**CHAPTER 4**

**EXPERIMENTAL RESULTS AND COMPARATIVE ANALYSIS**

This chapter list out all experiments, results and comparative analysis related to the proposed framework. Each component will be defined as a milestone in the proposed model. In fact, each of these components contains comparison with another approach to highlight the improvements of the proposed approach.

Section 42 will present about datasets used in this experiment. Section 5.3 will explain the experiment as well as comparative analysis of the controlled environment.

comparative analysis are presented in Section 5.4. Next, Section 5.5 will describe the implementation results, accuracy measurement and comparative analysis. The evaluation and analysis result on malware classification are presented in Section 5.6. Section 5.7 discuss about the proposed framework by highlighting overall milestone of the components. Lastly, Section 5.8 summarizes the chapter.

**4.1 IMPLEMENTATION PHASE**

SIMULATION

MATLAB

REASON TO CHOOSE MATLAB

COMPONENT OF MATLAB FOR CLOUD COMPUTING

SIMULATION DETAILS

Compare the algorithm privileges with other respected scholar works

Simulation Parameters and Performance Metrics

**4.1.1 REASON TO CHOOSE MATLAB**

MATLAB has several advantages over other methods or languages:

* Its basic data element is the matrix. A simple integer is considered an matrix of one row and one column. Several mathematical operations that work on arrays or matrices are built-in to the Matlab environment. For example, cross-products, dot-products, determinants, inverse matrices.
* Vectorized operations. Adding two arrays together needs only one command, instead of a for or while loop.
* The graphical output is optimized for interaction. You can plot your data very easily, and then change colors, sizes, scales, etc, by using the graphical interactive tools.
* Matlab’s functionality can be greatly expanded by the addition of toolboxes. These are sets of specific functions that provided more specialized functionality. Ex: Excel link allows data to be written in a format recognized by Excel, Statistics Toolbox allows more specialized statistical manipulation of data (Anova, Basic Fits, etc)

There are also disadvantages:

* It uses a large amount of memory and on slow computers it is very hard to use.
* It sits “on top” of Windows, getting as much CPU time as Windows allows it to have. This makes real-time applications very complicated.

**security tool performance parameter**

**Cloud Computing**

Cloud computing is the way to use computing resources as an utility. Both hardware and software can be delivered as a service over a network (typically the Internet).According to Ref. [43], there are two types of cloud: public cloud and private cloud.

building cloud computing

A public cloud is designed to apply a pay-as-you-go manner to the general public to provide services. And a private cloud is usually used to deal with the inside data of an organization which are not open to the public. For example, Ref. shows that the IBM smart cloud can provide private cloud service by giving threat protection for every layer of virtual infrastructure, limiting access to critical data, tracking user access and getting virtual infrastructure reports.

## 4.3.3 CLOUD WEB SERVER

The Java platform has turned into a pillar of big business IT improvement since the presentation of the Enterprise Edition in 1998, in two unique routes: Through the coupling of Java to the cloud web server, the Java platform has turned into a main platform for coordinating the Web with big business backend frameworks. This has permitted organizations to move part or the majority of their business to the Internet environment by method for very intuitive online situations.

The Java platform has developed into an Enterprise Integration part in which legacy frameworks are opened to the outside world through extensions based on the Java stage. This pattern has been bolstered for Java stage support for EAI norms like informing and Web benefits and has powered the incorporation of the Java platform as an advancement premise in such measures as SCA, XAM and others.These platforms shown in table 4.3

Table 4.3: Software tools and platforms

|  |  |  |  |
| --- | --- | --- | --- |
| **Component** | **JBoss EWS 2.0.0** | **JBoss EWS 2.0.1** | **JBoss EWS 2.1.0** |
| Apache Web Server (httpd) | 2.2.22 | 2.2.22 | 2.2.26 |
| Apache Tomcat 7 | 7.0.30 | 7.0.40 | 7.0.54 |
| Apache Tomcat 6 | 6.0.35 | 6.0.37 | 6.0.41 |
| Apache Tomcat Native | 1.1.24 | 1.1.27 | 1.1.30 |
| Mod.jk | 1.2.36 | 1.2.37 | 1.2.40 |
| Mod cluster | 1.2.3.Final | 1.2.4.Final | 1.2.9.Final |

As a major aspect of the JBoss Enterprise Web Server (JBoss EWS) discharge handle, a few segments may be included or evacuated for reliance or similarity purposes. Also, a few segments may be incorporated as innovation sneak peaks. The following are the individual segment subtle elements for each JBoss EWS discharge.

## 4.3.4 JBOSS ENTERPRISE WEBSERVER 2

The following components have been integrated into JBoss EWS 2.0 and are fully supported for use in both development and production (according to your corresponding subscription agreement). The following components have been tested with JBoss EWS 2.0 and are fully supported for use in both development and production with a subscription to EWS Plus (according to your corresponding subscription agreement).

# 4.3.5 TOMCAT

Apache Tomcat, regularly alluded to as Tomcat, is an open-source web server created by the Apache Software Foundation (ASF). Tomcat executes a few Java EE determinations including Java Servlet, JavaServer Pages (JSP), Java EL, and WebSocket, and gives an "immaculate Java" HTTP web server environment for Java code to keep running in. Tomcat is created and kept up by an open group of designers under the support of the Apache Software Foundation, discharged under the Apache License 2.0 permit, and is open-source programming. Tomcat began off as a servlet reference usage by James Duncan Davidson, a product designer at Sun Microsystems. He later made the undertaking open source and assumed a key part in its gift by Sun Microsystems to the Apache Software Foundation. The Apache Ant programming fabricate robotization device was produced as a side-

Impact of the formation of Tomcat as an open source project.Davidson had at first trusted that the undertaking would get to be publicly released and, since numerous open source ventures had O'Reilly books connected with them highlighting a creature on the spread, he needed to name the task after a creature. Tomcat was discharged with four parts: Catalina (servlet holder), cote (HTTP connector) Jaspera (JSP motor). Catalina is tomcat servlet holder, it actualizes Sun Microsystem determinations for servlet and Java server pages. In tomcat a domain component speaks to a Database of username, secret word and moves like UNIX gatherings relegated to those clients. Distinctive executions a domain permit Catalina to be incorporated into situations where such validation data is as of now been made and kept up and after that utilization that data to actualize contain an administration security as portrayed in the servlet determination

Coyote is Tomcats HTP connector component. It support HTTP1.1 protocol for the web cloud server application container. Onyote listens for incoming connections on a specific TCP port the server and forwards the resource to the Tomcat Engine. The Tomcat in process the request and sends back a response to requesting client. Coytote can execute the JSPs and servlets

Jesper is Tomcats JSP engine and JSP files to compile them into Java codes as servlets. The compile Java code can be handled by Catlina and runtime Jasper detects changes to JSP files and recomple slim. Jesper recompiles JSP files. As version 5 tomecat uses Jesper 2. Who is an implemntaion of Sun Microsystem is Jay sp2 specification.

**4.5 DEPLOYMENT OPTIONS**

Converting your Matlab code to a web service makes it possible that this code is used anywhere on the world. Your software can be called from a HTTP server, directltyfrom a browser, or by another computer program. Apart from this interoperability advantage, making software available as a webservice may help you comply withcorporate IT requirements, allows you to protect sensitive data, or simply increase computing capacity by running many instances of single webservice at the sametime.The toolbox has many deployment options (see Figure 1), and new ones can beadded upon request. The table below shows options that have been applied in the past.

# **4.5.1 REQUIREMENTS FOR DEVELOPMENT ENVIRONMENT**

To develop a webservice software package ready for deployment the following isneeded:

1. A Windows or Linux computer with Matlab 2006b or higher installed on it, (optional) Matlab compiler
2. Matlab Web service toolbox

Matlab compiler is not a requirement, but is recommended. Otherwise each web service instance will occupy a Matlab license. In compiled mode, multiple web service instances can run at no additional costs.

To deploy this web service software package you need the following:

1. A Windows or Linux machine with Matlab or Matlab Runtime installed
2. Apache Tomcat
   1. **Dataset**

Two DDoS datasets are used for evaluating the proposed attack detection technique. One of these datasets represents a network layer flooding attack while the other two represent different types of application level attacks. Each of these attack datasets are simulated for a period of 5 minutes: first by two minutes of normal activity, followed by three minutes of flooding. Each of these datasets is now described.

* 1. The ICMP dataset is based on the CAIDA DDoS Attack Dataset, which represents a real-world DDoS attack and contains approximately an hour(20:50:08 UTC to 21:56:16) of pseudonymised traces from a DDoS attack that occurred on August 4, 2007 . The traffic profile of the CAIDA dataset represents a flooding attack where the attacking machines send a large number of ICMP Echo Requests (ping) to the victim aimed at consuming the victim’s networking bandwidth. The simulated attack uses an identical packet rate and the same number of source IP addresses, as in the CAIDA dataset, to generate a 5-minute attack dataset containing one-way traffic i.e., directed against the target host
  2. Jkjk
  3. Jkj
  4. Ll

Table 4.1: Macro-level Statistics of the 1998 FIFA World Cup and the CAIDA  
dataset.

|  |  |  |
| --- | --- | --- |
| **Characteristic Features** | **CAIDA Dataset** | UCSL Daataset |
| Activity type | DDoS Attack |  |
| Duration | 66 min |  |
| Packets typ e | ICMP ECHO |  |
| Numb er of target(s) | 1 |  |
| Numb er of packets sen | 359,655,826 |  |
| File format | p cap |  |
| Size (uncompressed) | 5.3 GB |  |

Table 4.2: Micro-level statistics of the 1998 FIFA World Cup and the CAIDA  
dataset during the observation period.

|  |  |  |
| --- | --- | --- |
| **Characteristic Features** | **CAIDA Dataset** | UCSL Daataset |
| Observation time | 10 mins |  |
| **Numb er of Requests/Packets** | 74,478,486 |  |
| **Numb er of Source IPs** | 8,585 |  |
| Requests p er source IP | 8675.42 |  |
|  |  |  |
|  |  |  |
|  |  |  |

* 1. : **Dataset Pro cessing**

A subset of CAIDA and as used for the experiments described in this section. The dataset is a six hour network trace around the1st semi-final match of the CAIDA . The original dataset was first filtered to remove all traffic except HTTP GET requests. Secondly, only packets containing 200, 206 and 400 HTTP status codes were extracted and used as the input trace (processed dataset) for the experiments. Table 4. shows the highlevel characteristics of the original dataset and the processed dataset used in thesimulations

Table 5.1: Characteristic features of the dataset used for simulations

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Original Dataset** | **Pro cessed Dataset** |
| Numb er of requests | 29,662,465 | 19,615,109 |
| Numb er of unique source IPs | 79,033 | 78,646 |
| Numb er of unique source IPs | 11,885 | 11,839 |
| Numb er of unique source IPs | 4,089 | 2,376 |
| Numb er of unique source IPs | GET, POST, HEAD | GET |
| HTTP status co des | all | 200, 206 and 400 |
|  |  |  |

## In order to ensure that the same web-resources are downloaded during the simulation as were present in the original trace, a fake directory structure was created and installed on the target machine’s cloud aptachie web server . All the files within this directory structure used the same file name and contained the same amount of data as in the original traffic trace, although the contents were junk data.

**Incoming Traffic**

The incoming traffic for the simulated and the original

The total traffic volume of the simulated

**Source IP Addresses**

**The source IP address parameter of the original traces compared with the simulated traffic in two forms: the total number of unique source IPs, and the variations in the number of unique source IPs over time.**

## Hardware Platform

One of the major ECC requirement is to implement in high speciation of hardware; the complexity of district logarithms.

Previous studies employed relevant implementation of ECC software. Dedicated PC resources were used in implementing ECC in the study by Diffie Hellman ([Sankar, Subashri, & Vaidehi, 2011](#_ENREF_35)). Other studies show several types of hardware used ([Gupta, Sikka, & Katiyar](#_ENREF_15); [Hardie & Connell, 2004](#_ENREF_17); [Katiyar, Dutta, & Gupta, 2010](#_ENREF_22); [Shohdy, Elsisi, & Ismail, 2010](#_ENREF_41)).

Table 4.1 shows the PC specifications used in implementing our proposed ECC compiler.

Table 4.1: PC specifications used in implementation

|  |  |  |
| --- | --- | --- |
| Processor | Main Circuit Board | Memory Modules |
| 2.93 gigahertz Intel Core2 Duo.64 kilobyte primary memory cache 3072 kilobyte secondary memory cache 64-bit ready Multi-core (2 total) Not hyper-threaded | Board: ASUSTeK Computer INC. P5G41C-M LX Rev X.0x Bus Clock: 266 megahertz | 1024 Megabytes Usable Installed Memory  Slot 'DIMM A1' has 1024 MB Slot 'DIMM B1' has 2048 MB |

Table 5.1: Summary of the Datasets

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset | DDoS | Total | Purpose |
|  |  |  |  |
|  |  |  |  |

Features for All attacks

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **UDP Features** | **ICMP features** | **TCP Sync** | **TCP PSH** |
| 1 | No of packets | No of packets | No of packets | No of packets |
| 2 | Average packet size | Average packet size | Average packet size | Average packet size |
| 3 | No of bytes | No of bytes | No of bytes | No of bytes |
| 4 | Packet rate | Packet rate | Packet rate | Packet rate |
| 5 | Bit rate | Bit rate | Bit rate | Bit rate |
| 6 | Ratio udp | Ratio udp | Ratio udp | Ratio udp |
| 7 | Ratio tcp | Ratio tcp | Ratio tcp | Ratio tcp |
| 8 | Ratio icmp | Ratio icmp | Ratio icmp | Ratio icmp |
| 9 | prototype | prototype | prototype | prototype |
| 10 | Source count | Source count | Sync Count | Sync Count |
| 11 | Destination count | Destination count | Push count | Push count |
| 12 | - | - | Fin count | Fin count |
| 13 | - | - | Ack count | Ack count |

**4.6 IDPS Metrics and parameters**

Machine learning algorithms need deliver not only the answer but also the reliability of the answer under various conditions in order for the algorithms to be of any practical use. Most existing machine learning research, however, focuses on solution development with limited effort on systematically characterizing the reliability (precision) of the solution under different conditions. It is especially important to identify conditions where the algorithm's performance can be optimized and conditions where the algorithm may fail. This objective can be achieved by performance evaluation. Therefore, performance evaluation of a machine learning system is a fundamental issue in machine learning and receives increasing attention from machine learning researchers.

In this thesis, we will mainly use the well-established ROC analysis to assess the performance of different methods compared in our work. For over a decade, ROC analysis is gaining popularity more intensely also in the field of machine learning. First applications date back to late 1980’s when ROC curves were demonstrated to be applicable to the rating of algorithms [1]. In the present, these curves already represent one of the standard metrics for assessing machine learning algorithms. A detailed introduction to the use of ROC analysis in research (with stress on machine learning) may be found in [2].

ROC analysis in its original form is used to deal with two-class classification problems. In our work, the two classes refer to the **Normal Network Traffic** mode and the **Attack Network Traffic** mode. After the learning phase, a system should be able to predict a network traffic record to be either **Normal** traffic or **Attack** traffic.

Since predicted classes of given instances are not necessarily same as true classes, a matrix is used to keep a record of the number of prediction errors. This matrix is called ***a contingency table*** or ***a confusion matrix*** (since it represents the confusion between classes). There are four possible outputs for a classification of each instance, each individual test that is performed can yield one of four possible results, as follows:

* **True positive**: (also known as true acceptance or true match) occurs when an intrusion detection that should yield a correct result does so.
* **True negative**: (also known as true rejection or true non-match) occurs when an intrusion detection that should yield an incorrect result does so.
* **False positive**: (also known as false rejection, false non-match or type I error) occurs when an intrusion detection that should yield a correct result actually yields an incorrect one (in vision testing, usually the wrong class of output).
* **False negative**: (also known as false acceptance, false match, false alarm or type II error) occurs when an intrusion detection that should yield an incorrect result actually yields a correct one (e.g., finding that a face detector works on a picture of a coffee cup).

The number of correct classifier decisions thus lies on the main diagonal of a contingency table, while other table elements represent a number of misclassifications. By summing the values in cells that are diagonally adjacent, you can determine the overall accuracy of the model. One diagonal tells you the total number of accurate predictions, and the other diagonal tells you the total number of erroneous predictions.

A contingency table is also a source for calculating further knowledge evaluation measures, such as the true positive rate (TPR), false positive rate (FPR), true negative rate (TNR) and false negative rate (FNR). In some fields of study, TPR is also called **sensitivity** or **recall** as well as the term **specificity** denotes TNR. Using the abbreviations defined above, (i.e. TP as the number of true positives, TN as the number of true negatives, FP as the number of false positives, and FN as the number of false negatives), then a number of quantities, mostly originating from the information retrieval field, can be derived from them. Those in most wide spread use and also used in this thesis are:

* **Sensitivity** or **Recall or TPR**: the proportion of actual positive cases which are correctly identified. Defined as **TP / (TP + FN).**
* **Specificity or TNR**: the proportion of actual negative cases which are correctly identified. Defined as **TN/(TN+FP).**
* **Fall-Out or FPR:** Defined as **FP/(FP + TN) = 1 – Specificity.**
* **False Negative Rate (FNR):** Defined as **FN/(TP + FN) = 1 - Sensitivity**.
* **Positive Predictive Value (PPV)** or **Precision**: the proportion of positive cases that were correctly identified. Defined as **TP/(TP + FP).**
* **Negative Predictive Value (NPV)**: the proportion of negative cases that were correctly identified. Defined as **TN/(TN + FN).**
* **Accuracy**: the proportion of the total number of predictions that were correct. Defined as **(TP + TN) / (TP + FP + TN + FN).**
* **F measure:** the harmonic mean of precision and recall. Defined as
  + **F Score = 2 x Precision x Recall / (Precision + recall)**

An ROC graph for original two-class problems is defined as a two-dimensional plot which represents **TPR** (sensitivity) on y-axis in dependence of **FPR** (= 1-specificity) on x-axis. Performance of a particular classifier, represented by its sensitivity and specificity, is denoted as a single point on an ROC graph. There are some basic characteristic points on a graph of this type. The point with coordinates (0,0) (**TPR** = 0, **FPR** = 0) represents a classifier which never predicts a positive class. While such a classifier would never misclassify a negative instance as positive, it is usually not a good choice, since it would never make a single correct classification of a positive instance neither. Its relative in the point (1,1) represents the opposite situation (**TPR** = 1, **FPR** = 1) as it classifies all instances as positive, thus also producing a possibly high number of false positives. The classifiers in (0,0) and (1,1) are called default classifiers. In (0,1) the perfect classifier is located (**TPR** = 1, **FPR** = 0). While it is not realistic to expect such performance from any classifier on a real-world problem it represents a goal at which the induction of classifiers should aim. Classifiers which are located on the ascending diagonal of an ROC graph have the same performance as random guessing. For such classifiers one can conclude that they have no information about the problem. Useful classifiers are located above the ascending diagonal. Those under it are performing worse than random guessing. Nevertheless, they can be made useful very easily by inverting their predictions. Such classifiers are said to have useful information but are employing it in a wrong way.

An ROC curve is a curve on an ROC graph with start point in (0,0) and end point in (1,1). Drawing procedure for this curve depends on the type of classifiers one want to evaluate. In view of the amount of returned information, classifiers may roughly be divided into three groups: discrete (predicting a class membership), scoring (predicting a class score) and probability estimating (predicting a class probability). The score is defined as posterior probability (not necessarily calibrated) of the positive class. A class score offers more information than a class membership. Similarly, the amount of information contained in a class probability is higher than in a class score. Main reason for the use of scores is that good probability estimates are not always available, for example, in a case of small amount of learning data. The meaning of scores may be interpreted as follows: if a classifier returns scores for two instances where the score of the first instance is greater than the score of the second, this indicates that the first instance has a higher probability as well. A disadvantage, however, is that scores from different classifiers cannot be compared to each other in contrast to predicted probabilities which have a common interpretation.

The important property of ROC curves lies in that they measure the capability of classifiers to output good scores. Analyzed classifiers thus do not have to produce exact probabilities, all they have to do is discriminate positive instances from negative ones. Another useful feature of ROC curves is that they remain unchanged when altering class distribution. Class distribution is the proportion of positive instances to negative instances. An ROC curve is based on **TPR** and **FPR** values and since **TPR** and **FPR** are each calculated from values of one column, ROC curves are consequently independent of class distribution. The fact that ROC curves take into consideration sensitivity (i.e. **TPR**) and specificity (i.e. **TNR** = 1 − **FPR**) also represents an advantage of these curves over simpler evaluation measures, such as classification accuracy.

An example of an ROC graph with four different ROC curves each representing one classifier is given in Fig. 1. Classifier A is by far better than the other three classifiers. ROC curves of classifiers B and C cross – each of these two is superior to the other for some deployment contexts (i.e. combinations of class distribution and misclassification costs). Classifier D is of no use as its performance is no better than chance.

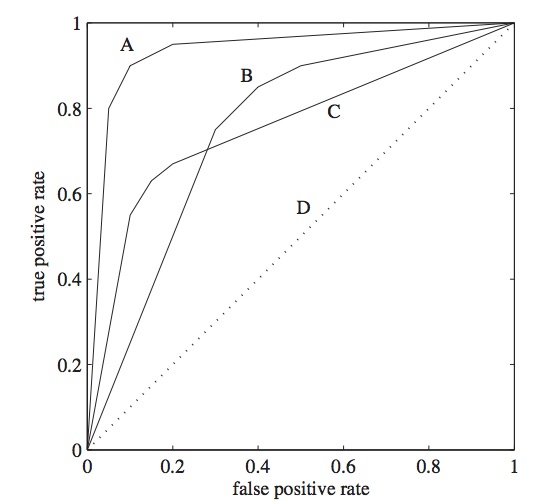


Figure 1. ROC graph with 4 ROC curves

In real classification scenario, ROC curves tend not to be as straightforward as those shown in Figure 1. Often the curves to be compared cross each other, and then it is up to the user to decide which curve represents the best method for their application. For example, Figure 2, shows that **alg1** may be superior to **alg2** when a high true-positive rate is required but **alg2** may be preferred when a low false-positive rate is required.

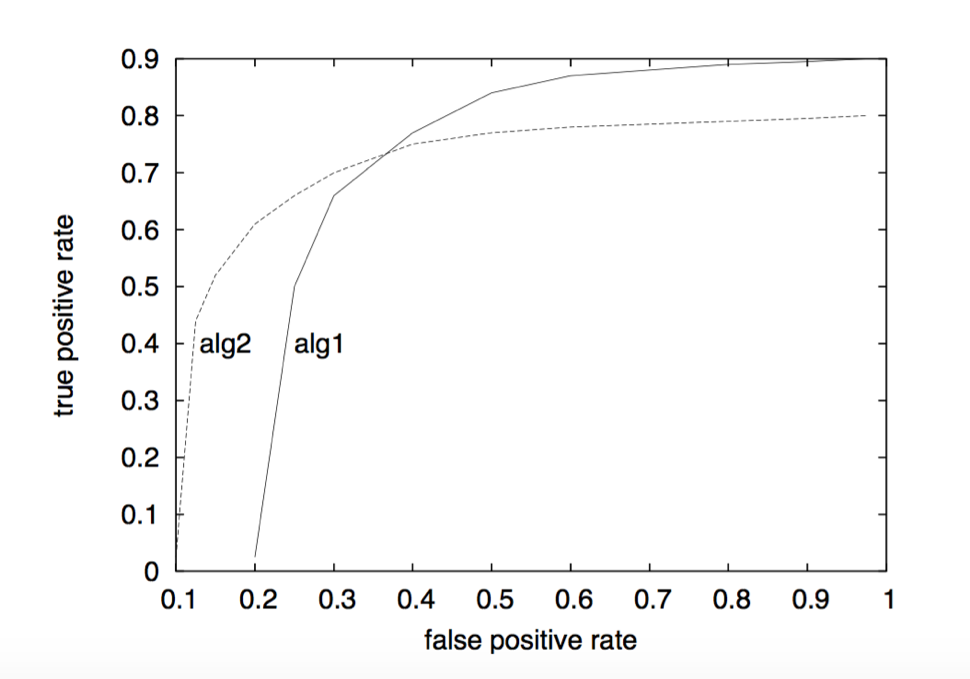


Figure 2.

**Area Under ROC Curve (AUC).**

There are situations where one may want to compare more than two ROC curves. If the number gets high, visual comparison of these curves may become a non-trivial task. This is especially true in the case that many of them intersect (meaning that the underlying classifiers do not dominate each other). To this end, another measure of classification model performance has been introduced in ROC analysis: Area Under the ROC Curve (**AUC**). The purpose of this measure is to summarize individual ROC curves in the form of numerical information. Comparison of the quality of classifiers thus reduces to comparison of numerical values.

Statistical meaning of the AUC measure is the following: AUC of a classifier is equivalent to the probability that the classifier will evaluate a randomly chosen positive instance as better than a randomly chosen negative instance. This statistical property is often referred to as the probabilistic form of the **AUC** measure. It originates from signal detection theory and was introduced to machine learning community mainly through the use of ROC analysis in radiology.

AUC is related to other well-known measures. It is equivalent to the Wilcoxon statistic and to the Mann-Whitney statistic [3]. Further, the AUC is related to the Gini index [4]. In [5] the relation between statistical properties of the AUC and those of the Wilcoxon statistic are discussed in detail.

Value of the AUC measure may be calculated using the formula below:

where the sum passes over all pairs of one positive and one negative instance. Value of the variable *difference* is equal to the difference between the score of a positive and the score of a negative instance (in exactly this order) in an individual pair. Conditional statement is in the form

**if (condition; a; b)**

where **a** is the value returned when a condition is met and **b** the value returned when a condition is not met.

To be convinced that the value of AUC of some ROC curve may be calculated using the above equation, one may consider y and x axes of an ROC graph to be divided to P and N sections, respectively, where P is the number of all positive instances and N the number of all negative instances. An ROC graph may thus be seen as composed of P · N rectangles (i.e. P rows and N columns). If we then have a set of instances sorted according to their scores in decreasing manner, the value of AUC is calculated as follows. For every positive instance, we count the number of negative instances which have lower score than the chosen positive instance. We accumulate the sum. At the end, we divide the final sum with the number of all pairs of one positive and one negative instance (= P · N), and finally we obtain the value of AUC.

The gain of one positive instance may thus be regarded as one row on an ROC graph. The scalar value of the AUC metric thus exactly corresponds to what may graphically be seen as the portion of the area of an ROC graph lying under an ROC curve.

Value of the AUC lies on the interval from 0 to 1. Since any useful classification model should lie above the ascending diagonal of an ROC graph, AUC of such models exceeds the value of 0.5.

In [6] the use of AUC as a performance measure for machine learning algorithms is investigated. AUC and overall accuracy measures are compared. It is shown that AUC has some convenient features: standard error decreasing when AUC and the number of test samples increase; it is independent of a decision threshold; it is invariant to prior class probabilities; and it indicates to what degree the negative and positive classes are separated.

While there are several possible measures that allow users to measure association between sensitivity and specificity (various information measures, such as mutual information gain etc.), AUC additionally provides a geometrical interpretation of the ROC graph. As there is no general rule specifying which measure has advantages or disadvantages in particular problem domains and using particular models, it is up to a user to select such a measure which has a required interpretation.

* 1. **Evaluating Controlled Environment**
  2. **Attacks results**
  3. **UDP Attack on cloud computing**
     1. **PCA**

The performance of the Intrusion Detection and Prevention System (IDPS) based on PCA method can be evaluated quantitatively using the metrics described in Section 4.6.

The input features are extracted from in total 504 records of UDP packages after PCA operation. Therefore the input matrix is a 5x504 matrix defining 5 attributes of 504 UDP data. The target input is a 504 matrix where each column indicates a correct category with a one in either element 1 (normal) or element 2 (attack).

For PCA only method, the input feature set is used directly to train a standard feedforward neural network that can be used to detect abnormal network traffic patterns in IDPS. The standard neural network that is used is a two-layer feedforward network, with a sigmoid transfer function in the hidden layer, and a softmax transfer function in the output layer. The number of hidden neurons is set to 10. The number of output neurons is set to 2, which is equal to the number of categories in the target vector (i.e. the normal network traffic and the attack network traffic).

One of the problems that occur during neural network training is called overfitting. The error on the training set is driven to a very small value, but when new data is presented to the network the error is large. The network has memorized the training examples, but it has not learned to generalize to new situations.

To address the overfitting problem, the general practice is to first divide the data into three subsets. The first subset is the training set, which is used for computing the gradient and updating the network weights and biases. The second subset is the validation set. The error on the validation set is monitored during the training process. The validation error normally decreases during the initial phase of training, as does the training set error. However, when the network begins to over fit the data, the error on the validation set typically begins to rise. The network weights and biases are saved at the minimum of the validation set error.

The third subset is the test set. The test set error is not used during training, but it is used to compare different models. It is also useful to plot the test set error during the training process. If the error on the test set reaches a minimum at a significantly different iteration number than the validation set error, this might indicate a poor division of the data set.

For the evaluation carried out in this thesis, validation and test data sets are each set to 15% of the original data. With these settings, the input network feature vectors and output target vectors will be randomly divided into three sets as follows:

* 70% are used for training.
* 15% are used to validate that the network is generalizing and to stop training before overfitting.
* The last 15% are used as a completely independent test of network generalization.

The following figure visualises the neural network used to evaluation the different methods in this thesis, which is configured as described above.

It can be seen clearly, that the feedforward network consists of a series of three layers in the setup. The first layer has a connection from the network input. Each subsequent layer has a connection from the previous layer. The final layer produces the network's output.

The diagram also reveals some other important characteristics of the neural network in question. For example, the total input data set is divided randomly divided into the three subsets using the division parameters specified in the above, i.e. with the ratios for training, testing and validation are 0.7, 0.15 and 0.15, respectively. The Scaled Conjugate Gradient schema is used to train for the optimal values of network weights and bias. The training performance is evaluated during the training process using Cross Entropy metric calculated based on network responses.

The summary diagram also indicates that it took in total 11 iterations for the neural network to converge after the validation errors have increased for 6 consecutive iterations (Validation Checks).

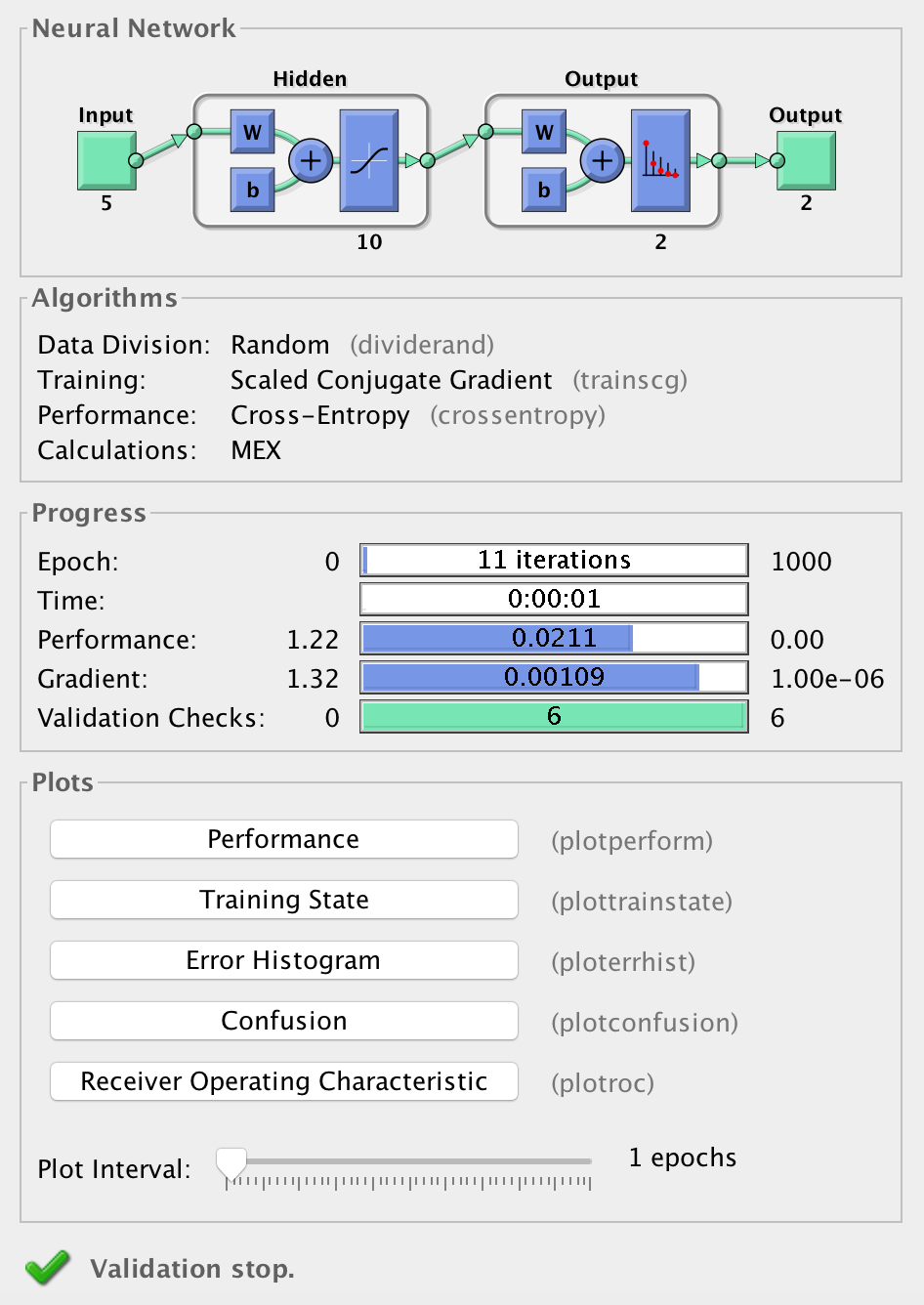


Figure Neural Network Structure

The classification results using trained neural network based on PCA method is shown in Figure 2 in the form of confusion matrix.

A Confusion Matrix is a visual performance assessment of a classification algorithm in the form of a table layout or matrix. Each column of the matrix represents predicted classifications and each row represents actual defined classifications. The diagonal cells show for how many (and what percentage) of the examples the trained network correctly estimates the classes of observations. That is, it shows what percentage of the true and predicted classes match. The off diagonal cells show where the classifier has made mistakes. The column on the far right of the plot shows the accuracy for each predicted class, while the row at the bottom of the plot shows the accuracy for each true class. The cell in the bottom right of the plot shows the overall accuracy.

Using the evaluation metrics defined in Section 4.6, the confusion matrix represents the metric scores in a compact way so that it also makes easy to calculate other representative evaluation measures such as Recall, Precision, Accuracy and F-Measure etc.

Figure 2 shows the confusion matrices for training, testing, and validation, and the three kinds of data combined. As mentioned before, there are 352 network records (70%) used for neural network training purpose, 76 network records (15%) used for validation and test respectively. For each confusion matrix, the first two diagonal cells show the number and percentage of correct classifications by the trained network. Take Training Confusion Matrix as example, 175 network traffic packages are correctly classified as normal. This corresponds to 49.7% of all 352 training samples. Similarly, 173 network traffic packages are correctly classified as attack. This corresponds to 50% of the total training samples. Overall, 98.9% of the predictions on the training samples are correct and 1.1% are wrong classifications.

The network outputs are very accurate, as you can see by the high numbers of correct responses in the green squares and the low numbers of incorrect responses in the red squares. The lower right blue squares illustrate the overall accuracies.

****

Figure Confusion Matrices for different data set

As a result, the trained model made in total 497 correction predictions, including 252 normal traffic and 245 attack attempts. There are in total 7 instances where the model misclassifies the real attacks to be normal traffic (false negative). This means the overall classification result has achieve 98.6% of accuracy, with the overall error rate to be 1.4%.

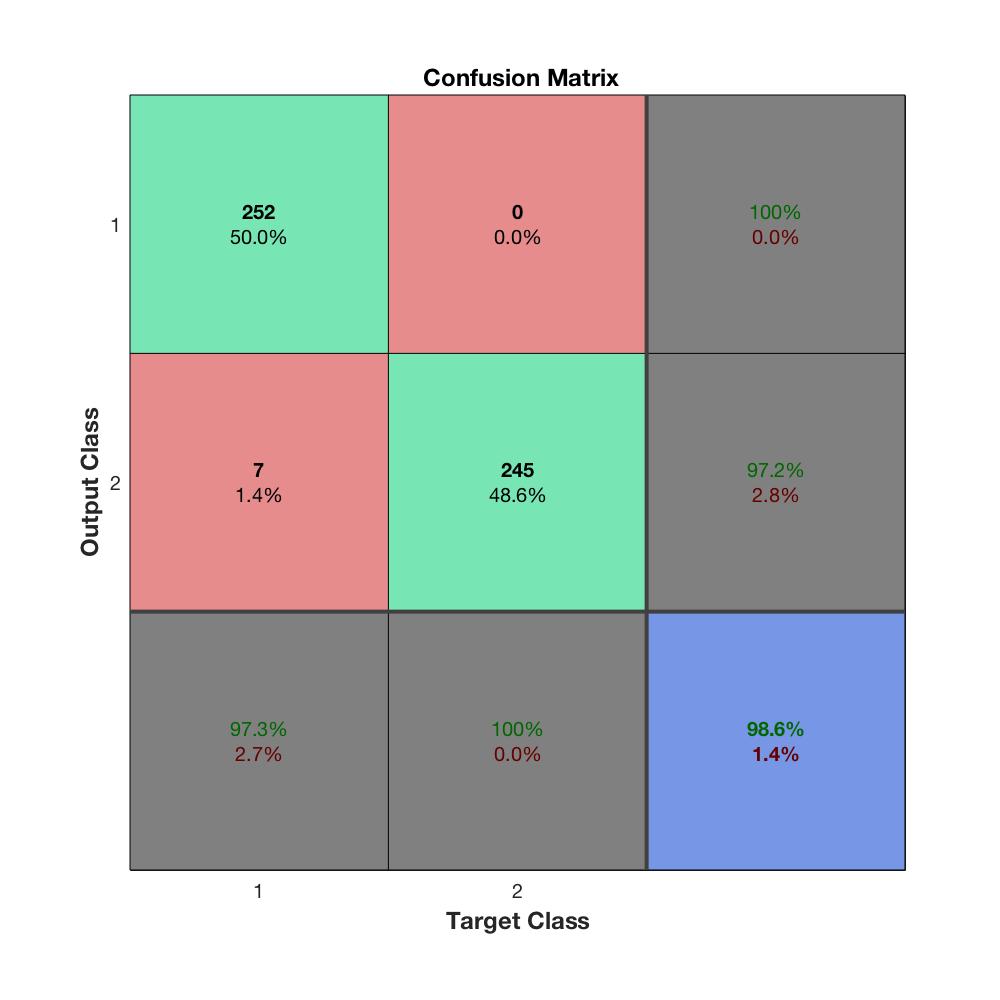
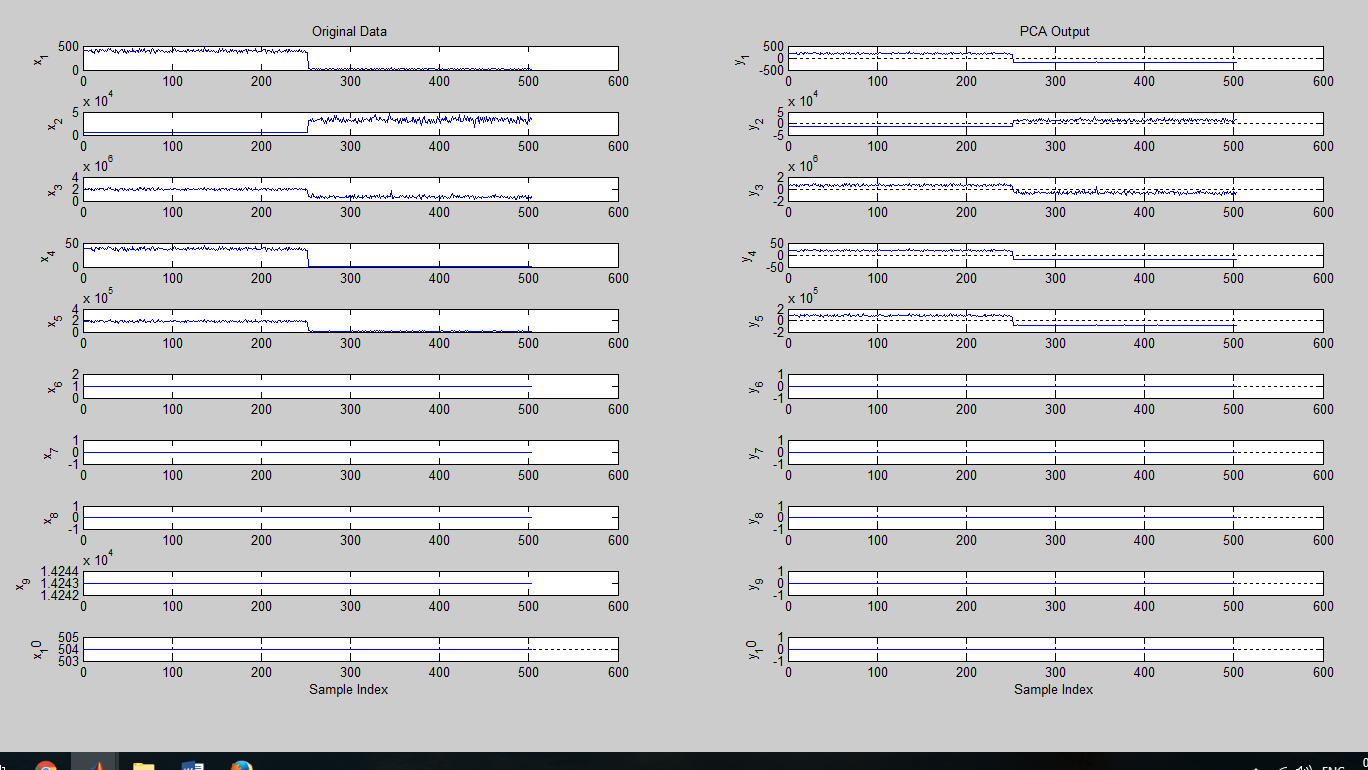
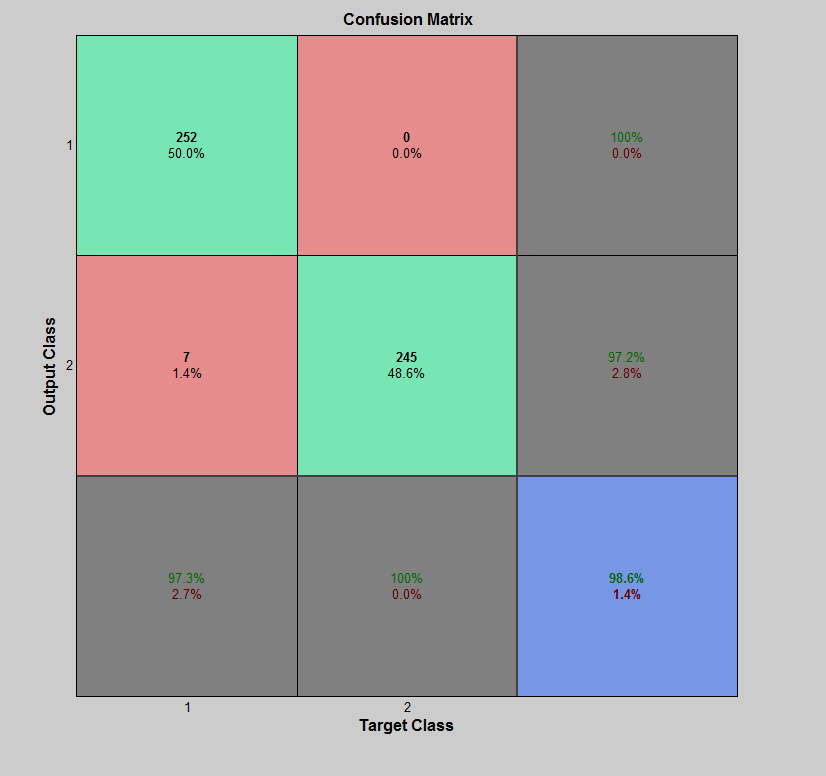
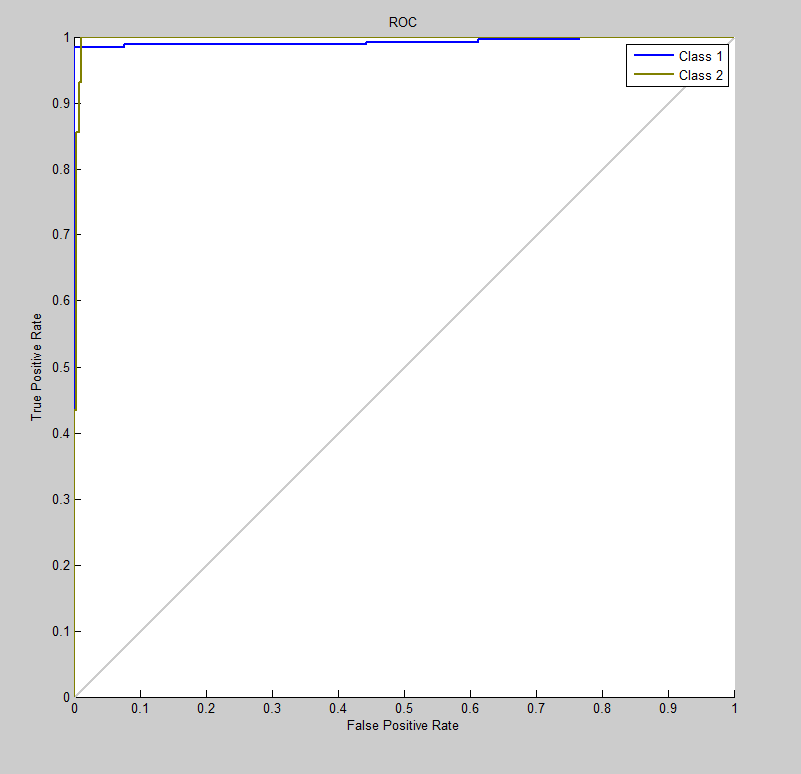
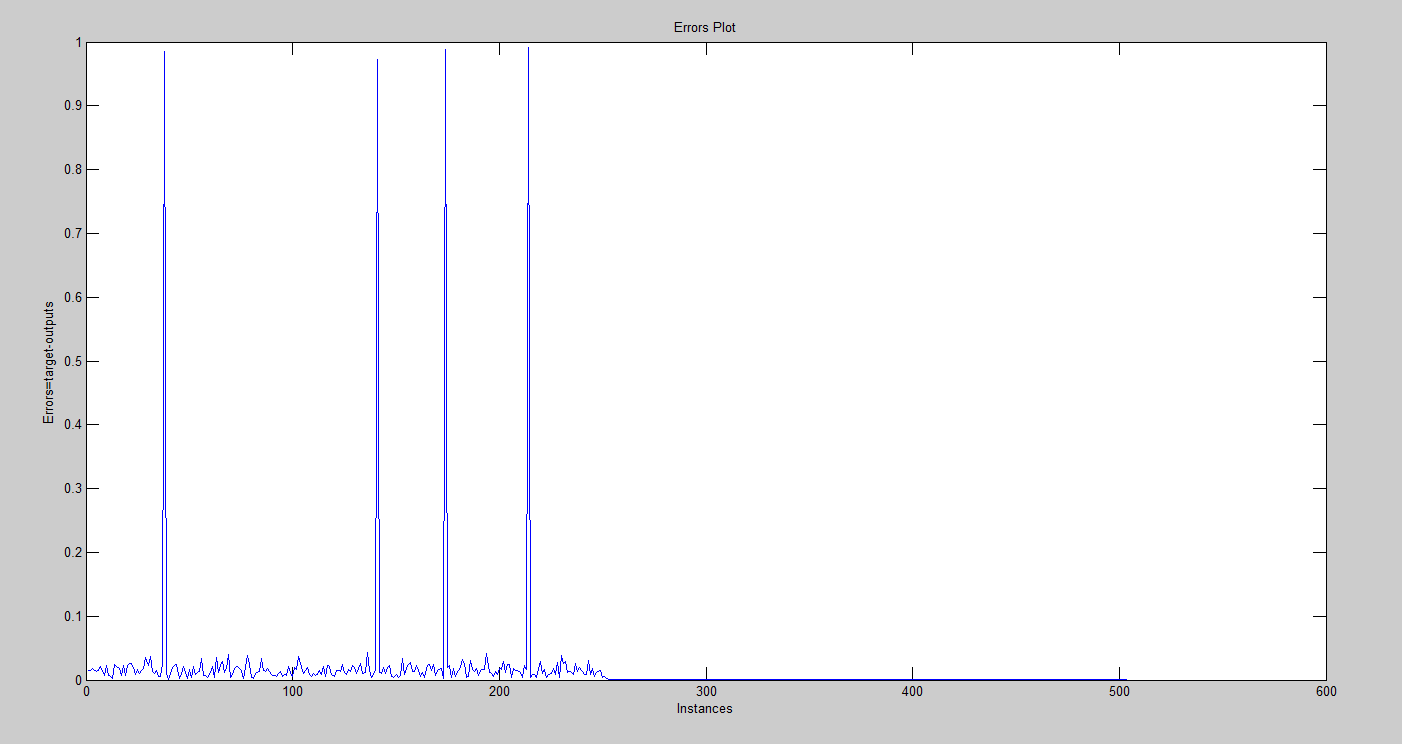
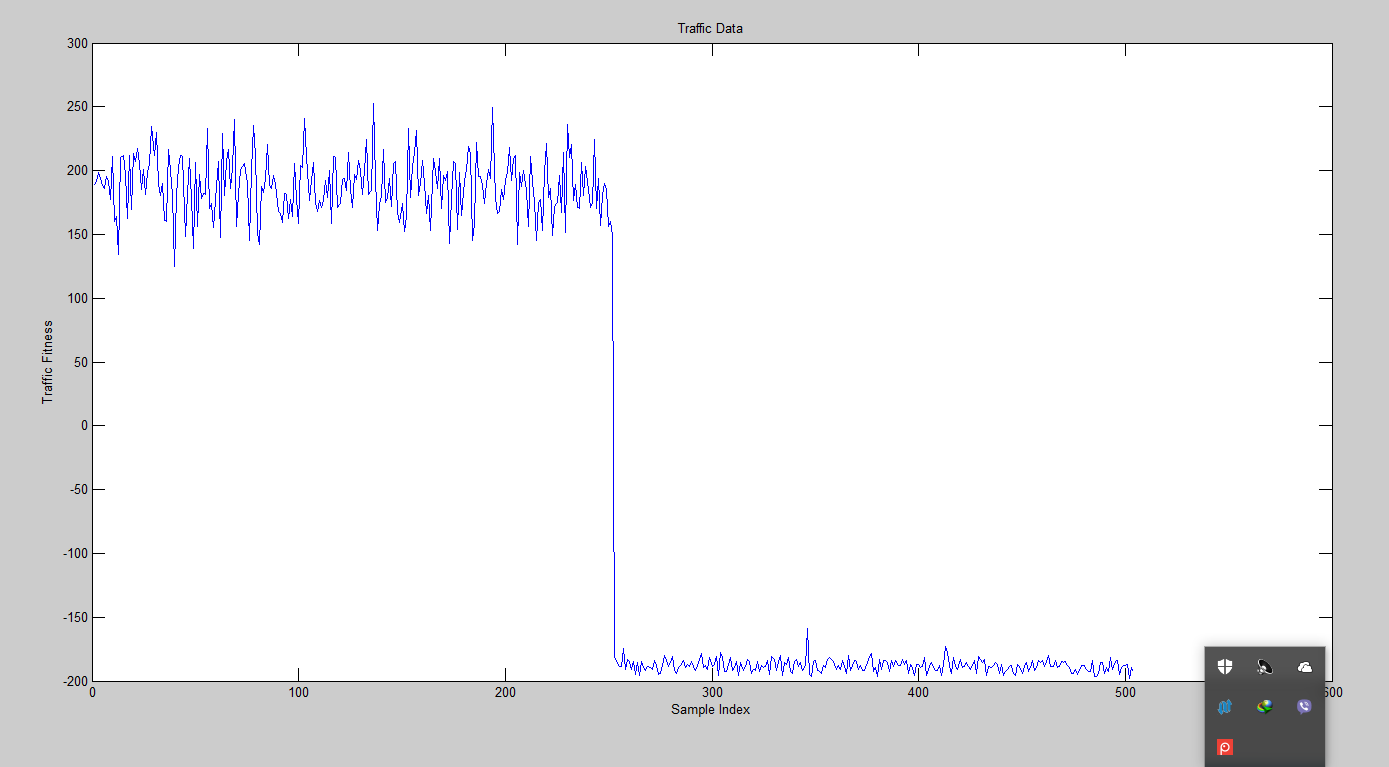
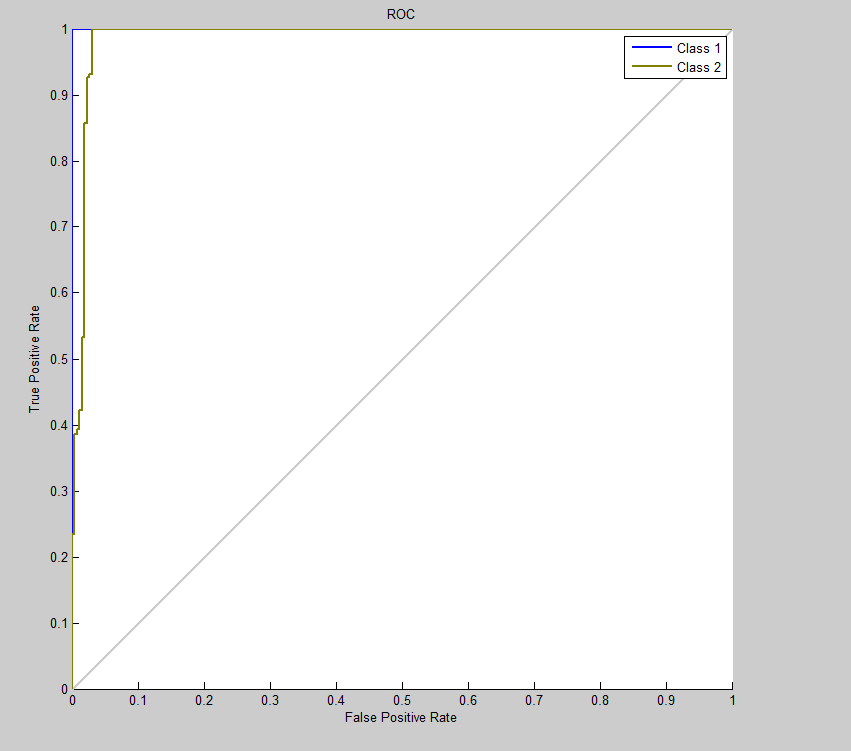


Figure Confusion Matrix for PCA only method



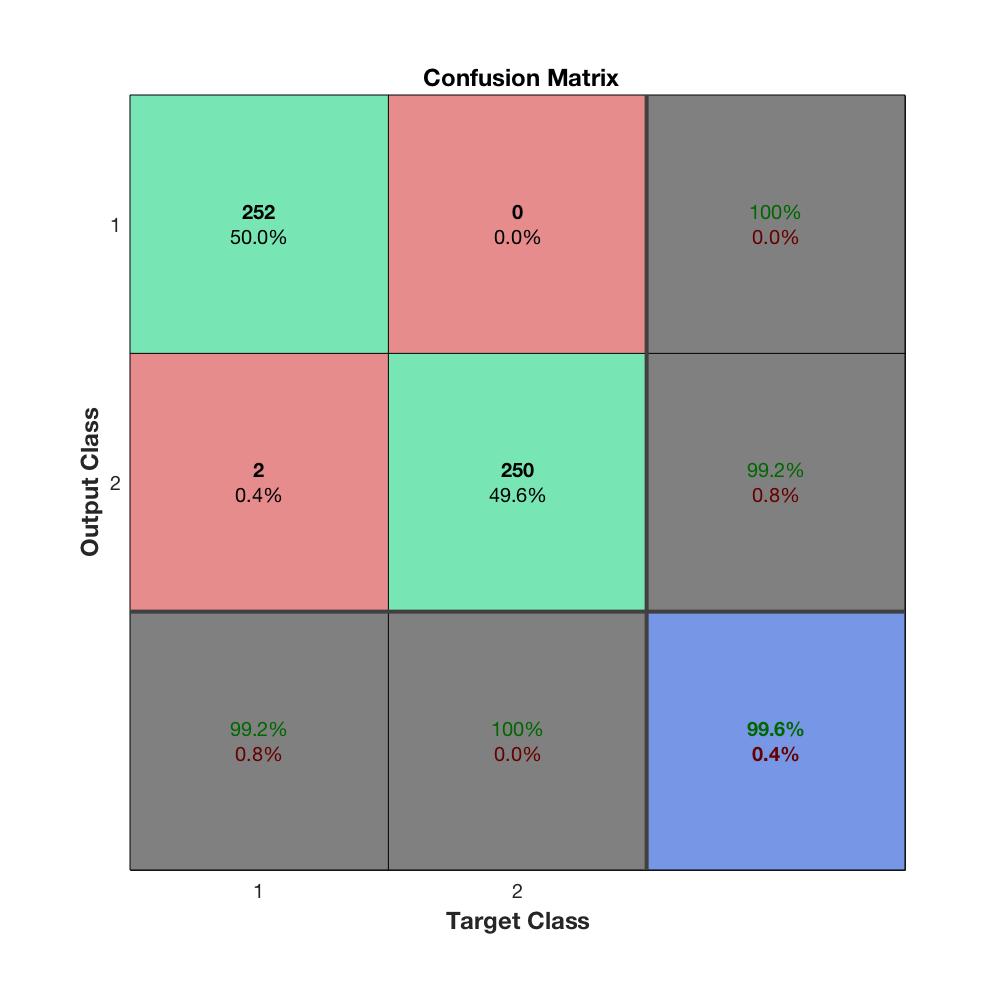
* + 1. **LDA**

The prime difference between LDA and PCA is that PCA does more of feature classification and LDA does data classification. In PCA, the shape and location of the original data sets changes when transformed to a different space whereas LDA doesn?t change the location but only tries to provide more class separability and draw a decision region between the given classes.This method also helps to better understand the distribution of the feature data.

****

The above figure shows the confusion matrices for training, testing, and validation, and the three kinds of data combined. As mentioned before, there are 352 network records (70%) used for neural network training purpose, 76 network records (15%) used for validation and test respectively. For each confusion matrix, the first two diagonal cells show the number and percentage of correct classifications by the trained network. Take Validation Confusion Matrix as example, 39 network traffic packages are correctly classified as normal. This corresponds to 50% of all 76 validation samples. Similarly, 35 network traffic packages are correctly classified as attack. This corresponds to 46% of the total validation samples. Overall, 97.4% of the predictions on the validation samples are correct and 2.9% are wrong classifications.

The network outputs are very accurate, as you can see by the high numbers of correct responses in the green squares and the low numbers of incorrect responses in the red squares. The lower right blue squares illustrate the overall accuracies.

****

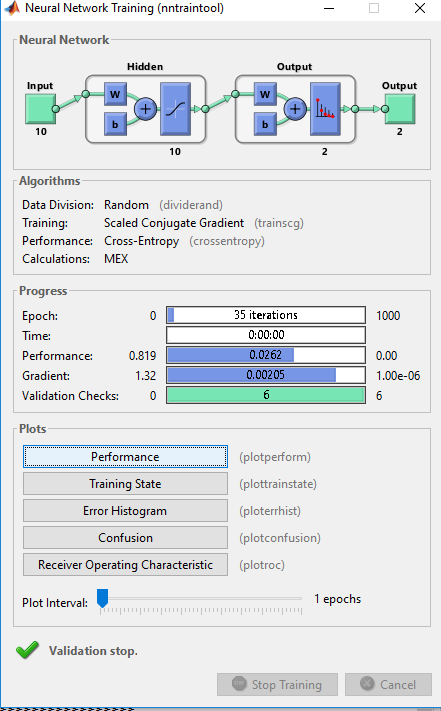
In the above figure, the first two diagonal cells show the number and percentage of correct classifications by the trained network. For example, 252 biopsies are correctly classified as normal. This corresponds to 50% of all 504 network traffic record. Similarly, 250 cases are correctly classified as attack. This corresponds to 49.6% of all network traffic.

2 of the normal traffic records are incorrectly classified as attack and this corresponds to 0.3% of all 504 network traffic records in the data. In this case, all of the attack records are correctly classified so there is no miss detection in this case.

Out of 252 normal predictions, 100% are correct and there is no wrong detections. Out of 252 attack predictions, 99.2% are correct and 0.8% are wrong. Out of 254 normal cases, 99.2% are correctly predicted as normal and 0.8% are predicted as attack. Out of 250 attack cases, 100% are correctly classified as attack and 0% are classified as normal.

As a result, the trained model made in total 502 correction predictions, including 252 normal traffic and 250 attack attempts. There are in total 2 instances where the model misclassifies the normal traffic record to be attack traffic (false positive). This means the overall classification result has achieve 99.6% of accuracy, with the overall error rate to be 0.4%.

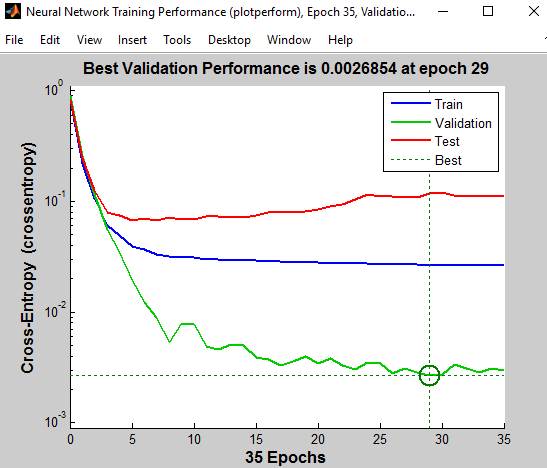
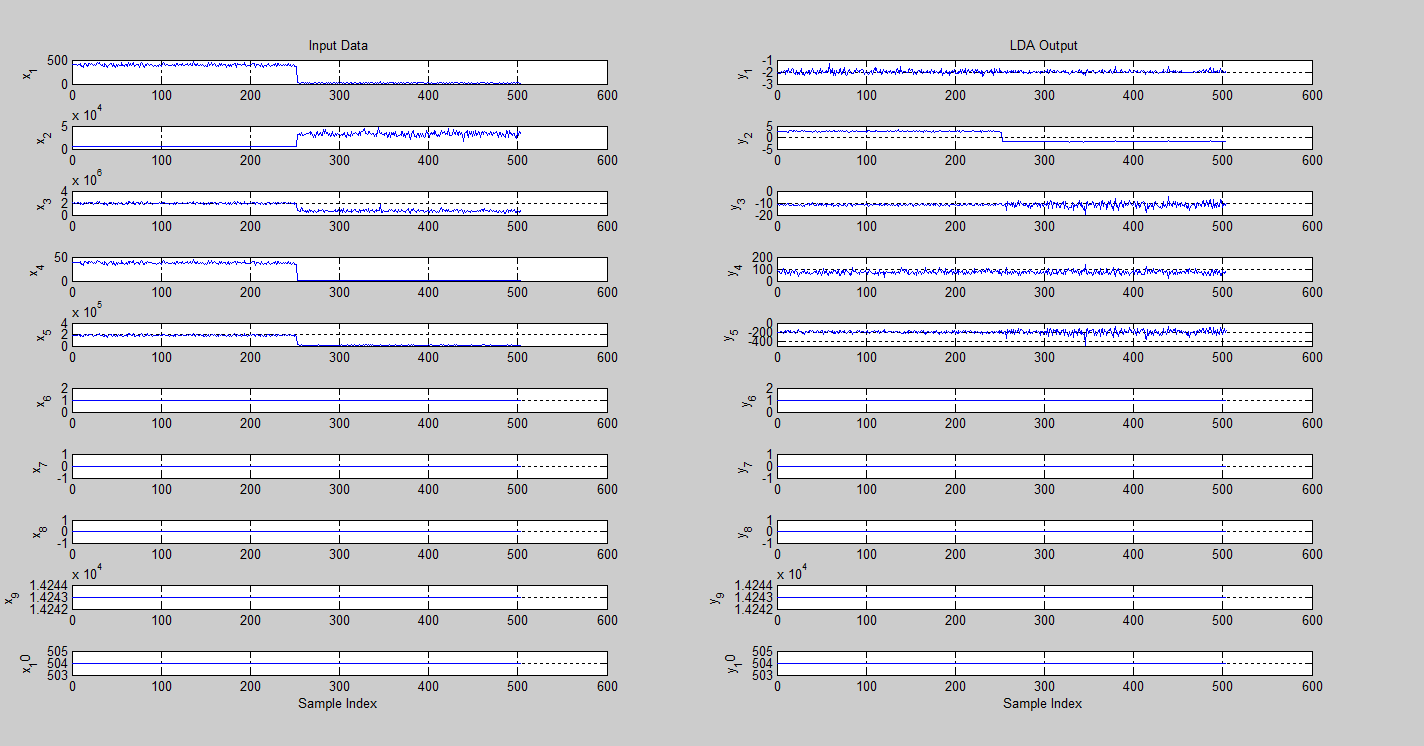
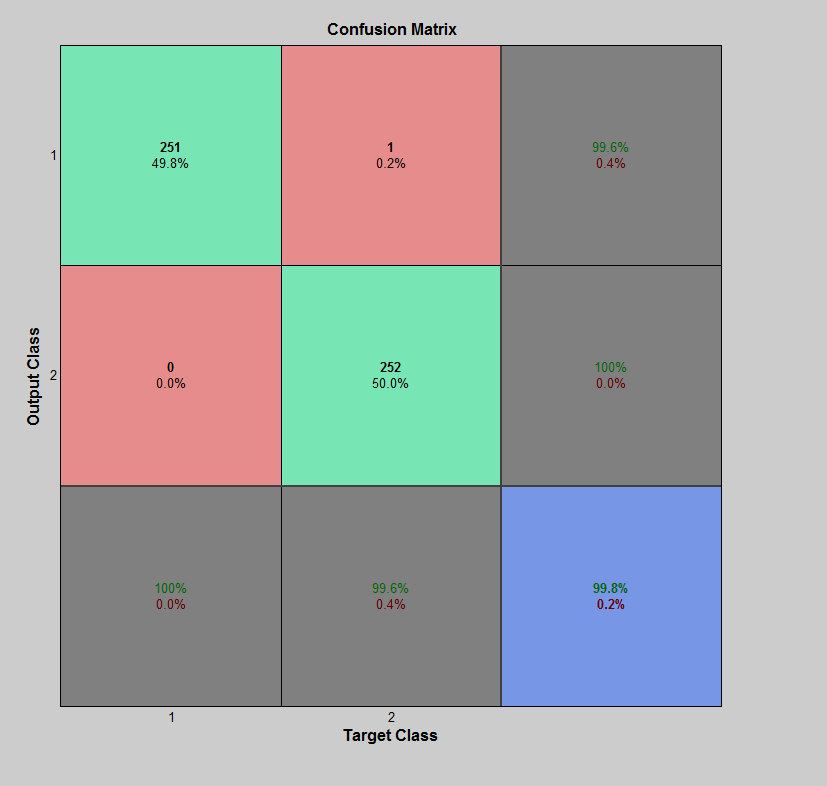
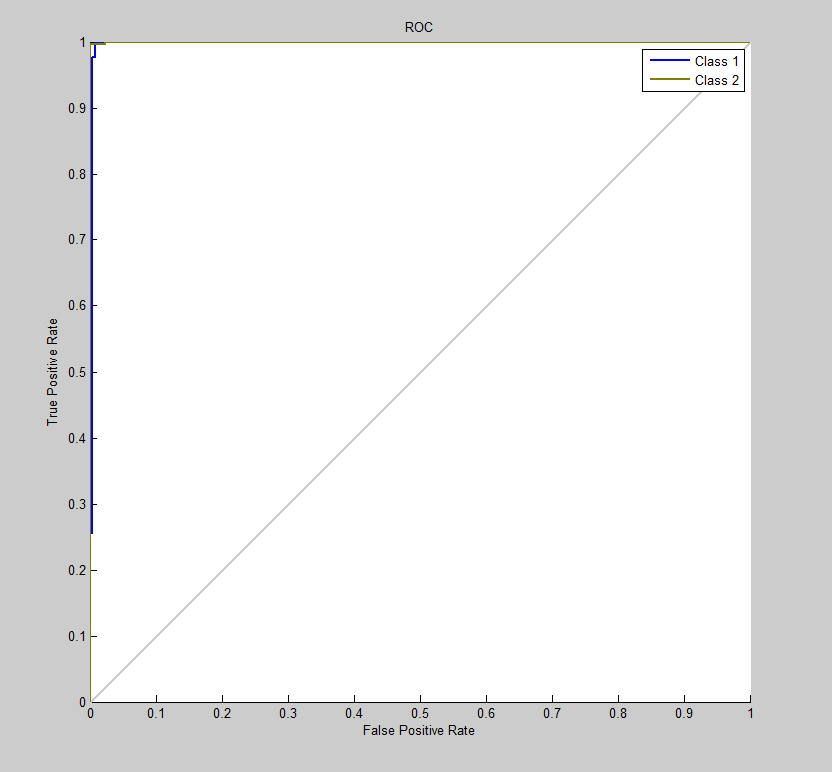
Overall, 97.6% of the predictions are correct and 2.4% are wrong classifications.



The configured feed forward neural network has 3 layers in total. It consists of one hidden layer only in our work. The reason to choose this simple structure is because we want to focus the comparison on the different methods used to compress the effectiveness and efficiency of the input data and to select a subset of features (i.e. Feature Selection).

All methods are using the same neural network with the same structure. This way, the only difference between different techniques in question will be their ability to reduce the redundant information in the very large input data space. This guarantee a fair comparison for the methods we used in our work.

The output layer of the neural network has only two nodes, corresponding to the two output classes of our data: Normal Traffic and Attack Traffic.

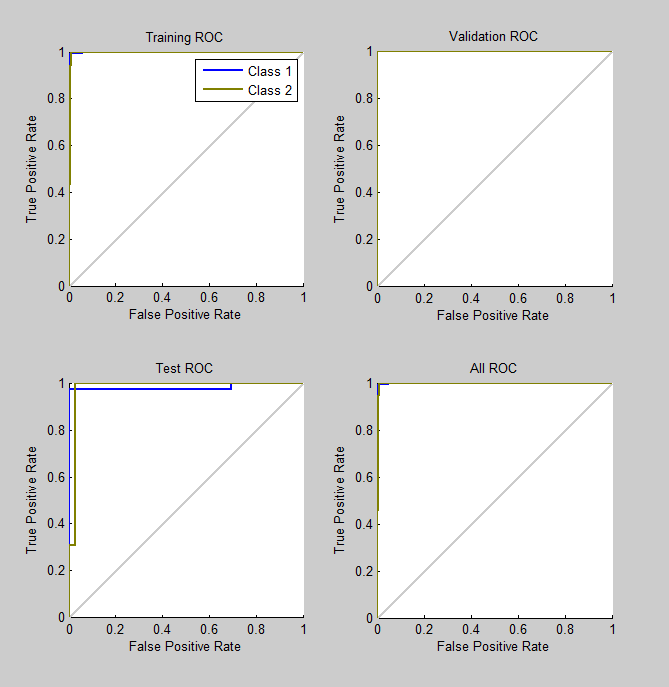
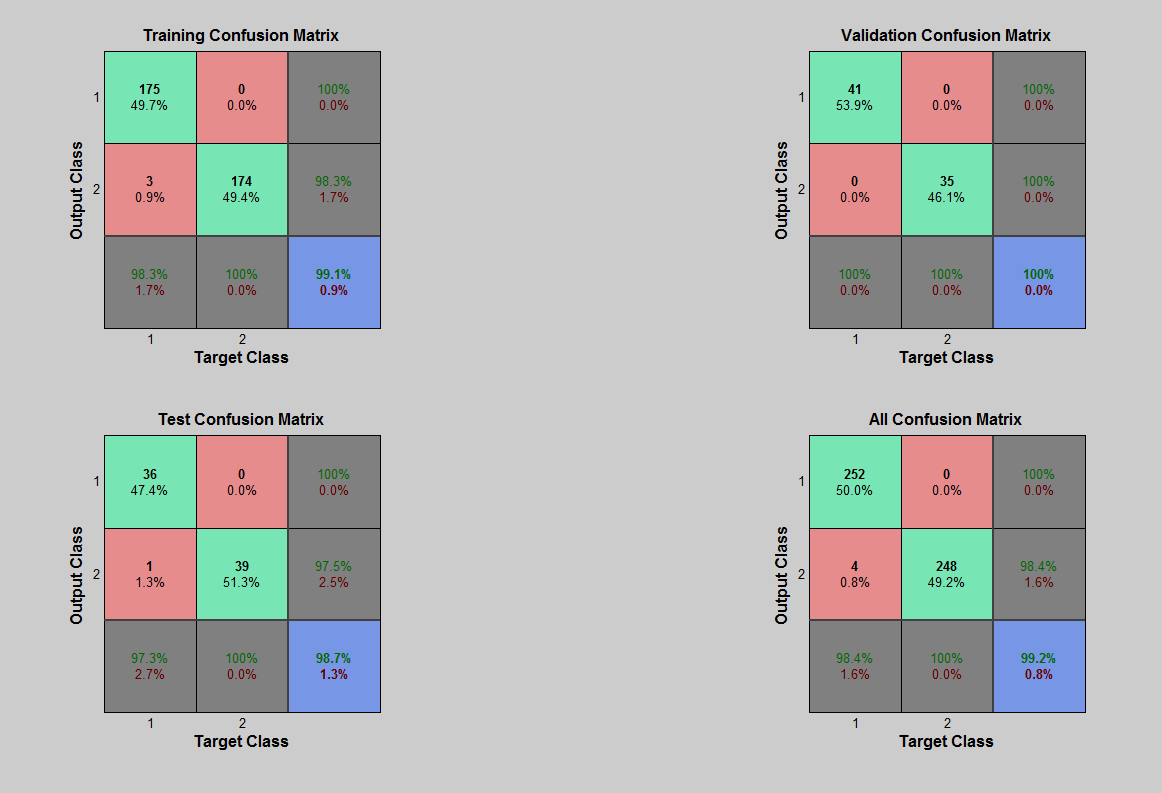
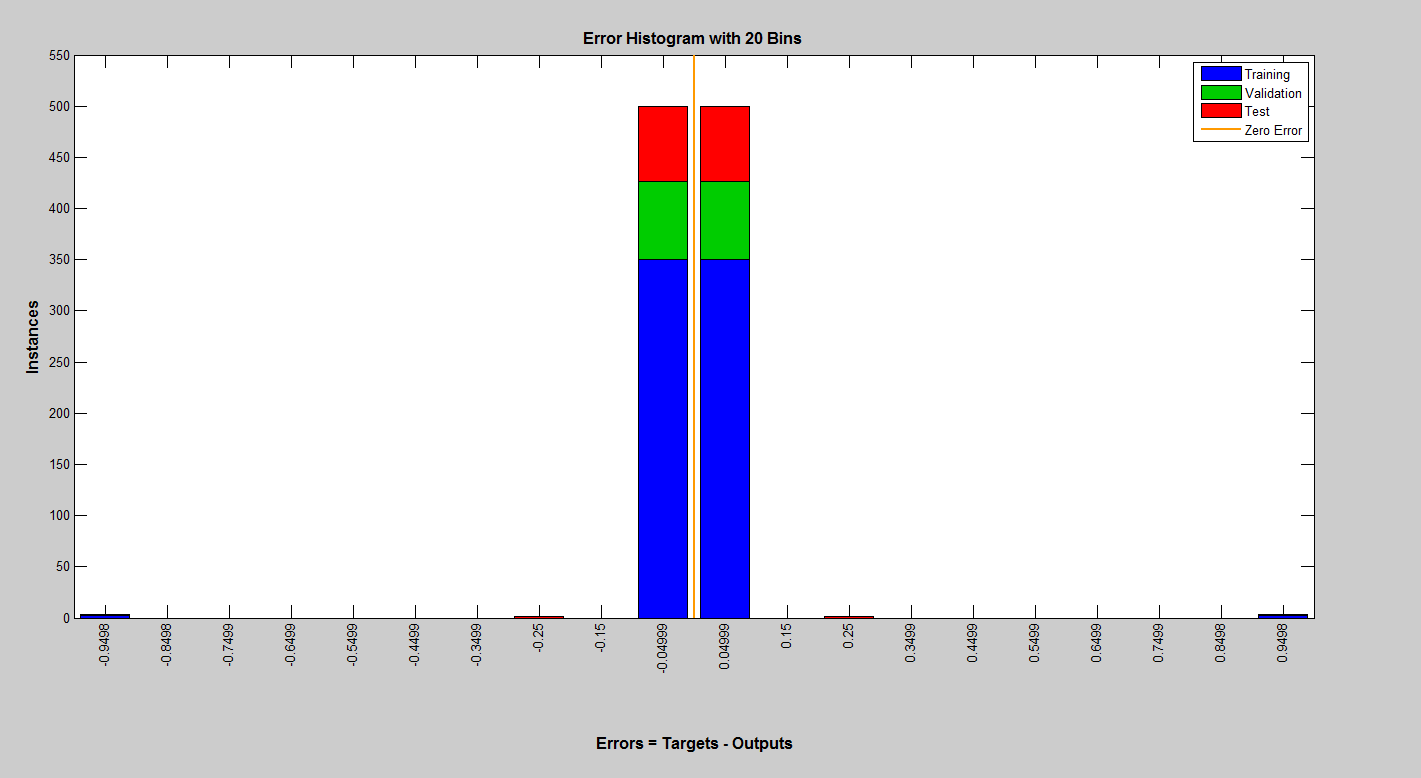
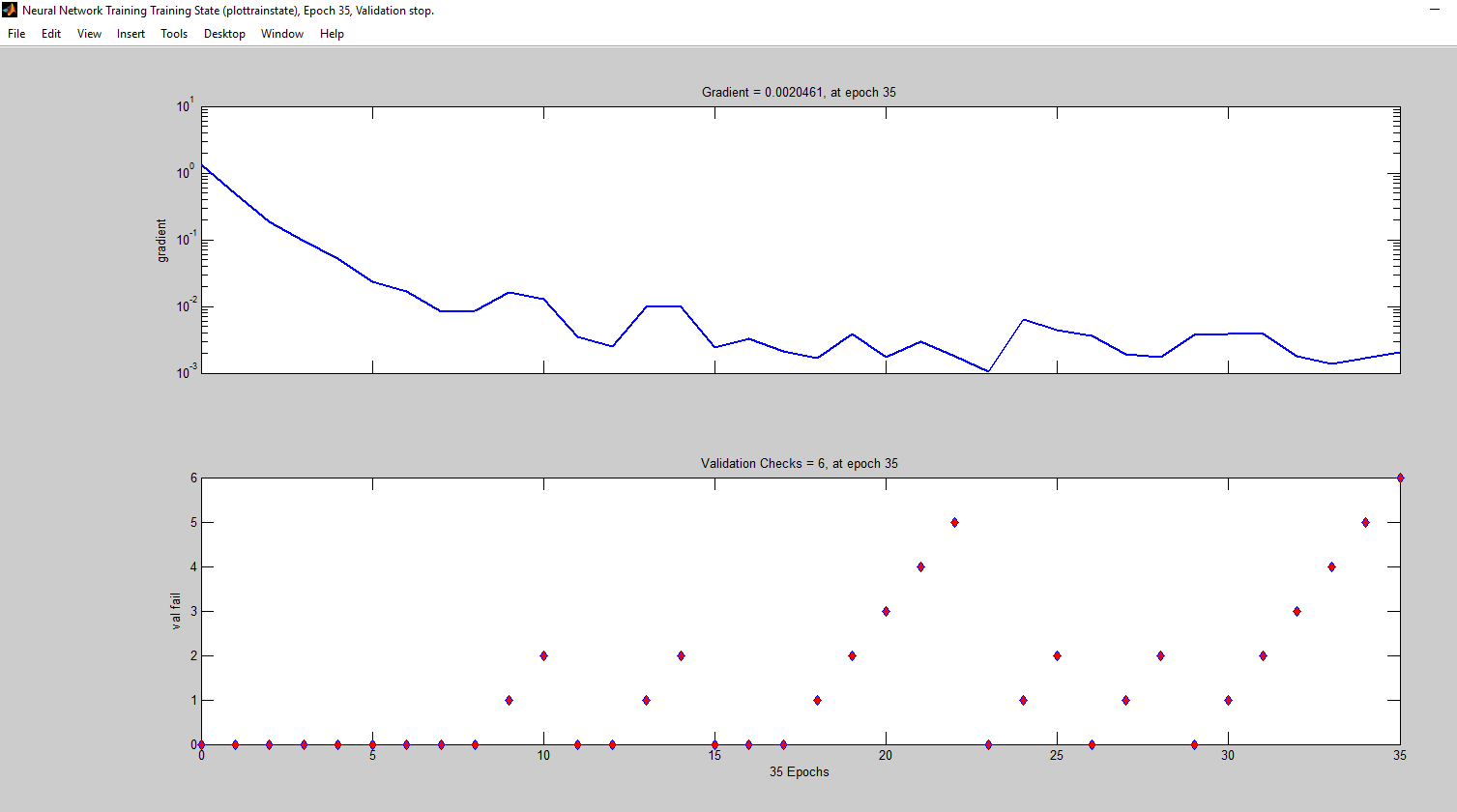


It consists of tabular instances shown as adjacent rectangles erected over discrete bins.

The graphical representation of the visual impression of the distribution of Errors (i.e. Targets vs. Outputs) is shown as the Error Histogram plot for the given data in the above

Figure shows the performance of the system taken into account training, validation and test data set. It shows that the best validation performance was xxxx seconds at epoch 41.

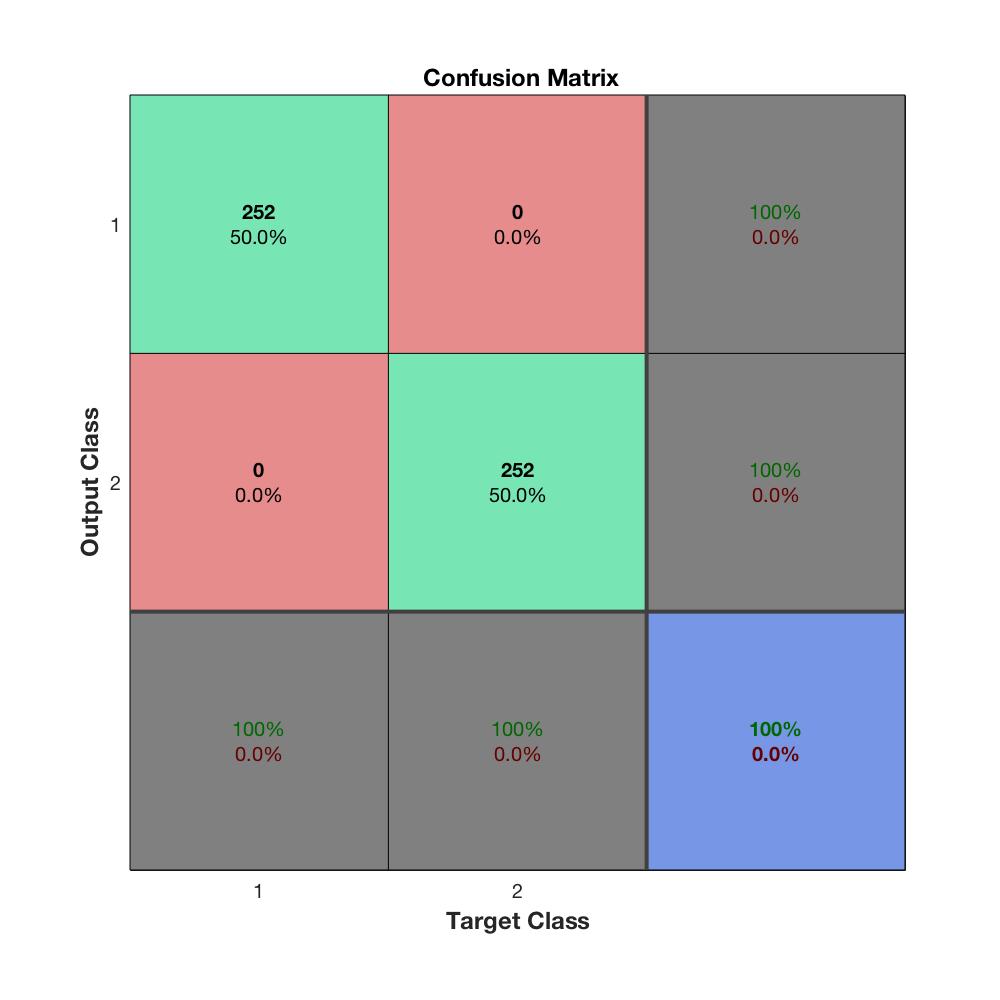
Figure xxx shows the neural network training state plot. It also shows validation check at epoch xxx and highlights that there is no validation failure up to this epoch.



As one can easily observed from the plotted ROC curves for Training ROC, Validation ROC, Test ROC and All ROC, the classified outputs of this method is as expected. One can see the curve is hugged close the top-left corner.

* + 1. **Proposed model (PCA+LDA+ALO)**

****

****

As can be seen easily from the above figures, including confusion matrix for training, validation and test data set, the proposed method has achieved 100% classification accuracy for all cases.

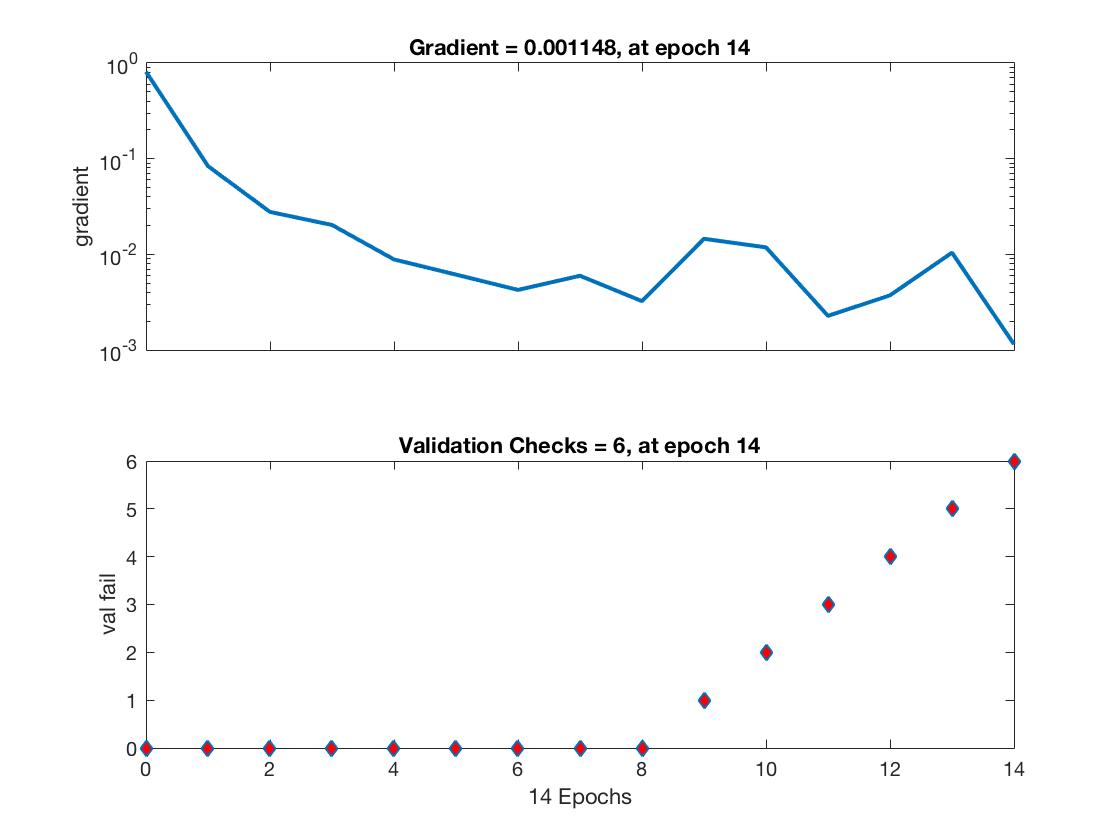
The selected features of each method are classified using Neural Network configured in this thesis, and the obtained classification accuracies of the testing network traffic data set are shown in Figure below. The following points can be deduced:

The feature selection methods were able to achieve classification accuracy similar to that of the LPR baseline feature vector with far less number of features (|Sj| < 15 features for GA, and |Sj| < 10 features for ALO).

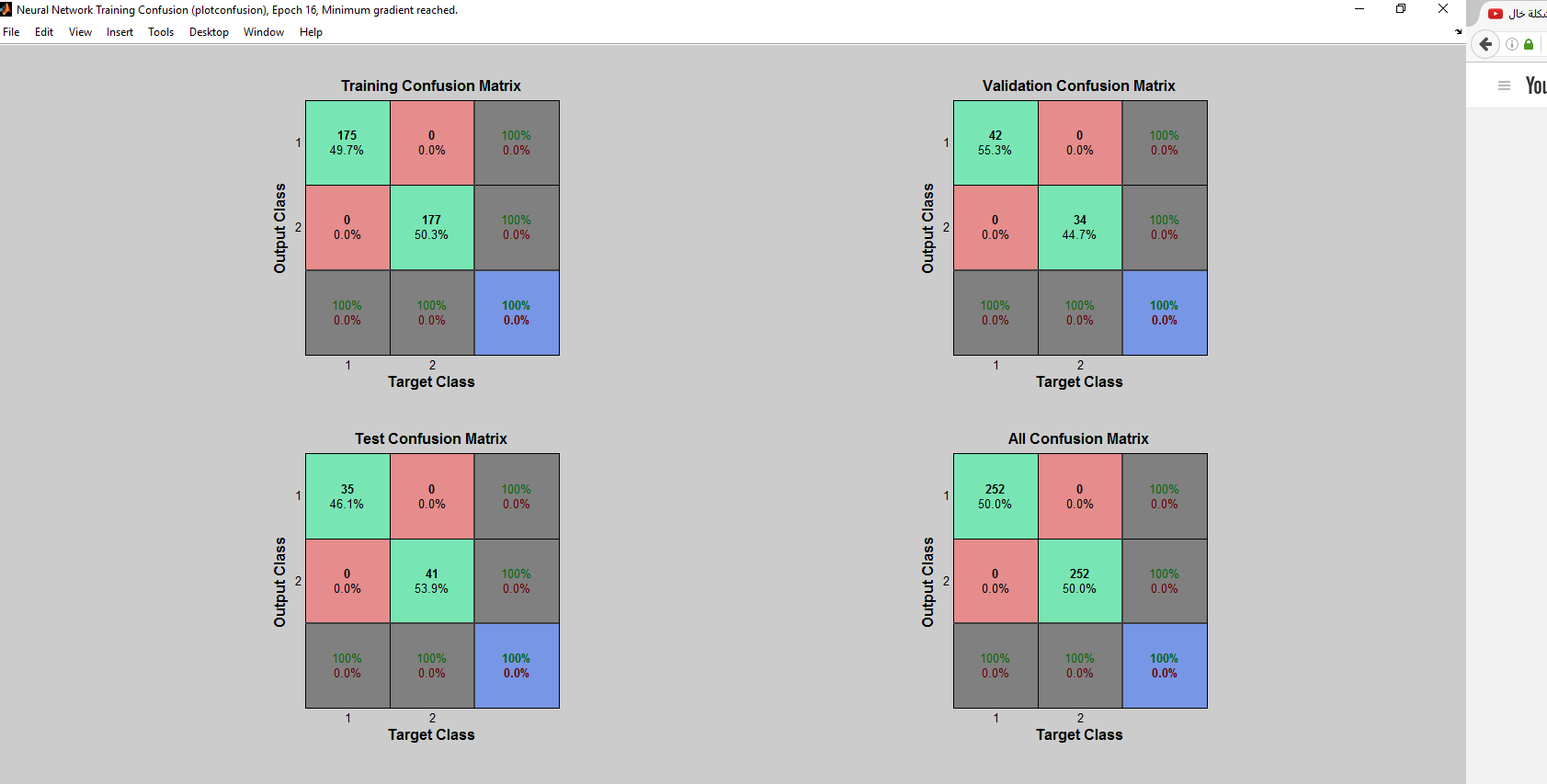
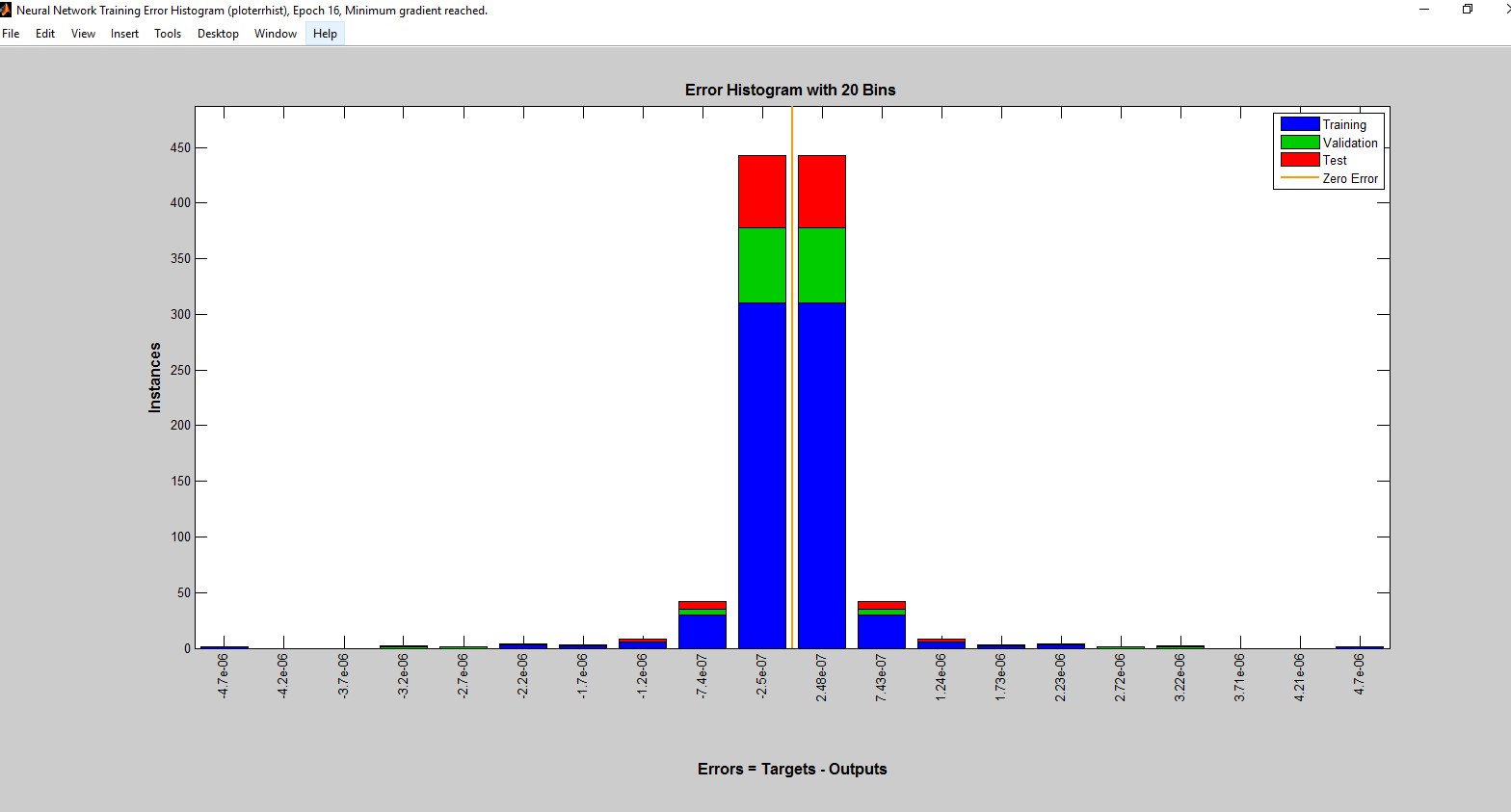
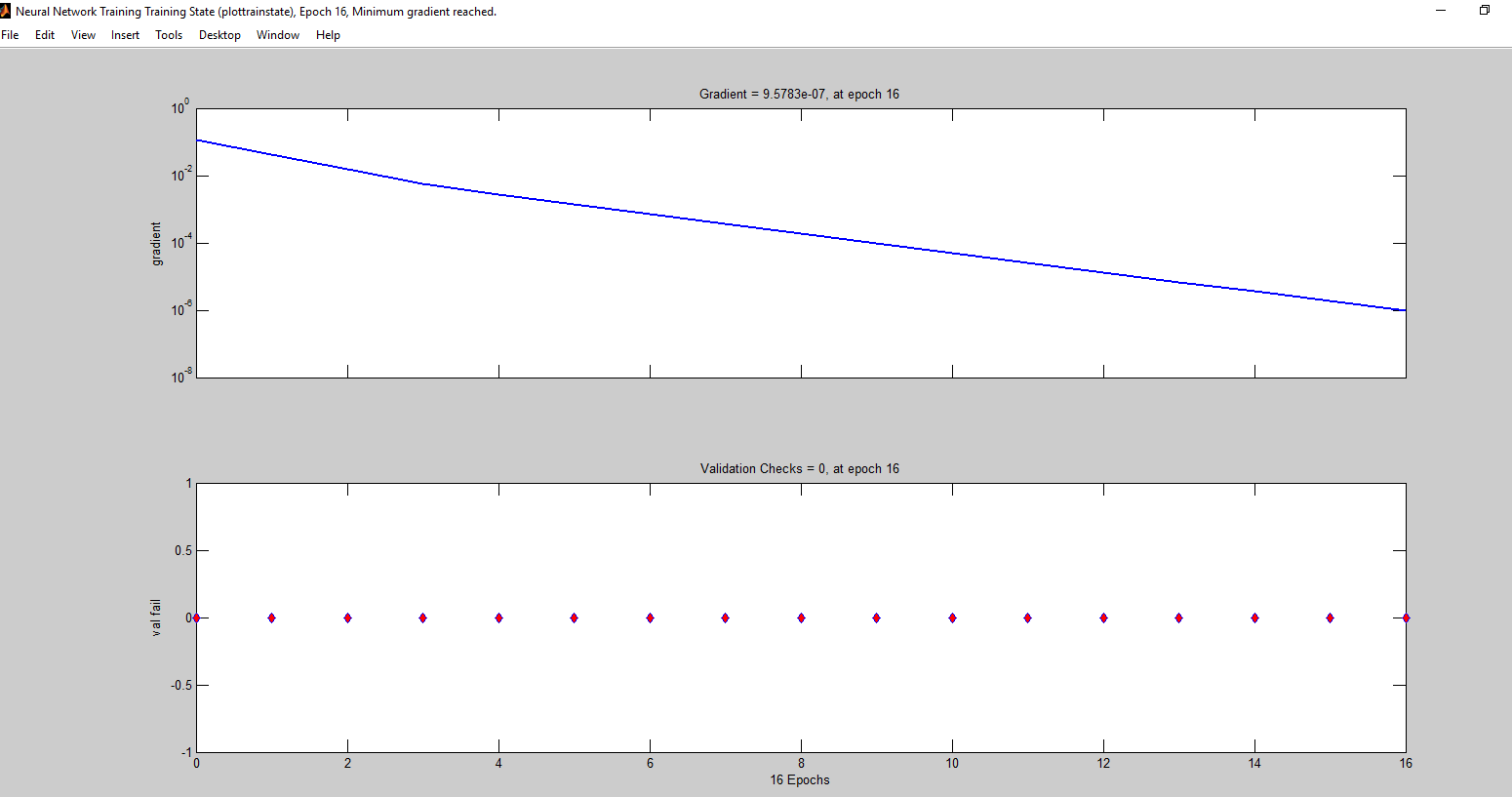
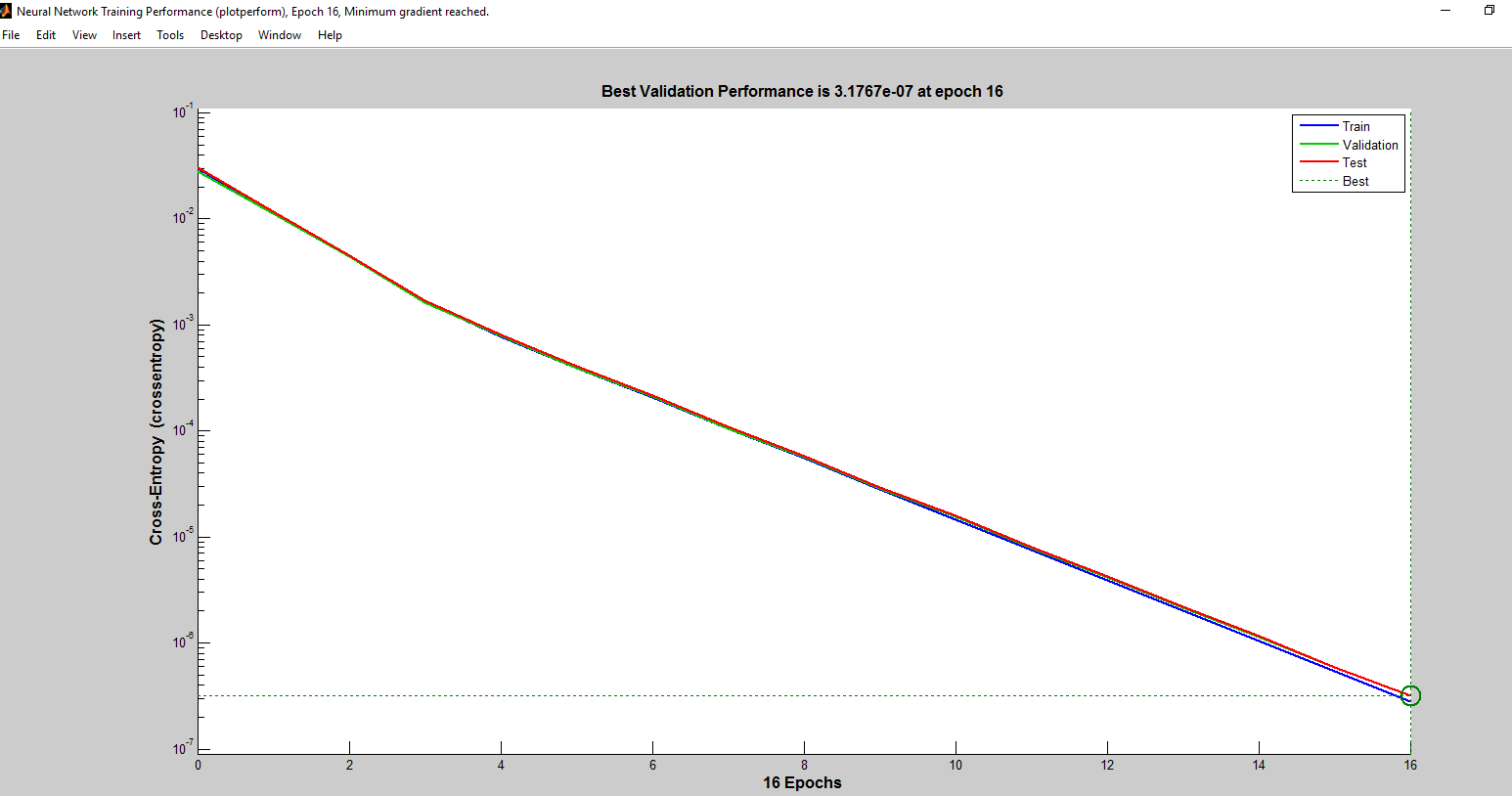
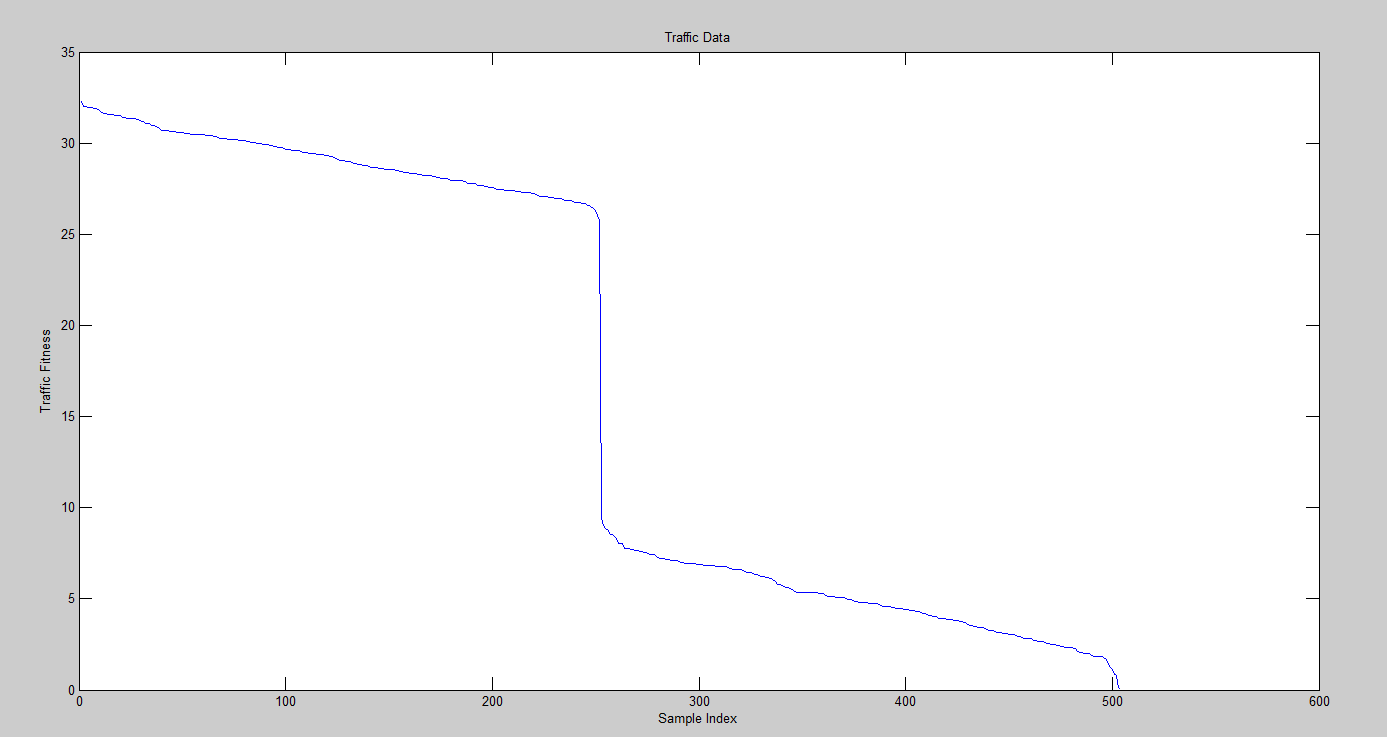
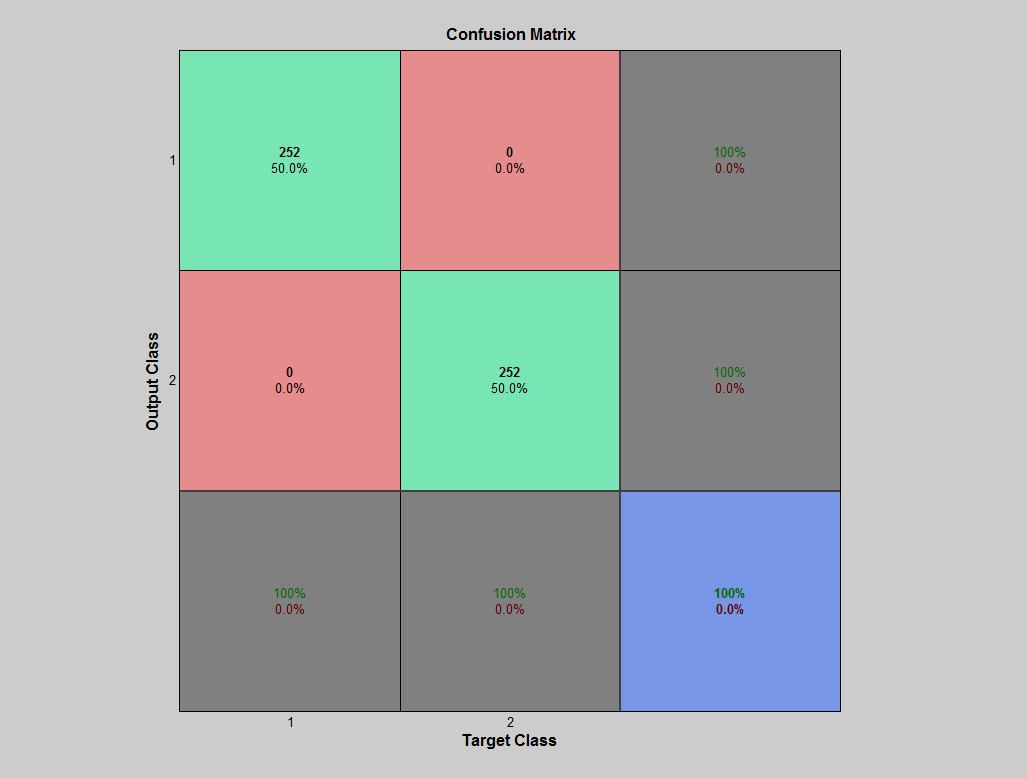
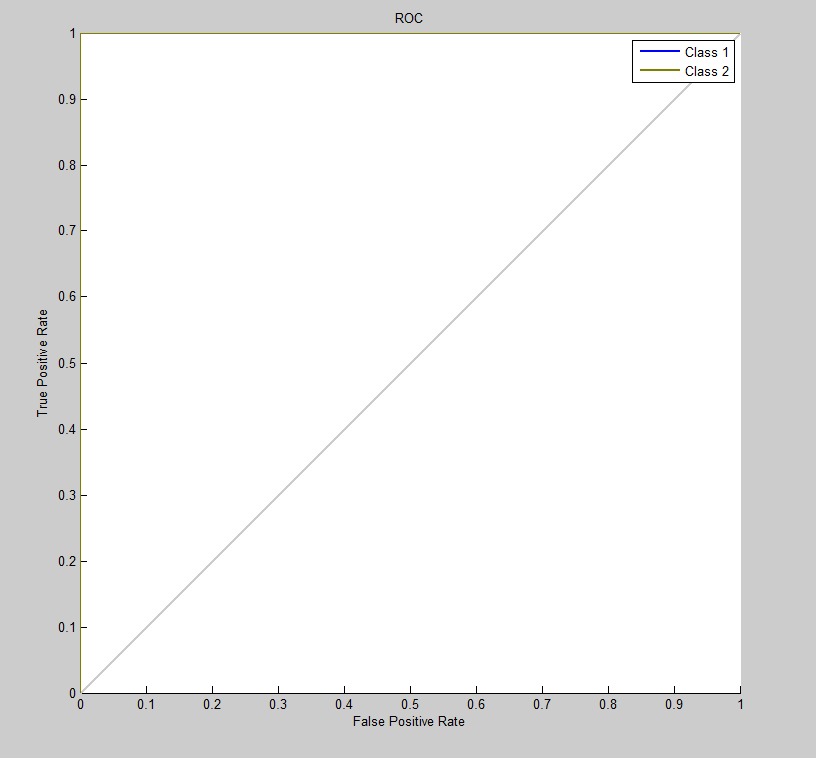
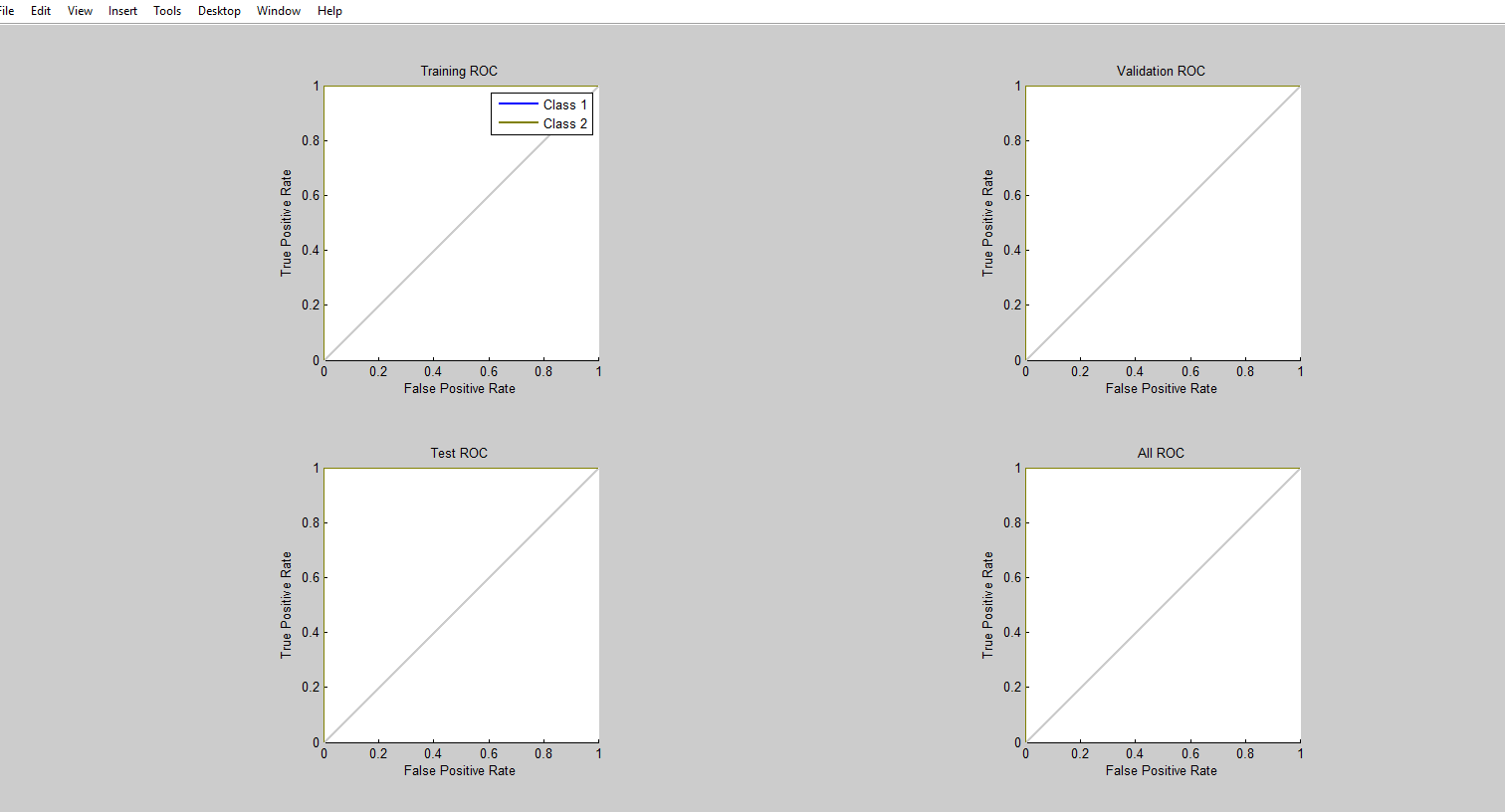
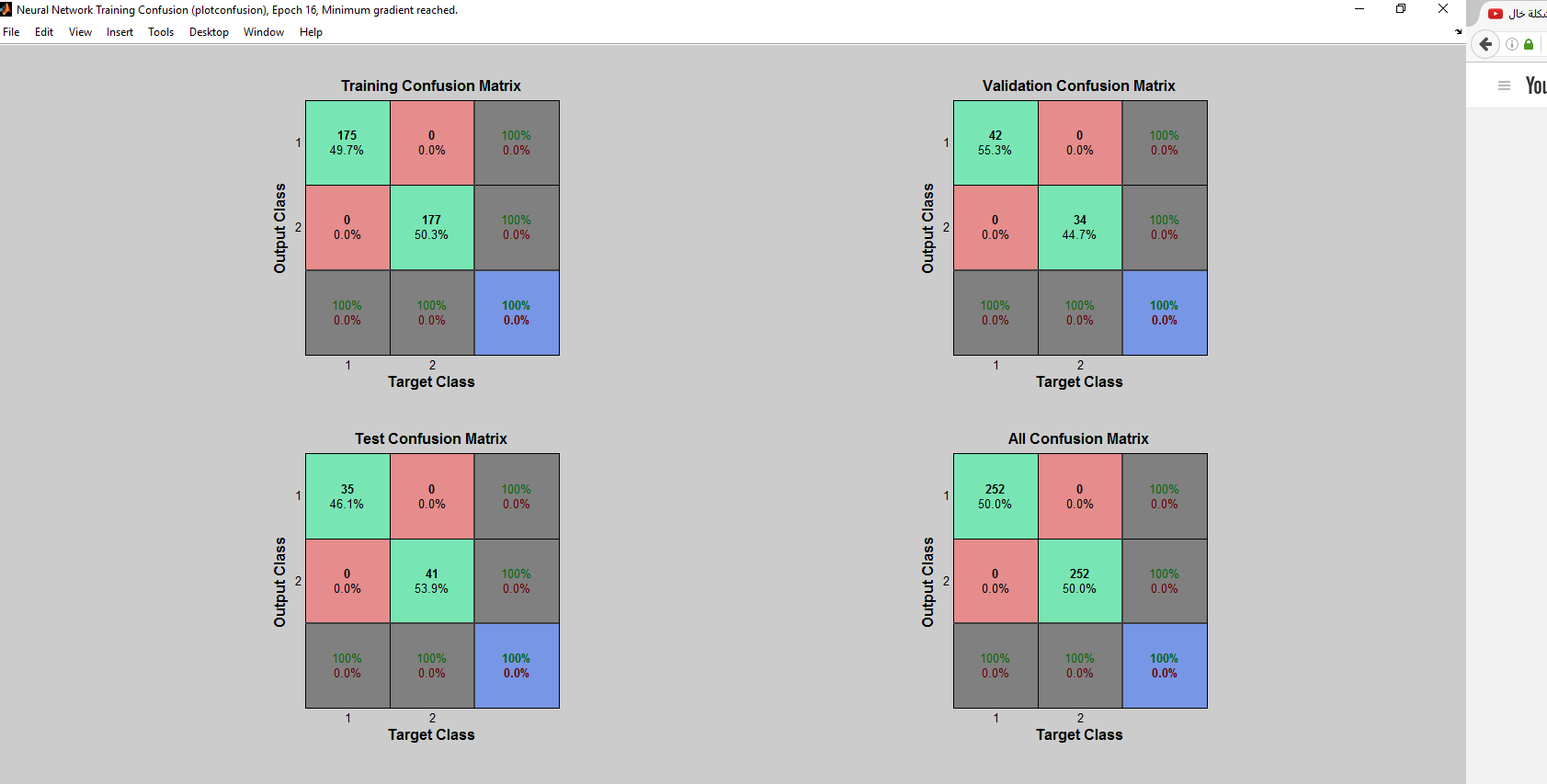
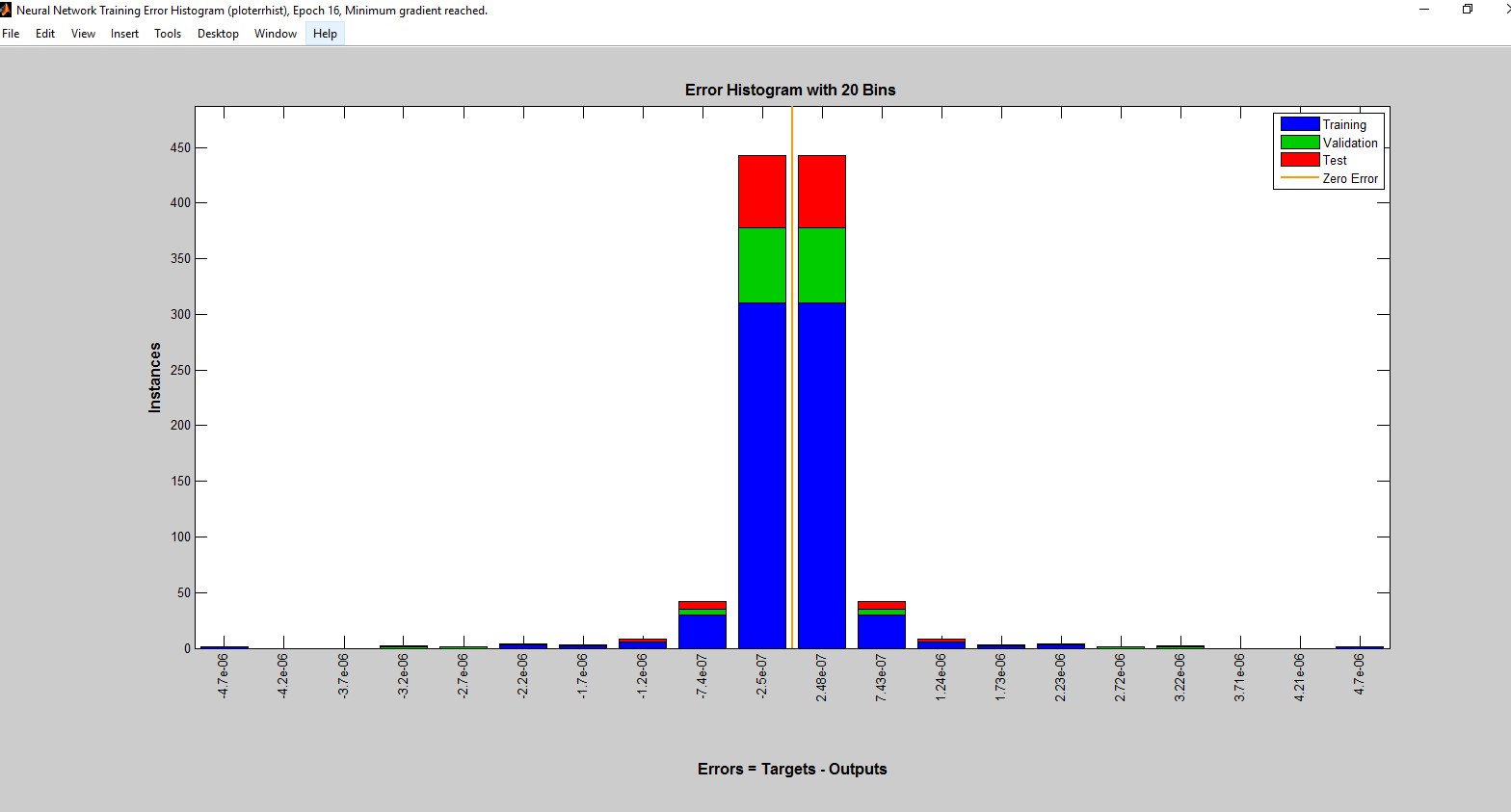
The ALO was able to achieve similar classification accuracy to that of the WVT baseline feature vector with smaller number of features (|Sj| < 35). On the other hand, the 40 features selected using GA was not enough to match the performance of WVT.

When ACO and GA are used to select 64 features, they both achieved similar or slightly better performance than that of the MFB baseline feature vector.

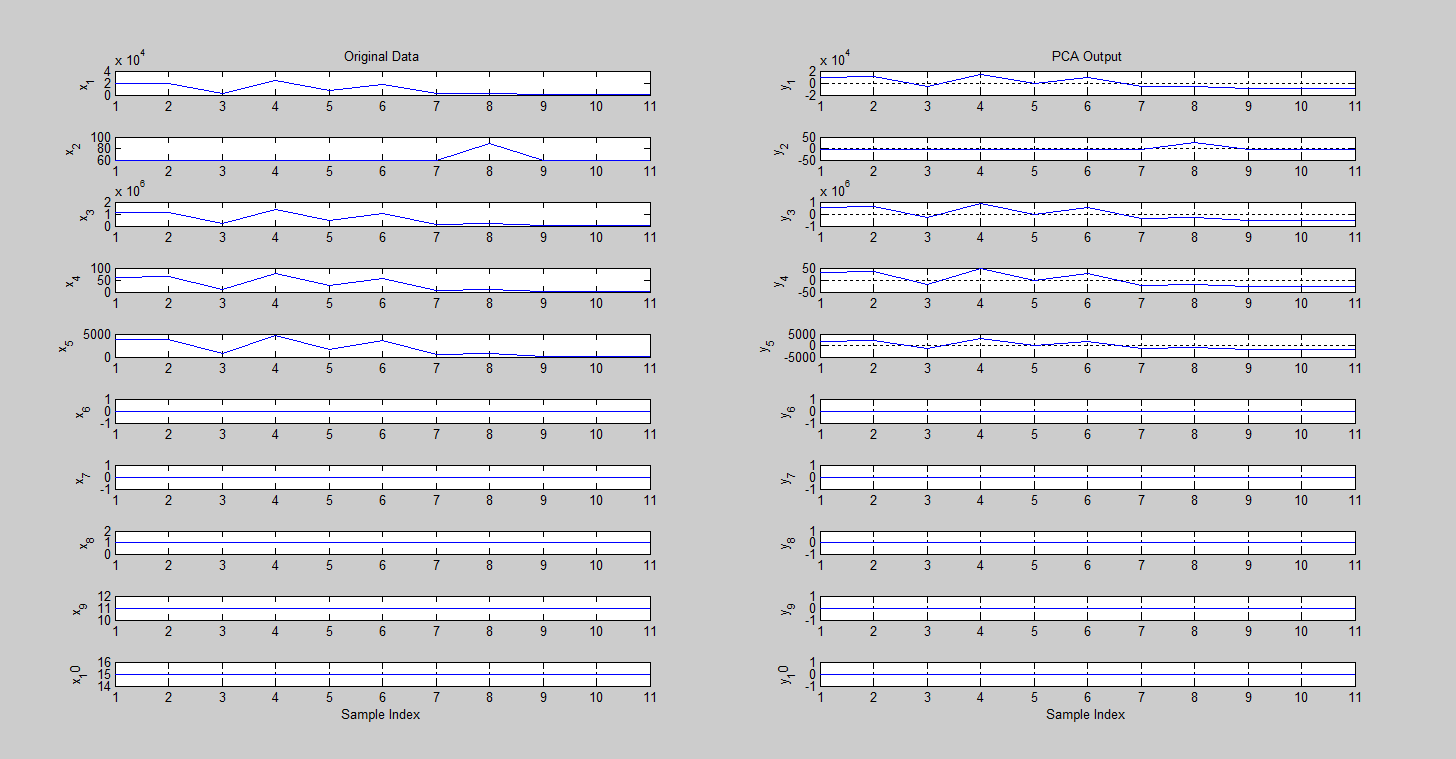
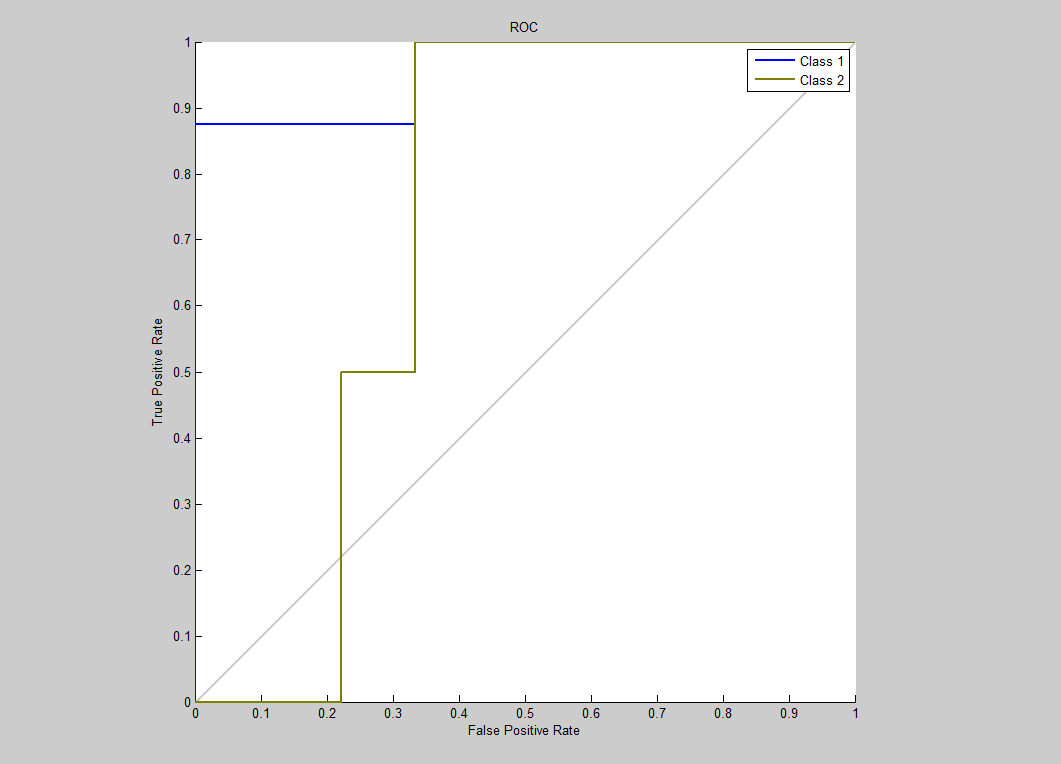
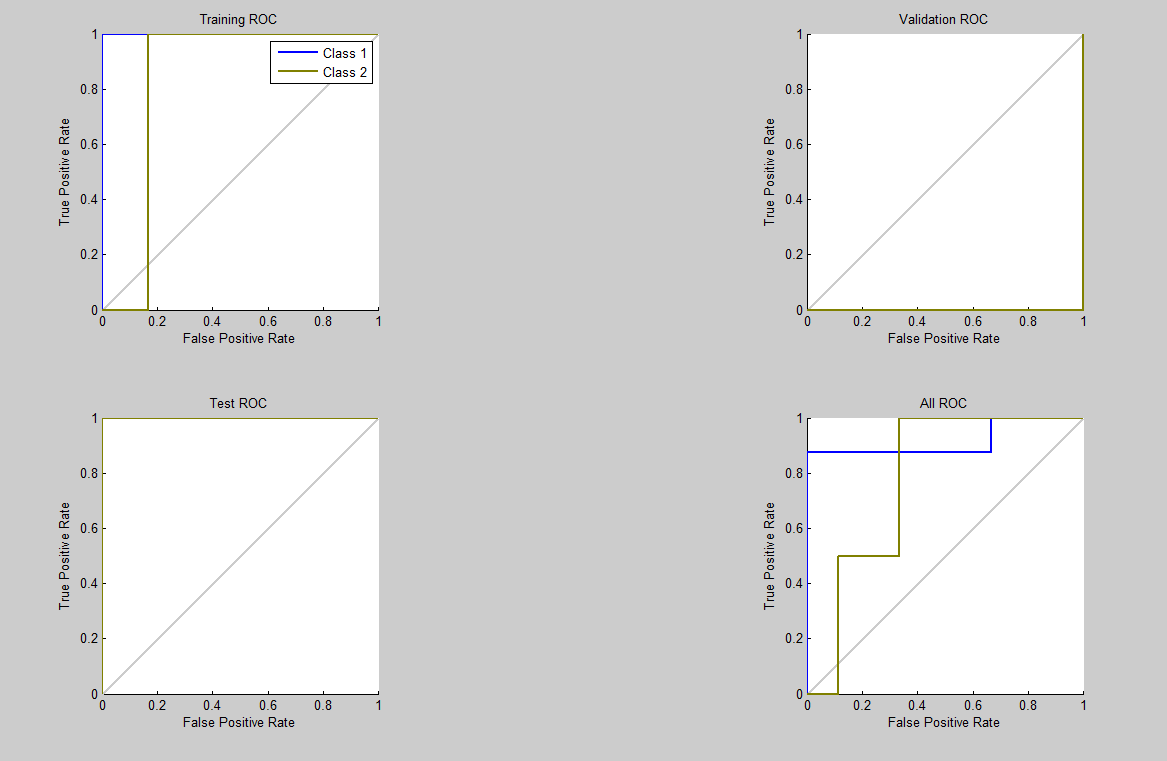
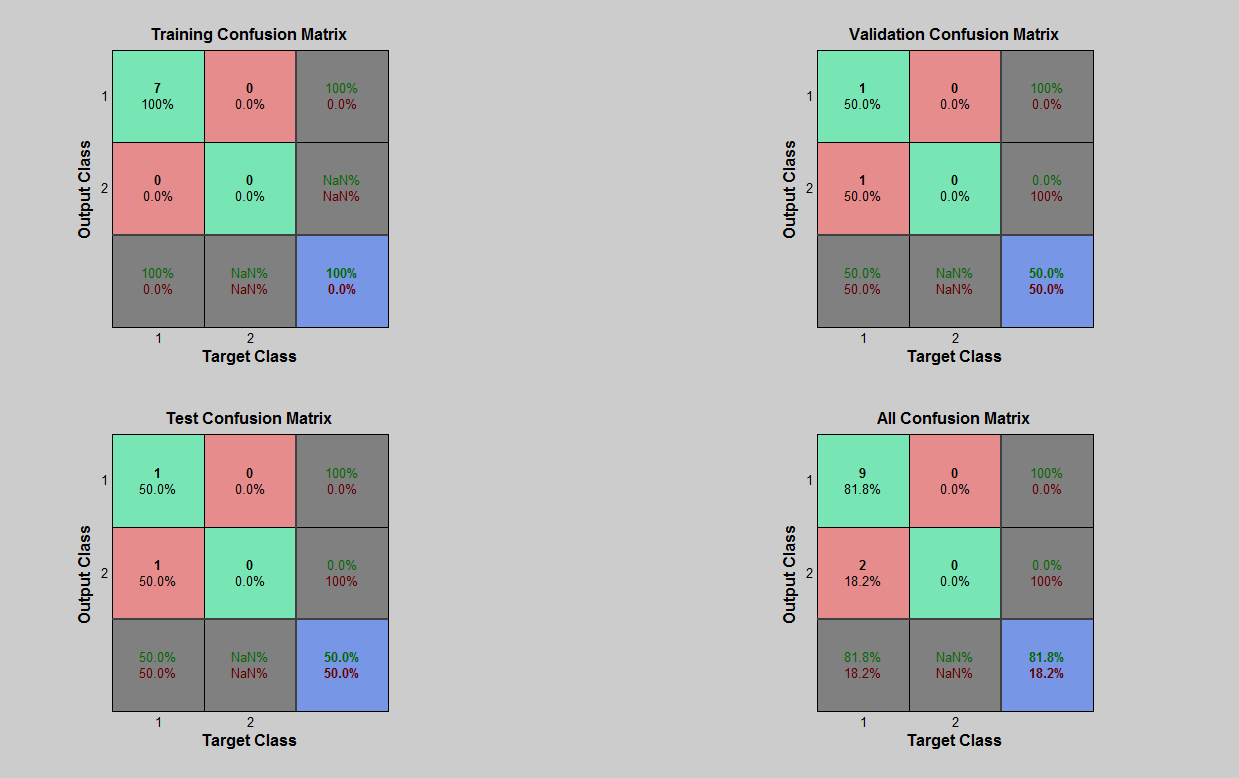
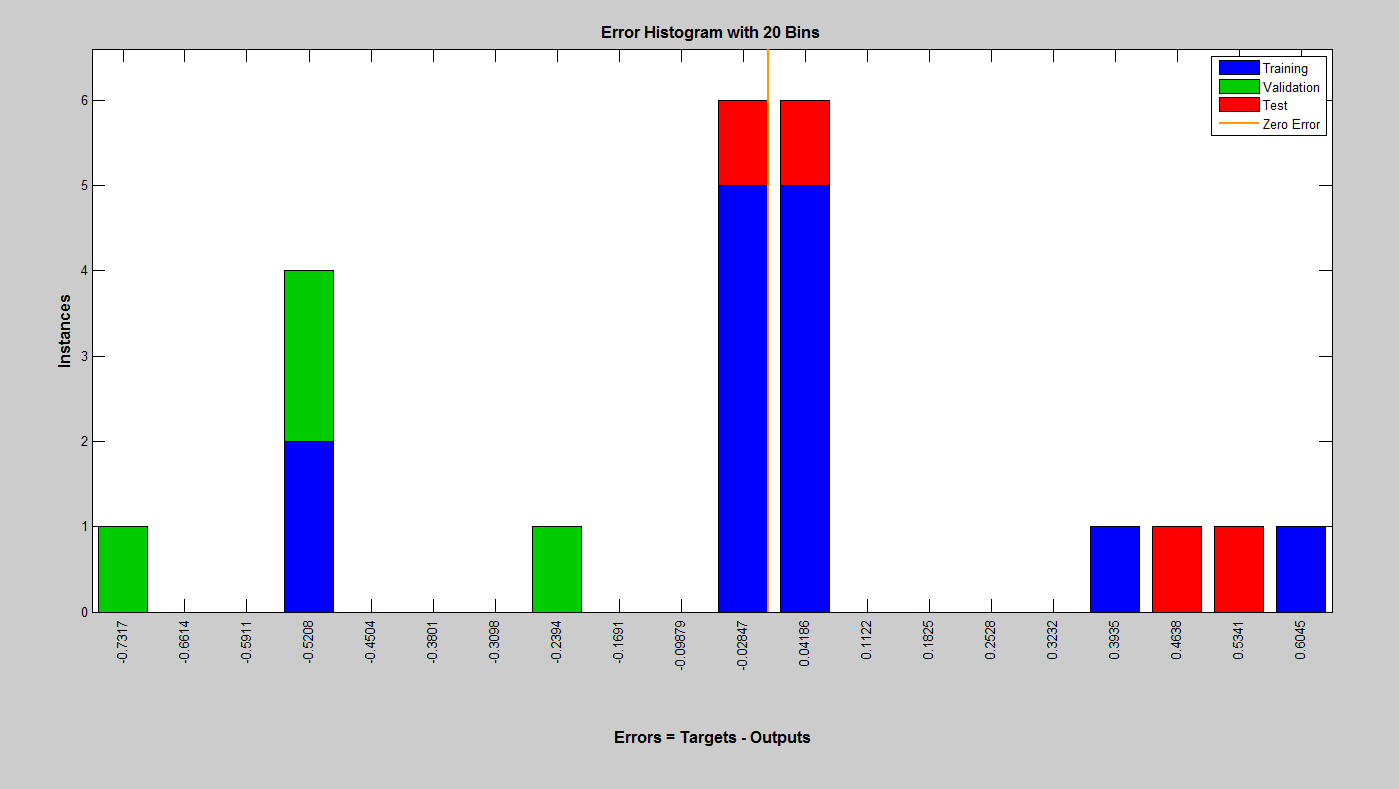
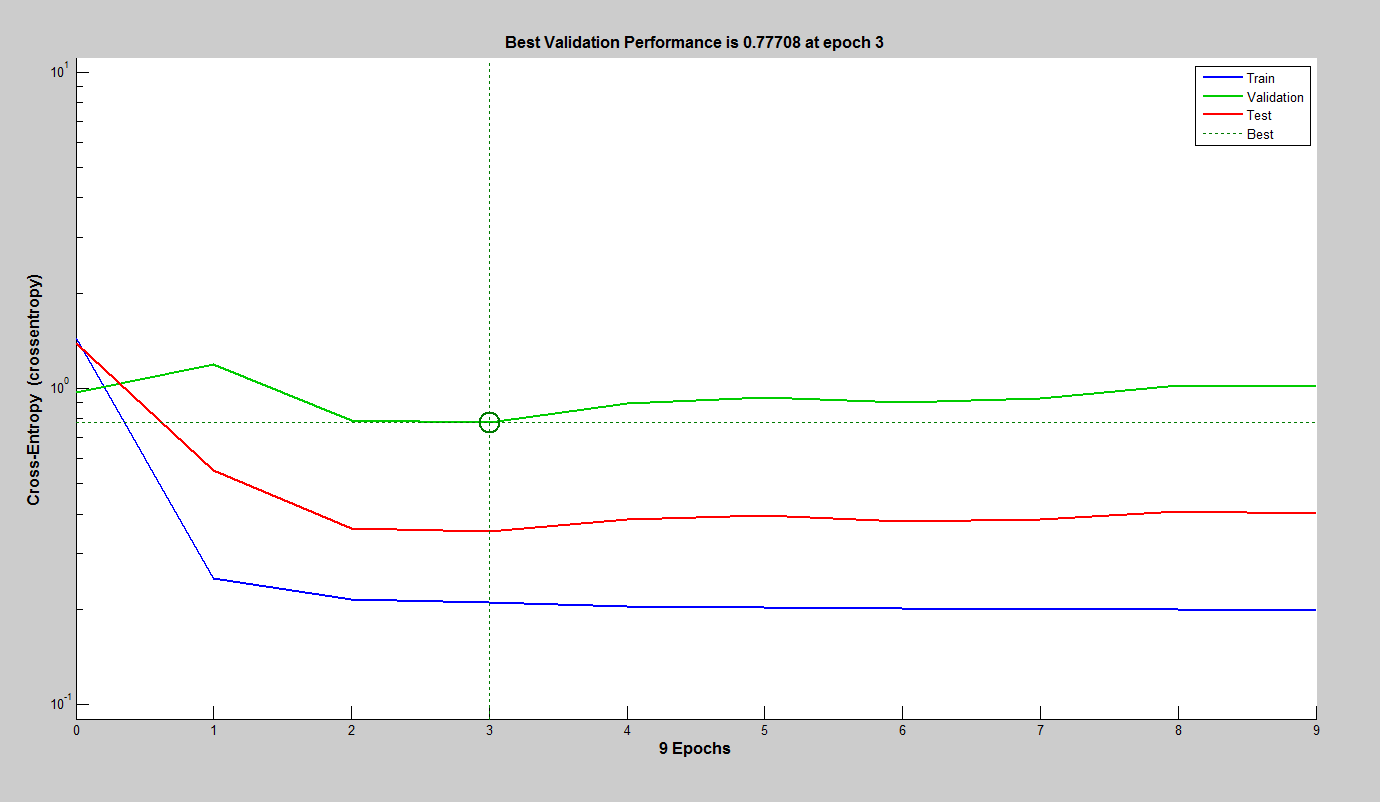
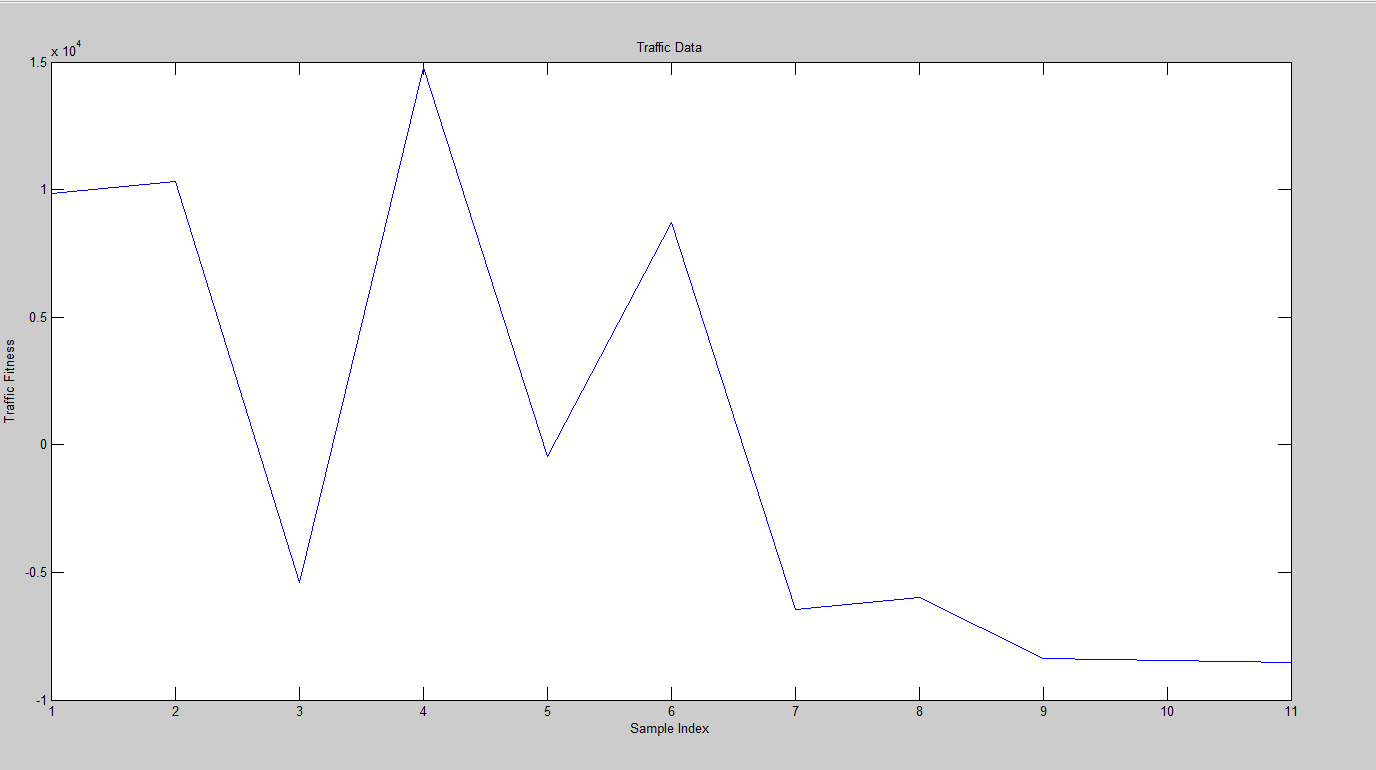
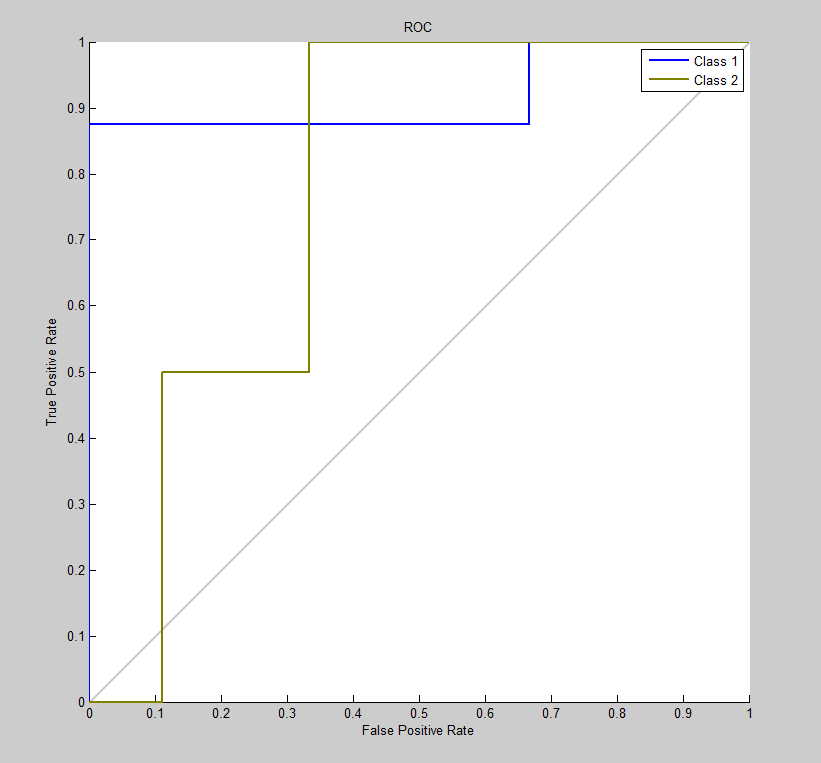
The overall performance of ACO is better than that of GA, where the average classification accuracy of ACO and GA over all the cases are: 84.23% and 83.47% respectively.

****

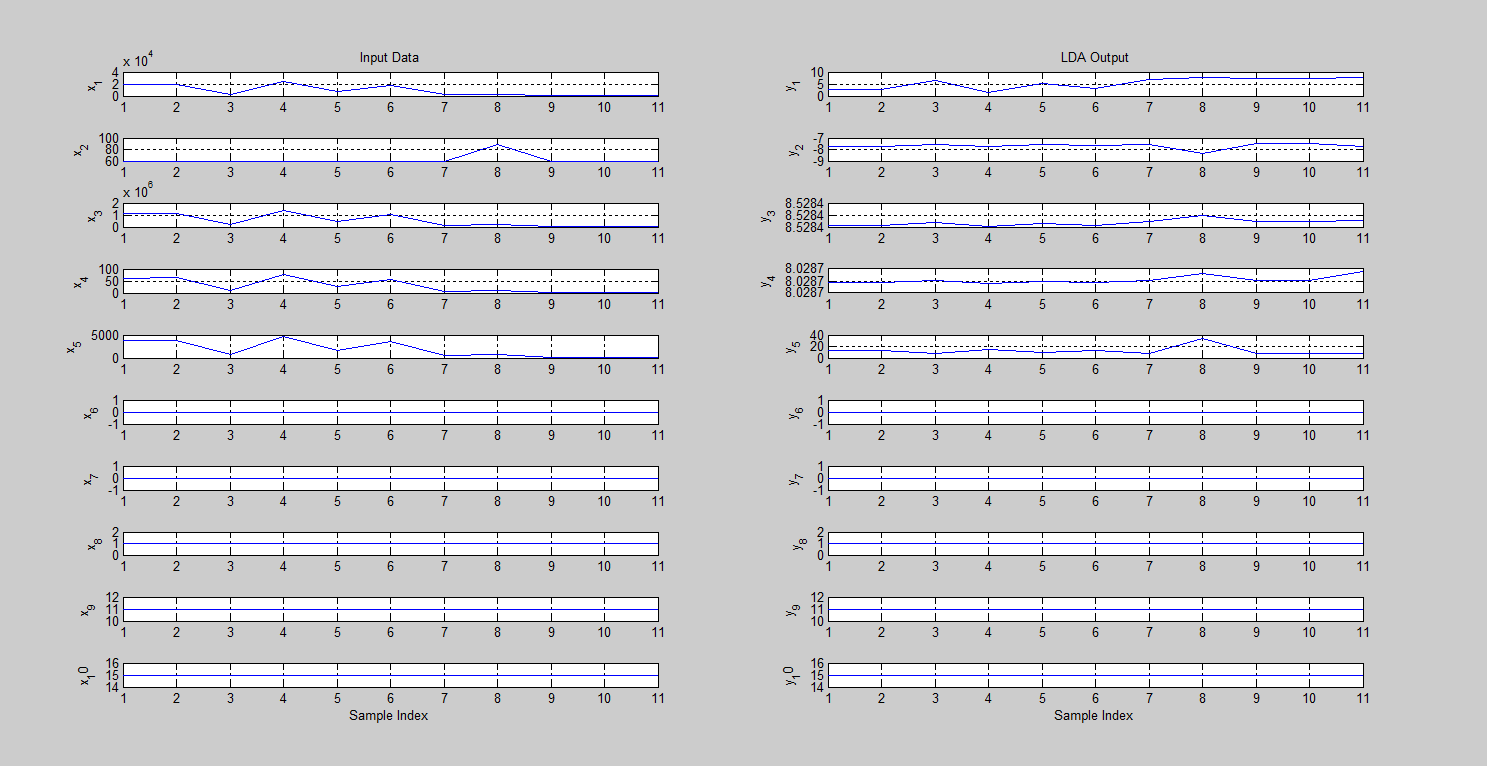
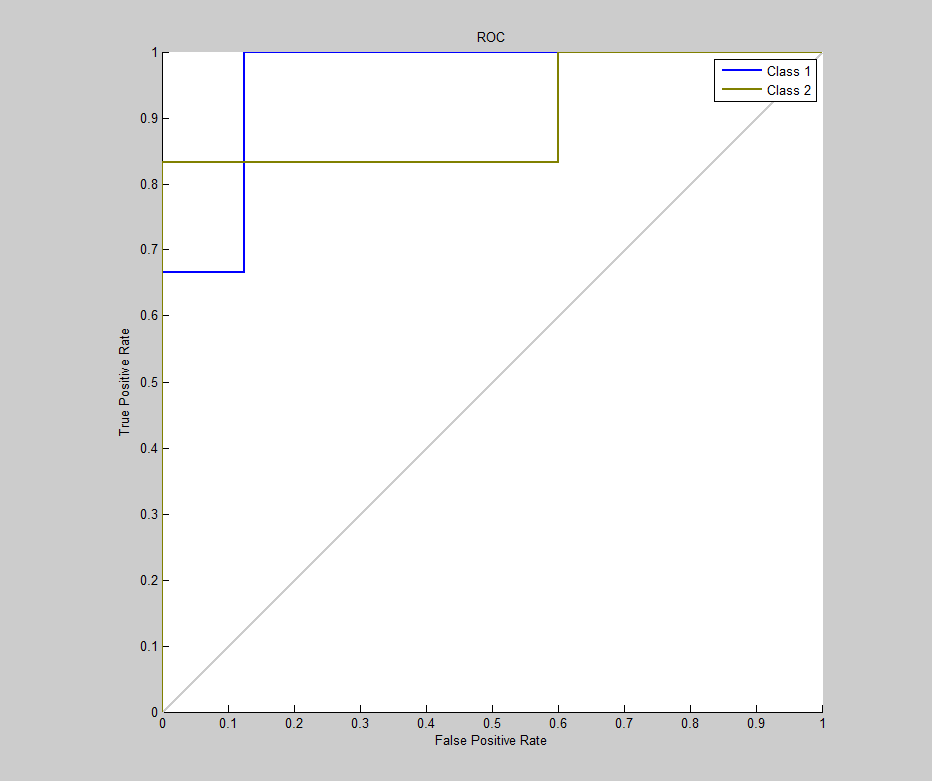
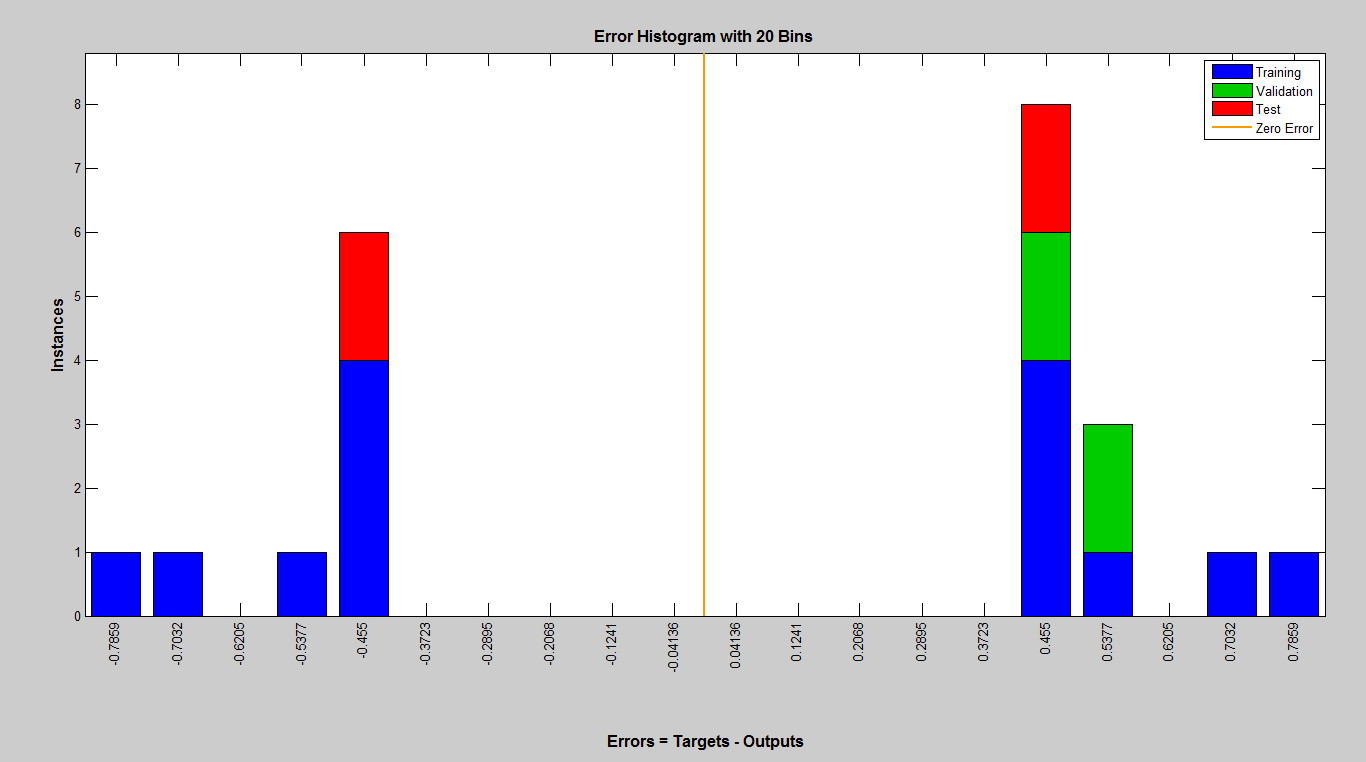
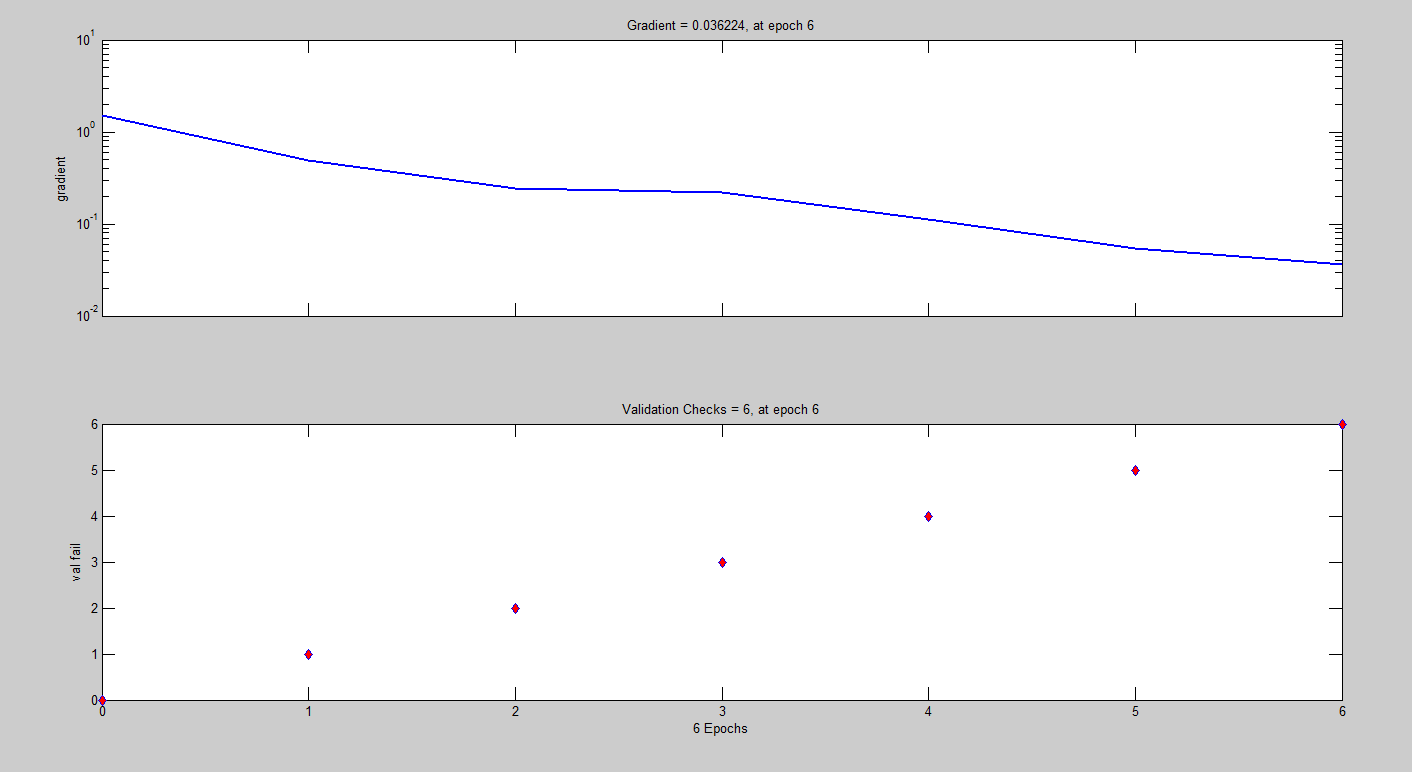
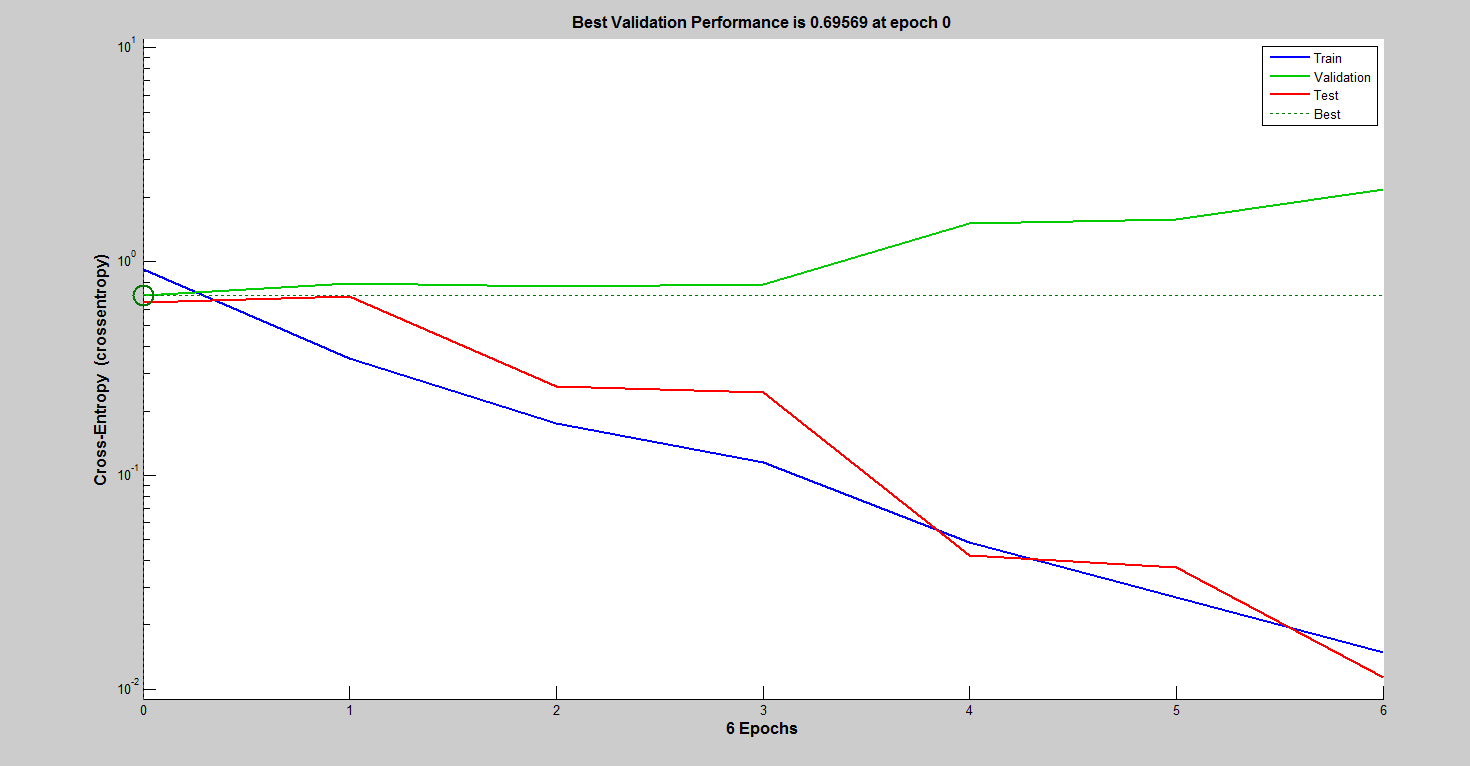
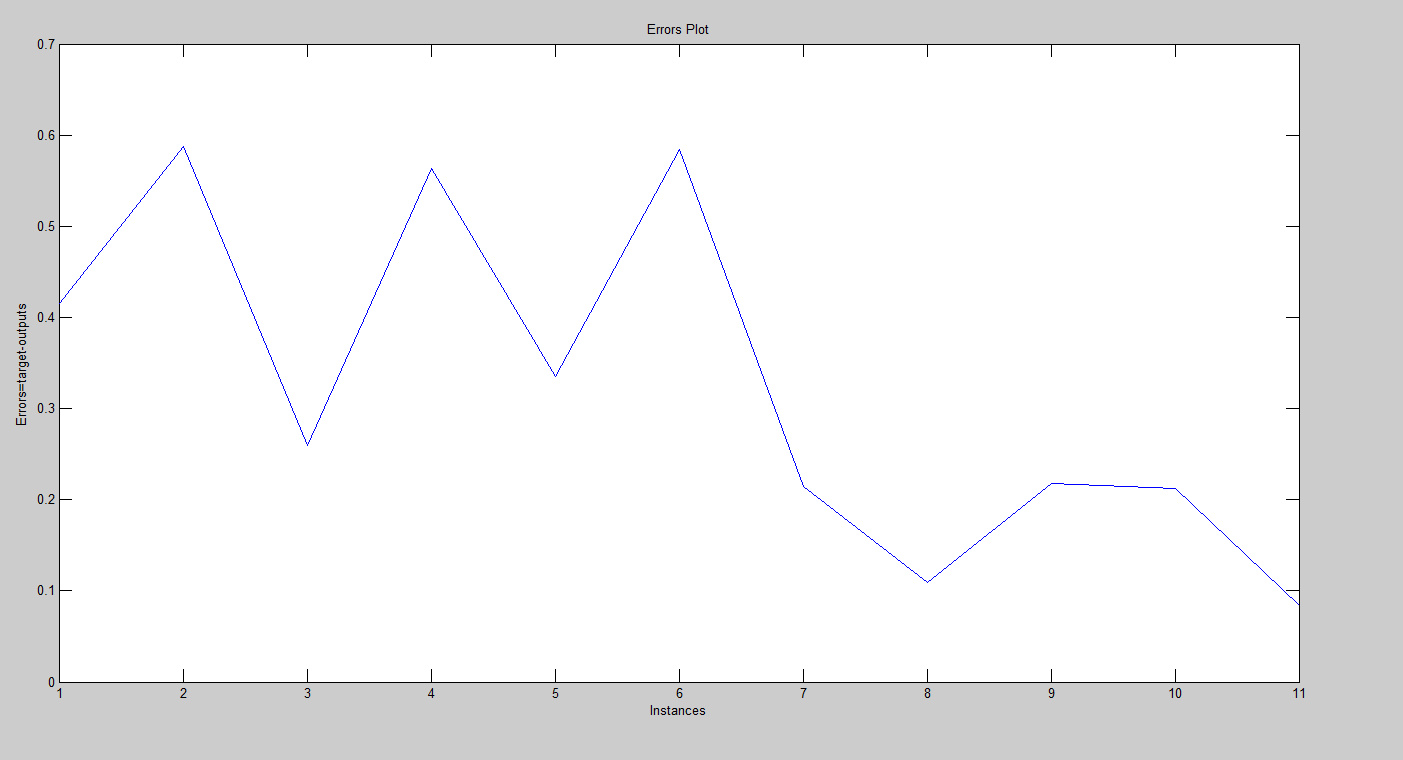
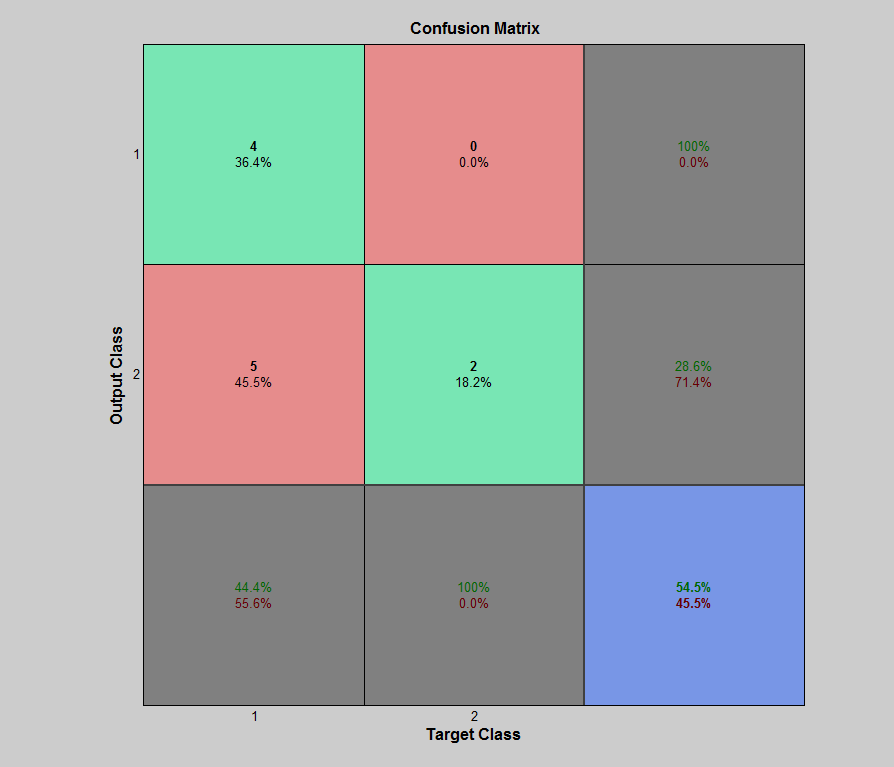
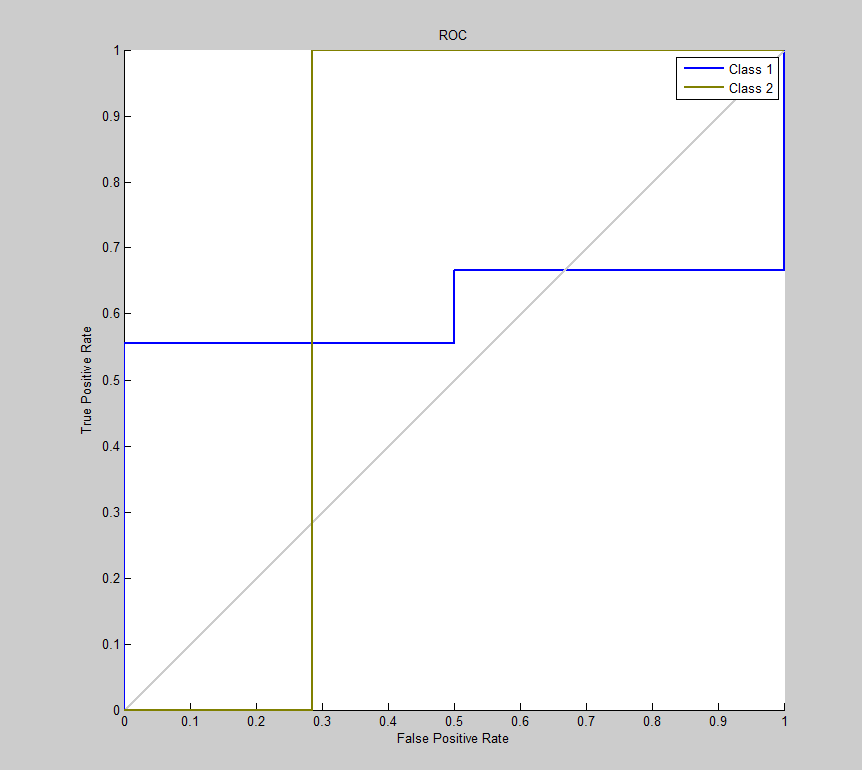
The above figure xxx shows the configured feed forward neural network training state plot. It also shows validation check at epoch 14 and highlights that there is no validation failure up to this epoch.



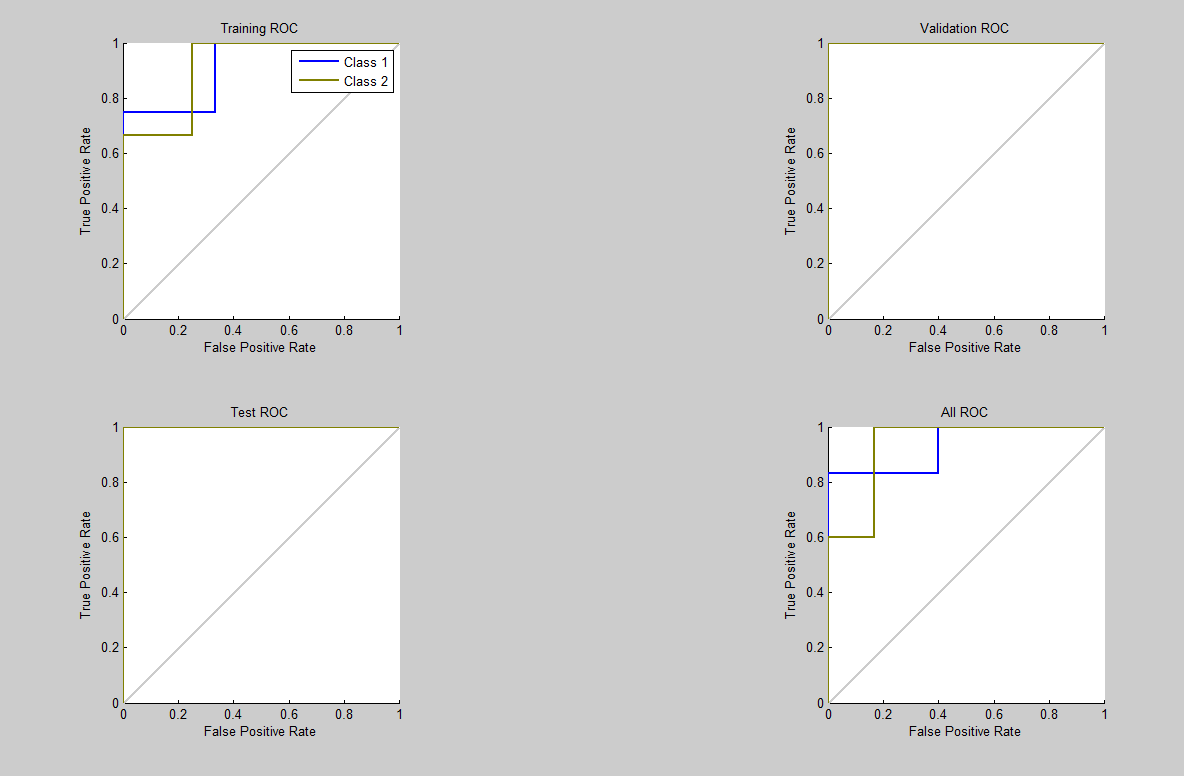
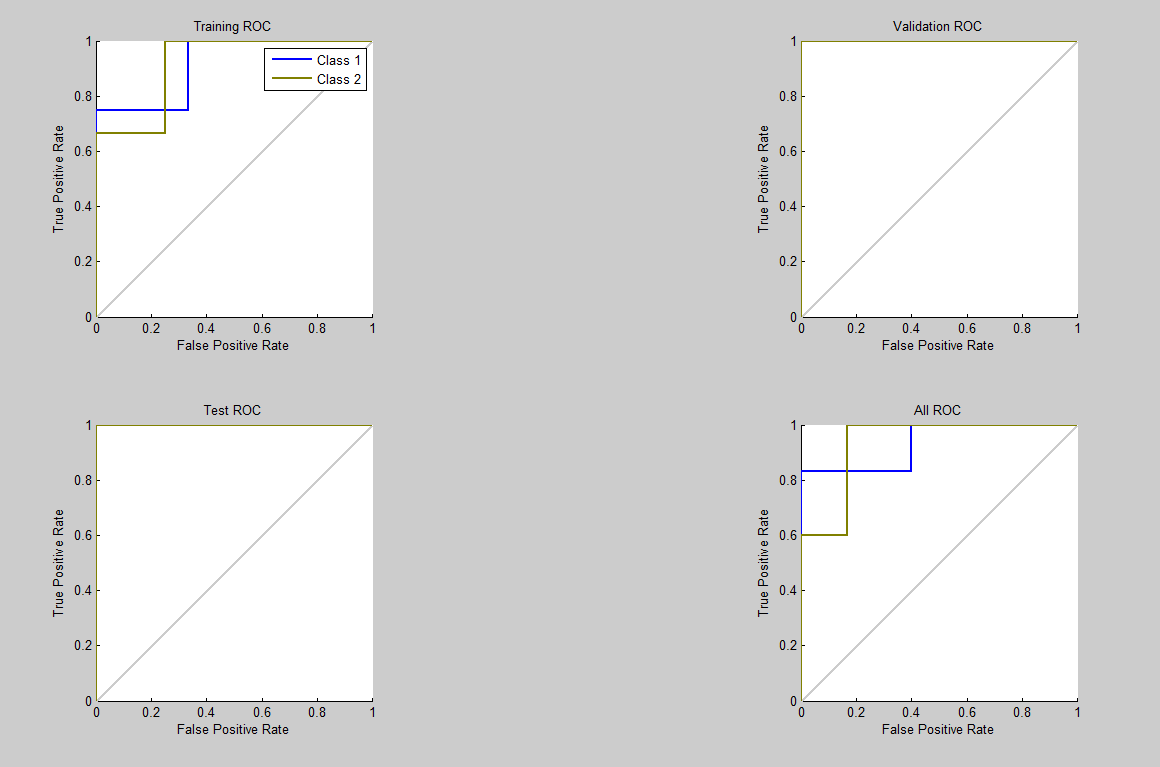
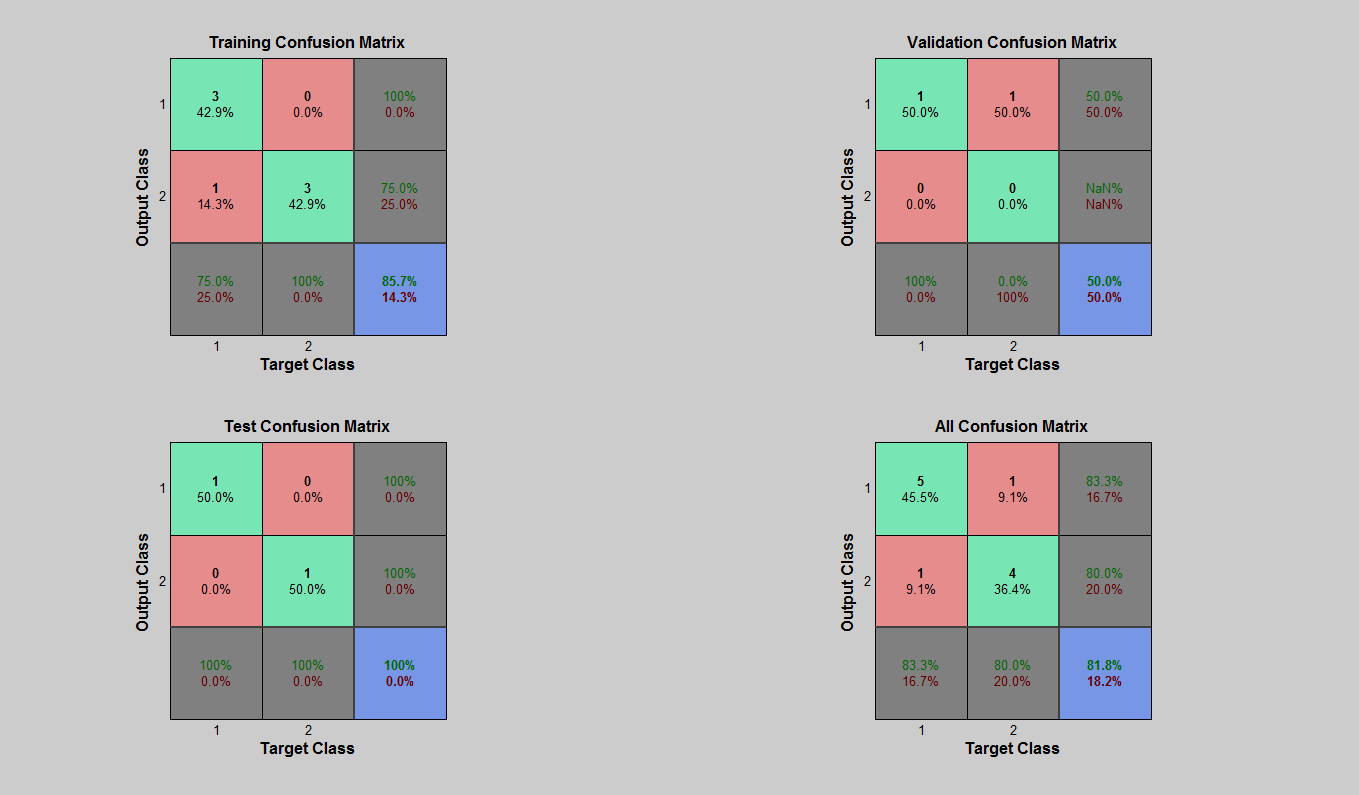
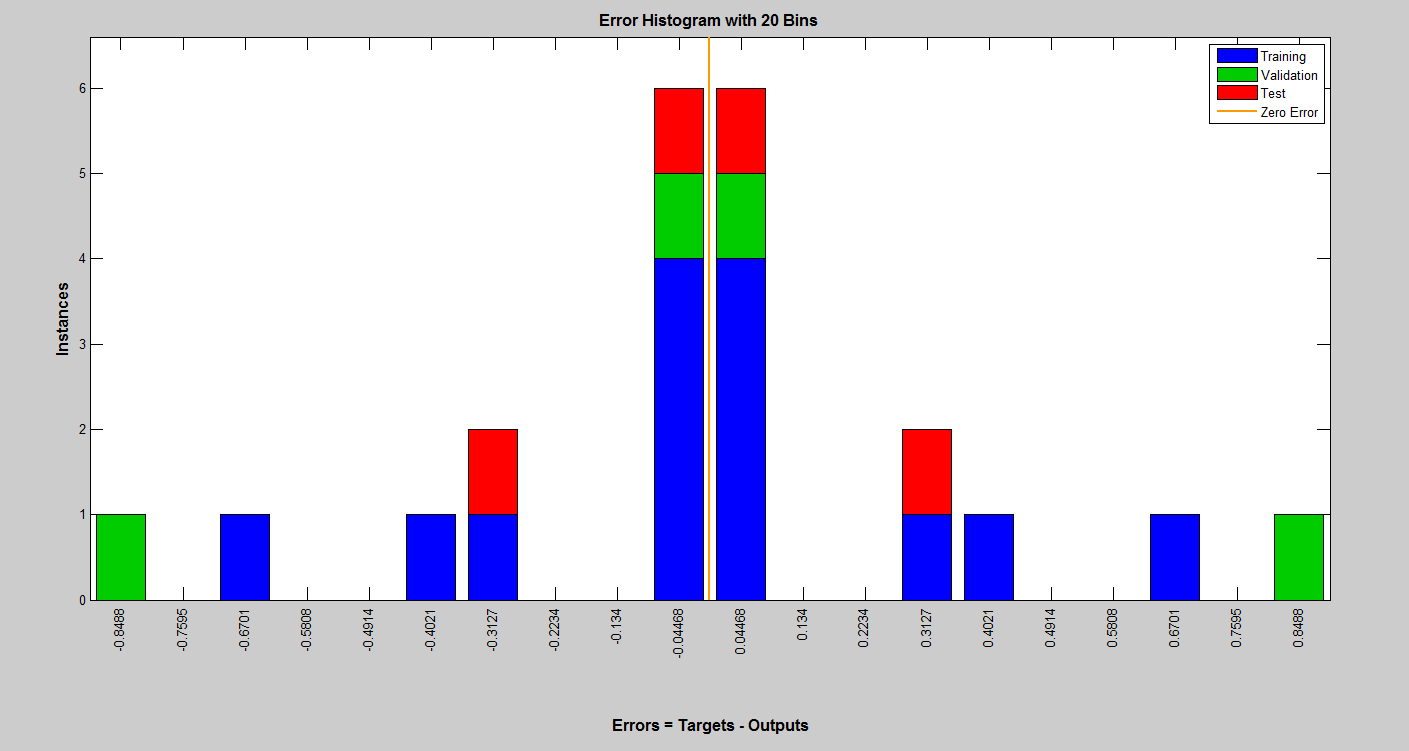
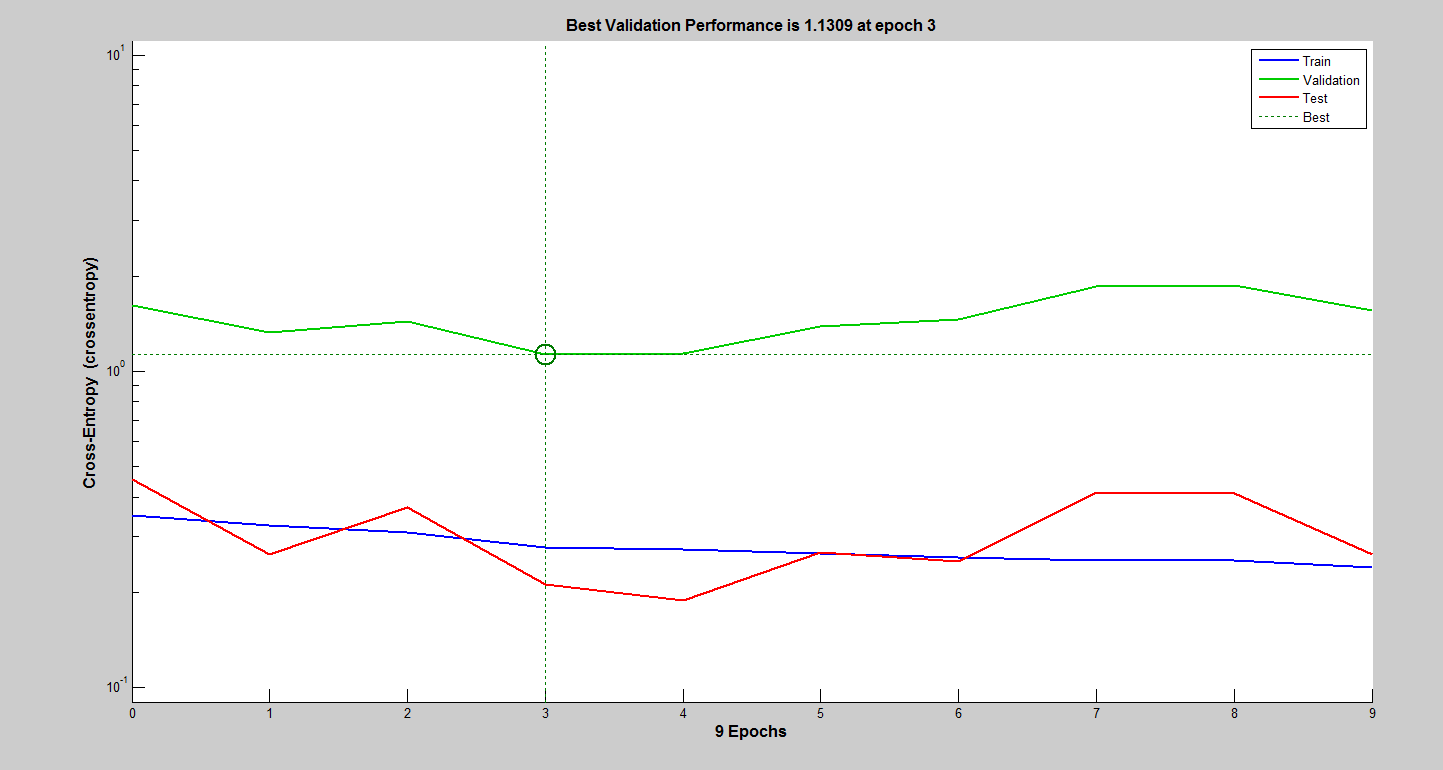
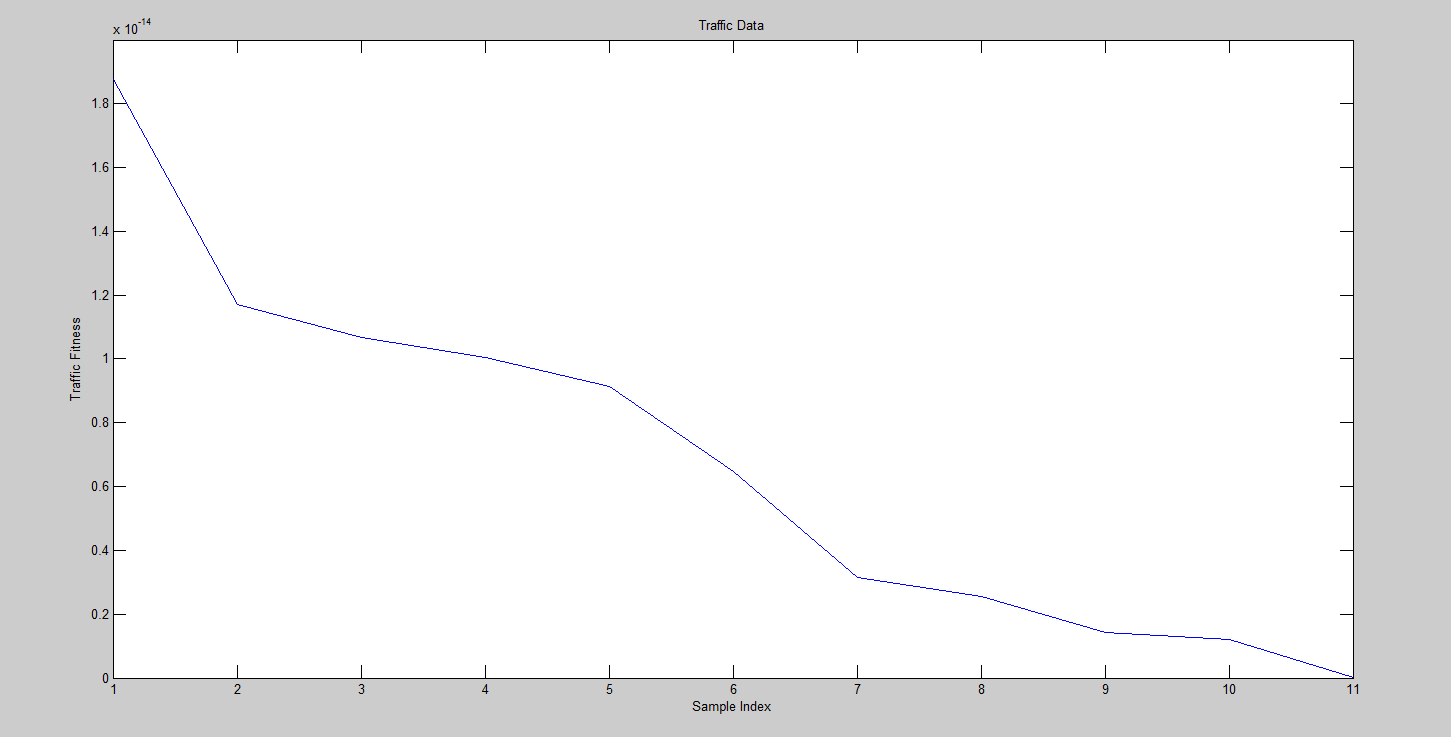
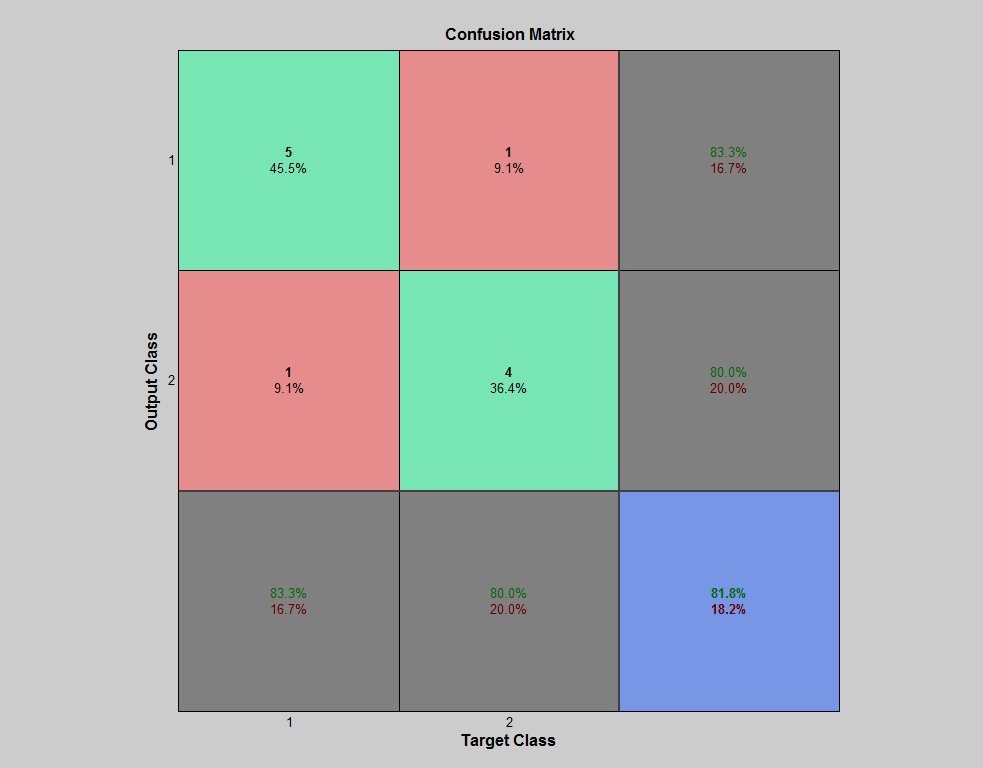
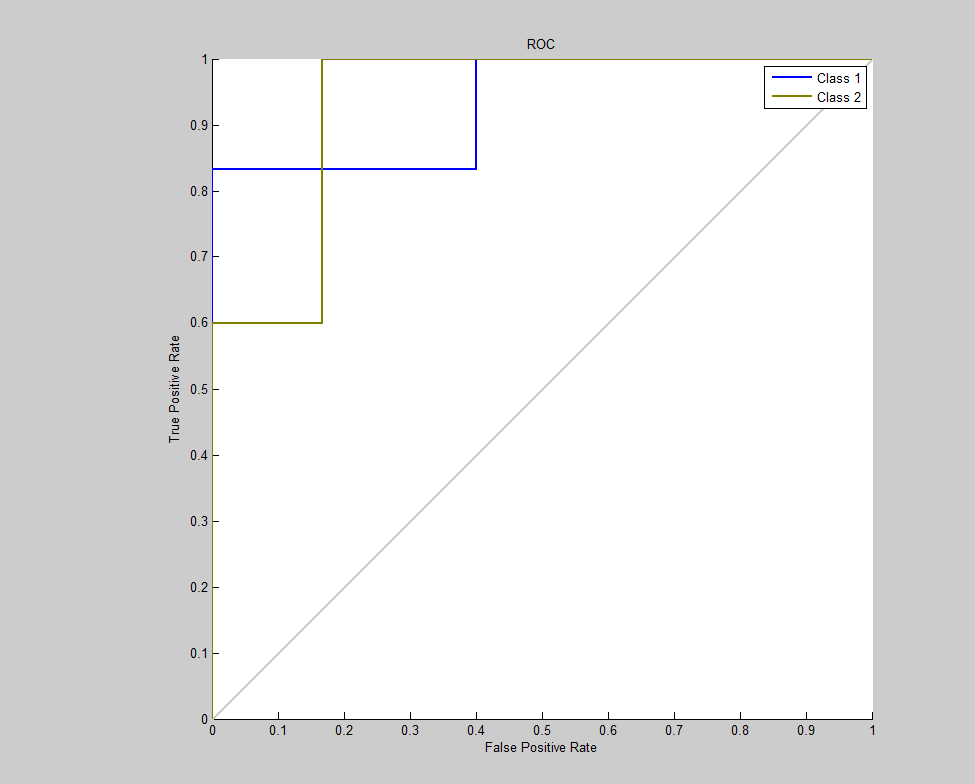
* 1. **ICMP Attack on cloud computing**
     1. **PCA**



* + 1. **LDA**



* + 1. **Proposed (PCA+LDA+ALO)**

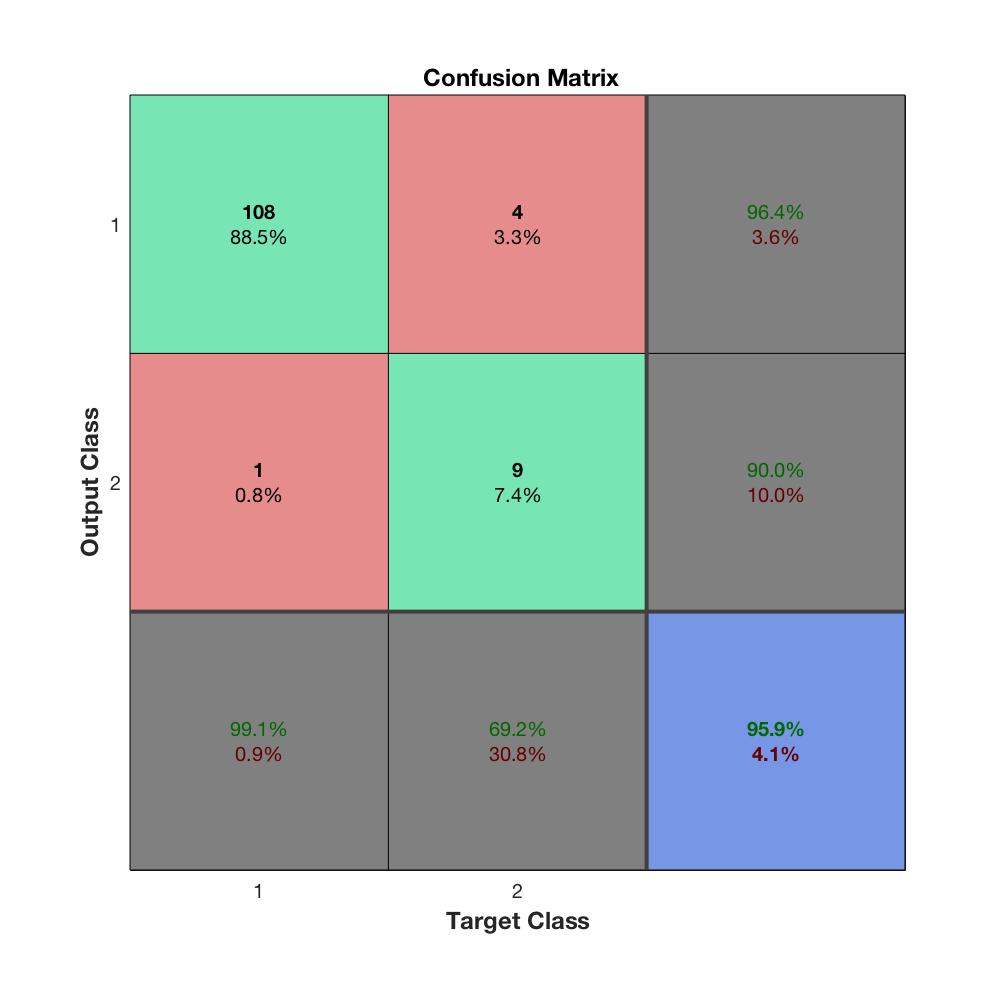


* 1. **TCP Sync Attack cloud computing**
     1. **PCA**

****

The above figure shows the confusion matrices for training, testing, and validation, and the three kinds of data combined. As mentioned before, there are 86 network records (70%) used for neural network training purpose, 18 network records (15%) used for validation and test respectively. For each confusion matrix, the first two diagonal cells show the number and percentage of correct classifications by the trained network. Take Training Confusion Matrix as example, 76 network traffic packages are correctly classified as normal. This corresponds to 50% of all 86 training samples. Similarly, 6 network traffic packages are correctly classified as attack. This corresponds to 6.9% of the total training samples. Overall, 95.9% of the predictions on the validation samples are correct and 4.1% are wrong classifications.

The network outputs are very accurate, as you can see by the high numbers of correct responses in the green squares and the low numbers of incorrect responses in the red squares. The lower right blue squares illustrate the overall accuracies.

****

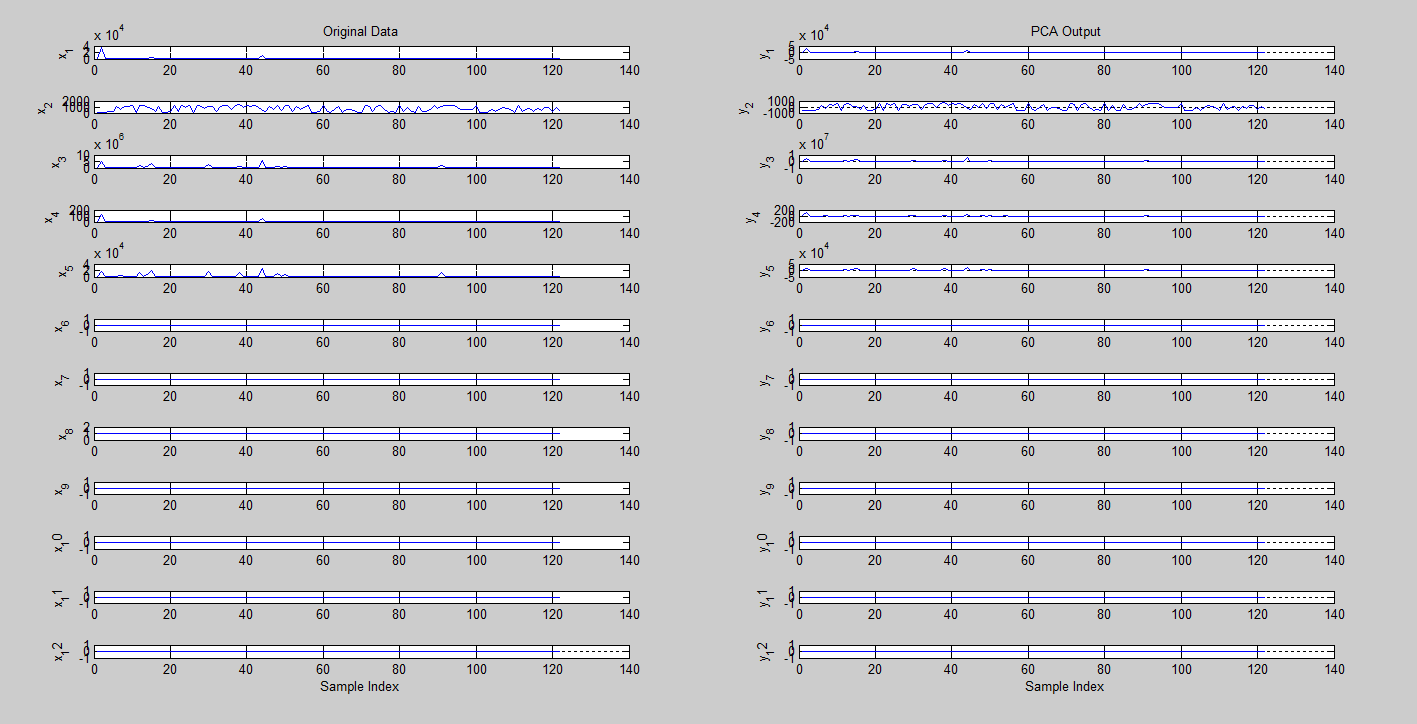
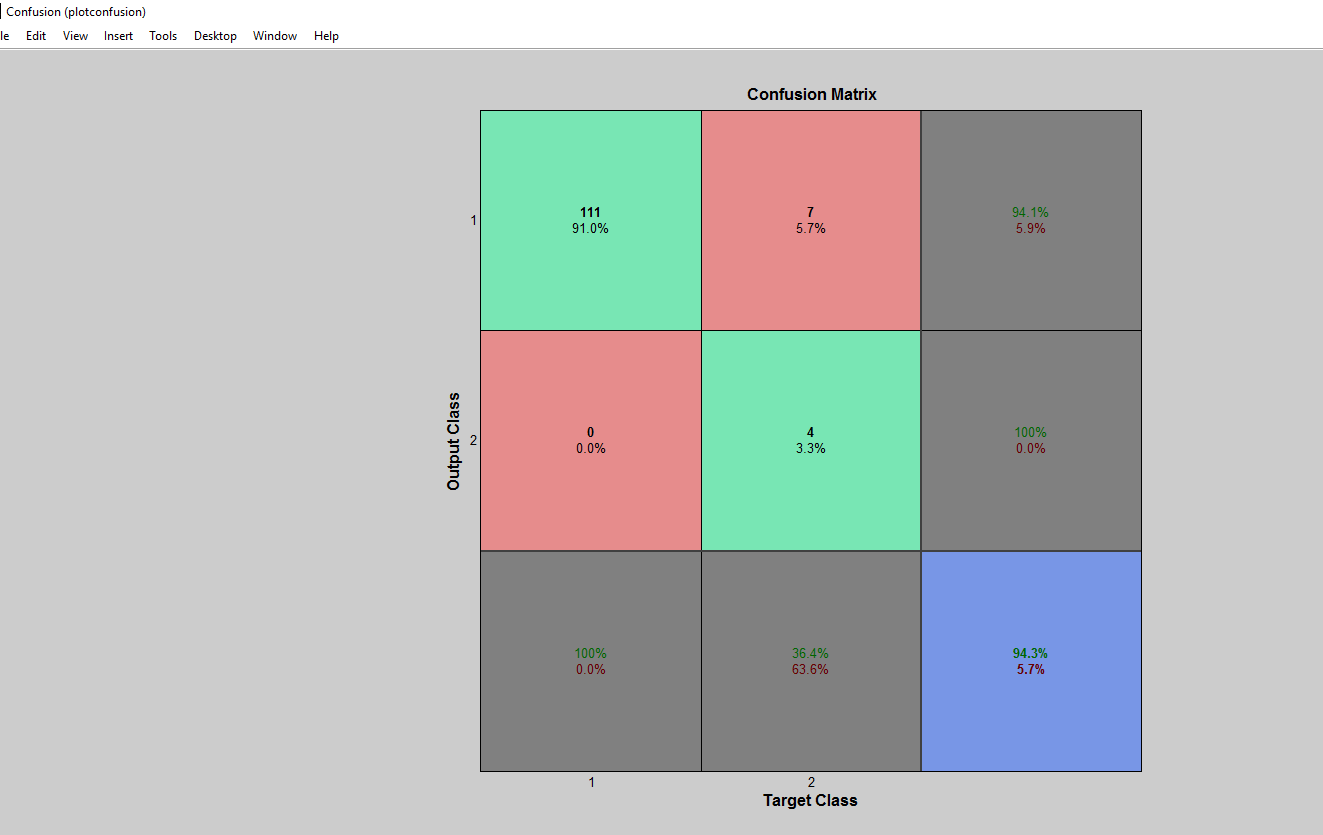
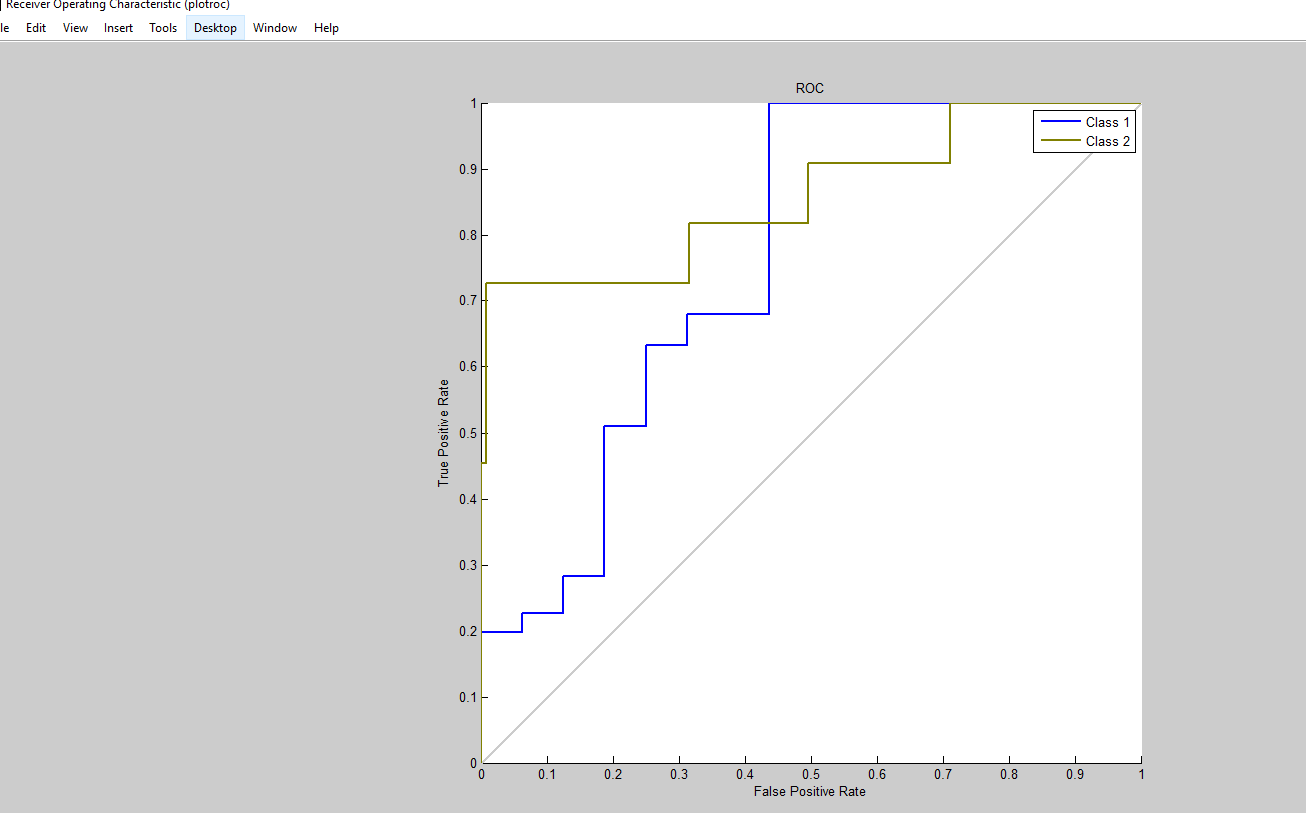
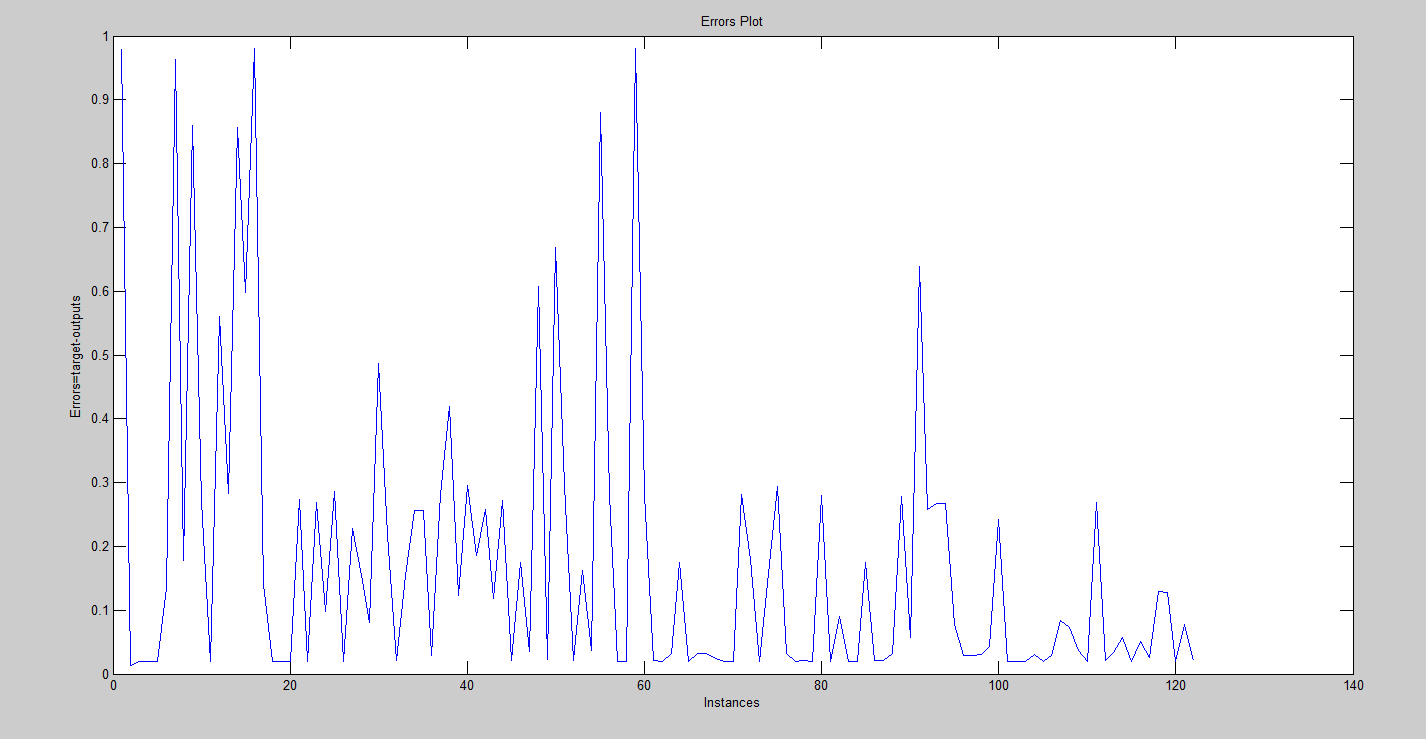
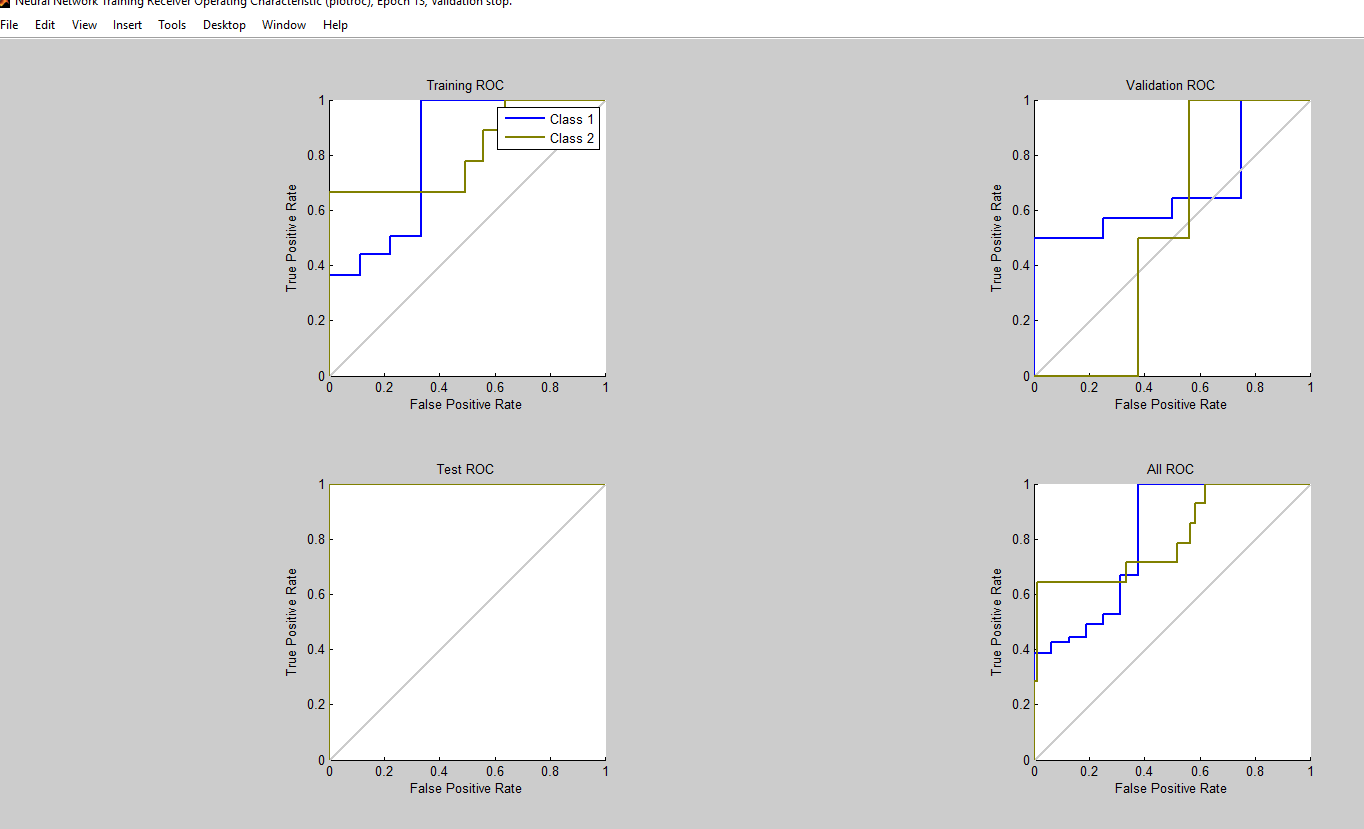
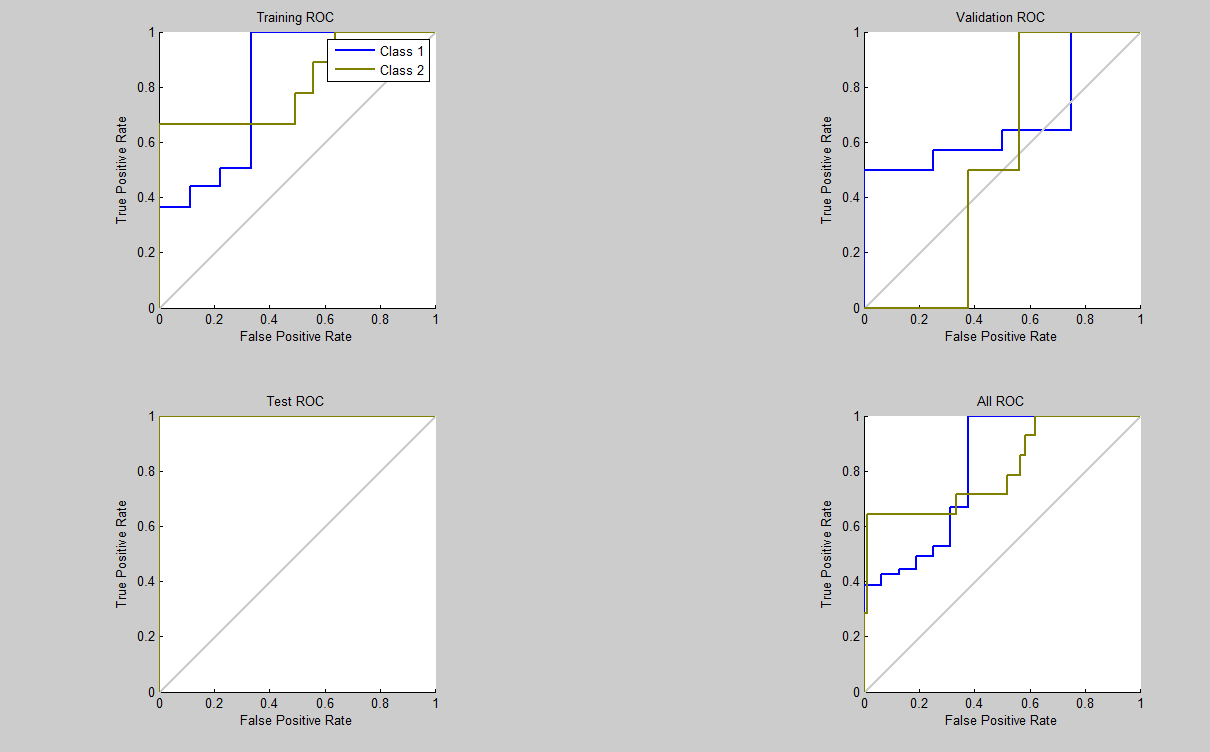
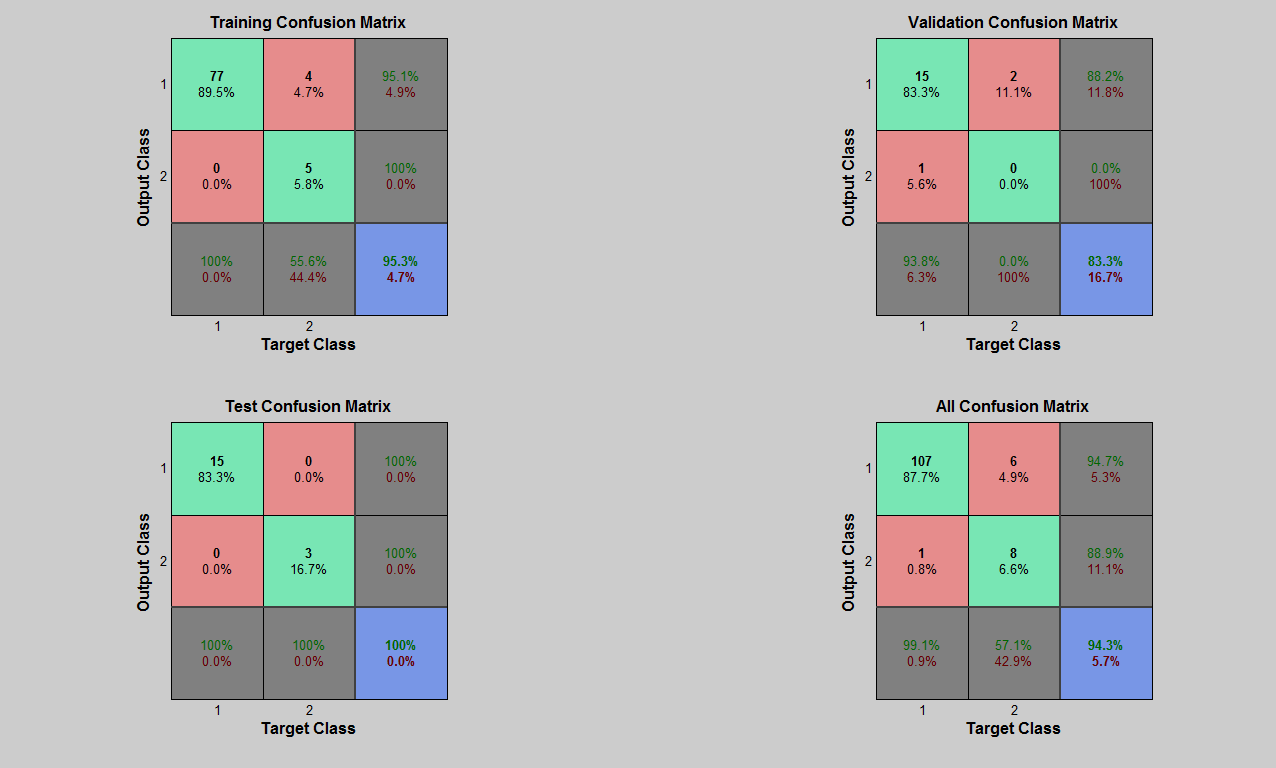
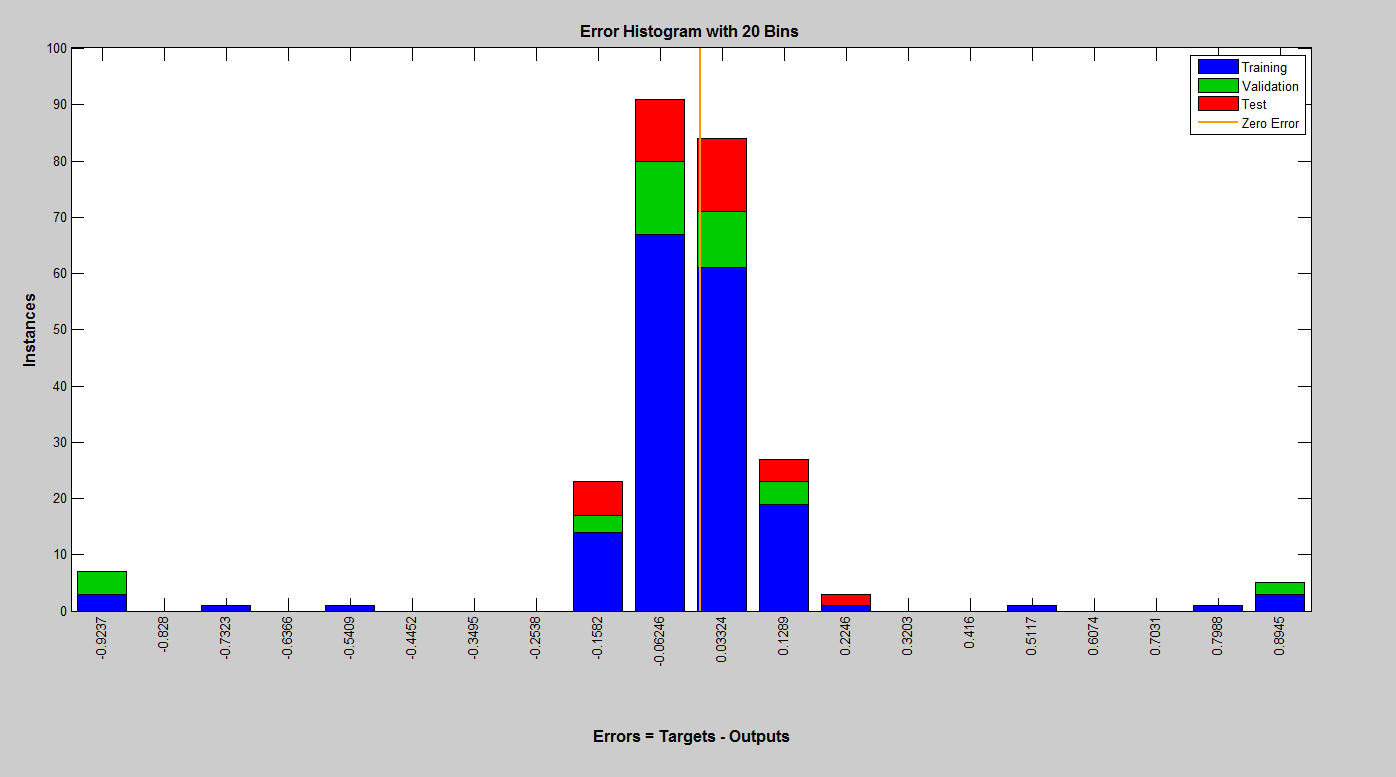
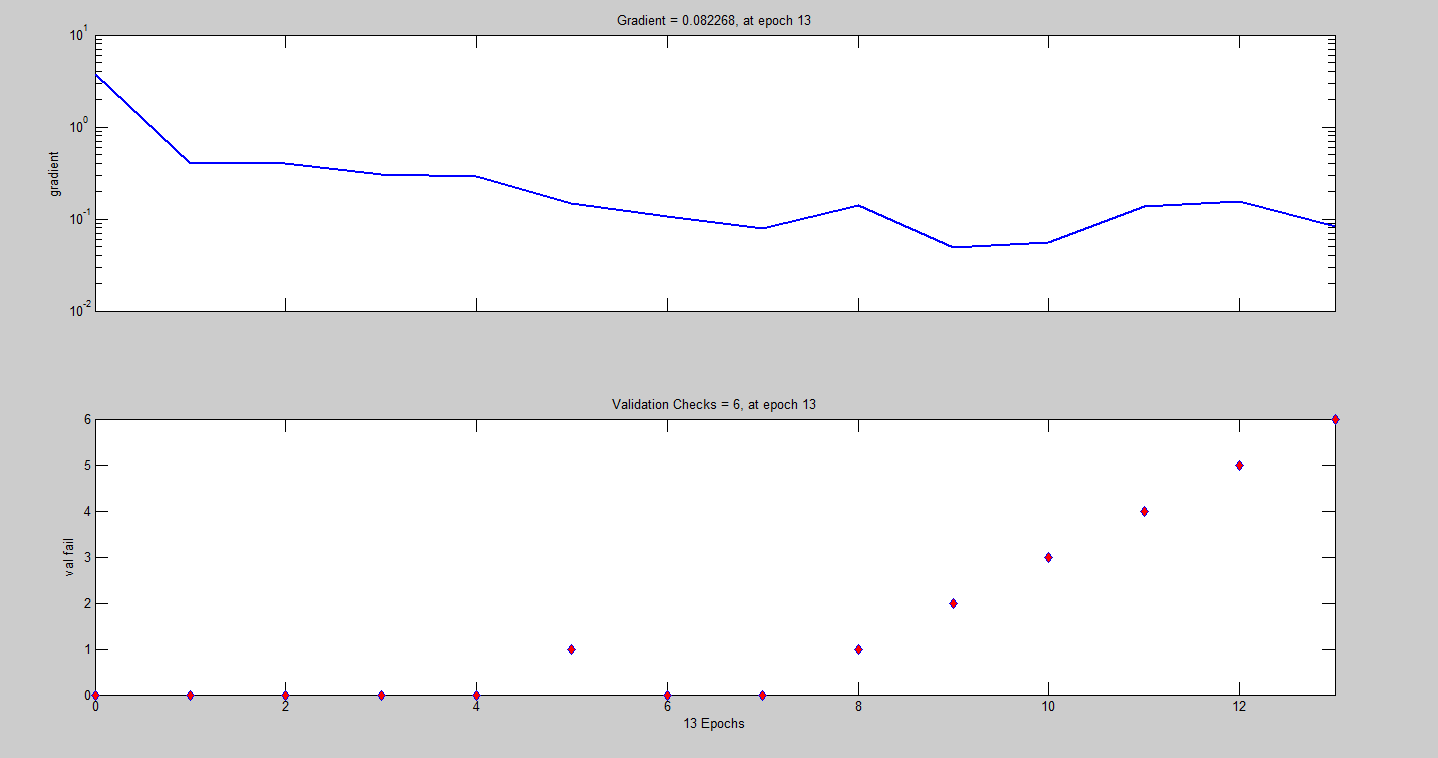
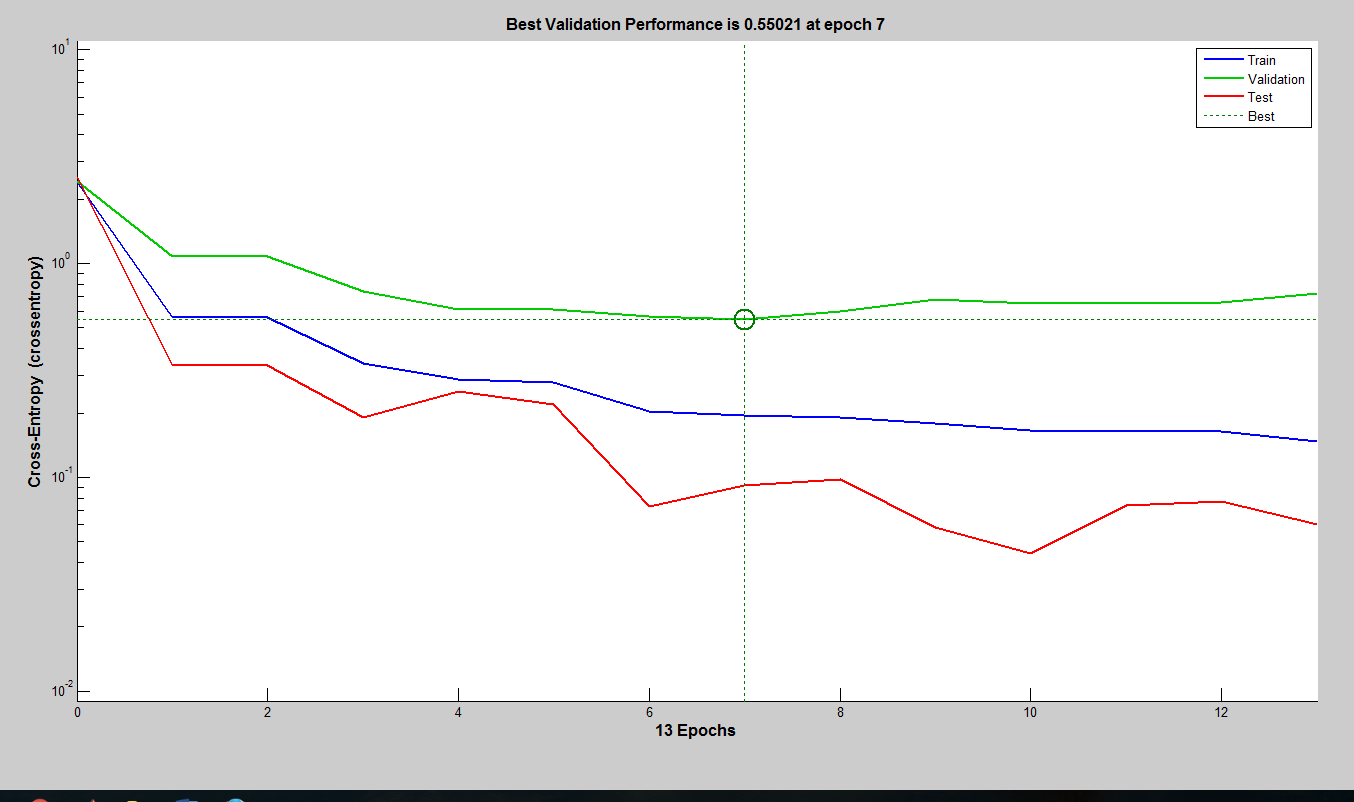
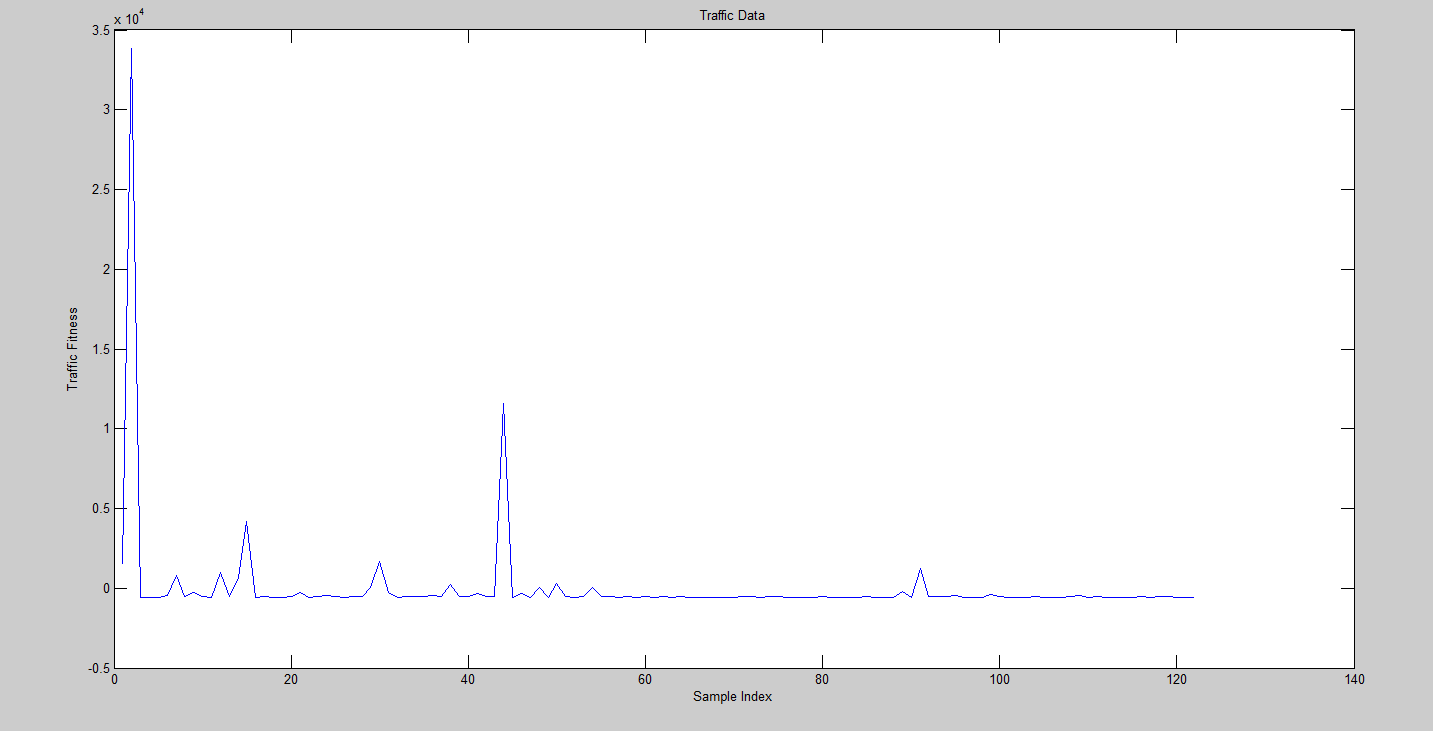
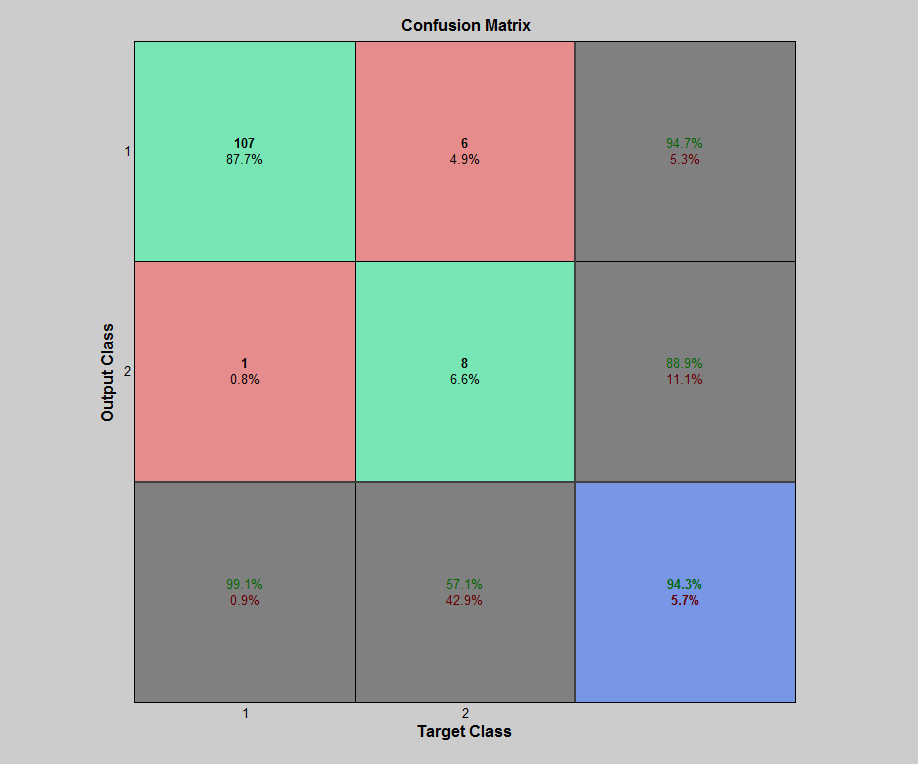
4 of the normal traffic records are incorrectly classified as attack and this corresponds to 3.2% of all 122 network traffic records in the data. In this case, all of the attack records are correctly classified so there is no miss detection in this case.

The network outputs are very accurate, as you can see by the high numbers of correct responses in the green squares and the low numbers of incorrect responses in the red squares. The lower right blue squares illustrate the overall accuracies.

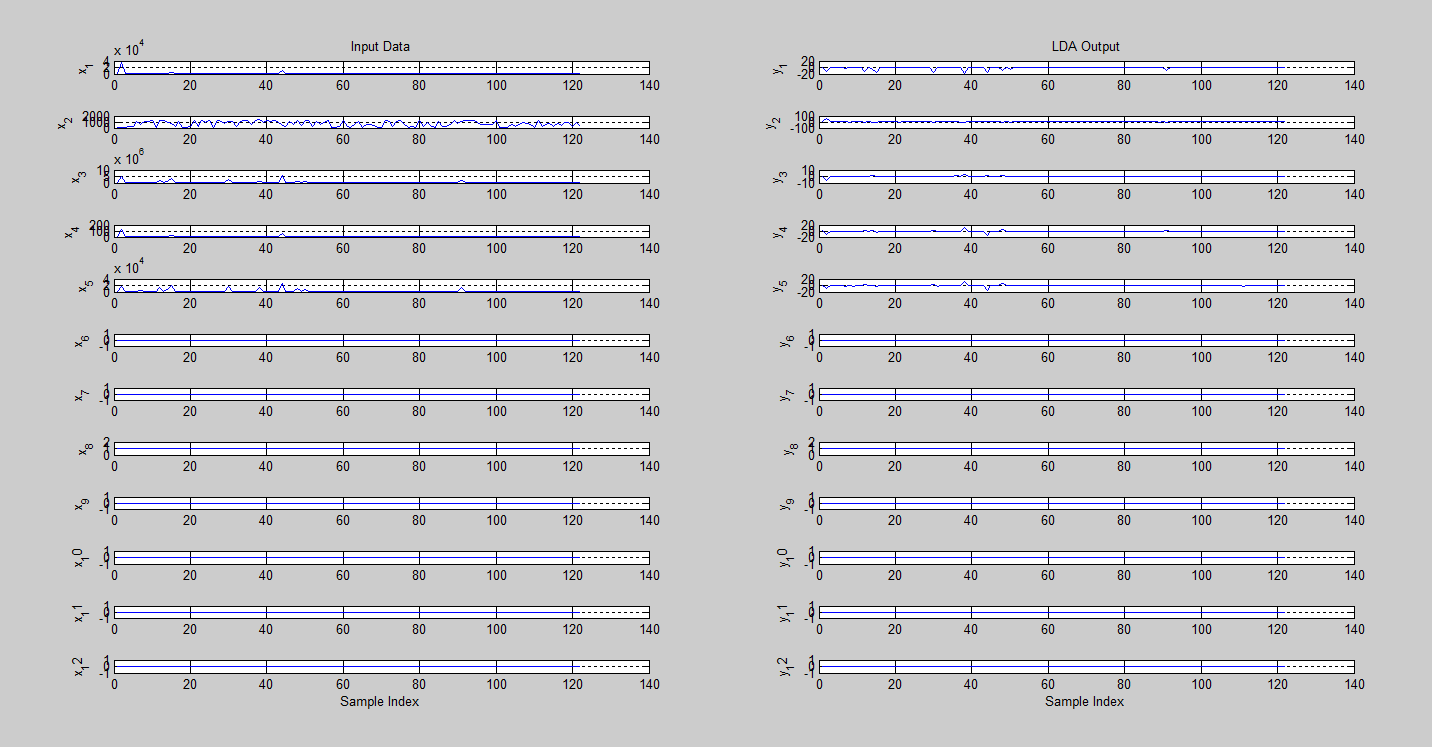
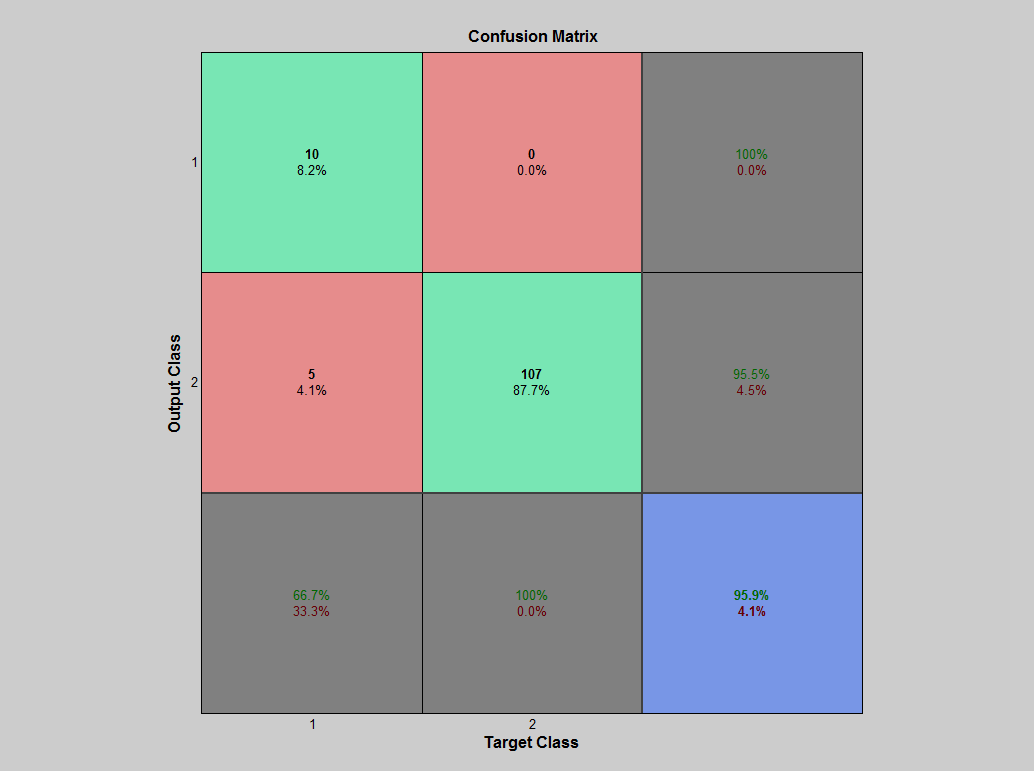
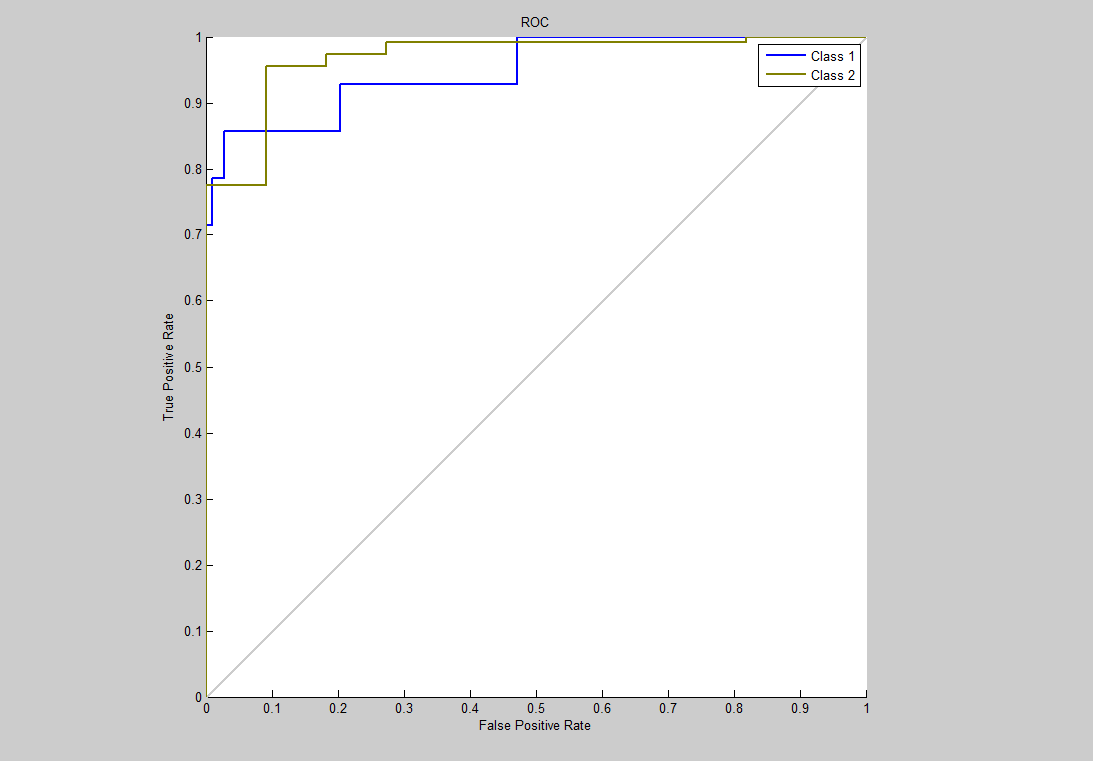
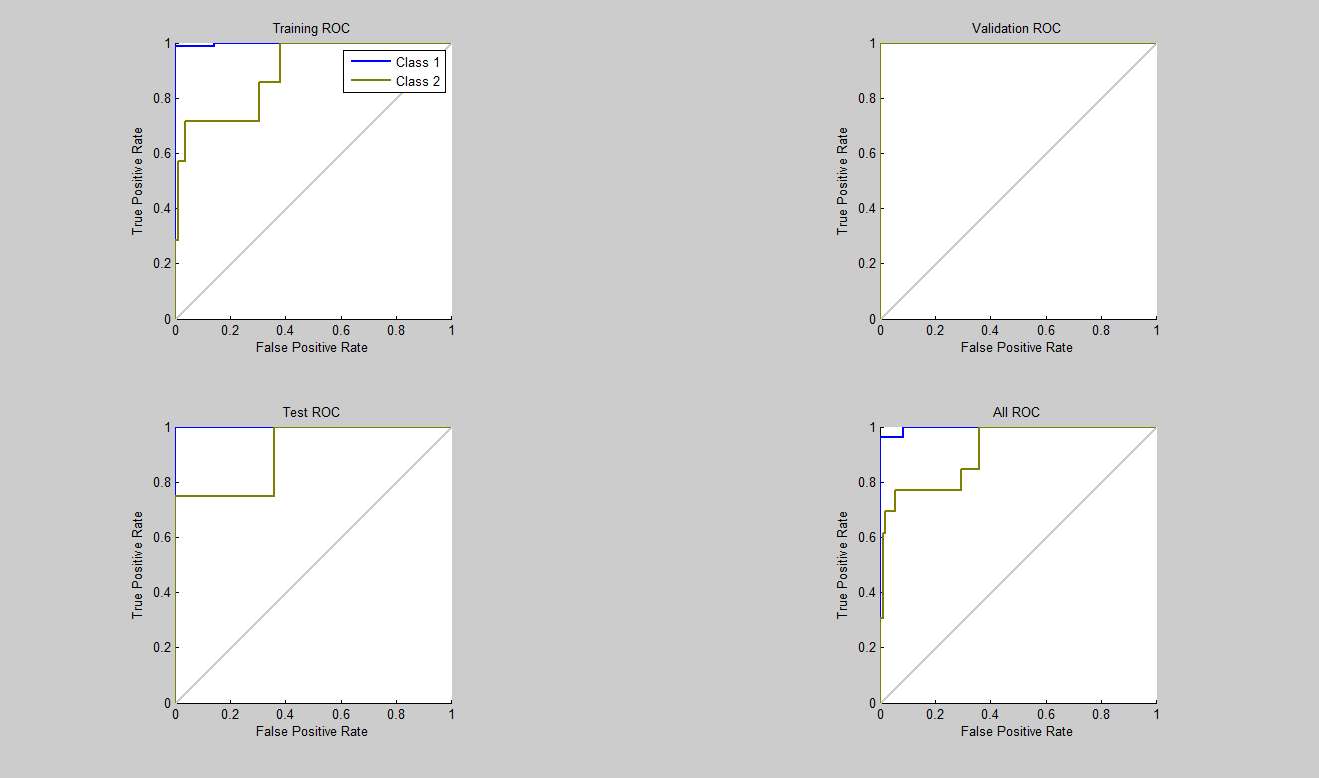
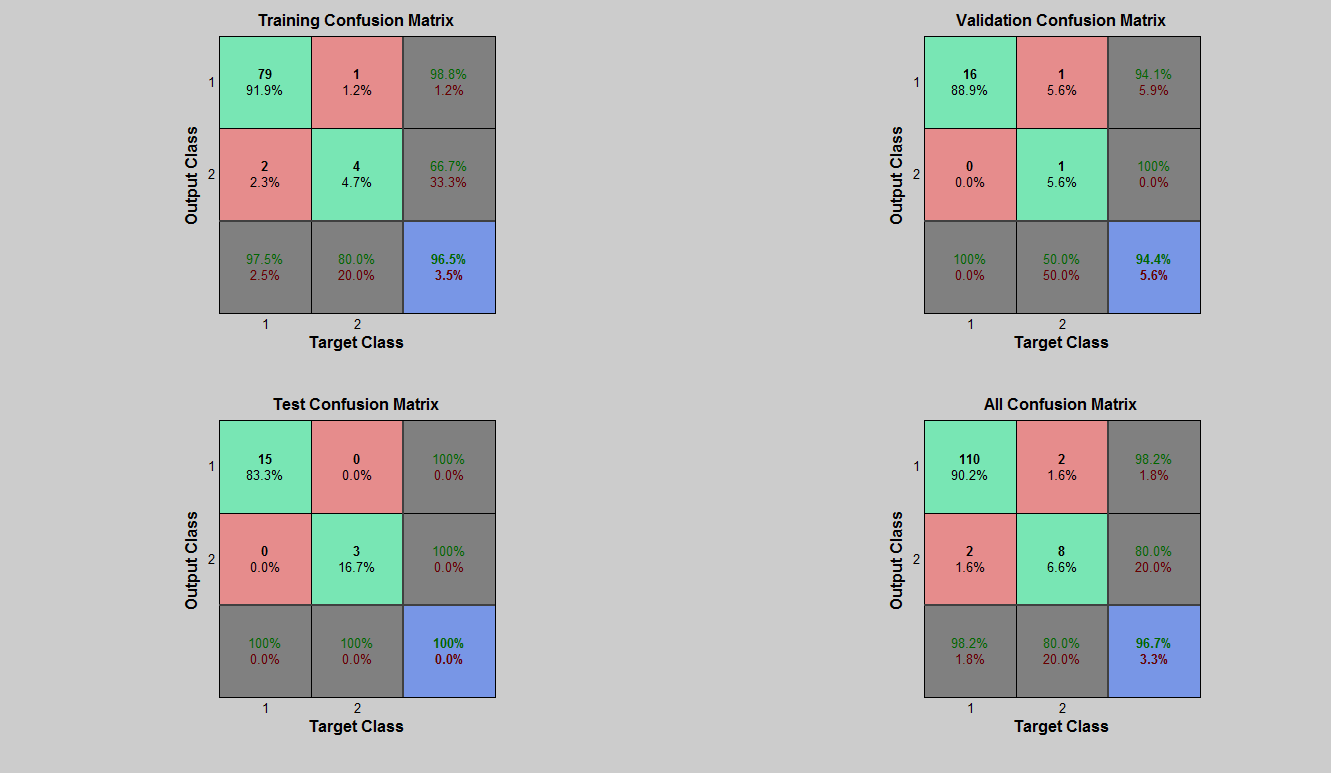
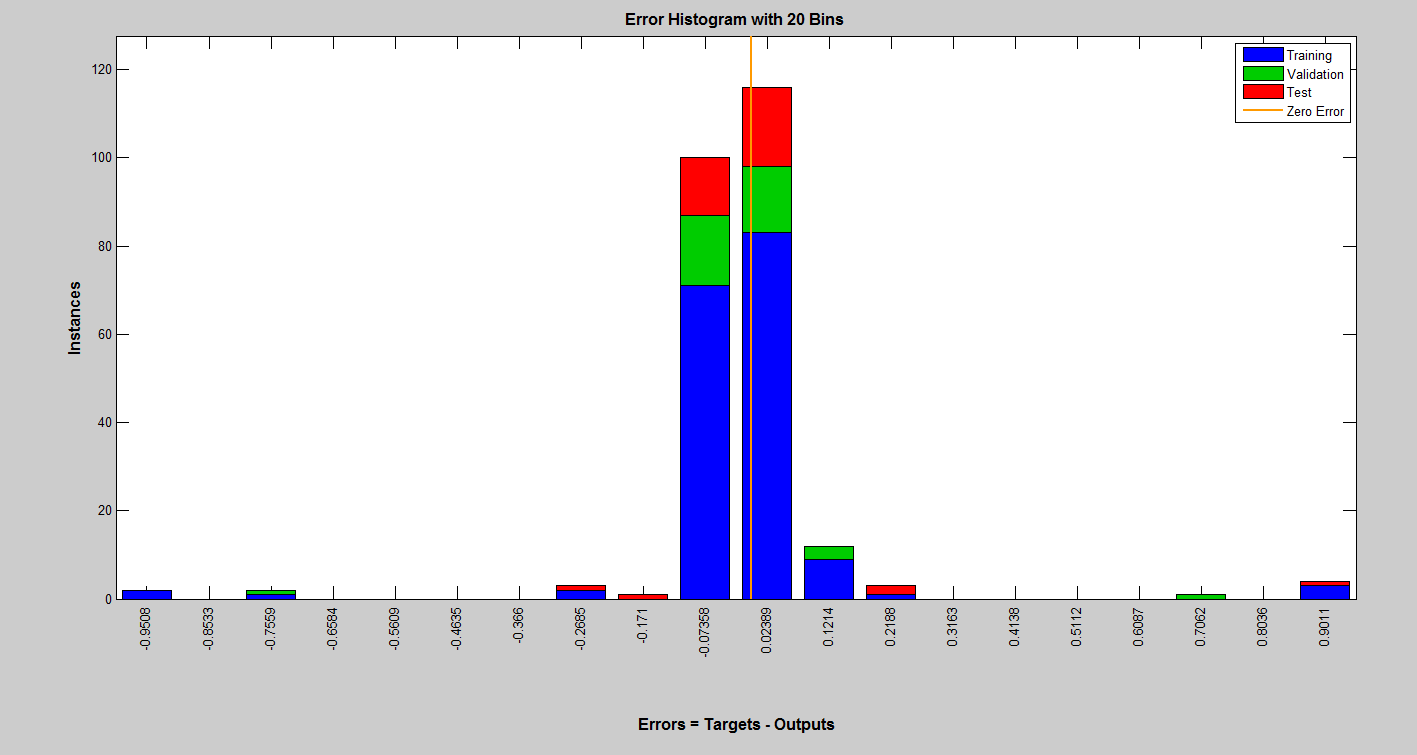
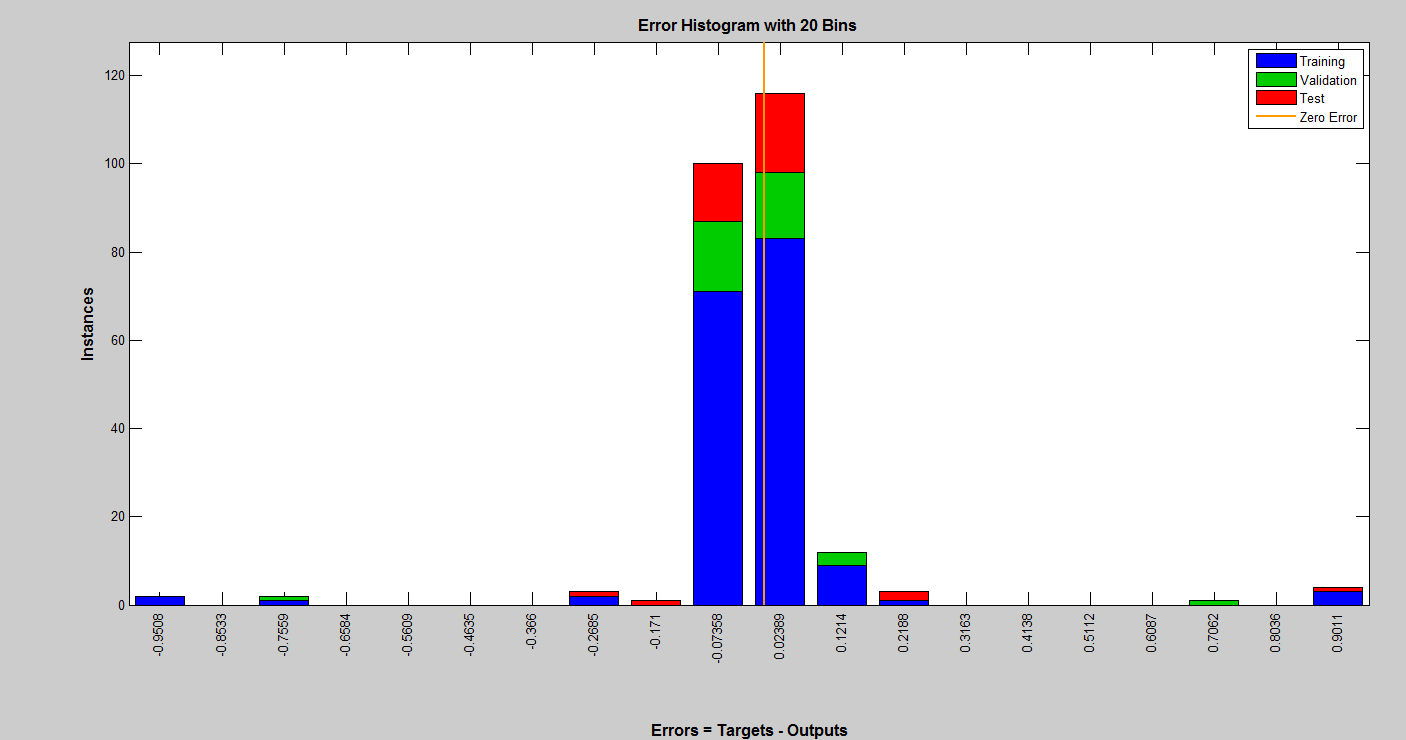
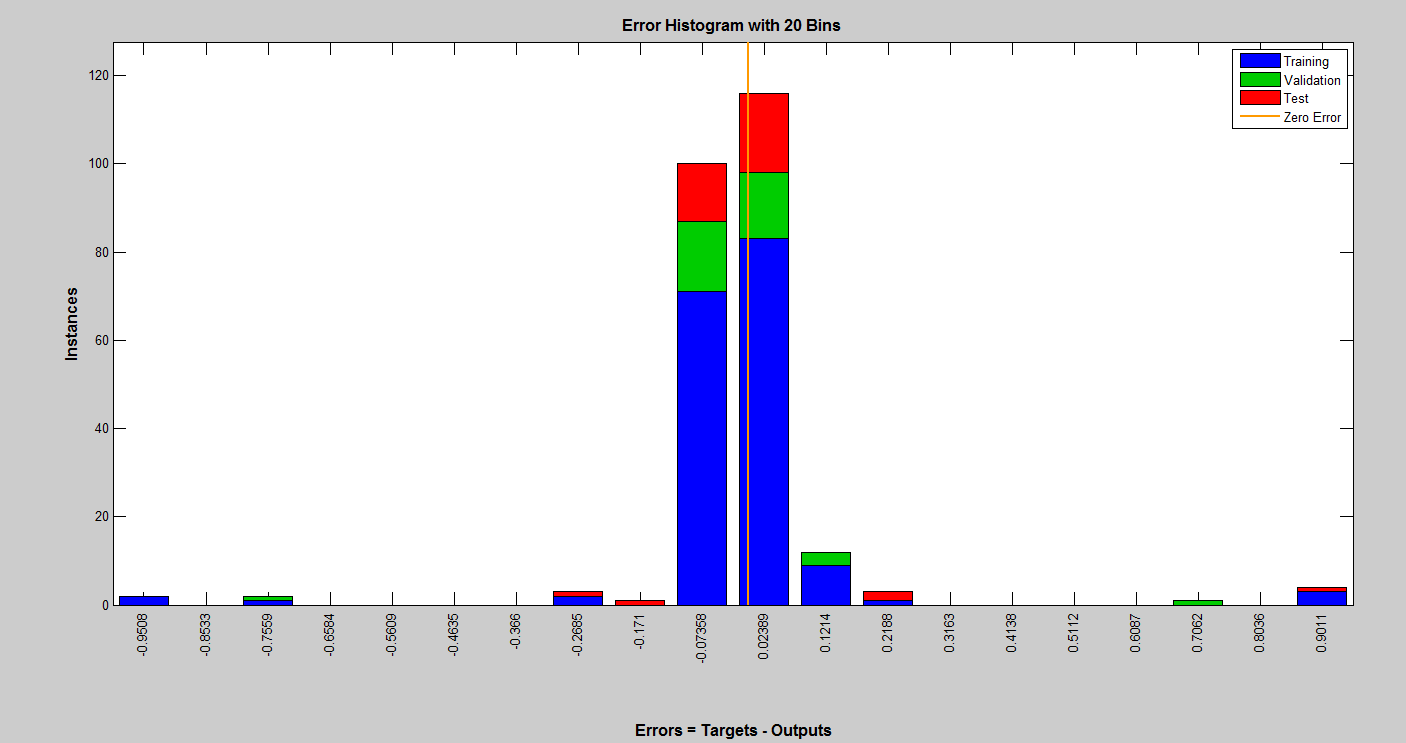
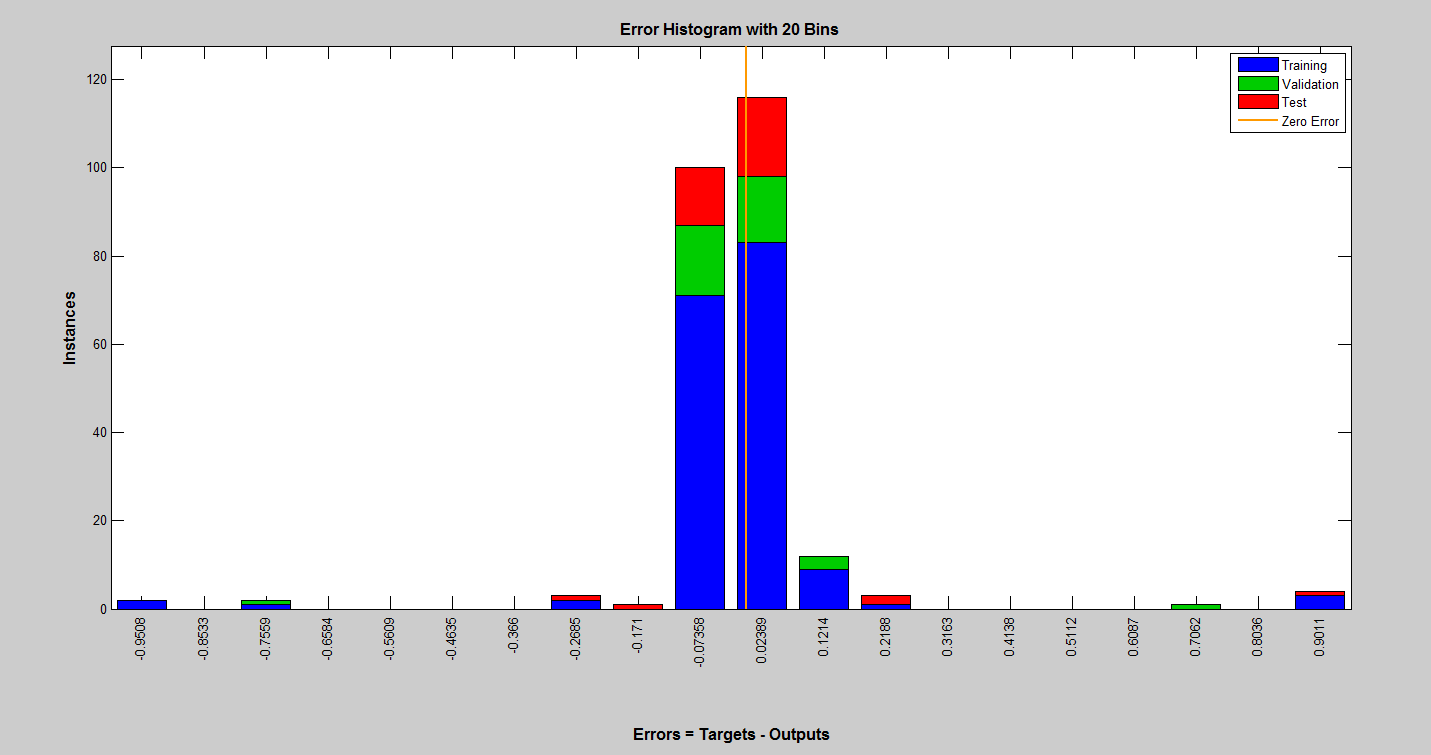
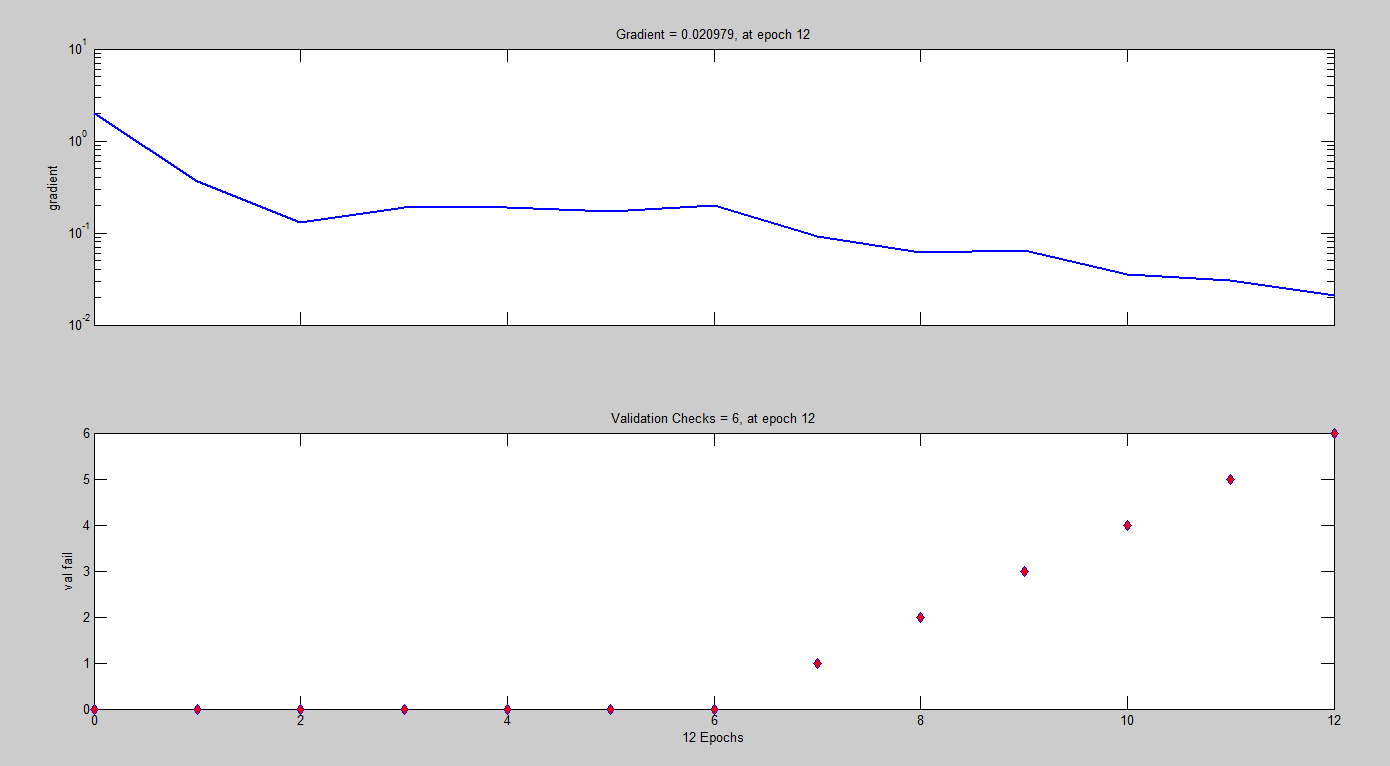
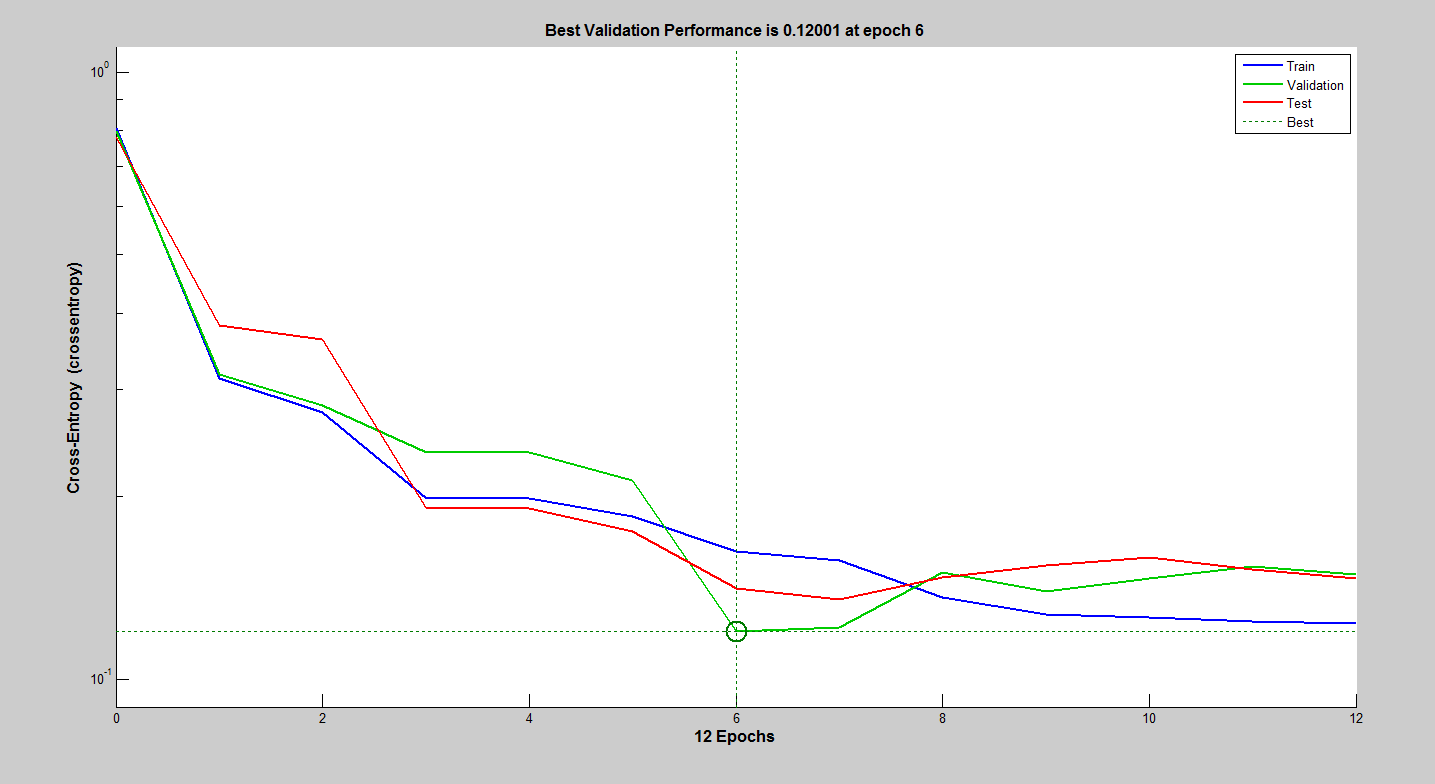
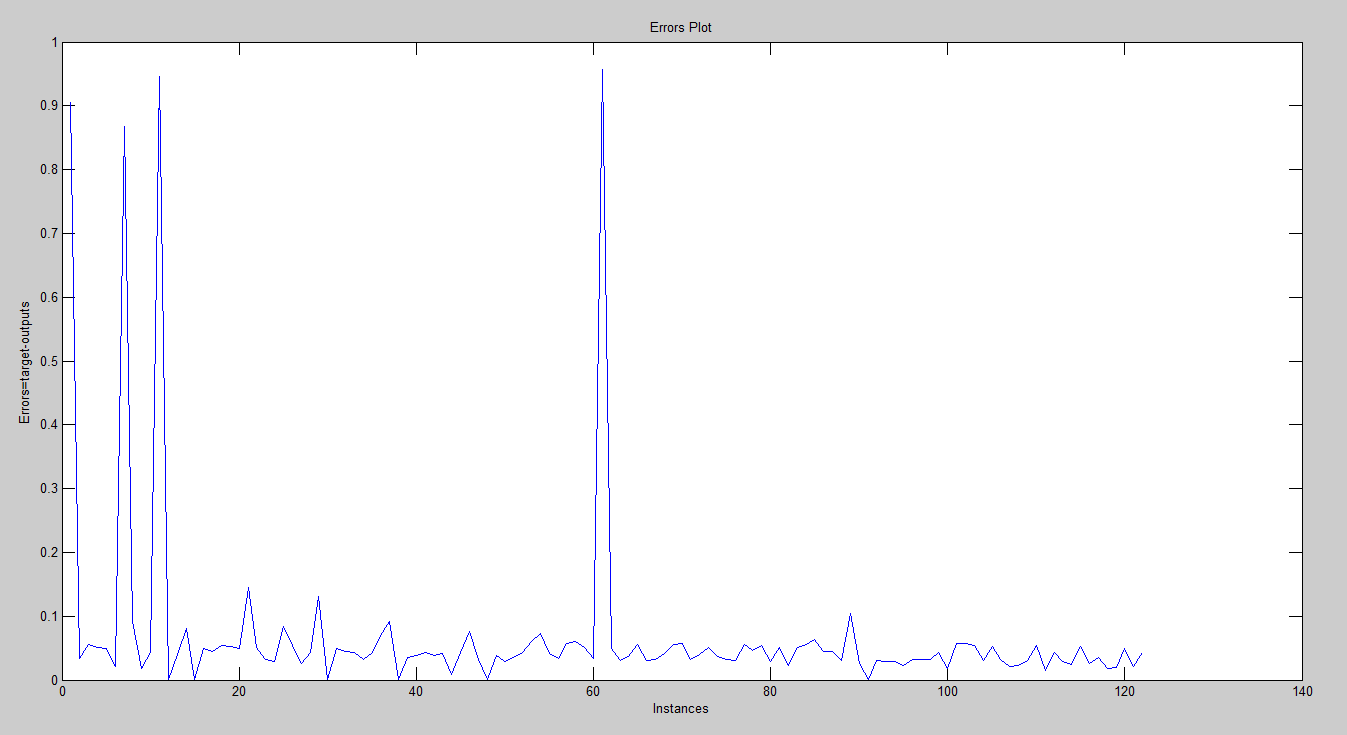
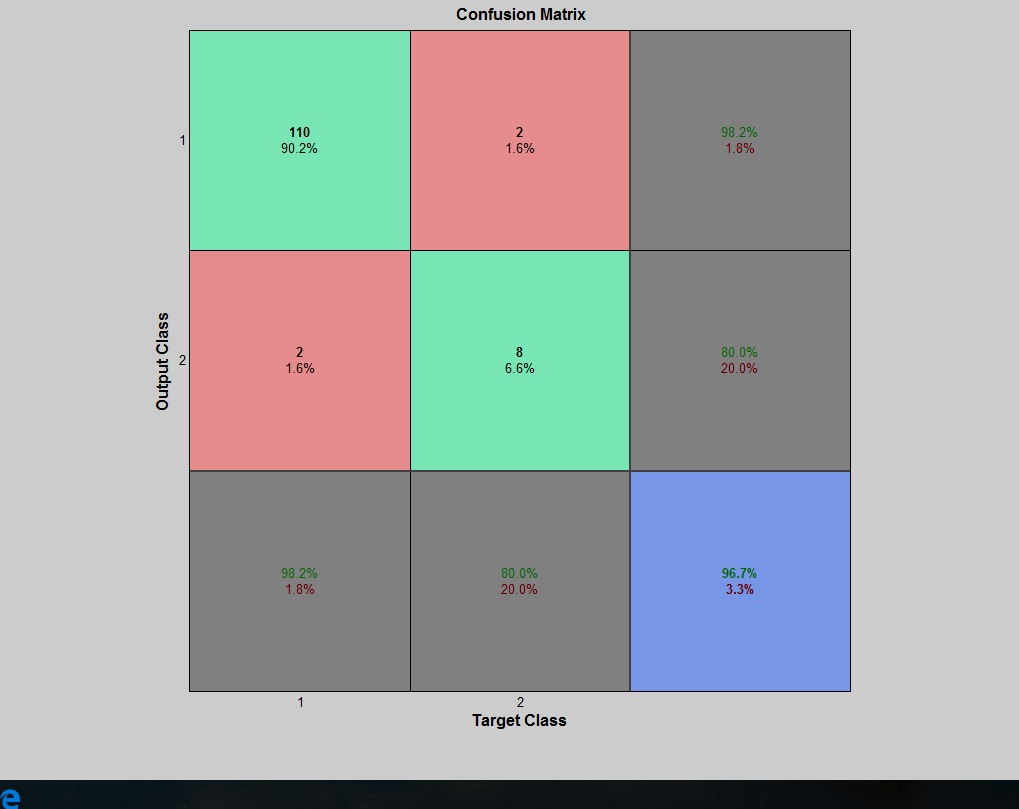
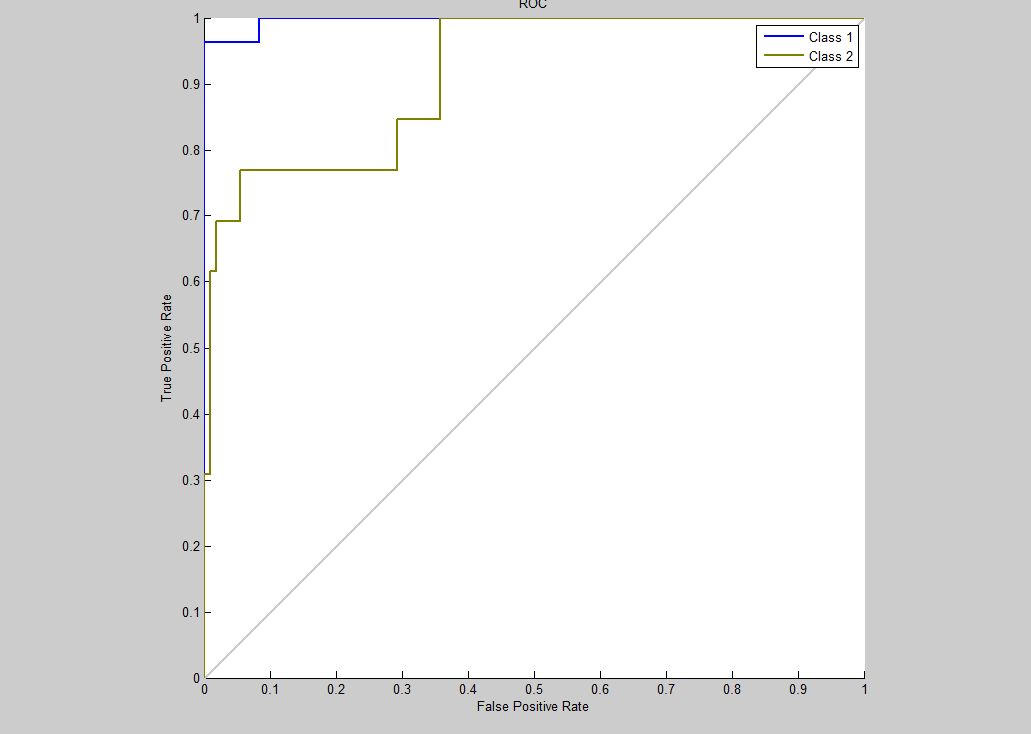
Out of 109 normal predictions, 100% are correct and there is no wrong detections. Out of 252 attack predictions, 99.2% are correct and 0.8% are wrong. Out of 254 normal cases, 99.2% are correctly predicted as normal and 0.8% are predicted as attack. Out of 250 attack cases, 100% are correctly classified as attack and 0% are classified as normal.

In the above figure, the first two diagonal cells show the number and percentage of correct classifications by the trained network. For example, 108 normal network traffic are correctly classified as normal. This corresponds to 88.5% of all 122 network traffic record. Similarly, 9 cases are correctly classified as attack. This corresponds to 7.4% of all network traffic.

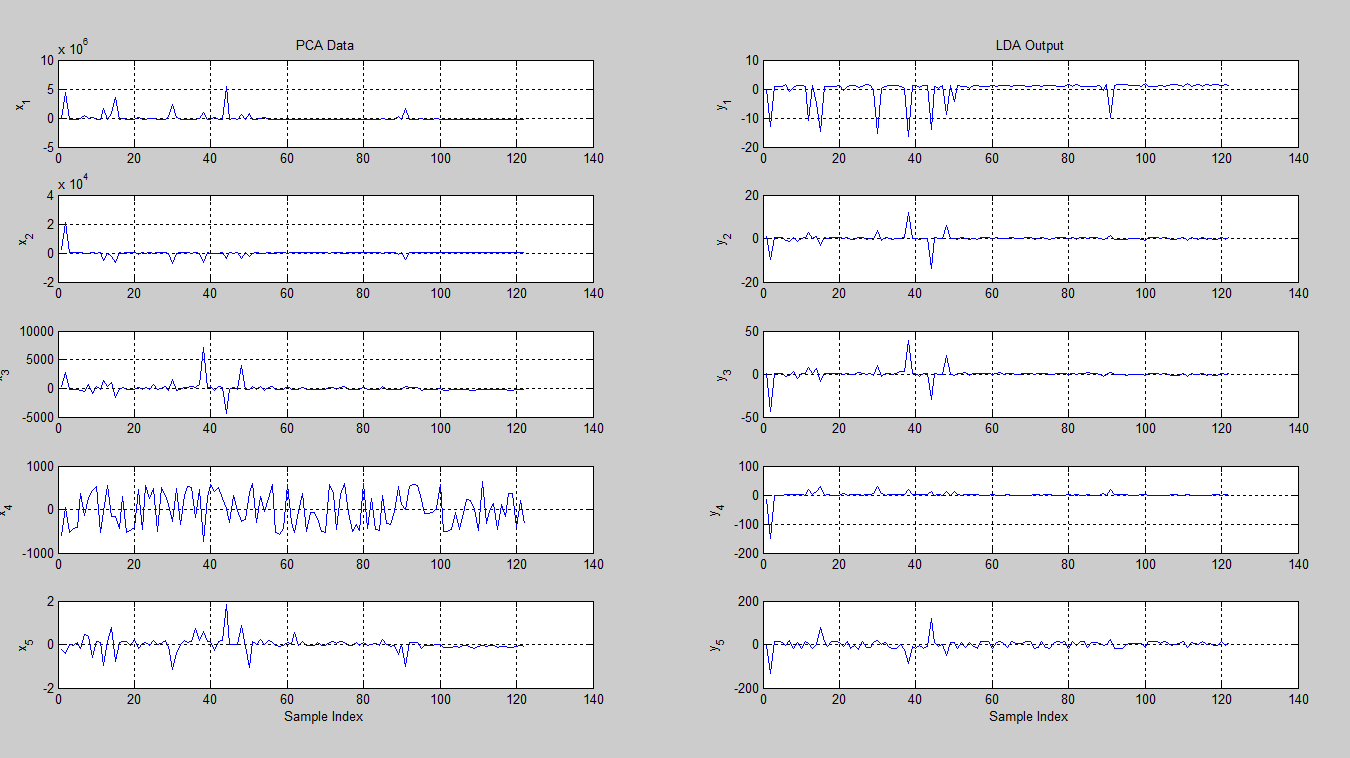
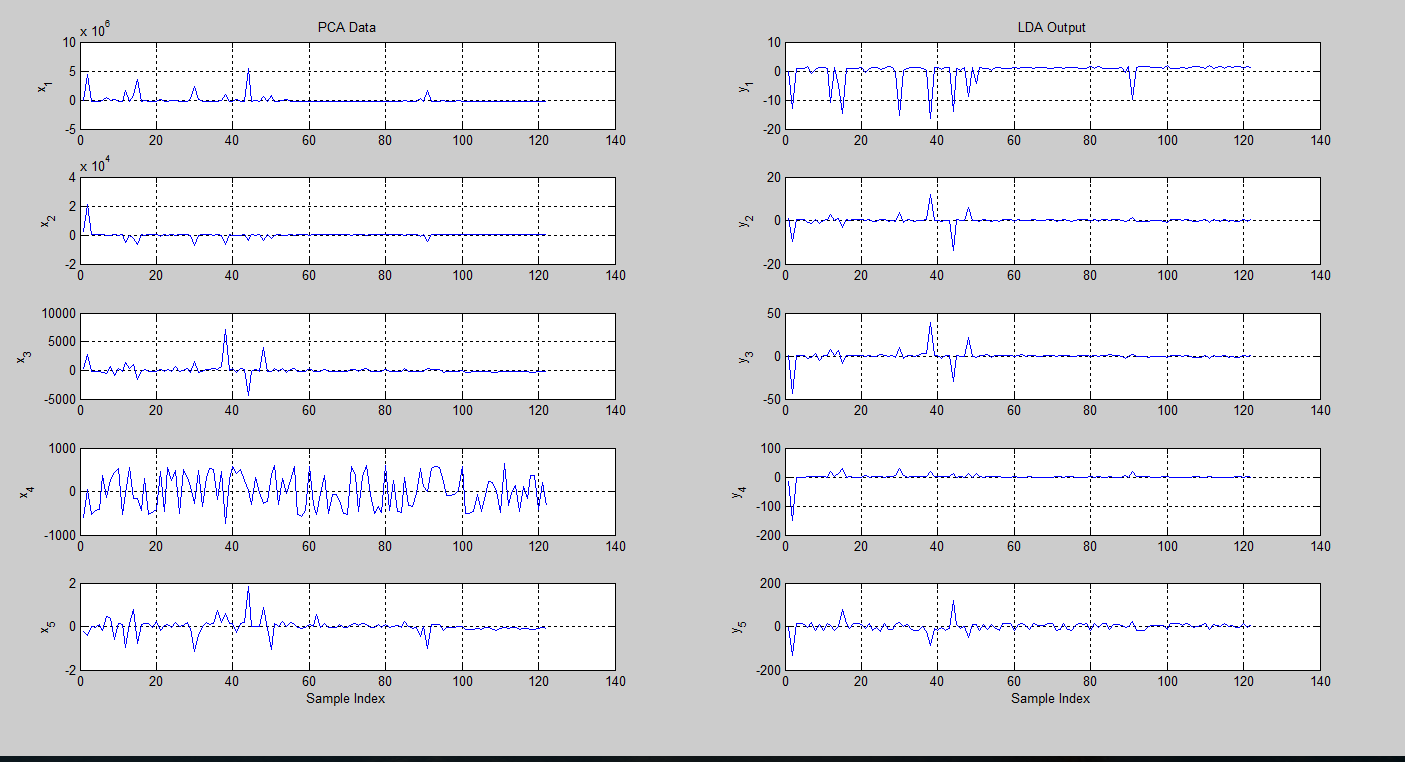
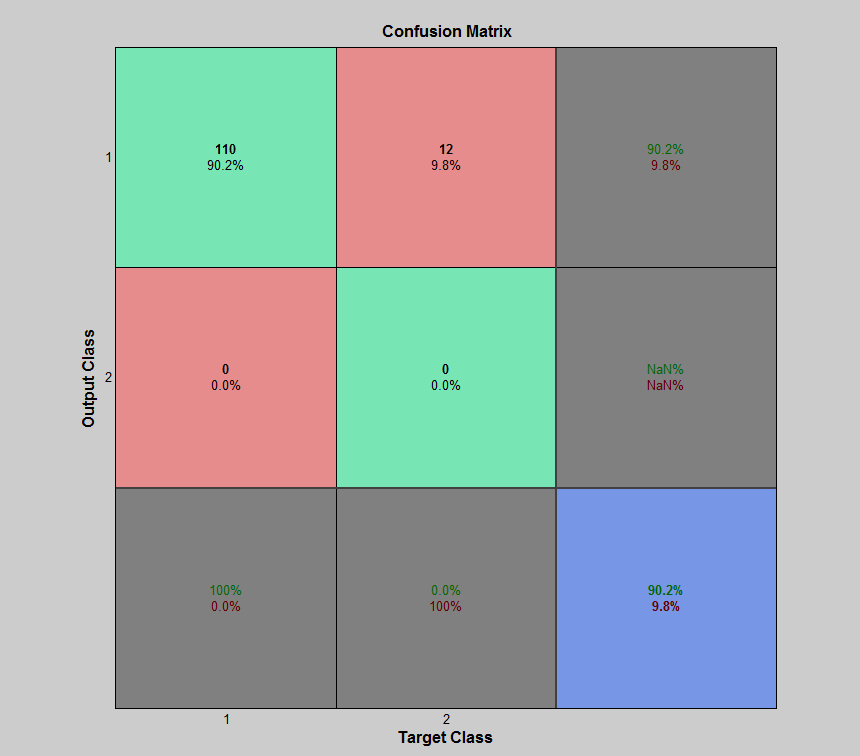
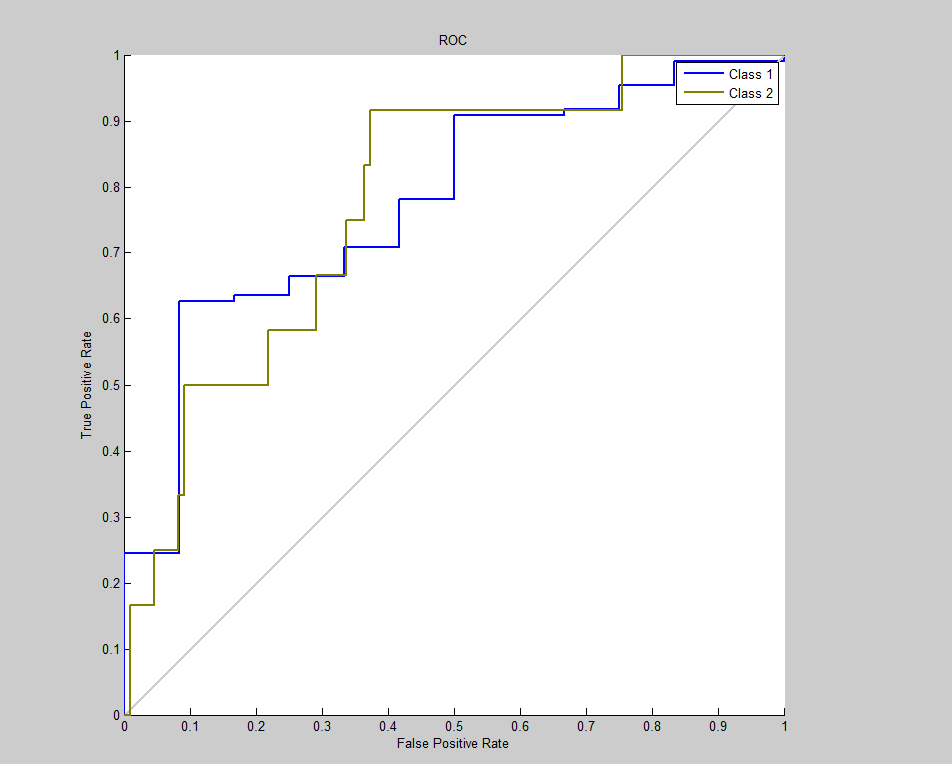
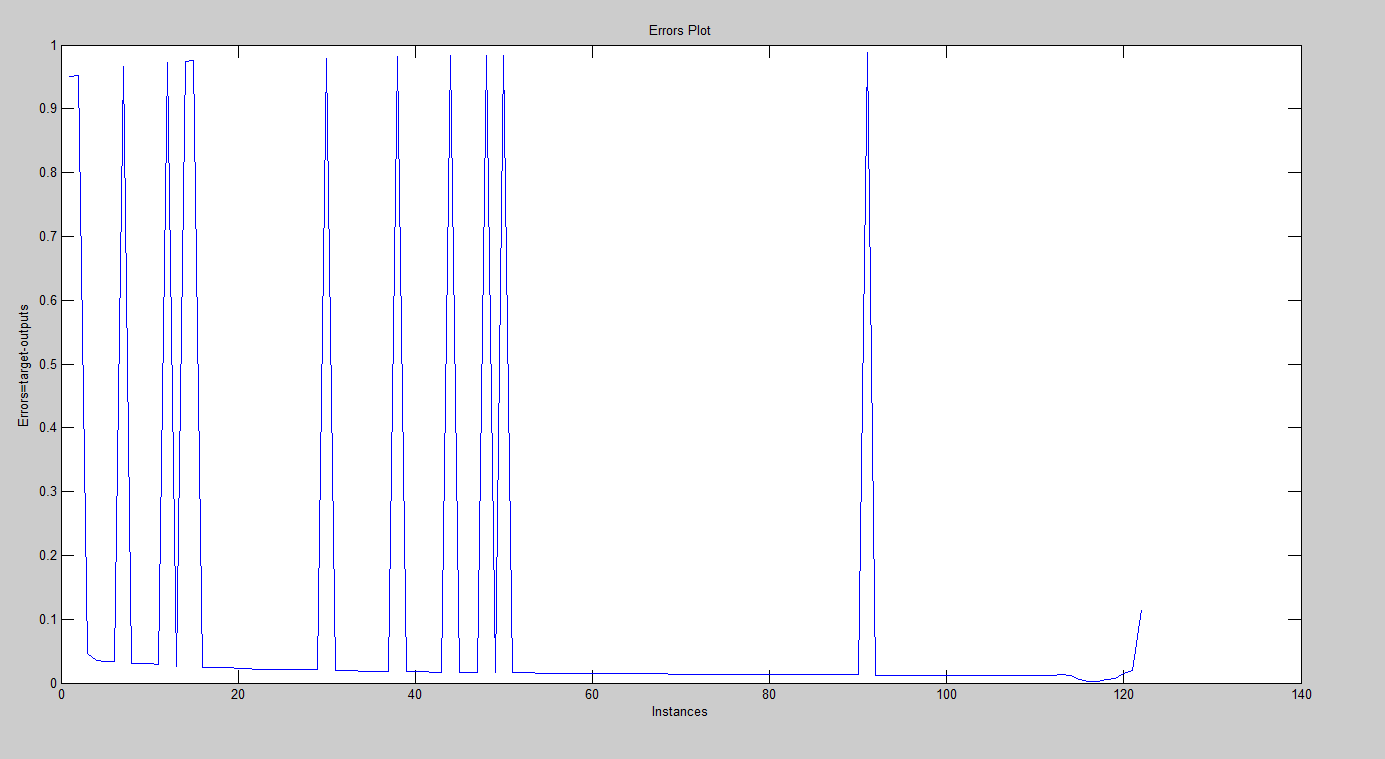
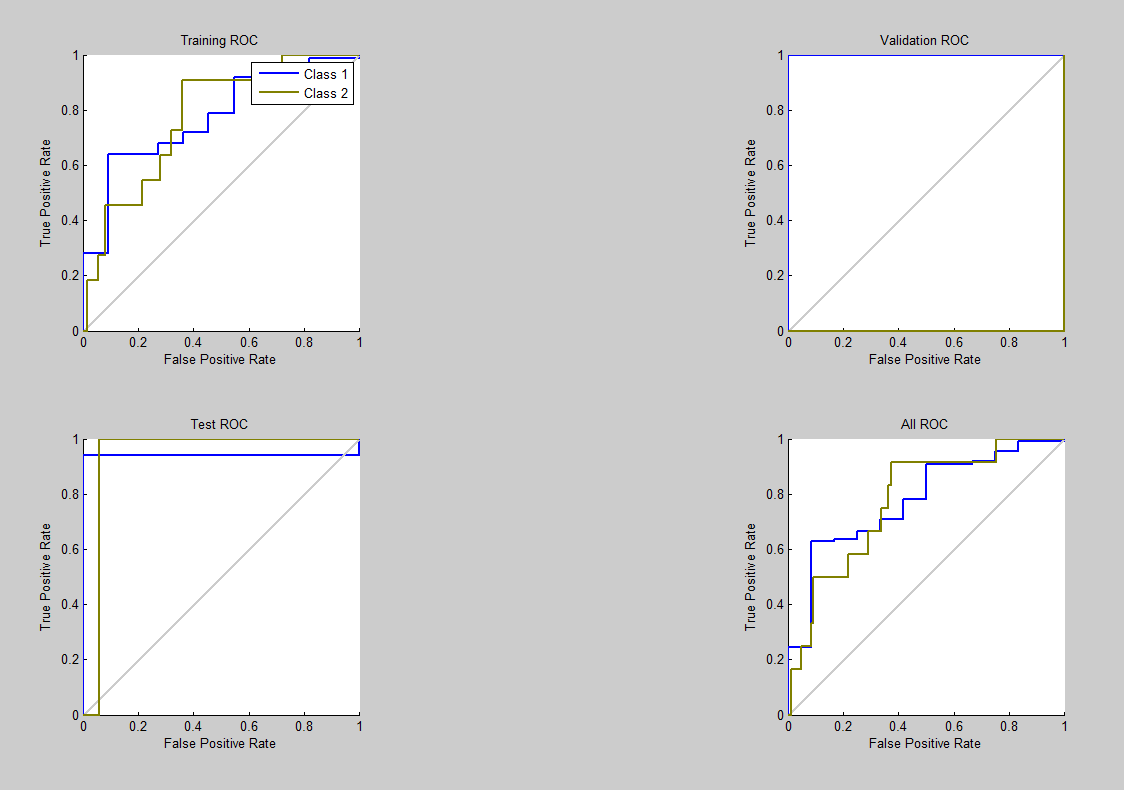
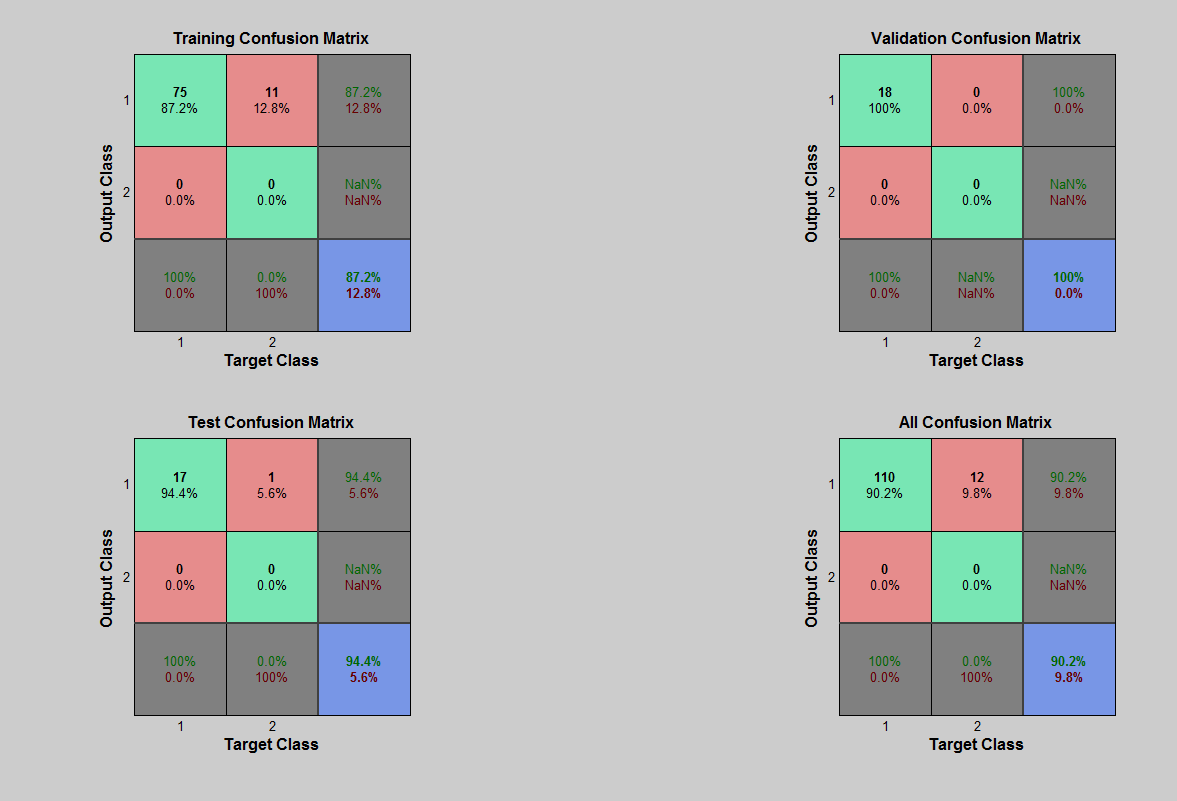
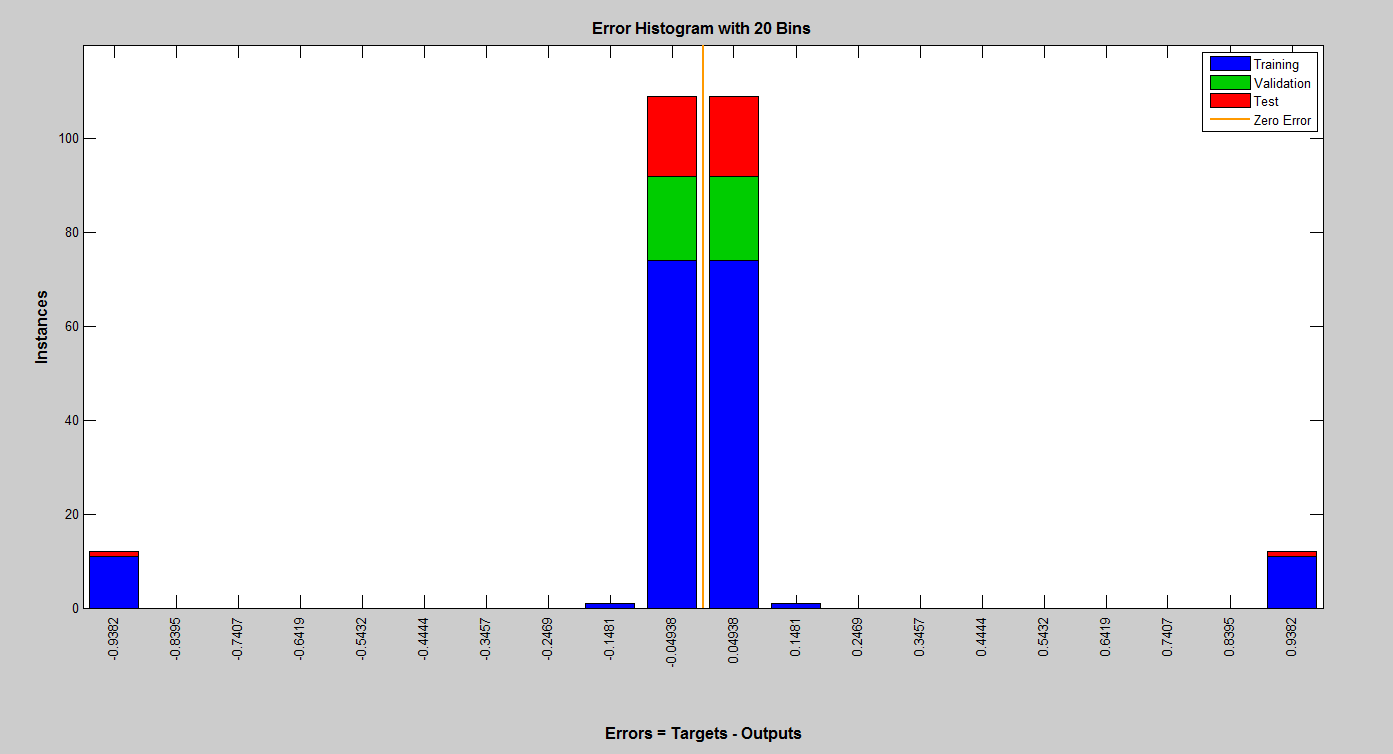
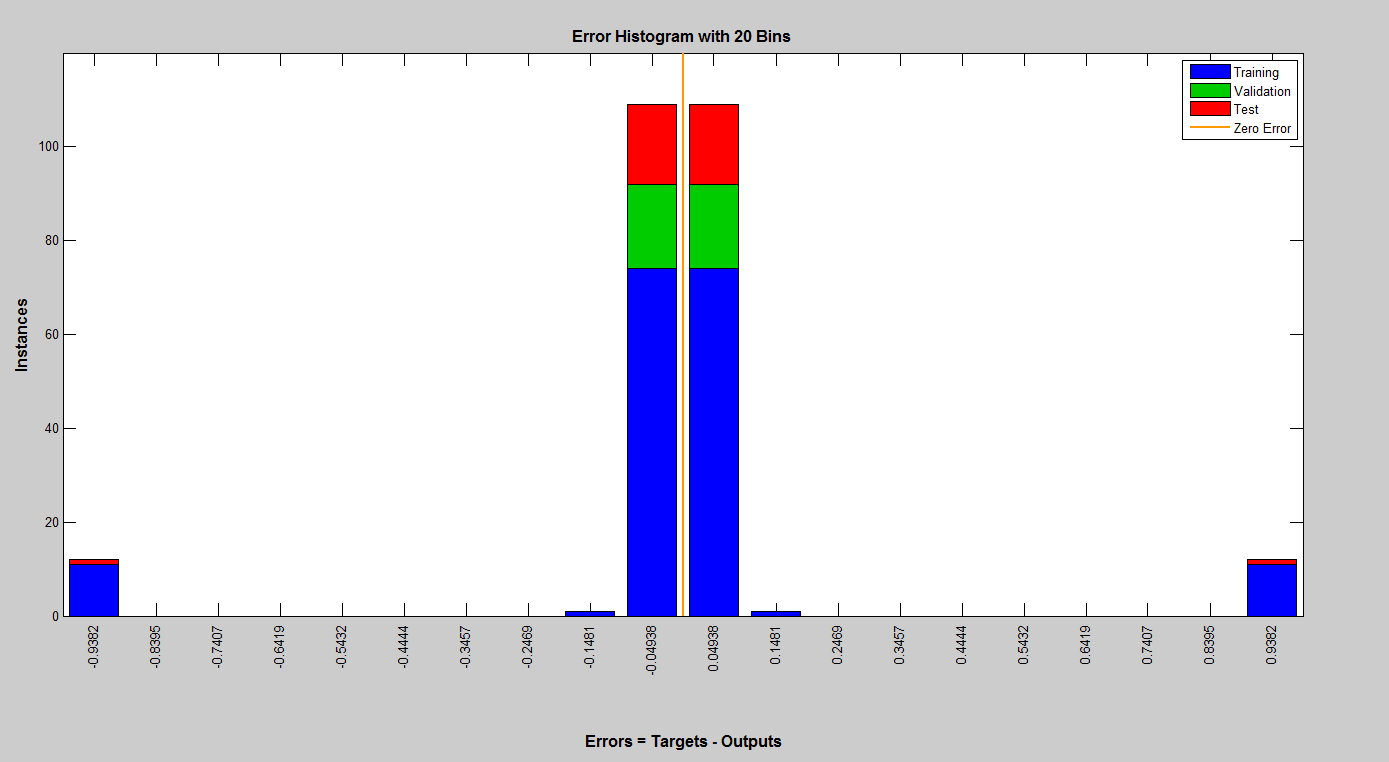
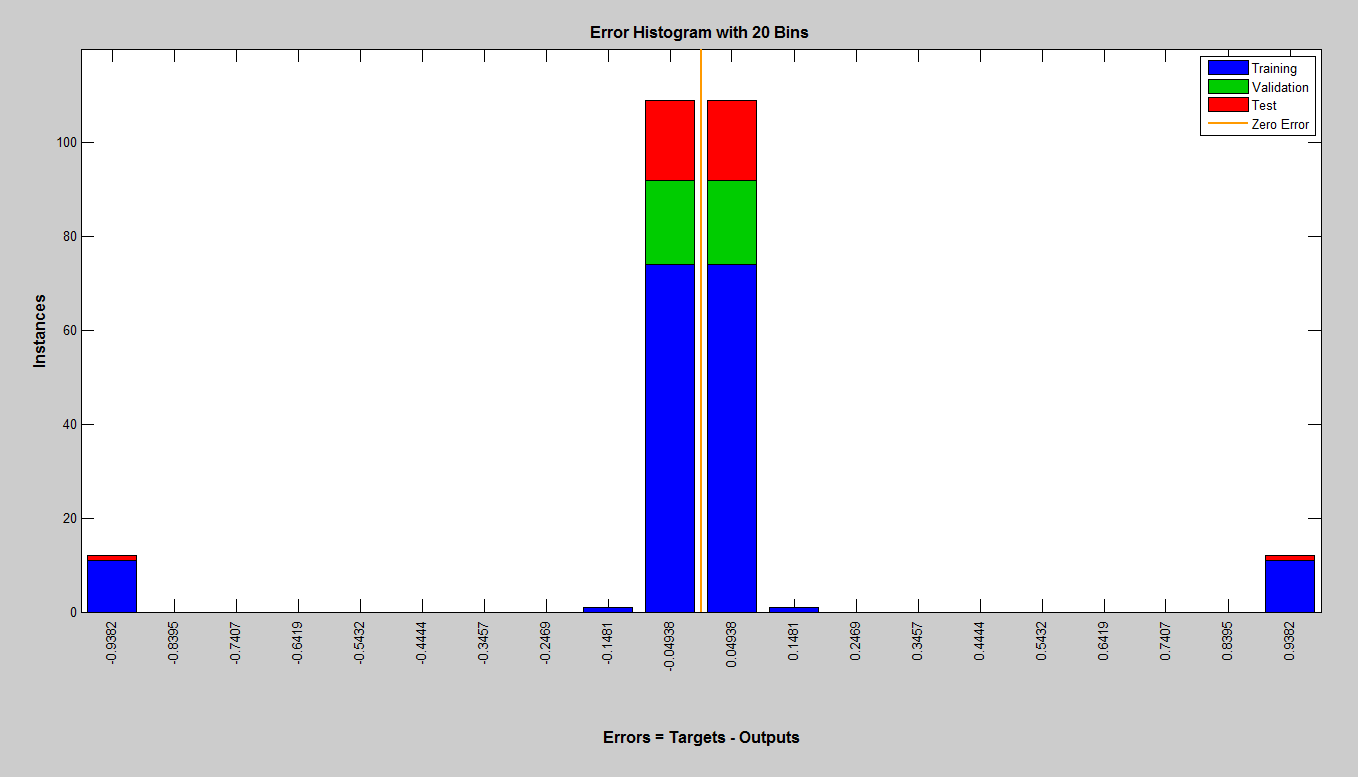
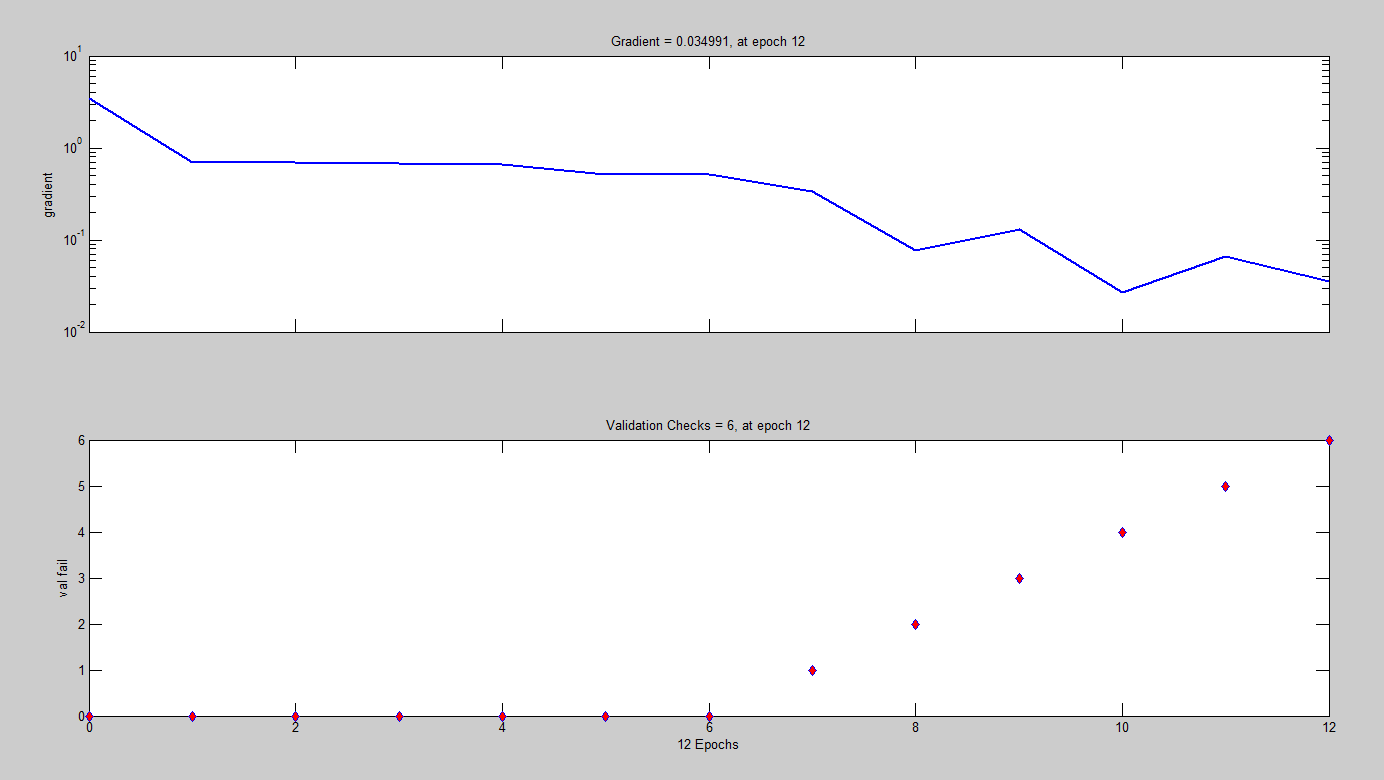
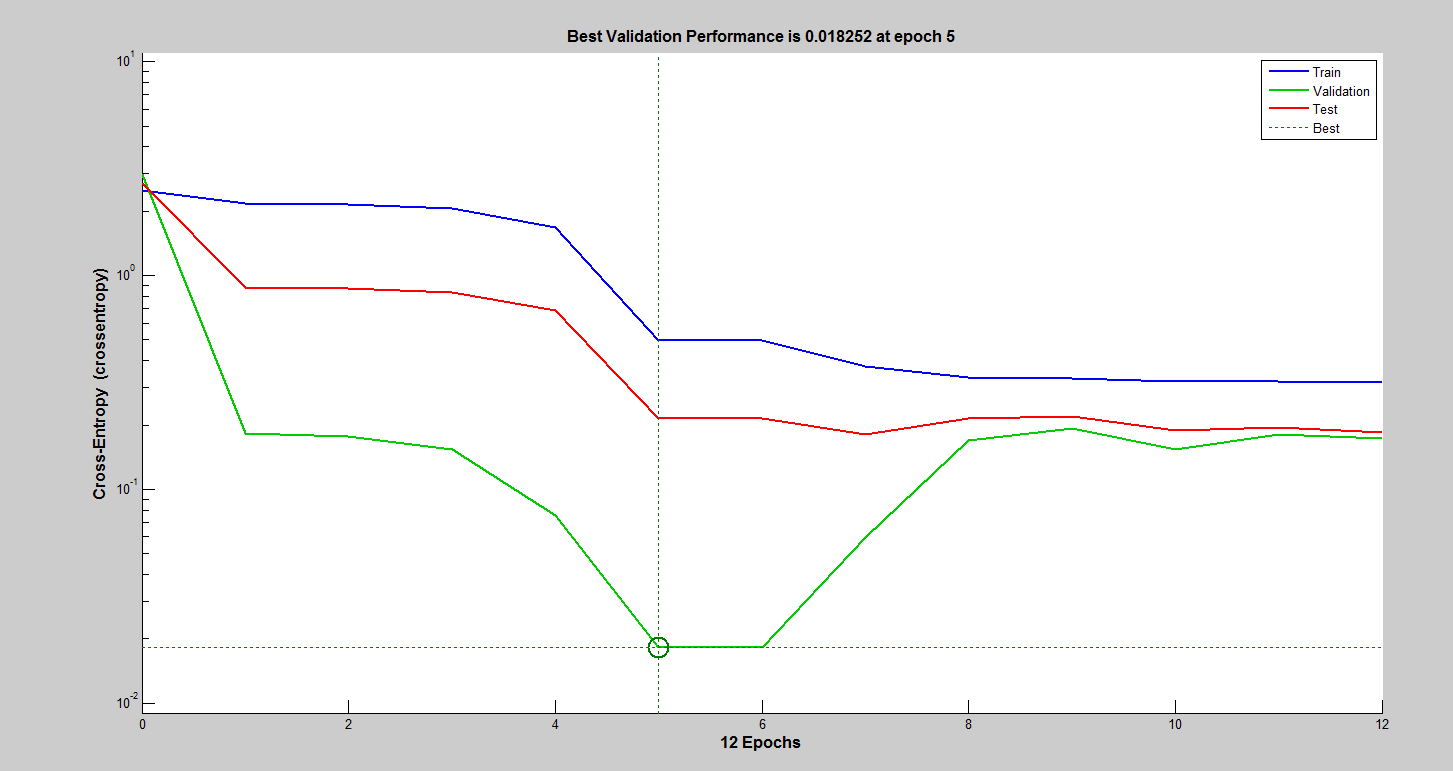
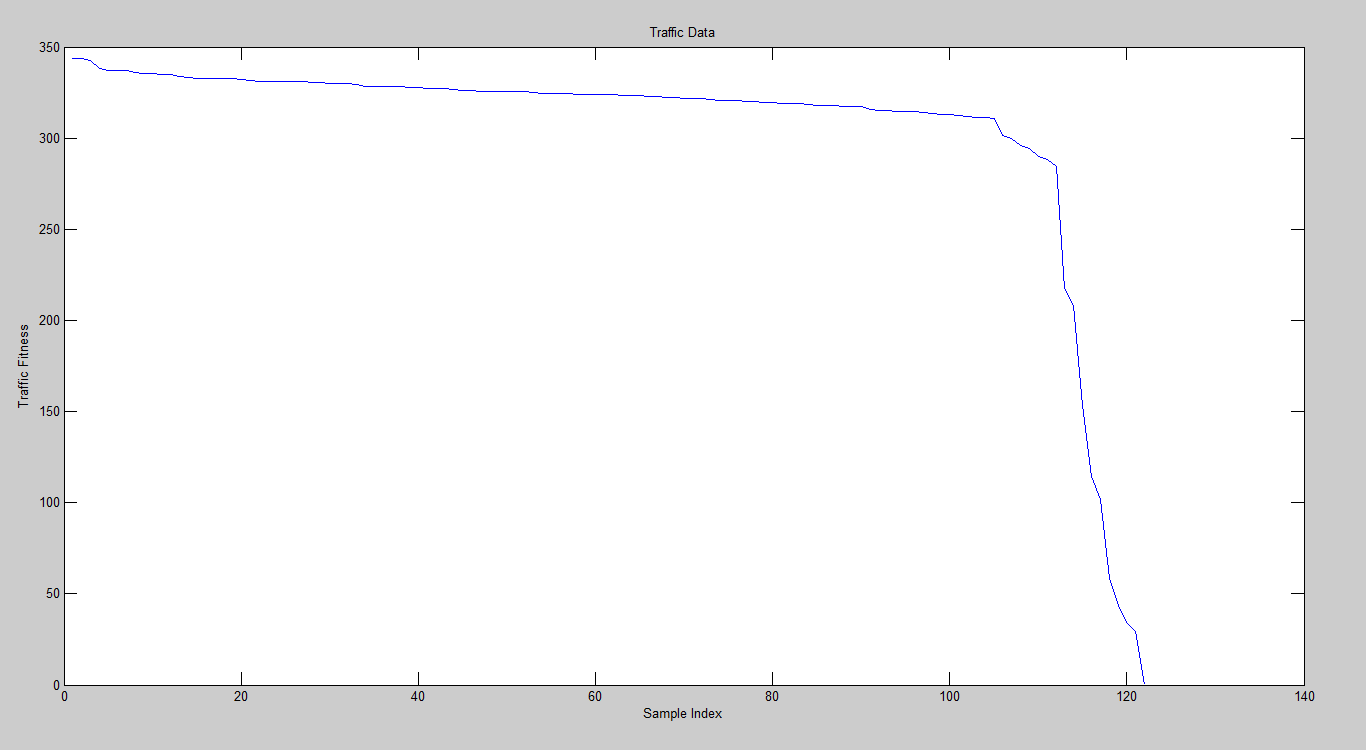
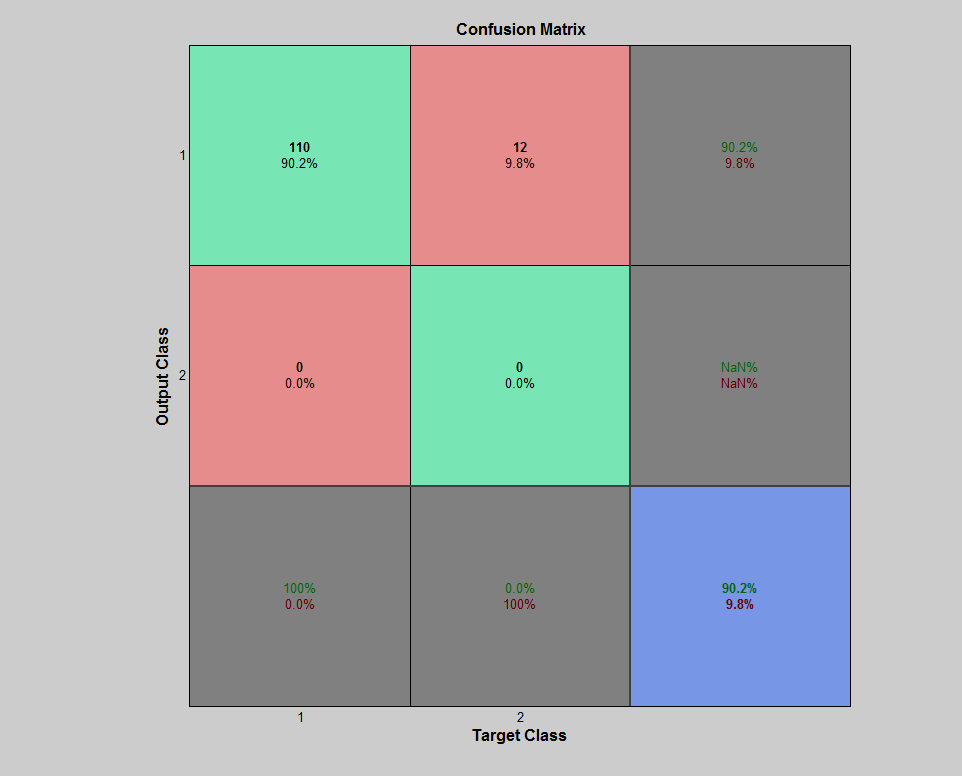
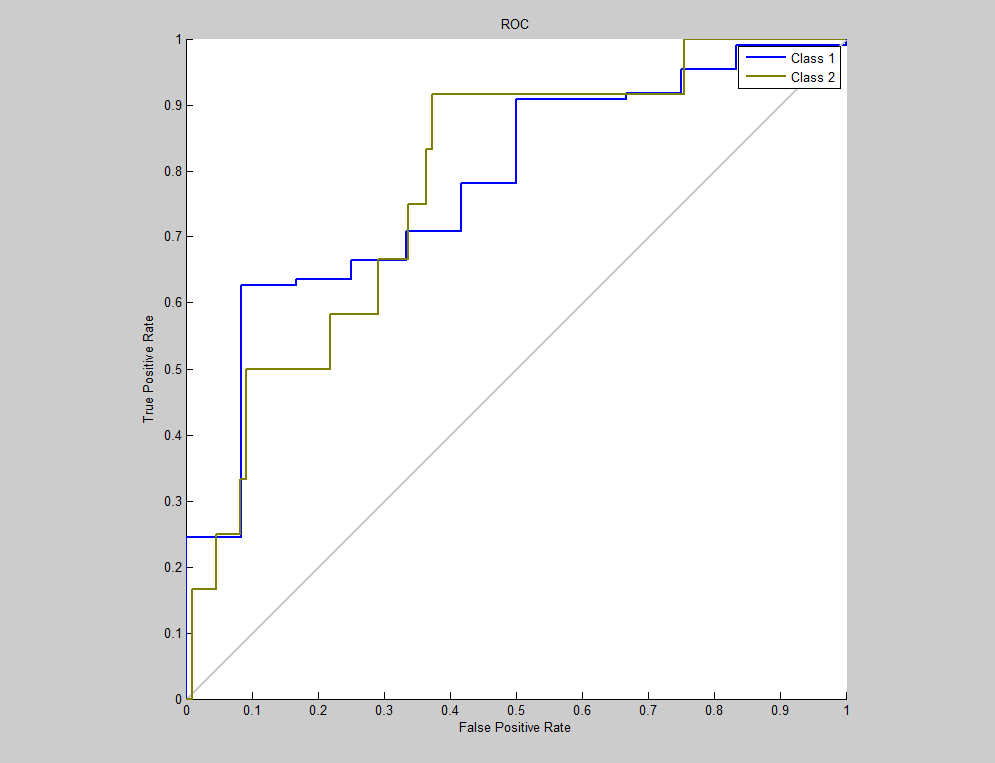
As a result, the trained model made in total 117 correction predictions, including 108 normal traffic and 9 attack attempts. There are in total 1 instances where the model misclassifies the normal traffic record to be attack traffic (false positive). There are in total 4 instances where the model misclassifies the normal traffic record to be attack traffic (false negative). This means the overall classification result has achieve 95.9% of accuracy, with the overall error rate to be 4.1%.



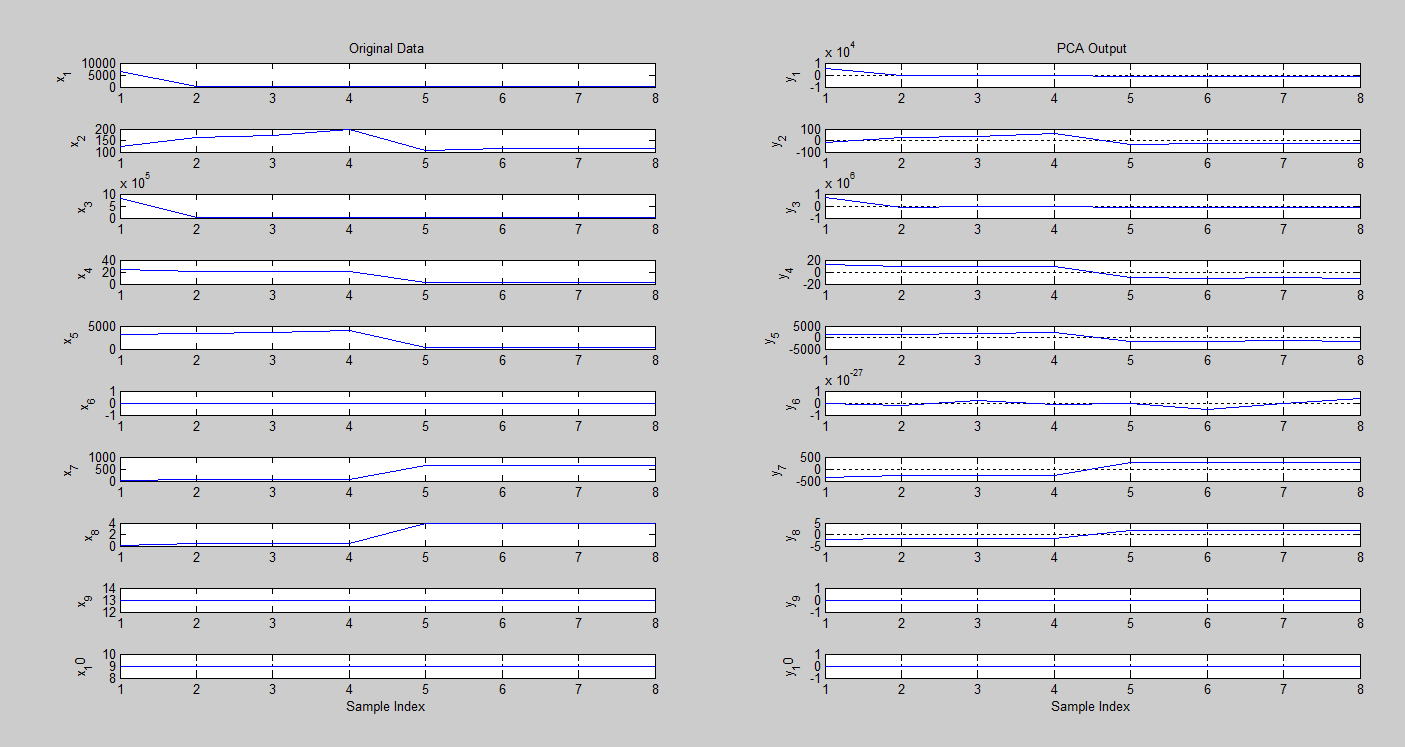
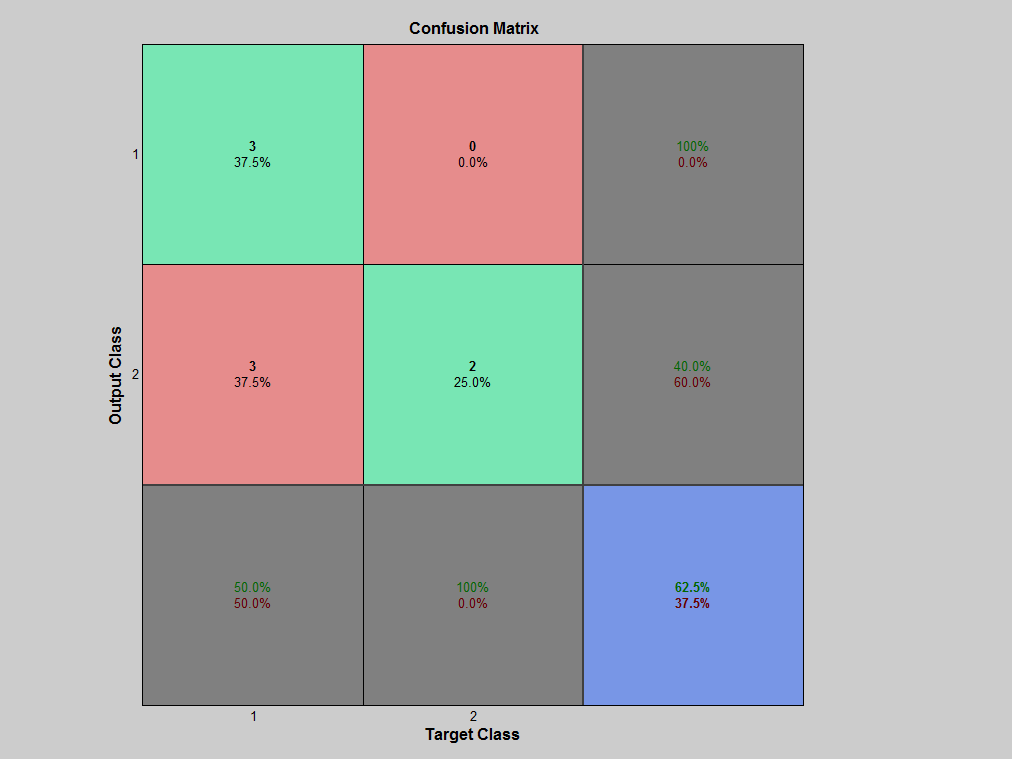
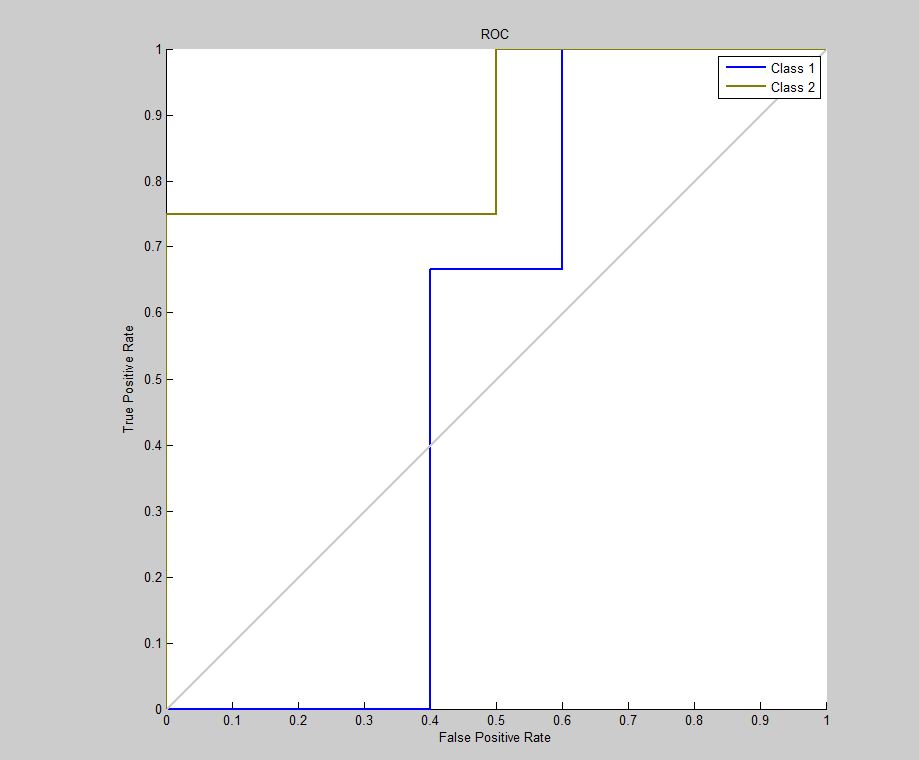
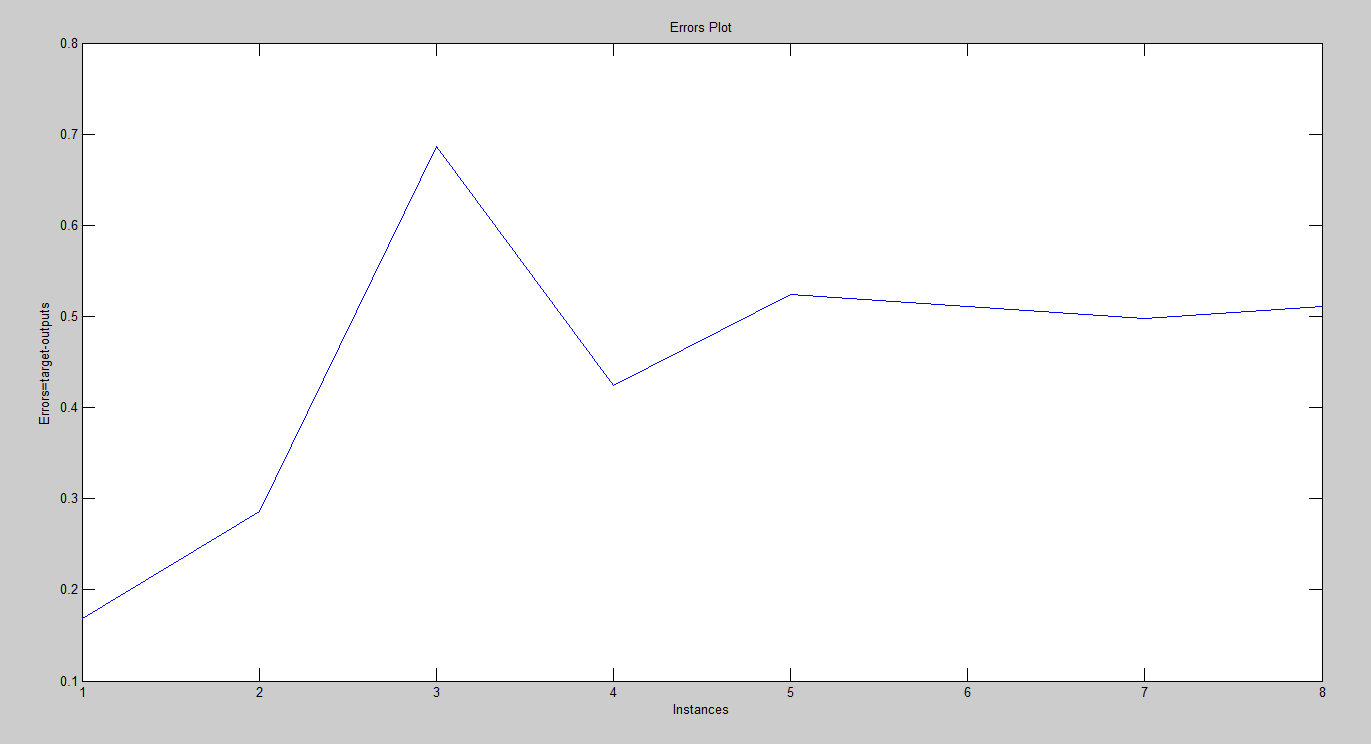
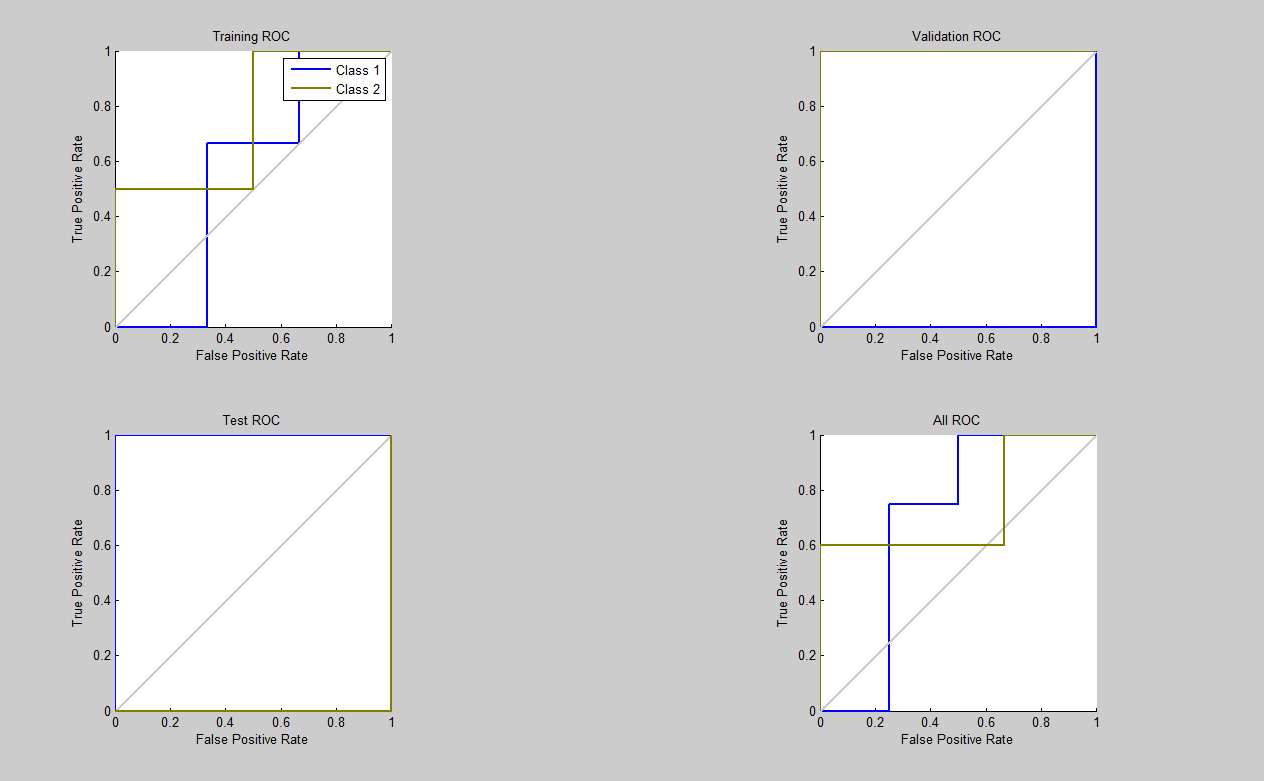
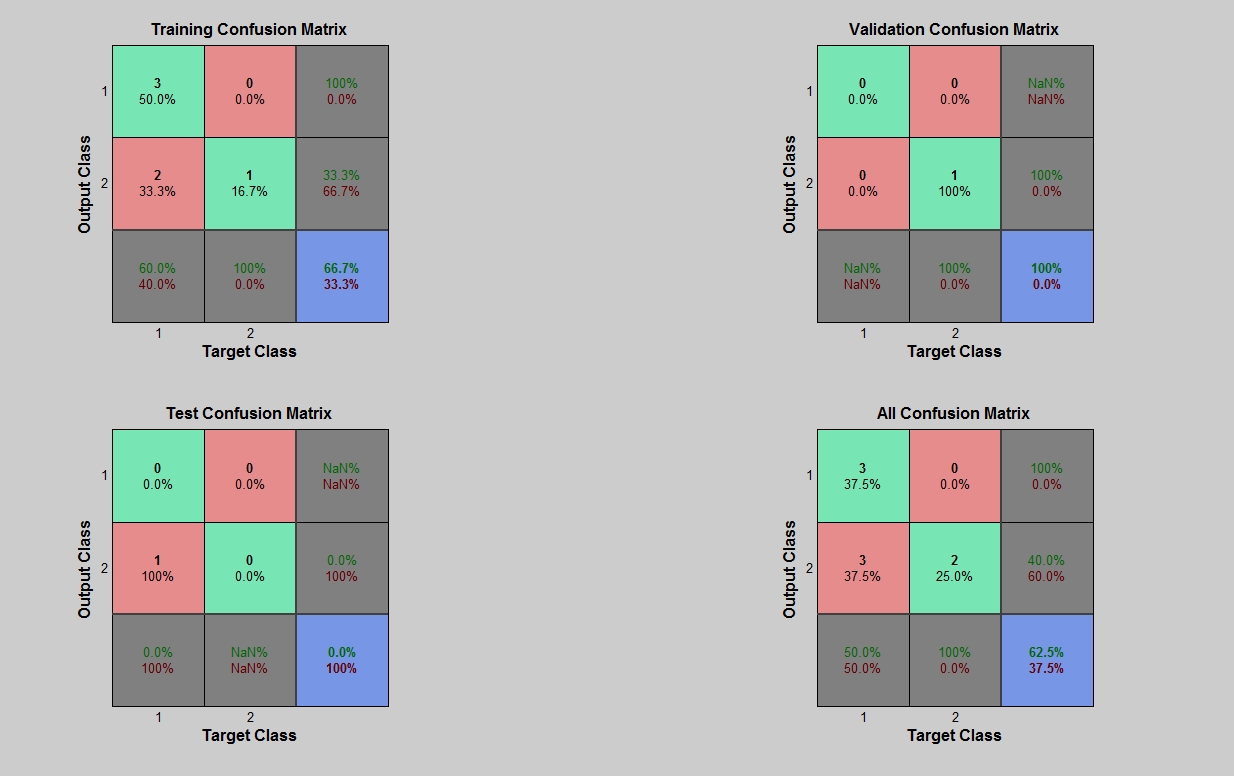
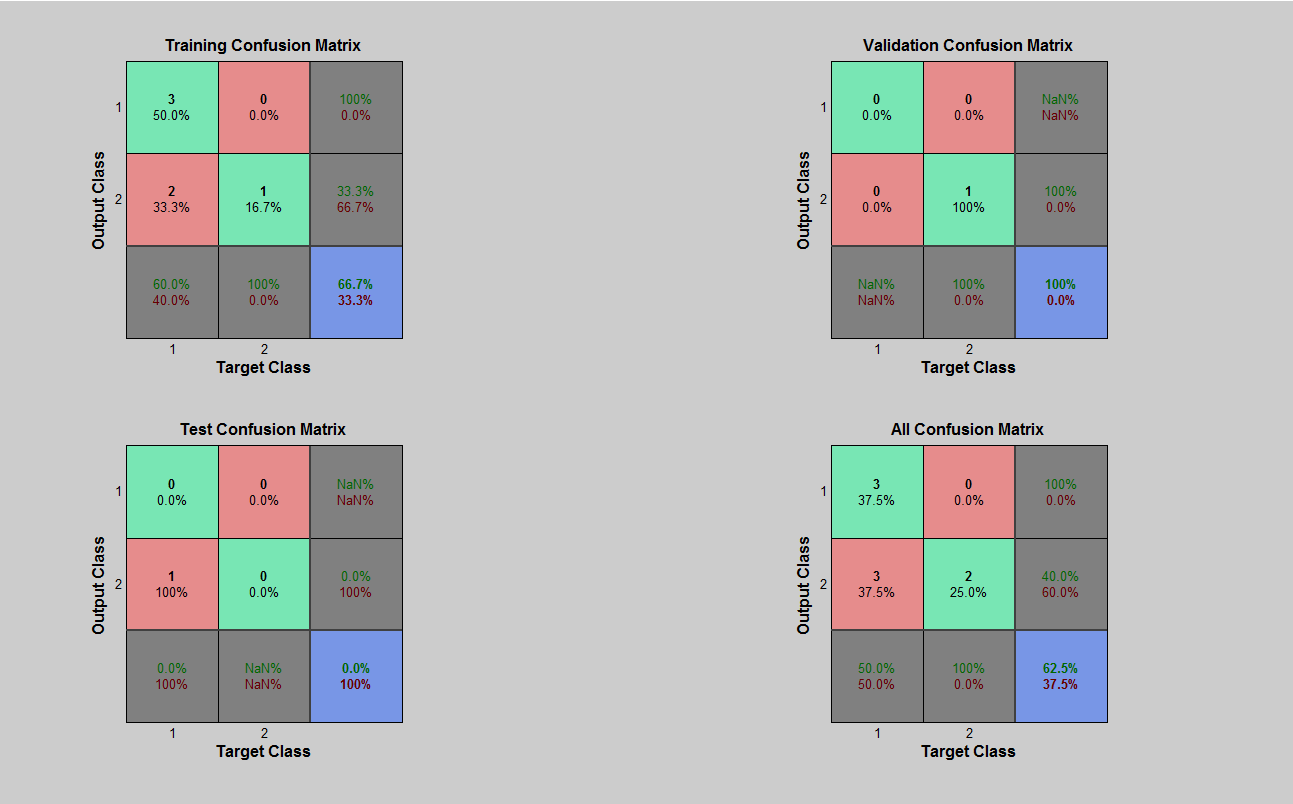
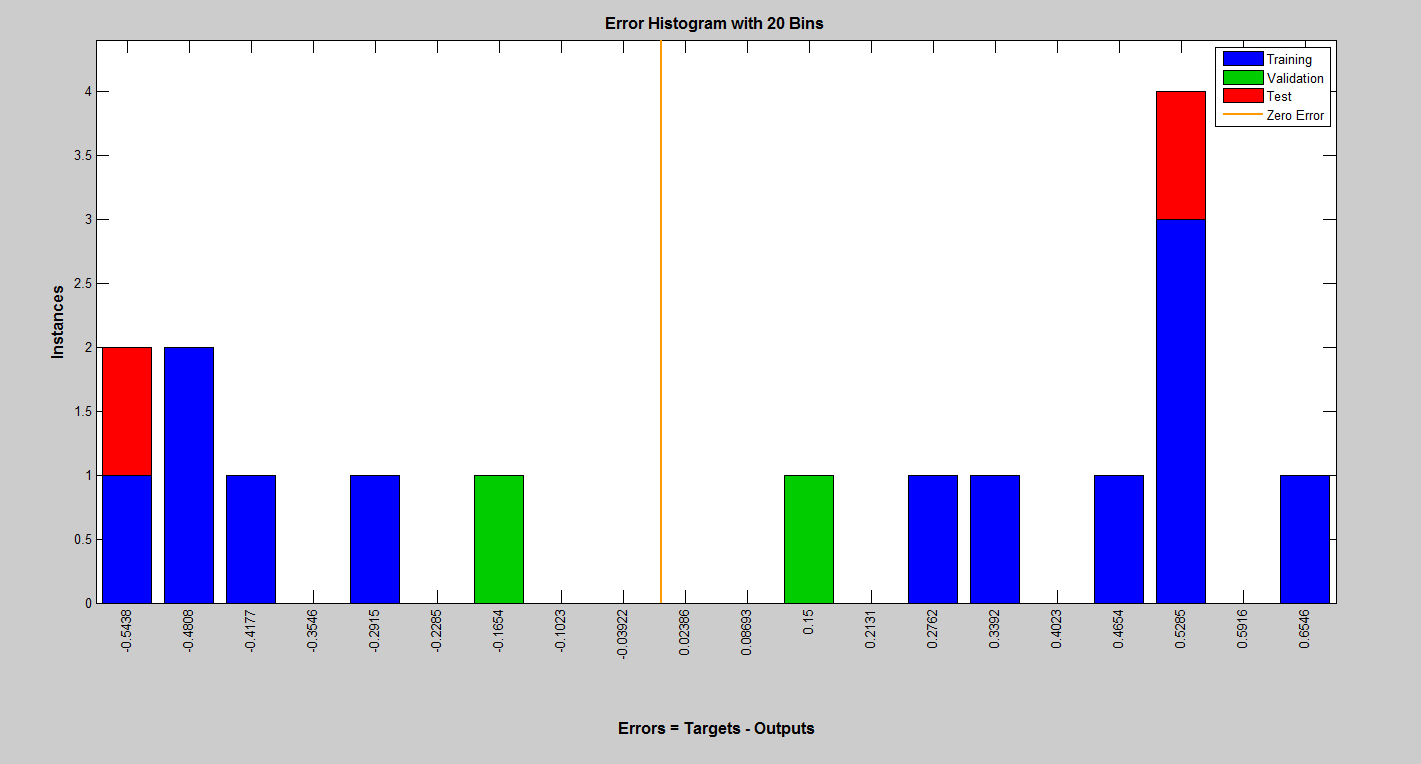
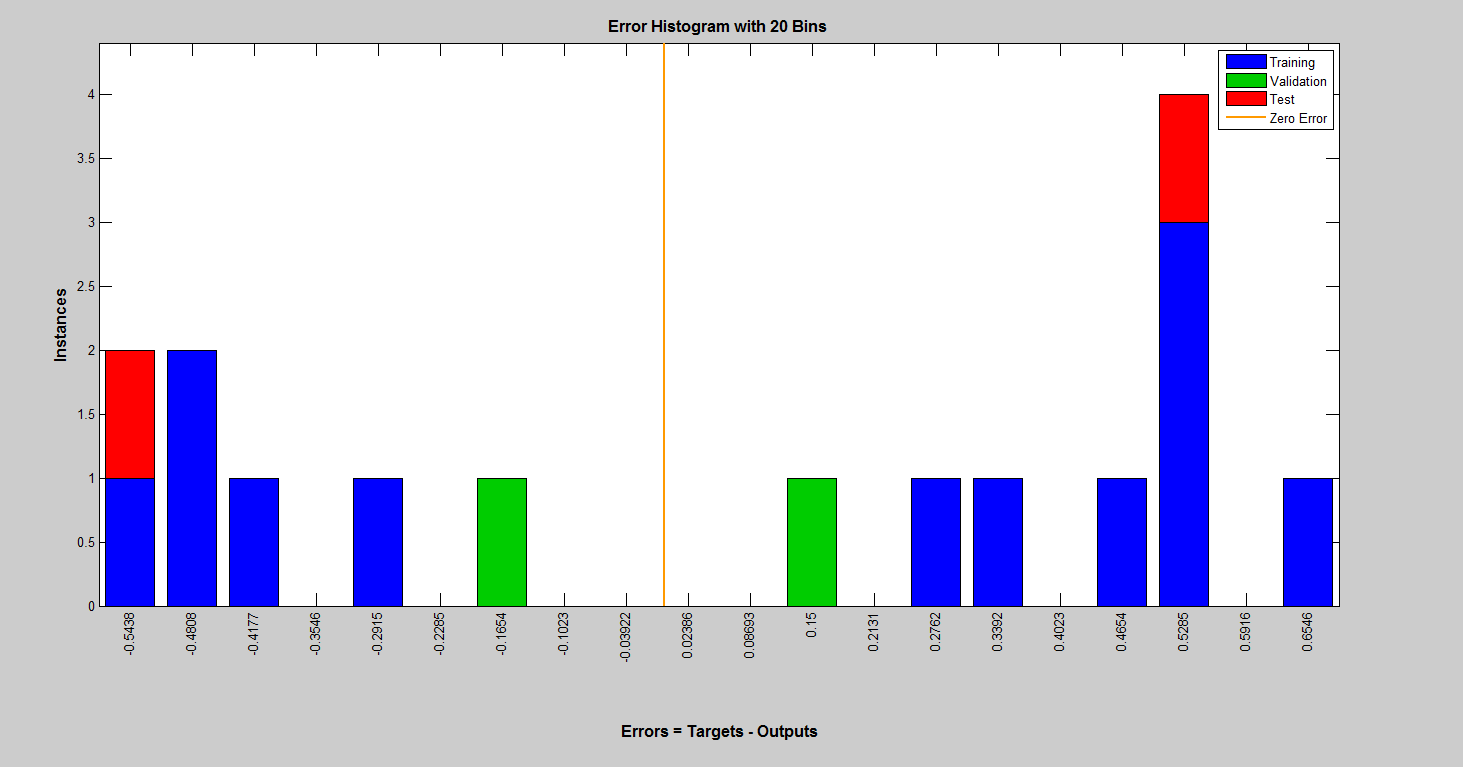
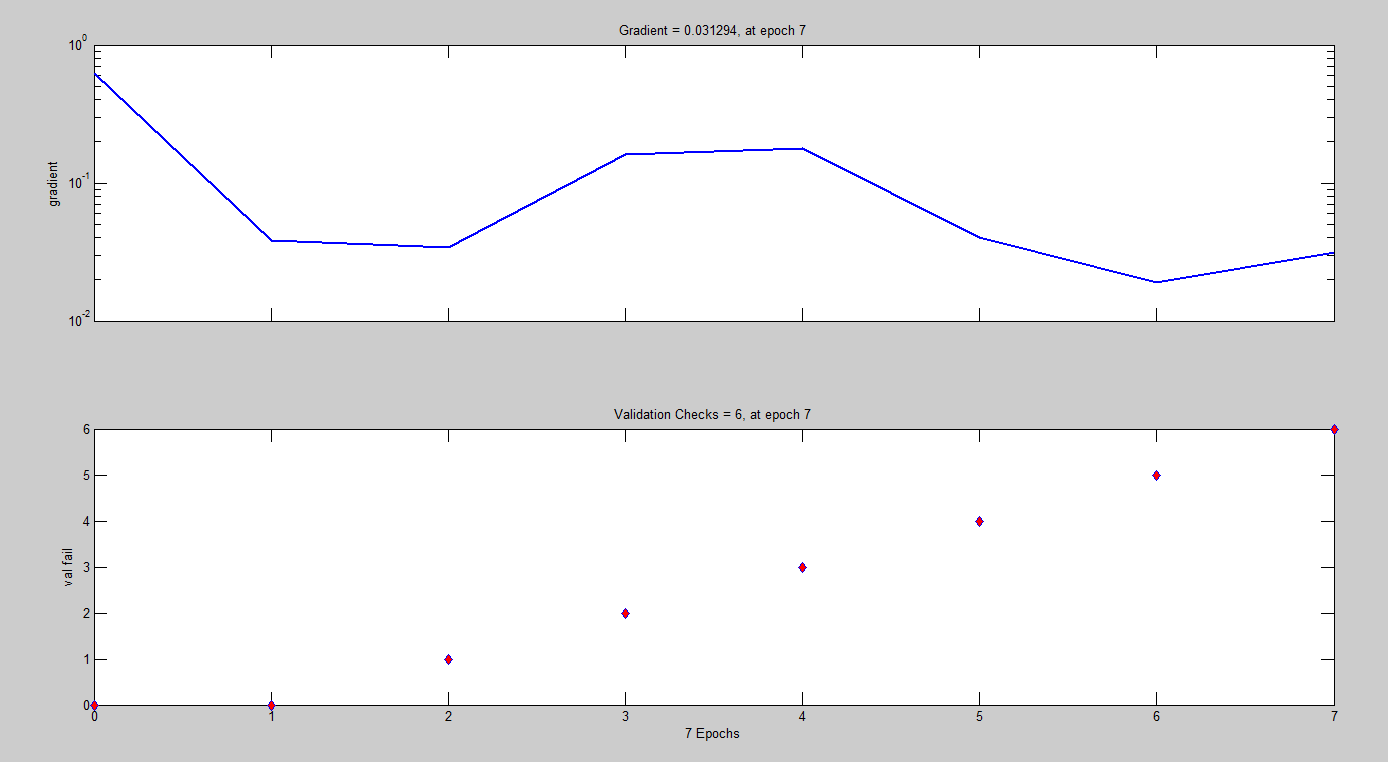
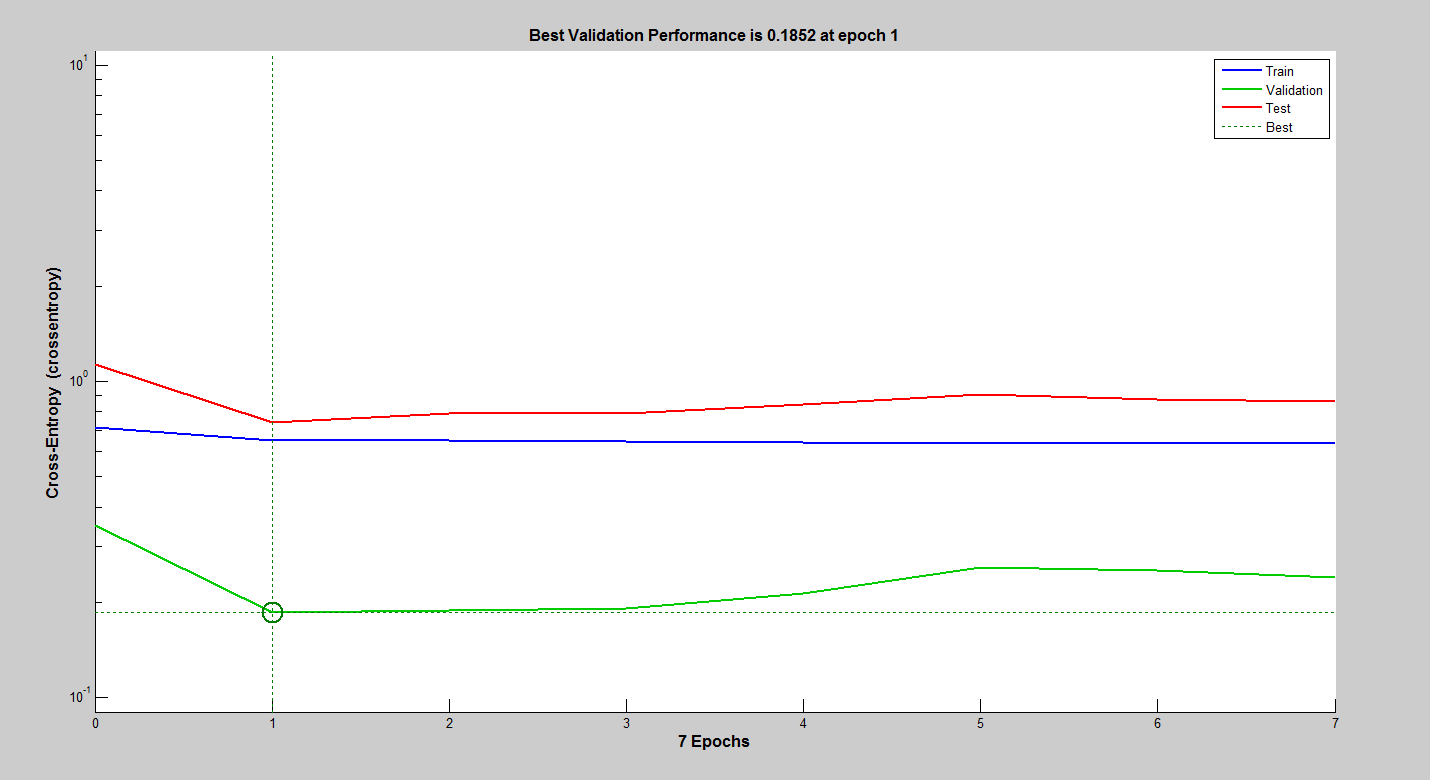
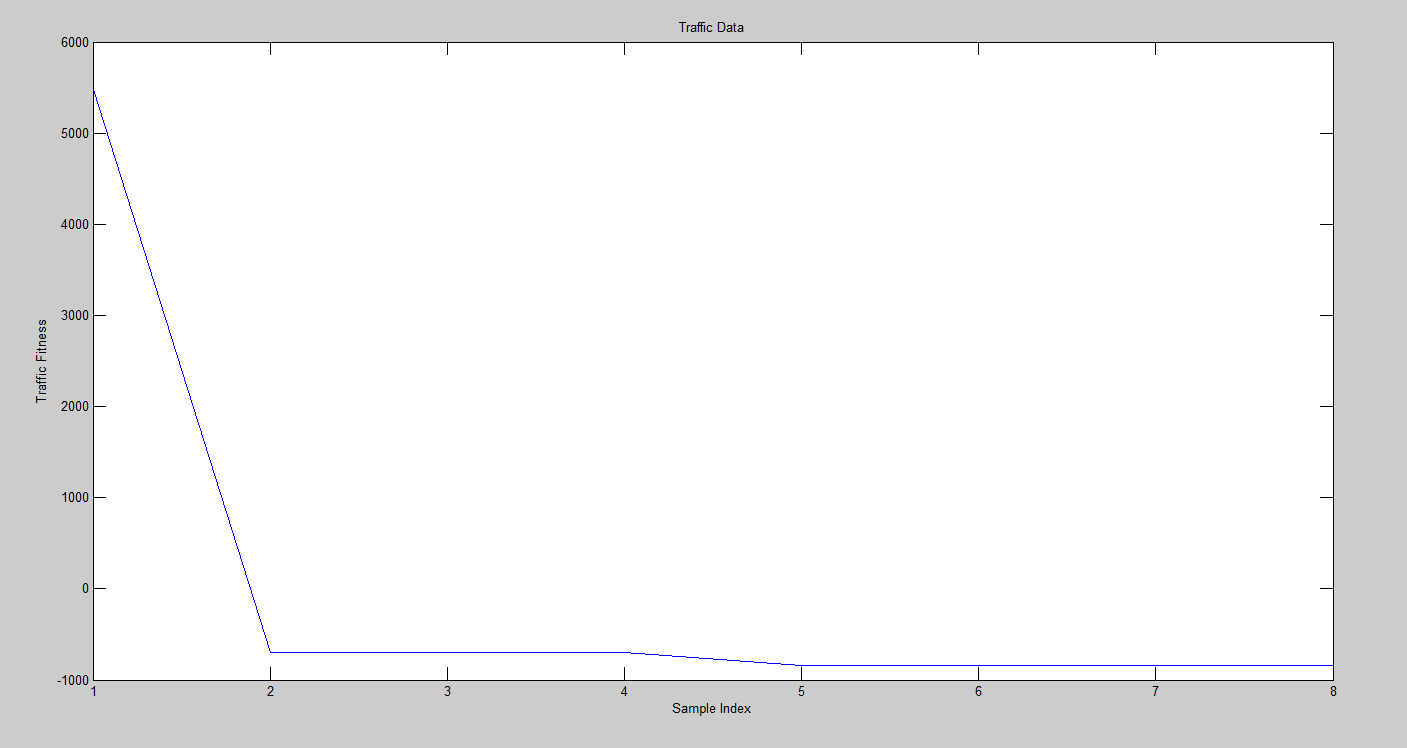
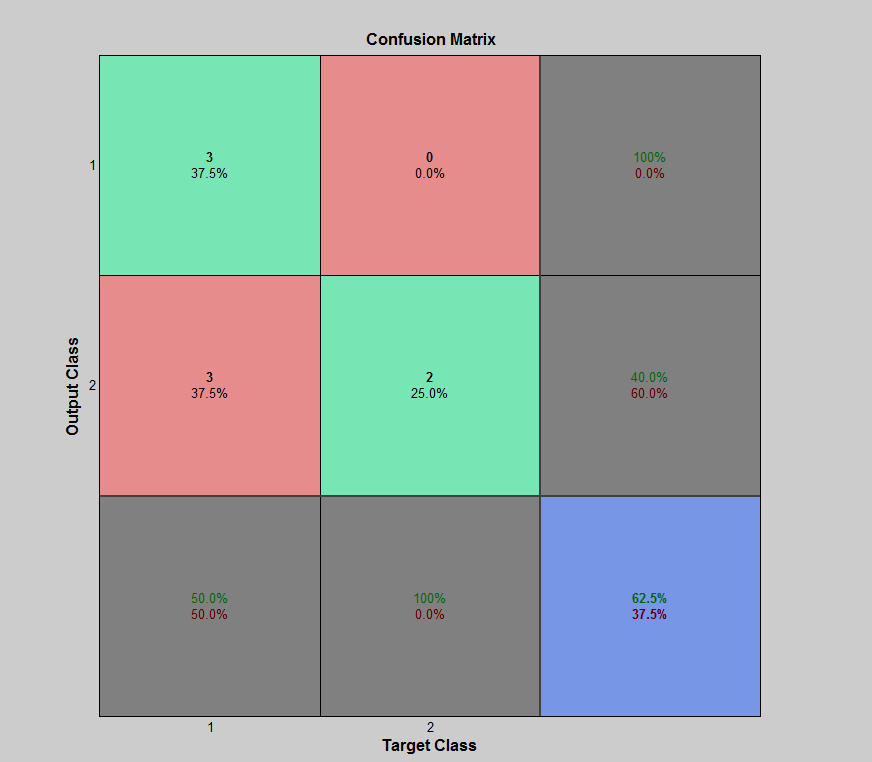
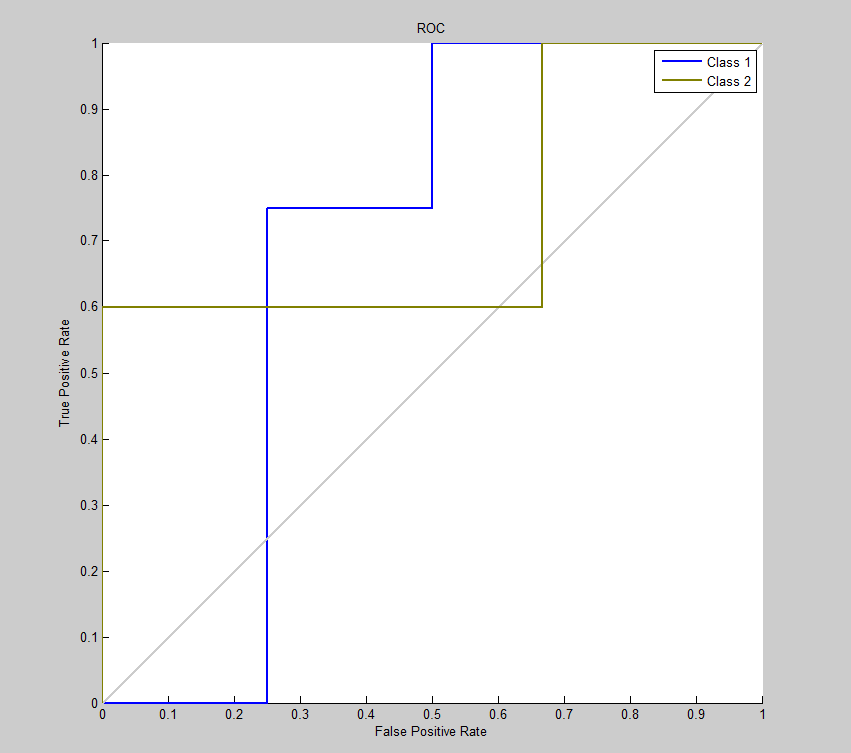
* + 1. **LDA**



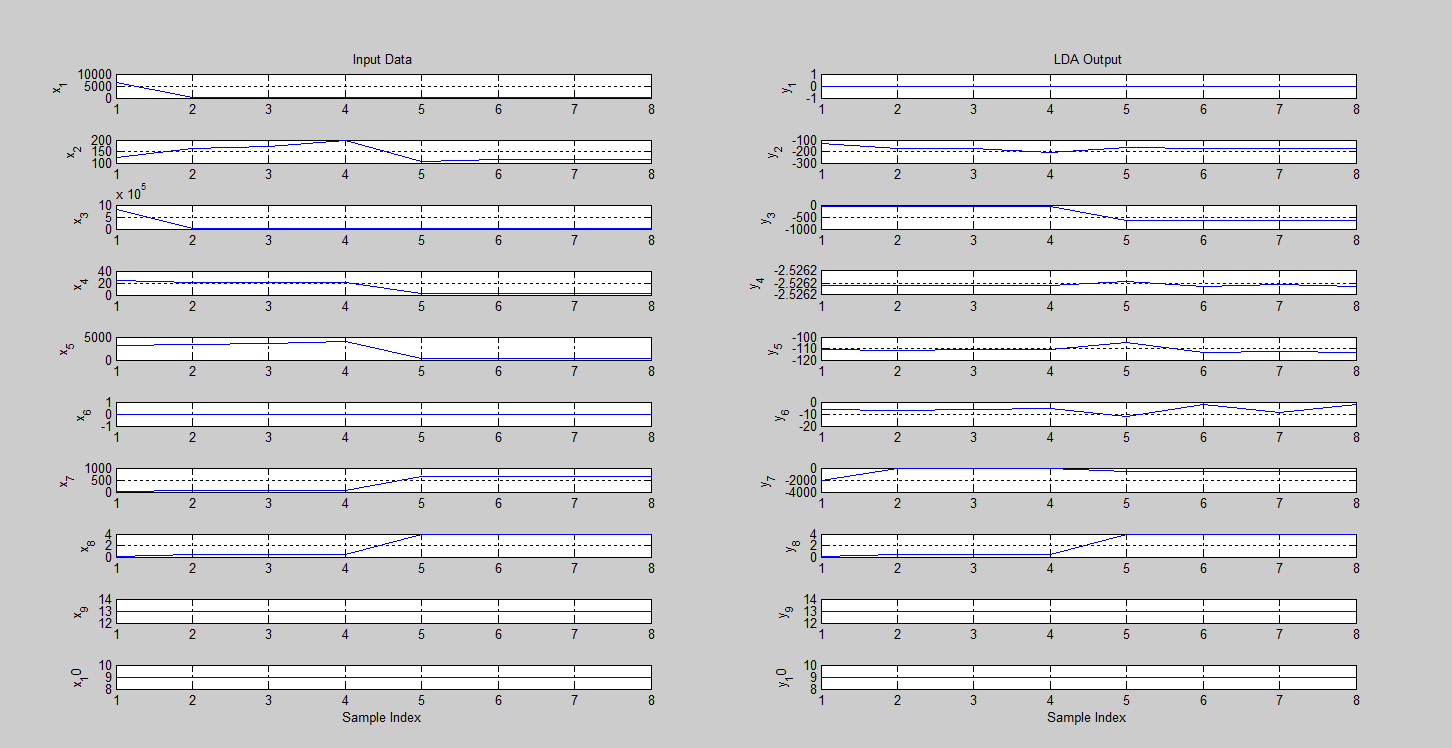
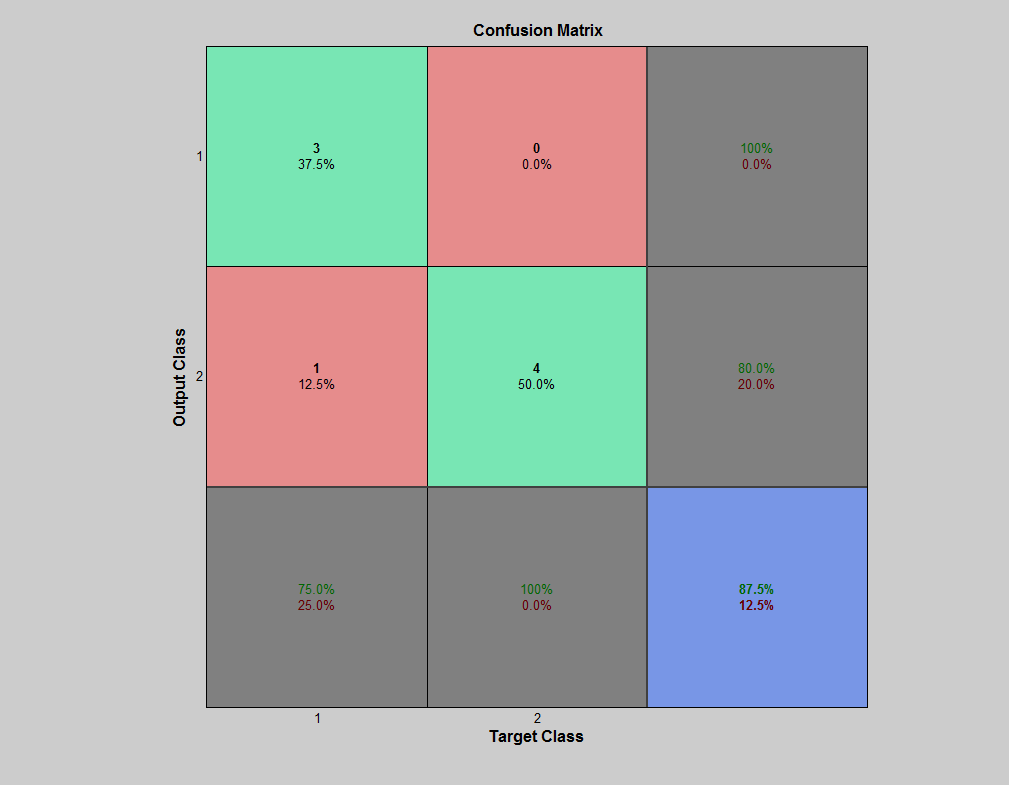
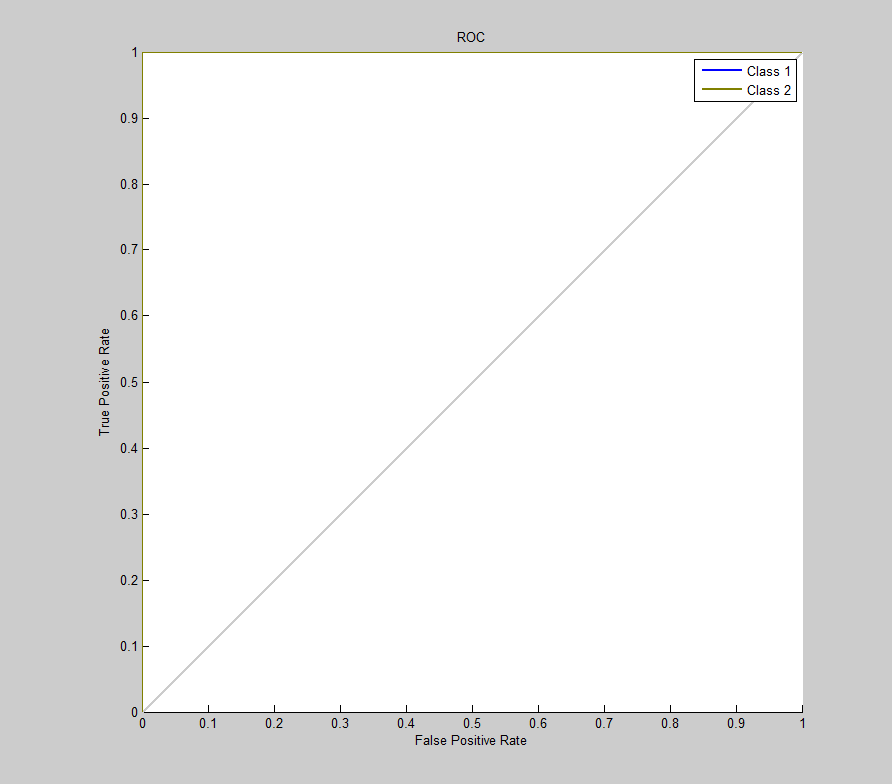
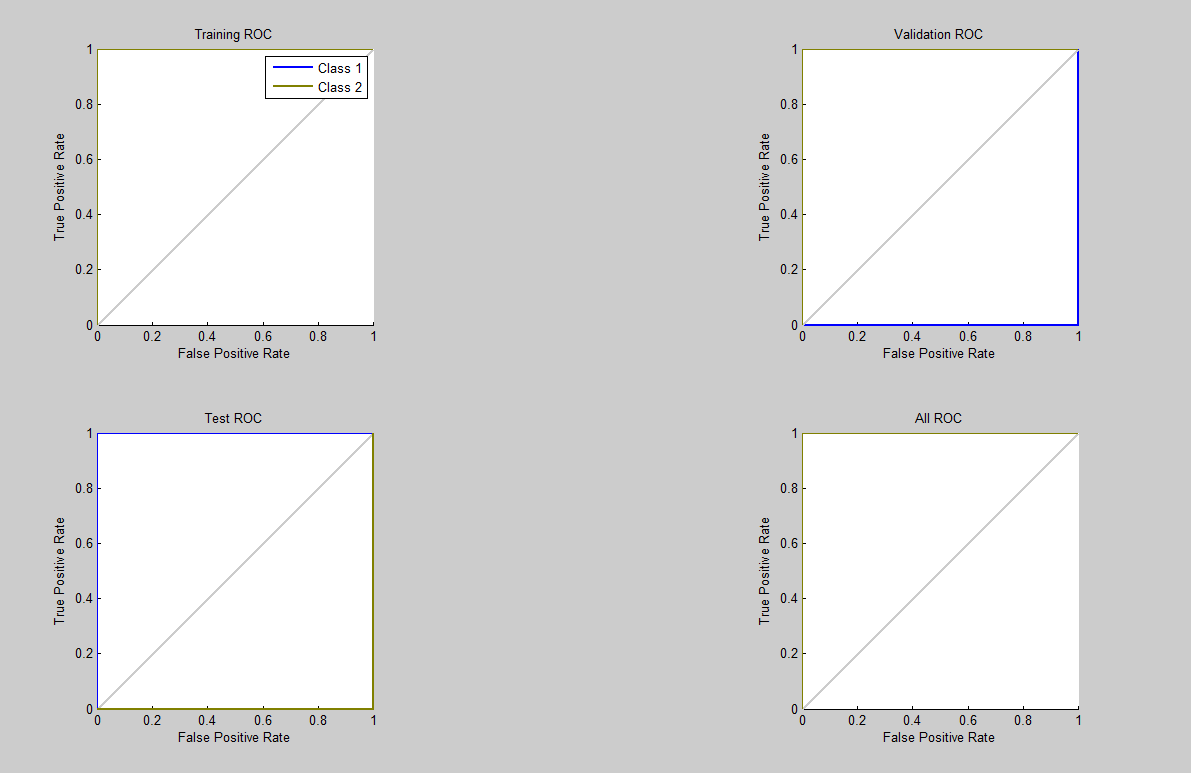
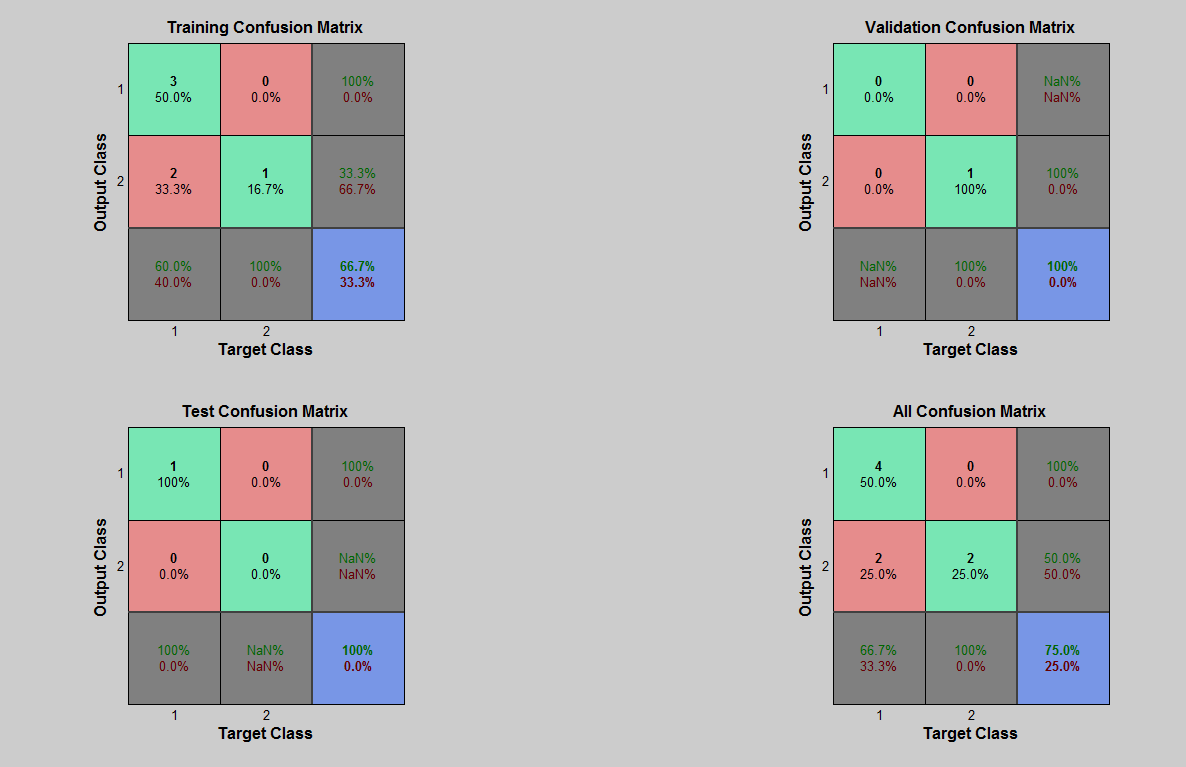
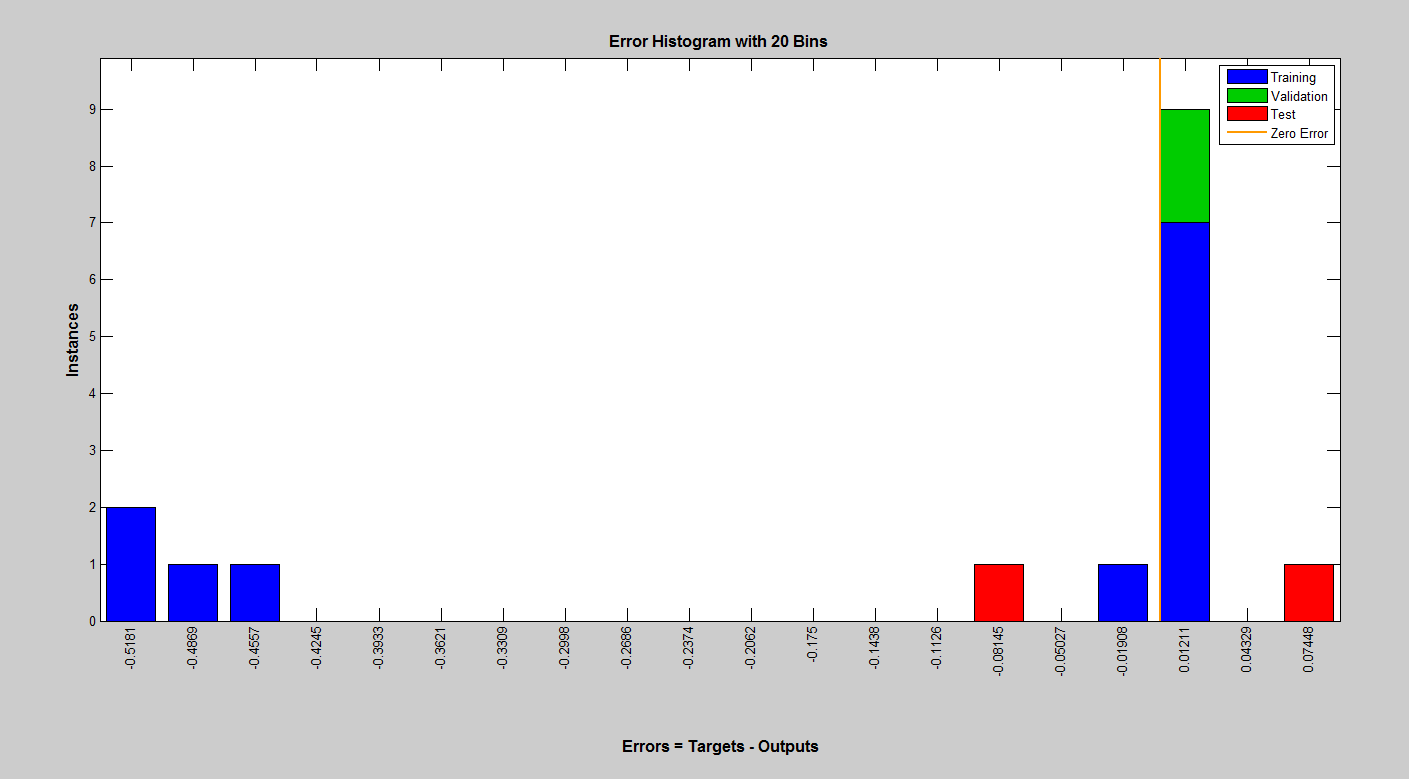
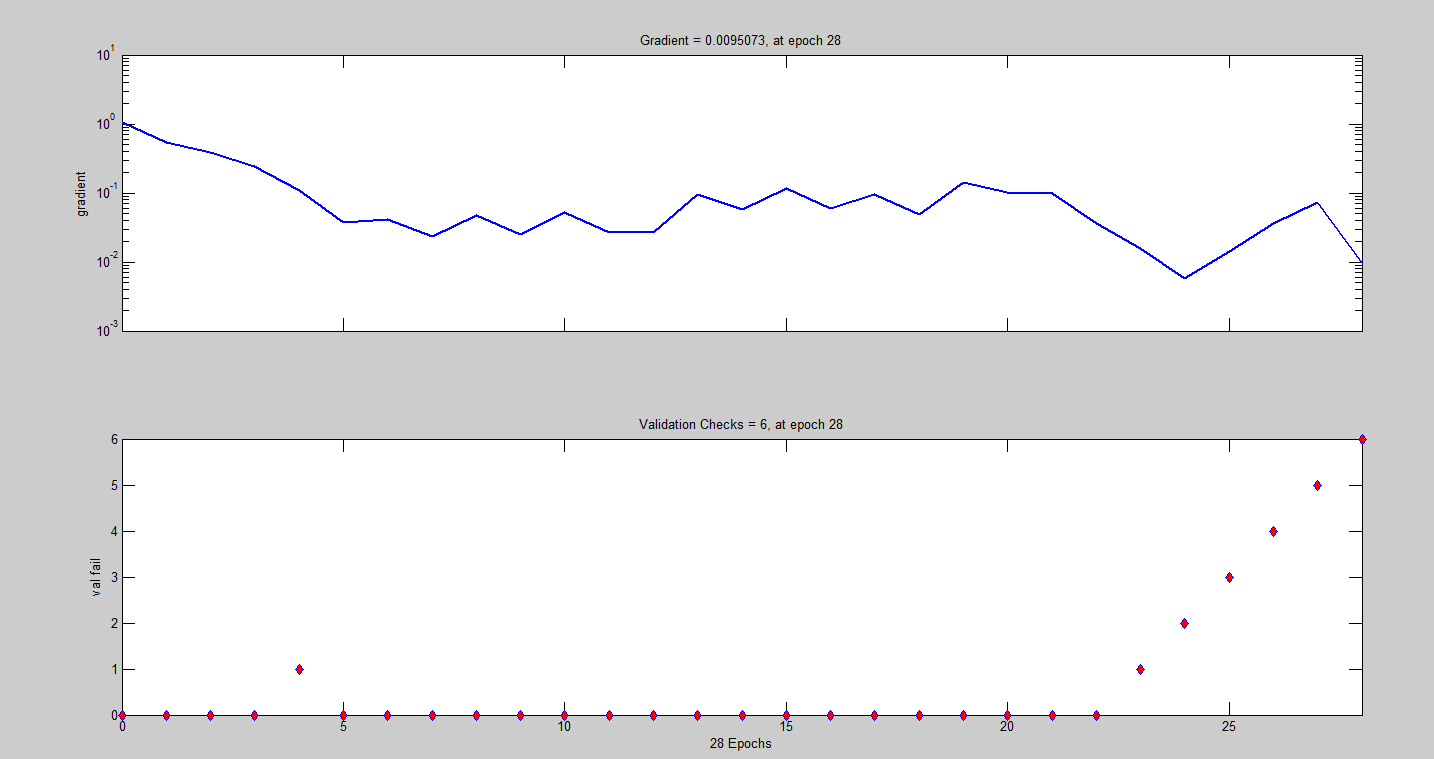
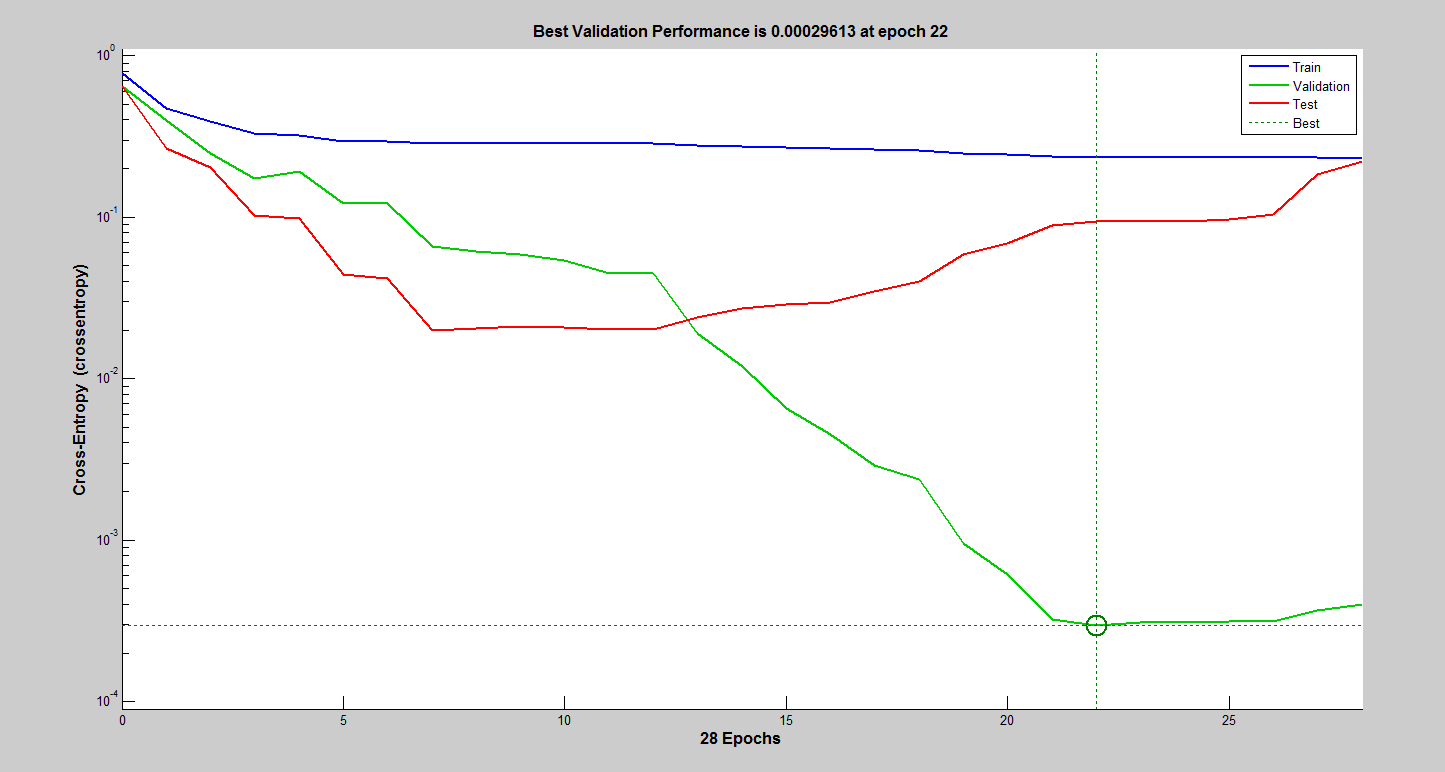
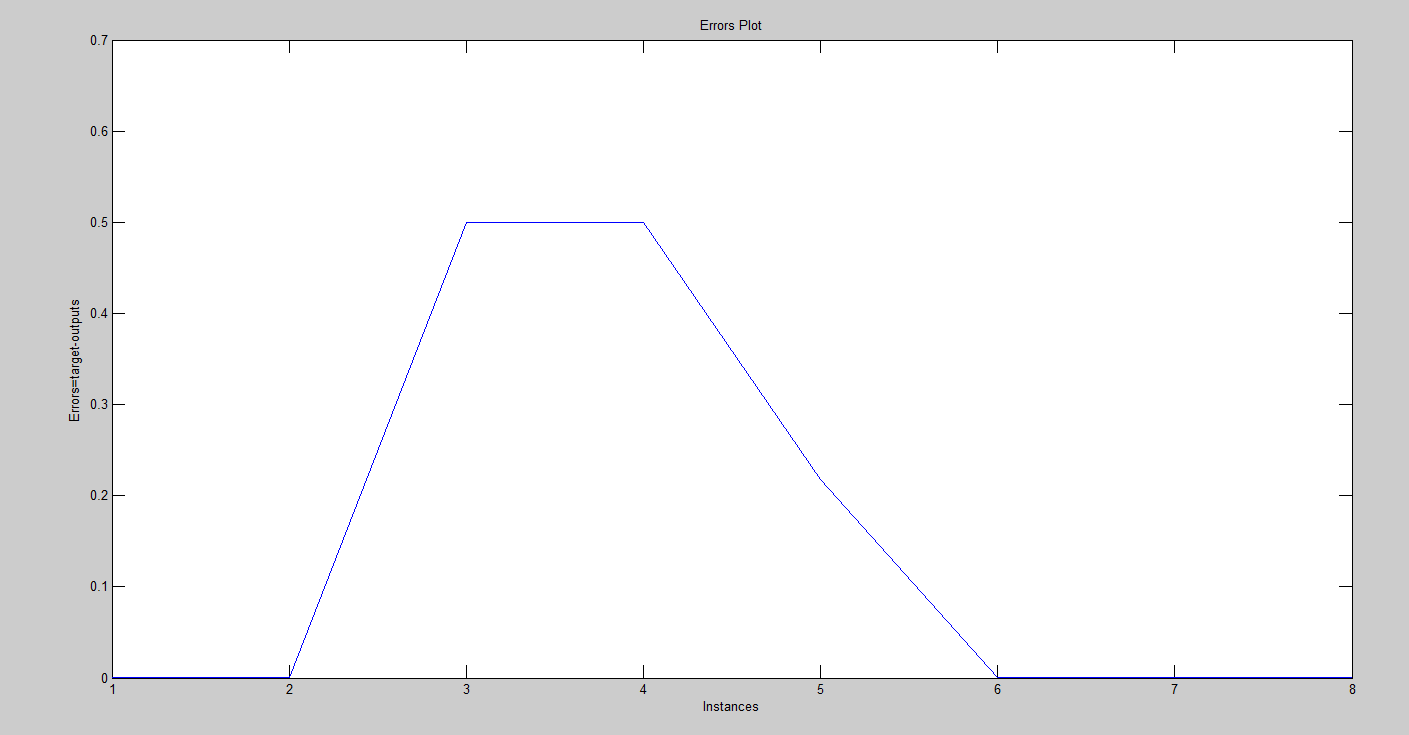
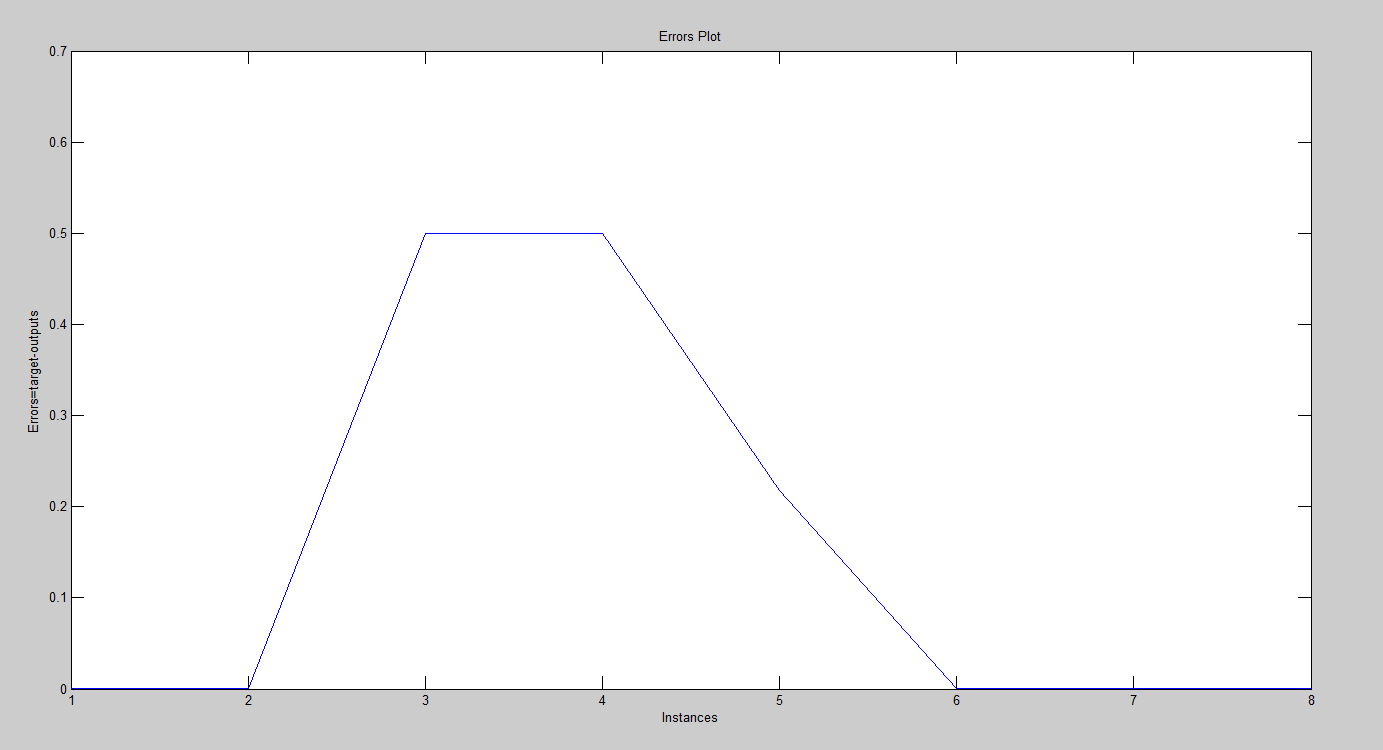
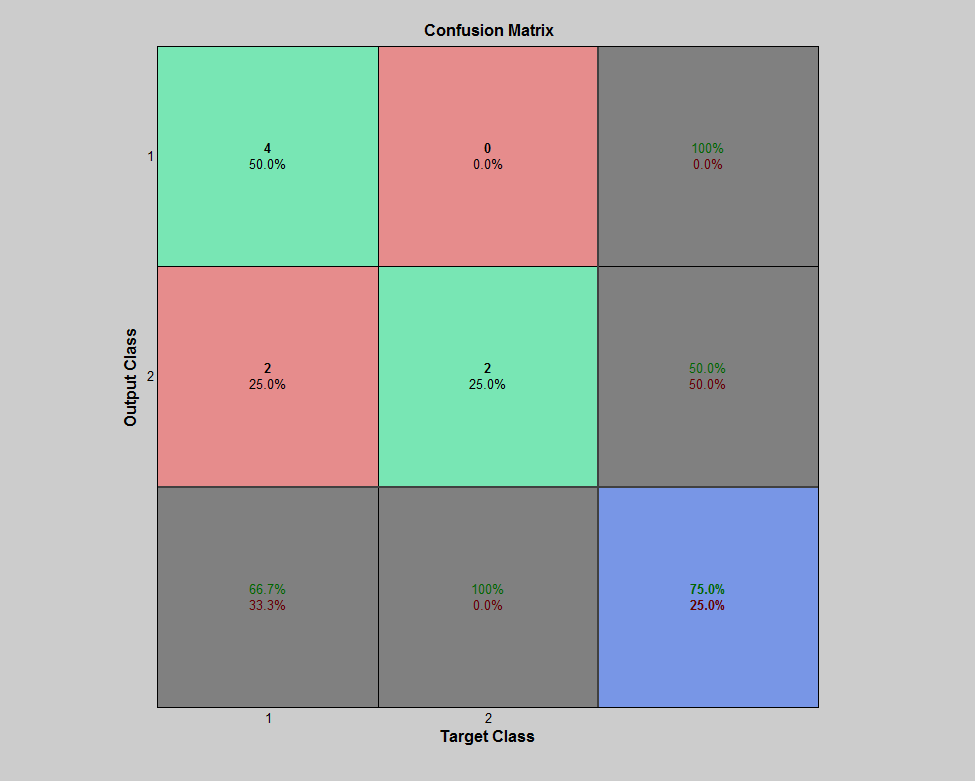
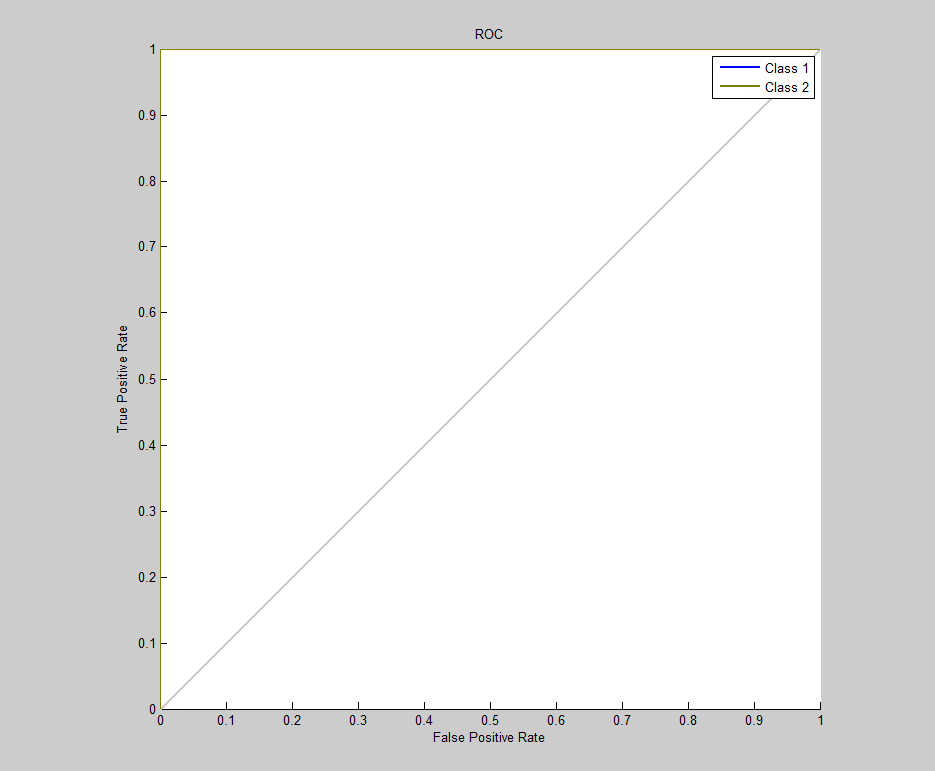
* + 1. **Proposed (PCA+LDA+ALO)**



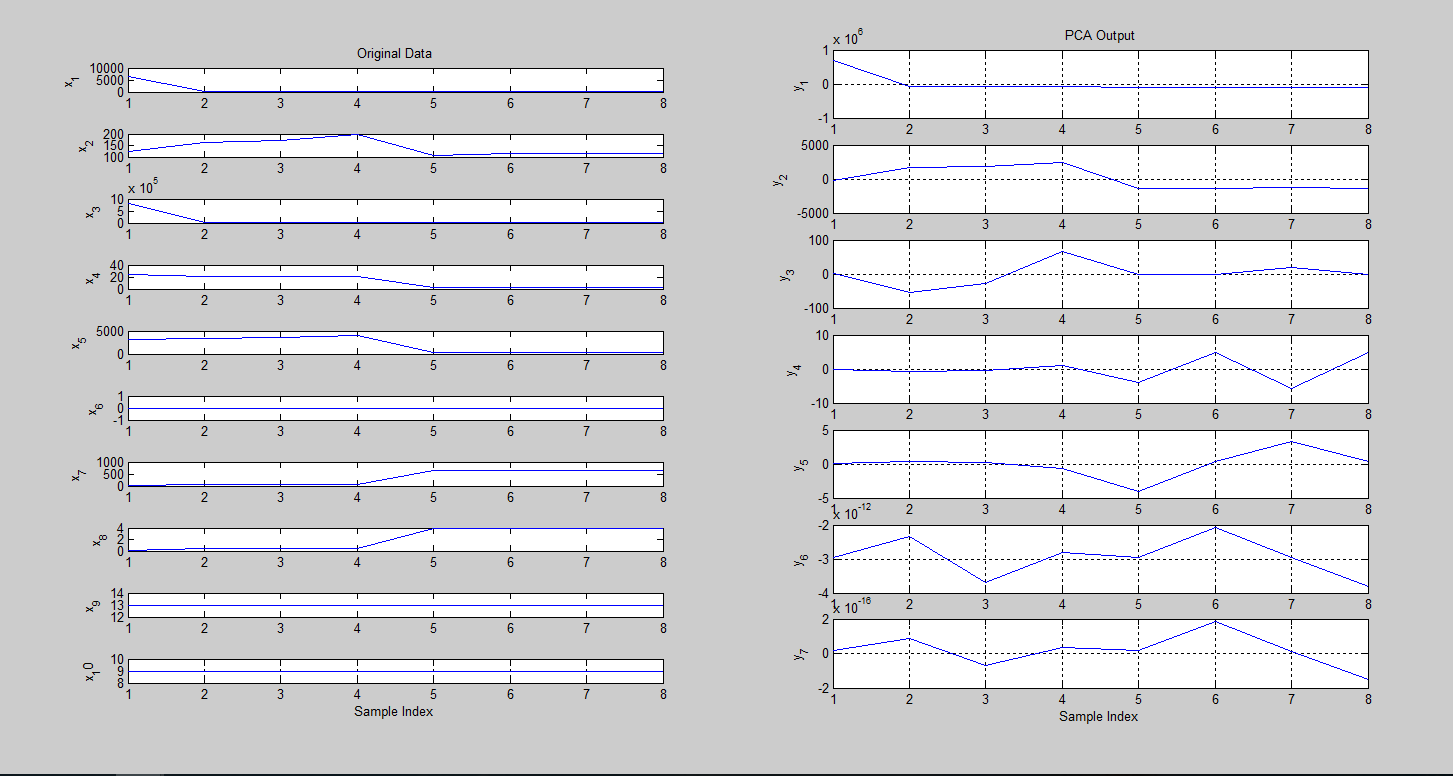
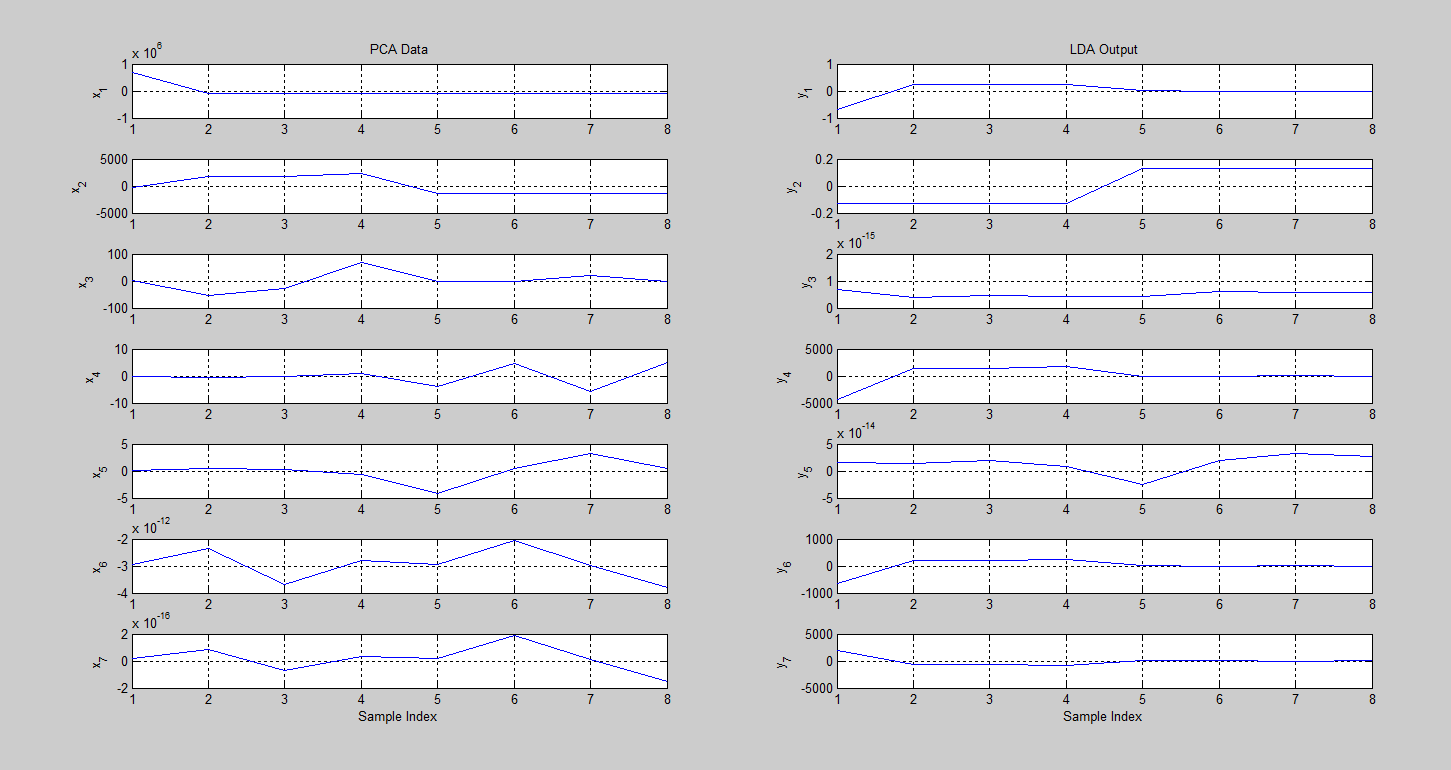
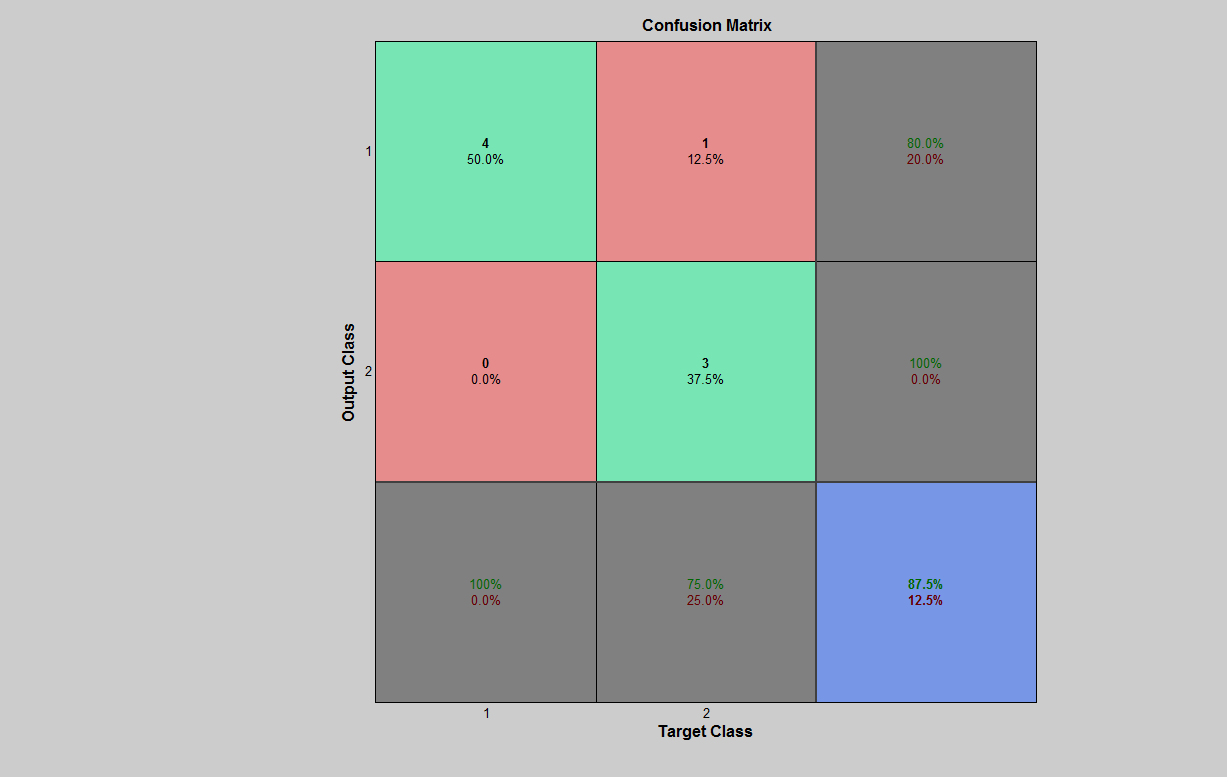
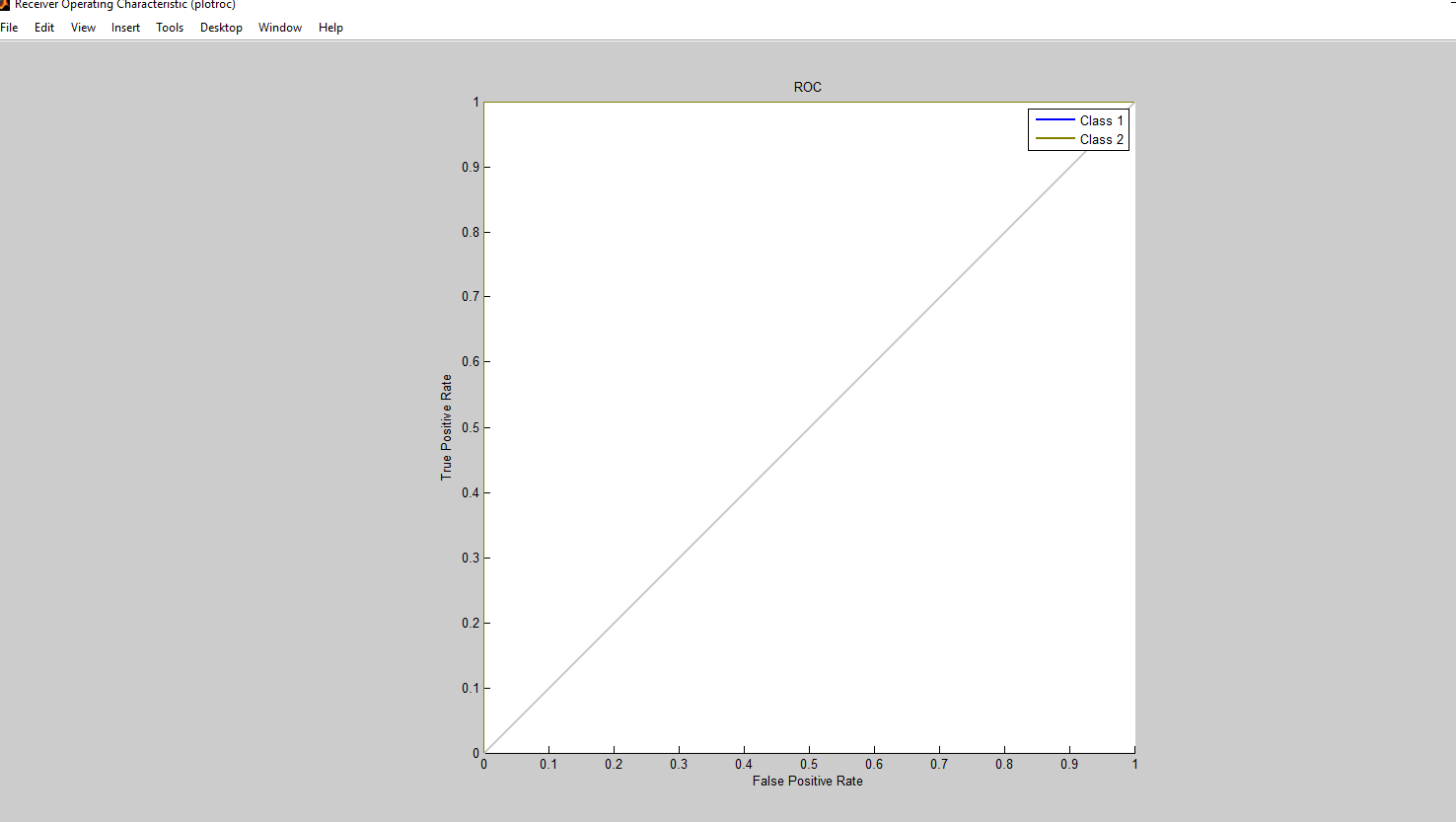
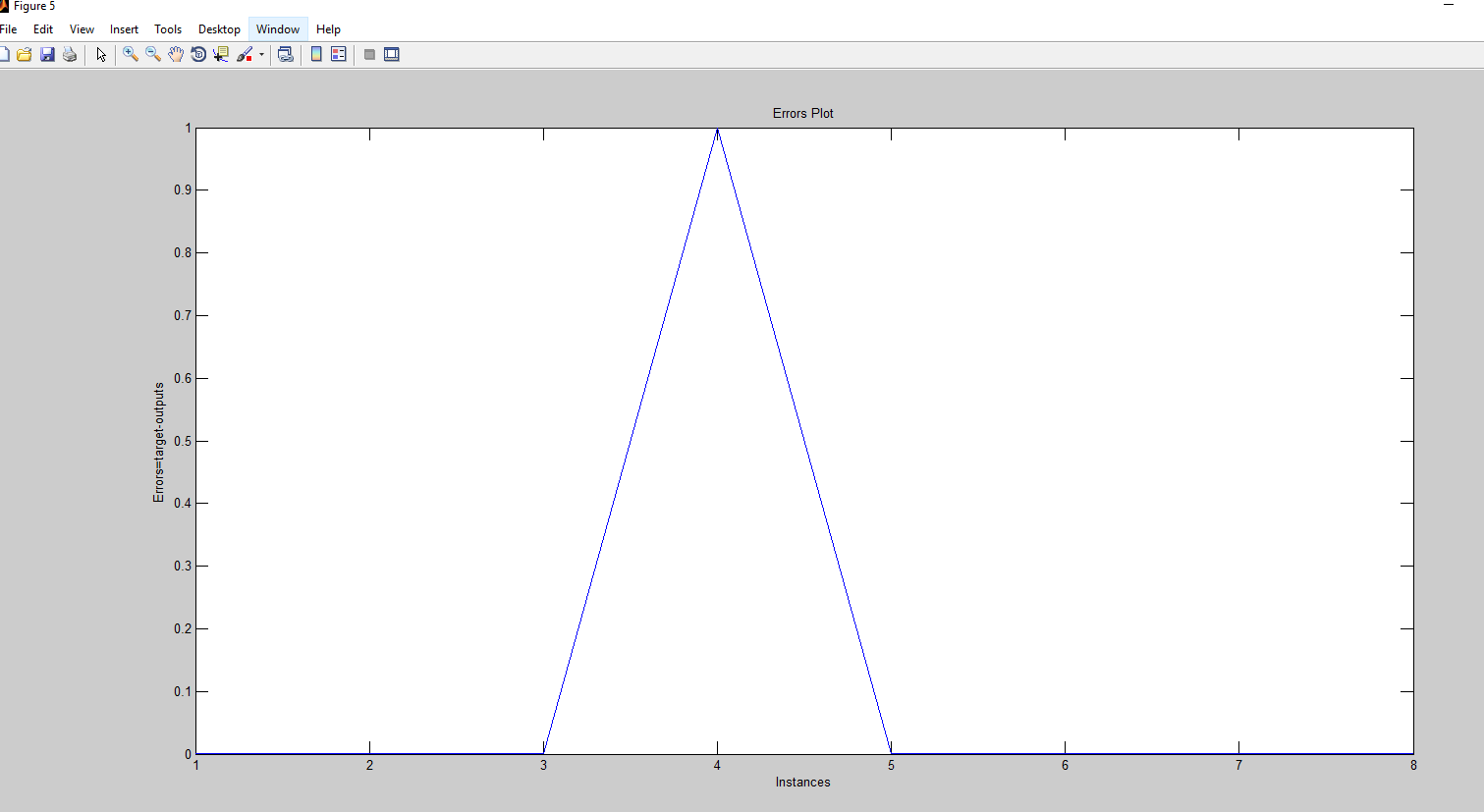
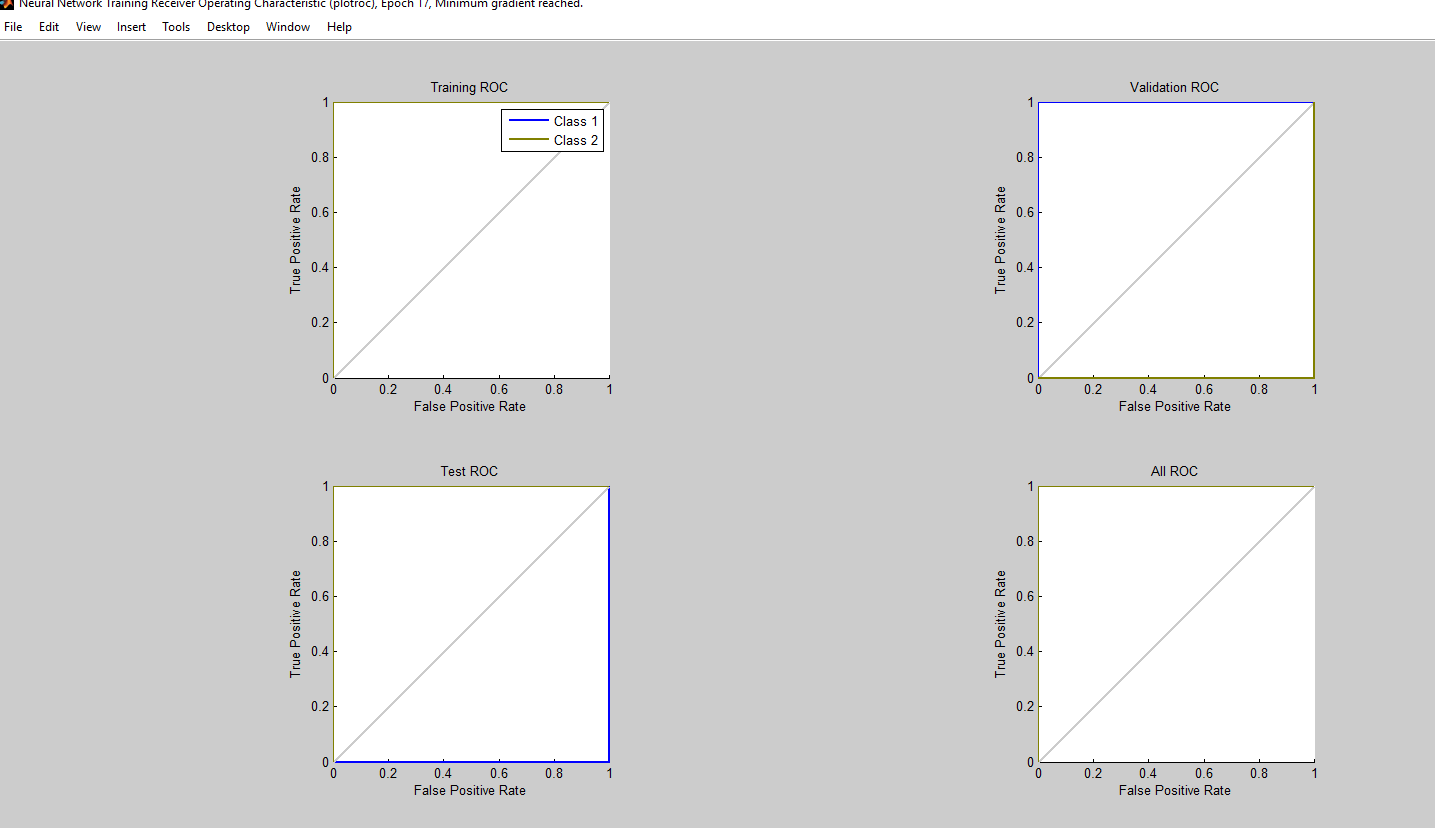
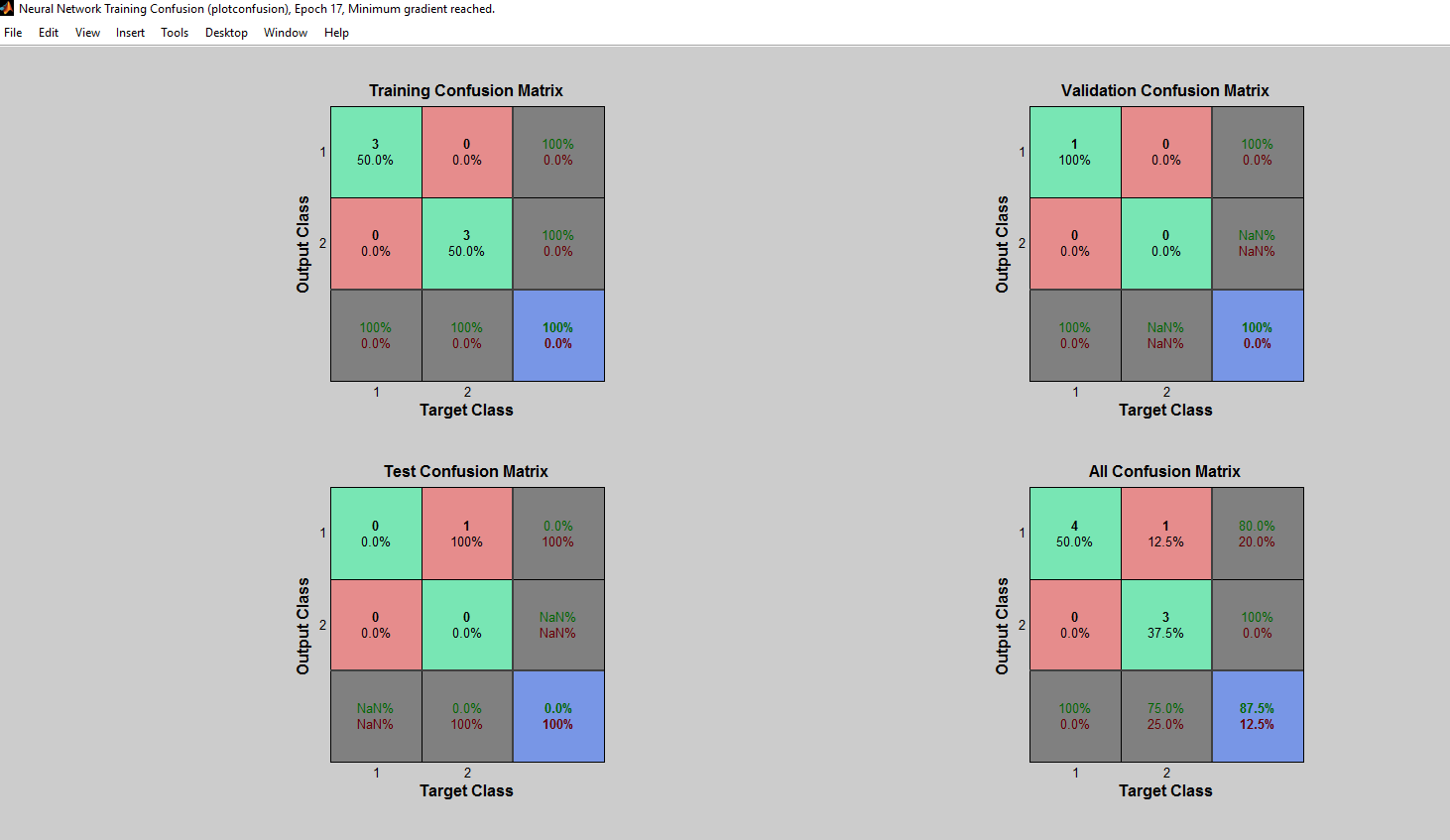
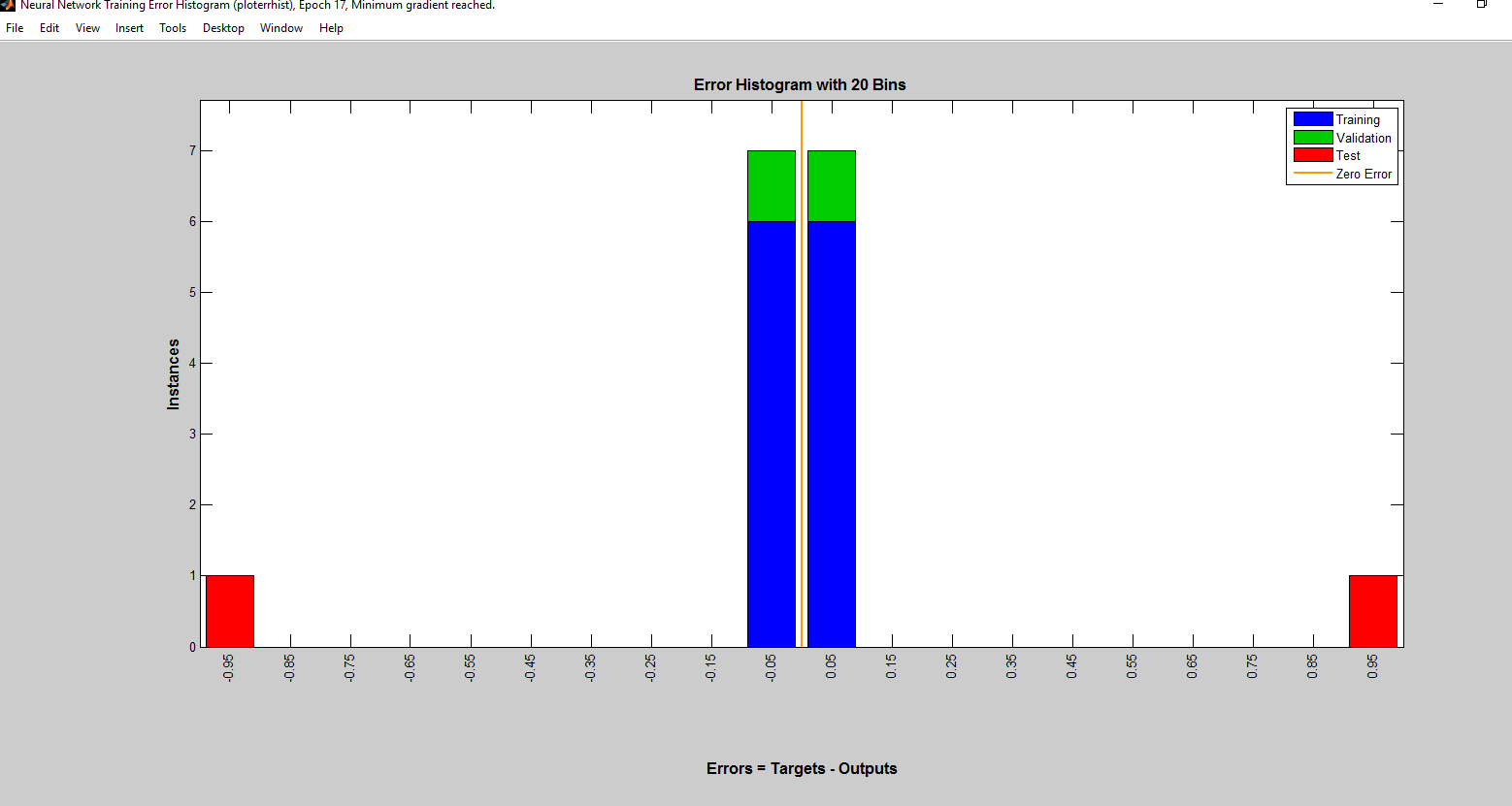
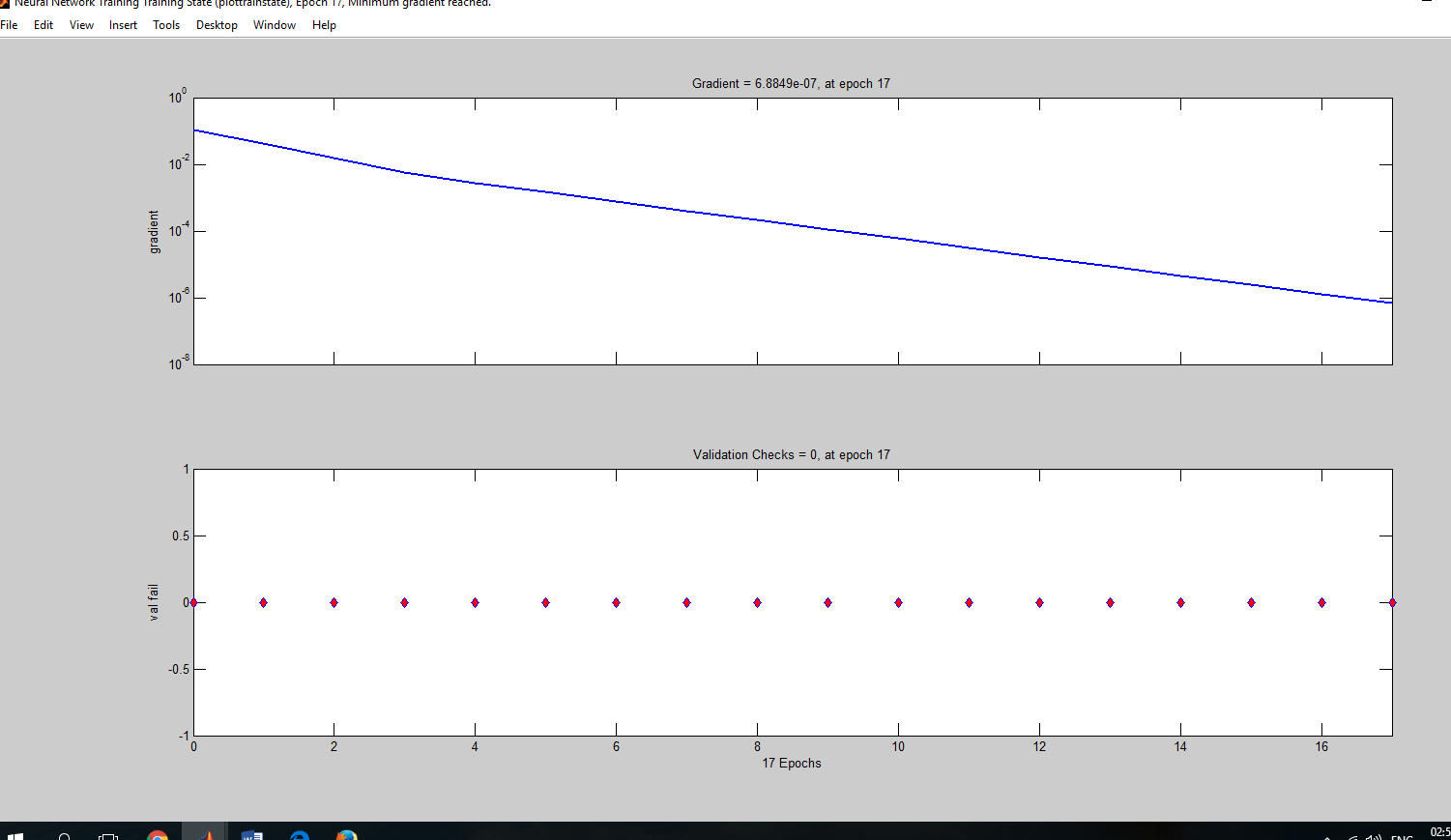
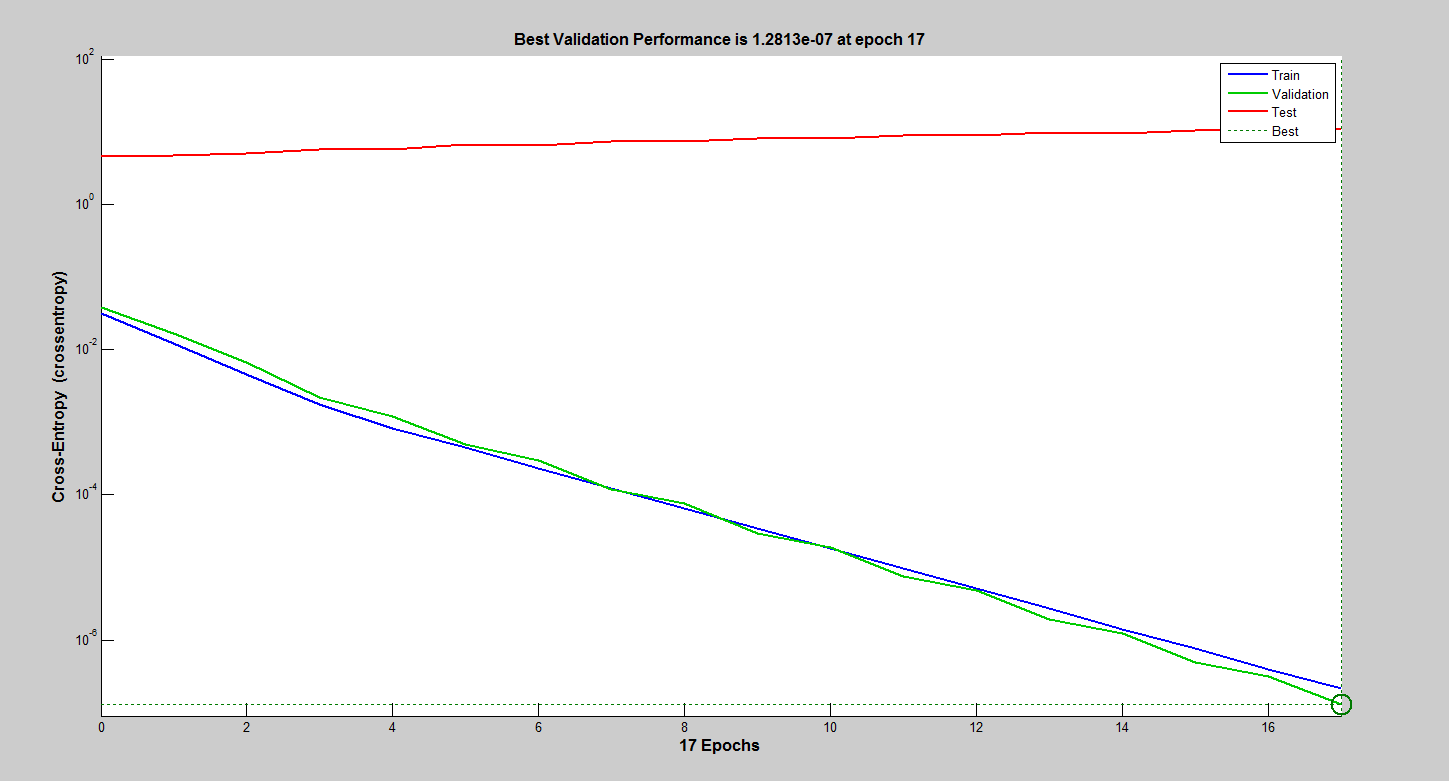
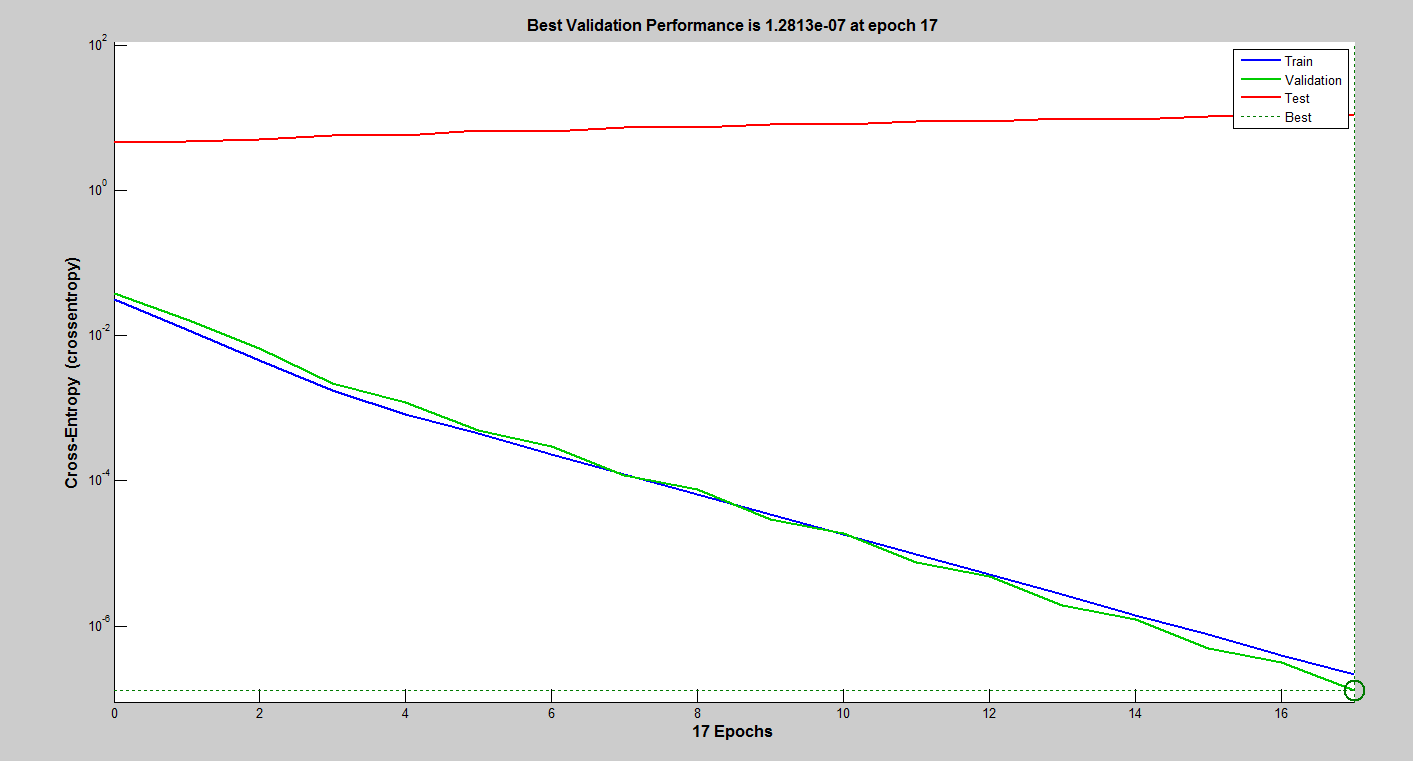
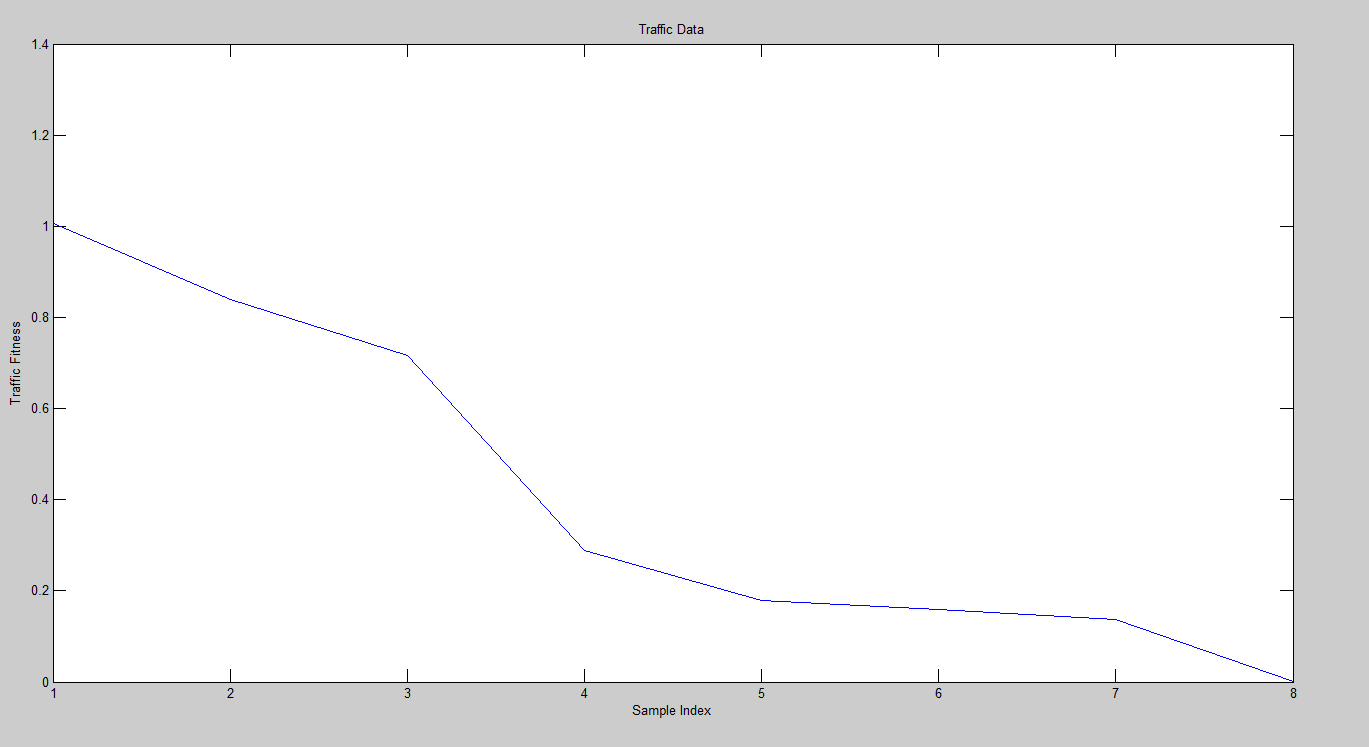
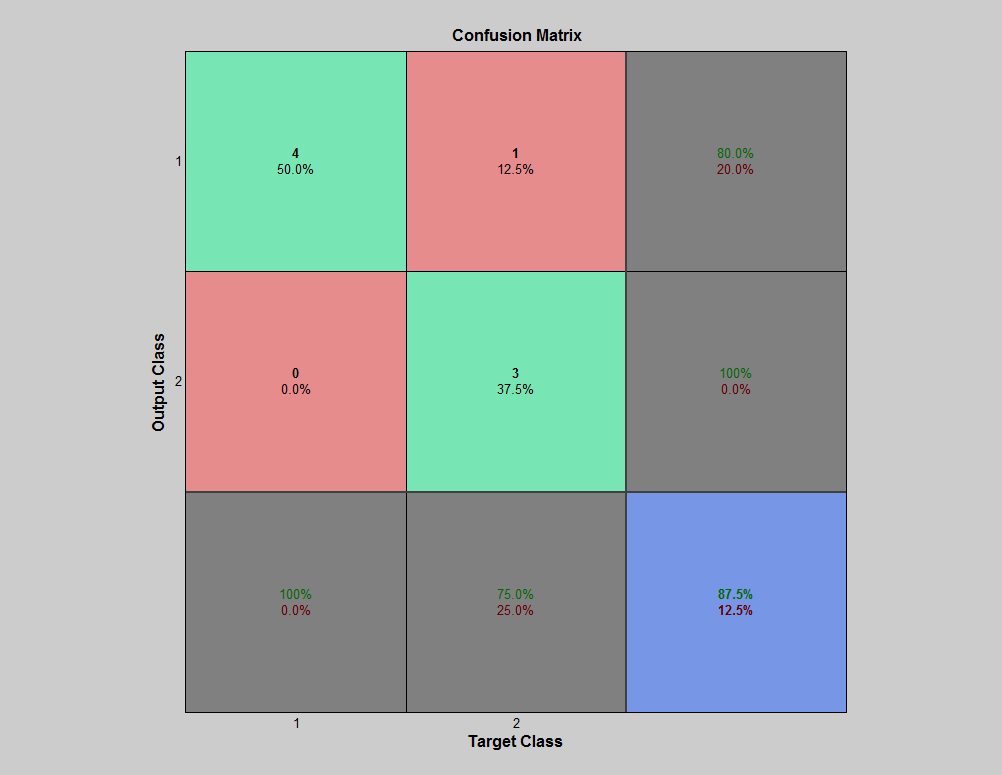
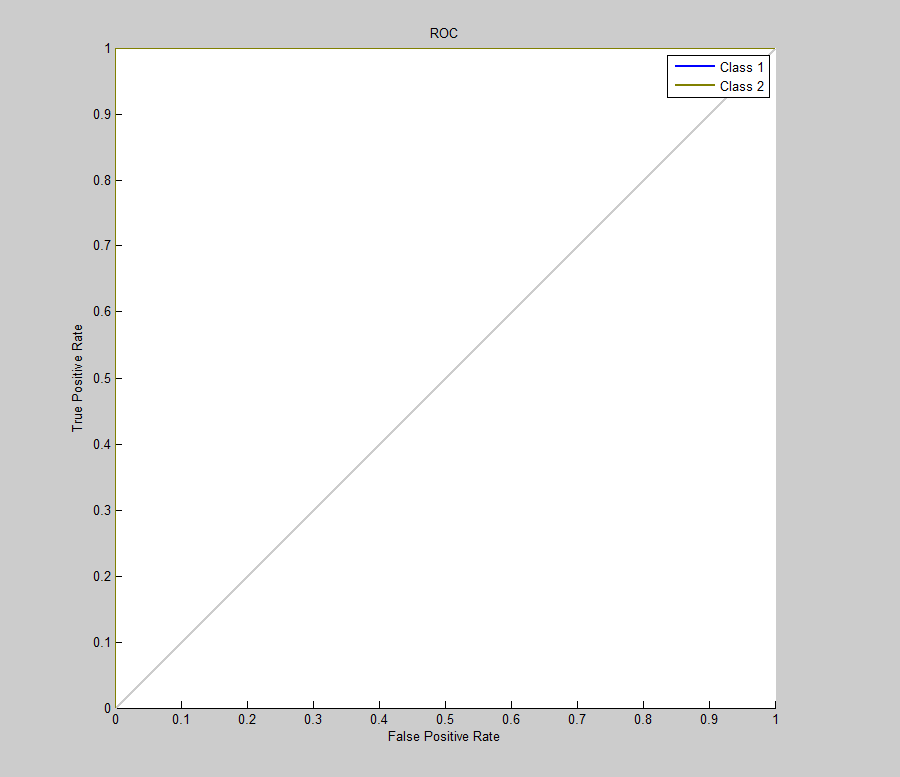
* 1. **TCP PSH ACK Attack cloud computing**
     1. **PCA**



* + 1. **LDA**



* + 1. **Proposed (PCA+LDA+ALO)**



**Result phase**

The incoming traffic profile (packets per second) of the CAIDA dataset has one-way traffic profile clearly shows two rather distinct traffic rates, as experienced by the target victim. The first one is the packet rate during the first-half of the trace i.e. from the start of the trace until around the 1500second mark. For simulation purposes, the packet rate during this period is referred to as the normal packet rate. The average packet rate for this period was approximately 385 packets per second in the original dataset. The second distinct traffic rate is seen during the second half of the trace i.e., starting from approximately 1500 seconds until the end of the trace. During this period, the packet rate is referred to as the attack packet rate, and contains approximately125,705 packets per second on average. In order to simulate network traffic similar to the CAIDA dataset, the normal and the attack packet rates, calculated from the original dataset, are provided to the attack configuration file of the MATLAB to generate a 5 minute sample network trace.

IDPS DDoS attack preventing model

**Evaluation and comparison phase**

**Reference**

**[1]** K.A. Spackman, Signal detection theory: Valuable tools for evaluating inductive learning. In Proceedings of the Sixth International Workshop on Machine Learning, San Francisco, CA, USA, 1989, pp. 160–163. Morgan Kaufmann Publishers.

[2] T. Fawcett, An introduction to ROC analysis, Pattern Recognition Letters, special issue on ROC analysis 27(8) (2006), 861–874

[3] D. Bamber, The area above the ordinal dominance graph and the area below the receiver operating characteristic graph, Journal of Mathematical Psychology 12(4) (1975), 387–415.

[4] L. Breiman, J. Friedman, R. Olshen and C. Stone, Classification and regression trees. Wadsworth International Group, Belmont, CA, USA, 1984.

[5] J.A. Hanley and B.J. McNeil, The meaning and use of the area under a receiver operating characteristic (ROC) curve, Radiology 143(1) (1982), 29–36.