, 1 mins, 7 secs
```

```
Analysis of [Binary] #57
Results for output field Binary
  Comparing $N-Binary with Binary
    Correct     1,424    26.49%
    Wrong       3,952    73.51%
    Total       5,376
  Coincidence Matrix for $N-Binary (rows show actuals)
                0        1
    1       3,952    1,424
  Performance Evaluation
```

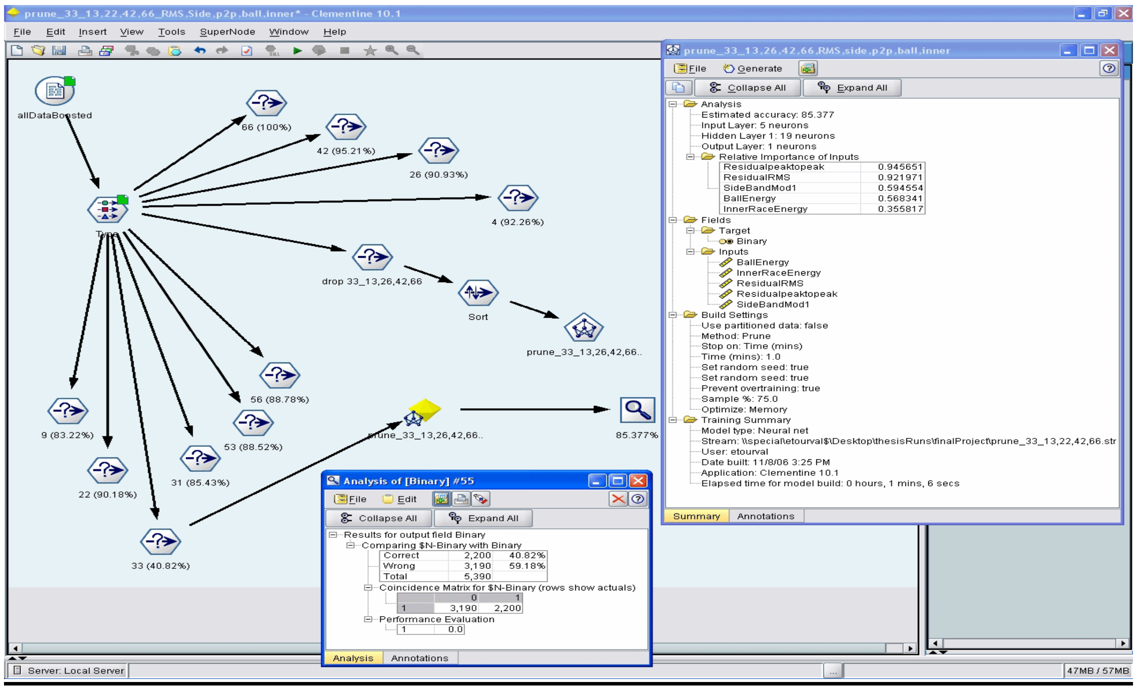
Figure 41. Stepwise Model Using 5 Predictors (Predict Bad 22)



```
prune_31_13,26,42,66,RMS,side,p2p,ball,inner
Analysis
    Estimated accuracy: 87.345
    Input Layer: 5 neurons
    Hidden Layer 1: 19 neurons
    Output Layer: 1 neurons
    Relative Importance of Inputs
        BallEnergy            0.932585
        Residualpeaktopeak     0.83843
        SideBandMod1          0.396542
        InnerRaceEnergy       0.387251
        ResidualRMS            0.30253
Fields
    Target
        Binary
    Inputs
        BallEnergy
        InnerRaceEnergy
        ResidualRMS
        Residualpeaktopeak
        SideBandMod1
Build Settings
    Use partitioned data: false
    Method: Prune
    Stop on: Time (mins)
    Time (mins): 1.0
    Set random seed: true
    Set random seed: true
    Prevent overtraining: true
    Sample %: 75.0
    Optimize: Memory
Training Summary
    Model type: Neural net
    Stream: C:\thesisRuns\finalProject\with5predictoprs\prune_31_13,22,42,66_RMS,Side,p2p
    User: ETOURVAL
    Date built: 11/16/06 9:01 AM
    Application: Clementine 10.1
    Elapsed time for model build: 0 hours, 1 mins, 6 secs
```

```
Analysis of [Binary] #56
Results for output field Binary
    Comparing $N-Binary with Binary
        Correct       306     5.63%
        Wrong       5,130    94.37%
        Total       5,436
    Coincidence Matrix for $N-Binary (rows show actuals)
                    0      1
        1       5,130    306
    Performance Evaluation
```

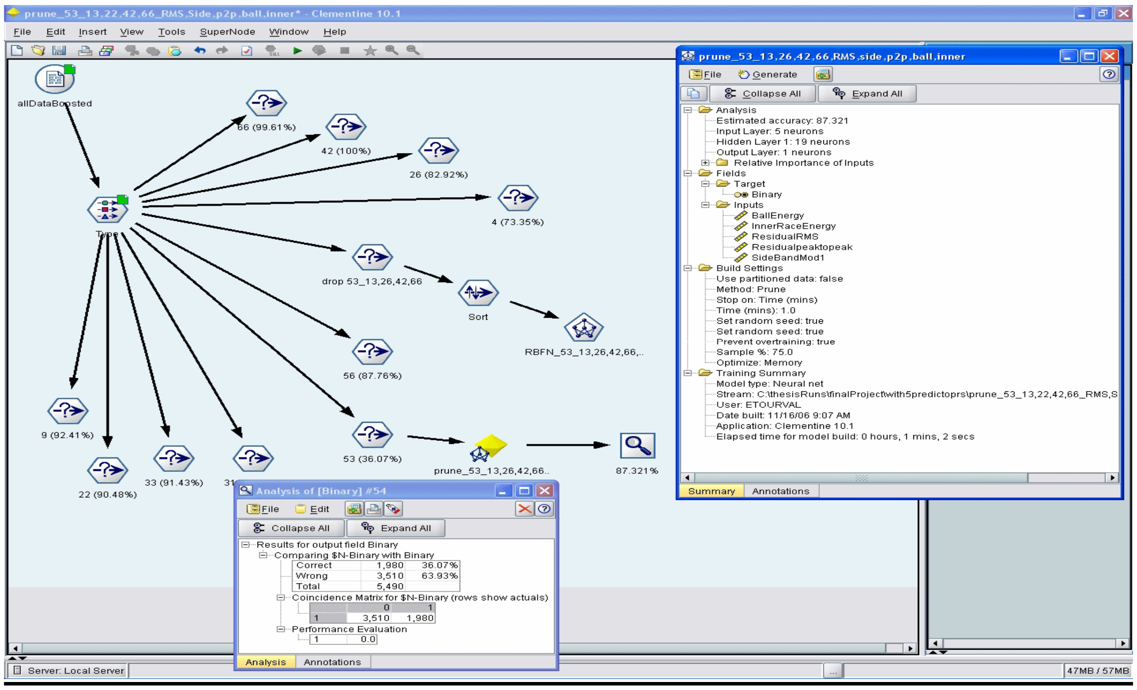
Figure 42. Stepwise Model Using 5 Predictors (Predict Bad 31)



```
prune_33_13,26,42,66,RMS,side,p2p,ball,inner
Analysis
    Estimated accuracy: 85.377
    Input Layer: 5 neurons
    Hidden Layer 1: 19 neurons
    Output Layer: 1 neurons
    Relative Importance of Inputs
        Residualpeaktopeak    0.945651
        ResidualRMS           0.921971
        SideBandMod1          0.594554
        BallEnergy            0.568341
        InnerRaceEnergy       0.355817
Fields
    Target
        Binary
    Inputs
        BallEnergy
        InnerRaceEnergy
        ResidualRMS
        Residualpeaktopeak
        SideBandMod1
Build Settings
    Use partitioned data: false
    Method: Prune
    Stop on: Time (mins)
    Time (mins): 1.0
    Set random seed: true
    Set random seed: true
    Prevent overtraining: true
    Sample %: 75.0
    Optimize: Memory
Training Summary
    Model type: Neural net
    Stream: \\special\etourval$\Desktop\thesisRuns\finalProject\prune_33_13,22,42,66.str
    User: etourval
    Date built: 11/8/06 3:25 PM
    Application: Clementine 10.1
    Elapsed time for model build: 0 hours, 1 mins, 6 secs
```

```
Analysis of [Binary] #55
Results for output field Binary
    Comparing $N-Binary with Binary
        Correct     2,200    40.82%
        Wrong       3,190    59.18%
        Total       5,390
    Coincidence Matrix for $N-Binary (rows show actuals)
                    0        1
        1       3,190    2,200
    Performance Evaluation
        1         0.0
```

Figure 43. Stepwise Model Using 5 Predictors (Predict Bad 33)



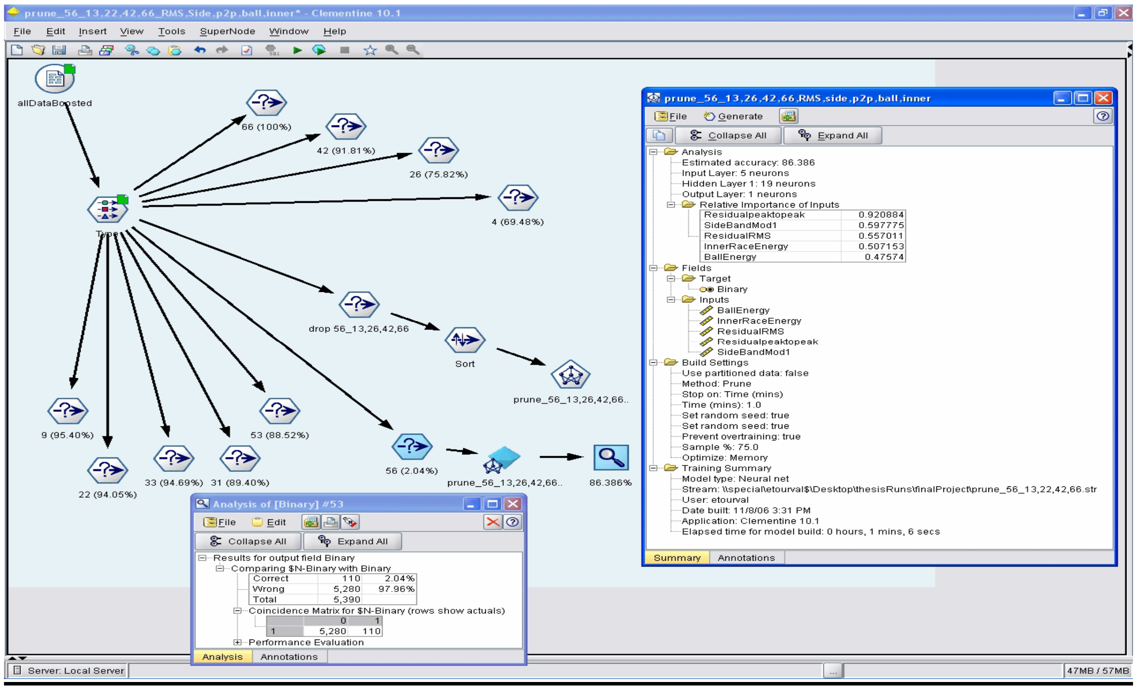
prune_53_13,26,42,66,RMS,side,p2p,ball,inner

```
Analysis
  Estimated accuracy: 87.321
  Input Layer: 5 neurons
  Hidden Layer 1: 19 neurons
  Output Layer: 1 neurons
  Relative Importance of Inputs
Fields
  Target
    Binary
  Inputs
    BallEnergy
    InnerRaceEnergy
    ResidualRMS
    Residualpeaktopeak
    SideBandMod1
Build Settings
  Use partitioned data: false
  Method: Prune
  Stop on: Time (mins)
  Time (mins): 1.0
  Set random seed: true
  Set random seed: true
  Prevent overtraining: true
  Sample %: 75.0
  Optimize: Memory
Training Summary
  Model type: Neural net
  Stream: C:\thesisRuns\finalProject\with5predictoprs\prune_53_13,22,42,66_RMS,S
  User: ETOURVAL
  Date built: 11/16/06 9:07 AM
  Application: Clementine 10.1
  Elapsed time for model build: 0 hours, 1 mins, 2 secs
```

Analysis of [Binary] #54

```
Results for output field Binary
  Comparing $N-Binary with Binary
    Correct     1,980    36.07%
    Wrong       3,510    63.93%
    Total       5,490
  Coincidence Matrix for $N-Binary (rows show actuals)
                0        1
    1       3,510    1,980
  Performance Evaluation
    1         0.0
```

Figure 44. Stepwise Model Using 5 Predictors (Predict Bad 53)



prune_56_13,26,42,66,RMS,side,p2p,ball,inner

```
Analysis
  Estimated accuracy: 86.386
  Input Layer: 5 neurons
  Hidden Layer 1: 19 neurons
  Output Layer: 1 neurons
  Relative Importance of Inputs
    Residualpeaktopeak   0.920884
    SideBandMod1         0.597775
    ResidualRMS          0.557011
    InnerRaceEnergy      0.507153
    BallEnergy           0.47574
Fields
  Target
    Binary
  Inputs
    BallEnergy
    InnerRaceEnergy
    ResidualRMS
    Residualpeaktopeak
    SideBandMod1
Build Settings
  Use partitioned data: false
  Method: Prune
  Stop on: Time (mins)
  Time (mins): 1.0
  Set random seed: true
  Set random seed: true
  Prevent overtraining: true
  Sample %: 75.0
  Optimize: Memory
Training Summary
  Model type: Neural net
  Stream: \\special\etourval$\Desktop\thesisRuns\finalProject\prune_56_13,22,42,66.str
  User: etourval
  Date built: 11/8/06 3:31 PM
  Application: Clementine 10.1
  Elapsed time for model build: 0 hours, 1 mins, 6 secs
```

Analysis of [Binary] #53

```
Results for output field Binary
  Comparing $N-Binary with Binary
    Correct       110     2.04%
    Wrong       5,280    97.96%
    Total       5,390
  Coincidence Matrix for $N-Binary (rows show actuals)
                0        1
    1       5,280      110
  Performance Evaluation
```

Figure 45. Stepwise Model Using 5 Predictors (Predict Bad 56)

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Bechhoefer E., Bernhard A., Use of Non-Gaussian Distribution for Analysis of Shaft Components, IEEE, 2005.

Bechhoefer E., Mayhew E., Mechanical Diagnostics Systems Engineering in IMD-HUMS, IEEE, 2005.

Bechhoefer E., Power D., IMD HUMS Rotor Track and Balance Techniques, IEEE, 2002.

Bishop, C., NNs for Pattern Recognition, Clarendon Press, 1995.

Buttrey S., Koyak R., Read R., Whitaker L., Prognostics of Complex Rotating Machinery Statistical Concepts and Capabilities of the NPS Team, Naval Postgraduate School, 2003.

Clementine 10.0 User’s Guide, SSPS, Integral Solutions, 2005.

Davis A., Handbook of Condition Monitoring, Chapman & Hall, 1998.

Duda R., Hart P., Stork D., Pattern Classification, Second Edition, John Wiley & Sons, 2000.

Elyurek M., Establishing a Vibration Threshold Value, Which Ensures a Negligible False Alarm Rate for Each Gear in CH-53 Aircraft Using the Operational Data, Master’s Thesis, Department of Operations Analysis, Naval Postgraduate School, 2003.

Fausett L., Fundamentals of NNs, Prentice Hall, 1994.

GGobi Data Visualization System, http://www.ggobi.org, 10 August 2006.

Harris C., Harris' Shock and Vibration Handbook, Fifth Edition, McGraw-Hill, 2002.

Kartalopoulos S., Understanding NNs And Fuzzy Logic, IEEE Press, 1996.

Lawrence J., Introduction to Neural Networks, Fifth Edition, California Scientific, 1993.

Mark, W.D., “Gear Noise Origins,” AGARD Conference Proceedings, 1998.

Norgaardm M., Ravn O., Poulsen N., Hansen L., NNs for Modeling and Control of Dynamic Systems, Springer, 2000.

Rao S., Mechanical Vibrations, Fourth Edition, Pearson Prentice Hall, 2004.

Ripley B., Pattern Recognition and NNs, Cambridge University Press, 1996.

Simpson P., NN Applications, IEEE Technical Activities Board, 1996.

Sobajic D., NN Computing for the Electric Power Industry, Lawrence Erlbaun Associates, 1993.

Software Requirements Specifications, Goodrich Corporation, 1998.

System Users Manual for Integrated Mechanical Diagnostics Health and Usage Management System (IMD-HUMS), U.S. Army UH-60A/L, February 2005.

Technical Manual 1-1520-237-10, Operators Manual for UH-60L Helicopter, Headquarters, Department of the Army, May 2003.

Tsang, A., Condition-Based Maintenance: Tools and Decision Making, Journal of Quality in Maintenance Engineering, 1995.

Willard L., Klesch G., Using Integrated Mechanical Diagnostics Health And Usage Management System (Imd-Hums) Data To Predict Uh-60l Electrical Generator Condition, Master’s Thesis, Department of Operations Analysis, Naval Postgraduate School, 2005.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Professor Lyn R. Whitaker
   Department of Operations Research
   Naval Postgraduate School
   Monterey, California

4. Professor Samuel E. Buttrey
   Department of Operations Research
   Naval Postgraduate School
   Monterey, California

5. Maj Tourvalis Evangelos
   Greece, Volos