University of the Philippines Manila
College of Arts and Science
Department of Physical Sciences and Mathematics

GNAT: GENETIC NEURAL NETWORK ANALYTIC TOOL

APPLICATION OF GENETIC ALGORITHM OPTIMIZED ARTIFICIAL NEURAL NETWORK FOR THE MEDICAL DIAGNOSIS OF DENGUE FEVER

A special problem in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Computer Science

Submitted by:
Palomo, Mikyle Laurence O.
2008-53909

April 2013

ACCEPTANCE SHEET

The Special Problem entitled "**GNAT: Genetic Neural Network Analytic Tool. Application of Genetic Algorithm Optimized Neural Network for the Medical Diagnosis of Dengue Fever"** prepared and submitted by **Mikyle Laurence O. Palomo** in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

_____

Bernie B. Terrado, M.S. (candidate)
Adviser

EXAMINERS:

		Approved	**Disapproved**
1.	Gregorio B. Baes, Ph.D. (candidate)	_____	_____
2.	Avegail D. Carpio, M.S.	_____	_____
3.	Richard Bryann L. Chua, Ph.D. (candidate)	_____	_____
4.	Aldrich Colin K. Co, M.S. (candidate)	_____	_____
5.	Perlita E. Gasmen M.S. (candidate)	_____	_____
6.	Vincent Peter C. Magboo, M.D., M.S.	_____	_____
7.	Geoffrey A. Solano,Ph.D. (candidate)	_____	_____

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science

_____

Avegail D. Carpio, M.S.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences and Mathematics

_____

Marcelina B. Lirazan, Ph.D.
Chair
Department of Physical Sciences and Mathematics

_____

Alex C. Gonzaga, Ph.D.
Dean
College of Arts and Sciences

ABSTRACT

Dengue fever is the most rapidly spreading mosquito-borne viral disease in the world. An estimated 50 million dengue infections occur annually and approximately 2.5 billion people live in dengue endemic countries, one of which is the Philippines. Dengue is often characterized with influenza-like symptoms which include fever, rash, and body pain. Carried by the Aedes Egypti mosquito, the disease spreads as mosquitoes feed on an infected host and is passed on and transmitted by feeding on another host. Reported cases significantly increase during the rainy season wherein stagnant water becomes home to the mosquitoes and their offspring.

Often, failure to properly diagnose the disease leads to further infection and in worst cases, fatality. Ergo, early detection and proper diagnosis can be critical and life-saving. Clinical decision support systems (CDSS) help achieve and attain that. CDSSs combine knowledge and data to generate and present helpful information to health care providers as care is being delivered.

The Genetic Neural Network Analytic Tool for Dengue Fever (GNAT) is devised and created to be a CDSS that provides a novel, efficient, and robust way to detect and diagnose dengue fever. GNAT utilizes artificial neural networks (ANN) and genetic algorithm (GA) to classify whether patients are infected with dengue fever. Inspired by the biological brain, ANNs emulate the signal integration and threshold firing of biological neurons with the use of mathematical models and statistical weights. Hampered by the lack of an efficient training method, GA is used to optimize the weights between artificial neurons. GAs, as it mimics evolutionary processes, provides the best if not the correct solution to a wide range of problems.

Keywords: Neural Networks, Genetic Algorithm, Clinical Decision Support System, Dengue Fever

CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

A. BACKGROUND OF THE STUDY

In the Philippines, most especially during the rainy season, a certain disease outbreak threatens the health and safety of the general population. Dengue is an infection that causes a severe influenza-like illness, and sometimes a deadly complication called dengue hemorrhagic fever [1]. Spread by the Aedes Egypti mosquito, the number of dengue cases rises significantly during the rainy season when stagnant water become home to these mosquitoes [2].

Failure to properly diagnose whether a patient is infected often occurs. Several known cases have been reported to have failed diagnoses, resulting in worsening of the patients' condition, and at worst cases, fatality. To cope with this predicament, a system is devised to help classify a patient whether infected or not. An artificial neural network with incorporated genetic algorithm is trained to recognize the said disease and help support clinical diagnosis.

Artificial neural networks (ANN) provide a powerful tool to help doctors to analyze, model, and comprehend complex clinical data across a broad range of medical applications. Most applications of artificial neural networks to medicine are classification problems; that is, the task is on the basis of the measured features to assign the patient to one of a small set of classes [3].

ANNs also have several applications among numerous fields. This is mainly due to its ability to learn from training data and employ what it has learned to classify pattern from unseen test data. This is known as generalization ability [4].

Genetic algorithms (GAs) are search algorithms based on the mechanics of natural selection and genetics as observed in the biological world [3]. They use both direction ("survival of the fittest") and randomization to robustly explore a function. Importantly, to implement a

genetic algorithm it is not even necessary to know the form of the function, just its output for a given set of input [5].

GAs belong to a particular class of evolutionary algorithms that uses techniques inspired by evolutionary biology such as inheritance, selection, crossover, and mutation. Recently the trend to hybridize GA and ANN is getting popular among researchers. The advantages offered by these techniques are forming a better genetic neural network hybrid intelligence system that can improve generalization and at the same time ease the ANN design process [4].

Successful application examples show that human diagnostic capabilities are significantly worse than the neural diagnostic systems. To further enhance the results generated within this system, the GAs is used for the optimization of connection weights [6, 7].

B. STATEMENT OF THE PROBLEM

The health of the population, which is primarily based on the result of medical research and the efficiency of healthcare, has a strong impact upon all human activities. Among the most important medical aspects are considered to be good data interpretation and diagnosis.

One of the most essential tasks in healthcare is medical diagnosis. It is as important as the prescription itself. Almost all physicians are confronted during their formation by the task of learning to diagnose. They have to solve the problem of deducing certain diseases or formulating a treatment based on specified observations and knowledge. Most certainly, there is a standard knowledge of seminars, course, manuals, and books, but on one hand medical knowledge quickly outdates and this does not replace own experience [6].

Another difficulty in making a valid diagnosis is having the experience. A physician must have witnessed a sufficient amount of cases to properly identify a certain illness. This can only be reached in the middle and the latter part of a physician's career and therefore not yet present at the end of the academic formation. This system can help in diagnosing certain diseases, in this particular study, dengue fever.

Moreover, humans do not resemble statistic computers but pattern recognition system. Humans can recognize patterns with precision and ease but fails when probabilities have to be assigned to objects and observations. This study attempts to maximize the computing ability of a computer and utilizing its ability to generalize and classify certain groups of data.

With regards to the actual diagnosis, on the part of the patient, the current techniques to detect dengue are invasive due to the serological confirmation using serum or plasma. Furthermore, there is a need to observe which can only be performed by trained medical expert. In addition, current techniques are slow. They require a span of two hours to process the data. With this set-up, patients can be given initial diagnosis with the use of non-invasive

procedure and using only vital statistics and hematologic data. With this setup, results and finding are generated in a less erroneous manner and in a relatively faster rate.

C. OBJECTIVES OF THE STUDY

The proposed system aims to support medical decisions by providing a simple, non-invasive, rapid tool. The main goal of the project is to develop a software system that allows users to:

- Upload Data Set
    - Import valid .CSV, .DAT, .TXT System File (Semi-colon delimited)
    - Set Training Set Ratio
    - Set Validating Set Ratio

- Download Data Set Template
    - Download Training/Validating Template
    - Download Diagnosis Template

- Manage Neural Network Parameters
    - Number of Hidden Nodes
    - Maximum Tolerable Error

- Manage Genetic Algorithm Parameters
    - Population Size (number of chromosomes per generation)
    - Number of Generations
    - Mutation Rate

- Train the Genetic Neural Network
    - Display a graph(Time vs. Sum of Squares Error)
    - User can choose to pause training
    - User can choose to stop training
    - User can choose to continue training after a pause

- Classify Patients
    - Batch Upload
    - Single Entry Upload

- Export Diagnoses Report

D. SIGNIFICANCE OF THE STUDY

With the implementation of the system, medical professionals is provided a powerful tool to help analyze, model and interpret complex clinical data across a broad range of medical application. Often, neural networks are used to resolve classification problem that is, assigning a particular patient to certain set of the population [3].

Furthermore, with the proposed system medical experts are able to detect whether a patient is infected or not based on manifested vital signs and hematologic data, thus, speeding up diagnosis and treatment, as they can already give out prescription to combat further infection.

This system could also be used as a template for other diagnostic systems. Though specialized in the detection of the dengue fever, a few alterations on the network structure and the data set can create a totally different classification system. This is made possible as a neural network has the ability to learn through example and is not bounded by algorithmic or step-by-step procedures. Compared to rules-based system, neural networks are much easier to maintain and update as it is not constrained by a fixed, rigid knowledge.

The genetic algorithm, as it mimics evolutionary processes, provides the best if not the correct solution to a certain problem, in this case, weights of the neural network. Though the training time can be extended by genetic algorithm, several studies have shown significant improvement of results on the patient classification [16].

E. SCOPE AND LIMITATIONS

- The architecture of the diagnostic system is designed in a way that could be used by different scientific disciplines but this specific design is limited to medical diagnosis, specifically, dengue.

- The data set greatly affects the processing time of the neural network. A larger data set results to a more accurate diagnosis but, ultimately, results to a longer processing time. On the other hand, a small number of data would lead to a faster processing time but may cause inappropriate evaluation.

- The data set that is used for the system is obtained from the Amang Rodriguez Memorial Medical Center's medical records. Located at Marikina, Metro Manila, Philippines. Since the paper is dealing with personal and confidential clinical data, its quantity is limited and the personal information of the patients is omitted.

- Data gathered that are used for training and validation, are limited to the following: temperature; platelet count; cardiac rate; blood pressure both systolic and diastolic; respiratory rate; white; and red blood cell count. The data set that is provided by the user should conform to the format found in the downloadable data set template.

- The Multilayer Feedforward Network structure is utilized in this study. It is incorporated with Genetic Algorithm, instead of Back Propagation Algorithm, to populate and adjust the connection weights. Furthermore, the layers of the neural network are structured as follows: 1 input layer, with 8 nodes corresponding to each data column; 1 hidden layer, with the number of nodes to be determined by the user; and the output layer, with 1 node, representing the diagnosis as positive or negative for dengue.

- The Genetic Algorithm has been proven to improve the overall performance of neural networks. It can be used to optimize connection weights, feature extraction, preprocess data inputs, and evolve network architecture. But in this study, the genetic algorithm is solely used to optimize the connection weights between neurons.

- There several known types of dengue fever, which are caused by different type of harmful agents. In this study, the specific types of dengue are not identified and be diagnosed for a particular patient. Instead, it classifies whether an individual is, in general, infected with dengue, with no indication of its type.

- Results generated by the system are solely for clinical support and is not treated as the final diagnosis. The health care providers, and/or medical professionals, and researchers still have the final say regarding the patients' condition.

II. REVIEW OF RELATED LITERATURE

The implementation of human intelligence in scientific equipment has been the subject of scientific research for a long time and of the medical research in the last few decades.

In the 1950's, computer simulation of biological neural network was first introduced. In 1951 McCullogh and Pitts stated the definition of the first artificial neuron.

Together with the evolution of computer technology, modeling of increasingly intricate neural functions and activity of simple neural clusters was defined. Mathematical models that could be applied for practical applications were developed between 1982 and 1987 based on the works of McLelland, Rummelhart, Hopfield and Kohonen [8, 9, 10].

Neural networks can be applied to medicine in four basic fields: modeling, bioelectric signal processing, diagnosing and prognostics, and Image Analysis (Table 1).

Field	Role
Modeling	Simulating and modeling the functions of the biological brain and neurosensory organs.
Signal Processing	Signal filtering and Evaluation
System Control and Checking	Intelligent artificial machine control and checking based on responses of biological or technical systems given to any signals
Classification	Interpretation of physical and instrumental findings to achieve more accurate diagnosis
Prediction	Provides prognostic information based on retrospective

	parameter analysis
Image Analysis and Interpretation	Pattern recognition and feature extraction

Table 1 Functional division of neural network application

Mostly, the purpose applications of artificial neural networks to medicine are classification problems; that is, the task is on the basis of measured features to assign the individual to a specific cluster [5]. Several studies have successfully implemented systems that classify patients.

El-Solh, et al., used a general regression neural network. Clinical and radiographic data were utilized to predict active pulmonary tuberculosis [12]. In that study, 21 distinct parameters were classified into three groups: demographic variables, constitutional symptoms and radiographic findings. The authors employed a ten-fold cross-validation procedure to train the network and provide estimation for diagnosing tuberculosis. The authors recorded and reported approximately 92.3% diagnosis accuracy.

Also, Er, et al., used two different multilayer neural networks with one and two hidden layers and trained the network with the Levenberg-Marquadt training algorithm [13]. 38 parameters were used to evaluate the disease. Three-fold cross-validation procedure was also used to train the network. The authors reported approximately 93% diagnosis accuracy.

In chest disease diagnosis, Er, et al., presented a comparative chest disease diagnosis which was realized by using multilayer, probabilistic, learning vector optimization, and generalized regression [14].

Francisco, et al., developed a new system, from a model based, in a multiagent system. The system integrates a heuristic approach in order to make it more resistant against possible inconsistencies. The heuristic used is based on orthogonal associative memory for neural networks. Knowledge through training has been added to the method using correct patterns of behavior and symptoms of the urinary tract resulting from dysfunctions [15].

. Elveren, et al., published a study wherein four different neural network architecture was used for diseases detection, namely: General Regression, with one hidden layer(GRNN), Multilayer Back propagation, with one hidden layer(MLNNBP), Multilayer Back propagation with Levenberg-Marquardt training algorithm, with two hidden layers and finally, Multilayer neural network with genetic algorithm(MLNNGA). Results show that the GRNN marked an accuracy of 92.30%, 77.00% for the MLNNBP (one layer), and 93.93% for the MLNNBP (two hidden layers). The one with the highest accuracy is the MLNNGA which had an accuracy of 94.88% which can be attributed to the genetic algorithm as its learning procedure [16].

In line with dengue diagnosis, several studies have successfully implemented an effective classification system. Ibrahim, et al., introduced a novel approach to classify the risk in Dengue Hemorrhagic Fever (DHF) patients using the BIA technique. The system involves the application of a small average constant current of less than 1mA at a single frequency of 50 kHz through the human body. The body's bioelectrical resistance, phase angle, body capacitance, and capacitive reactance were measured via 4 surface electrodes. Another conventional method for monitoring risk in DHF patients is to monitor their platelet count and liver function status. These techniques are invasive, tedious, and time-consuming. Moreover, frequent blood taking causes further injury to the subcutaneous tissue and potentially risky to DHF patients. Their experimental results illustrate that the reactance pattern of female dengue fever (DF) patients are more sensitive and risky than the male DHF patients. Any change in the value of reactance indicates the changes in electrical conductivity of the body, and this can be used to classify the risk severity. In conclusion, reactance was found to be a potentially useful tool in classifying the risk factor of DHF patients [17].

Ibrahim, et al, created a noninvasive approach to diagnose and classify early risk in dengue patients using bioelectrical impedance analysis (BIA) and artificial neural networks (ANN). A total of 223 healthy subjects and 207 hospitalized dengue patients were included in the study. The dengue risk severity criteria was determined and grouped based on three blood investigation namely: platelet count, hematocrit, and aspatate aminotransferase levels. The dengue patients were classified and were subsequently corresponding BIA parameters were obtained and quantified. Four parameters were used for training and testing the ANN which were fever, reactance, gender, and risk group quantification. As a result, the system was able to classify and diagnose the risk in dengue patients with an overall accuracy of 96.27% [18].

Aburas, et al., created a prediction system for the dengue confirmed cases using artificial neural networks. Data was gathered from the Singaporean National Environment Agency (NEA) was used to model the behavior of dengue cases based on the physical parameter of mean temperature, mean relative humidity, and total rainfall. A total of 14,209 dengue reported cases have been analyzed using ANNs. Results show that the four parameters were very effective in predicting the number of dengue confirmed cases. ANN has been found to be very effective in predicting the number of confirmed cases. . The proposed prediction model can be used world-wide and in any period of time since the approach does not use time information in building it [19].

Salim, et al., studied and compared back propagation neural network and nonlinear regression models for predicting dengue outbreaks. Their research aimed to design a neural network (NN) and nonlinear regression model (NLRM) using different architectures and several parameters to define the best architecture for early prediction of a dengue outbreak. Four architectures were developed in the study. Architecture I involved only dengue cases, Architecture II involves dengue cases plus rainfall data, Architecture III involved proximity location and dengue cases. Finally, Architecture IV involved a combination of all the criteria. The performance of overall architecture was analyzed and the result shows that the MSE for all

architectures by using NN is better compared by NNR. Furthermore, the results also indicate that architecture IV performs significantly better than other architectures in predicting dengue outbreak using NNM compared to NLRM.The approach was proven helpful in the problem of time series prediction of dengue outbreak [20].

A similar study, Ibrahim, et al., created a symptoms analysis system for dengue. Their expert system maps out clinical procedures observed by medical experts in clinical diagnosis and determination of dengue fever (DF) and dengue hemorrhagic fever (DHF). The symptoms of the said diseases were studied, represented in a tree-like structure amenable to the expert system. It is designed that such a framework, automated through an expert system rule-chaining technology aids medical personnel to carry out DF and DHF diagnostic procedures accurately and efficiently. DHF specific symptoms can only be identified by a medical expert. In order to recognize and differentiate between DF and DHF symptoms' characteristics of the dengue diseases, knowledge base is formed via information gathered from medical experts, patients' clinical epidemiological questionnaire, medical staff, and published works in dengue cases. This information was analyzed and the corresponding clinical procedures conducted by the medical experts in diagnosing and determining the DF and DHF were schematized into a tree-like structure amenable to expert system applications. The model provides a new perspective for visualizing and differentiating DF from DHF by means of diagnostics symptoms [21].

Several studies have delved into the subject of genetic algorithms used as a learning algorithm for neural networks. These studies have proven the efficiency of the application of genetic algorithm over the backpropagation algorithm for the learning algorithms for neural networks.

J. Gupta and R. Sexton, published a study that compared the use of backpropagation(BP) and genetic algorithm(GA) for optimizing artificial neural networks(NN). For their chaotic time series, empirical results show that the GA is superior to BP in effectiveness, ease-of-use, and

efficiency in training neural networks. Since the GA did not need to find optimal parameter settings, its ease-of-use was much greater than BP. This algorithm also found superior solution in less time, which is a major concern in NN research. Although BP is by far the most popular method of optimization for NNs, it is apparent from their research that a genetic algorithm may be more suitable for training the NNs. Even though this study only includes one function for comparison, it can be seen that the problem of local convergence undoubtedly affects a wide variety of unknown functions that can be estimated using neural networks. By using a global search technique, such as the GA, for neural network training, many if not all problems associated with BP can be overcomed [22].

R. Sexton and R. Dorsey conducted a study to compare the performance of backpropagation(BP) and genetic algorithm(GA) on real-world data sets. To gain confidence that this alternative training technique, genetic algorithm, is suitable for classification problems, a collection of 10 benchmark real world data sets were used in an extensive Monte Carlo study. Results show that GA reliably outperforms the commonly used BP algorithm as an alternative NN training technique. Although BP used the validation data for training and trained for many more epochs, it still performed poorer than the GA for these classification problems. A critique of this study might be that the BP configuration used is limited to one set of user-defined parameter settings, for example, setting the initial learning rate to 1.0 and the momentum value to 0.9.Changing any or all of the parameters available to users of the BP algorithm could possibly result in better performance. However, since there are no heuristics for setting these parameters and each BP NN is problem dependent, the problem facing the decision maker is which set to choose and recommendations provided in the software are reasonable choices. The GA, as used the study, on the other hand, has predefined parameter settings and can be readily used for any given problem. Our findings indicate that the GA is not dependent upon the initial random weights for finding superior solutions and is more consistent and predictable in its abilities for finding those solutions. This research has shown the GA to be a viable alternative for NN optimization that finds superior solutions in a more consistent and predictable manner than those using BP. Although in this study, the GA actually

used less time than the BP algorithm, it is often the case that BP is used because it converges rapidly. Managers and concerned users must keep in mind, however, that solutions other than global solutions may lead to unreliable forecasts. This study, of course, does not prove that the GA dominates for all problems but does indicate that the GA may enable managers to use NNs with more confidence that the reported solution is in fact the desired solution and, thus, allow the NN to become a powerful tool for managers [23].

III. THEORETICAL FRAMEWORK

A. ARTIFICIAL NEURAL NETWORKS (ANN)

The human brain is made consists of more than a billion neural cells that process information. Each cell works as a processor and only the massive interactions between all these cells and their parallel processing makes the brain's capabilities possible.

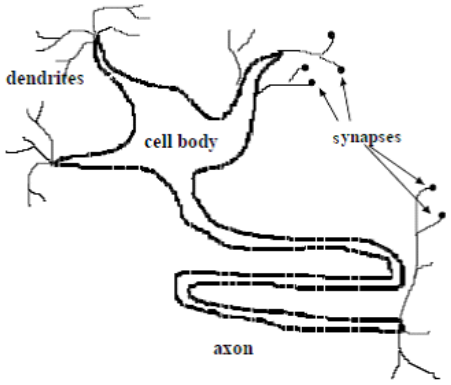


Figure 1 Neural Cell

Each neural cell (see Figure 1), or neuron, collects information, by the form of signals, from others through a host of fine structure called dendrites. The neuron then sends out surges of electrical activity through long, thin strands called axons, which, in turn, splits into thousands of branches. At the end of each branch, a structure called synapse converts the activity from the axon into electric stimulations that inhibit activity in connected neurons. If the stimulation has exceeded a certain threshold, the neuron is said to be activated. On the other hand, if the stimulation was too weak or low, the information is not passed on any further. In this case, the neuron is said to be inhibited. The connections between the neurons are adaptive. The connection structure is changing dynamically. It is known that the learning ability of the human brain is based on this adaptation.

Inspired by the biological neuron, an artificial neuron (see Figure 2) is a simplistic representation that emulates the signal integration and threshold firing behavior of biological neurons by means of mathematical equations. Artificial neurons are attached together by connections that determine the flow of information between among other neurons. Much like their biological counterpart, stimuli are transmitted via synapses or interconnections. It can also be either excitatory or inhibitory. Likewise, if the input is excitatory, the neuron is activated and the signal is passed on. Whereas, an inhibitory input is most likely disregarded.

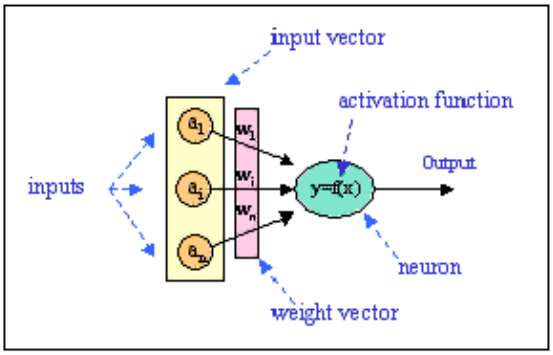


Figure 2 Basic Model of a Single Neuron

A subgroup of a processing element is called a network layer. A typical ANN has the input layer as the first layer and the output layer as the last. Between the input and output layer there may be additional layer(s) or units called hidden layer(s). An ANN can be trained to execute a particular function by adjusting the values of the connections, or weights, between elements. Figure 3 represents a typical neural network.

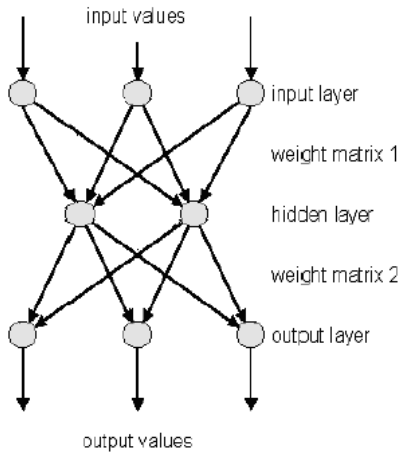


Figure 3 Typical Neural Network

There are many types of neural networks. Certain networks are more efficient in optimization and other performs well in data modeling and so forth. Based on their structure neural networks can be classified as a Perceptron, Multi-layered, Backpropagation, Hopfield, and the Kohenen feature map. Also, neural networks can be classified as feedforward or feedback. Feedforward neural networks allow only neuron connections between two different layers, while networks of the feedback type have also connections between neuron of the same layer. For this study, a Multilayer Feedfoward Network is used.

Furthermore, based on the way neural network learns, it is classified into two learning categories: supervised and unsupervised.

In supervised learning, a network is trained by providing it with input and output patterns. During this phase, the neural network is able to adjust the connection weights to match it output with the actual output in an iterative process until a desirable result is reached. Paradigms of supervised learning include error-correction learning, reinforcement learning and stochastic learning. A concern for the supervised learning is the problem of error convergence. The aim is to determine a set of weights which minimizes the error between the desired and computed values.

While an unsupervised learning type, such as the self-organizing map, the input are only provided. There are no known answers. The network must then develop its own interpretation of the input stimuli by calculating acceptable connection weights. By clustering the input data, the network can then find inherent features intrinsic to the problem. Also aims to let the network detect emergent collective properties.

Another factor that contributes to the behavior of a neural network is the activation function. Activation function is applied to the weighted sum of the inputs to produce an output. Activation functions are added to the hidden unit to introduce nonlinearity into the network [24]. Without nonlinearity, hidden units would not make neural networks more powerful than just plain perceptrons (which do not have any hidden units, just input and output units). There are 3 main types of activation function which tan-sigmoid, log-sigmoid and linear. Different activation functions affect the performance of an ANN. In this study, log-sigmoid activation function is used and preferred since it only outputs positive values of the neural network, in this case, is 0 for negative dengue cases and 1 for positive dengue cases. Tan-sigmoid activation function are not used since it outputs negative and positive values which can be useful for other classification problems wherein the output of the neural network has a range of a negative number to a positive number.

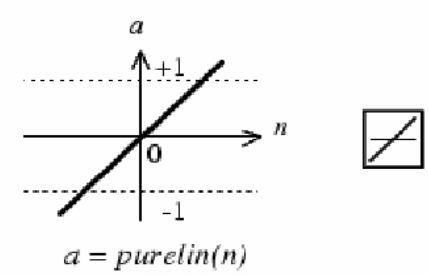


Figure 4 Linear Activation Function

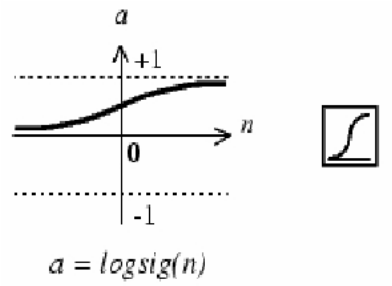


Figure 5 Log-Sigmoid Activation Function

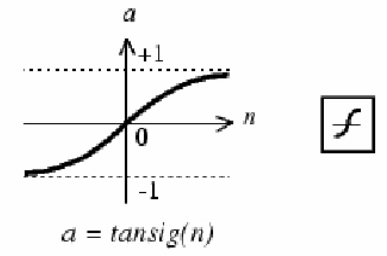


Figure 6 Tan-Sigmoid Activation Function

Another essential aspect of any neural network is the error calculation. Ultimately, the goal of any neural network is to minimize the error it may commit. To evaluate the error for supervised trainings, two values are determined. First, the error of each element of the training set is obtained as it is processed. Also, the average of the errors for all the elements, across each sample, must also be determined.

Basic error calculation, involves the difference between the actual output of the neural network and the ideal output provided by the data set. This value are then used to determine the overall error or the fitness of the neural network by determing the average error across all the data by obtaining the Sum of Squares Error, as shown below:

$$\text{Sum of Squares Error (SSE)} = \sum_{i=1}^{N} (\boldsymbol{actual}_i - \boldsymbol{ideal}_i)^2$$

B. GENETIC ALGORITHM (GA)

A chromosome is a long, complicated thread of DNA (deoxyribonucleic acid). Particular traits of an individual are strung along the length of these chromosomes. Each trait is coded by some combination of DNA bases. Like words in a sentence, meaningful combinations of these bases produce specific instruction to the cell

Changes within an organism occur during reproduction. The chromosome from the parents exchange randomly by a process called crossover. This brings about an offspring that exhibits some combination of traits from both parents. Sometimes, a rarer process occurs called mutation also brings about changes in the offspring. Mutation may lead to a better species but may also have the risk of deteriorating the pool of traits.

Many scientific discoveries and inventions were inspired by nature. One of which is genetic algorithms. Genetic algorithms (see Table 2) search for a solution to a problem by simulating evolution, starting from an initial set of solutions or hypotheses, and generating successive generations of solutions. This particular branch of artificial intelligence was instigated by the way living things evolve into more successful organisms in nature. The main idea of this method is survival of the fittest, also known as, natural selection [25].

Genetic algorithms (GA) are global optimization algorithms drawn for evolutionary ideas and inspiration [25]. These algorithms encode a potential solution to a specific problem on a chromosome-like data structure. Recombination operators are then applied to these structures. This preserves critical information about the current data. Genetic algorithms are often viewed as function optimizer, though, a wide range of problems are handled by genetic algorithms. It has gained extensive application in image processing, biological science, neural networks, pattern recognition, machine learning and the like.

```
Choose initial population
Evaluate each individual's fitness
Determine population's average fitness
Repeat
    Select best-ranking individuals to reproduce
    Mate pairs at random
    Apply crossover operator
    Apply mutation operator
    Evaluate each individual's fitness
    Determine population's average fitness
Until terminating condition (e.g. until at least one individual has the desired fitness or enough
generations have passed)
```

Table 2 Genetic Algorithm Pseudocode

Unlike many search algorithms, which performs a local, greedy search, genetic algorithm is a stochastic general search method. It is capable of effectively exploring large search spaces. A genetic algorithm is mainly composed of four operators: encoding, production, crossover, and mutation [26].

Genetic Algorithms	Explanation
Chromosome(string, individual)	Solution (coding)
Genes (bits)	Part of solution
Locus	Position of gene
Alleles	Values of gene
Phenotype	Decoded solution
Genotype	Encoded solution

Table 3 Genetic Algorithm Terms

Encoding is one of the most essential properties of genetic algorithms. Encoding deals with transforming potential solutions to a problem to a format that a computer can work on and process. There are several types of encoding methods. Some successful encoding methods are: binary; permutation, value and tree (see Table 4).

Binary encoding is the most common encoding scheme, primarily because first works regarding genetic algorithms were encoded in binary. Basically, binary encoding every chromosome is a string of bits, 0 or 1. Binary encoding gives many possible chromosomes even with a small number of alleles. Then again, this encoding is often not natural for many problems and sometimes corrections must be made after genetic operations.

Permutation encoding can be used in ordering problems, such as the travelling salesman problem or a task ordering problem. Permutation encoding represents the solution in a string of numbers, which corresponds to a sequence.

Value encoding is used for problems that have complicated values as data, such as real numbers. Chromosome of this scheme can be a string of some values. Values can be anything connected to a problem. Genotype can be either real numbers or character, and even some complicated objects. In this particular project, value encoding is utilized.

Tree encoding is used mainly for evolving programs or expressions, for genetic programming. In tree encoding every chromosome is a tree of some objects, such as functions or commands in programming language. Tree encoding is good for evolving programs. Programming language, such as, LISP is often used for this, because programs in it are represented in this form and can be easily parsed as a tree, so the crossover and mutation can be done relatively easier.

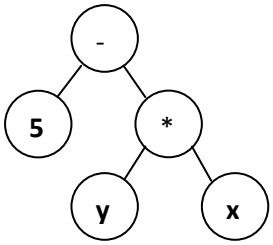
Scheme	Example
Binary	Chromosome 1: 101010100010100011 Chromosome 2: 101011000010101000
Permutation	Chromosome 1: 134205798 Chromosome 2: 923415760
Value	Chromosome 1: 1.2341 2.4423 0.2412 2.2242. 3.212 Chromosome 2: (back),(back),(right),(left),(forward) Chromosome 3: AVEEGABEAFAWDAWFAW
Tree	Chromosome 1: (-5 (* y x)) 

Table 4 Genetic Encoding Schemes

Another genetic operator, reproduction, it is an operator that makes more copies of better strings or encoded data. Reproduction is usually the first operator applied on a population. Reproduction is induced to a population by selecting the best strings from the

population and forming a mating pool. Thus, in the process of reproduction, natural selection allows the better individuals to produce more copies. In order to sustain the next generation, the reproduction of the individuals in the current population is necessary. To be produce a better generation, these individuals should be from the fittest individuals of the previous population. There exist a number of reproduction methods, but the essential idea in all of them, is that the superlative strings are picked from the current population and their multiple copies are inserted in the mating pool in a probabilistic manner.

Crossover (see Table 5) is used to create offspring, given two parent strings. The crossover operation is used to recombine two strings to get a better string. By combining materials from two individuals of the previous generation, two different individuals are created for the succeeding generation. When chromosomes reproduce and a mating pool is formed, good strings are assigned, through probability, a larger number of copies than the worse ones. It is essential to note that no new strings are formed in this phase. Strings are only created by exchanging information among strings in the mating pool. Two strings that are use in this stage are called parent strings and the resulting strings are known as children strings.

Chromosome 1	1110 | 0001
Chromosome 2	1010 | 1001
Offspring 1	1110 | 1001
Offspring 2	1010 | 0001

Table 5 Crossover Operation

Mutation (see Table 6) helps add new information in an arbitrary way to the genetic search process and ultimately helps avoid having the solution at the local optima. This operator introduces diversity into the population when it tends to become homogeneous due to repeated use of reproduction and crossover operations. Mutation causes certain chromosome to be different from those of their parent individuals. Furthermore, mutation randomly distributes genetic information. Mutation operates at bit level, when bits, or genes, are transferred from the parent, there is a probability it may become mutated. Usually valued at a

small value, the mutation probability gives better optima and even modifies a part of a genetic code that is helpful in later operations. Alternatively, it may produce a weak individual that is never selected for further operations. To summarize, increasing the mutation probability tends to transform the genetic search into random search, but it also helps reintroduce lost genetic material [27].

Chromosome 1	1110 0001
Chromosome 2	1010 1001
Mutated Chromosome 1	1111 1001
Mutated Chromosome 2	1010 0011

Table 6 Mutation Operation

C. HYBRID GENETIC NEURAL NETWORK

Multilayered feedforward neural networks possess a number of properties which make them particularly suited to complex pattern classification problems. However, their application to some real world problems has been hampered by lack of a training algorithm which reliably finds a nearly globally optimal set of weights in a relatively short time [28]. Genetic algorithms are classes of optimization procedures which do well at exploring a large and complex space in an intelligent manner to find values close to the global optimum. Hence, they are well suited to the problem of training feedforward networks. GA-NN combination (see Figure 7) provides powerful classification capabilities with tuning flexibility for either performance or cost-efficiency.

When a genetic algorithm is run using a representation that usefully encodes solutions to a problem and operators that can generate better children from good parents, the algorithm can produce populations of better and better individuals, converging finally on results close to a global optimum. In many cases the standard operators, mutation and crossover, are sufficient

for performing the optimization. In such cases, genetic algorithms can serve as a black-box function optimizer not requiring their creator to input any knowledge about the domain [28]. This is one advantage of genetic algorithms over other learning algorithms.

Other known advantages of genetic algorithms over other learning algorithms include: the issue of the local optima; better generalization ability; and its natural occurrence.

With regard solely to the problem of weight selection for networks with fixed topologies and transfer functions, there is the issue of local optima. As the number of exemplars and the complexity of the network topology increase, the complexity of the search space also increases insofar as the error function obtains more and more local minima spread out over a larger portion of the space. Gradient-based techniques often have problems getting past the local optima to find global (or at least nearly global) optima. However, genetic algorithms are particularly good at efficiently searching large and complex spaces to find nearly global optima. As the complexity of the search space increases, genetic algorithms present an increasingly attractive alternative to gradient-based techniques such as backpropagation. Even better, genetic algorithms are an excellent complement to gradient-based techniques such as backpropagation for complex searches. A genetic algorithm is run first to get to the "right hill" (i.e., a part of the space in a neighborhood of a nearly global optimum) and the gradient-based technique climbs the hill to its peak [28].

A second advantage of genetic algorithms is their generality. With only minor changes to the algorithm, genetic algorithms can be used to train all different varieties of networks. They can select weights for recurrent networks. They can train networks with different transfer functions than sigmoids, such as the Gaussians used by other neural networks or even step transfer functions, which are discontinuous and hence not trainable by gradient techniques. Furthermore, genetic algorithms can optimize not just weights and biases but any combination of weights, biases, topology, and transfer functions. Furthermore, genetic algorithms can

optimize not just weights and biases but any combination of weights, biases, topology, and transfer functions [28].

Finally, a third reason to utilize genetic algorithms for learning neural networks is that this is an important method used in nature. While it is not clear that evolutionary learning methods are used in individual organisms, it is relatively certain that evolutionary learning at a species level is a major method of neural network training. For example, in parts of the brain such as the early visual processing component, the neural network weights (in addition to the topology and transfer functions) are essentially "hard-wired" genetically and hence were "learned" via the process of natural selection [28].

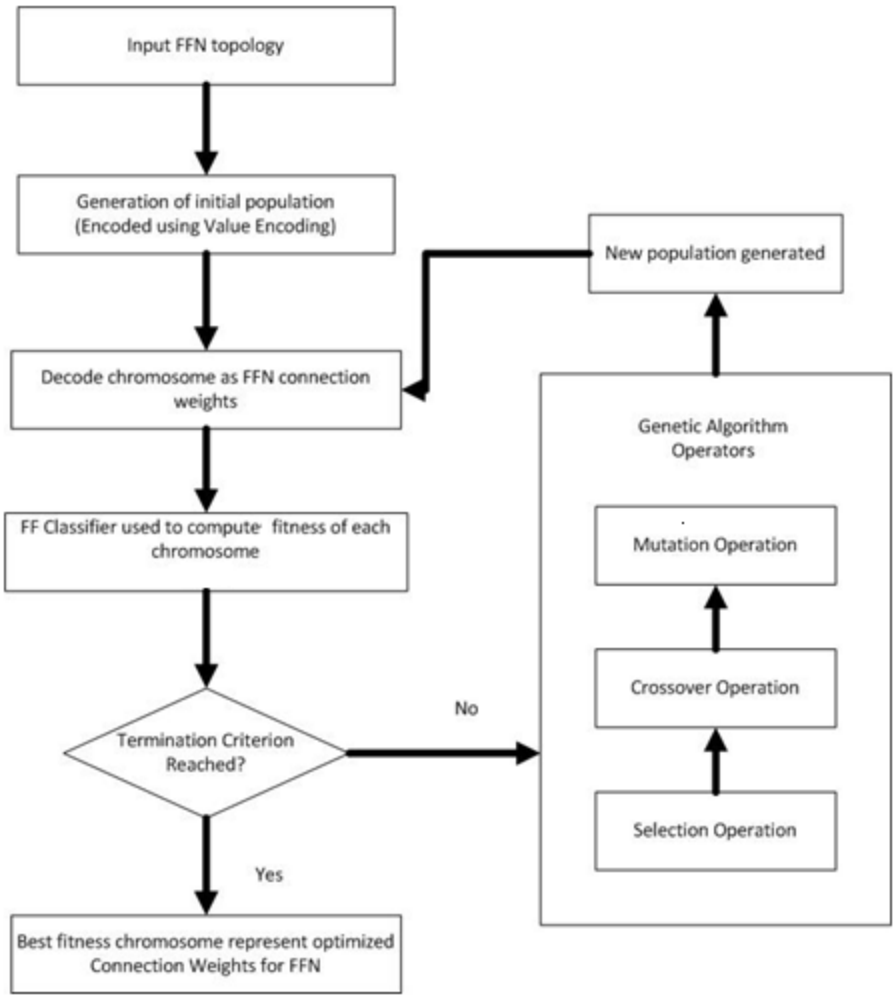


Figure 7 Basic FF-GA Algorithm

D. DENGUE FEVER

The dengue fever is a sickness brought about by a family of viruses that are transmitted by mosquitoes. It is an acute illness of abrupt onset that usually follows a benign course with headache, fever, exhaustion, severe joint and muscle pain, swollen glands, mouth bleeding, nose bleeding, vomiting, diarrhea, and rash(see Figure 8). The “dengue triad” of fever, rash, and body pain is the chiefly characteristic of dengue [30].

Dengue fever symptoms usually appear after 4 to 7 days. These are usually symptoms similar to influenza, such as high fever or pain behind the eyes. Dengue fever is also called breakbone fever because of the joint pain. Victims of dengue often have contortions due to the intense joint and muscle pain, hence the name breakbone fever.

In more severe cases, infected people may suffer from dengue hemorrhagic fever. Dengue hemorrhagic fever can possibly cause a shock. With adequate medical care, death occurs in only 1% of cases. This complication, dengue hemorrhagic fever, usually occurs in people who have previously suffered from this disease. Initially it produces more bleeding in various places in the body and there are some small red spots (petechiae) in the skin that occur because of small blood vessels rupture in the skin. Then may also occur disseminated intravascular coagulation, or blood coagulates in blood vessels. This is a body reaction to lack of oxygen due to blood loss. This form of dengue fever can be life-threatening or even fatal [29].

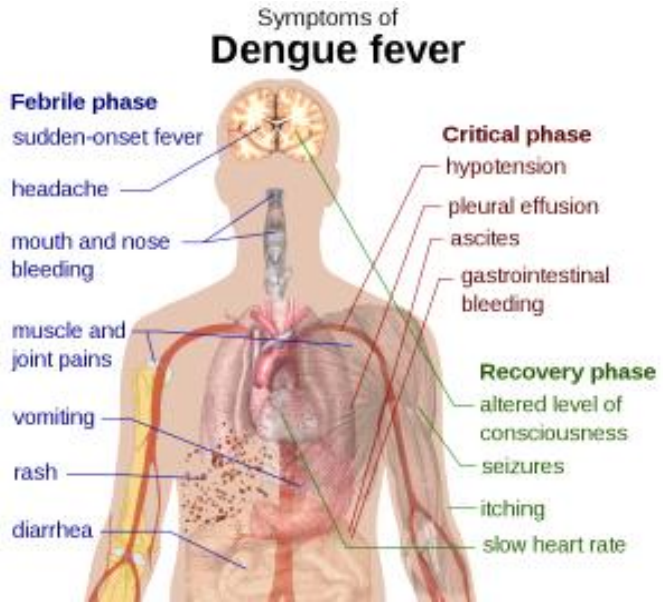


Figure 8 Dengue Symptoms

Any specific cure or vaccine for dengue fever is yet to be successfully formulated. This is the main reason behind the fatality of the dengue fever. So, the treatment of dengue is done by simple things like keeping the patient hydrated all the time, giving him healthy food, and providing him ample rest. Yes, the dengue fever can be controlled, though it cannot be properly cured. Governments of tropical countries, such as the Philippines, are taking great measures to manage dengue by controlling the mosquito's population. Those measures include the fogging operations on susceptible areas, sanitization of the moist places, releasing mosquito's larva killer Gambusia fish into the water bodies, distributing mosquito nets among the citizens and many others [31].

E. CLINICAL DECISION SUPPORT SYSTEM (CDSS)

Clinical decision support systems (CDSS) are sophisticated health IT components. It requires computable biomedical knowledge, person-specific data, and a reasoning mechanism that combines knowledge and data to generate and present helpful information to clinicians as care is being delivered. This data must be filtered, organized, and presented in a way that

supports the current workflow, allowing the end user to make an informed decision quickly and take action [33].

There are two main type of clinical decision support systems, namely: knowledge based and non-knowledge based systems. Knowledge based systems apply rules to patient data using an inference engine and displays results to the end user. While the latter, rely on machine learning to analyze clinical data. These CDSSs may be ideal for different processes of care in different settings. GNAT may be classified as non-knowledge based.

A CDSS, often, is integrated with a health care organization's clinical workflow, which is, likely, already complex. Most CDSSs are standalone products that lack interoperability with reporting and electronic health record software. Additionally, it is difficult to incorporate resulting data since the sheer number of clinical researches and medical trials being published is enough to overwhelm the system. Moreover, increasing amount of data places significant strain on application and infrastructure maintenance.

Nevertheless, such systems are utilized because it helps reduce medication dosage and diagnosis error. CDSS give health care providers the ability to easily comprehend available information, which may include proper drug-specific doses and full drug monographs. CDSSs reduce clinical misdiagnoses. Approximately, 10 to 30 percent of medical errors are diagnosis errors. CDSSs drastically improve the margin of diagnostic error. CDSSs, also, provide the entire care team with consistent, reliable information, which may be critical upon diagnosis of a complex disease [34].

IV. DESIGN AND IMPLEMENTATION

A. CONTEXT DIAGRAM

The user supplies the system with necessary data. Input data, such as the data sets, both for training and validating, neural network parameters, such the number of hidden nodes, maximum tolerable error, and the genetic algorithm parameters, such as the number of generations, population size, and mutation size is provided by the user. The system provides the user with the desired information. The system, with the use of the input data, can classify patients and provide a clinical summary with a graph representation. The Context Diagram is shown in Figure 9.

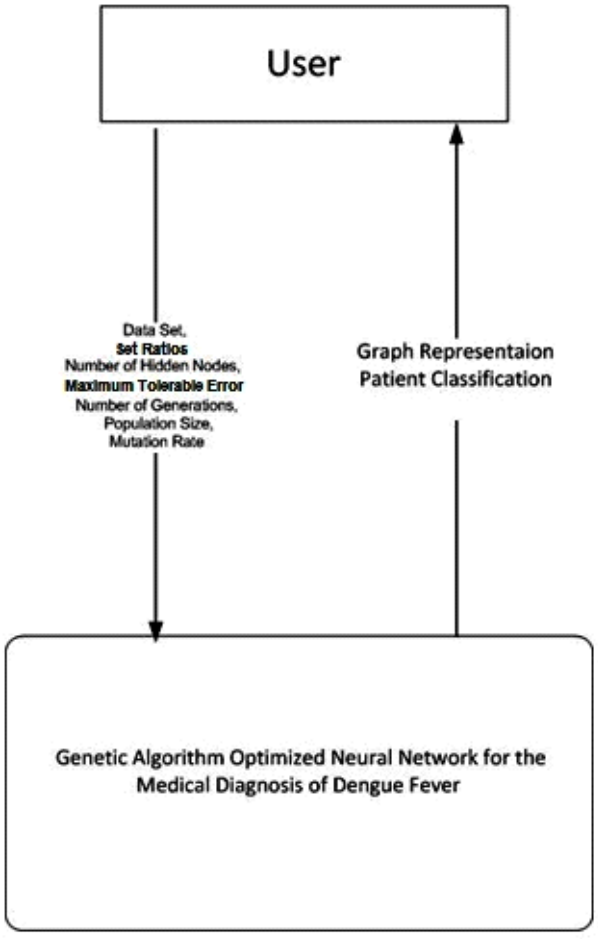


Figure 9 Context Diagram,
Genetic Algorithm Optimized Neural Network for the Medical Diagnosis of Dengue Fever

B. PROJECT FLOW DIAGRAM

Generally, the system runs in this manner,

B.1. *Determine the Network Parameters*

The user supplies the system with necessary data. Input data, such as the data set, neural network parameters, such as the number of hidden nodes, and maximum tolerable error, and the genetic algorithm parameters, such as the number of generations, population size, and mutation size.

B.2. *Randomize the initial population*

The weights of the initial members of the population are chosen at random. The random weights are uniformly distributed between -1.0 and 1.0 and are multiplied by a random weight multiplier.

B.3. *Assign input and output values to Artificial Neural Network*

With the current weights, each link in the ANN is assigned to a specific weight from the population. The computation for the input and output values are also included in this step.

B.4. *Compute Fitness*

To calculate the fitness of each chromosome, the network is trained with the dengue data and the sum of the squares of the error is returned. A small value, closes to zero, indicates the network has learned well and is suited for the classification problem. The computation for the error is indicated below.

$$\text{Sum of Squares Error (SSE)} = \sum_{i=1}^{N} (\boldsymbol{actual}_i - \boldsymbol{ideal}_i)^2$$

B.5. *Tolerable Error*

B.5.i *Tolerable Error*

If error calculated in B.4 is less than maximum error specified by the user, then the fittest chromosome is taken as the vector of optimized weight connections. Proceed to step B.6.

B.5.ii *Intolerable Error*

If error calculated in B.4. is greater than maximum error that was specified by the user, genetic operators are utilized to create a new population of weights. Weights are adjusted accordingly in each link. Repeat step B.3.

B.6. *Apply vector of weights to Feedforward Neural Network*

Optimized weights are assigned to each link in the Neural Network.

B.7. *Evaluate Performance by Introducing Validating Data*

Accuracy of the optimized weights is determined by feeding the network with testing data and comparing the obtained values with the desired values.

B.8. *Optimized ANN for Dengue Fever Diagnosis*

The network can now be utilized to classify patients.

The Project Flow Diagram is depicted in Figure 10.

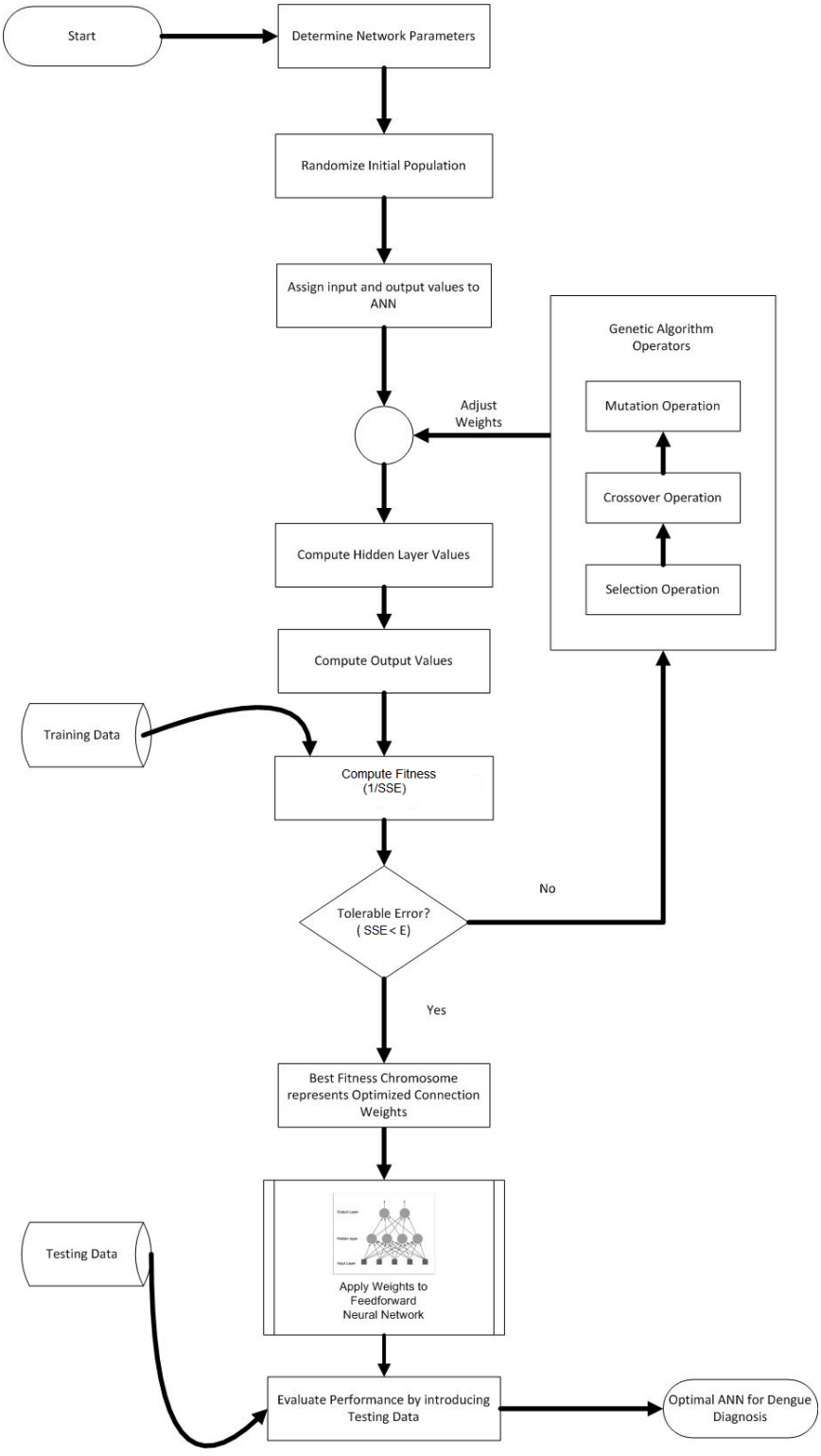


Figure 10 Genetic Feedforward Neural Network

C. DATA FLOW DIAGRAM

TOP-LEVEL DFD

User can add data, such as the network parameters and the genetic parameters, into the system. The parameters, together with the training data set are used to train the neural network. After the training of the neural network, the validating data set can be used to evaluate the performance of the network. If the neural network is found to be satisfactory, it can now be used for patient diagnosis. The system provides the user with a clinical summary of the validating data set and can classify each patient. The top level DFD is shown in Figure 11.

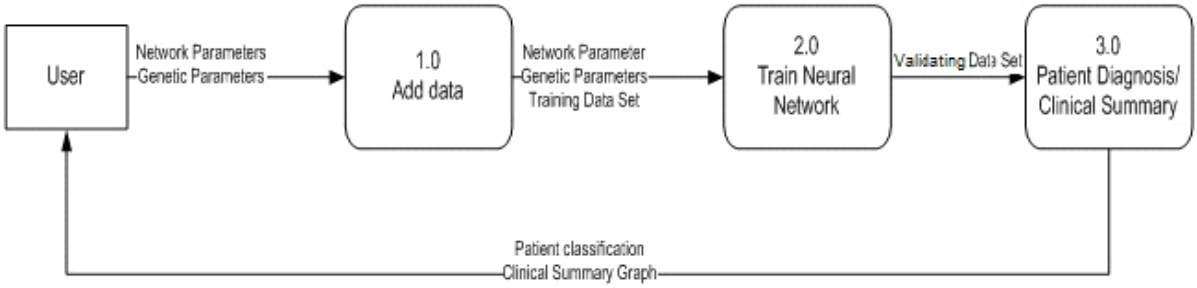


Figure 11 Data Flow Diagram,
Genetic Algorithm Optimized Neural Network for the Medical Diagnosis of Dengue Fever

DFD SUBEXPLOSIONS

Subexplosion: Level 1 – 1.0 Add data

To add data into the system, the network parameters are provided by the user. This includes: the number of hidden nodes; and the maximum tolerable error. Also, the genetic parameters are provided by the user, namely: the number of generations; population size; and the mutation rate. All of the above information is used in further steps. The DFD subexplosion is shown in Figure 12.

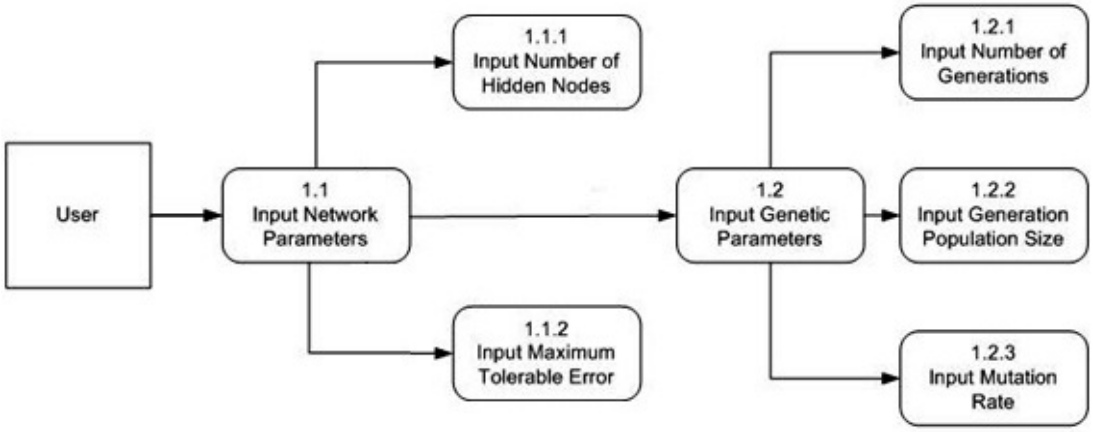


Figure 12 Subexplosion 1.0
Add Data

Subexplosion: Level 1 – 2.0 Train Neural Network

With the given network and genetic parameters, and the data set, the neural network is trained. An initial vector of weights is generated randomly. The given sets' fitness is determined with use of the sum of squares error. The set is found fit if the error is less than the maximum tolerable error provided by the user. It is then applied to the neural netowork. Otherwise, another set is generated with the use of genetic algorithms. The generation of another population can also be terminated if the number of generations has exceeded that of the number provided by the user. The DFD subexplosion is shown in Figure 13.

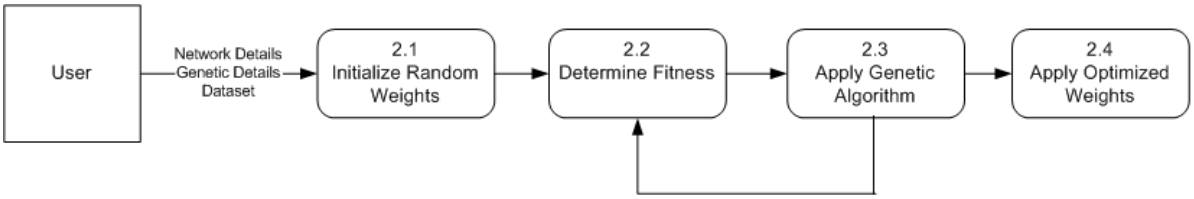


Figure 13 Subexplosion 2.0
Train Neural Network

Subexplosion: Level 1 – 3.0 Clinical Summary/Patient Diagnosis

With the optimized neural network, the system is set to diagnose a patient with the disease. The patient data is provided and the system provides a diagnosis. A clinical summary is provided if more than one patient data or a separate patient data set is provided.

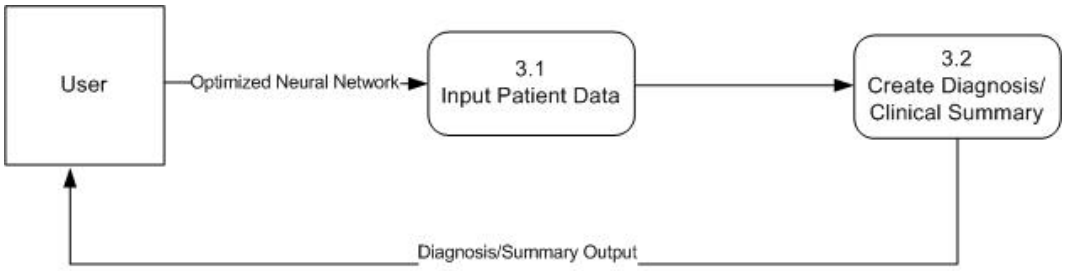


Figure 14 Subexplosion 3.0
Patient Diagnosis/Clinical Summary

V. Architecture

A. Hardware Requirements

GNAT requires at least:

- 2GB RAM
- Processor speed of 2.2 GHz

B. Software Requirements

GNAT requires at least:

- Java Runtime Environment of at least version 1.6

VI. Results

In general, the figures below depict how the Genetic Neural Network Analytic Tool (GNAT) functions. Additionally, this section also shows the effect of the varying parameters on the validation rate of an optimized genetic neural network.

The main screen of the GNAT is shown on Figure 15. All the application functions can be seen within this page. This includes uploading a data set, configuring the neural network and genetic algorithm parameters, training the neural network, classifying, either batch or single entry, patients, and exporting diagnosis reports.

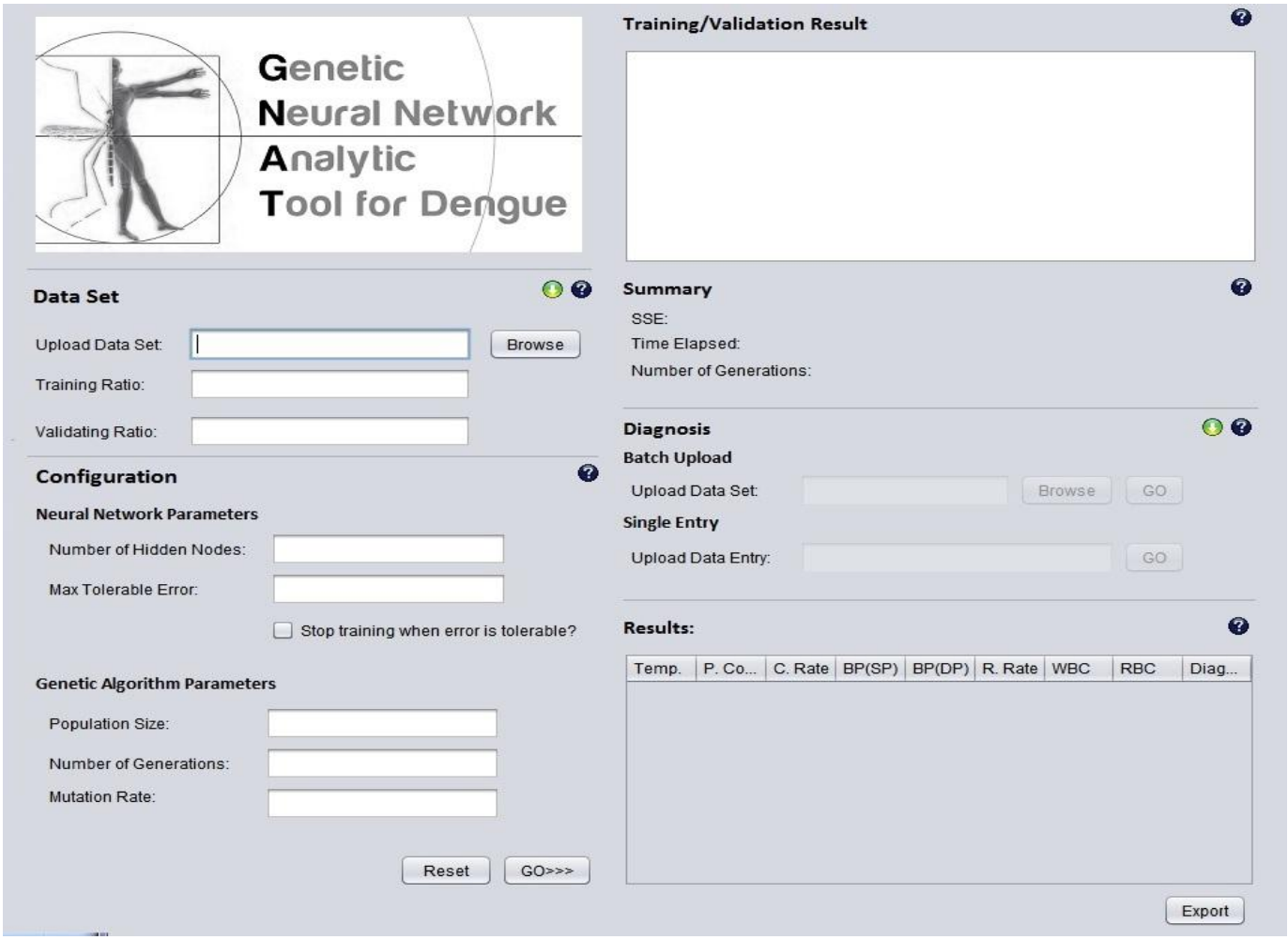


Figure 15 GNAT Main Screen

Within this screen, the user can choose to upload and configure the data set, as shown in Figure 16. The user can use the browse button to upload a desired data set. Also, the user can provide ratio at which the data set is divided, a part is used for training and the other for validating the neural network.

Figure 16 Data Set Configuration Panel

A training and validating data set template can also be downloaded from the application via the download button, the one beside the data set help button. As shown in Figure 17, the application provides a file that contains the proper format of the training/validating data sets. The user's file must conform to the template to be properly handled by the application.

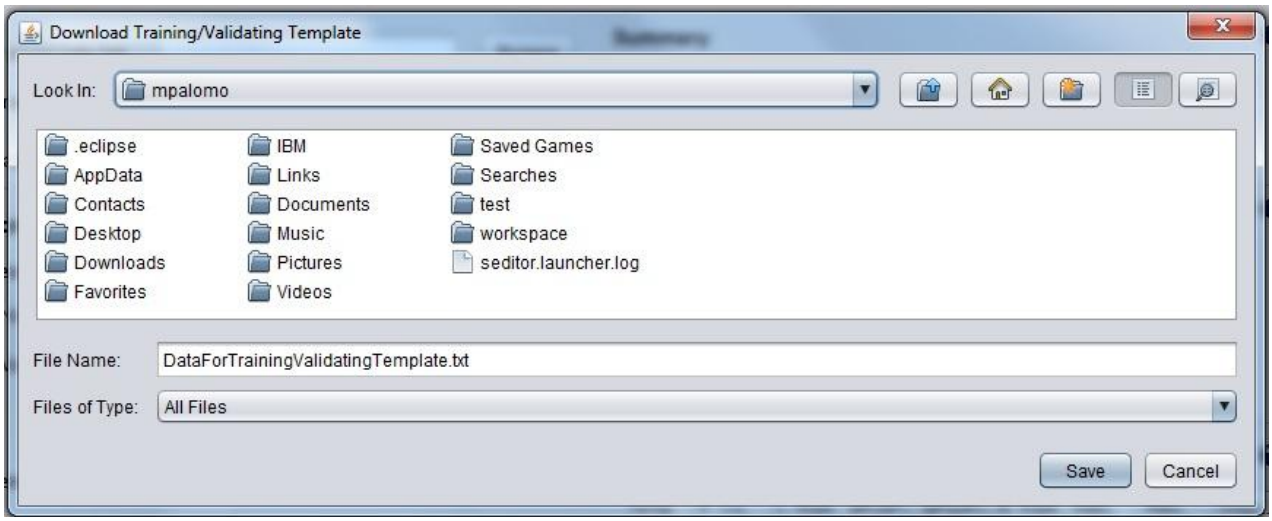


Figure 17 Download Training/Validating Template

The user can upload a data set, as shown in Figure 18, and upon selection of a valid data set, the user can provide the necessary parameters for both the neural network and the genetic algorithm parameters.

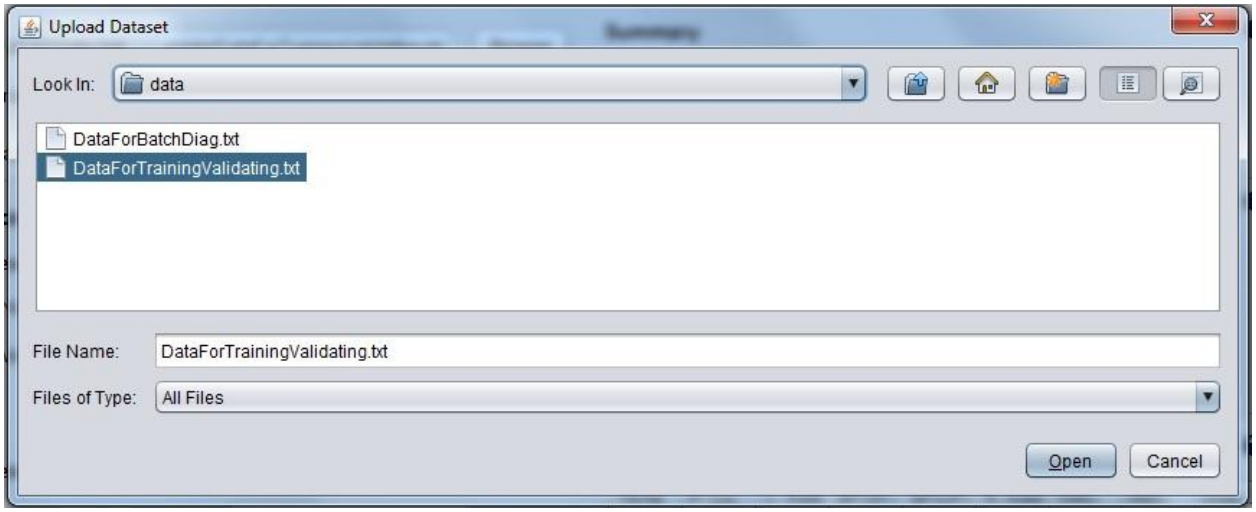


Figure 18 Upload Training/Validating Data Set

As shown in Figure 19, the user can provide the neural network parameters, namely: number of hidden nodes; max tolerable error; and stop the training if the error is tolerable.

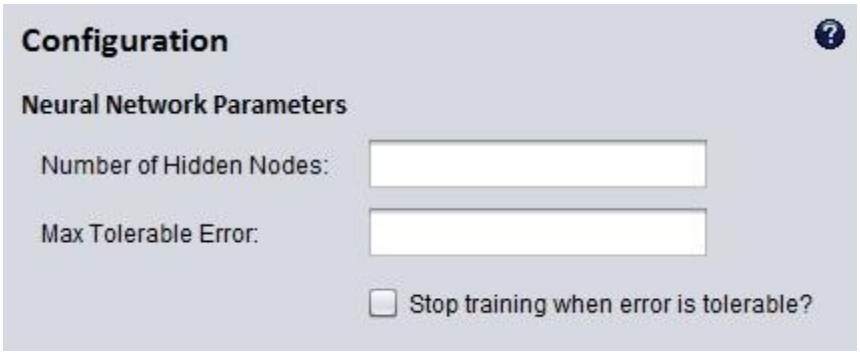


Figure 19 Neural Network Configuration Panel

As shown in Figure 20, user can provide the genetic algorithm parameters, namely: population size; number of generations; and the mutation rate.

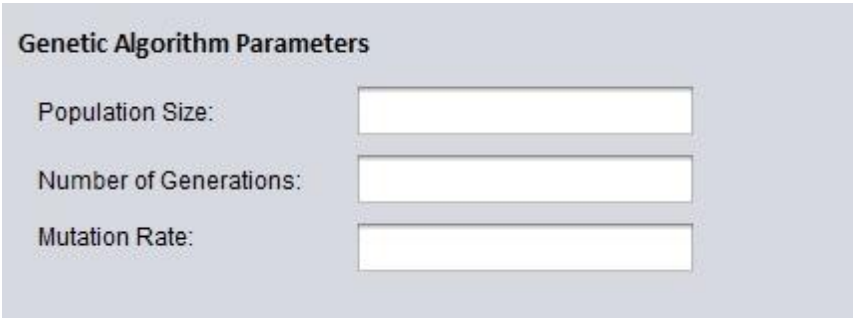


Figure 20 Genetic Algorithm Configuration Panel

With the data set and the parameters supplied, the user can start the training by clicking the go button or reset button, if not. The buttons are shown in Figure 21.

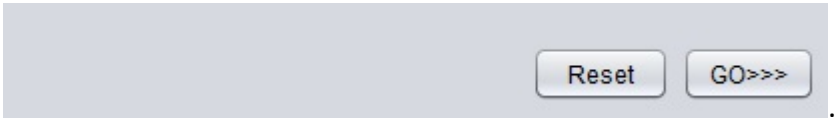


Figure 21 Reset and Go Buttons

During the training of the neural network, a graph shows the sum of squares error of the fittest chromosome of the current generation versus the current time. Shown in Figure 22, the chart panel also contains buttons that the user can be used to either start, pause, or stop the training. When the user starts training the neural network, it can be paused, and be continued later on. Additionally, it can also be stopped or ended with the use of the end button. Figure 23 depicts an ongoing training wherein the changes in SSE in shown though a line graph.

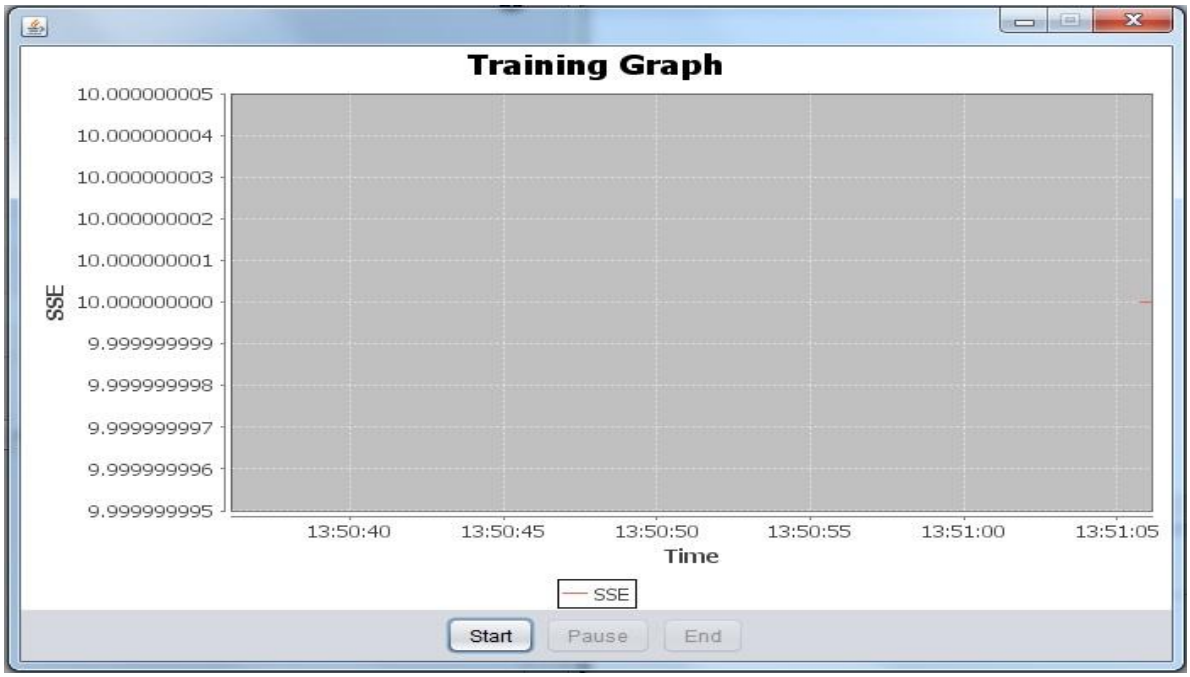


Figure 22 Training Graph Panel

(before starting)

Figure 23 Training Graph Panel

(with an ongoing training)

Upon the start of the training, the application displays the current sum of squares error of the fittest candidate in the training result textarea, as shown in Figure 24. When the network error is tolerable, the application may stop training if the user opted and ticked the checkbox to stop the training.

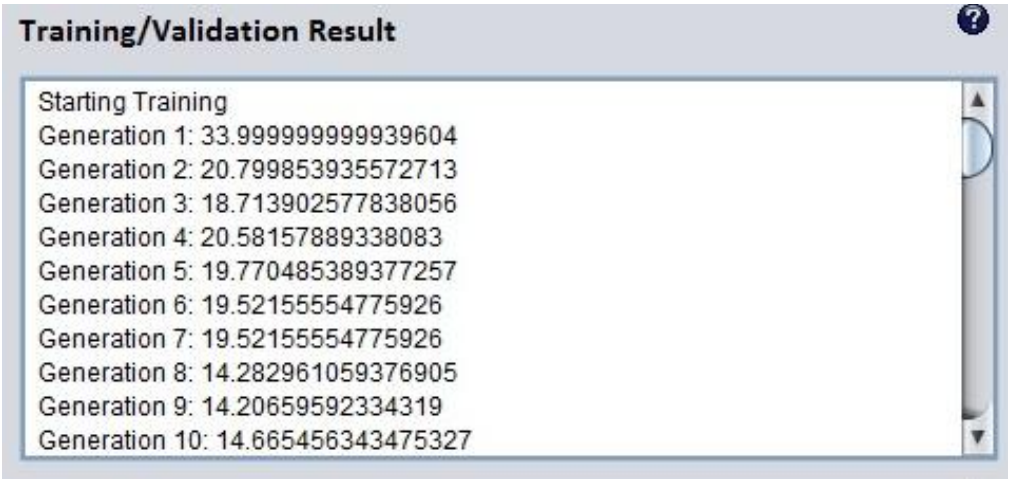


Figure 24 Training/Validation Result

Shown in Figure 25, upon reaching a tolerable error or reaching the number of generations, the application then validates the network by using the validating set from data set provided by the user.

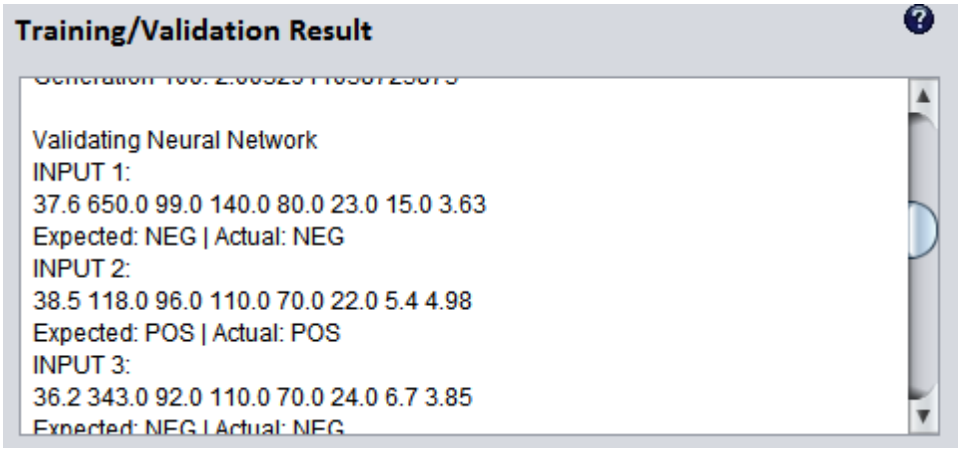


Figure 25 Neural Network Validation

After which, the system provides a summary, shown in Figure 26, which contains: final sum of square errors; time elapsed; and the number of generations.

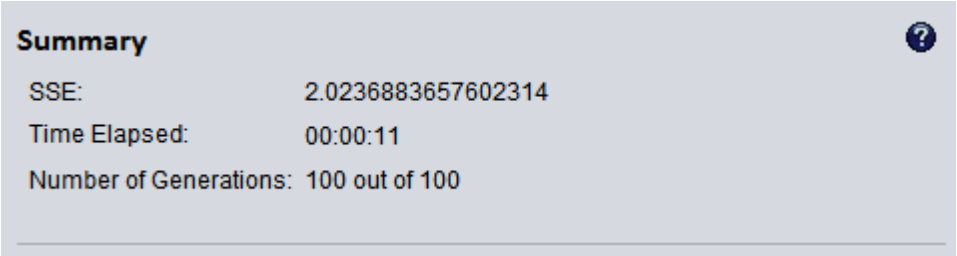


Figure 26 Summary Panel

After training, validating, and obtaining the optimized neural network, it can now be used to classify patients. As shown in Figure 27, the diagnosis data set can be provided via a data set, or a single entry. Given a data set or an entry, it can be diagnosed when the go button is clicked.

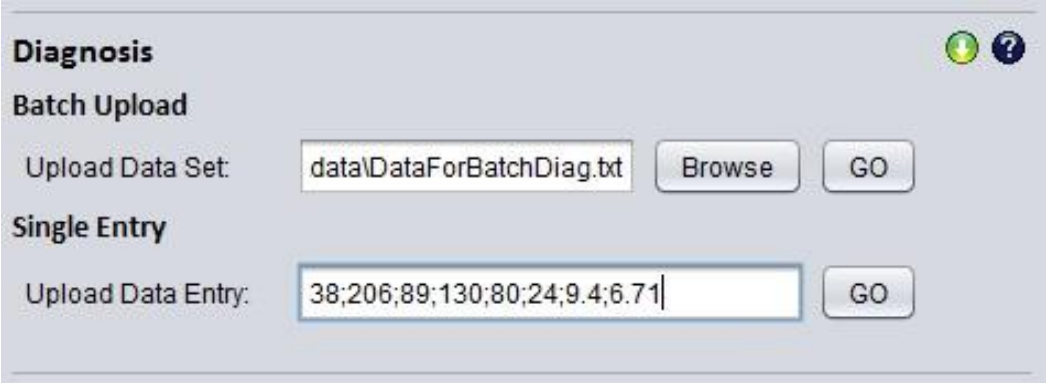


Figure 27 Diagnosis Data Set Panel

A diagnosis data set template can also be downloaded from the application via the download button, the one beside the data set help button. As shown in Figure 28, the application provides a file that contains the proper format of the diagnosis data sets. The user's file must conform to the template to be properly handled by the application.

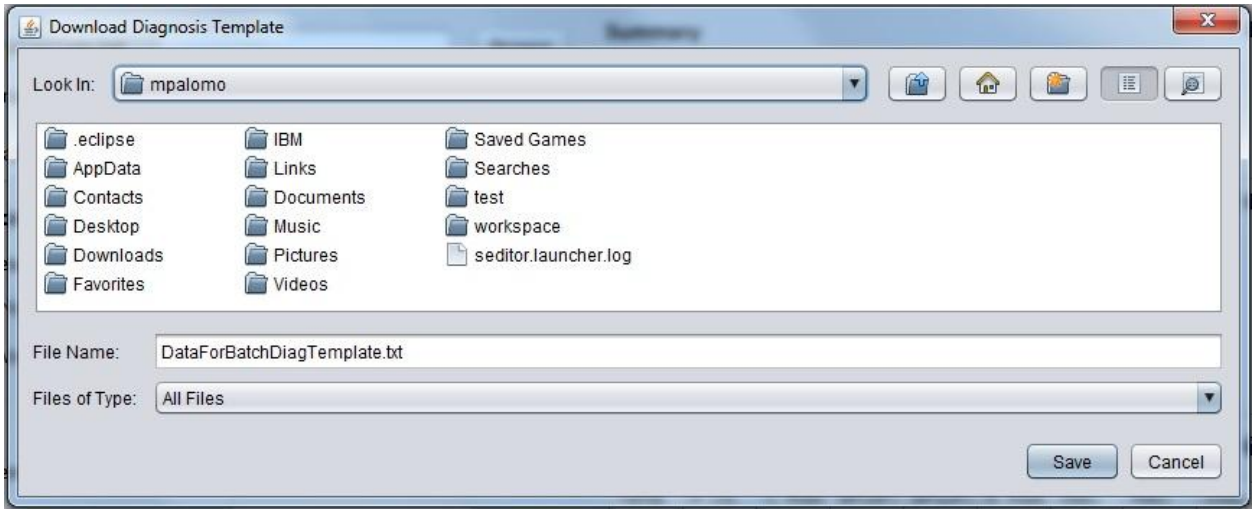


Figure 28 Download Batch Diagnosis Template

The diagnosis result is shown through a table. Figure 28 shows the results table. On each diagnosis, the result is appended at the end of the table. The table contains the patient data and the diagnosis of each, as shown in Figure 29.

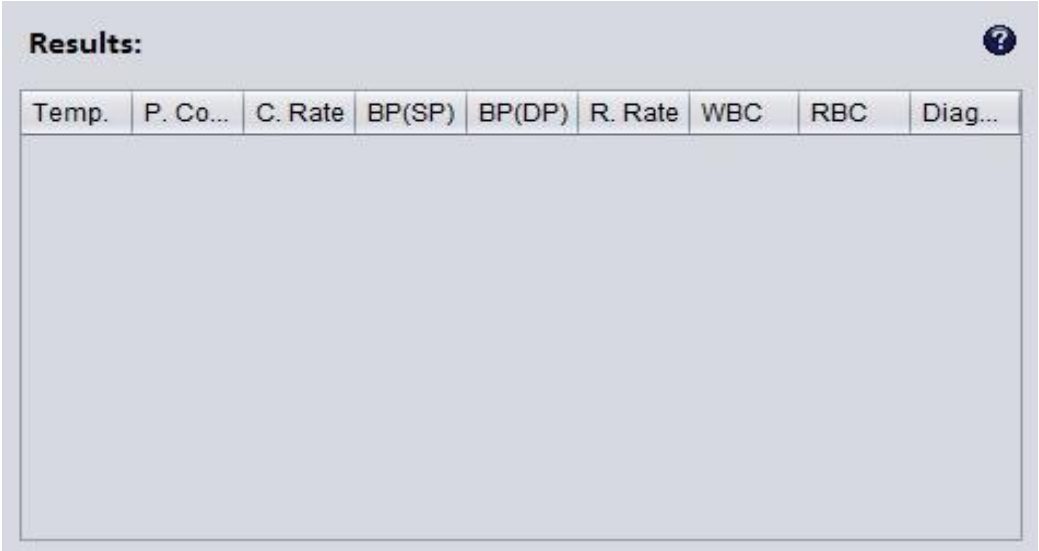


Figure 29 Empty Results Table

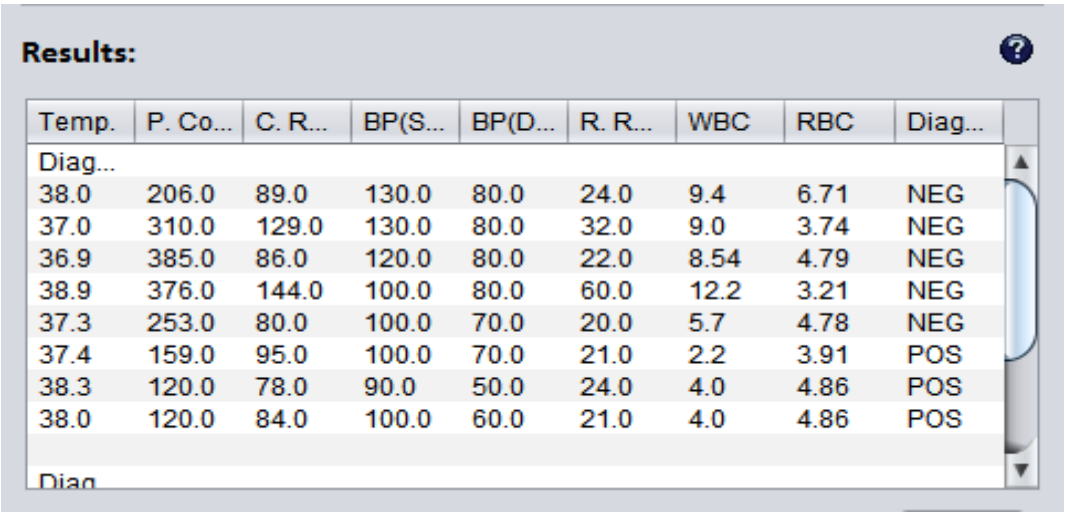


Figure 30 Active Results Table

Furthermore, shown in Figure 30, the user can export the diagnosis results in a system file by clicking the export button.

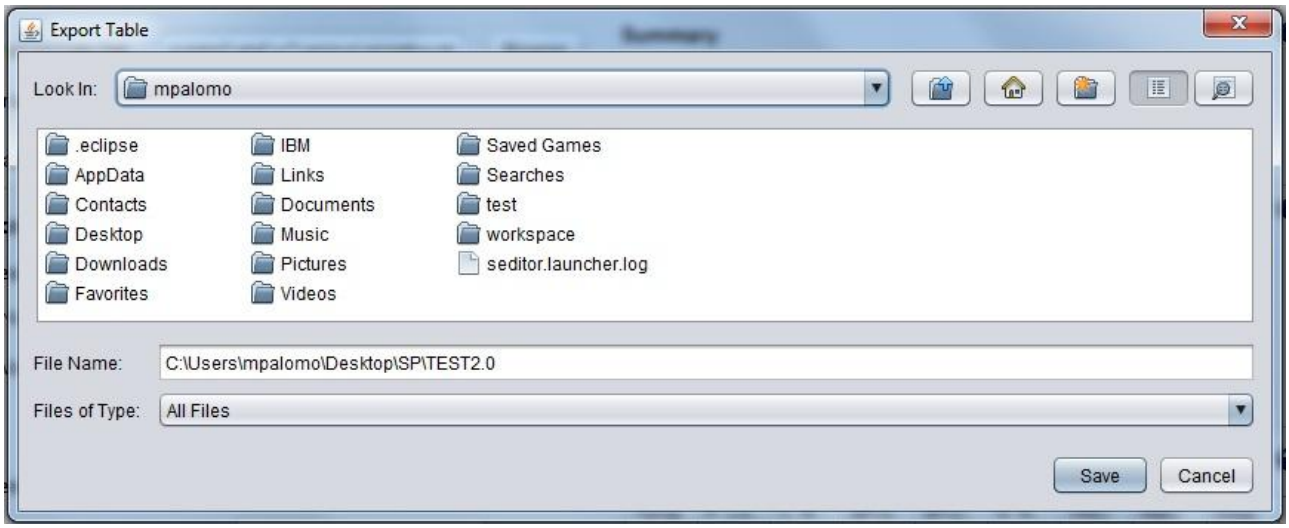


Figure 31 Export Table

Accordingly, the user may view help dialogs, by clicking the help buttons, as shown below:

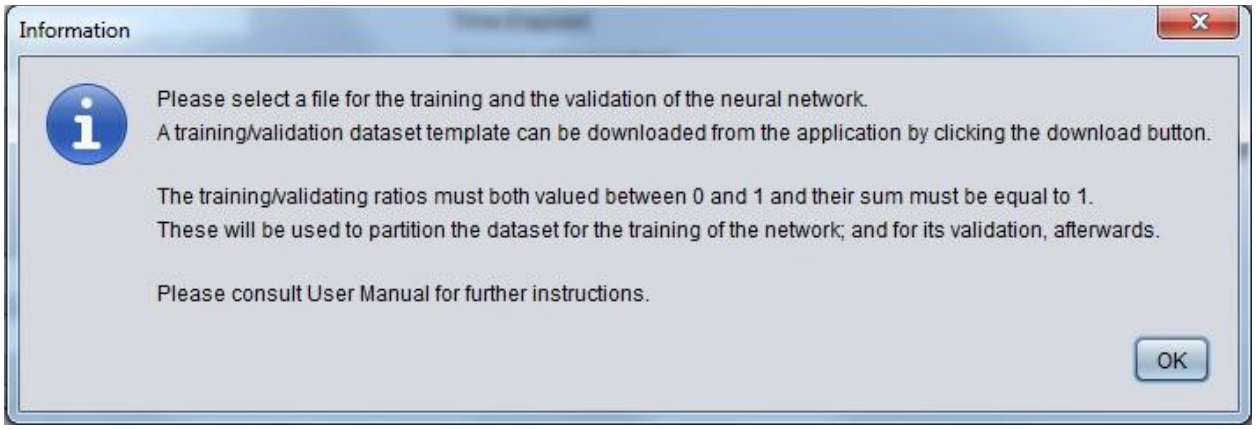


Figure 32 Training/Validating Data Set Help Dialog

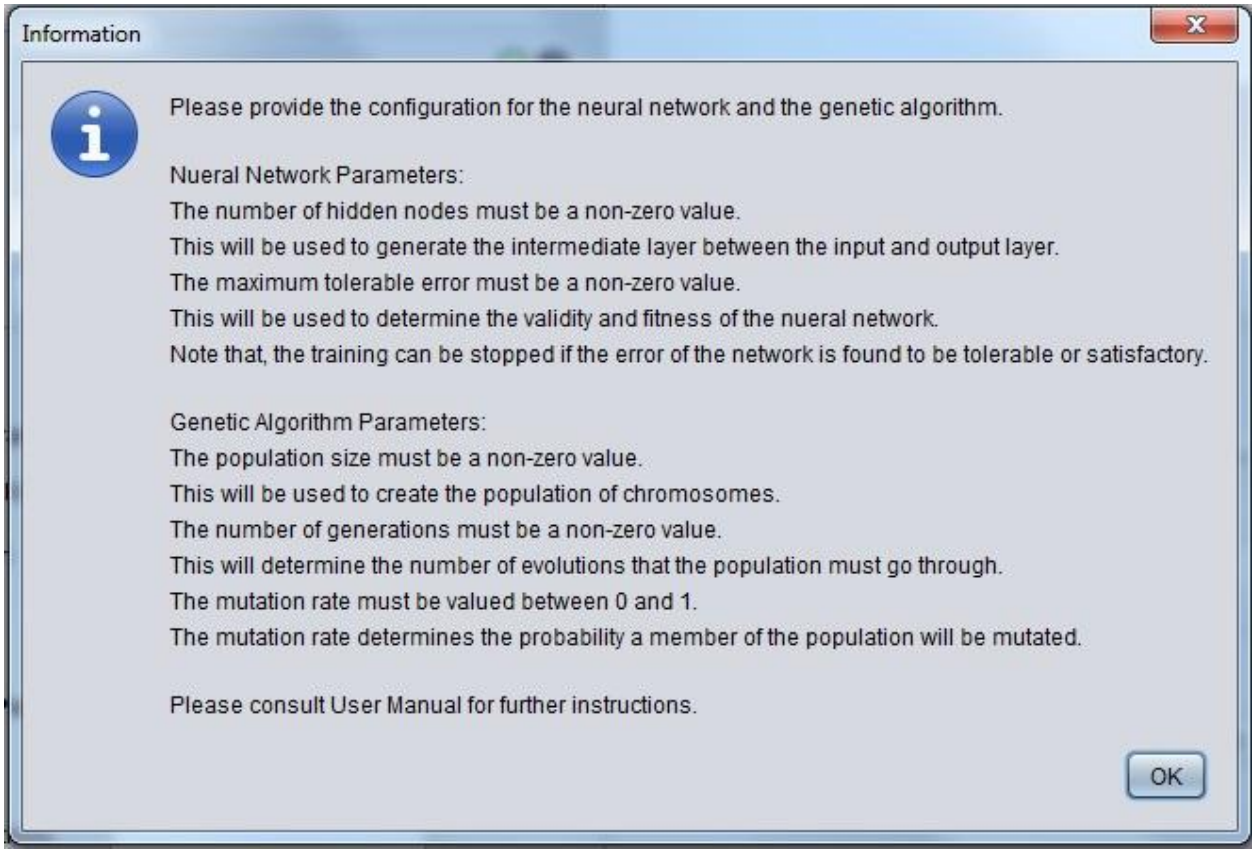


Figure 33 NN/GA Configuration Help Dialog

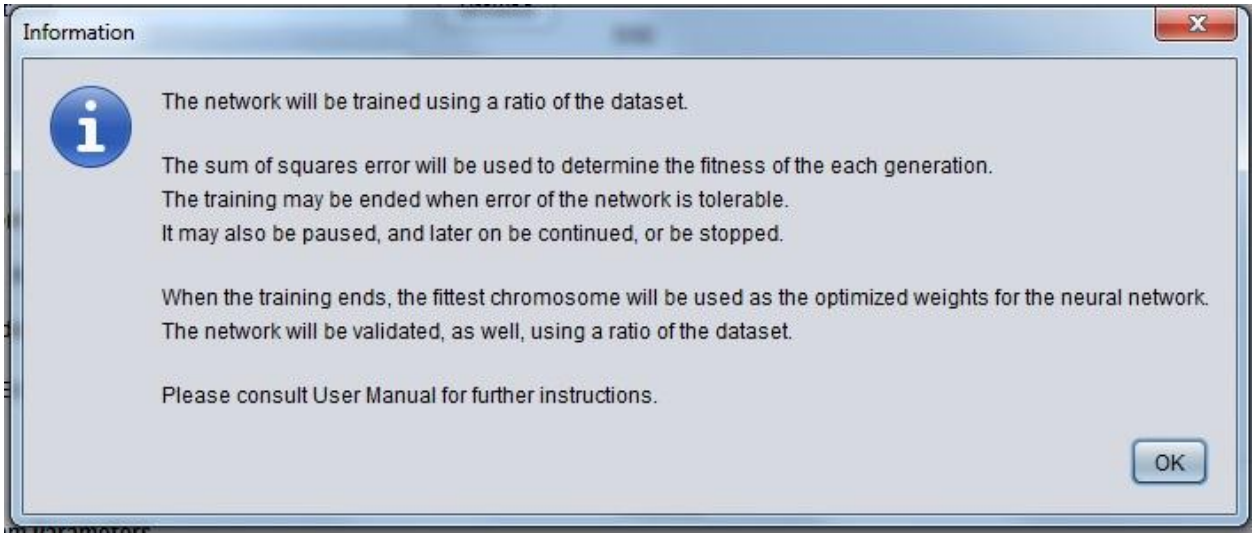


Figure 34 Training/Validating Help Dialog

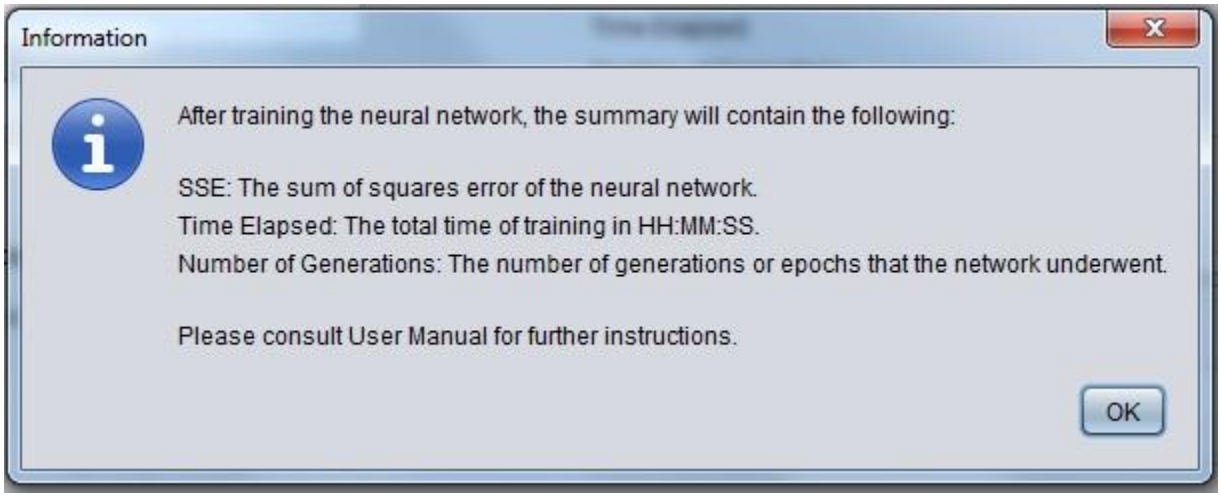


Figure 35 Summary Help Dialog

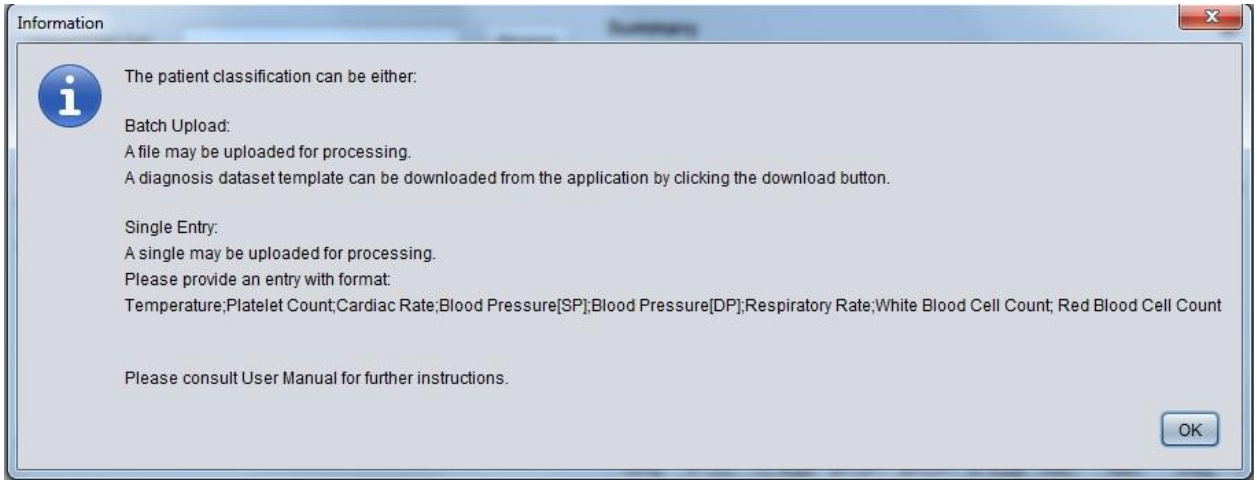


Figure 36 Diagnosis Data Set Help Dialog

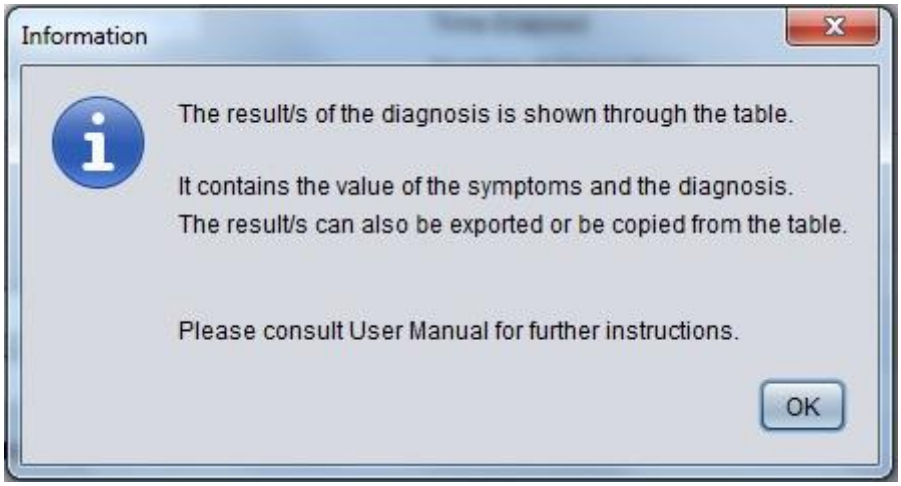


Figure 37 Diagnosis Results Help Dialog

Regarding the performance of GNAT, it is tested by setting all, but the tested parameter, with arbitrary values that remains constant across a particular test run. The default values for each parameter are depicted in Table 7.

Parameters	Default Value
Data Set (Training Ratio)	0.5
Data Set (Validation Ratio)	0.5
Neural Network (Number of Hidden Nodes)	6
Neural Network (Maximum Tolerable Error)	0.01
Genetic Algorithm (Population Size)	300
Genetic Algorithm (Number of Generations)	1000
Genetic Algorithm (Mutation Rate)	0.05

Table 7 Default Parameter Values

GNAT is tested for the effect of varying parameters on the validation rate of the optimized neural network. The ratio, for both training and validation sets are kept constant, at 0.05. Both the training and validation data sets contain 55 sample data. Also, the maximum tolerable is kept constant at 0.01. The parameters tested are as follows: Number of hidden nodes (6, 7, 8, 9, 10); Population size (300, 400, 500, 600, 700); Number of generations (1000, 2000, 3000, 4000, 5000); and Mutation Rate (0.01, 0.03, 0.05, 0.07, 0.1, 0.5). Tables below show the results of the various test runs.

Parameters	Run1	Run2	Run3	Run4	Run5
Data Set (Training Ratio)	0.5	0.5	0.5	0.5	0.5
Data Set (Validation Ratio)	0.5	0.5	0.5	0.5	0.5
Neural Network (Number of Hidden Nodes)	6	7	8	9	10
Neural Network (Maximum Error)	0.01	0.01	0.01	0.01	0.01
Genetic Algorithm (Population Size)	300	300	300	300	300
Genetic Algorithm (Number of Generations)	1000	1000	1000	1000	1000
Genetic Algorithm (Mutation Rate)	0.05	0.05	0.05	0.05	0.05
Results (Sum of Squares Error)	$1.6x10^{-72}$	$7.2x10^{-4}$	$2.6x10^{-41}$	$7.3x10^{-4}$	$2.1x10^{-32}$
Results (Time Elapsed)	00:03:49	00:04:00	00:04:18	00:04:31	00:03:52
Results (Validation Rate)	75.0%	67.0%	73.0%	78.0%	71.0%

Table 8 Varying NN: Number of Hidden Nodes

As shown in Table 8, the number of hidden nodes is varied and the validation rate is in the range of 67.0% to 78.0%. 9 hidden nodes is have the best validation rate, while 7 seven hidden nodes have the worst validation rate.

Parameters	Run1	Run2	Run3	Run4	Run5
Data Set (Training Ratio)	0.5	0.5	0.5	0.5	0.5
Data Set (Validation Ratio)	0.5	0.5	0.5	0.5	0.5
Neural Network (Number of Hidden Nodes)	6	6	6	6	6
Neural Network (Maximum Error)	0.01	0.01	0.01	0.01	0.01
Genetic Algorithm (Population Size)	100	200	300	400	500
Genetic Algorithm (Number of Generations)	1000	1000	1000	1000	1000
Genetic Algorithm (Mutation Rate)	0.05	0.05	0.05	0.05	0.05
Results (Sum of Squares Error)	$3.0x10^{-5}$	0.07	0.51	0.49	$4.0x10^{-10}$
Results (Time Elapsed)	00:02:00	00:03:12	00:04:13	00:04:54	00:05:58
Results (Validation Rate)	93.0%	87.0%	91.0%	98.0%	87.0%

Table 9 Varying GA: Population Size

As shown in Table 9, the population size, of each generation, is varied and the validation rate is in the range of 87.0% to 98.0%. 400, as the population size, have the best validation rate, while 200 and 500, as population sizes, have the worst validation rate.

Parameters	Run1	Run2	Run3	Run4	Run5
Data Set (Training Ratio)	0.5	0.5	0.5	0.5	0.5
Data Set (Validation Ratio)	0.5	0.5	0.5	0.5	0.5
Neural Network (Number of Hidden Nodes)	6	6	6	6	6
Neural Network (Maximum Error)	0.01	0.01	0.01	0.01	0.01
Genetic Algorithm (Population Size)	300	300	300	300	300
Genetic Algorithm (Number of Generations)	1000	2000	3000	4000	5000
Genetic Algorithm (Mutation Rate)	0.05	0.05	0.05	0.05	0.05
Results (Sum of Squares Error)	2.4^{-43}	$3.0x10^{-32}$	$4.8x10^{-31}$	5.7^{-32}	$4.0x10^{-31}$
Results (Time Elapsed)	00:10:24	00:07:55	00:12:29	16:29	00:22:02
Results (Validation Rate)	75.0%	82.0%	80.0%	85.0%	78.0%

Table 10 Varying GA: Number of Generations

As shown in Table 10, the number of generations was varied and the validation rate was in the range of 75.0% to 85.0%. 4000, as the number of generation, have the best validation rate. While 1000, as the number of generation, have worst validation rate.

Parameters	Run1	Run2	Run3	Run4	Run5	Run6
Data Set (Training Ratio)	0.5	0.5	0.5	0.5	0.5	0.5
Data Set (Validation Ratio)	0.5	0.5	0.5	0.5	0.5	0.5
Neural Network (Number of Hidden Nodes)	6	6	6	6	6	6
Neural Network (Maximum Error)	0.01	0.01	0.01	0.01	0.01	0.01
Genetic Algorithm (Population Size)	300	300	300	300	300	300
Genetic Algorithm (Number of Generations)	1000	1000	1000	1000	1000	1000
Genetic Algorithm (Mutation Rate)	0.01	0.03	0.05	0.07	0.1	0.5
Results (Sum of Squares Error)	$3.6x10^{-5}$	$5.8x10^{-27}$	$1.3x10^{-32}$	$3.2x10^{-17}$	$1.4x1^{-19}$	$2.3x10^{-32}$
Results (Time Elapsed)	00:04:11	00:04:13	00:03:57	00:04:17	00:03:55	00:03:51
Results (Validation Rate)	67.0%	76.0%	89.0%	85.0%	75.0%	71.0%

Table 11 Varying GA: Mutation Rate

As shown in Table 11, the mutation rate is varied and the validation rate is in the range of 67.0% to 89.0%. 0.05, as the mutation rate, have the best validation rate, while, 0.01, as the mutation rate, have the worst validation rate.

VII. Discussion

The Genetic Neural Network Analytic Tool (GNAT) is designed to work as efficiently as possible without the assistance of the user. All the functions are automated and processed seamlessly in background. Given a sole data set of clinical reports and the necessary parameters, the application is able to create an optimized neural network that helps determine and classify patients based on their measured features.

With a valid data set, the application creates two subsets of the data set; one is used for training; and the other for validation. Note that, all malformed data values are removed from the data set and are not used to train or validate the network. It is advised to provide a greater number of training data against validating data, in order to have a better generalization about the whole data set. Moreover, the data set must be of type .txt, .csv, or .dat to be processed by the application. The individual values must also be separated by a semi-colon, ';', and each patient data must be separated by a new line.

The neural network parameters include the number of hidden nodes, maximum tolerable error, and the option to stop training if the maximum error is met. For this paper the neural network is limited to 8 input nodes for the input layer. These include numerical values from a combination of vital signs and data from hematological reports. All data are normalized to value ranging from -1 to 1. To further discuss, normalization is done since the contribution of an input heavily depends on its variability relative to other input values. It is essential to normalize the data since input values with a larger value range, often, have a more significant effect as compared to input values that have a lesser value range. With normalized data, their variability represents their importance. Additionally, the neural network is limited to a sole hidden layer, but the user is given a choice to provide the number of hidden nodes for this layer. Generally, the number of hidden nodes is between the number, of input nodes, in this case 8, and the number output nodes, in this case 1. Hidden nodes are necessary since it provide nonlinearity to the generated neural network. Also, the user can provide a maximum

tolerable error, in which the network must satisfy or reach in order for it to be considered ready for classification. Often, it may take a while for a neural network to converge, but having genetic algorithm as its training algorithm takes advantages of the nature of evolution, thus, it may indiscriminately reach convergence faster, as compared to other training methods.

Together with neural networks, as stated above, genetic algorithm is utilized as its training algorithm. Genetic algorithm takes advantage the biological nature of evolution. The sets of weights are encoded in a format that is usable and alterable by the application. Value encoding is used for encoding, where all the data remained as is, but is normalized to remove the dimensionality of the data. The solutions that are fit enough are allowed to crossover producing the next sets of weights until another population is created. Mutation may also randomly occur on the individual, this may help introduce solution that may have been left out or may have no relevance to the current or generated population. Mutation encourages new traits that may or may not resemble the correct set of weights. If a mutation may prove harmful, it is not be allowed to produce another generation. But if a mutated individual was proven helpful or fit, it may be used as parent of the succeeding generation.

Training starts by creating a random set of solutions, where its length is based on the number specified by the user. Genetic operators, such as selection, crossover, and mutation, is applied unto these until a terminating condition is satisfied, which is either a tolerable error or the number of generation has been reached. The population generated is based on the parameters set by the user. The population size refers to the number of candidates per generation. Generally, it is advised to provide a high number. Though it may lengthen the processing time, it helps find the optimum weights, those close to the global optima, in a relatively efficient way. More candidates increase the chance of finding the best set weights. The number of generations specifies the number of epochs or iterations in which the generic operators is utilized to create fitter populations. Finally, mutation rate is essential since it provides and introduces variability to the population, which can be homogenous since the operators are repeatedly applied unto it. It, ultimately, prevents having the solution stuck at a

local minimum. Often, the mutation rate must be kept low, just enough to provide variability and preserve the traits of the population at hand. Note, as well, that each neural network training the randomized initial weights vary, which may hinder or hasten the convergence of a particular training session.

Each population generated is the tested for its fitness. The weights contained within that population are used as neural network weights and the actual diagnosis is compared to the ideal diagnosis, which is provided by the data set. Its sum of squares error is used as its fitness. Once the error is satisfied or the number of generations is reached, the optimized neural network is used for classification.

The optimized neural network is validated and its performance is checked. From the base data set, the validation subset is inputted into the neural network. Its diagnosis is verified based on the ones specified by the subset versus the ones from the neural network diagnosis. The diagnosis of the network is cross checked with the data from the clinical records which contains the health care providers' diagnosis. It has been observed that the greater the number of generation, the better its performance, it terms of diagnosis rate, upon validation. Once done, the neural network is ready for diagnosis.

The batch diagnosis accepts files of type: .csv, .dat, and .txt. All patient columns must be separated by a semicolon,';', and all patient rows is separated by a new line, while the single entry diagnosis, only accepts an input that is delimited by a semicolon, ';'. Much like the training and validation data set, the diagnosis, both by batch and single entry, is normalized, using the same indices for the training/validating data sets. The data from the diagnosis data set is adjusted accordingly. Once more, the malformed data is removed from the data set and excluded from the diagnosis.

Once normalized, it is fed to the optimized neural network that was obtained through the training. The output of the neural network is either positive or negative for dengue. All diagnosis report can be found at the diagnosis report table. It displays the inputted patient data and the system's interpretation. The results can be exported and be saved as a system file.

GNAT undergoes testing by altering parameters and determining the validation rates of each of the setups. The neural network and genetic algorithm parameters that are varied are as follows: for the neural network, the number of hidden nodes; for the genetic algorithm, the population size, number of generations, and the mutation rate.

The validation rate of each genetic neural network is determined by comparing the output produced by the optimized genetic neural network with the output provided by the validation subset of the training/validating data set. All the output of the network is compared and all its true positives and true negatives is divided by the total number of data samples for the validation subset. Several tests are conducted to determine optimal parameters for the genetic neural network.

Currently, there is no known convention at which dictates the proper number of hidden nodes for a certain feed forward neural network but several studies confirm that the optimal number of hidden nodes should be a number between the number of input nodes and the number of output nodes, which are 8 and 1 respectively. As for this study, at worst, it can be observed that 7 hidden nodes produce a genetic neural network with a 67.0% validation rate, while a setup of 9 hidden nodes produce a genetic neural network with a 78.0%.

The population size refers to the number of individual is a generation. A small number of individuals could produce a stagnant trait pool while a large number could, often, produce a diverse trait pool but could, ultimately, lengthen the process and training time of the genetic neural network. As observed from the tests, the best result is having 400 individuals per generation; it produces a genetic neural network with a validation rate of 98.0% while having

200 and 500 individuals, both produce a genetic neural network with a validation rate of 87%. Furthermore, it can be observed, that the time elapsed approximately increases by a minute per 100 individuals added to the population per generation.

The number of generations refers to each epoch at which the members of a certain population is allowed to reproduce, and create the succeeding generation, by several different genetic operators. Much like the population size, an increase in the number of generations can significantly increase training and processing time. It is suggested that the number of generation be limited to flexibly balance the validation rate, the sum of squares error, and the training time. As observed from the tests and as expected, 1000 generations produces the worst validation rate while 4000 generation best validation rate. But, as predicted, as well, the processing time significantly increases as the number of generations increases.

The mutation rate refers to the rate at which, an alteration to a certain trait is introduced into the population. Mutation rate is essential since it helps and, could possibly, introduce an optimal solution to the current set of solution but, conversely, it can also introduce malicious traits that can cause unwanted solutions, is not allowed to produce a successor in the next generation. As observed from the tests, 0.05 as mutation rate produces the best validation rate and 0.01 as mutation rate has the worst validation rate. It can be stated that a lack of mutation can be detrimental to the population as the optimal solution can be missed out on the initial randomization of weights.

Another observation is that the sum of squares errors, of each produced genetic neural network, are superlatively small and are all almost zeroes. This can be caused by the small number of samples in the training subsets. Consequently, the training times are significantly cut with same reason as stated above. Also note that, the initial randomization of weights, the random selection of the samples in each data subsets, plus the random evolutionary trait of genetic algorithms may and can have a significant bearing the validation rate of certain genetic neural network.

VIII. Conclusion

This paper has achieved a novel way of dengue diagnosis. Genetic Neural Network Analytic Tool (GNAT) is implemented with the use of neural networks with genetic algorithm as its training method. The system has taken advantage of the generalization ability of neural networks and the evolutionary capabilities of genetic algorithms.

As a result, a rapid, non-invasive, and an efficient tool for dengue diagnosis is created. The application paves way for users to further understand and model complex clinical data. With the application of genetic operators and the computation of weights on encoded solutions, the genetic neural networks from GNAT have been proven to have the ability to see data patterns via probabilities and statistics, which can be difficult to determine by plain human processing and analysis. Furthermore, with the incorporation of genetic algorithm, the system is able to train the neural networks in an intelligent and much efficient manner as compared to other training methods. Genetic algorithm, excellent function optimizers, helps achieve nearly globally optimal set of weights in a relative short period of time. It also provides tuning flexibilities for either performance or cost efficiency. Though limited to weight optimization, the genetic algorithm may also be utilized to evolve other essential parameters such as node count, selection of activation functions, and evolution of network architecture.

GNAT, as a clinical decision support system, may help provide health care providers a robust way of verifying their findings and cross checking clinical data to further prove the positivity of their diagnosis. GNAT may also prove critical when immediate diagnosis may be needed to identify the disease and combat further infection. Moreover, though the diagnosis of the system may vary with the diagnosis of a health care provider, the health care provider still have the final say.

IX. Recommendations

GNAT, or genetic neural network analytic tool, currently, is specialized for dengue diagnosis. It capabilities can be extended by creating a disease monitoring module that allows users to create diagnosis for a particular disease provided that a valid data set is available and the system is adjusted accordingly. Additionally, for future works, it is recommended that a user is able to load and/or save neural network and genetic algorithm settings; this may also be applied for its weights. At present, the network must be completely trained. Regarding the data sets, it is advised to secure a greater number of samples to further enhance the generalization ability of the genetic neural network. Certainly, data gathering must remain confidential and private to protect all the parties involved. Moreover, in order to fully maximize the tuning capabilities of the neural network, it is recommended that a module be designed to fine tune each layer of the neural network. This may help create better and more optimized weights, and may save time used for training the neural network. Also, the genetic algorithm can be used to further optimize the other parameters. As evidenced in other studies, genetic algorithm can be used to filter and preprocess input variables and select the features that have significant bearing to the result, optimize layer and node count, preprocessing inputs, and may also be applied to the evolve network architecture.

IX. Bibliography

[1] Philippine Dengue. Information Bulletin.
<http://www.ifrc.org/docs/appeals/rpts10/PHep29071001.pdf>

[2] Dengue and Hemorrhagic Fever.
<http://www.who.int/mediacentre/factsheets/fs117/en/>

[3] Richard Kemp, “An introduction to genetic algorithms for neural networks.”

[4] K. Hornik, M. Stinchcombe, and H. White, "Multilayerfeedforward networks are universal approximators." Neural Networks, Vol. 2, pp. 359-366, 1989.

[5] R. Dybowski and V. Gant, Clinical Applications of Artificial Neural Networks. Cambridge University Press, 2007.

[6] Rüdiger W. Brause, “Medical Analysis and Diagnosis by Neural Networks.” 2001.

[7] Wesley Kerr and Suranga Hettarachchi, “Evolution of Neural Networks Using Genetic Algorithms.” University of Wyoming, 2003.

[8] J.J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities.” Proceedings of the National Academy of Sciences, pp. 2554-2558, 1982.

[9] T. Kohonen, “Self-organised formation of topologically correct featuremaps.” Biological Cybernetics, Vol 43, pp. 59-69, 1982.

[10] D. Rummelhart, and J. L. McLelland, “Parallel distributed processing: Explorations in the microstructure of Cognition.” Vol. I. Foundation, MIT Press, 1987.

[11] Fadzil Ahmad, Nor Ashidi Mat-Isa, Zakaria Hussain, Rozan Doudville, and Muhammad Khusairi Osman, “Genetic Algorithm - Artificial Neural Network (GA-ANN) Hybrid Intelligence for Cancer Diagnosis.” Second International Conference on Computational Intelligence, 2010.

[12]. El-Solh, A. A., Hsiao, C.-B., Goodnough, S., Serghani, J., and Grant, B. J. B., ”Predicting active pulmonary tuberculosis using an artificial neural network.” Chest, Vol. 116, pp. 968–973, 2001.

[13] Er, O., Temurtas, F., and Tanrikulu, A. Ç., “Tuberculosis disease diagnosis using artificial neural networks.” J. Med. Syst., 2008.

[14] O. Er, N. Yumusak and F. Temurtas, "Chest disease diagnosis using artificial neural networks." Expert Systems with Applications, Vol.37, No.12, pp.7648-7655. 2010.

[15] M. P. Francisco, G. C. Juan Manuel, S. P. Antonio and R.F. Daniel, "A robust model of the neuronal regulator of the lower urinary tract based on artificial neural networks." Neurocomputing, Vol.71, No.4-6, pp. 743-754. 2008.

[16] E. Elveren and N. Yumusak, “Tuberculosis Disease Diagnosis with the use of Neural Networks trained with Genetic Algorithm.” J Med System, 2011.

[17] F. Ibrahim, M.N. Taib, W.A.B.W. Abas, C.C. Guan, and S. Sulaiman, "A novel approach to classify risk in dengue hemorrhagic fever (DHF) using bioelectrical impedance analysis (BIA)." IEEE Journal, pp.237-244, 2005.

[18] Ibrahim F, Faisal T, Salim MI, Taib MN, "Non-invasive diagnosis of risk in dengue patients using bioelectrical impedance analysis and artificial neural network." Medical, Biological Engineering, and Computing, pp. 1141-1148, 2010.

[19] Hani Aburas, B. Cetiner, and Murat Sari, "Dengue confirmed-cases prediction: A neural network model." Expert Systems with Applications: An International Journal archive, Volume 37 ssue 6, June, 2010.

[20] Salim, Naomie and Husin, Nor Azura, "A comparative study for back propagation neural network and nonlinear regression models for predicting dengue outbreak." Jurnal Teknologi Maklumat, pp. 97-112, 2008.

[21] F.Ibrahim, M.N.Taib, S.Sulaiman, W.A.B.W.Abas, "Dengue fever (DF) and dengue hemorrhagic fever (DHF) symptoms analysis from an expert system perspective," IEEE Journal, pp. 212-215, 2001.

[22] Jatinder N.D. Guptaa and Randall S. Sexton, "Comparing Backpropagation with a Genetic Algorithm for Neural Network Training." Elsevier Science Ltd, 1999.

[23] Randall S. Sexton and Robert E. Dorsey, "Reliable classification using neural networks: a genetic algorithm and backpropagation comparison." Elsevier Science B.V., pp. 11-22, 2000.

[24] Activation Function.
<http://www.faqs.org/faqs/ai-faq/neural-nets/part2/section-10.html>

[25] Charles Darwin, The Origin of Species by Means of Natural Selection. 1859.

[26] Genetic Algorithms.
< http://www.obitko.com/tutorials/genetic-algorithms/>

[27] Garima Singh, Laxmi Srivastava.“ Genetic Algorithm-Based Artificial Neural Network for Voltage Stability Assessment.” Hindawi Publishing Corporation, 2011.

[28] David J. Montana and Lawrence Davis. Training Feedforward Neural Networks Using Genetic Algorithms.

[29] David J. Montana. “Neural Network Weight Selection Using Genetic Algorithms.” Bolt Beranek and Newman Inc.

[30] Dengue Fever Symptoms.
<http://www.denguefeversymptoms.org/>

[31] Dengue Fever.
<http://denguefever.co.in/>

[32] D. Shanthi, G. Sahoo, and N. Saravanan,”Evolving Connection Weights of Artificial Neural Networks Using Genetic Algorithms with Application to the Prediction of Stroke Disease.” Medwell Journals, 2009.

[33] Clinical Decision Support System.
<http://www.govhealthit.com>

[34] Clinical Decision Support System.
<http://www.searchhealthit.techtarget.com>

X. Appendix

A. Source Codes

---

allSet.java

```
import java.util.Random;

/**
* The allSet Class contains all the data set, whether
training, validating, or diagnosis.
* The dataset is read from a file.
*
* @author Mikyle Laurence O. Palomo AllSet Class
*/
public class allSet {

private set[] patterns; // whole pattern set including all
patterns
private set[] trainPatterns; // patterns to be used during
training
private set[] valPatterns; // patterns to be used during
validation
private set[] toDiagPatterns; //patterns to diagnose
private set[] trainPatternsCopy; // patterns to be used
during training
private set[] valPatternsCopy; // patterns to be used
during  validation
private set[] toDiagPatternsCopy; //patterns to diagnose
double mins[];
double maxs[];
private Random rand = new Random();
boolean init = false;
private static int setLen;
private static int diagnosisSetLen;
private static int validationSetLen;

/**
* No-argument constructor
*/
public allSet() {
}

/**
* Set diagnosis patterns from file
* @param sourceFile Source file
* @param noofinputs Number of input nodes
* @param nooftargets Number of target nodes
* @param isBatch Flag for batch diagnosis
*

*/
public void setDiagPatterns(String sourceFile, int
noofinputs, int nooftargets, boolean isBatch) {
LineReader linereader = new LineReader();
int counter = 0;
int invalid = 0;
double temp_double = 0;
//if batch, read from file
System.out.println("isBatch:" + isBatch);
if (isBatch) {
linereader = new LineReader(sourceFile);
while (linereader.NextLineSplitted()) {
try {
// count numeric data only
for (int i = 0; i < (noofinputs); i++) {
temp_double =
Double.parseDouble(linereader.column[i]);
}
counter++;
} catch (NumberFormatException e) {
System.err.println("Invalid Entry Removed!");
System.err.println("Entries Removed: " + ++invalid);
} catch (ArrayIndexOutOfBoundsException e) {
System.err.println("Insufficient Values. Entry
Removed!");
System.err.println("Entries Removed: " + ++invalid);
} catch (Exception e) {
System.err.println("Something else wrong!");
System.err.println("Entries Removed: " + ++invalid);
}
}
} else {
//single, read from input
//if single, counter = 1
counter = 1;

}
linereader = null;
patterns = new set[counter];
setDiagnosisSetLen(counter);
System.out.println("Number of diagnosis data samples: "
+ getDiagnosisSetLen());

// then read each pattern and place it into the array
double[] temp_in = new double[noofinputs];
//double[] temp_tar = new double[nooftargets];
//if single
counter = 0;
if (isBatch) {
linereader = new LineReader(sourceFile);
while (linereader.NextLineSplitted()) {
try { // count numeric data only
for (int i = 0; i < noofinputs; i++) {
temp_in[i] = Double.parseDouble(linereader.column[i]);
}
/*
```

```
* for (int i = noofinputs; i < noofinputs + nooftargets;
* i++) { temp_tar[i - noofinputs] =
* Double.parseDouble(linereader.column[i]); }
*/
patterns[counter++] = new set(temp_in);
} catch (NumberFormatException e) {
System.err.println("Not a number!");
} catch (Exception e) {
System.err.println("Error!");
}
}
} else {
try { // count numeric data only
String[] temp = sourceFile.split(";");
for (int i = 0; i < noofinputs; i++) {
temp_in[i] = Double.parseDouble(temp[i]);
}
patterns[counter++] = new set(temp_in);
} catch (NumberFormatException e) {
System.err.println("Not a number!");
} catch (Exception e) {
System.err.println("Error!");

}

}
linereader = null;
this.toDiagPatterns = patterns;
this.toDiagPatternsCopy = patterns;
System.out.println("Number of diagnosis data samples
after transfer: " + this.toDiagPatterns.length);

}

/**
* Set training patterns from file
* @param sourceFile Source file
* @param noofinputs Number of node in the input layer
* @param nooftargets Number of nodes in the target
layer
* @param ratioTraining Ratio for Training
* @param ratioValidating Ration for validating
*
*/
public void setTrainValPatterns(String sourceFile, int
noofinputs, int nooftargets, double ratioTraining, double
ratioValidating) {

// load patterns from the file
// 1st determine how many patterns there are
LineReader linereader = new LineReader(sourceFile);
int counter = 0;
int invalid = 0;
double temp_double = 0;
while (linereader.NextLineSplitted()) {
try {
// count numeric data only
for (int i = 0; i < (noofinputs + nooftargets); i++) {
temp_double =
Double.parseDouble(linereader.column[i]);

}
counter++;
} catch (NumberFormatException e) {
System.err.println("Invalid Entry Removed!");
System.err.println("Entries Removed: " + invalid++);
} catch (ArrayIndexOutOfBoundsException e) {
System.err.println("Insufficient Values. Entry
Removed!");
System.err.println("Entries Removed: " + invalid++);
}

}
linereader = null;
patterns = new set[counter];
setSetLen(counter);

// then read each pattern and place it into the array
double[] temp_in = new double[noofinputs];
double[] temp_tar = new double[nooftargets];
linereader = new LineReader(sourceFile);
counter = 0;
while (linereader.NextLineSplitted()) {
try { // count numeric data only
for (int i = 0; i < noofinputs; i++) {
temp_in[i] = Double.parseDouble(linereader.column[i]);
}
for (int i = noofinputs; i < noofinputs + nooftargets; i++) {
temp_tar[i - noofinputs] =
Double.parseDouble(linereader.column[i]);
}
patterns[counter++] = new set(temp_in, temp_tar);
} catch (NumberFormatException e) {
System.err.println("Invalid data format!");
} catch (ArrayIndexOutOfBoundsException e) {
System.err.println("Exceed column length!");
}
}
linereader = null;
System.out.println("Number of data samples: " +
getSetLen());

// now determine the no. of training, cross val. and test
patterns
this.trainPatterns = new set[(int)
(Math.ceil(patterns.length * ratioTraining))];
this.valPatterns = new set[(int)
(Math.floor(patterns.length * ratioValidating))];
this.trainPatternsCopy = new set[(int)
(Math.ceil(patterns.length * ratioTraining))];
this.valPatternsCopy = new set[(int)
(Math.floor(patterns.length * ratioValidating))];

int patterntoselect;
int patternsnotselected = patterns.length;
// first create training patterns
```

```
System.out.println("Generate Training Set of length: " +
Math.ceil(patterns.length * ratioTraining));
for (int i = 0; i < trainPatterns.length; i++) {
patterntoselect = rand.nextInt(patternsnotselected);
counter = 0;
for (int j = 0; j < patterns.length; j++) {

if (!patterns[j].selected) {
if (counter == patterntoselect) {
trainPatterns[i] = patterns[j];
trainPatternsCopy[i] = new set(patterns[j].input,
patterns[j].target);

patterns[j].selected = true;
patternsnotselected--;
break;
}
counter++;
}
}
}
// then create cross validation patterns
System.out.println("Generate Validation Set of length: "
+ (Math.floor(patterns.length * ratioValidating)));
setValidationSetLen(valPatterns.length);
for (int i = 0; i < valPatterns.length; i++) {
patterntoselect = rand.nextInt(patternsnotselected);
counter = 0;
for (int j = 0; j < patterns.length; j++) {
if (!patterns[j].selected) {
if (counter == patterntoselect) {
valPatterns[i] = patterns[j];
valPatternsCopy[i] = new set(patterns[j].input,
patterns[j].target);
patterns[j].selected = true;
patternsnotselected--;
break;
}
counter++;
}
}
}

System.out.println("Clearing Selected Statuses");
// and now switch all patterns as !selected, so that they
are ready for training
for (int i = 0; i < patterns.length; i++) {
patterns[i].selected = false;
}

System.out.println("Original Training set: " +
this.trainPatterns.length + " entries");
System.out.println("Original Validating set: " +
this.valPatterns.length + " entries");
}

/**
*
* @return  Diagnosis set length
*/
public static int getDiagnosisSetLen() {
return diagnosisSetLen;
}

/**
*
* @param dsl Diagnosis set length
*/
public static void setDiagnosisSetLen(int dsl) {
diagnosisSetLen = dsl;
}

/**
*
* @return Validation set length
*/
public static int getValidationSetLen() {
return validationSetLen;
}

/**
*
* @param vsl Validation set length
*/
public static void setValidationSetLen(int vsl) {
allSet.validationSetLen = vsl;
}

/**
*
* @return All set length
*/
public static int getSetLen() {
return setLen;
}

/**
*
* @param sl  All set length
*/
public static void setSetLen(int sl) {
setLen = sl;
}

/**
*
* @return  Diagnosis patterns
*/
public set[] getToDiagPatterns() {
return toDiagPatterns;
}

/**
*
* @param toDiagPatterns Diagnosis patterns
*/
```

```
public void setToDiagPatterns(set[] toDiagPatterns) {
this.toDiagPatterns = toDiagPatterns;
}

/**
*
* @return  Diagnosis patterns
*/
public set[] getTrainPatterns() {
return trainPatterns;
}

/**
*
* @param trainPatterns Training patterns
*/
public void setTrainPatterns(set[] trainPatterns) {
this.trainPatterns = trainPatterns;
}

/**
*
* @return Validation patterns
*/
public set[] getValPatterns() {
return valPatterns;
}

/**
*
* @param valPatterns Validation patterns
*/
public void setValPatterns(set[] valPatterns) {
this.valPatterns = valPatterns;
}

/**
*
* @return Diagnosis patterns copy
*/
public set[] getToDiagPatternsCopy() {
return toDiagPatternsCopy;
}

/**
*
* @param toDiagPatternsCopy Diagnosis patterns copy
*/
public void setToDiagPatternsCopy(set[]
toDiagPatternsCopy) {
this.toDiagPatternsCopy = toDiagPatternsCopy;
}

/**
*
* @return Training patterns copy
*/
public set[] getTrainPatternsCopy() {
return trainPatternsCopy;
}

/**
*
* @param trainPatternsCopy patterns copy
*/
public void setTrainPatternsCopy(set[]
trainPatternsCopy) {
this.trainPatternsCopy = trainPatternsCopy;
}

/**
*
* @return Validation patterns copy
*/
public set[] getValPatternsCopy() {
return valPatternsCopy;
}

/**
*
* @param valPatternsCopy Set validation patterns copy
*/
public void setValPatternsCopy(set[] valPatternsCopy) {
this.valPatternsCopy = valPatternsCopy;
}
}
```

Connection.java

```
/**
 *
 *The Connection class contains the weight of the
connection, preceding neuron,
 *the succeeding neuron, and the neuron id
 *
 *@author Mikyle Laurence O. Palomo
 *
 *
 */

public class Connection {

   double weight = 0;
   /**
    *Preceding neuron. Neuron to the left of the current
neuron.
    */
   public Neuron leftNeuron;
   /**
    *Succeeding Neuron. Neuron to the right of the current
neuron.
    */
   public Neuron rightNeuron;
```

```
/**
 *Current number of neurons
 */
public static int counter = 0;
/**
 *Current neuron id
 */
public int id;

/**
 * Creates a new connection
 * @param fromN From neuron
 * @param toN To neuron
 */
public Connection(Neuron fromN, Neuron toN) {
   leftNeuron = fromN;
   rightNeuron = toN;
   id = counter;
   counter++;
}

/**
 * @return current neuron weight
 *
 */
public double getWeight() {
   return weight;
}

/**
 *
 * @param w Current neuron weight
 */
public void setWeight(double w) {
   weight = w;
}

/**
 * @return Preceeding neuron
 */
public Neuron getFromNeuron() {
   return leftNeuron;
}

/**
 * @param fromNeuron Preceeding neuron
 */
public void setFromNeuron(Neuron fromNeuron) {
   this.leftNeuron = fromNeuron;
}

/**
 * @return Succeeding neuron
 */
public Neuron getToNeuron() {
   return rightNeuron;
}

   /**
    * @param toNeuron Succeeding neuron
    */
   public void setToNeuron(Neuron toNeuron) {
      this.rightNeuron = toNeuron;
   }
}
```

Hybrid.java

```
import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.LinkedList;
import java.util.Random;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.ValueAxis;
import org.jfree.chart.plot.XYPlot;
import org.jfree.data.time.Millisecond;
import org.jfree.data.time.TimeSeries;
import org.jfree.data.time.TimeSeriesCollection;

/**
 *
 * Hybrid is the main class of the application, contains the
GANN, hybrid
 * Genetic Neural Network.
 *
 * @author Mikyle Laurence O. Palomo
 *
 */
public class Hybrid {

   //population of encoded solutions
   LinkedList<NN> population = new LinkedList<NN>();
   //genetic parameters
   static int populationSize = 100;
   static int epochs = 1000;
   static double mutate = 0.01;
   //nn parameters
   static double maxError = 0.001;
   static int numOfHidden = 3;
   static int numOfInput = 8;
   private static int layers[] = {numOfInput, numOfHidden,
1}; // input, hidden, output
   final static int INPUT = 0;
   final int HIDDEN = 1;
```

```
   final static int OUTPUT = 2;
   static double SSE = 10;
   static boolean breakIfTolerable;
   private boolean reachedMaxErr = false;
   //dataset variables
   static double ratioTraining = 1;
   static double ratioValidating = 1;
   static allSet currSet;
   private static SetNormalizer normed;
   private static String fileName;
   private static String toDiagFileName;
   private static String toSingleDiagnose;
   //graph variables
   static TimeSeries ts;
   int finalCount;
   double finalSSE;
   //diagnosis variables
   private String[] symptomsList = {"Temp.", "P. Count", "C.
Rate", "BP(SP)", "BP(DP)", "R. Rate", "WBC", "RBC"};
   private int diagCount = 0;
   private static boolean training;
   private static boolean batchDiagnosing = false;
   private static boolean singleDiagnosing = false;
   //others
   static Random rand = new Random();
   static UI currUI;
   //optimized nn
   private static NN optimizedNN;

   /**
    * Creates a random initial population
    *
    */
   public Hybrid() {
      for (int i = 0; i < populationSize; i++) {
         NN c = new NN();
         population.add(c);

      }
   }

   private static class graphWorkerFrame extends JFrame {

      private JButton startProcessingBtn = new
JButton("Start");
      private JToggleButton pauseProcessingBtn = new
JToggleButton("Pause");
      private JButton endProcessingBtn = new
JButton("End");
      private boolean abortProcessing;
      private boolean pauseProcessing;
      private volatile boolean stop = false;

      public graphWorkerFrame() {
         System.out.println("GRAPH WORKER CREATED");

         initHybrid();
         genChart myGen = new genChart();
         new Thread(myGen).start();
         JPanel buttonPanel = new JPanel();

         buttonPanel.add(startProcessingBtn);
         buttonPanel.add(pauseProcessingBtn);
         buttonPanel.add(endProcessingBtn);

         startProcessingBtn.setEnabled(true);
         pauseProcessingBtn.setEnabled(false);
         endProcessingBtn.setEnabled(false);

         startProcessingBtn.addActionListener(new
ActionListener() {

            public void actionPerformed(ActionEvent e) {
               startProcessingActionPerformed();
            }

            private void startProcessingActionPerformed() {
               abortProcessing = false;
               pauseProcessing = false;

               new grapher().execute();
               startProcessingBtn.setEnabled(false);
               endProcessingBtn.setEnabled(true);
               pauseProcessingBtn.setEnabled(true);
               pauseProcessingBtn.setSelected(false);
            }
         });

         endProcessingBtn.addActionListener(new
ActionListener() {

            public void actionPerformed(ActionEvent e) {
               endProcessingActionPerformed();
            }

            private void endProcessingActionPerformed() {
               abortProcessing = true;

               startProcessingBtn.setEnabled(true);
               endProcessingBtn.setEnabled(false);
               pauseProcessingBtn.setEnabled(false);
               pauseProcessingBtn.setSelected(false);
            }
         });
         pauseProcessingBtn.addActionListener(new
ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
               pauseProcessingActionPerformed();
            }

            private void pauseProcessingActionPerformed() {
               pauseProcessing =
pauseProcessingBtn.isSelected();
            }
```

```
        });
        ts = new TimeSeries("SSE", Millisecond.class);
        TimeSeriesCollection dataset = new
TimeSeriesCollection(ts);
        JFreeChart chart =
ChartFactory.createTimeSeriesChart(
                "Training Graph",
                "Time",
                "SSE",
                dataset,
                true,
                true,
                false);

        final XYPlot plot = chart.getXYPlot();
        ValueAxis axis = plot.getDomainAxis();
        axis.setAutoRange(true);
        axis.setFixedAutoRange(30000.0);

this.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
        ChartPanel label = new ChartPanel(chart);
        JPanel p = new JPanel(new BorderLayout());
        p.add(label, BorderLayout.NORTH);
        p.add(buttonPanel, BorderLayout.SOUTH);
        this.getContentPane().add(p);
        this.setAlwaysOnTop(false);
        this.setResizable(false);

    }

    private class grapher extends SwingWorker<Boolean,
Double> {

        @Override
        protected Boolean doInBackground() throws
Exception {
            try {
                return backGroundTask();
            } catch (Exception e) {
                return false;
            }

        }

        @Override
        protected void done() {
            System.out.println("DONE!");

System.out.println("==========================");
        }
    }

    private void initHybrid() {
        System.out.println("INIT HYBRID");
        setPatterns(getFileName());
        set[] origTrain = currSet.getTrainPatterns();
        set[] origVal = currSet.getValPatterns();

        setNormed(new SetNormalizer(origTrain, origVal,
getLayers()[INPUT], getLayers()[OUTPUT]));
        getNormed().normalize();
    }

    public boolean backGroundTask() {
        long BEGIN = System.currentTimeMillis();

        Hybrid h = new Hybrid();
        int INDEX_MAX = epochs;
        System.out.println("Fitness Random Initialization");
        h.print();
        h.setIsTraining(true);
        NN nn = h.population.getFirst();
        for (int i = 0; i < INDEX_MAX; i++) {
            if (!h.reachedMaxErr) {
                h.setIsTraining(true);
                h.run(nn, i);
                long CURR = System.currentTimeMillis();

currUI.timeElapsedLabel.setText(formatIntoHHMMSS((CU
RR - BEGIN) / 1000));
            } else {
                INDEX_MAX = i;
                break;
            }
            try {
                Thread.sleep(40);
            } catch (InterruptedException e) {
            }

            if (abortProcessing) {
                INDEX_MAX = i;
                break;
            }

            while (pauseProcessing) {
                try {
                    Thread.sleep(50);
                    if (abortProcessing) {
                        return false;
                    }
                } catch (InterruptedException e) {
                }
            }
        }
        h.finalCount = INDEX_MAX;

        System.out.println("\nFitness Result after " +
INDEX_MAX + " evolutions");
        System.out.println("======Fittest Chromosome");
        nn = h.population.getFirst(); //get fittest
chromosome
        nn.printAllWeights();
        System.out.println("======SSE:" + 1 / nn.fitness());
        h.finalSSE = (1 / nn.fitness());
        System.out.println("Sum Squared Error= " +
h.finalSSE);
```

```
        System.out.println("SETTING OPTIMIZED NN");
        h.setOptimizedNN(nn);

        //disable retraining
        currUI.goButton.setEnabled(false);

        //enable fields for diagnosis
        currUI.batchEntryBox.setEnabled(true);
        currUI.diagBrowseButton.setEnabled(true);
        currUI.batchDiagButton.setEnabled(true);
        currUI.singleEntryDiagButton.setEnabled(true);
        currUI.singleEntryBox.setEnabled(true);

        //enable reset button
        currUI.resetButton.setEnabled(true);

        long END = System.currentTimeMillis();
        System.out.println("Time: " + (END - BEGIN) /
1000.0 + " sec.");

        //display clinical summary
        currUI.finalSSELabel.setText(h.finalSSE + "");
        currUI.timeElapsedLabel.setText((END - BEGIN) /
1000.0 + "sec.(" + formatIntoHHMMSS((END - BEGIN) /
1000) + ")");

currUI.timeElapsedLabel.setText(formatIntoHHMMSS((EN
D - BEGIN) / 1000));
        currUI.numGenLabel.setText(h.finalCount + " out of
" + h.epochs);

        //validate nn
        h.getOptimizedNN().printAllWeights();
        h.getOptimizedNN().validateNet();
        return true;

      }
    }

    private static String formatIntoHHMMSS(long secsIn) {
      int hours = (int) secsIn / 3600;
      int remainder = (int) secsIn % 3600;
      int minutes = remainder / 60;
      int seconds = remainder % 60;

      return ((hours < 10 ? "0" : "") + hours + ":" + (minutes
< 10 ? "0" : "") + minutes + ":" + (seconds < 10 ? "0" : "") +
seconds);

    }

    void run(NN nn, int count) {
      count++;
      nn = population.getFirst();
      System.out.println("Generation " + count + " SSE: " +
(1 / nn.fitness()));
      SSE = 1 / nn.fitness();
      if (count == 1) {
        UI.trainingResultTextArea.append("Starting
Training");
      }

      //update display
      UI.trainingResultTextArea.append("\nGeneration " +
count + ": " + SSE);
      UI.numGenLabel.setText(count + " out of " +
this.epochs);
      UI.finalSSELabel.setText(SSE + "");

      if (SSE <= maxError && breakIfTolerable) {
        UI.trainingResultTextArea.append("\nTolerable
error: " + SSE + " Reached!");
        reachedMaxErr = true;
      } else {
        produceNextGen();
      }

      //scroll down the text area

UI.trainingResultTextArea.setCaretPosition(UI.trainingResu
ltTextArea.getDocument().getLength());

    }

    void print() {
      System.out.println("-- print");
      for (NN c : population) {
        System.out.println(c);
      }
    }

    /**
     * Selection strategy: Tournament method Replacement
strategy: steady state
     * Find 4 random in population not same let 2 fight, and
2 fight the winners
     * makes 2 children
     */
    void produceNextGen() {
      LinkedList<NN> newpopulation = new
LinkedList<NN>();
      while (newpopulation.size() < populationSize) {
        int size = population.size();
        int i = rand.nextInt(size);
        int j, k, l;
        j = k = l = i;
        //select unique parents
        while (j == i) {
          j = rand.nextInt(size);
        }
        while (k == i || k == j) {
          k = rand.nextInt(size);
        }
        while (l == i || l == j || k == l) {
          l = rand.nextInt(size);
        }
```

```
        //candidate genes
        NN c1 = population.get(i);
        NN c2 = population.get(j);
        NN c3 = population.get(k);
        NN c4 = population.get(l);

        double f1 = c1.fitness();
        double f2 = c2.fitness();
        double f3 = c3.fitness();
        double f4 = c4.fitness();

        //winner genes
        NN w1, w2;

        if (f1 > f2) {
           w1 = c1;
        } else {
           w1 = c2;
        }
        if (f3 > f4) {
           w2 = c3;
        } else {
           w2 = c4;
        }

        NN child1, child2;

        // Method uniform crossover
        NN[] childs = newChilds(w1, w2);
        child1 = childs[0];
        child2 = childs[1];

        double mutatePercent = mutate;

        boolean m1 = rand.nextDouble() <= mutatePercent;
        boolean m2 = rand.nextDouble() <= mutatePercent;

        if (m1) {
           mutate(child1);
        }
        if (m2) {
           mutate(child2);
        }

        //is child better?
        boolean isChild1Good = child1.fitness() >=
w1.fitness();
        boolean isChild2Good = child2.fitness() >=
w2.fitness();

        newpopulation.add(isChild1Good ? child1 : w1);
        newpopulation.add(isChild2Good ? child2 : w2);
     }

     population = newpopulation;
  }

  void mutate(NN c) {
     int i = rand.nextInt(NN.SIZE);
     double range = getRandom() * 2; // range [-1;1[ * 2
     c.chromosome[i] = c.chromosome[i] + range;
     c.setChromosome(c.chromosome); // update
  }

  double getRandom() {
     return (rand.nextDouble() * 2) - 1; // [-1;1[
  }

  // Uniform crossover
  static NN[] newChilds(NN c1, NN c2) {
     NN child1 = new NN();
     NN child2 = new NN();

     double[] chromosome1 = new double[NN.SIZE];
     double[] chromosome2 = new double[NN.SIZE];

     for (int i = 0; i < NN.SIZE; i++) {
        boolean b = rand.nextDouble() >= 0.5;
        if (b) {
           chromosome1[i] = c1.chromosome[i];
           chromosome2[i] = c2.chromosome[i];
        } else {
           chromosome1[i] = c2.chromosome[i];
           chromosome2[i] = c1.chromosome[i];
        }
     }
     child1.setChromosome(chromosome1);
     child2.setChromosome(chromosome2);

     return new NN[]{child1, child2};
  }

  static void setPatterns(String filename) {
     if (!batchDiagnosing && !singleDiagnosing && null ==
currSet) {
        System.out.println("SETTING TRAIN VAL PATTERNS
FROM FILE");
        System.out.println("no currSet found");
        currSet = new allSet();
        currSet.setTrainValPatterns(filename,
getLayers()[INPUT], getLayers()[OUTPUT], ratioTraining,
ratioValidating);
        System.out.println("File init ok");
     } else if (batchDiagnosing && null != currSet) {
        System.out.println("SETTING BATCH DIAG
PATTERNS FROM FILE");
        System.out.println("currSet found");
        getNormed().setNormalizedDiag(false);
        currSet.setDiagPatterns(filename,
getLayers()[INPUT], getLayers()[OUTPUT], true);
     } else if (singleDiagnosing && null != currSet) {
        System.out.println("SETTING SINGLE DIAG
PATTERNS FROM INPUT");
        System.out.println("currSet found");
        //overload set diag/add param and if-clause in
method
```

```
        getNormed().setNormalizedDiag(false);
        currSet.setDiagPatterns(filename,
getLayers()[INPUT], getLayers()[OUTPUT], false);
      }
   }

   /**
    * Create and show chart object
    */
   protected static void createAndShowChart() {
      JFrame frame = new graphWorkerFrame();
      currUI.goButton.setEnabled(false);
      frame.pack();
      frame.setLocationRelativeTo(null);
      frame.setVisible(true);
   }

   /**
    * Initialize batch diagnosis
    */
   protected static void initBatchDiagnosis() {
      System.out.print("INIT BATCH DIAG");
      batchDiagnosing = true;
      setPatterns(getToDiagFileName());
      set[] origDiag = currSet.getToDiagPatterns(); //keep
original
      normed.setDiagPatterns(origDiag);
      getNormed().normalizeDiag();
   }

   /**
    * Initialize single diagnosis
    */
   protected static void initSingleDiagnosis() {
      System.out.println("INIT SINGLE DIAG");
      singleDiagnosing = true;
      //get user input
      setPatterns(getToSingleDiagnose());
      set[] origDiag = currSet.getToDiagPatterns(); //keep
original
      normed.setDiagPatterns(origDiag);
      getNormed().normalizeDiag();
   }

   void showResultsOnTable() {
      set[] diagData = getNormed().getDiagPatterns();
      double[][] diagResults =
getOptimizedNN().getResultOutputsDiag();
      Object[][] toDisplay = new
Object[diagData.length][layers[INPUT] + layers[OUTPUT]];

      for (int y = 0; y < diagData.length; y++) {
         //add data
         for (int x = 0; x < layers[INPUT]; x++) {
            toDisplay[y][x] = diagData[y].input[x];
         }
         //add diagnosis result
         for (int z = 0; z < layers[OUTPUT]; z++) {
            //convert to +/-
            if (Math.round(diagResults[y][z]) == 1) {
               toDisplay[y][z + layers[INPUT]] = "POS";
            } else {
               toDisplay[y][z + layers[INPUT]] = "NEG";
            }
         }
      }

      for (Object[] d : toDisplay) {
         for (Object dd : d) {
            System.out.print(dd + " ");
         }
         System.out.println();
      }


      String[] colNames = new String[layers[INPUT] +
layers[OUTPUT]];
      for (int i = 0; i < layers[INPUT] + layers[OUTPUT]; i++) {
         if (i == 8) {
            colNames[i] = "Diagnosis";
         } else {
            colNames[i] = symptomsList[i];
         }

      }

      DefaultTableModel tableModel = (DefaultTableModel)
currUI.diagTable.getModel();

      String newFormat = "yyyy-MM-dd HH:mm:ss";
      SimpleDateFormat sdf = new
SimpleDateFormat(newFormat);

      String diagLabel = "Diagnosis " + ++this.diagCount + "(
" + sdf.format(new Date()) + " )";
      tableModel.addRow(new Object[]{diagLabel});

      for (Object[] o : toDisplay) {
         tableModel.addRow(o);
      }

      //put result on result table
      tableModel.addRow(new Object[toDisplay.length +
1]);
      tableModel.setColumnIdentifiers(colNames);
      tableModel.fireTableDataChanged();

      //clear diag arrays to ready for next diagnosis
      getNormed().setDiagPatterns(null);
      getNormed().setNormalizedDiagPatterns(null);
      getNormed().setNormalizedDiag(false);
      getOptimizedNN().setResultOutputsDiag(null);


   }

   static class genChart implements Runnable {
```

```
   private Random randGen = new Random();
   double currSSE = SSE;

   public void run() {
      while (true) {
         currSSE = SSE;
         if (null == ts) {
            ts = new TimeSeries("SSE", Millisecond.class);
         }

         ts.addOrUpdate(new Millisecond(), currSSE);
         try {
            Thread.sleep(20);
         } catch (InterruptedException ex) {
            System.out.println(ex);
         }
      }
   }
}

/**
 *
 * @return Number of epochs
 */
public int getEpochs() {
   return epochs;
}

/**
 *
 * @param e Number of epochs
 */
public void setEpochs(int e) {
   epochs = e;
}

/**
 *
 * @return Mutation rate
 */
public double getMutate() {
   return mutate;
}

/**
 *
 * @param m Mutation rate
 */
public void setMutate(double m) {
   mutate = m;
}

/**
 *
 * @return Maximum tolerable error
 */
public double getMaxError() {
   return maxError;
}

/**
 *
 * @param me Maximum error
 */
public void setMaxError(double me) {
   maxError = me;
}

/**
 *
 * @return Number of hidden nodes
 */
public int getNumOfHidden() {
   return numOfHidden;
}

/**
 *
 * @param nh Number of hidden nodes
 */
public void setNumOfHidden(int nh) {
   numOfHidden = nh;
}

/**
 *
 * @return Get number of inputs
 */
public int getNumOfInput() {
   return numOfInput;
}

/**
 *
 * @param ni Number of inputs
 */
public void setNumOfInput(int ni) {
   numOfInput = ni;
}

/**
 *
 * @return Population size
 */
public int getPopulationSize() {
   return populationSize;
}

/**
 *
 * @param ps Population size
 */
public void setPopulationSize(int ps) {
   populationSize = ps;
}
```

```
/**
 *
 * @return Training Ratio
 *
 */
public static double getRatioTraining() {
   return ratioTraining;
}

/**
 *
 * @param rt Training ratio
 *
 */
public static void setRatioTraining(double rt) {
   ratioTraining = rt;
}

/**
 *
 * @return ratioValidating Get validating ratio
 */
public static double getRatioValidating() {
   return ratioValidating;
}

/**
 *
 * @param rv Validating ratio
 */
public void setRatioValidating(double rv) {
   ratioValidating = rv;
}

/**
 * @return Set normalizer
 *
 */
public static SetNormalizer getNormed() {
   return normed;
}

/**
 * @param sn Set normalizer
 */
public static void setNormed(SetNormalizer sn) {
   normed = sn;
}

/**
 * @return Neural network layer
 */
public static int[] getLayers() {
   return layers;
}

/**
 * @param aLayers Neural network layer
 */
public static void setLayers(int[] aLayers) {
   layers = aLayers;
}

/**
 *
 * @return Diagnosis filename
 */
public static String getToDiagFileName() {
   return toDiagFileName;
}

/**
 *
 * @param toDiagFileName Diagnosis filename
 */
public static void setToDiagFileName(String
toDiagFileName) {
   Hybrid.toDiagFileName = toDiagFileName;
}

/**
 * @return Filename
 */
public static String getFileName() {
   return fileName;
}

/**
 *
 * @param name Filename
 */
public static void setFileName(String name) {
   fileName = name;
}

/**
 *
 * @return Flag for training stoppage
 */
public boolean isBreakIfTolerable() {
   return breakIfTolerable;
}

/**
 *
 * @param breakIfTolerable Flag for training stoppage
 */
public void setBreakIfTolerable(boolean
breakIfTolerable) {
   this.breakIfTolerable = breakIfTolerable;
}

/**
 *
 * @return Flag for training
```

```
     *
     */
    public static boolean isTraining() {
        return training;
    }

    /**
     *
     * @param isTraining Flag for training
     */
    public void setIsTraining(boolean isTraining) {
        this.training = isTraining;
    }

    /**
     *
     * @return Optimized neural network
     */
    public NN getOptimizedNN() {
        return optimizedNN;
    }

    /**
     *
     * @param optimizedNN Optimized neural network
     */
    public void setOptimizedNN(NN optimizedNN) {
        this.optimizedNN = optimizedNN;
    }

    /**
     *
     * @return String for single diagnosis
     *
     */
    public static String getToSingleDiagnose() {
        return toSingleDiagnose;
    }

    /**
     *
     * @param toSingleDiagnose Set string for single
diagnosis
     */
    public static void setToSingleDiagnose(String
toSingleDiagnose) {
        Hybrid.toSingleDiagnose = toSingleDiagnose;
    }

    /**
     *
     * @return batchDiagnosing Return flag for batch
diagnosis
     *
     */
    public static boolean isBatchDiagnosing() {
        return batchDiagnosing;
    }

    /**
     *
     * @param batchDiagnosing Set flag for batch diagnosis
     *
     */
    public static void setBatchDiagnosing(boolean
batchDiagnosing) {
        Hybrid.batchDiagnosing = batchDiagnosing;
    }

    /**
     *
     * @return singleDiagnosing Get flag for single diagnosis
     */
    public static boolean isSingleDiagnosing() {
        return singleDiagnosing;
    }

    /**
     *
     * @param singleDiagnosing Set flag for single diagnosis
     *
     */
    public static void setSingleDiagnosing(boolean
singleDiagnosing) {
        Hybrid.singleDiagnosing = singleDiagnosing;
    }

    /**
     *
     * @param args Starts the application
     *
     */
    public static void main(String[] args) {
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassNam
e());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(UI.class.getName()).log(j
ava.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(UI.class.getName()).log(j
ava.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(UI.class.getName()).log(j
ava.util.logging.Level.SEVERE, null, ex);
```

```
      } catch
(javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(UI.class.getName()).log(j
ava.util.logging.Level.SEVERE, null, ex);

      }
      System.out.println("GNAT START");
      //display splash screen
      SplashScreen splash = new SplashScreen(10000);
      java.awt.EventQueue.invokeLater(new Runnable() {

         public void run() {
            currUI = new UI();
            int width = currUI.getWidth();
            int height = currUI.getHeight();
            Dimension screen =
Toolkit.getDefaultToolkit().getScreenSize();
            int x = (screen.width -  width) / 2;
            int y = (screen.height - height) / 2;

            currUI.setBounds(x, y, width, height);
            currUI.setVisible(true);

         }
      });
   }
}
```

LineReader.java

```
import java.io.BufferedInputStream;
import java.io.FileInputStream;
import java.io.IOException;

/**
 *
 * This LineReader class reads one line at a time and
 * returns it as a string.
 *
 * @author Mikyle Laurence O. Palomo
 */
public class LineReader {

   String[] column;
   FileInputStream fis;
   BufferedInputStream bis;

   /**
    * No-argument Constructor
    */
   public LineReader() {
   }

   /**
    *
    * @param path Constructor. Reads contents from
specified path
    */
   public LineReader(String path) {
      try {
         fis = new FileInputStream(path);
         bis = new BufferedInputStream(fis);
      } catch (IOException e) {
         System.err.print("File Error!");
      }

   }

   /**
    *
    * @return If next line exists Read the next line, split it,
and return the
    * parts as a string array
    */
   public boolean NextLineSplitted() {
      column = null;
      column = NextLine().split(";");
      if (column[0] != "#EOF#") {
         for (int i = 0; i < column.length; i++) {
            column[i] = column[i].trim();
         }
         return true;
      } else {
         return false;
      }
   }

   /**
    *
    * @return Next line Read the next line, return the line
as string
    */
   public String NextLine() {
      int i;
      char[] temp_array = new char[50000];
      char[] temp_array2;
      boolean last_line;
      int counter;
      String temp_line = "";

      do {
         temp_array2 = null;
         counter = 0;
         last_line = true;
         // read a line
         try {
            while ((i = bis.read()) != -1) {
               last_line = false;
               if (i == 13 || i == 10) { //see a space newline
                  break;
               } else if (i != 10 && i != 13) {
                  temp_array[counter++] = (char) i;
```

```
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        } catch (NullPointerException e) {
            e.printStackTrace();
        }
        // put the array into a string
        if (last_line) {
            temp_line = "#EOF#";
        } else if (counter != 0) {
            temp_array2 = new char[counter];
            boolean all_spaces = true;
            for (int j = 0; j < counter; j++) {
                if (temp_array[j] != ' ') {
                    all_spaces = false;
                }
                temp_array2[j] = temp_array[j];
            }
            if (all_spaces) {
                temp_line = "";
            } else {
                temp_line = new String(temp_array2);
            }
            if (temp_line.length() >= 2 &&
temp_line.charAt(0) == '/' && temp_line.charAt(1) == '/') {
                temp_line = "";
            }
        } else {
            temp_line = "";
        }

    } while (temp_line == "");
    return temp_line.trim();
  }

  /**
   *
   * @return Buffered input stream
   */
  public BufferedInputStream getBis() {
    return bis;
  }

  /**
   *
   * @param bis Buffered input stream
   */
  public void setBis(BufferedInputStream bis) {
    this.bis = bis;
  }

  /**
   *
   * @return Column array
   */
  public String[] getColumn() {
    return column;
  }

  /**
   *
   * @param column Column array
   */
  public void setColumn(String[] column) {
    this.column = column;
  }

  /**
   *
   * @return File input stream
   */
  public FileInputStream getFis() {
    return fis;
  }

  /**
   *
   * @param fis File input stream
   */
  public void setFis(FileInputStream fis) {
    this.fis = fis;
  }
}
```

Neuron.java

```
import java.util.ArrayList;
import java.util.HashMap;

/**
 * Neuron object. Neurons make up the nodes of each
layers
 * @author Mikyle Laurence O. Palomo Neuron class
 */
public class Neuron {

  /**
   * Neuron count
   */
  public static int counter = 0;
  /**
   * Neuron id
   */
  public int id;
  Connection biasConnection;
  final double bias = -1;
  double output;
  ArrayList<Connection> Inconnections = new
ArrayList<Connection>();
  HashMap<Integer, Connection> connectionLookup =
new HashMap<Integer, Connection>();

  /**
```

```
    * Default constructor
    */
   public Neuron() {
      id = counter;
      counter++;
   }

   /**
    * Compute Sj = Wij*Aij + w0j*bias Apply activation
function to get output
    */
   public void calculateOutput() {
      double s = 0;
      for (Connection con : Inconnections) {
         Neuron leftNeuron = con.getFromNeuron();
         double weight = con.getWeight();
         double a = leftNeuron.getOutput(); //output from
previous layer

         s += (weight * a);
      }
      s += (biasConnection.getWeight() *  bias);
      output = g(s);
   }

   double g(double x) {
      return sigmoid(x);
   }

   double sigmoid(double x) {
      return 1.0 / (1.0 + (Math.exp(-x)));
   }

   /**
    *
    * @param inNeurons Add all in connections
    */
   public void addInConnectionsS(ArrayList<Neuron>
inNeurons) {
      for (Neuron n : inNeurons) {
         Connection con = new Connection(n, this);
         Inconnections.add(con);
         connectionLookup.put(n.getId(), con);
      }
   }

   /**
    *
    * @param neuronIndex Index of current neuron
    * @return  All connection of specified neuron
    */
   public Connection getConnection(int neuronIndex) {
      return connectionLookup.get(neuronIndex);
   }

   /**
    *
    * @param con Add an in-connection
    */
   public void addInConnection(Connection con) {
      Inconnections.add(con);
   }

   /**
    *
    * @param n Add a bias connection
    */
   public void addBiasConnection(Neuron n) {
      Connection con = new Connection(n, this);
      biasConnection = con;
      Inconnections.add(con);
   }

   /**
    *
    * @return All in-connections
    */
   public ArrayList<Connection> getAllInConnections() {
      return Inconnections;
   }

   /**
    *
    * @return  Neuron id
    */
   public int getId() {
      return id;
   }

   /**
    *
    * @param i  Neuron id
    */
   public void setId(int i) {
      id = i;
   }

   /**
    *
    * @return bias
    */
   public double getBias() {
      return bias;
   }

   /**
    *
    * @return  Output of a certain group of neurons
    */
   public double getOutput() {
      return output;
   }

   /**
    *
    * @param o  Output of a certain group of neurons
```

```
    */
   public void setOutput(double o) {
      output = o;
   }
}
```

NN.java

```
import java.util.ArrayList;
import java.util.Random;

/**
 *
 * This Neural Network Class contains the neural network
parameters.
 *
 *@author Mikyle Laurence O. Palomo
 *
 *
 */
public class NN {

   final static int randomWeightMultiplier = 30;
   final Neuron bias = new Neuron();
   final ArrayList<Neuron> inputLayer = new
ArrayList<Neuron>();
   final ArrayList<Neuron> hiddenLayer = new
ArrayList<Neuron>();
   final ArrayList<Neuron> outputLayer = new
ArrayList<Neuron>();
   final static int INPUT = 0;
   final static int HIDDEN = 1;
   final static int OUTPUT = 2;
   final static double ratioTraining =
Hybrid.getRatioTraining();
   final static double ratioValidating =
Hybrid.getRatioValidating();
   Random rand = new Random();
   private double inputsSingleDiag[][];
   private double inputsBatchDiag[][];
   static int layers[] = Hybrid.getLayers(); // input, hidden,
output
   final static int SIZE = (layers[INPUT] * layers[HIDDEN]) +
(layers[HIDDEN] * layers[OUTPUT]) + layers[HIDDEN] +
layers[OUTPUT]; // adjusts accdg num of neurons and
weights
   public double[] chromosome = new
double[SIZE];//(numIN*numHID + numHID*OUT) +
numHID + numOUT
   private boolean validating = false;
   private double resultOutputsTrain[][];
   private double resultOutputsVal[][];
   private double resultOutputsDiag[][];
   double output[];

   /**
    * Default Constructor
    */
   public NN() {
      // create all Neurons and connections
      // connections are created in the neuron class
      for (int i = 0; i < layers.length; i++) {
         if (i == INPUT) { // input layer
            for (int j = 0; j < layers[i]; j++) {
               Neuron neuron = new Neuron();
               inputLayer.add(neuron);
            }
         } else if (i == HIDDEN) { // hidden layer
            for (int j = 0; j < layers[i]; j++) {
               Neuron neuron = new Neuron();
               neuron.addInConnectionsS(inputLayer);
               neuron.addBiasConnection(bias);
               hiddenLayer.add(neuron);
            }
         } else if (i == OUTPUT) { // output layer
            for (int j = 0; j < layers[i]; j++) {
               Neuron neuron = new Neuron();
               neuron.addInConnectionsS(hiddenLayer);
               neuron.addBiasConnection(bias);
               outputLayer.add(neuron);
            }
         } else {
            System.out.println("Error NeuralNetwork init!");
         }
      }

      // initialize random weights
      int i = 0;
      for (Neuron neuron : hiddenLayer) {
         ArrayList<Connection> connections =
neuron.getAllInConnections();
         for (Connection conn : connections) {
            double newWeight = getRandom();
            conn.setWeight(newWeight);
            chromosome[i++] = newWeight;
         }
      }

      for (Neuron neuron : outputLayer) {
         ArrayList<Connection> connections =
neuron.getAllInConnections();
         for (Connection conn : connections) {
            double newWeight = getRandom();
            conn.setWeight(newWeight);
            chromosome[i++] = newWeight;
         }
      }
      // reset id counters
      Neuron.counter = 0;
      Connection.counter = 0;
   }

   // random
```

```
   // bipolar mapping
   double getRandom() {
      return randomWeightMultiplier * (rand.nextDouble()
* 2 - 1); // [-1;1[
   }

   void setInput(double inputs[]) {
      // There is equally many neurons in the input layer as
there are
      // input variables
      for (int i = 0; i < inputLayer.size(); i++) {
         inputLayer.get(i).setOutput(inputs[i]);
      }
   }

   double[] getOutput() {
      double[] outputs = new double[outputLayer.size()];
      for (int i = 0; i < outputLayer.size(); i++) {
         outputs[i] = outputLayer.get(i).getOutput();
      }
      return outputs;
   }

   /**
    * Calculate the output of the neural network based on
the input The forward
    * operation
    */
   void activate() {
      for (Neuron n : hiddenLayer) {
         n.calculateOutput();
      }

      for (Neuron n : outputLayer) {
         n.calculateOutput();
      }
   }

   double[] getAllWeights() {
      double[] ds = new double[SIZE];
      int i = 0;
      // weights for the hidden layer
      for (Neuron n : hiddenLayer) {
         ArrayList<Connection> connections =
n.getAllInConnections();
         for (Connection con : connections) {
            double w = con.getWeight();
            ds[i++] = w;
         }
      }
      // weights for the output layer
      for (Neuron n : outputLayer) {
         ArrayList<Connection> connections =
n.getAllInConnections();
         for (Connection con : connections) {
            double w = con.getWeight();
            ds[i++] = w;
         }
      }

      return ds;
   }

   /**
    *
    * @param weights Set chromosome with its
corresponding weight
    */
   public void setChromosome(double[] weights) {
      chromosome = weights;
      int i = 0;
      // weights for the hidden layer
      for (Neuron n : hiddenLayer) {
         ArrayList<Connection> connections =
n.getAllInConnections();
         for (Connection con : connections) {
            double w = weights[i];
            con.setWeight(w);
            i++;
         }
      }
      // weights for the output layer
      for (Neuron n : outputLayer) {
         ArrayList<Connection> connections =
n.getAllInConnections();
         for (Connection con : connections) {
            double w = weights[i];
            con.setWeight(w);
            i++;
         }
      }
   }

   /**
    * Displays current weights of all nodes in all layers
    */
   public void printAllWeights() {
      // weights for the hidden layer
      for (Neuron n : hiddenLayer) {
         ArrayList<Connection> connections =
n.getAllInConnections();
         for (Connection con : connections) {
            double w = con.getWeight();
            System.out.println("n=" + n.id + " c=" + con.id + "
w=" + w);
         }
      }
      // weights for the output layer
      for (Neuron n : outputLayer) {
         ArrayList<Connection> connections =
n.getAllInConnections();
         for (Connection con : connections) {
            double w = con.getWeight();
            System.out.println("n=" + n.id + " c=" + con.id + "
w=" + w);
         }
```

```
    }
  }

  /**
   * Obtain fitness of current neural network
   */
  double fitness() {
    double error = 0;
    int normPatternsLen =
Hybrid.getNormed().getNormalizedTrainPatterns().length;
    resultOutputsTrain = new double[(int)
Math.ceil(allSet.getSetLen() *
ratioTraining)][layers[OUTPUT]];
    for (int p = 0; p < normPatternsLen; p++) {

setInput(Hybrid.getNormed().getNormalizedTrainPatterns(
)[p].input);
      activate();
      output = getOutput();
      resultOutputsTrain[p] = output;
      //(actual - ideal)^2
      for (int j = 0; j <
Hybrid.getNormed().getNormalizedTrainPatterns()[p].targ
et.length; j++) {
        double err = Math.pow(resultOutputsTrain[p][j] -
Hybrid.getNormed().getNormalizedTrainPatterns()[p].targ
et[j], 2);
        error += err;

      }
    }
    return 1 / error; // error can not be 0
  }

  /**
   * Diagnose with this neural network
   */
  void diagnose() {
    resultOutputsDiag = new
double[allSet.getDiagnosisSetLen()][layers[OUTPUT]];
    System.out.println("Diagnosis:");
    for (int p = 0; p <
Hybrid.getNormed().getNormalizedDiagPatterns().length;
p++) {

setInput(Hybrid.getNormed().getNormalizedDiagPatterns()
[p].input);
      activate();
      output = getOutput();
      getResultOutputsDiag()[p] = output;
    }

    for (int p = 0; p <
Hybrid.getNormed().getNormalizedDiagPatterns().length;
p++) {
      System.out.print("INPUTS: ");
      for (int x = 0; x < layers[INPUT]; x++) {

System.out.print(Double.toString(Hybrid.getNormed().get
NormalizedDiagPatterns()[p].input[x]) + " ");

      }

      System.out.print("ACTUAL: ");
      for (int x = 0; x < layers[OUTPUT]; x++) {
        System.out.print(getResultOutputsDiag()[p][x] + "
");
      }

      System.out.println();
    }

    //setResultOutputsDiag(null);
    Hybrid.setBatchDiagnosing(false);
    Hybrid.setSingleDiagnosing(false);
  }

  /**
   * After training, the neural network is validated
   */
  void validateNet() {
    int correct = 0;
    resultOutputsVal = new
double[allSet.getValidationSetLen()][layers[OUTPUT]];
    System.out.println("VALIDATING NEURAL
NETWORK");
    UI.trainingResultTextArea.append("\n\nValidating
Neural Network\n");
    //feed val patterns
    for (int p = 0; p <
Hybrid.getNormed().getNormalizedValPatterns().length;
p++) {

setInput(Hybrid.getNormed().getNormalizedValPatterns()[
p].input);
      activate();
      output = getOutput();
      getResultOutputsVal()[p] = output;
    }
    for (int p = 0; p <
Hybrid.getNormed().getValPatterns().length; p++) {
      UI.trainingResultTextArea.append("INPUT " + (p + 1)
+ ": \n");
      for (int x = 0; x < layers[INPUT]; x++) {

UI.trainingResultTextArea.append(Double.toString(Hybrid.
getNormed().getValPatterns()[p].input[x]) + " ");
      }
      UI.trainingResultTextArea.append("\n");
      UI.trainingResultTextArea.append("Expected: ");
      String toDispEx = "";
      String toDispAc = "";

      for (int x = 0; x < layers[OUTPUT]; x++) {
```

```
            Double tempTar =
Hybrid.getNormed().getNormalizedValPatterns()[p].target[
x];
            //convert to +/-
            if (Math.round(tempTar) == 1) {
               toDispEx = "POS";
            } else {
               toDispEx = "NEG";
            }

            UI.trainingResultTextArea.append(toDispEx + " |
");
         }

         UI.trainingResultTextArea.append("Actual: ");
         for (int x = 0; x < layers[OUTPUT]; x++) {
            Double tempAc = getResultOutputsVal()[p][x];

            //convert to +/-
            if (Math.round(tempAc) == 1) {
               toDispAc = "POS";
            } else {
               toDispAc = "NEG";
            }
            UI.trainingResultTextArea.append(toDispAc);
         }

         if (toDispAc.equalsIgnoreCase(toDispEx)) {
            correct++;
         }

         UI.trainingResultTextArea.append("\n");
      }
      double len =
Hybrid.getNormed().getValPatterns().length;
      double rate = Math.round((correct / len) * 100);
      UI.trainingResultTextArea.append("\nValidation
Result: \n"
            + rate + "% (" + correct + " out of " +
Hybrid.getNormed().getValPatterns().length + ") properly
diagnosed.\n");

   }

   @Override
   public String toString() {
      return fitness() + "";
   }

   /**
    *
    * @return Inputs for batch diagnosis
    */
   public double[][] getInputsBatchDiag() {
      return inputsBatchDiag;
   }

   /**
    *
    * @param inputsBatchDiag Inputs for batch diagnosis
    */
   public void setInputsBatchDiag(double[][]
inputsBatchDiag) {
      this.inputsBatchDiag = inputsBatchDiag;
   }

   /**
    *
    * @return Inputs for single diagnosis
    */
   public double[][] getInputsSingleDiag() {
      return inputsSingleDiag;
   }

   /**
    *
    * @param inputsSingleDiag Inputs for single diagnosis
    */
   public void setInputsSingleDiag(double[][]
inputsSingleDiag) {
      this.inputsSingleDiag = inputsSingleDiag;
   }

   /**
    * @return  Results for training
    */
   public double[][] getResultOutputsTrain() {
      return resultOutputsTrain;
   }

   /**
    * @param resultOutputsTrain Results for training
    */
   public void setResultOutputsTrain(double[][]
resultOutputsTrain) {
      this.resultOutputsTrain = resultOutputsTrain;
   }

   /**
    * @return Results for validation
    */
   public double[][] getResultOutputsVal() {
      return resultOutputsVal;
   }

   /**
    * @param resultOutputsVal Results for validation
    */
   public void setResultOutputsVal(double[][]
resultOutputsVal) {
      this.resultOutputsVal = resultOutputsVal;
   }

   /**
    * @return Results for diagnosis
    */
```

```
   public double[][] getResultOutputsDiag() {
      return resultOutputsDiag;
   }

   /**
    * @param resultOutputsDiag  Results for diagnosis
    */
   public void setResultOutputsDiag(double[][]
resultOutputsDiag) {
      this.resultOutputsDiag = resultOutputsDiag;
   }
}
```

---

set.java

```
/**
 * The set class represent a single unit in the all set class.
 * A set represents a single patient data entry
 * @author Mikyle Laurence O. Palomo Set class
 */
public class set {

   /**
    * Flag for selected
    */
   public boolean selected;
   /**
    * Array for inputs
    */
   public double input[];
   /**
    * Array for target
    */
   public double target[];

   /**
    * Constructor for training/validation sets
    * @param temp_inputs Input array
    * @param temp_targets Target array
    *
    */
   public set(double[] temp_inputs, double[] temp_targets)
{
      input = new double[temp_inputs.length];
      target = new double[temp_targets.length];
      for (int i = 0; i < temp_inputs.length; i++) {
         input[i] = temp_inputs[i];
      }
      for (int i = 0; i < temp_targets.length; i++) {
         target[i] = temp_targets[i];
      }
      selected = false;
   }

   /**
    * Constructor for diagnosis sets
    * @param temp_inputs Input array
    */
   public set(double[] temp_inputs) {
      input = new double[temp_inputs.length];
      for (int i = 0; i < temp_inputs.length; i++) {
         input[i] = temp_inputs[i];
      }
      selected = false;
   }
}
```

SetNormalizer.java

```
/**
 * Set normalizer class. This
 * @author Mikyle Laurence O. Palomo
 *
 */
public class SetNormalizer {

   private set[] normalizedTrainPatterns; // patterns to be
used during training
   private set[] normalizedValPatterns; // patterns to be
used during  validation
   private set[] normalizedDiagPatterns; //patterns to be
diagnosed
   private set[] trainPatterns; //orig train pattern
   private set[] valPatterns; //orig val pattern
   private set[] diagPatterns; //orig diag pattern
   private boolean normalizedTrain = false;
   private boolean normalizedVal = false;
   private boolean normalizedDiag = false;
   double newMin = -1;
   double newMax = 1;
   int numofinputs;
   int numofoutputs;

   /**
    * No-argument Constructor
    */
   public SetNormalizer() {
   }

   /**
    * Initialize normalizer with training/validation patterns
    * @param x Training patterns
    * @param y Validating patterns
    * @param numofinputs Number of input nodes
    * @param numofoutputs Number of output nodes
    * *
    */
   public SetNormalizer(set[] x, set[] y, int numofinputs, int
numofoutputs) {
      setTrainPatterns(x);
```

```
      setValPatterns(y);
      this.numofinputs = numofinputs;
      this.numofoutputs = numofoutputs;
   }

   /**
    * Initialize normalizer with diagnosis patterns
    * @param x Diagnosis Patterns
    * @param numofinputs Number of input nodes
    * @param numofoutputs Number of output nodes
    */
   public SetNormalizer(set[] x, int numofinputs, int
numofoutputs) {
      diagPatterns = x;
      this.numofinputs = numofinputs;
      this.numofoutputs = numofoutputs;

   }

   void normalize() {
      normalizeTrainSet();
      normalizeValidSet();

   }

   void normalizeDiag() {
      normalizeDiagSet();
   }

   private void normalizeDiagSet() {
      System.out.println("val:" + isNormalizedVal());
      System.out.println("train:" + isNormalizedTrain());
      System.out.println("diag:" + isNormalizedDiag());
      if (isNormalizedTrain() && !isNormalizedDiag()) {
         System.out.println("Normalizing Diagnosis Set");

         //copy patterns to temp pass-by-value
         //copy diagpattern to temp pass-by-value
         set[] tempD = new set[getDiagPatterns().length];
         int tempCounter = 0;
         for (set s : getDiagPatterns()) {
            tempD[tempCounter] = new set(s.input);
            tempCounter++;
         }

         //copy trainpattern to temp pass-by-value
         set[] tempV = new set[getTrainPatterns().length];
         tempCounter = 0;
         for (set s : getTrainPatterns()) {
            tempV[tempCounter] = new set(s.input, s.target);
            tempCounter++;
         }

         //copy pattern to temp pass-by-value
         set[] tempArr = new set[getDiagPatterns().length +
getTrainPatterns().length];
         int tempArrCounter = 0;
         for (set s : tempV) {
            tempArr[tempArrCounter] = new set(s.input,
s.target);
            tempArrCounter++;
         }

         for (set s : tempD) {
            tempArr[tempArrCounter] = new set(s.input);
            tempArrCounter++;
         }

         double[] minsIn = new double[numofinputs];
         double[] minsTar = new double[numofoutputs];
         double[] maxsIn = new double[numofinputs];
         double[] maxsTar = new double[numofoutputs];
         boolean firstEnt = true;

         //get all max and mins
         for (int i = 0; i < tempArr.length; i++) {
            double[] currRow = tempArr[i].input;
            for (int j = 0; j < numofinputs; j++) {
               if (i > 0) {
                  firstEnt = false;
               }

               if (firstEnt) {
                  minsIn[j] = tempArr[i].input[j];
                  maxsIn[j] = tempArr[i].input[j];
               }


               if (currRow[j] > maxsIn[j]) {
                  maxsIn[j] = currRow[j];
               }

               if (currRow[j] < minsIn[j]) {
                  minsIn[j] = currRow[j];
               }


            }


         }

         //normalize inputs
         for (int i = 0; i < tempArr.length; i++) {
            double[] currRow = tempArr[i].input;
            for (int j = 0; j < numofinputs; j++) {
               double temp_double = 0;

               temp_double = ((currRow[j] - minsIn[j]) /
(maxsIn[j] - minsIn[j])) * (newMax - newMin) + newMin;
               tempArr[i].input[j] = temp_double;

            }
         }
         //get normalized diag patterns from tempArr
         int tempDCounter = 0;
```

```
        //reset tempD
        tempD = new set[getDiagPatterns().length];
        for (int i = tempV.length; i < tempArr.length; i++) {
          tempD[tempDCounter] = new
set(tempArr[i].input);
          tempDCounter++;
        }


        setNormalizedDiagPatterns(tempD);
        System.out.println("NormalizedDiagLen: " +
tempD.length);
        setNormalizedDiag(true);

        System.out.println("Norm Diag Content");
        for (set s : tempD) {
          for (int col = 0; col < 8; col++) {
            System.out.print(s.input[col] + " ");
          }

          System.out.println();
        }
        System.out.println();
        System.out.println("Orig Diag Content");
        for (set s : getDiagPatterns()) {
          for (int col = 0; col < 8; col++) {
            System.out.print(s.input[col] + " ");
          }

          System.out.println();
        }
        System.out.println();

      }



    }

    void normalizeTrainSet() {
      if (!isNormalizedTrain()) {
        System.out.println("Normalizing Train Set");
        //copy pattern to temp pass-by-value
        set[] temp = new set[getTrainPatterns().length];
//copy trainpattern to temp pass-by-value
        int tempCounter = 0;

        for (set s : getTrainPatterns()) {
          temp[tempCounter] = new set(s.input, s.target);
          tempCounter++;
        }
        //copy trainpattern to temp pass-by-value

        double[] minsIn = new double[numofinputs];
        double[] minsTar = new double[numofoutputs];
        double[] maxsIn = new double[numofinputs];
        double[] maxsTar = new double[numofoutputs];

        boolean firstEnt = true;
        //init arbitrary initial entries


        //get all max and mins
        for (int i = 0; i < temp.length; i++) {
          double[] currRow = temp[i].input;
          for (int j = 0; j < numofinputs; j++) {
            if (i > 0) {
              firstEnt = false;
            }

            if (firstEnt) {
              minsIn[j] = temp[i].input[j];
              maxsIn[j] = temp[i].input[j];
            }


            if (currRow[j] > maxsIn[j]) {
              maxsIn[j] = currRow[j];
            }

            if (currRow[j] < minsIn[j]) {
              minsIn[j] = currRow[j];
            }


          }

          currRow = temp[i].target;
          for (int k = 0; k < numofoutputs; k++) {
            if (i > 0) {
              firstEnt = false;
            }

            if (firstEnt) {
              minsTar[k] = temp[i].target[k];
              maxsTar[k] = temp[i].target[k];
            }

            if (currRow[k] > maxsTar[k]) {
              maxsTar[k] = currRow[k];
            }
            if (currRow[k] < minsTar[k]) {
              minsTar[k] = currRow[k];
            }

          }
        }

        //normalize inputs
        for (int i = 0; i < temp.length; i++) {
          double[] currRow = temp[i].input;
          for (int j = 0; j < numofinputs; j++) {
            double temp_double = 0;
```

```
            temp_double = ((currRow[j] - minsIn[j]) /
(maxsIn[j] - minsIn[j])) * (newMax - newMin) + newMin;
            temp[i].input[j] = temp_double;

         }
         //normalize targets
         /*currRow = temp[i].target;
         for(int j = 0; j < numofoutputs; j++){
               double temp_double = 0;
               temp_double = ((currRow[j] -
minsTar[j])/(maxsTar[j]-minsTar[j])*(newMax-
newMin))+newMin;
               temp[i].target[j] = temp_double;

         }*/
      }
      setNormalizedTrainPatterns(temp);
      System.out.println("NormalizedTrainLen: " +
temp.length);
      setNormalizedTrain(true);

      System.out.println("======================");
      System.out.println("NORM TRAIN");
      for (set s : getNormalizedTrainPatterns()) {
         for (double d : s.input) {
            System.out.print(d + " ");
         }
         for (double d : s.target) {
            System.out.print(d + " ");
         }
         System.out.println();
      }
      System.out.println();

      System.out.println("======================");
      System.out.println("ORIG TRAIN");
      for (set s : getTrainPatterns()) {
         for (double d : s.input) {
            System.out.print(d + " ");
         }
         for (double d : s.target) {
            System.out.print(d + " ");
         }
         System.out.println();
      }
      System.out.println();
      System.out.println("======================");

   }
}

void normalizeValidSet() {
   System.out.println("Normalizing Diagnosis Set");

   //copy pattern  to temp pass-by-value
   set[] temp = new set[getValPatterns().length]; //copy
trainpattern to temp pass-by-value
   int tempCounter = 0;
for (set s : getValPatterns()) {
   temp[tempCounter] = new set(s.input, s.target);
   tempCounter++;
}


double[] minsIn = new double[numofinputs];
double[] minsTar = new double[numofoutputs];
double[] maxsIn = new double[numofinputs];
double[] maxsTar = new double[numofoutputs];

boolean firstEnt = true;
//init arbitrary initial entries

//get all max and mins
for (int i = 0; i < temp.length; i++) {
   double[] currRow = temp[i].input;
   for (int j = 0; j < numofinputs; j++) {
      if (i > 0) {
         firstEnt = false;
      }

      if (firstEnt) {
         minsIn[j] = temp[i].input[j];
         maxsIn[j] = temp[i].input[j];
      }


      if (currRow[j] > maxsIn[j]) {
         maxsIn[j] = currRow[j];
      }

      if (currRow[j] < minsIn[j]) {
         minsIn[j] = currRow[j];
      }


   }

   currRow = temp[i].target;
   for (int k = 0; k < numofoutputs; k++) {
      if (i > 0) {
         firstEnt = false;
      }

      if (firstEnt) {
         minsTar[k] = temp[i].target[k];
         maxsTar[k] = temp[i].target[k];
      }

      if (currRow[k] > maxsTar[k]) {
         maxsTar[k] = currRow[k];
      }
      if (currRow[k] < minsTar[k]) {
         minsTar[k] = currRow[k];
      }
```

```
        }
    }

    //normalize inputs
    for (int i = 0; i < temp.length; i++) {
        double[] currRow = temp[i].input;
        for (int j = 0; j < numofinputs; j++) {
            double temp_double = 0;

            temp_double = ((currRow[j] - minsIn[j]) /
(maxsIn[j] - minsIn[j])) * (newMax - newMin) + newMin;
            temp[i].input[j] = temp_double;

        }
        //normalize target
        /*currRow = temp[i].target;
        for(int j = 0; j < numofoutputs; j++){
            double temp_double = 0;
            temp_double = ((currRow[j] -
minsTar[j])/(maxsTar[j]-minsTar[j])*(newMax-
newMin))+newMin;
            temp[i].target[j] = temp_double;

        }*/
    }
    setNormalizedValPatterns(temp);
    System.out.println("NormalizedValLen: " +
temp.length);
    setNormalizedVal(true);

    System.out.println("=======================");
    System.out.println("NORM VAL");
    for (set s : getNormalizedValPatterns()) {
        for (double d : s.input) {
            System.out.print(d + " ");
        }
        for (double d : s.target) {
            System.out.print(d + " ");
        }
        System.out.println();
    }
    System.out.println();

    System.out.println("=======================");
    System.out.println("ORIG VAL");
    for (set s : getValPatterns()) {
        for (double d : s.input) {
            System.out.print(d + " ");
        }
        for (double d : s.target) {
            System.out.print(d + " ");
        }
        System.out.println();
    }
    System.out.println();
    System.out.println("=======================");
```

```
    }

    /**
     * @return Normalized diagnosis patterns
     */
    public set[] getNormalizedDiagPatterns() {
        return normalizedDiagPatterns;
    }

    /**
     * @param normalizedDiagPatterns Normalized
diagnosis patterns
     */
    public void setNormalizedDiagPatterns(set[]
normalizedDiagPatterns) {
        this.normalizedDiagPatterns =
normalizedDiagPatterns;
    }

    /**
     * @return Flag for normalized diagnosis pattern
     */
    public boolean isNormalizedDiag() {
        return normalizedDiag;
    }

    /**
     * @param normalizedDiag Flag for normalized
diagnosis pattern
     */
    public void setNormalizedDiag(boolean normalizedDiag)
{
        this.normalizedDiag = normalizedDiag;
    }

    /**
     *
     * @return Normalized training patterns
     */
    public set[] getNormalizedTrainPatterns() {
        return normalizedTrainPatterns;
    }

    /**
     *
     * @param normalizedTrainPatterns Normalized training
patterns
     */
    public void setNormalizedTrainPatterns(set[]
normalizedTrainPatterns) {
        this.normalizedTrainPatterns =
normalizedTrainPatterns;
    }

    /**
     *
     * @return Normalized validation patterns
```

```
    */
    public set[] getNormalizedValPatterns() {
        return normalizedValPatterns;
    }

    /**
     *
     * @param normalizedValPatterns Normalized
validation patterns
     */
    public void setNormalizedValPatterns(set[]
normalizedValPatterns) {
        this.normalizedValPatterns = normalizedValPatterns;
    }

    /**
     *
     * @return Flag for normalized training patterns
     */
    public boolean isNormalizedTrain() {
        return normalizedTrain;
    }

    /**
     *
     * @param normalizedTrain Flag for normalized training
patterns
     */
    public void setNormalizedTrain(boolean
normalizedTrain) {
        this.normalizedTrain = normalizedTrain;
    }

    /**
     *
     * @return Flag for normalized validation patterns
     */
    public boolean isNormalizedVal() {
        return normalizedVal;
    }

    /**
     *
     * Set flag for normalized validation patterns
     * @param nomalizedVal
     */
    public void setNormalizedVal(boolean nomalizedVal) {
        this.normalizedVal = nomalizedVal;
    }

    /**
     *
     * @return Diagnosis patterns
     */
    public set[] getDiagPatterns() {
        return diagPatterns;
    }

    /**
     *
     * @param diagPatterns Diagnosis patterns
     */
    public void setDiagPatterns(set[] diagPatterns) {
        this.diagPatterns = diagPatterns;
    }

    /**
     *
     * @return Training patterns
     */
    public set[] getTrainPatterns() {
        return trainPatterns;
    }

    /**
     *
     * @param trainPatterns Training patterns
     */
    public void setTrainPatterns(set[] trainPatterns) {
        this.trainPatterns = trainPatterns;
    }

    /**
     *
     * @return Validation patterns
     */
    public set[] getValPatterns() {
        return valPatterns;
    }

    /**
     *
     * @param valPatterns Validation patterns
     */
    public void setValPatterns(set[] valPatterns) {
        this.valPatterns = valPatterns;

    }
}
```

SplashScreen.java

```
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Toolkit;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JWindow;
```

```
/**
 * Splash Screen Class. Displays the splash screen for a
given duration
 * @author Mikyle Laurence O. Palomo
 */

public class SplashScreen extends JWindow{
  private int duration;

  /**
   * Splash Screen Class. Display a screen before loading
the application
   * @param d Duration for the Splash Screen Dispaly
   */
  public SplashScreen(int d) {
   duration = d;

   JPanel content = (JPanel) getContentPane();
   JLabel label = new JLabel();
   label.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/gnatsplas
h.png"))); // NOI18N

   content.setBackground(Color.white);
   int width = 605;
   int height = 375;
   Dimension screen =
Toolkit.getDefaultToolkit().getScreenSize();
   int x = (screen.width - width) / 2;
   int y = (screen.height - height) / 2;
   setBounds(x, y, width, height);
   content.add(label, BorderLayout.CENTER);
   setVisible(true);
   try {
    Thread.sleep(duration);
   } catch (Exception e) {
   }
   setVisible(false);
  }

}
```

UI.java

```
import java.awt.Window;
import java.io.*;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.table.TableModel;

/**
 * UI Class contains the GUI of the application. Also, this
object stores all
 * user specified parameters
 * @author Mikyle Laurence O. Palomo UI Class
 */
public class UI extends javax.swing.JFrame {

  /**
   * Creates new form UI
   */
  public UI() {
    initComponents();
  }

  /**
   * Initialize UI components
   */
  @SuppressWarnings("unchecked")
  // <editor-fold defaultstate="collapsed"
desc="Generated Code">//GEN-BEGIN:initComponents
  private void initComponents() {

    FileChooser = new javax.swing.JFileChooser();
    jDialog2 = new javax.swing.JDialog();
    downloadTrainValTemplate = new
javax.swing.JLabel();
    jPanel1 = new javax.swing.JPanel();
    jLabel23 = new javax.swing.JLabel();
    jLabel24 = new javax.swing.JLabel();
    configurationPanel = new javax.swing.JPanel();
    jLabel11 = new javax.swing.JLabel();
    maxErrorBox = new javax.swing.JTextField();
    numOfHiddenNodesBox = new
javax.swing.JTextField();
    popSizeBox = new javax.swing.JTextField();
    jLabel1 = new javax.swing.JLabel();
    numOfGenBox = new javax.swing.JTextField();
    jLabel12 = new javax.swing.JLabel();
    jLabel13 = new javax.swing.JLabel();
    mutateBox = new javax.swing.JTextField();
    tolerableCheckBox = new javax.swing.JCheckBox();
    jLabel7 = new javax.swing.JLabel();
    jLabel8 = new javax.swing.JLabel();
    jLabel9 = new javax.swing.JLabel();
    jLabel10 = new javax.swing.JLabel();
    goButton = new javax.swing.JButton();
    configHelp = new javax.swing.JLabel();
    resetButton = new javax.swing.JButton();
    trainDatasetPanel = new javax.swing.JPanel();
    jLabel15 = new javax.swing.JLabel();
    trainDatasetName = new javax.swing.JTextField();
    jLabel3 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();
    trainBrowseButton = new javax.swing.JButton();
    jLabel5 = new javax.swing.JLabel();
    trainRatioBox = new javax.swing.JTextField();
    validRatioBox = new javax.swing.JTextField();
    fileNotif = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    jSeparator5 = new javax.swing.JSeparator();
```

```
        dlTvTemplate = new javax.swing.JLabel();
        jSeparator2 = new javax.swing.JSeparator();
        jSeparator6 = new javax.swing.JSeparator();
        jPanel2 = new javax.swing.JPanel();
        fileNotif1 = new javax.swing.JLabel();
        jScrollPane1 = new javax.swing.JScrollPane();
        trainingResultTextArea = new javax.swing.JTextArea();
        jLabel25 = new javax.swing.JLabel();
        resultHelp = new javax.swing.JPanel();
        jScrollPane2 = new javax.swing.JScrollPane();
        diagTable = new javax.swing.JTable();
        jLabel19 = new javax.swing.JLabel();
        resultsHelp = new javax.swing.JLabel();
        jSeparator4 = new javax.swing.JSeparator();
        jButton1 = new javax.swing.JButton();
        summaryPanel = new javax.swing.JPanel();
        jLabel6 = new javax.swing.JLabel();
        jLabel20 = new javax.swing.JLabel();
        jLabel21 = new javax.swing.JLabel();
        jLabel22 = new javax.swing.JLabel();
        finalSSELabel = new javax.swing.JLabel();
        timeElapsedLabel = new javax.swing.JLabel();
        numGenLabel = new javax.swing.JLabel();
        jSeparator3 = new javax.swing.JSeparator();
        diagnosisPanel = new javax.swing.JPanel();
        jLabel18 = new javax.swing.JLabel();
        jLabel17 = new javax.swing.JLabel();
        jLabel16 = new javax.swing.JLabel();
        batchDiagButton = new javax.swing.JButton();
        batchEntryBox = new javax.swing.JTextField();
        diagBrowseButton = new javax.swing.JButton();
        jLabel27 = new javax.swing.JLabel();
        singleEntryBox = new javax.swing.JTextField();
        jLabel28 = new javax.swing.JLabel();
        singleEntryDiagButton = new javax.swing.JButton();
        diagnosisHelp = new javax.swing.JLabel();
        downloadDiagTemplate = new javax.swing.JLabel();
        summaryHelp = new javax.swing.JLabel();
        trainResultHelp = new javax.swing.JLabel();

        FileChooser.setDialogTitle("Upload Data Set");

        jDialog2.setMinimumSize(new
java.awt.Dimension(100, 100));

        javax.swing.GroupLayout jDialog2Layout = new
javax.swing.GroupLayout(jDialog2.getContentPane());
        jDialog2.getContentPane().setLayout(jDialog2Layout);
        jDialog2Layout.setHorizontalGroup(

jDialog2Layout.createParallelGroup(javax.swing.GroupLay
out.Alignment.LEADING)
            .addGap(0, 400, Short.MAX_VALUE)
        );
        jDialog2Layout.setVerticalGroup(

jDialog2Layout.createParallelGroup(javax.swing.GroupLay
out.Alignment.LEADING)
            .addGap(0, 300, Short.MAX_VALUE)
        );

        downloadTrainValTemplate.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/dl.png")));
// NOI18N
        downloadTrainValTemplate.setToolTipText("Click to
download template");
        downloadTrainValTemplate.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        downloadTrainValTemplate.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void
mousePressed(java.awt.event.MouseEvent evt) {
                downloadTrainValTemplateOnClick(evt);
            }
        });


setDefaultCloseOperation(javax.swing.WindowConstants.E
XIT_ON_CLOSE);
        setBounds(new java.awt.Rectangle(0, 0, 0, 0));
        setResizable(false);

        jPanel1.setEnabled(false);

        jLabel24.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/gnatlogos
mall.png"))); // NOI18N

        jLabel11.setText("Population Size:");

        maxErrorBox.setToolTipText("Please provide a non-
zero value");

        numOfHiddenNodesBox.setToolTipText("Please
provide a non-zero value");

        popSizeBox.setToolTipText("Please provide a non-zero
value");

        jLabel1.setFont(new java.awt.Font("Calibri", 1, 18)); //
NOI18N
        jLabel1.setText("Configuration");

        numOfGenBox.setToolTipText("Please provide a non-
zero value");

        jLabel12.setText("Number of Generations:");

        jLabel13.setText("Mutation Rate:");

        mutateBox.setToolTipText("Please provide a value
from 0 - 1");

        tolerableCheckBox.setText("Stop training when error
is tolerable?");
        tolerableCheckBox.setToolTipText("");
```

```
    jLabel7.setFont(new java.awt.Font("Calibri", 1, 14)); //
NOI18N
    jLabel7.setText("Neural Network Parameters");

    jLabel8.setFont(new java.awt.Font("Calibri", 1, 14)); //
NOI18N
    jLabel8.setText("Genetic Algorithm Parameters");

    jLabel9.setText("Number of Hidden Nodes:");

    jLabel10.setText("Max Tolerable Error:");

    goButton.setText("GO>>>");
    goButton.setToolTipText("Click to proceed with
training");
    goButton.addActionListener(new
java.awt.event.ActionListener() {
        public void
actionPerformed(java.awt.event.ActionEvent evt) {
            goButtonActionPerformed(evt);
        }
    });

    configHelp.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/qMark.pn
g"))); // NOI18N
    configHelp.setToolTipText("Click to access help");
    configHelp.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
    configHelp.addMouseListener(new
java.awt.event.MouseAdapter() {
        public void
mousePressed(java.awt.event.MouseEvent evt) {
            configHelpdatasetHelpOnClick(evt);
        }
    });

    resetButton.setText("Reset");
    resetButton.setToolTipText("Click to Reset");
    resetButton.addActionListener(new
java.awt.event.ActionListener() {
        public void
actionPerformed(java.awt.event.ActionEvent evt) {
            resetButtonActionPerformed(evt);
        }
    });

    javax.swing.GroupLayout configurationPanelLayout =
new javax.swing.GroupLayout(configurationPanel);

configurationPanel.setLayout(configurationPanelLayout);
    configurationPanelLayout.setHorizontalGroup(

configurationPanelLayout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING)

.addGroup(configurationPanelLayout.createSequentialGro
up()
            .addContainerGap()

.addGroup(configurationPanelLayout.createParallelGroup(
javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(configurationPanelLayout.createSequentialGro
up()

.addGroup(configurationPanelLayout.createParallelGroup(
javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel8)

.addGroup(configurationPanelLayout.createSequentialGro
up()
                        .addGap(10, 10, 10)

.addGroup(configurationPanelLayout.createParallelGroup(
javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel12)
                            .addComponent(jLabel13)
                            .addComponent(jLabel11))
                        .addGap(26, 26, 26)

.addGroup(configurationPanelLayout.createParallelGroup(
javax.swing.GroupLayout.Alignment.TRAILING, false)
                            .addComponent(numOfGenBox,
javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(mutateBox,
javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(popSizeBox,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE, 172,
javax.swing.GroupLayout.PREFERRED_SIZE))))
                    .addGap(0, 0, Short.MAX_VALUE))

.addGroup(configurationPanelLayout.createSequentialGro
up()

.addGroup(configurationPanelLayout.createParallelGroup(
javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel7)

.addGroup(configurationPanelLayout.createSequentialGro
up()
                        .addGap(10, 10, 10)

.addGroup(configurationPanelLayout.createParallelGroup(
javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel9)
                            .addComponent(jLabel10))
                        .addGap(18, 18, 18)

.addGroup(configurationPanelLayout.createParallelGroup(
javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(tolerableCheckBox)
```

```
.addGroup(configurationPanelLayout.createParallelGroup(
javax.swing.GroupLayout.Alignment.TRAILING, false)
                        .addComponent(maxErrorBox,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 173,
Short.MAX_VALUE)

.addComponent(numOfHiddenNodesBox,
javax.swing.GroupLayout.Alignment.LEADING)))))
                .addContainerGap())

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
configurationPanelLayout.createSequentialGroup()
                .addGap(0, 0, Short.MAX_VALUE)
                .addComponent(resetButton,
javax.swing.GroupLayout.PREFERRED_SIZE, 69,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)
                .addComponent(goButton)
                .addGap(4, 4, 4))

.addGroup(configurationPanelLayout.createSequentialGro
up()
                .addComponent(jLabel1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(configHelp))))
        );
        configurationPanelLayout.setVerticalGroup(

configurationPanelLayout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING)

.addGroup(configurationPanelLayout.createSequentialGro
up()

.addGroup(configurationPanelLayout.createParallelGroup(
javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel1)
                .addComponent(configHelp))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.UNRELATED)
                .addComponent(jLabel7)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)

.addGroup(configurationPanelLayout.createParallelGroup(
javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel9)
                .addComponent(numOfHiddenNodesBox,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(6, 6, 6)

.addGroup(configurationPanelLayout.createParallelGroup(
javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel10)
                .addComponent(maxErrorBox,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.UNRELATED)
                .addComponent(tolerableCheckBox)
                .addGap(29, 29, 29)
                .addComponent(jLabel8)
                .addGap(10, 10, 10)

.addGroup(configurationPanelLayout.createParallelGroup(
javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(popSizeBox,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel11))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)

.addGroup(configurationPanelLayout.createParallelGroup(
javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(numOfGenBox,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel12))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)

.addGroup(configurationPanelLayout.createParallelGroup(
javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(mutateBox,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel13))
            .addGap(29, 29, 29)

.addGroup(configurationPanelLayout.createParallelGroup(
javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(resetButton)
                .addComponent(goButton))
            .addContainerGap(88, Short.MAX_VALUE))
        );
```

```
        jLabel15.setFont(new java.awt.Font("Calibri", 1, 18));
// NOI18N
        jLabel15.setText(" Data Set");

        trainDatasetName.setToolTipText("Please select a
file");

        jLabel3.setText("Upload Data Set:");

        jLabel4.setText("Training Ratio:");

        trainBrowseButton.setText("Browse");
        trainBrowseButton.setToolTipText("Click to browse
for a data set");
        trainBrowseButton.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                trainBrowseButtonActionPerformed(evt);
            }
        });

        jLabel5.setText("Validating Ratio:");

        trainRatioBox.setToolTipText("Please provide a value
from  0 - 1");

        validRatioBox.setToolTipText("Please provide a value
from  0 - 1");

        fileNotif.setForeground(new java.awt.Color(0, 255,
0));
        fileNotif.setName("fileNotif");

        jLabel2.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/qMark.pn
g"))); // NOI18N
        jLabel2.setToolTipText("Click to access help");
        jLabel2.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        jLabel2.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void
mousePressed(java.awt.event.MouseEvent evt) {
                datasetHelpOnClick(evt);
            }
        });

        dlTvTemplate.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/dl.png")));
// NOI18N
        dlTvTemplate.setToolTipText("Click to download
template");
        dlTvTemplate.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        dlTvTemplate.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void
mousePressed(java.awt.event.MouseEvent evt) {
                downloadTrainValTemplateOnClick(evt);
            }
        });

        fileNotif.setVisible(false);

        javax.swing.GroupLayout trainDatasetPanelLayout =
new javax.swing.GroupLayout(trainDatasetPanel);

trainDatasetPanel.setLayout(trainDatasetPanelLayout);
        trainDatasetPanelLayout.setHorizontalGroup(

trainDatasetPanelLayout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)

.addGroup(trainDatasetPanelLayout.createSequentialGrou
p()
                .addComponent(jLabel15)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(dlTvTemplate)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)
                .addComponent(jLabel2))

.addGroup(trainDatasetPanelLayout.createSequentialGrou
p()

.addGroup(trainDatasetPanelLayout.createParallelGroup(j
avax.swing.GroupLayout.Alignment.LEADING)

.addGroup(trainDatasetPanelLayout.createSequentialGrou
p()
                        .addContainerGap()

.addGroup(trainDatasetPanelLayout.createParallelGroup(j
avax.swing.GroupLayout.Alignment.TRAILING, false)

.addGroup(trainDatasetPanelLayout.createSequentialGrou
p()
                                .addComponent(jLabel4,
javax.swing.GroupLayout.PREFERRED_SIZE, 94,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                .addComponent(trainRatioBox,
javax.swing.GroupLayout.PREFERRED_SIZE, 207,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addGroup(javax.swing.GroupLayout.Alignment.LEADING,
trainDatasetPanelLayout.createSequentialGroup()
```

```
                    .addComponent(jLabel3,
javax.swing.GroupLayout.PREFERRED_SIZE, 94,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(18, 18, 18)
                    .addComponent(trainDatasetName,
javax.swing.GroupLayout.PREFERRED_SIZE, 207,
javax.swing.GroupLayout.PREFERRED_SIZE)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.UNRELATED)
                .addComponent(trainBrowseButton,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))

.addGroup(trainDatasetPanelLayout.createSequentialGrou
p()

.addGroup(trainDatasetPanelLayout.createParallelGroup(j
avax.swing.GroupLayout.Alignment.LEADING)

.addGroup(trainDatasetPanelLayout.createSequentialGrou
p()
                    .addGap(158, 158, 158)
                    .addComponent(fileNotif,
javax.swing.GroupLayout.PREFERRED_SIZE, 171,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addGroup(trainDatasetPanelLayout.createSequentialGrou
p()
                    .addContainerGap()
                    .addComponent(jLabel5,
javax.swing.GroupLayout.PREFERRED_SIZE, 94,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(18, 18, 18)
                    .addComponent(validRatioBox,
javax.swing.GroupLayout.PREFERRED_SIZE, 207,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGap(0, 0, Short.MAX_VALUE)))
            .addContainerGap())
        .addComponent(jSeparator5)
    );
    trainDatasetPanelLayout.setVerticalGroup(

trainDatasetPanelLayout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)

.addGroup(trainDatasetPanelLayout.createSequentialGrou
p()
            .addComponent(jSeparator5,
javax.swing.GroupLayout.PREFERRED_SIZE, 5,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)

.addGroup(trainDatasetPanelLayout.createParallelGroup(j
avax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel15,
javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel2)
                .addComponent(dlTvTemplate))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.UNRELATED)

.addGroup(trainDatasetPanelLayout.createParallelGroup(j
avax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(trainBrowseButton)
                .addComponent(trainDatasetName,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel3,
javax.swing.GroupLayout.PREFERRED_SIZE, 14,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)

.addGroup(trainDatasetPanelLayout.createParallelGroup(j
avax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel4)
                .addComponent(trainRatioBox,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.UNRELATED)

.addGroup(trainDatasetPanelLayout.createParallelGroup(j
avax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(validRatioBox,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel5,
javax.swing.GroupLayout.PREFERRED_SIZE, 14,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.UNRELATED)
            .addComponent(fileNotif,
javax.swing.GroupLayout.PREFERRED_SIZE, 16,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZ
E, Short.MAX_VALUE))
    );

    javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
    jPanel1.setLayout(jPanel1Layout);
    jPanel1Layout.setHorizontalGroup(
```

```
jPanel1Layout.createParallelGroup(javax.swing.GroupLayo
ut.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
            .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.TRAILING)

.addGroup(jPanel1Layout.createSequentialGroup()
                    .addGap(0, 0, Short.MAX_VALUE)
                    .addComponent(jLabel23)
                    .addGap(18, 18, 18)
                    .addComponent(jLabel24,
javax.swing.GroupLayout.PREFERRED_SIZE, 412,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addGroup(jPanel1Layout.createSequentialGroup()
                    .addComponent(jSeparator2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                        .addComponent(configurationPanel,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addComponent(jSeparator6)

.addGroup(jPanel1Layout.createSequentialGroup()
                            .addComponent(trainDatasetPanel,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addGap(0, 0, Short.MAX_VALUE)))))
            .addContainerGap())
        );
        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayo
ut.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.TRAILING)

.addGroup(jPanel1Layout.createSequentialGroup()
                    .addGap(160, 160, 160)
                    .addComponent(jLabel23)
                    .addGap(0, 0, Short.MAX_VALUE)
                    .addComponent(jSeparator2,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(12, 12, 12))

.addGroup(jPanel1Layout.createSequentialGroup()

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZ
E, Short.MAX_VALUE)
                    .addComponent(jLabel24,
javax.swing.GroupLayout.PREFERRED_SIZE, 214,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)
                    .addComponent(trainDatasetPanel,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)))
                .addComponent(jSeparator6,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)
                .addComponent(configurationPanel,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(401, 401, 401))
        );


jLabel24.getAccessibleContext().setAccessibleName("DNLo
go");

        fileNotif1.setForeground(new java.awt.Color(0, 255,
0));
        fileNotif1.setName("fileNotif");

        trainingResultTextArea.setColumns(20);
        trainingResultTextArea.setEditable(false);
        trainingResultTextArea.setRows(5);

jScrollPane1.setViewportView(trainingResultTextArea);

        jLabel25.setFont(new java.awt.Font("Calibri", 1, 16));
// NOI18N
        jLabel25.setText("Training/Validation Result");

        diagTable.setModel(new
javax.swing.table.DefaultTableModel(
            new Object [][] {

            },
```

```
            new String [] {
                "Temp.", "P. Count", "C. Rate", "BP(SP)",
"BP(DP)", "R. Rate", "WBC", "RBC", "Diagnosis"
            }
        ) {
            boolean[] canEdit = new boolean [] {
                false, false, false, false, false, false, false, false,
false
            };

            public boolean isCellEditable(int rowIndex, int
columnIndex) {
                return canEdit [columnIndex];
            }
        });
        diagTable.setCellSelectionEnabled(true);
        jScrollPane2.setViewportView(diagTable);

diagTable.getColumnModel().getSelectionModel().setSelec
tionMode(javax.swing.ListSelectionModel.SINGLE_INTERV
AL_SELECTION);

        jLabel19.setFont(new java.awt.Font("Calibri", 1, 16));
// NOI18N
        jLabel19.setText("Results:");

        resultsHelp.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/qMark.pn
g"))); // NOI18N
        resultsHelp.setToolTipText("Click to access help");
        resultsHelp.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        resultsHelp.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void
mousePressed(java.awt.event.MouseEvent evt) {
                diagResultsHelpOnClick(evt);
            }
        });

        jButton1.setText("Export");
        jButton1.setToolTipText("Click to export diagnosis
results");
        jButton1.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                exportButtonOnClick(evt);
            }
        });

        javax.swing.GroupLayout resultHelpLayout = new
javax.swing.GroupLayout(resultHelp);
        resultHelp.setLayout(resultHelpLayout);
        resultHelpLayout.setHorizontalGroup(

resultHelpLayout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.LEADING)
            .addComponent(jScrollPane2)

.addGroup(resultHelpLayout.createSequentialGroup()

.addGroup(resultHelpLayout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.TRAILING, false)

.addGroup(resultHelpLayout.createSequentialGroup()
                    .addComponent(jLabel19)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                    .addComponent(resultsHelp,
javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addComponent(jSeparator4,
javax.swing.GroupLayout.PREFERRED_SIZE, 463,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(0, 0, Short.MAX_VALUE))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
resultHelpLayout.createSequentialGroup()

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZ
E, Short.MAX_VALUE)
            .addComponent(jButton1)
            .addContainerGap())
        );
        resultHelpLayout.setVerticalGroup(

resultHelpLayout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.LEADING)

.addGroup(resultHelpLayout.createSequentialGroup()
            .addComponent(jSeparator4,
javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)

.addGroup(resultHelpLayout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel19)
                .addComponent(resultsHelp))
            .addGap(10, 10, 10)
            .addComponent(jScrollPane2,
javax.swing.GroupLayout.DEFAULT_SIZE, 229,
Short.MAX_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)
            .addComponent(jButton1))
        );

        jLabel6.setFont(new java.awt.Font("Calibri", 1, 16)); //
NOI18N
```

```
        jLabel6.setText("Summary");

        jLabel20.setText("SSE:");

        jLabel21.setText("Time Elapsed:");

        jLabel22.setText("Number of Generations:");

        numGenLabel.setToolTipText("");

        jLabel18.setText("Upload Data Set:");

        jLabel17.setFont(new java.awt.Font("Calibri", 1, 14));
// NOI18N
        jLabel17.setText("Batch Upload");

        jLabel16.setFont(new java.awt.Font("Calibri", 1, 16));
// NOI18N
        jLabel16.setText("Diagnosis");

        batchDiagButton.setText("GO");
        batchDiagButton.setToolTipText("Click to proceed
with diagnosis");
        batchDiagButton.setEnabled(false);
        batchDiagButton.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                batchDiagButtonActionPerformed(evt);
            }
        });

        batchEntryBox.setToolTipText("Please select a file");
        batchEntryBox.setEnabled(false);

        diagBrowseButton.setText("Browse");
        diagBrowseButton.setToolTipText("Click to browse for
a data set");
        diagBrowseButton.setEnabled(false);
        diagBrowseButton.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                diagBrowseButtonActionPerformed(evt);
            }
        });

        jLabel27.setFont(new java.awt.Font("Calibri", 1, 14));
// NOI18N
        jLabel27.setText("Single Entry");

        singleEntryBox.setToolTipText("Please provide an
entry with format:<value1>;<value2>;<value3>;...;");
        singleEntryBox.setEnabled(false);

        jLabel28.setText("Upload Data Entry:     ");

        singleEntryDiagButton.setText("GO");
        singleEntryDiagButton.setToolTipText("Click to
proceed with diagnosis");
        singleEntryDiagButton.setEnabled(false);
        singleEntryDiagButton.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                singleEntryDiagButtonActionPerformed(evt);
            }
        });

        diagnosisHelp.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/qMark.pn
g"))); // NOI18N
        diagnosisHelp.setToolTipText("Click to access help");
        diagnosisHelp.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        diagnosisHelp.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void
mousePressed(java.awt.event.MouseEvent evt) {
                diagnosisHelpOnClick(evt);
            }
        });

        downloadDiagTemplate.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/dl.png")));
// NOI18N
        downloadDiagTemplate.setToolTipText("Click to
download template");
        downloadDiagTemplate.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        downloadDiagTemplate.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void
mousePressed(java.awt.event.MouseEvent evt) {
                downloadDiagTemplateOnClick(evt);
            }
        });

        javax.swing.GroupLayout diagnosisPanelLayout = new
javax.swing.GroupLayout(diagnosisPanel);
        diagnosisPanel.setLayout(diagnosisPanelLayout);
        diagnosisPanelLayout.setHorizontalGroup(

diagnosisPanelLayout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.LEADING)

.addGroup(diagnosisPanelLayout.createSequentialGroup()

.addGroup(diagnosisPanelLayout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel17)

.addGroup(diagnosisPanelLayout.createSequentialGroup()
                        .addContainerGap()
```

```
.addGroup(diagnosisPanelLayout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel18,
javax.swing.GroupLayout.PREFERRED_SIZE, 93,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel28))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)

.addGroup(diagnosisPanelLayout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING, false)

.addGroup(diagnosisPanelLayout.createSequentialGroup()
                    .addComponent(batchEntryBox,
javax.swing.GroupLayout.PREFERRED_SIZE, 155,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)
                    .addComponent(diagBrowseButton))
                .addComponent(singleEntryBox))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)

.addGroup(diagnosisPanelLayout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                .addComponent(singleEntryDiagButton)
                .addComponent(batchDiagButton)))
            .addComponent(jLabel27))
        .addContainerGap(53, Short.MAX_VALUE))

.addGroup(diagnosisPanelLayout.createSequentialGroup()
        .addComponent(jLabel16)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
        .addComponent(downloadDiagTemplate)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)
        .addComponent(diagnosisHelp,
javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE))
    );
    diagnosisPanelLayout.setVerticalGroup(

diagnosisPanelLayout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.LEADING)

.addGroup(diagnosisPanelLayout.createSequentialGroup()
        .addGap(2, 2, 2)

.addGroup(diagnosisPanelLayout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                .addComponent(downloadDiagTemplate)
                .addComponent(diagnosisHelp)

.addGroup(diagnosisPanelLayout.createSequentialGroup()
                .addComponent(jLabel16)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)
                .addComponent(jLabel17)
                .addGap(4, 4, 4)

.addGroup(diagnosisPanelLayout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel18,
javax.swing.GroupLayout.PREFERRED_SIZE, 14,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(batchEntryBox,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(diagBrowseButton)
                    .addComponent(batchDiagButton))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)
                .addComponent(jLabel27)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)

.addGroup(diagnosisPanelLayout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel28,
javax.swing.GroupLayout.PREFERRED_SIZE, 14,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(singleEntryBox,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(singleEntryDiagButton))))
            .addContainerGap(31, Short.MAX_VALUE))
    );

    summaryHelp.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/qMark.pn
g"))); // NOI18N
    summaryHelp.setToolTipText("Click to access help");
    summaryHelp.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
    summaryHelp.addMouseListener(new
java.awt.event.MouseAdapter() {
        public void
mousePressed(java.awt.event.MouseEvent evt) {
            summaryHelpOnClick(evt);
        }
    });
```

```
    javax.swing.GroupLayout summaryPanelLayout = new
javax.swing.GroupLayout(summaryPanel);
    summaryPanel.setLayout(summaryPanelLayout);
    summaryPanelLayout.setHorizontalGroup(

summaryPanelLayout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.LEADING)
        .addComponent(diagnosisPanel,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)

.addGroup(summaryPanelLayout.createSequentialGroup()
            .addComponent(jLabel6)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
            .addComponent(summaryHelp,
javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addGroup(summaryPanelLayout.createSequentialGroup()
            .addContainerGap()

.addGroup(summaryPanelLayout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel20)
                .addComponent(jLabel21)
                .addComponent(jLabel22))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)

.addGroup(summaryPanelLayout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING, false)
                .addComponent(timeElapsedLabel,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(numGenLabel,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(finalSSELabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 225,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(0, 0, Short.MAX_VALUE))
        .addComponent(jSeparator3,
javax.swing.GroupLayout.Alignment.TRAILING)
    );
    summaryPanelLayout.setVerticalGroup(

summaryPanelLayout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.LEADING)

.addGroup(summaryPanelLayout.createSequentialGroup()
            .addContainerGap()

.addGroup(summaryPanelLayout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel6)
                .addComponent(summaryHelp))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)

.addGroup(summaryPanelLayout.createParallelGroup(java
x.swing.GroupLayout.Alignment.TRAILING, false)

.addGroup(summaryPanelLayout.createSequentialGroup()
                .addComponent(jLabel20)
                .addGap(5, 5, 5)
                .addComponent(jLabel21))

.addGroup(summaryPanelLayout.createSequentialGroup()
                .addComponent(finalSSELabel,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)
                .addComponent(timeElapsedLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 14,
javax.swing.GroupLayout.PREFERRED_SIZE)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)

.addGroup(summaryPanelLayout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel22,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(numGenLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 18,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(22, 22, 22)
            .addComponent(jSeparator3,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)
            .addComponent(diagnosisPanel,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap())
    );
```

```
        trainResultHelp.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/qMark.pn
g"))); // NOI18N
        trainResultHelp.setToolTipText("Click to access help");
        trainResultHelp.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        trainResultHelp.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void
mousePressed(java.awt.event.MouseEvent evt) {
                trainResultHelpOnClick(evt);
            }
        });

        fileNotif.setVisible(false);

        javax.swing.GroupLayout jPanel2Layout = new
javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayo
ut.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()
                        .addComponent(resultHelp,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(27, 27, 27)
                        .addComponent(fileNotif1)
                        .addGap(0, 0, Short.MAX_VALUE))

.addGroup(jPanel2Layout.createSequentialGroup()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.TRAILING, false)

.addGroup(jPanel2Layout.createSequentialGroup()
                            .addComponent(jLabel25,
javax.swing.GroupLayout.PREFERRED_SIZE, 215,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                            .addComponent(trainResultHelp,
javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addComponent(jScrollPane1,
javax.swing.GroupLayout.DEFAULT_SIZE, 463,
Short.MAX_VALUE)
                        .addComponent(summaryPanel,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addContainerGap())))
        );
        jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayo
ut.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel2Layout.createSequentialGroup()
                .addGap(6, 6, 6)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addComponent(trainResultHelp)
                    .addComponent(jLabel25,
javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)
                .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 183,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()
                        .addGap(0, 0, Short.MAX_VALUE)
                        .addComponent(fileNotif1)
                        .addGap(40, 40, 40))

.addGroup(jPanel2Layout.createSequentialGroup()

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)
                        .addComponent(summaryPanel,
javax.swing.GroupLayout.PREFERRED_SIZE, 266,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(12, 12, 12)
                        .addComponent(resultHelp,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addContainerGap())))
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
```

```
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE, 442, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE, 473, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap(19, Short.MAX_VALUE))
      );
      layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE, 750, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(0, 72, Short.MAX_VALUE))
        .addComponent(jPanel2, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
      );

      pack();
   }// </editor-fold>//GEN-END:initComponents

   private void trainBrowseButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_trainBrowseButtonActionPerformed

      try {
         File name = getFile();
         String path = name.getAbsolutePath();
         trainDatasetName.setText(path);
         if (name.isFile()) {   //file ok
            String type = name.getName().substring((name.getName().indexOf(".")) + 1); //extract file type
            if (type.equals("csv") || type.equals("txt") || type.equals("dat")) {
               /*
                * fileNotif.setText("File OK!");
                * fileNotif.setForeground(new java.awt.Color(0, 255, 0));
               goButton.setEnabled(true);
                */
            } else {
               JOptionPane.showMessageDialog(this, "Select a .CSV, .DAT, or .TXT file.", "Dataset Error!", JOptionPane.ERROR_MESSAGE);
            }
         } else if (name.isDirectory()) {
            JOptionPane.showMessageDialog(this, "File error! Please select a file.", "Dataset Error!", JOptionPane.ERROR_MESSAGE);
         }
      } catch (NullPointerException e) {
         System.err.println("Null Pointer Exception: File");
         System.err.print(e);
      } catch (Exception e) {
         System.err.println("Exception: File");
         System.err.print(e);
      }
   }//GEN-LAST:event_trainBrowseButtonActionPerformed

   private void goButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_goButtonActionPerformed

      //disable diagnosis fields before training
      batchDiagButton.setEnabled(false);
      batchEntryBox.setEnabled(false);
      diagBrowseButton.setEnabled(false);

      singleEntryBox.setEnabled(false);
      singleEntryDiagButton.setEnabled(false);

      boolean hasError = true;
      StringBuilder errorToOutput = new StringBuilder("");
      int errSignal = 0;
      //CHECK FOR EXCEPTIONS!

      //FILE VALIDATION
      try {
         File testFile = new File(trainDatasetName.getText());
         if (null == testFile || !testFile.exists()) {
            System.err.println("File Error: File not found");
            errorToOutput.append("Please provide a valid file.\n");
            errSignal++;
         }
      } catch (Exception e) {
         System.err.println("Exception: File");
         errSignal++;
      }
      //DATASET VALIDATION
      try {
         String tempTitle = trainDatasetName.getText();
         if (tempTitle.trim().length() == 0 || tempTitle.isEmpty()) {
            System.err.println("Dataset Error: No file provided");
            errorToOutput.append("Please provide a valid file.\n");
            errSignal++;
```

```
        }

    } catch (NullPointerException e) {
        System.err.println("Null Pointer Exception:
Dataset");
        errSignal++;

    } catch (Exception e) {
        System.err.println("Exception: Dataset");
        errSignal++;

    }

    //RATIO VALIDATION
    try {

        double tempTRatio =
Double.parseDouble(trainRatioBox.getText());
        double tempVRatio =
Double.parseDouble(validRatioBox.getText());
        if (tempTRatio > 1 || tempTRatio <= 0) {
            System.err.println("Training Ratio Error: must be
0 < x < 1");
            errorToOutput.append("Training Ratio must be
greater than 0 and less than 1.\n");
            errSignal++;
            //notify user
        }
        if (tempVRatio > 1 || tempVRatio < 0) {
            System.err.println("Validating Ratio Error: must
be 0 < x < 1");
            errorToOutput.append("Validating Ratio must be
greater than 0 and less than 1.\n");
            errSignal++;
            //notify user
        }

        if (tempTRatio + tempVRatio > 1 || tempTRatio +
tempVRatio <= 0) {
            System.err.println("Ratio Error: max sum: 1; min
sum > 0");
            errorToOutput.append("Ratio sums must be
greater than 0 and less than 1.\n");
            errSignal++;
            //notify user
        }


    } catch (NumberFormatException e) {
        System.err.println("Number Format Exception:
Ratio");
        errorToOutput.append("Please check data set
parameters.\n");
        errSignal++;

    } catch (NullPointerException e) {
        System.err.println("Null Pointer Exception: Ratio");
        errorToOutput.append("Please check data set
parameters.\n");
        errSignal++;

    } catch (Exception e) {
        System.err.println("Exception: Ratio");
        errorToOutput.append("Dataset Error!\n");
        errSignal++;

    }

    //NN VALIDATION
    try {
        int tempNodes =
Integer.parseInt(numOfHiddenNodesBox.getText());
        double tempMaxError =
Double.parseDouble(maxErrorBox.getText());

        if (tempNodes <= 0) {
            System.err.println("NN Error: tempNodes must
be > 0");
            errorToOutput.append("Number of nodes must
be greater than 0.\n");
            errSignal++;
        }
        if (tempMaxError <= 0) {
            System.err.println("NN Error: tempMaxError
must be > 0");
            errorToOutput.append("Tolerable Error must be
greater than 0.\n");
            errSignal++;
        }


    } catch (NumberFormatException e) {
        System.err.println("Number Format: NN");
        errorToOutput.append("Please check NN
parameters.\n");
        errSignal++;

    } catch (NullPointerException e) {
        System.err.println("Null Pointer Exception: NN");
        errorToOutput.append("Please check NN
parameters.\n");
        errSignal++;

    } catch (Exception e) {
        System.err.println("Exception: NN");
        errorToOutput.append("Please check NN
parameters.\n");
        errSignal++;

    }

    try {
        int tempPop =
Integer.parseInt(popSizeBox.getText());
```

```
        int tempGen =
Integer.parseInt(numOfGenBox.getText());
        double tempMut =
Double.parseDouble(mutateBox.getText());

        if (tempPop <= 0) {
           System.err.println("GA Error: tempPop must be >
0");
           errorToOutput.append("Population size must be
greater than 0.\n");
           errSignal++;
        }

        if (tempGen <= 0) {
           System.err.println("GA Error: tempGen must be >
0");
           errorToOutput.append("Number of generations
must be greater than 0.\n");
           errSignal++;
        }

        if (tempMut <= 0) {
           System.err.println("GA Error: tempMut must be >
0");
           errorToOutput.append("Mutation rate must be
greater than 0.\n");
           errSignal++;
        }


     } catch (NumberFormatException e) {
        System.err.println("Number Format: GA");
        errorToOutput.append("Please check GA
parameters.\n");
        errSignal++;
     } catch (NullPointerException e) {
        System.err.println("Null Pointer Exception: GA");
        errorToOutput.append("Please check GA
parameters.\n");
        errSignal++;
     } catch (Exception e) {
        System.err.println("Exception: GA");
        errorToOutput.append("Please check GA
parameters.\n");
        errSignal++;

     }

     if (errSignal == 0) {
        System.out.println("USER PARAMS OK");
        System.out.println("SETTING USER PARAMS");

        //dataset
        setFileName(trainDatasetName.getText());

setTrainRatio(Double.parseDouble(trainRatioBox.getText()
));

setValidRatio(Double.parseDouble(validRatioBox.getText()
));

        //nn

setNumNodes(Integer.parseInt(numOfHiddenNodesBox.ge
tText()));

setMaxError(Double.parseDouble(maxErrorBox.getText()))
;

setBreakIfTolerable(tolerableCheckBox.isSelected());

        //ga
        setPopSize(Integer.parseInt(popSizeBox.getText()));

setNumGen(Integer.parseInt(numOfGenBox.getText()));

setMutate(Double.parseDouble(mutateBox.getText()));
        hasError = false;

     }

     if (!hasError) {
        System.out.println("==================UI
PARAMETERS:");
        System.out.println("Filename: " + getFileName());
        System.out.println("Train Ratio: " + getTrainRatio());
        System.out.println("Valid Ratio: " + getValidRatio());
        System.out.println("Num Nodes: " +
getNumNodes());
        System.out.println("Max Error: " + getMaxError());
        System.out.println("Break if Tolerable: " +
isBreakIfTolerable());
        System.out.println("Pop Size: " + getPopSize());
        System.out.println("Num of Gen: " + getNumGen());
        System.out.println("Mutate Rate: " + getMutate());
        System.out.println();

        h = new Hybrid();
        h.setFileName(getFileName());
        h.setRatioTraining(getTrainRatio());
        h.setRatioValidating(getValidRatio());
        h.setNumOfHidden(getNumNodes());
        h.setMaxError(getMaxError());
        h.setPopulationSize(getPopSize());
        h.setMutate(getMutate());
        //h.setPatterns(getFileName());
        h.setEpochs(getNumGen());
        h.setBreakIfTolerable(isBreakIfTolerable());

        System.out.println("==================HYBRID
PARAMETERS:");
        System.out.println("Filename: " + h.getFileName());
        System.out.println("Train Ratio: " +
h.getRatioTraining());
```

```
        System.out.println("Valid Ratio: " +
h.getRatioValidating());
        System.out.println("Num Nodes: " +
h.getNumOfHidden());
        System.out.println("Max Error: " + h.getMaxError());
        System.out.println("Break: " +
h.isBreakIfTolerable());
        System.out.println("Pop Size: " +
h.getPopulationSize());
        System.out.println("Num of Gen: " + h.getEpochs());
        System.out.println("Mutate Rate: " +
h.getMutate());

        System.out.println();
        int confirm = JOptionPane.showConfirmDialog(this,
"Proceed with training?", "Confirm",
JOptionPane.YES_NO_OPTION,
JOptionPane.QUESTION_MESSAGE);
        if (confirm == JOptionPane.YES_OPTION) {
            h.createAndShowChart();
        }
      } else {
        System.err.println("STOPPED OPERATION. ERR'S: " +
errSignal + "\n");
        errorToOutput.append("\nPlease consult User
Manual for further instructions.\n");
        JOptionPane.showMessageDialog(this,
errorToOutput, "Error!", JOptionPane.ERROR_MESSAGE);
      }



    }//GEN-LAST:event_goButtonActionPerformed

    private void
diagBrowseButtonActionPerformed(java.awt.event.Action
Event evt) {//GEN-
FIRST:event_diagBrowseButtonActionPerformed
      File name = getFile();
      String path = name.getAbsolutePath();
      batchEntryBox.setText(path);
      if (name.isFile()) {   //file ok
//          diagnosisFileNotif.setText(name.getName());
//          diagnosisFileNotif.setVisible(true);
        String type =
name.getName().substring((name.getName().indexOf("."))
+ 1); //extract file type
        if (type.equals("csv") || type.equals("txt") ||
type.equals("dat")) {
        } else {
            JOptionPane.showMessageDialog(this, "Select a
.CSV, .DAT, or .TXT file.", "Dataset Error",
JOptionPane.ERROR_MESSAGE);
        }
      } else if (name.isDirectory()) {
        JOptionPane.showMessageDialog(this, "File error!
Please select a file.", "Dataset Error",
JOptionPane.ERROR_MESSAGE);
      }

    }//GEN-LAST:event_diagBrowseButtonActionPerformed

    private void
batchDiagButtonActionPerformed(java.awt.event.ActionEv
ent evt) {//GEN-
FIRST:event_batchDiagButtonActionPerformed
      //update result table
      try {
        if (null == batchEntryBox.getText() ||
batchEntryBox.getText().trim().length() == 0) {
            JOptionPane.showMessageDialog(this, "Please
check diagnosis entry.", "Diagnosis Error",
JOptionPane.ERROR_MESSAGE);
        } else {
            h.setToDiagFileName(batchEntryBox.getText());
            h.initBatchDiagnosis();
            System.out.println("======Fittest
Chromosome");
            NN nn = h.getOptimizedNN(); //get fittest
chromosome
            nn.printAllWeights();
            System.out.println("======SSE:" + 1 /
nn.fitness());

//nn.setBatchInputs(Hybrid.getNormed().normalizedDiagP
atterns);
            nn.diagnose();

            System.out.println("LOADING RESULTS ON
TABLE");
            h.showResultsOnTable();
          }

      } catch (Exception e) {
        JOptionPane.showMessageDialog(this, "Please
check diagnosis entry.", "Diagnosis Error",
JOptionPane.ERROR_MESSAGE);
      }
    }//GEN-LAST:event_batchDiagButtonActionPerformed

    private void
singleEntryDiagButtonActionPerformed(java.awt.event.Act
ionEvent evt) {//GEN-
FIRST:event_singleEntryDiagButtonActionPerformed
      try {
        h.setToSingleDiagnose(singleEntryBox.getText());
        h.initSingleDiagnosis();
        System.out.println("======Fittest Chromosome");
        NN nn = h.getOptimizedNN(); //get fittest
chromosome
        nn.printAllWeights();
        System.out.println("======SSE:" + 1 / nn.fitness());
        nn.diagnose();

        System.out.println("LOADING RESULTS ON TABLE");
        h.showResultsOnTable();
```

```
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, "Please
check diagnosis entry.", "Diagnosis Error",
JOptionPane.ERROR_MESSAGE);
        }
    }//GEN-
LAST:event_singleEntryDiagButtonActionPerformed

    private void
datasetHelpOnClick(java.awt.event.MouseEvent evt)
{//GEN-FIRST:event_datasetHelpOnClick
        JOptionPane.showMessageDialog(this, "Please select
a file for the training and the validation of the neural
network.\n"
                + "A training/validation data set template can be
downloaded from the application by clicking the download
button.\n\n"
                + "The training/validating ratios must both valued
between 0 and 1 and their sum must be equal to 1.\n"
                + "These will be used to partition the data set for
the training of the network; and for its validation,
afterwards.\n\n"
                + "Please consult User Manual for further
instructions.", "Information",
JOptionPane.INFORMATION_MESSAGE);
    }//GEN-LAST:event_datasetHelpOnClick

    private void
configHelpdatasetHelpOnClick(java.awt.event.MouseEvent
evt) {//GEN-FIRST:event_configHelpdatasetHelpOnClick
        JOptionPane.showMessageDialog(this, "Please
provide the configuration for the neural network and the
genetic algorithm.\n\n"
                + "Nueral Network Parameters:\n"
                + "The number of hidden nodes must be a non-
zero value.\n"
                + "This will be used to generate the intermediate
layer between the input and output layer.\n"
                + "The maximum tolerable error must be a non-
zero value.\n"
                + "This will be used to determine the validity and
fitness of the nueral network.\n"
                + "Note that, the training can be stopped if the
error of the network is found to be tolerable or
satisfactory.\n\n"
                + "Genetic Algorithm Parameters:\n"
                + "The population size must be a non-zero
value.\n"
                + "This will be used to create the population of
chromosomes.\n"
                + "The number of generations must be a non-zero
value.\n"
                + "This will determine the number of evolutions
that the population must go through.\n"
                + "The mutation rate must be valued between 0
and 1.\n"
                + "The mutation rate determines the probability a
member of the population will be mutated.\n"
                + "\nPlease consult User Manual for further
instructions.", "Information",
JOptionPane.INFORMATION_MESSAGE);
    }//GEN-LAST:event_configHelpdatasetHelpOnClick

    private void
trainResultHelpOnClick(java.awt.event.MouseEvent evt)
{//GEN-FIRST:event_trainResultHelpOnClick
        JOptionPane.showMessageDialog(this, "The network
will be trained using a ratio of the data set.\n\n"
                + "The sum of squares error will be used to
determine the fitness of the each generation.\n"
                + "The training may be ended when error of the
network is tolerable.\n"
                + "It may also be paused, and later on be
continued, or be stopped.\n\n"
                + "When the training ends, the fittest
chromosome will be used as the optimized weights for the
neural network.\n"
                + "The network will be validated, as well, using a
ratio of the data set.\n"
                + "\nPlease consult User Manual for further
instructions.", "Information",
JOptionPane.INFORMATION_MESSAGE);
    }//GEN-LAST:event_trainResultHelpOnClick

    private void
resetButtonActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_resetButtonActionPerformed
        int confirm = JOptionPane.showConfirmDialog(this,
"Are you sure you want to reset? All data will be lost.",
"Confirm", JOptionPane.YES_NO_OPTION,
JOptionPane.QUESTION_MESSAGE);
        if (confirm == JOptionPane.YES_OPTION) {
            this.dispose();
            Window[] w = this.getWindows();
            for (Window _w : w) {
                _w.dispose();
            }
            //confirm?
            //clear graph
            /*
             * Set<Thread> threadSet =
Thread.getAllStackTraces().keySet();
             * System.out.println("THREAD COUNT: " +
threadSet.size()); for
             * (Thread t : threadSet) { if (t.isAlive() &&
             * t.getName().equalsIgnoreCase("Thread-3")) {
t.interrupt();
             * System.out.println("STOPPING" + t.getName()); }
            }
             */

            Hybrid.main(new String[0]);
            Hybrid.ts = null;
        }
```

```
    }//GEN-LAST:event_resetButtonActionPerformed

    private void
summaryHelpOnClick(java.awt.event.MouseEvent evt)
{//GEN-FIRST:event_summaryHelpOnClick
        JOptionPane.showMessageDialog(this, "After training
the neural network, the summary will contain the
following:\n\n"
                + "SSE: The sum of squares error of the neural
network.\n"
                + "Time Elapsed: The total time of training in
HH:MM:SS.\n"
                + "Number of Generations: The number of
generations or epochs that the network underwent.\n"
                + "\nPlease consult User Manual for further
instructions.", "Information",
JOptionPane.INFORMATION_MESSAGE);

    }//GEN-LAST:event_summaryHelpOnClick

    private void
diagnosisHelpOnClick(java.awt.event.MouseEvent evt)
{//GEN-FIRST:event_diagnosisHelpOnClick
        JOptionPane.showMessageDialog(this, "The patient
classification can be either:\n\n"
                + "Batch Upload:\n"
                + "A file may be uploaded for processing.\n"
                + "A diagnosis data set template can be
downloaded from the application by clicking the download
button.\n\n"
                + "Single Entry:\n"
                + "A single may be uploaded for processing.\n"
                + "Please provide an entry with format:\n"
                + "Temperature;Platelet Count;Cardiac
Rate;Blood Pressure[SP];Blood Pressure[DP];Respiratory
Rate;White Blood Cell Count; Red Blood Cell Count\n\n"
                + "\nPlease consult User Manual for further
instructions.", "Information",
JOptionPane.INFORMATION_MESSAGE);

    }//GEN-LAST:event_diagnosisHelpOnClick

    private void
diagResultsHelpOnClick(java.awt.event.MouseEvent evt)
{//GEN-FIRST:event_diagResultsHelpOnClick
        JOptionPane.showMessageDialog(this, "The result/s
of the diagnosis is shown through the table.\n\n"
                + "It contains the value of the symptoms and the
diagnosis.\n"
                + "The result/s can also be exported or be copied
from the table.\n\n"
                + "\nPlease consult User Manual for further
instructions.", "Information",
JOptionPane.INFORMATION_MESSAGE);

    }//GEN-LAST:event_diagResultsHelpOnClick

    private void
exportButtonOnClick(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_exportButtonOnClick
        File toExcel = saveFile();
        if (null != toExcel) {
            try {
                //init objects
                TableModel model = diagTable.getModel();
                FileWriter excel = new FileWriter(toExcel);

                //write column headers
                for (int i = 0; i < model.getColumnCount(); i++) {
                    excel.write(model.getColumnName(i) + "\t");

                }
                excel.write("\n");

                //write data
                for (int i = 0; i < model.getRowCount(); i++) {
                    for (int j = 0; j < model.getColumnCount(); j++) {
                        excel.write(model.getValueAt(i, j) == null ? ""
: model.getValueAt(i, j).toString() + "\t");
                        //excel.write(model.getValueAt(i, j).toString()
+ "\t");
                    }
                    excel.write("\n");

                }
                excel.close();
                JOptionPane.showMessageDialog(this, "Table
data saved!", "File Export",
JOptionPane.INFORMATION_MESSAGE);
            } catch (FileNotFoundException e) {
                JOptionPane.showMessageDialog(this, "Please
check file entry.", "File Error",
JOptionPane.ERROR_MESSAGE);
            } catch (Exception e) {
                JOptionPane.showMessageDialog(this, "An error
has occured.", "Error", JOptionPane.ERROR_MESSAGE);
            }
        }

    }//GEN-LAST:event_exportButtonOnClick

    private void
downloadDiagTemplateOnClick(java.awt.event.MouseEve
nt evt) {//GEN-
FIRST:event_downloadDiagTemplateOnClick
        InputStream configStream =
getClass().getResourceAsStream("DataForBatchDiagTempl
ate.txt");
        BufferedReader configReader = null;
        try {
            configReader = new BufferedReader(new
InputStreamReader(configStream, "UTF-8"));
        } catch (UnsupportedEncodingException e) {
```

```
            JOptionPane.showMessageDialog(this, "Something
went wrong!", "File Error",
JOptionPane.ERROR_MESSAGE);
        }

        File userFile = saveTemplate(TEMPLATE_DIAG);
        if (null != configReader && null != userFile) {
            try {
                String sCurrentLine = "";
                FileWriter writer = new FileWriter(userFile);
                while ((sCurrentLine = configReader.readLine()) !=
null) {
                    writer.write(sCurrentLine);
                    writer.write("\n");
                }

                configReader.close();
                writer.close();
                JOptionPane.showMessageDialog(this, "Diagnosis
data set template saved!", "File Export",
JOptionPane.INFORMATION_MESSAGE);
            } catch (FileNotFoundException e) {
                JOptionPane.showMessageDialog(this, "File not
found!", "File Error", JOptionPane.ERROR_MESSAGE);
            } catch (IOException e) {
                JOptionPane.showMessageDialog(this, "Please
check file entry.", "File Error",
JOptionPane.ERROR_MESSAGE);
            } catch (Exception e) {
                JOptionPane.showMessageDialog(this, "An error
has occured.", "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    }//GEN-LAST:event_downloadDiagTemplateOnClick

    private void
downloadTrainValTemplateOnClick(java.awt.event.Mouse
Event evt) {//GEN-
FIRST:event_downloadTrainValTemplateOnClick
        InputStream configStream =
getClass().getResourceAsStream("DataForTrainingValidatin
gTemplate.txt");
        BufferedReader configReader = null;
        try {
            configReader = new BufferedReader(new
InputStreamReader(configStream, "UTF-8"));
        } catch (UnsupportedEncodingException e) {
            JOptionPane.showMessageDialog(this, "Something
went wrong!", "File Error",
JOptionPane.ERROR_MESSAGE);
        }
        File userFile = saveTemplate(TEMPLATE_TV);
        if (null != configReader && null != userFile) {
            try {
                String sCurrentLine = "";
                FileWriter writer = new FileWriter(userFile);
                while ((sCurrentLine = configReader.readLine()) !=
null) {
                    writer.write(sCurrentLine);
                    writer.write("\n");
                }

                configReader.close();
                writer.close();
                JOptionPane.showMessageDialog(this,
"Training/validating data set template saved!", "File
Export", JOptionPane.INFORMATION_MESSAGE);
            } catch (FileNotFoundException e) {
                JOptionPane.showMessageDialog(this, "File not
found!", "File Error", JOptionPane.ERROR_MESSAGE);
            } catch (IOException e) {
                JOptionPane.showMessageDialog(this, "Please
check file entry.", "File Error",
JOptionPane.ERROR_MESSAGE);
            } catch (Exception e) {
                JOptionPane.showMessageDialog(this, "An error
has occured.", "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    }//GEN-LAST:event_downloadTrainValTemplateOnClick

    private File getFile() {

FileChooser.setFileSelectionMode(JFileChooser.FILES_AND
_DIRECTORIES);
        FileChooser.setDialogTitle("Upload Dataset");
        int result = FileChooser.showOpenDialog(this);
        if (result == JFileChooser.CANCEL_OPTION) {

        }
        File fileName = FileChooser.getSelectedFile();
//      if ((fileName == null) ||
(fileName.getName().equals(""))) {
//          JOptionPane.showMessageDialog(this, "Invalid
File Name", "Invalid File Name",
JOptionPane.ERROR_MESSAGE);
//      } // end if
        return fileName;
    }

    private File saveTemplate(String type) {
        String filename = "";
        if (type.equalsIgnoreCase("tv")) {
            filename =
"DataForTrainingValidatingTemplate.txt";
        } else if (type.equalsIgnoreCase("diag")) {
            filename = "DataForBatchDiagTemplate.txt";
        }

        FileChooser = new JFileChooser();

FileChooser.setFileSelectionMode(JFileChooser.FILES_AND
_DIRECTORIES);

FileChooser.setDialogTitle(type.equalsIgnoreCase("tv") ?
```

```
"Download Training/Validating Template" : "Download
Diagnosis Template");
      FileChooser.setSelectedFile(new
File(System.getProperty("user.home") + "\\" + filename));
      int result = FileChooser.showSaveDialog(this);
      if (result == JFileChooser.CANCEL_OPTION) {
         //System.exit(1);
      }
      if (result == JFileChooser.APPROVE_OPTION) {
         String path =
FileChooser.getSelectedFile().getParentFile().getPath();

         int len = filename.length();
         String ext = "";
         String file;

         if (len > 4) {
            ext =
filename.substring(filename.lastIndexOf("."), len);
         }

         if (ext.equalsIgnoreCase(".xlsx") ||
ext.equalsIgnoreCase(".xls")) {
            file = path + "\\" + filename;
         } else if (ext.equalsIgnoreCase(".csv")) {
            file = path + "\\" + filename;
         } else if (ext.equalsIgnoreCase(".txt")) {
            file = path + "\\" + filename;
         } else { //txt default
            file = path + "\\" + filename + ".txt";
         }

         System.out.println("SAVING TEMPLATE: " + file);

         File testExists = new File(file);
         if (!testExists.exists() || okToReplace(testExists)) {
            return testExists;
         }
      }
      return null;

   }

   private File saveFile() {
      FileChooser = new JFileChooser();

FileChooser.setFileSelectionMode(JFileChooser.FILES_AND
_DIRECTORIES);
      FileChooser.setDialogTitle("Export Table");

      int result = FileChooser.showSaveDialog(this);

      if (result == JFileChooser.APPROVE_OPTION) {
         String filename =
FileChooser.getSelectedFile().getName();
         String path =
FileChooser.getSelectedFile().getParentFile().getPath();

         int len = filename.length();
         String ext = "";
         String file = "";

         if (len > 4) {
            ext =
filename.substring(filename.lastIndexOf("."), len);
         }

         if (ext.equalsIgnoreCase(".xlsx") ||
ext.equalsIgnoreCase(".xls")) {
            file = path + "\\" + filename;
         } else if (ext.equalsIgnoreCase(".csv")) {
            file = path + "\\" + filename;
         } else if (ext.equalsIgnoreCase(".txt")) {
            file = path + "\\" + filename;
         } else {
            file = path + "\\" + filename + ".xls";
         }

         File testExists = new File(file);
         if (!testExists.exists() || okToReplace(testExists)) {
            System.out.println("SAVING TABLE: " + file);
            return testExists;
         }

      }
      return null;
   }

   private boolean okToReplace(File f) {
      final Object[] options = {"Yes", "No", "Cancel"};
      return JOptionPane.showOptionDialog(this,
            "The file '" + f.getName()
            + "' already exists.  "
            + "Replace existing file?",
            "Warning",
            JOptionPane.YES_NO_CANCEL_OPTION,
            JOptionPane.WARNING_MESSAGE,
            null,
            options,
            options[2]) == JOptionPane.YES_OPTION;
   }



   /**
    *
    * @return User specified training ratio
    */
   public double getTrainRatio() {
      return getUserTrainRatio();
   }

   /**
    *
    * @param tr User specified training ratio
    */
```

```
public void setTrainRatio(double tr) {
   setUserTrainRatio(tr);
}

/**
 *
 * @return User specified training ratio
 */
public double getValidRatio() {
   return getUserValidRatio();
}

/**
 *
 * @param vr User specified validating ratio
 */
public void setValidRatio(double vr) {
   setUserValidRatio(vr);
}

/**
 *
 * @return User specified file name
 */
public String getFileName() {
   return getUserFileName();
}

/**
 *
 * @param fn User specified file name
 */
public void setFileName(String fn) {
   setUserFileName(fn);
}

/**
 *
 * @return User specified number of nodes
 */
public int getNumNodes() {
   return getUserNumNodes();
}

/**
 *
 * @param nm User specified number of nodes
 */
public void setNumNodes(int nm) {
   setUserNumNodes(nm);
}

/**
 *
 * @param me User specified maximum error
 */
public void setMaxError(double me) {
   setUserMaxError(me);
}

/**
 *
 * @return User specified maximum error
 */
public double getMaxError() {
   return getUserMaxError();
}

/**
 *
 * @param ps User specified population size
 */
public void setPopSize(int ps) {
   setUserPopSize(ps);
}

/**
 *
 * @return User specified population size
 */
public int getPopSize() {
   return getUserPopSize();
}

/**
 *
 * @param ng User specified number of generations
 */
public void setNumGen(int ng) {
   setUserNumGen(ng);
}

/**
 *
 * @return User specified number of generations
 */
public int getNumGen() {
   return getUserNumGen();
}

/**
 *
 * @param mr User specified mutation rate
 */
public void setMutate(double mr) {
   setUserMutate(mr);
}

/**
 *
 * @return User specified mutation rate
 */
public double getMutate() {
   return getUserMutate();
}
```

```
  /**
   * @return Flag for break if tolerable
   */
  public boolean isBreakIfTolerable() {
    return breakIfTolerable;
  }

  /**
   * @param breakIfTolerable flag for break if tolerable
   */
  public void setBreakIfTolerable(boolean
breakIfTolerable) {
    this.breakIfTolerable = breakIfTolerable;
  }

  /**
   * @return User specified file name
   */
  public String getUserFileName() {
    return userFileName;
  }

  /**
   * @param userFileName User specified file name
   */
  public void setUserFileName(String userFileName) {
    this.userFileName = userFileName;
  }

  /**
   * @return User specified training ratio
   *
   */
  public double getUserTrainRatio() {
    return userTrainRatio;
  }

  /**
   * @param userTrainRatio User specified training ratio
   *
   */
  public void setUserTrainRatio(double userTrainRatio) {
    this.userTrainRatio = userTrainRatio;
  }

  /**
   * @return User specified validating ratio
   */
  public double getUserValidRatio() {
    return userValidRatio;
  }

  /**
   * @param userValidRatio User specified validating ratio
   */
  public void setUserValidRatio(double userValidRatio) {
    this.userValidRatio = userValidRatio;
  }

  /**
   * @return User specified number of nodes
   */
  public int getUserNumNodes() {
    return userNumNodes;
  }

  /**
   * @param userNumNodes User specified number of
nodes
   */
  public void setUserNumNodes(int userNumNodes) {
    this.userNumNodes = userNumNodes;
  }

  /**
   * @return User specified maximum error
   */
  public double getUserMaxError() {
    return userMaxError;
  }

  /**
   * @param userMaxError User specified maximum error
   */
  public void setUserMaxError(double userMaxError) {
    this.userMaxError = userMaxError;
  }

  /**
   * @return User specified population size
   */
  public int getUserPopSize() {
    return userPopSize;
  }

  /**
   * @param userPopSize User specified population size
   */
  public void setUserPopSize(int userPopSize) {
    this.userPopSize = userPopSize;
  }

  /**
   * @return User specified number of generation
   */
  public int getUserNumGen() {
    return userNumGen;
  }

  /**
   * @param userNumGen User specified number of
generation
   */
  public void setUserNumGen(int userNumGen) {
    this.userNumGen = userNumGen;
  }
```

```
    /**
     * @return User specified mutation rate
     */
    public double getUserMutate() {
        return userMutate;
    }

    /**
     * @param userMutate User specified mutation rate
     */
    public void setUserMutate(double userMutate) {
        this.userMutate = userMutate;
    }

    //dataset
    private String userFileName;
    private String userToDiagFileName;
    private double userTrainRatio;
    private double userValidRatio;
    //nnparams
    private int userNumNodes;
    private double userMaxError;
    private boolean breakIfTolerable;
    //gaparams
    private int userPopSize;
    private int userNumGen;
    private double userMutate;
    private Hybrid h;
    //files
    private final String TEMPLATE_TV = "tv";
    private final String TEMPLATE_DIAG = "diag";
    // Variables declaration - do not modify//GEN-
BEGIN:variables
    private javax.swing.JFileChooser FileChooser;
    public static javax.swing.JButton batchDiagButton;
    public javax.swing.JTextField batchEntryBox;
    private javax.swing.JLabel configHelp;
    private javax.swing.JPanel configurationPanel;
    public javax.swing.JButton diagBrowseButton;
    protected static javax.swing.JTable diagTable;
    private javax.swing.JLabel diagnosisHelp;
    public javax.swing.JPanel diagnosisPanel;
    private javax.swing.JLabel dlTvTemplate;
    private javax.swing.JLabel downloadDiagTemplate;
    private javax.swing.JLabel downloadTrainValTemplate;
    private javax.swing.JLabel fileNotif;
    private javax.swing.JLabel fileNotif1;
    public static javax.swing.JLabel finalSSELabel;
    public static javax.swing.JButton goButton;
    private javax.swing.JButton jButton1;
    private javax.swing.JDialog jDialog2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel10;
    private javax.swing.JLabel jLabel11;
    private javax.swing.JLabel jLabel12;
    private javax.swing.JLabel jLabel13;
    private javax.swing.JLabel jLabel15;
    private javax.swing.JLabel jLabel16;
    private javax.swing.JLabel jLabel17;
    private javax.swing.JLabel jLabel18;
    private javax.swing.JLabel jLabel19;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel20;
    private javax.swing.JLabel jLabel21;
    private javax.swing.JLabel jLabel22;
    private javax.swing.JLabel jLabel23;
    private javax.swing.JLabel jLabel24;
    private javax.swing.JLabel jLabel25;
    private javax.swing.JLabel jLabel27;
    private javax.swing.JLabel jLabel28;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JSeparator jSeparator2;
    private javax.swing.JSeparator jSeparator3;
    private javax.swing.JSeparator jSeparator4;
    private javax.swing.JSeparator jSeparator5;
    private javax.swing.JSeparator jSeparator6;
    private javax.swing.JTextField maxErrorBox;
    private javax.swing.JTextField mutateBox;
    public static javax.swing.JLabel numGenLabel;
    private javax.swing.JTextField numOfGenBox;
    private javax.swing.JTextField numOfHiddenNodesBox;
    private javax.swing.JTextField popSizeBox;
    public static javax.swing.JButton resetButton;
    private javax.swing.JPanel resultHelp;
    private javax.swing.JLabel resultsHelp;
    public javax.swing.JTextField singleEntryBox;
    public javax.swing.JButton singleEntryDiagButton;
    private javax.swing.JLabel summaryHelp;
    private javax.swing.JPanel summaryPanel;
    public javax.swing.JLabel timeElapsedLabel;
    private javax.swing.JCheckBox tolerableCheckBox;
    private javax.swing.JButton trainBrowseButton;
    private javax.swing.JTextField trainDatasetName;
    private javax.swing.JPanel trainDatasetPanel;
    private javax.swing.JTextField trainRatioBox;
    private javax.swing.JLabel trainResultHelp;
    public static javax.swing.JTextArea
trainingResultTextArea;
    private javax.swing.JTextField validRatioBox;
    // End of variables declaration//GEN-END:variables


}
```

UI.form

```
<?xml version="1.0" encoding="UTF-8" ?>

<Form version="1.5" maxVersion="1.8"
type="org.netbeans.modules.form.forminfo.JFrameFormInfo">
  <NonVisualComponents>
    <Component class="javax.swing.JFileChooser"
name="FileChooser">
      <Properties>
        <Property name="dialogTitle" type="java.lang.String"
value="Upload Data Set"/>
      </Properties>
    </Component>
    <Container class="javax.swing.JDialog"
name="jDialog2">
      <Properties>
        <Property name="minimumSize"
type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
          <Dimension value="[100, 100]"/>
        </Property>
      </Properties>

      <Layout>
        <DimensionLayout dim="0">
          <Group type="103" groupAlignment="0"
attributes="0">
            <EmptySpace min="0" pref="400" max="32767"
attributes="0"/>
          </Group>
        </DimensionLayout>
        <DimensionLayout dim="1">
          <Group type="103" groupAlignment="0"
attributes="0">
            <EmptySpace min="0" pref="300" max="32767"
attributes="0"/>
          </Group>
        </DimensionLayout>
      </Layout>
    </Container>
    <Component class="javax.swing.JLabel"
name="downloadTrainValTemplate">
      <Properties>
        <Property name="icon" type="javax.swing.Icon"
editor="org.netbeans.modules.form.editors2.IconEditor">
          <Image iconType="3" name="/dl.png"/>
        </Property>
        <Property name="toolTipText" type="java.lang.String"
value="Click to download template"/>
        <Property name="cursor" type="java.awt.Cursor"
editor="org.netbeans.modules.form.editors2.CursorEditor
">
          <Color id="Hand Cursor"/>
        </Property>
      </Properties>
      <Events>
        <EventHandler event="mousePressed"
listener="java.awt.event.MouseListener"
parameters="java.awt.event.MouseEvent"
handler="downloadTrainValTemplateOnClick"/>
      </Events>
    </Component>
  </NonVisualComponents>
  <Properties>
    <Property name="defaultCloseOperation" type="int"
value="3"/>
    <Property name="bounds" type="java.awt.Rectangle"
editor="org.netbeans.beaninfo.editors.RectangleEditor">
      <Rectangle value="[0, 0, 0, 0]"/>
    </Property>
    <Property name="resizable" type="boolean"
value="false"/>
  </Properties>
  <SyntheticProperties>
    <SyntheticProperty name="formSizePolicy" type="int"
value="1"/>
  </SyntheticProperties>
  <AuxValues>
    <AuxValue name="FormSettings_autoResourcing"
type="java.lang.Integer" value="0"/>
    <AuxValue
name="FormSettings_autoSetComponentName"
type="java.lang.Boolean" value="false"/>
    <AuxValue name="FormSettings_generateFQN"
type="java.lang.Boolean" value="true"/>
    <AuxValue
name="FormSettings_generateMnemonicsCode"
type="java.lang.Boolean" value="false"/>
    <AuxValue name="FormSettings_i18nAutoMode"
type="java.lang.Boolean" value="false"/>
    <AuxValue name="FormSettings_layoutCodeTarget"
type="java.lang.Integer" value="1"/>
    <AuxValue
name="FormSettings_listenerGenerationStyle"
type="java.lang.Integer" value="0"/>
    <AuxValue name="FormSettings_variablesLocal"
type="java.lang.Boolean" value="false"/>
    <AuxValue name="FormSettings_variablesModifier"
type="java.lang.Integer" value="2"/>
  </AuxValues>

  <Layout>
    <DimensionLayout dim="0">
      <Group type="103" groupAlignment="0"
attributes="0">
        <Group type="102" attributes="0">
          <Component id="jPanel1" min="-2" pref="442"
max="-2" attributes="0"/>
          <EmptySpace max="-2" attributes="0"/>
          <Component id="jPanel2" min="-2" pref="473"
max="-2" attributes="0"/>
```

```
        <EmptySpace pref="19" max="32767"
attributes="0"/>
       </Group>
    </Group>
   </DimensionLayout>
   <DimensionLayout dim="1">
    <Group type="103" groupAlignment="0"
attributes="0">
       <Group type="102" attributes="0">
         <Component id="jPanel1" min="-2" pref="750"
max="-2" attributes="0"/>
         <EmptySpace min="0" pref="72" max="32767"
attributes="0"/>
       </Group>
       <Component id="jPanel2" alignment="1"
max="32767" attributes="0"/>
    </Group>
   </DimensionLayout>
  </Layout>
  <SubComponents>
   <Container class="javax.swing.JPanel" name="jPanel1">
    <Properties>
     <Property name="enabled" type="boolean"
value="false"/>
    </Properties>

    <Layout>
     <DimensionLayout dim="0">
      <Group type="103" groupAlignment="0"
attributes="0">
         <Group type="102" alignment="1" attributes="0">
           <EmptySpace max="-2" attributes="0"/>
           <Group type="103" groupAlignment="1"
attributes="0">
             <Group type="102" attributes="0">
               <EmptySpace min="0" pref="0"
max="32767" attributes="0"/>
               <Component id="jLabel23" min="-2" max="-
2" attributes="0"/>
               <EmptySpace min="-2" pref="18" max="-2"
attributes="0"/>
               <Component id="jLabel24" min="-2"
pref="412" max="-2" attributes="0"/>
             </Group>
             <Group type="102" attributes="0">
               <Component id="jSeparator2" max="32767"
attributes="0"/>
               <EmptySpace max="-2" attributes="0"/>
               <Group type="103" groupAlignment="0"
attributes="0">
                 <Component id="configurationPanel"
alignment="1" max="32767" attributes="0"/>
                 <Component id="jSeparator6"
max="32767" attributes="0"/>
                 <Group type="102" alignment="0"
attributes="0">
                   <Component id="trainDatasetPanel"
min="-2" max="-2" attributes="0"/>
                   <EmptySpace min="0" pref="0"
max="32767" attributes="0"/>
                 </Group>
               </Group>
             </Group>
             <EmptySpace max="-2" attributes="0"/>
           </Group>
         </Group>
       </DimensionLayout>
       <DimensionLayout dim="1">
        <Group type="103" groupAlignment="0"
attributes="0">
          <Group type="102" alignment="0" attributes="0">
            <Group type="103" groupAlignment="1"
attributes="0">
              <Group type="102" alignment="1"
attributes="0">
                <EmptySpace min="-2" pref="160" max="-
2" attributes="0"/>
                <Component id="jLabel23" min="-2" max="-
2" attributes="0"/>
                <EmptySpace min="0" pref="0"
max="32767" attributes="0"/>
                <Component id="jSeparator2" min="-2"
max="-2" attributes="0"/>
                <EmptySpace min="-2" pref="12" max="-2"
attributes="0"/>
              </Group>
              <Group type="102" alignment="1"
attributes="0">
                <EmptySpace max="32767"
attributes="0"/>
                <Component id="jLabel24" min="-2"
pref="214" max="-2" attributes="0"/>
                <EmptySpace max="-2" attributes="0"/>
                <Component id="trainDatasetPanel" min="-
2" max="-2" attributes="0"/>
                <EmptySpace max="-2" attributes="0"/>
              </Group>
            </Group>
            <Component id="jSeparator6" min="-2" max="-
2" attributes="0"/>
            <EmptySpace max="-2" attributes="0"/>
            <Component id="configurationPanel" min="-2"
max="-2" attributes="0"/>
            <EmptySpace min="-2" pref="401" max="-2"
attributes="0"/>
          </Group>
        </Group>
       </DimensionLayout>
     </Layout>
     <SubComponents>
      <Component class="javax.swing.JLabel"
name="jLabel23">
      </Component>
      <Component class="javax.swing.JLabel"
name="jLabel24">
```

```
      <Properties>
        <Property name="icon" type="javax.swing.Icon"
editor="org.netbeans.modules.form.editors2.IconEditor">
          <Image iconType="3"
name="/gnatlogosmall.png"/>
        </Property>
      </Properties>
      <AccessibilityProperties>
        <Property
name="AccessibleContext.accessibleName"
type="java.lang.String" value="DNLogo"/>
      </AccessibilityProperties>
    </Component>
    <Container class="javax.swing.JPanel"
name="configurationPanel">

      <Layout>
        <DimensionLayout dim="0">
          <Group type="103" groupAlignment="0"
attributes="0">
            <Group type="102" attributes="0">
              <EmptySpace max="-2" attributes="0"/>
              <Group type="103" groupAlignment="0"
attributes="0">
                <Group type="102" alignment="0"
attributes="0">
                  <Group type="103" groupAlignment="0"
attributes="0">
                    <Component id="jLabel8"
alignment="0" min="-2" max="-2" attributes="0"/>
                    <Group type="102" alignment="0"
attributes="0">
                      <EmptySpace min="10" pref="10"
max="-2" attributes="0"/>
                      <Group type="103"
groupAlignment="0" attributes="0">
                        <Component id="jLabel12"
alignment="0" min="-2" max="-2" attributes="0"/>
                        <Component id="jLabel13"
alignment="0" min="-2" max="-2" attributes="0"/>
                        <Component id="jLabel11"
alignment="0" min="-2" max="-2" attributes="0"/>
                      </Group>
                      <EmptySpace min="-2" pref="26"
max="-2" attributes="0"/>
                      <Group type="103"
groupAlignment="1" max="-2" attributes="0">
                        <Component id="numOfGenBox"
alignment="0" max="32767" attributes="0"/>
                        <Component id="mutateBox"
alignment="0" max="32767" attributes="0"/>
                        <Component id="popSizeBox"
alignment="0" min="-2" pref="172" max="-2"
attributes="0"/>
                      </Group>
                    </Group>
                  </Group>
                  <EmptySpace min="0" pref="0"
max="32767" attributes="0"/>
                </Group>
                <Group type="102" attributes="0">
                  <Group type="103" groupAlignment="0"
attributes="0">
                    <Component id="jLabel7"
alignment="0" min="-2" max="-2" attributes="0"/>
                    <Group type="102" alignment="0"
attributes="0">
                      <EmptySpace min="10" pref="10"
max="-2" attributes="0"/>
                      <Group type="103"
groupAlignment="0" attributes="0">
                        <Component id="jLabel9"
alignment="0" min="-2" max="-2" attributes="0"/>
                        <Component id="jLabel10"
alignment="0" min="-2" max="-2" attributes="0"/>
                      </Group>
                      <EmptySpace type="separate"
max="-2" attributes="0"/>
                      <Group type="103"
groupAlignment="0" attributes="0">
                        <Component
id="tolerableCheckBox" min="-2" max="-2"
attributes="0"/>
                        <Group type="103" alignment="0"
groupAlignment="1" max="-2" attributes="0">
                          <Component id="maxErrorBox"
alignment="0" pref="173" max="32767" attributes="0"/>
                          <Component
id="numOfHiddenNodesBox" alignment="0" max="32767"
attributes="0"/>
                        </Group>
                      </Group>
                    </Group>
                    <EmptySpace max="32767"
attributes="0"/>
                  </Group>
                  <Group type="102" alignment="1"
attributes="0">
                    <EmptySpace min="0" pref="0"
max="32767" attributes="0"/>
                    <Component id="resetButton" min="-2"
pref="69" max="-2" attributes="0"/>
                    <EmptySpace max="-2" attributes="0"/>
                    <Component id="goButton" min="-2"
max="-2" attributes="0"/>
                    <EmptySpace min="-2" pref="4" max="-2"
attributes="0"/>
                  </Group>
                  <Group type="102" attributes="0">
                    <Component id="jLabel1" min="-2"
max="-2" attributes="0"/>
                    <EmptySpace max="32767"
attributes="0"/>
```

```
                    <Component id="configHelp" min="-2"
max="-2" attributes="0"/>
                  </Group>
                </Group>
              </Group>
            </Group>
          </DimensionLayout>
          <DimensionLayout dim="1">
            <Group type="103" groupAlignment="0"
attributes="0">
              <Group type="102" alignment="0"
attributes="0">
                <Group type="103" groupAlignment="0"
attributes="0">
                  <Component id="jLabel1" min="-2" max="-
2" attributes="0"/>
                  <Component id="configHelp" min="-2"
max="-2" attributes="0"/>
                </Group>
                <EmptySpace type="unrelated" max="-2"
attributes="0"/>
                <Component id="jLabel7" min="-2" max="-2"
attributes="0"/>
                <EmptySpace max="-2" attributes="0"/>
                <Group type="103" groupAlignment="3"
attributes="0">
                  <Component id="jLabel9" alignment="3"
min="-2" max="-2" attributes="0"/>
                  <Component id="numOfHiddenNodesBox"
alignment="3" min="-2" max="-2" attributes="0"/>
                </Group>
                <EmptySpace min="-2" pref="6" max="-2"
attributes="0"/>
                <Group type="103" groupAlignment="3"
attributes="0">
                  <Component id="jLabel10" alignment="3"
min="-2" max="-2" attributes="0"/>
                  <Component id="maxErrorBox"
alignment="3" min="-2" max="-2" attributes="0"/>
                </Group>
                <EmptySpace type="unrelated" max="-2"
attributes="0"/>
                <Component id="tolerableCheckBox" min="-
2" max="-2" attributes="0"/>
                <EmptySpace min="-2" pref="29" max="-2"
attributes="0"/>
                <Component id="jLabel8" min="-2" max="-2"
attributes="0"/>
                <EmptySpace min="-2" pref="10" max="-2"
attributes="0"/>
                <Group type="103" groupAlignment="3"
attributes="0">
                  <Component id="popSizeBox"
alignment="3" min="-2" max="-2" attributes="0"/>
                  <Component id="jLabel11" alignment="3"
min="-2" max="-2" attributes="0"/>
                </Group>
                <EmptySpace max="-2" attributes="0"/>
                <Group type="103" groupAlignment="3"
attributes="0">
                  <Component id="numOfGenBox"
alignment="3" min="-2" max="-2" attributes="0"/>
                  <Component id="jLabel12" alignment="3"
min="-2" max="-2" attributes="0"/>
                </Group>
                <EmptySpace max="-2" attributes="0"/>
                <Group type="103" groupAlignment="3"
attributes="0">
                  <Component id="mutateBox"
alignment="3" min="-2" max="-2" attributes="0"/>
                  <Component id="jLabel13" alignment="3"
min="-2" max="-2" attributes="0"/>
                </Group>
                <EmptySpace min="-2" pref="29" max="-2"
attributes="0"/>
                <Group type="103" groupAlignment="3"
attributes="0">
                  <Component id="resetButton"
alignment="3" min="-2" max="-2" attributes="0"/>
                  <Component id="goButton" alignment="3"
min="-2" max="-2" attributes="0"/>
                </Group>
                <EmptySpace pref="88" max="32767"
attributes="0"/>
              </Group>
            </Group>
          </DimensionLayout>
        </Layout>
        <SubComponents>
          <Component class="javax.swing.JLabel"
name="jLabel11">
            <Properties>
              <Property name="text" type="java.lang.String"
value="Population Size:"/>
            </Properties>
          </Component>
          <Component class="javax.swing.JTextField"
name="maxErrorBox">
            <Properties>
              <Property name="toolTipText"
type="java.lang.String" value="Please provide a non-zero
value"/>
            </Properties>
          </Component>
          <Component class="javax.swing.JTextField"
name="numOfHiddenNodesBox">
            <Properties>
              <Property name="toolTipText"
type="java.lang.String" value="Please provide a non-zero
value"/>
            </Properties>
          </Component>
          <Component class="javax.swing.JTextField"
name="popSizeBox">
            <Properties>
```

```
          <Property name="toolTipText"
type="java.lang.String" value="Please provide a non-zero
value"/>
        </Properties>
      </Component>
      <Component class="javax.swing.JLabel"
name="jLabel1">
        <Properties>
          <Property name="font" type="java.awt.Font"
editor="org.netbeans.beaninfo.editors.FontEditor">
            <Font name="Calibri" size="18" style="1"/>
          </Property>
          <Property name="text" type="java.lang.String"
value="Configuration"/>
        </Properties>
      </Component>
      <Component class="javax.swing.JTextField"
name="numOfGenBox">
        <Properties>
          <Property name="toolTipText"
type="java.lang.String" value="Please provide a non-zero
value"/>
        </Properties>
      </Component>
      <Component class="javax.swing.JLabel"
name="jLabel12">
        <Properties>
          <Property name="text" type="java.lang.String"
value="Number of Generations:"/>
        </Properties>
      </Component>
      <Component class="javax.swing.JLabel"
name="jLabel13">
        <Properties>
          <Property name="text" type="java.lang.String"
value="Mutation Rate:"/>
        </Properties>
      </Component>
      <Component class="javax.swing.JTextField"
name="mutateBox">
        <Properties>
          <Property name="toolTipText"
type="java.lang.String" value="Please provide a value from
0 - 1"/>
        </Properties>
      </Component>
      <Component class="javax.swing.JCheckBox"
name="tolerableCheckBox">
        <Properties>
          <Property name="text" type="java.lang.String"
value="Stop training when error is tolerable?"/>
          <Property name="toolTipText"
type="java.lang.String" value=""/>
        </Properties>
      </Component>
      <Component class="javax.swing.JLabel"
name="jLabel7">
        <Properties>
          <Property name="font" type="java.awt.Font"
editor="org.netbeans.beaninfo.editors.FontEditor">
            <Font name="Calibri" size="14" style="1"/>
          </Property>
          <Property name="text" type="java.lang.String"
value="Neural Network Parameters"/>
        </Properties>
      </Component>
      <Component class="javax.swing.JLabel"
name="jLabel8">
        <Properties>
          <Property name="font" type="java.awt.Font"
editor="org.netbeans.beaninfo.editors.FontEditor">
            <Font name="Calibri" size="14" style="1"/>
          </Property>
          <Property name="text" type="java.lang.String"
value="Genetic Algorithm Parameters"/>
        </Properties>
      </Component>
      <Component class="javax.swing.JLabel"
name="jLabel9">
        <Properties>
          <Property name="text" type="java.lang.String"
value="Number of Hidden Nodes:"/>
        </Properties>
      </Component>
      <Component class="javax.swing.JLabel"
name="jLabel10">
        <Properties>
          <Property name="text" type="java.lang.String"
value="Max Tolerable Error:"/>
        </Properties>
      </Component>
      <Component class="javax.swing.JButton"
name="goButton">
        <Properties>
          <Property name="text" type="java.lang.String"
value="GO&amp;gt;&amp;gt;&amp;gt;"/>
          <Property name="toolTipText"
type="java.lang.String" value="Click to proceed with
training"/>
        </Properties>
        <Events>
          <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener"
parameters="java.awt.event.ActionEvent"
handler="goButtonActionPerformed"/>
        </Events>
        <AuxValues>
          <AuxValue
name="JavaCodeGenerator_VariableModifier"
type="java.lang.Integer" value="9"/>
        </AuxValues>
      </Component>
      <Component class="javax.swing.JLabel"
name="configHelp">
        <Properties>
```

```
          <Property name="icon" type="javax.swing.Icon"
editor="org.netbeans.modules.form.editors2.IconEditor">
            <Image iconType="3" name="/qMark.png"/>
          </Property>
          <Property name="toolTipText"
type="java.lang.String" value="Click to access help"/>
          <Property name="cursor" type="java.awt.Cursor"
editor="org.netbeans.modules.form.editors2.CursorEditor
">
            <Color id="Hand Cursor"/>
          </Property>
        </Properties>
        <Events>
          <EventHandler event="mousePressed"
listener="java.awt.event.MouseListener"
parameters="java.awt.event.MouseEvent"
handler="configHelpdatasetHelpOnClick"/>
        </Events>
      </Component>
      <Component class="javax.swing.JButton"
name="resetButton">
        <Properties>
          <Property name="text" type="java.lang.String"
value="Reset"/>
          <Property name="toolTipText"
type="java.lang.String" value="Click to Reset"/>
        </Properties>
        <Events>
          <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener"
parameters="java.awt.event.ActionEvent"
handler="resetButtonActionPerformed"/>
        </Events>
        <AuxValues>
          <AuxValue
name="JavaCodeGenerator_VariableModifier"
type="java.lang.Integer" value="9"/>
        </AuxValues>
      </Component>
    </SubComponents>
  </Container>
  <Container class="javax.swing.JPanel"
name="trainDatasetPanel">

    <Layout>
      <DimensionLayout dim="0">
        <Group type="103" groupAlignment="0"
attributes="0">
          <Group type="102" attributes="0">
            <Component id="jLabel15" min="-2" max="-2"
attributes="0"/>
            <EmptySpace max="32767" attributes="0"/>
            <Component id="dlTvTemplate" min="-2"
max="-2" attributes="0"/>
            <EmptySpace max="-2" attributes="0"/>
            <Component id="jLabel2" min="-2" max="-2"
attributes="0"/>
          </Group>
          <Group type="102" attributes="0">
            <Group type="103" groupAlignment="0"
attributes="0">
              <Group type="102" attributes="0">
                <EmptySpace max="-2" attributes="0"/>
                <Group type="103" groupAlignment="1"
max="-2" attributes="0">
                  <Group type="102" alignment="1"
attributes="0">
                    <Component id="jLabel4" min="-2"
pref="94" max="-2" attributes="0"/>
                    <EmptySpace max="32767"
attributes="0"/>
                    <Component id="trainRatioBox"
min="-2" pref="207" max="-2" attributes="0"/>
                  </Group>
                  <Group type="102" alignment="0"
attributes="0">
                    <Component id="jLabel3" min="-2"
pref="94" max="-2" attributes="0"/>
                    <EmptySpace min="-2" pref="18"
max="-2" attributes="0"/>
                    <Component id="trainDatasetName"
min="-2" pref="207" max="-2" attributes="0"/>
                  </Group>
                </Group>
                <EmptySpace type="unrelated" max="-2"
attributes="0"/>
                <Component id="trainBrowseButton"
max="32767" attributes="0"/>
              </Group>
              <Group type="102" attributes="0">
                <Group type="103" groupAlignment="0"
attributes="0">
                  <Group type="102" alignment="0"
attributes="0">
                    <EmptySpace min="-2" pref="158"
max="-2" attributes="0"/>
                    <Component id="fileNotif" min="-2"
pref="171" max="-2" attributes="0"/>
                  </Group>
                  <Group type="102" alignment="0"
attributes="0">
                    <EmptySpace max="-2"
attributes="0"/>
                    <Component id="jLabel5" min="-2"
pref="94" max="-2" attributes="0"/>
                    <EmptySpace min="-2" pref="18"
max="-2" attributes="0"/>
                    <Component id="validRatioBox"
min="-2" pref="207" max="-2" attributes="0"/>
                  </Group>
                </Group>
                <EmptySpace min="0" pref="0"
max="32767" attributes="0"/>
              </Group>
            </Group>
            <EmptySpace max="-2" attributes="0"/>
```

```
            </Group>
            <Component id="jSeparator5" alignment="0"
max="32767" attributes="0"/>
          </Group>
        </DimensionLayout>
        <DimensionLayout dim="1">
          <Group type="103" groupAlignment="0"
attributes="0">
            <Group type="102" alignment="0"
attributes="0">
              <Component id="jSeparator5" min="-2"
pref="5" max="-2" attributes="0"/>
              <EmptySpace max="-2" attributes="0"/>
              <Group type="103" groupAlignment="0"
attributes="0">
                <Component id="jLabel15" min="-2"
pref="28" max="-2" attributes="0"/>
                <Component id="jLabel2" min="-2" max="-
2" attributes="0"/>
                <Component id="dlTvTemplate"
alignment="0" min="-2" max="-2" attributes="0"/>
              </Group>
              <EmptySpace type="unrelated" max="-2"
attributes="0"/>
              <Group type="103" groupAlignment="3"
attributes="0">
                <Component id="trainBrowseButton"
alignment="3" min="-2" max="-2" attributes="0"/>
                <Component id="trainDatasetName"
alignment="3" min="-2" max="-2" attributes="0"/>
                <Component id="jLabel3" alignment="3"
min="-2" pref="14" max="-2" attributes="0"/>
              </Group>
              <EmptySpace max="-2" attributes="0"/>
              <Group type="103" groupAlignment="3"
attributes="0">
                <Component id="jLabel4" alignment="3"
min="-2" max="-2" attributes="0"/>
                <Component id="trainRatioBox"
alignment="3" min="-2" max="-2" attributes="0"/>
              </Group>
              <EmptySpace type="unrelated" max="-2"
attributes="0"/>
              <Group type="103" groupAlignment="3"
attributes="0">
                <Component id="validRatioBox"
alignment="3" min="-2" max="-2" attributes="0"/>
                <Component id="jLabel5" alignment="3"
min="-2" pref="14" max="-2" attributes="0"/>
              </Group>
              <EmptySpace type="unrelated" max="-2"
attributes="0"/>
              <Component id="fileNotif" min="-2" pref="16"
max="-2" attributes="0"/>
              <EmptySpace max="32767" attributes="0"/>
            </Group>
          </Group>
        </DimensionLayout>
      </Layout>
      <SubComponents>
        <Component class="javax.swing.JLabel"
name="jLabel15">
          <Properties>
            <Property name="font" type="java.awt.Font"
editor="org.netbeans.beaninfo.editors.FontEditor">
              <Font name="Calibri" size="18" style="1"/>
            </Property>
            <Property name="text" type="java.lang.String"
value=" Data Set"/>
          </Properties>
        </Component>
        <Component class="javax.swing.JTextField"
name="trainDatasetName">
          <Properties>
            <Property name="toolTipText"
type="java.lang.String" value="Please select a file"/>
          </Properties>
        </Component>
        <Component class="javax.swing.JLabel"
name="jLabel3">
          <Properties>
            <Property name="text" type="java.lang.String"
value="Upload Data Set:"/>
          </Properties>
        </Component>
        <Component class="javax.swing.JLabel"
name="jLabel4">
          <Properties>
            <Property name="text" type="java.lang.String"
value="Training Ratio:"/>
          </Properties>
        </Component>
        <Component class="javax.swing.JButton"
name="trainBrowseButton">
          <Properties>
            <Property name="text" type="java.lang.String"
value="Browse"/>
            <Property name="toolTipText"
type="java.lang.String" value="Click to browse for a data
set"/>
          </Properties>
          <Events>
            <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener"
parameters="java.awt.event.ActionEvent"
handler="trainBrowseButtonActionPerformed"/>
          </Events>
        </Component>
        <Component class="javax.swing.JLabel"
name="jLabel5">
          <Properties>
            <Property name="text" type="java.lang.String"
value="Validating Ratio:"/>
          </Properties>
        </Component>
```

```
        <Component class="javax.swing.JTextField"
name="trainRatioBox">
          <Properties>
            <Property name="toolTipText"
type="java.lang.String" value="Please provide a value from
0 - 1"/>
          </Properties>
        </Component>
        <Component class="javax.swing.JTextField"
name="validRatioBox">
          <Properties>
            <Property name="toolTipText"
type="java.lang.String" value="Please provide a value from
0 - 1"/>
          </Properties>
        </Component>
        <Component class="javax.swing.JLabel"
name="fileNotif">
          <Properties>
            <Property name="foreground"
type="java.awt.Color"
editor="org.netbeans.beaninfo.editors.ColorEditor">
              <Color blue="0" green="ff" red="0" type="rgb"/>
            </Property>
            <Property name="name" type="java.lang.String"
value="fileNotif"/>
          </Properties>
          <AuxValues>
            <AuxValue
name="JavaCodeGenerator_AddingCodePre"
type="java.lang.String"
value="fileNotif.setVisible(false);"/>
          </AuxValues>
        </Component>
        <Component class="javax.swing.JLabel"
name="jLabel2">
          <Properties>
            <Property name="icon" type="javax.swing.Icon"
editor="org.netbeans.modules.form.editors2.IconEditor">
              <Image iconType="3" name="/qMark.png"/>
            </Property>
            <Property name="toolTipText"
type="java.lang.String" value="Click to access help"/>
            <Property name="cursor" type="java.awt.Cursor"
editor="org.netbeans.modules.form.editors2.CursorEditor
">
              <Color id="Hand Cursor"/>
            </Property>
          </Properties>
          <Events>
            <EventHandler event="mousePressed"
listener="java.awt.event.MouseListener"
parameters="java.awt.event.MouseEvent"
handler="datasetHelpOnClick"/>
          </Events>
        </Component>
        <Component class="javax.swing.JSeparator"
name="jSeparator5">
        </Component>
        <Component class="javax.swing.JLabel"
name="dlTvTemplate">
          <Properties>
            <Property name="icon" type="javax.swing.Icon"
editor="org.netbeans.modules.form.editors2.IconEditor">
              <Image iconType="3" name="/dl.png"/>
            </Property>
            <Property name="toolTipText"
type="java.lang.String" value="Click to download
template"/>
            <Property name="cursor" type="java.awt.Cursor"
editor="org.netbeans.modules.form.editors2.CursorEditor
">
              <Color id="Hand Cursor"/>
            </Property>
          </Properties>
          <Events>
            <EventHandler event="mousePressed"
listener="java.awt.event.MouseListener"
parameters="java.awt.event.MouseEvent"
handler="downloadTrainValTemplateOnClick"/>
          </Events>
        </Component>
      </SubComponents>
    </Container>
    <Component class="javax.swing.JSeparator"
name="jSeparator2">
    </Component>
    <Component class="javax.swing.JSeparator"
name="jSeparator6">
    </Component>
  </SubComponents>
 </Container>
 <Container class="javax.swing.JPanel" name="jPanel2">

  <Layout>
    <DimensionLayout dim="0">
      <Group type="103" groupAlignment="0"
attributes="0">
        <Group type="102" attributes="0">
          <EmptySpace max="-2" attributes="0"/>
          <Group type="103" groupAlignment="0"
attributes="0">
            <Group type="102" attributes="0">
              <Component id="resultHelp" min="-2"
max="-2" attributes="0"/>
              <EmptySpace min="-2" pref="27" max="-2"
attributes="0"/>
              <Component id="fileNotif1" min="-2"
max="-2" attributes="0"/>
              <EmptySpace min="0" pref="0"
max="32767" attributes="0"/>
            </Group>
            <Group type="102" attributes="0">
              <Group type="103" groupAlignment="1"
max="-2" attributes="0">
                <Group type="102" attributes="0">
```

```
                    <Component id="jLabel25" min="-2"
pref="215" max="-2" attributes="0"/>
                    <EmptySpace max="32767"
attributes="0"/>
                    <Component id="trainResultHelp"
min="-2" pref="20" max="-2" attributes="0"/>
                  </Group>
                  <Component id="jScrollPane1"
pref="463" max="32767" attributes="0"/>
                  <Component id="summaryPanel"
alignment="0" min="-2" max="-2" attributes="0"/>
                </Group>
                <EmptySpace max="32767"
attributes="0"/>
              </Group>
            </Group>
          </Group>
        </Group>
      </DimensionLayout>
      <DimensionLayout dim="1">
        <Group type="103" groupAlignment="0"
attributes="0">
          <Group type="102" alignment="1" attributes="0">
            <EmptySpace min="-2" pref="6" max="-2"
attributes="0"/>
            <Group type="103" groupAlignment="0"
attributes="0">
              <Component id="trainResultHelp" min="-2"
max="-2" attributes="0"/>
              <Component id="jLabel25" min="-2"
pref="28" max="-2" attributes="0"/>
            </Group>
            <EmptySpace max="-2" attributes="0"/>
            <Component id="jScrollPane1" min="-2"
pref="183" max="-2" attributes="0"/>
            <Group type="103" groupAlignment="0"
attributes="0">
              <Group type="102" attributes="0">
                <EmptySpace min="0" pref="0"
max="32767" attributes="0"/>
                <Component id="fileNotif1" min="-2"
max="-2" attributes="0"/>
                <EmptySpace min="-2" pref="40" max="-2"
attributes="0"/>
              </Group>
              <Group type="102" alignment="0"
attributes="0">
                <EmptySpace max="-2" attributes="0"/>
                <Component id="summaryPanel" min="-2"
pref="266" max="-2" attributes="0"/>
                <EmptySpace min="-2" pref="12" max="-2"
attributes="0"/>
                <Component id="resultHelp" max="32767"
attributes="0"/>
                <EmptySpace max="-2" attributes="0"/>
              </Group>
            </Group>
          </Group>
        </Group>
      </DimensionLayout>
    </Layout>
    <SubComponents>
      <Component class="javax.swing.JLabel"
name="fileNotif1">
        <Properties>
          <Property name="foreground"
type="java.awt.Color"
editor="org.netbeans.beaninfo.editors.ColorEditor">
            <Color blue="0" green="ff" red="0" type="rgb"/>
          </Property>
          <Property name="name" type="java.lang.String"
value="fileNotif"/>
        </Properties>
        <AuxValues>
          <AuxValue
name="JavaCodeGenerator_AddingCodePre"
type="java.lang.String"
value="fileNotif.setVisible(false);"/>
        </AuxValues>
      </Component>
      <Container class="javax.swing.JScrollPane"
name="jScrollPane1">
        <AuxValues>
          <AuxValue name="autoScrollPane"
type="java.lang.Boolean" value="true"/>
        </AuxValues>

        <Layout
class="org.netbeans.modules.form.compat2.layouts.suppo
rt.JScrollPaneSupportLayout"/>
        <SubComponents>
          <Component class="javax.swing.JTextArea"
name="trainingResultTextArea">
            <Properties>
              <Property name="columns" type="int"
value="20"/>
              <Property name="editable" type="boolean"
value="false"/>
              <Property name="rows" type="int" value="5"/>
            </Properties>
            <AuxValues>
              <AuxValue
name="JavaCodeGenerator_VariableModifier"
type="java.lang.Integer" value="9"/>
            </AuxValues>
          </Component>
        </SubComponents>
      </Container>
      <Component class="javax.swing.JLabel"
name="jLabel25">
        <Properties>
          <Property name="font" type="java.awt.Font"
editor="org.netbeans.beaninfo.editors.FontEditor">
            <Font name="Calibri" size="16" style="1"/>
          </Property>
```

```
        <Property name="text" type="java.lang.String"
value="Training/Validation Result"/>
       </Properties>
      </Component>
      <Container class="javax.swing.JPanel"
name="resultHelp">

       <Layout>
        <DimensionLayout dim="0">
         <Group type="103" groupAlignment="0"
attributes="0">
             <Component id="jScrollPane2" alignment="0"
max="32767" attributes="0"/>
             <Group type="102" alignment="0"
attributes="0">
               <Group type="103" groupAlignment="1"
max="-2" attributes="0">
                 <Group type="102" attributes="0">
                   <Component id="jLabel19" min="-2"
max="-2" attributes="0"/>
                   <EmptySpace max="32767"
attributes="0"/>
                   <Component id="resultsHelp" min="-2"
pref="20" max="-2" attributes="0"/>
                 </Group>
                 <Component id="jSeparator4" min="-2"
pref="463" max="-2" attributes="0"/>
               </Group>
               <EmptySpace min="0" pref="0" max="32767"
attributes="0"/>
             </Group>
             <Group type="102" alignment="1"
attributes="0">
               <EmptySpace max="32767" attributes="0"/>
               <Component id="jButton1" min="-2" max="-
2" attributes="0"/>
               <EmptySpace max="-2" attributes="0"/>
             </Group>
           </Group>
         </DimensionLayout>
         <DimensionLayout dim="1">
          <Group type="103" groupAlignment="0"
attributes="0">
             <Group type="102" attributes="0">
               <Component id="jSeparator4" min="-2"
pref="10" max="-2" attributes="0"/>
               <EmptySpace max="-2" attributes="0"/>
               <Group type="103" groupAlignment="0"
attributes="0">
                 <Component id="jLabel19" min="-2" max="-
2" attributes="0"/>
                 <Component id="resultsHelp" min="-2"
max="-2" attributes="0"/>
               </Group>
               <EmptySpace min="-2" pref="10" max="-2"
attributes="0"/>
               <Component id="jScrollPane2" pref="229"
max="32767" attributes="0"/>
               <EmptySpace max="-2" attributes="0"/>
               <Component id="jButton1" min="-2" max="-
2" attributes="0"/>
             </Group>
           </Group>
         </DimensionLayout>
        </Layout>
        <SubComponents>
         <Container class="javax.swing.JScrollPane"
name="jScrollPane2">
          <AuxValues>
           <AuxValue name="autoScrollPane"
type="java.lang.Boolean" value="true"/>
          </AuxValues>

          <Layout
class="org.netbeans.modules.form.compat2.layouts.suppo
rt.JScrollPaneSupportLayout"/>
          <SubComponents>
           <Component class="javax.swing.JTable"
name="diagTable">
            <Properties>
             <Property name="model"
type="javax.swing.table.TableModel"
editor="org.netbeans.modules.form.editors2.TableModelE
ditor">
              <Table columnCount="9" rowCount="0">
               <Column editable="false" title="Temp."
type="java.lang.Object"/>
               <Column editable="false" title="P. Count"
type="java.lang.Object"/>
               <Column editable="false" title="C. Rate"
type="java.lang.Object"/>
               <Column editable="false" title="BP(SP)"
type="java.lang.Object"/>
               <Column editable="false" title="BP(DP)"
type="java.lang.Object"/>
               <Column editable="false" title="R. Rate"
type="java.lang.Object"/>
               <Column editable="false" title="WBC"
type="java.lang.Object"/>
               <Column editable="false" title="RBC"
type="java.lang.Object"/>
               <Column editable="false" title="Diagnosis"
type="java.lang.Object"/>
              </Table>
             </Property>
             <Property name="cellSelectionEnabled"
type="boolean" value="true"/>
             <Property name="columnModel"
type="javax.swing.table.TableColumnModel"
editor="org.netbeans.modules.form.editors2.TableColumn
ModelEditor">
              <TableColumnModel selectionModel="2">
               <Column maxWidth="-1" minWidth="-1"
prefWidth="-1" resizable="true">
                <Title/>
                <Editor/>
```

```
            <Renderer/>
          </Column>
          <Column maxWidth="-1" minWidth="-1"
prefWidth="-1" resizable="true">
            <Title/>
            <Editor/>
            <Renderer/>
          </Column>
          <Column maxWidth="-1" minWidth="-1"
prefWidth="-1" resizable="true">
            <Title/>
            <Editor/>
            <Renderer/>
          </Column>
          <Column maxWidth="-1" minWidth="-1"
prefWidth="-1" resizable="true">
            <Title/>
            <Editor/>
            <Renderer/>
          </Column>
          <Column maxWidth="-1" minWidth="-1"
prefWidth="-1" resizable="true">
            <Title/>
            <Editor/>
            <Renderer/>
          </Column>
          <Column maxWidth="-1" minWidth="-1"
prefWidth="-1" resizable="true">
            <Title/>
            <Editor/>
            <Renderer/>
          </Column>
          <Column maxWidth="-1" minWidth="-1"
prefWidth="-1" resizable="true">
            <Title/>
            <Editor/>
            <Renderer/>
          </Column>
          <Column maxWidth="-1" minWidth="-1"
prefWidth="-1" resizable="true">
            <Title/>
            <Editor/>
            <Renderer/>
          </Column>
          <Column maxWidth="-1" minWidth="-1"
prefWidth="-1" resizable="true">
            <Title/>
            <Editor/>
            <Renderer/>
          </Column>
        </TableColumnModel>
      </Property>
      <Property name="tableHeader"
type="javax.swing.table.JTableHeader"
editor="org.netbeans.modules.form.editors2.JTableHeade
rEditor">
        <TableHeader reorderingAllowed="true"
resizingAllowed="true"/>
            </Property>
          </Properties>
          <AuxValues>
            <AuxValue
name="JavaCodeGenerator_VariableModifier"
type="java.lang.Integer" value="12"/>
          </AuxValues>
        </Component>
      </SubComponents>
    </Container>
    <Component class="javax.swing.JLabel"
name="jLabel19">
      <Properties>
        <Property name="font" type="java.awt.Font"
editor="org.netbeans.beaninfo.editors.FontEditor">
          <Font name="Calibri" size="16" style="1"/>
        </Property>
        <Property name="text" type="java.lang.String"
value="Results:"/>
      </Properties>
    </Component>
    <Component class="javax.swing.JLabel"
name="resultsHelp">
      <Properties>
        <Property name="icon" type="javax.swing.Icon"
editor="org.netbeans.modules.form.editors2.IconEditor">
          <Image iconType="3" name="/qMark.png"/>
        </Property>
        <Property name="toolTipText"
type="java.lang.String" value="Click to access help"/>
        <Property name="cursor" type="java.awt.Cursor"
editor="org.netbeans.modules.form.editors2.CursorEditor
">
          <Color id="Hand Cursor"/>
        </Property>
      </Properties>
      <Events>
        <EventHandler event="mousePressed"
listener="java.awt.event.MouseListener"
parameters="java.awt.event.MouseEvent"
handler="diagResultsHelpOnClick"/>
      </Events>
    </Component>
    <Component class="javax.swing.JSeparator"
name="jSeparator4">
    </Component>
    <Component class="javax.swing.JButton"
name="jButton1">
      <Properties>
        <Property name="text" type="java.lang.String"
value="Export"/>
        <Property name="toolTipText"
type="java.lang.String" value="Click to export diagnosis
results"/>
      </Properties>
      <Events>
        <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener"
```

```
parameters="java.awt.event.ActionEvent"
handler="exportButtonOnClick"/>
          </Events>
        </Component>
      </SubComponents>
    </Container>
    <Container class="javax.swing.JPanel"
name="summaryPanel">

      <Layout>
        <DimensionLayout dim="0">
          <Group type="103" groupAlignment="0"
attributes="0">
            <Component id="diagnosisPanel" max="32767"
attributes="0"/>
            <Group type="102" attributes="0">
              <Component id="jLabel6" min="-2" max="-2"
attributes="0"/>
              <EmptySpace max="32767" attributes="0"/>
              <Component id="summaryHelp" min="-2"
pref="20" max="-2" attributes="0"/>
            </Group>
            <Group type="102" attributes="0">
              <EmptySpace max="-2" attributes="0"/>
              <Group type="103" groupAlignment="0"
attributes="0">
                <Component id="jLabel20" alignment="0"
min="-2" max="-2" attributes="0"/>
                <Component id="jLabel21" alignment="0"
min="-2" max="-2" attributes="0"/>
                <Component id="jLabel22" alignment="0"
min="-2" max="-2" attributes="0"/>
              </Group>
              <EmptySpace max="-2" attributes="0"/>
              <Group type="103" groupAlignment="0"
max="-2" attributes="0">
                <Component id="timeElapsedLabel"
alignment="0" max="32767" attributes="0"/>
                <Component id="numGenLabel"
alignment="0" max="32767" attributes="0"/>
                <Component id="finalSSELabel"
alignment="0" min="-2" pref="225" max="-2"
attributes="0"/>
              </Group>
              <EmptySpace min="0" pref="0" max="32767"
attributes="0"/>
            </Group>
            <Component id="jSeparator3" alignment="1"
max="32767" attributes="0"/>
          </Group>
        </DimensionLayout>
        <DimensionLayout dim="1">
          <Group type="103" groupAlignment="0"
attributes="0">
            <Group type="102" alignment="0"
attributes="0">
              <EmptySpace max="-2" attributes="0"/>
              <Group type="103" groupAlignment="0"
attributes="0">
                <Component id="jLabel6" min="-2" max="-
2" attributes="0"/>
                <Component id="summaryHelp" min="-2"
max="-2" attributes="0"/>
              </Group>
              <EmptySpace max="-2" attributes="0"/>
              <Group type="103" groupAlignment="1"
max="-2" attributes="0">
                <Group type="102" attributes="0">
                  <Component id="jLabel20" min="-2"
max="-2" attributes="0"/>
                  <EmptySpace min="-2" pref="5" max="-2"
attributes="0"/>
                  <Component id="jLabel21" min="-2"
max="-2" attributes="0"/>
                </Group>
                <Group type="102" attributes="0">
                  <Component id="finalSSELabel"
max="32767" attributes="0"/>
                  <EmptySpace max="-2" attributes="0"/>
                  <Component id="timeElapsedLabel"
min="-2" pref="14" max="-2" attributes="0"/>
                </Group>
              </Group>
              <EmptySpace max="-2" attributes="0"/>
              <Group type="103" groupAlignment="0"
attributes="0">
                <Component id="jLabel22" max="32767"
attributes="0"/>
                <Component id="numGenLabel" min="-2"
pref="18" max="-2" attributes="0"/>
              </Group>
              <EmptySpace min="-2" pref="22" max="-2"
attributes="0"/>
              <Component id="jSeparator3" min="-2"
max="-2" attributes="0"/>
              <EmptySpace max="-2" attributes="0"/>
              <Component id="diagnosisPanel" min="-2"
max="-2" attributes="0"/>
              <EmptySpace max="-2" attributes="0"/>
            </Group>
          </Group>
        </DimensionLayout>
      </Layout>
      <SubComponents>
        <Component class="javax.swing.JLabel"
name="jLabel6">
          <Properties>
            <Property name="font" type="java.awt.Font"
editor="org.netbeans.beaninfo.editors.FontEditor">
              <Font name="Calibri" size="16" style="1"/>
            </Property>
            <Property name="text" type="java.lang.String"
value="Summary"/>
          </Properties>
        </Component>
```

```
        <Component class="javax.swing.JLabel"
name="jLabel20">
          <Properties>
            <Property name="text" type="java.lang.String"
value="SSE:"/>
          </Properties>
        </Component>
        <Component class="javax.swing.JLabel"
name="jLabel21">
          <Properties>
            <Property name="text" type="java.lang.String"
value="Time Elapsed:"/>
          </Properties>
        </Component>
        <Component class="javax.swing.JLabel"
name="jLabel22">
          <Properties>
            <Property name="text" type="java.lang.String"
value="Number of Generations:"/>
          </Properties>
        </Component>
        <Component class="javax.swing.JLabel"
name="finalSSELabel">
          <AuxValues>
            <AuxValue
name="JavaCodeGenerator_VariableModifier"
type="java.lang.Integer" value="9"/>
          </AuxValues>
        </Component>
        <Component class="javax.swing.JLabel"
name="timeElapsedLabel">
          <AuxValues>
            <AuxValue
name="JavaCodeGenerator_VariableModifier"
type="java.lang.Integer" value="1"/>
          </AuxValues>
        </Component>
        <Component class="javax.swing.JLabel"
name="numGenLabel">
          <Properties>
            <Property name="toolTipText"
type="java.lang.String" value=""/>
          </Properties>
          <AuxValues>
            <AuxValue
name="JavaCodeGenerator_VariableModifier"
type="java.lang.Integer" value="9"/>
          </AuxValues>
        </Component>
        <Component class="javax.swing.JSeparator"
name="jSeparator3">
        </Component>
        <Container class="javax.swing.JPanel"
name="diagnosisPanel">
          <AuxValues>
            <AuxValue
name="JavaCodeGenerator_VariableModifier"
type="java.lang.Integer" value="1"/>
          </AuxValues>

          <Layout>
            <DimensionLayout dim="0">
              <Group type="103" groupAlignment="0"
attributes="0">
                <Group type="102" attributes="0">
                  <Group type="103" groupAlignment="0"
attributes="0">
                    <Component id="jLabel17" alignment="0"
min="-2" max="-2" attributes="0"/>
                    <Group type="102" alignment="0"
attributes="0">
                      <EmptySpace max="-2"
attributes="0"/>
                      <Group type="103"
groupAlignment="0" attributes="0">
                        <Component id="jLabel18"
alignment="0" min="-2" pref="93" max="-2"
attributes="0"/>
                        <Component id="jLabel28"
alignment="0" min="-2" max="-2" attributes="0"/>
                      </Group>
                      <EmptySpace max="-2"
attributes="0"/>
                      <Group type="103"
groupAlignment="0" max="-2" attributes="0">
                        <Group type="102" alignment="0"
attributes="0">
                          <Component id="batchEntryBox"
min="-2" pref="155" max="-2" attributes="0"/>
                          <EmptySpace max="-2"
attributes="0"/>
                          <Component
id="diagBrowseButton" min="-2" max="-2"
attributes="0"/>
                        </Group>
                        <Component id="singleEntryBox"
alignment="0" max="32767" attributes="0"/>
                      </Group>
                      <EmptySpace max="-2"
attributes="0"/>
                      <Group type="103"
groupAlignment="0" attributes="0">
                        <Component
id="singleEntryDiagButton" min="-2" max="-2"
attributes="0"/>
                        <Component id="batchDiagButton"
min="-2" max="-2" attributes="0"/>
                      </Group>
                    </Group>
                    <Component id="jLabel27" alignment="0"
min="-2" max="-2" attributes="0"/>
                  </Group>
                  <EmptySpace pref="53" max="32767"
attributes="0"/>
                </Group>
                <Group type="102" attributes="0">
```

```
                <Component id="jLabel16" min="-2" max="-
2" attributes="0"/>
                <EmptySpace max="32767"
attributes="0"/>
                <Component id="downloadDiagTemplate"
min="-2" max="-2" attributes="0"/>
                <EmptySpace max="-2" attributes="0"/>
                <Component id="diagnosisHelp" min="-2"
pref="20" max="-2" attributes="0"/>
              </Group>
            </Group>
          </DimensionLayout>
          <DimensionLayout dim="1">
            <Group type="103" groupAlignment="0"
attributes="0">
              <Group type="102" alignment="0"
attributes="0">
                <EmptySpace min="-2" pref="2" max="-2"
attributes="0"/>
                <Group type="103" groupAlignment="0"
attributes="0">
                  <Component
id="downloadDiagTemplate" min="-2" max="-2"
attributes="0"/>
                  <Component id="diagnosisHelp" min="-2"
max="-2" attributes="0"/>
                  <Group type="102" attributes="0">
                    <Component id="jLabel16" min="-2"
max="-2" attributes="0"/>
                    <EmptySpace max="-2"
attributes="0"/>
                    <Component id="jLabel17" min="-2"
max="-2" attributes="0"/>
                    <EmptySpace min="-2" pref="4" max="-
2" attributes="0"/>
                    <Group type="103"
groupAlignment="3" attributes="0">
                      <Component id="jLabel18"
alignment="3" min="-2" pref="14" max="-2"
attributes="0"/>
                      <Component id="batchEntryBox"
alignment="3" min="-2" max="-2" attributes="0"/>
                      <Component id="diagBrowseButton"
alignment="3" min="-2" max="-2" attributes="0"/>
                      <Component id="batchDiagButton"
alignment="3" min="-2" max="-2" attributes="0"/>
                    </Group>
                    <EmptySpace max="-2"
attributes="0"/>
                    <Component id="jLabel27" min="-2"
max="-2" attributes="0"/>
                    <EmptySpace max="-2"
attributes="0"/>
                    <Group type="103"
groupAlignment="3" attributes="0">
                      <Component id="jLabel28"
alignment="3" min="-2" pref="14" max="-2"
attributes="0"/>
                      <Component id="singleEntryBox"
alignment="3" min="-2" max="-2" attributes="0"/>
                      <Component
id="singleEntryDiagButton" alignment="3" min="-2"
max="-2" attributes="0"/>
                    </Group>
                  </Group>
                </Group>
                <EmptySpace pref="31" max="32767"
attributes="0"/>
              </Group>
            </Group>
          </DimensionLayout>
        </Layout>
        <SubComponents>
          <Component class="javax.swing.JLabel"
name="jLabel18">
            <Properties>
              <Property name="text" type="java.lang.String"
value="Upload Data Set:"/>
            </Properties>
          </Component>
          <Component class="javax.swing.JLabel"
name="jLabel17">
            <Properties>
              <Property name="font" type="java.awt.Font"
editor="org.netbeans.beaninfo.editors.FontEditor">
                <Font name="Calibri" size="14" style="1"/>
              </Property>
              <Property name="text" type="java.lang.String"
value="Batch Upload"/>
            </Properties>
          </Component>
          <Component class="javax.swing.JLabel"
name="jLabel16">
            <Properties>
              <Property name="font" type="java.awt.Font"
editor="org.netbeans.beaninfo.editors.FontEditor">
                <Font name="Calibri" size="16" style="1"/>
              </Property>
              <Property name="text" type="java.lang.String"
value="Diagnosis"/>
            </Properties>
          </Component>
          <Component class="javax.swing.JButton"
name="batchDiagButton">
            <Properties>
              <Property name="text" type="java.lang.String"
value="GO"/>
              <Property name="toolTipText"
type="java.lang.String" value="Click to proceed with
diagnosis"/>
              <Property name="enabled" type="boolean"
value="false"/>
            </Properties>
            <Events>
              <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener"
```

```
parameters="java.awt.event.ActionEvent"
handler="batchDiagButtonActionPerformed"/>
            </Events>
            <AuxValues>
              <AuxValue
name="JavaCodeGenerator_VariableModifier"
type="java.lang.Integer" value="9"/>
            </AuxValues>
          </Component>
          <Component class="javax.swing.JTextField"
name="batchEntryBox">
            <Properties>
              <Property name="toolTipText"
type="java.lang.String" value="Please select a file"/>
              <Property name="enabled" type="boolean"
value="false"/>
            </Properties>
            <AuxValues>
              <AuxValue
name="JavaCodeGenerator_VariableModifier"
type="java.lang.Integer" value="1"/>
            </AuxValues>
          </Component>
          <Component class="javax.swing.JButton"
name="diagBrowseButton">
            <Properties>
              <Property name="text" type="java.lang.String"
value="Browse"/>
              <Property name="toolTipText"
type="java.lang.String" value="Click to browse for a data
set"/>
              <Property name="enabled" type="boolean"
value="false"/>
            </Properties>
            <Events>
              <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener"
parameters="java.awt.event.ActionEvent"
handler="diagBrowseButtonActionPerformed"/>
            </Events>
            <AuxValues>
              <AuxValue
name="JavaCodeGenerator_VariableModifier"
type="java.lang.Integer" value="1"/>
            </AuxValues>
          </Component>
          <Component class="javax.swing.JLabel"
name="jLabel27">
            <Properties>
              <Property name="font" type="java.awt.Font"
editor="org.netbeans.beaninfo.editors.FontEditor">
                <Font name="Calibri" size="14" style="1"/>
              </Property>
              <Property name="text" type="java.lang.String"
value="Single Entry"/>
            </Properties>
          </Component>
          <Component class="javax.swing.JTextField"
name="singleEntryBox">
            <Properties>
              <Property name="toolTipText"
type="java.lang.String" value="Please provide an entry
with
format:&amp;lt;value1&amp;gt;;&amp;lt;value2&amp;gt;;&amp;lt;value3&amp;gt;;...;"/
>
              <Property name="enabled" type="boolean"
value="false"/>
            </Properties>
            <AuxValues>
              <AuxValue
name="JavaCodeGenerator_VariableModifier"
type="java.lang.Integer" value="1"/>
            </AuxValues>
          </Component>
          <Component class="javax.swing.JLabel"
name="jLabel28">
            <Properties>
              <Property name="text" type="java.lang.String"
value="Upload Data Entry:    "/>
            </Properties>
          </Component>
          <Component class="javax.swing.JButton"
name="singleEntryDiagButton">
            <Properties>
              <Property name="text" type="java.lang.String"
value="GO"/>
              <Property name="toolTipText"
type="java.lang.String" value="Click to proceed with
diagnosis"/>
              <Property name="enabled" type="boolean"
value="false"/>
            </Properties>
            <Events>
              <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener"
parameters="java.awt.event.ActionEvent"
handler="singleEntryDiagButtonActionPerformed"/>
            </Events>
            <AuxValues>
              <AuxValue
name="JavaCodeGenerator_VariableModifier"
type="java.lang.Integer" value="1"/>
            </AuxValues>
          </Component>
          <Component class="javax.swing.JLabel"
name="diagnosisHelp">
            <Properties>
              <Property name="icon"
type="javax.swing.Icon"
editor="org.netbeans.modules.form.editors2.IconEditor">
                <Image iconType="3" name="/qMark.png"/>
              </Property>
              <Property name="toolTipText"
type="java.lang.String" value="Click to access help"/>
```

```
            <Property name="cursor"
type="java.awt.Cursor"
editor="org.netbeans.modules.form.editors2.CursorEditor
">
              <Color id="Hand Cursor"/>
             </Property>
            </Properties>
            <Events>
             <EventHandler event="mousePressed"
listener="java.awt.event.MouseListener"
parameters="java.awt.event.MouseEvent"
handler="diagnosisHelpOnClick"/>
            </Events>
           </Component>
           <Component class="javax.swing.JLabel"
name="downloadDiagTemplate">
            <Properties>
             <Property name="icon"
type="javax.swing.Icon"
editor="org.netbeans.modules.form.editors2.IconEditor">
              <Image iconType="3" name="/dl.png"/>
             </Property>
             <Property name="toolTipText"
type="java.lang.String" value="Click to download
template"/>
             <Property name="cursor"
type="java.awt.Cursor"
editor="org.netbeans.modules.form.editors2.CursorEditor
">
              <Color id="Hand Cursor"/>
             </Property>
            </Properties>
            <Events>
             <EventHandler event="mousePressed"
listener="java.awt.event.MouseListener"
parameters="java.awt.event.MouseEvent"
handler="downloadDiagTemplateOnClick"/>
            </Events>
           </Component>
          </SubComponents>
         </Container>
         <Component class="javax.swing.JLabel"
name="summaryHelp">
          <Properties>
           <Property name="icon" type="javax.swing.Icon"
editor="org.netbeans.modules.form.editors2.IconEditor">
            <Image iconType="3" name="/qMark.png"/>
           </Property>
           <Property name="toolTipText"
type="java.lang.String" value="Click to access help"/>
           <Property name="cursor" type="java.awt.Cursor"
editor="org.netbeans.modules.form.editors2.CursorEditor
">
            <Color id="Hand Cursor"/>
           </Property>
          </Properties>
          <Events>
           <EventHandler event="mousePressed"
listener="java.awt.event.MouseListener"
parameters="java.awt.event.MouseEvent"
handler="summaryHelpOnClick"/>
          </Events>
         </Component>
        </SubComponents>
       </Container>
       <Component class="javax.swing.JLabel"
name="trainResultHelp">
        <Properties>
         <Property name="icon" type="javax.swing.Icon"
editor="org.netbeans.modules.form.editors2.IconEditor">
          <Image iconType="3" name="/qMark.png"/>
         </Property>
         <Property name="toolTipText"
type="java.lang.String" value="Click to access help"/>
         <Property name="cursor" type="java.awt.Cursor"
editor="org.netbeans.modules.form.editors2.CursorEditor
">
          <Color id="Hand Cursor"/>
         </Property>
        </Properties>
        <Events>
         <EventHandler event="mousePressed"
listener="java.awt.event.MouseListener"
parameters="java.awt.event.MouseEvent"
handler="trainResultHelpOnClick"/>
        </Events>
       </Component>
      </SubComponents>
     </Container>
    </SubComponents>
</Form>
```

B. Other Files

DataForTrainingValidatingTemplate.txt

```
//GNAT:
//GENETIC NEURAL NETWORK ANALYTIC TOOL FOR
DENGUE
//APPLICATION OF GENETIC ALGORITHM OPTIMIZED
//ARTIFICIAL NEURAL NETWORK
//FOR THE MEDICAL DIAGNOSIS OF DENGUE FEVER
//Copyright 2013 Mikyle Laurence O. Palomo

//By Mikyle Laurence O. Palomo
//2008-53909
//BS Computer Science
//mikylepalomo@yahoo.com
//mikpalomo@gmail.com

//====TEMPLATE FOR TRAINING/VALIDATING DATA SET
//Application does not read commented lines(lines that
start with "//")
//Application removes entries with incomplete values
//Application removes entries with invalid values
//format:<symptom1>;<symptom2>;...;<symptomN>;<diag
nosis>
//TEMP;PC;CR;BP-S;BP-D;RR;WBC;RBC;diagnosis
//Temperature[TEMP]: degrees celsius
//Platelet Count[PC]: count x 10^3/microliter
//Cardiac Rate[CR]: beats per minute
//Blood Pressure(systolic)[BP-S]: mmHg
//Blood Pressure(diastolic)[SP-D]: mmHg
//Respiratory Rate[RR]: breaths per minute
//White Blood Cell Count[WBC]: count x 10^6 microliter
//Red Blood Cell Count[RBC]: count x 10^6 microliter
//diagnosis: 0-Negative 1-Positive
//sample data:
//36.8;264;95;160;90;22;16.2;5.26;0
//37.4;164;92;100;60;24;8.9;4.3;1
```

DataForBatchDiagTemplate.txt

```
//GNAT:
//GENETIC NEURAL NETWORK ANALYTIC TOOL FOR
DENGUE
//APPLICATION OF GENETIC ALGORITHM OPTIMIZED
//ARTIFICIAL NEURAL NETWORK
//FOR THE MEDICAL DIAGNOSIS OF DENGUE FEVER
//Copyright 2013 Mikyle Laurence O. Palomo

//By Mikyle Laurence O. Palomo
//2008-53909
//BS Computer Science
//mikylepalomo@yahoo.com
//mikpalomo@gmail.com

//====TEMPLATE FOR DIAGNOSIS DATA SET
//Application does not read commented lines(lines that
start with "//")
//Application removes entries with incomplete values
//Application removes entries with invalid values
//format:<symptom1>;<symptom2>;...;<symptomN>
//TEMP;PC;CR;BP-S;BP-D;RR;WBC;RBC
//Temperature[TEMP]: degrees celsius
//Platelet Count[PC]: count x 10^3/microliter
//Cardiac Rate[CR]: beats per minute
//Blood Pressure(systolic)[BP-S]: mmHg
//Blood Pressure(diastolic)[SP-D]: mmHg
//Respiratory Rate[RR]: breaths per minute
//White Blood Cell Count[WBC]: count x 10^6 microliter
//Red Blood Cell Count[RBC]: count x 10^6 microliter
//sample data:
//36.8;264;95;160;90;22;16.2;5.26
//37.4;164;92;100;60;24;8.9;4.3
```

XI. Acknowledgement

Plain words cannot express how thankful I am to the following people, who helped me through this climb. Though long overdue, I am still grateful for all the people who were behind me. These people may not know how truly grateful I am for their invaluable contributions to this paper:

To the one up above; no one may truly guarantee that you exist but I can guarantee that the faith in you alone can be powerful. To God be the glory.

My family; I know that they already know that this was all for them. For Pa, Ma, Mica, Mitch, Michelle, and Micole, knowing that you are happy for all this, makes it all worthwhile. I just hope that they never stop believing in me and always be there by my side, through thick and thin. Pa, this one's for you. ☺

Tito Anton and Tita Bing; always remember that I am eternally grateful for all you have done, not only for me, but for all of us. Know that I will always be here whenever you need me and our doors are always open for you guys.

DPSM and all it professors; most definitely the best department in UP, may you never lose the enthusiasm and pride of being able to nurture and train the future heralds of our country. Speaking for all your students, we are all indebted for the timeless lessons and the priceless time that you have devoted for all of us.

My adviser, Sir Bernie Terrado; my sincerest gratitude for all the support you have given me. Thank you for the support, effort, and time that you have devoted for me and this paper. I cannot begin to express how truly grateful I am.

Computer Science '08; you guys are the best. Enough said.

To Future me; by the time you read this once more, I hope you look back and appreciate, not only this work, but the people that were behind you. Remember, that you owe the people above a favor, without them, all of this will not be possible. In whatever way, I hope you will never stop the pursuit of knowledge and always have an open mind. I hope you are happy with what you have right now and cherish all the fruits of what you have worked for in the past.

AAAAWWWWWWWWWWWW YYEAAAAAAAAAHHHHHHHHHH.