DATA MINING TECHNIQUES IN EMBOLI DETECTION

TÜRKALP KUCUR

JANUARY 2007

DATA MINING TECHNIQUES IN EMBOLI DETECTION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL
OF
THE UNIVERSITY of BAHCESEHIR
BY
TÜRKALP KUCUR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF COMPUTER ENGINEERING

JANUARY 2007

Approval of the INSTITUTE OF SCIENCE

Assoc.Prof.Dr. Irini Dimitriyadis

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Prof.Dr. Nizamettin Aydın	Asst.Prof.Dr. Adem Karahoca
Co-Supervisor	Supervisor

Examining Committee Members

Nizamettin Aydın ______________________

Adem Karahoca ______________________

Orhan Gökçöl ______________________

ABSTRACT

DATA MINING TECHNIQUES IN EMBOLI DETECTION

Kucur, Türkalp

M.S. Department of Computer Engineering

Supervisor: Asst.Prof.Dr. Adem Karahoca

Co-Supervisor: Prof.Dr. Nizamettin Aydın

JANUARY 2007, 77 Pages

Asymptomatic circulating cerebral emboli, which are particles bigger than blood cells, can be detected by transcranial Doppler ultrasound. In certain conditions asymptomatic embolic signals (ES) appear to be markers of increased stroke risk. A major problem with clinical implementation of the technique is the lack of a reliable automated system of ES detection. Recordings in patients may need to be hours in duration and analyzing the spectra visually is time consuming and subject to observer fatigue and error. ES, reflected by an embolus, has some distinctive characteristics. They have usually larger amplitude than the signals from normal blood flow (Doppler speckle) and show a transient characteristic. They are finite oscillating signals and resemble wavelets. Unlike many artifacts such as caused by probe movement or speech, ES are unidirectional and usually contained within the flow spectrum. A number of methods to detect cerebral emboli have been studied in the literature. In this study, data mining techniques have been used in order to increase sensitivity and specificity of an embolic signal detection system previously described by Aydin, et all, 2004.

Keywords: emboli, detection, data mining

ÖZET

EMBOLİ TESBİTİNDE VERİ MADENCİLİĞİ YÖNTEMLERİNİN KULLANIMI

Kucur, Türkalp

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Yrd.Doç.Dr. Adem Karahoca

Tez Yöneticisi: Prof.Dr. Nizamettin Aydın

JANUARY 2007, 77 Sayfa

Dolaşım sistemindeki kan hücrelerinden biraz daha büyükçe olan asimptomatik beyinsel emboli, transcranial Doppler ultrasound kullanılarak tespit edilebilir. Birçok durumda asimptomatik embolik sinyaller (ES) yüksek seviyedeki felç riskine işaret eder. Klinik uygulama olarak bu teknik ES deteksiyonunda güvenilir otomatik sistemin azlığından dolayı problem olur. Hastalardan elde edilen kayıtlar saatlerce sürebilir. Spektral görüntünün analiz edilmesi zaman kaybıdır ve bu gözlemcinin yorulmasına dolayısıyla hatalara neden olur. Embolus tarafından oluşturulan ES'nin kendine özgü özellikleri vardır. Bu sinyaller, kan akışı tarafından meydana getirilen sinyallerden (DS) daha büyük genliğe sahiptirler ve geçici karakteristik özellik taşırlar. Bu sinyaller kısıtlı osilasyonlu sinyallerdir ve wavelet'lere benzerler. Artifakt denen prob hareketinden veya konuşmadan oluşan birçok AR sinyalinden farklı olarak ES tek yönlüdür ve çoğunlukla akış spektrumunda yer alır. Literatürde beyinsel emboliyi ayırmak için bir çok metod çalışılmıştır. Bu çalışmada, önceki çalışmada yapılan embolik sinyal deteksiyonu sisteminin (Aydin, et all, 2004) hassasiyeti ve doğruluğunu arttırmak için veri madenciliği teknikleri kullanılmıştır.

Anahtar Kelime: emboli, deteksiyon, veri madenciliği

To My Parents

ACKNOWLEDGEMENTS

I would like to dedicate this thesis to my family.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF TABLES

LIST OF ABBREVIATIONS

AR	: Artifact Signal.
EM & ES	: Embolic Signal.
SP & DS	: Doppler Speckle Signal.
DWT	: Discrete Wavelet Transform.
DM	: Data Mining.
VM	: Veri Madenciliği.
NBTree	: Naïve Bayes Tree.
LMT	: Logistic Model Trees.
Ridor	: Ripple Down Rules.
Nnge	: Nearest Neighbor with Generalization.
VFI	: Voting Feature Interval.
Bagging	: Bootstrap Aggregating.
Dagging	: Disjoint Aggregating.
Decorate	: Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples.
SMO	: Sequential Minimal Optimization.
LWL	: Locally Weighted Learning.
RBF Network	: Radial Basis Function Neural Network.
ANFIS	: Adaptive Neuro Fuzzy Inference System.

1. INTRODUCTION

Asymptomatic circulating cerebral emboli, which are particles bigger than blood cells, can be detected by transcranial Doppler ultrasound. In certain conditions asymptomatic embolic signals (ES) appear to be markers of increased stroke risk . ES, reflected by an embolus, has some distinctive characteristics. They have usually larger amplitude than the signals from normal blood flow (Doppler speckle) and show a transient characteristic. They are finite oscillating signals and resemble wavelets. Unlike many artifacts such as caused by probe movement or speech, ES are unidirectional and usually contained within the flow spectrum. A number of methods to detect cerebral emboli have been studied in the literature. In Nebuya, et all, 2005, a phantom was constructed to simulate the electrical properties of the neck. A range of possible electrode configurations was then examined in order to improve the sensitivity of the impedance measurement method for the in vivo detection of air emboli. In Demchuk, et all, 2006, angiographically validated criteria for circle-of-Willis occlusion and thrombolysis in brain ischemia classification of residual flow have set the stage for the further development of Transcranial Doppler technique. In Chung, et all, 2005, the purpose of study was to improve reliability in the identification of Doppler embolic signals by determining the decibel threshold for reproducible detection of simulated "emboli" as a function of signal duration, frequency and cardiac-cycle position. In Okamura, et all, 2005, it has investigated that embolic particles could be detected as high-intensity transient signals with a Doppler guide wire during percutaneous coronary intervention in patients with acute myocardial infarction. In Girault, et all, 2006, in order to detect embolus, simple "off-line" synchronous detector has considered. In Kouame, et all, 2006, for detection of microemboli with an expert knowledge, an autoregressive modeling associated with an abrupt change detection technique was used. In Palanchon, et all, 2005, instead of using Doppler techniques for emboli detection, a new technique has presented. This new technique consists of a multi-frequency transducer with two independent transmitting elements and a separate receiving part with a wide frequency band. In Mackinnon, et all, 2005, to determine patterns of embolization in two conditions and optimal recording protocols, ambulatory Transcranial Doppler system has applied to patients who have symptomatic and asymptomatic carotid stenosis. In Cowe, et all, 2005, Artifacts generated by healthy volunteers and embolic signals recorded from a flow phantom were used to characterize the appearance of two types of event. In Kilicarslan, et all, 2006, the relationship between microembolic signals, microbubble detection, and neurological outcome has discussed. In Rodriguez, et all, 2006, the effect of choosing different thresholds on the sensitivity and specificity of detecting high-intensity transient signals during cardiopulmonary bypass has investigated. In Dittrich, et all, 2006, the aim was installing primary and secondary quality control measures in clopidogrel and aspirin for reduction of emboli in symptomatic carotid stenosis. This is because microembolic signals evaluation relies on subjective judgment by human experts. In Chen, et all, 2006, main goal was to use multi-frequency transcranial Doppler to initially characterize emboli which is detected during carotid stenting with distal protection. In Rosenkranz, et all, 2006, the association of the number of solid cerebral microemboli during unprotected Carotid artery stent placement with the frequency of silent cerebral lesions which have detected by diffusion-weighted MR imaging has prospectively evaluated. In this study, data mining techniques have been used in order to increase sensitivity and specificity of an embolic signal detection system previously described by Aydin, et all, 2004. Data mining techniques have been used in order to increase sensitivity and specificity of an embolic signal detection system. Similarly to the fuzzy approach, ANFIS has been used. Using sixteen methods in Weka and ANFIS, some results have been obtained. For the comparison purpose, testing specificity has been considered as the classification result of signals. Finally it has seen that these results were close or more accurate than the previous results in Aydin, et all, 2004.

1.1. Aim

In (Aydin, et all, 2004), main motivation was to detect asymptomatic emboli in arteries which may be an indication of stroke risk. For detecting asymptomatic emboli by using transcranial Doppler ultrasound signals, ES caused by emboli, DS caused by blood flow, and AR caused by other elements have been classified by fuzzy logic principle. The recorded signals from the patients consist of ES DS and AR. To distinguish ES from the others, a system consisting of DWT was developed (Aydin, et all, 2004). In this study, instead of using the fuzzy logic method, data mining techniques which is expected to enhance sensitivity and specificity of the previous system have been used.

2. BACKGROUND

A number of methods for detecting cerebral emboli using Doppler ultrasound have been studied in the literature. These include improving the sensitivity of the impedance measurement method for the in vivo detection of air emboli (Nebuya, et all, 2005), setting “Angiographically validated criteria for circle-of-Willis occlusion and thrombolysis in brain ischemia classification of residual flow” as stage for improvement of development of Transcranial Doppler technique (Demchuk, et all, 2006), improving reliability in the identification of Doppler embolic signals by determining the decibel threshold for reproducible detection of simulated "emboli" as a function of signal duration, frequency and cardiac-cycle position (Chung, et all, 2005), investigating whether embolic particles could be detected as high-intensity transient signals with a Doppler guide wire during percutaneous coronary intervention in patients with acute myocardial infarction (Okamura, et all, 2005), and considering a simple "off-line" synchronous detector to detect embolus (Girault, et all, 2006). The previous work (Aydin, et all, 2004) was about detection of asymptomatic emboli in arteries that could have been an evidence of stroke risk. To be able to detect emboli, fuzzy logic detection system was used. However, there are other kind of signals. Those are DS and AR and which are mixed with ES. In order to distinguish ES, these three signals must be analyzed. ES is a high intensity signal resulted from emboli particles. AR is produced by tissue movement, speaking, probe tapping, etc. DS is caused by blood flow. Usually, the bandwidth of ES is narrower than DS. Briefly, the process of the data preparation and fuzzy detection are given as follows:

1) The signals from 35 patients having symptomatic carotid stenosis were recorded by using a transcranial Doppler ultrasound system (Pioneer TC4040).
2) These recorded quadrature audio Doppler signals including ES were exported to a PC.
3) In order to evaluate feasibility of these signals, two independent data sets each including 100 ES, 100 AR and 100 DS were used.
4) After applying 8 order DWT to the exported data, 15 parameters which have ES, AR and DS each, were evaluated. In Table 1, 12 of these 15 parameters are shown.
5) Then, threshold values which are identified before were applied to each of the fifteen parameters.
6) At last; using output which was resulted from fuzzy logic, another fuzzy logic detection has been performed as well. Therefore, ES, AR and DS were classified. From the first data set, while ES has been detected as 98%, from the second data set, ES has been detected as 95%. In this study, the second result is considered and compared with results of used DM methods. Although the DWT coefficients of scales 5-6-7-8 were dominated by AR, the DWT coefficients of scales 1-2-3-4 were dominated by ES and DS.

P2TR means the scale with maximum peak to threshold ratio. TP2TR indicates the total power to the threshold ratio. RR is the ES rise rate; FR is the ES fall rate. F2RM indicates peak forward to reverse power ratio. TF2R means total forward to reverse power ratio. T_s^2 is time spreading term and B_s^2 is the frequency spreading term. VIE and VIF are algorithm variances as instantaneous envelope of signals and instantaneous frequency of signals, respectively. t_s is the average time of the signal and f_s is the average frequency of the signal, respectively. $s(t)$ is the probability distribution. $S(f)$ is the Fourier transform of $s(t)$.

Table 1. Twelve Parameters with Threshold Values.

	th1	***th2***	***th3***	***th4***
$P2TR$ (dB)	6	12	14	20
$TP2TR$(dB)	17	23	26	38
$F2RM$ (dB)	10	20	22	26
$TF2R$(dB)	4	8	10	20
RR (ms)	0.6	1.4	2	5
FR (ms)	0.6	1.4	2	6
t_s (ms)	10	20	60	120
f_s / F_s(unit)	0.01	0.035	0.08	0.1
T_s^2 (ms^2)	6	18	40	100
B_s^2 / F_s (unit)	0.03	0.06	0.1	0.4
VIE (unit)	12	60	100	140
VIF / F_s (unit)	0.008	0.016	0.021	0.04

The thresholds are obtained by statistical evaluation of the ES, DS and AR (Aydin, et all, 2004). These thresholds are used in fuzzy logic.

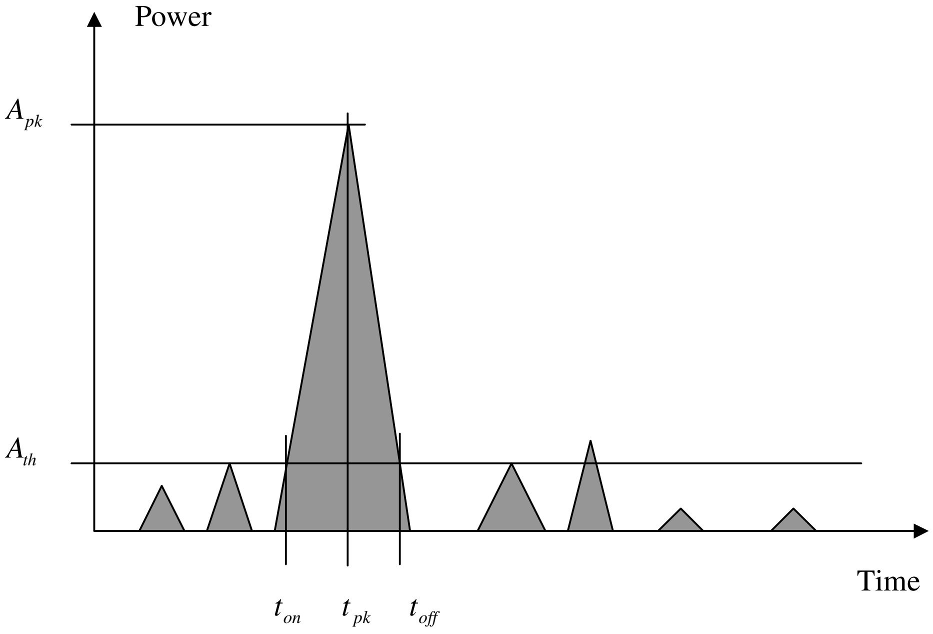


Figure 1. A Representative Constructed Wavelet Scale Containing an ES and Quantities to Calculate Parameters.

In (1) and (2), the mathematical representations of the parameters are given. $A_f(k)$ and $A_r(k)$ are instantaneous power (IP) of forward and reverse signals, respectively.

$P2TR = 10\log\frac{A_{pk}}{A_{th}}(dB)$ ***where*** A_{pk} ***and*** A_{th} ***are IP forward and reverse signals, respectively.***

$$TP2TR = 10\log\frac{A_{tot}}{A_{th}} = 10\log\frac{\sum_{k=t_{on}}^{t_{off}} A_f(k)}{A_{th}}(db)$$

$$RR = \frac{10\log\frac{A_{pk}}{A_{th}}}{t_{pk} - t_{on}} = \frac{P2TR}{t_{pk} - t_{on}}(dB/ms) \tag{1}$$

$$FR = \frac{10\log\frac{A_{pk}}{A_{th}}}{t_{off} - t_{pk}} = \frac{P2TR}{t_{off} - t_{pk}}(dB/ms)$$

$$TF2R = 10\log\frac{\sum_{k=t_{on}}^{t_{off}} A_f(k)}{\sum_{k=t_{in}}^{t_{off}} A_r(k)}(dB)$$

In (2), it is continued.

$t_s = \frac{1}{E_s}\int_{-\infty}^{\infty} t\left|s(t)\right|^2 dt$ ***where*** t_s ***is the average time of the signal.*** $s(t)$ ***is the probability distribution.***

$f_s = \frac{1}{E_s}\int_{-\infty}^{\infty} f\left|S(f)\right|^2 df$ ***where*** $S(f)$ ***is the fourier transform of*** $s(t)$***.*** $S(f)$ ***is the average frequency of the signal.*** (2)

$$T_s^2 = \frac{1}{E_s}\int_{-\infty}^{\infty} (t-t_s)^2\left|s(t)\right|^2 dt$$

$$B_s^2 = \frac{1}{E_s}\int_{-\infty}^{\infty} (f-f_s)^2\left|S(f)\right|^2 df \quad \textbf{\textit{where}} \quad E_s = \int_{-\infty}^{\infty}\left|s(t)\right|^2 dt < \infty$$

Data mining is a multidisciplinary field of research and development of algorithms and software environments to support this activity in the context of real-life problems where often huge amounts of data are available for mining. Data mining is sometimes considered as just a step in an overall process called **K**nowledge **D**iscovery in **D**atabases, KDD. Data mining includes a large set of technologies, including data warehousing, database management, data analysis algorithms and visualization (Apte, et all, 1997), (Mullins, et all, 2006). In this study, seventeen data mining methods have been used. By using DM methods in detection of emboli, it is expected to increase the sensitivity and specificity of the system in previous work. The detailed descriptions of these seventeen methods have been given as below.

2.1. Naive Bayes

Bayesian networks are a popular medium for graphically representing and manipulating attribute interdependencies and represent a joint probability distribution over a set of discrete, stochastic variables. Bayesian classification has been widely used in many machine learning applications, also in medical diagnosis. The Bayesian approach searches in a model space for the “best” class descriptions. A best classification optimally trades off predictive accuracy against the complexity of the classes, and so does not overfit the data. Such classes are also fuzzy, instead of each case being assigned to a class, a case has a probability of being a member of each of the different classes.

Bayesian networks have several advantages for data analysis. Firstly, since the model encodes dependencies among all variables, it readily handles situations where some data entries are missing. Secondly, a Bayesian network can be used to learn causal relationships and hence can be used to gain understanding about a problem domain and to predict the consequences. Thirdly, because the model includes both causal and probabilistic semantics, it is an ideal representation for combining prior knowledge, which often comes in causal form and data. Fourthly, bayesian statistical methods in conjunction with bayesian networks offer an efficient and widely recognized approach for avoiding the over-fitting of data. Finally, it is found that diagnostic performance with Bayesian networks is often surprisingly insensitive to imprecision in the numerical probabilities. Bayesian networks are directed acyclic graphs that allow for efficient and effective representation of joint probability distributions over a set of random variables.

The Naive Bayesian Classifier is one of the most computationally efficient algorithm for machine learning and data mining. The naive Bayesian classifier is a bayesian network, used for classification. It is a probabilistic approach to classification. Compared with neural networks, decision trees, clustering and regression, the naive bayesian classifier is a simple and, effective classifier. For example, in medical area, mineral potential mapping is used as naive bayesian classifier. Belonging to the bayesian network classifier, bayesian classifier predicts a class C for a pattern x. The expression of the Bayesian Classification is shown in equation (3) (Ouali,et all, 2005).

$$P(C|X = x) = \frac{p(X = x|C) \bullet P(C)}{p(X = x)} \tag{3}$$

2.2. Naive Bayes Tree (NBTree)

Naive Bayes Tree (NBTree) algorithm is a hybrid approach when many attributes are likely to be relevant for a classification task. NBTree is similar to the classical recursive partitioning schemes, except that the leaf nodes created are Naive Bayes categorizers instead of nodes predicting a single class. A threshold for continious attributes is chosen using the standard entorpy minimizatioin technique, for decision trees. NBTree uses decision tree techniques to partititon the whole instance space (root node) into many subspaces (leaf nodes) then trains Naïve Bayes classifier for each leaf node. NBTree produces highly accurate classifiers. In Algorithm 1, the process of Naive Bayes Tree is shown (Kohavi, et all, 1996), (Fonseca, et all, 2003), (Xie, et all,2004).

Algorithm 1. The Process of naive Bayes Method.

1) For each attribute X_i , evaluate u (x_i) of a split on attribute x_i .
2) Let j=argmax(u_i) (The attribute with highest value).
3) If u_j is significantly not better than the utility of the current node, create a Naive Bayes Classifier for the current node and return.
4) Partition T according to the test on x_j . If x_j is continuous, a threshold split is used ; if x_j is discrete, a multi-way split is made for all possible values.
5) For each child, call the algorithm recursively on the portion of T that leads to the child.

2.3. Logistic Model Trees (LMT)

Model trees predict a numeric value which is defined over a stationary set of numeric or nominal attributes. Unlike regression trees, model trees produces piecewise linear approximation to the destination function. At the end, the final model tree includes a decision tree with linear regression models at leaves also the prediction of instance is gathered with using the prediction of the linear model that is associated with the leaf. Unsimilarly with model trees, Logistic Model Trees or LMT employs an efficient and flexible approach for building logistic models using the well-known CART algorithm for pruning. The process of LMT starts by building a logistic model at the root using LogitBoost algorithm. The pseudocode for LogitBoost algorithm is shown in Algorithm 2.

Algorithm 2. The Process of Logit Boost Method.

```
1. Start with weights
     $w_{ij} = 1/\mathrm{N}$ , $\mathrm{i} = 1,\dots,\mathrm{N}$ , $\mathrm{j} = 1,\dots,\mathrm{J}$ , $\mathrm{F_j}(x) = 0$ and $\mathrm{p_j}(x) = 1/J$
2. Repeat for
     $m = 1,\dots,M$ :
     $a)$ Repeat for $\mathrm{j} = 1,\dots,\mathrm{J}$ :
          i. Compare working responses and weights in jth class
               $\mathrm{z_{ij}} = \frac{y_{ij} - p_j(x_i)}{p_j(x_i)(1 - p_j(x_i))}$ , $\mathrm{w_{ij}} = p_j(x_i)(1 - p_j(x_i))$
          $ii.$ Fit the function $\mathrm{f_{mj}}(x)$ by a weighted least squares regression
               of $\mathrm{z_{ij}}$ to $\mathrm{x_i}$ with weights $\mathrm{w_{ij}}$
     $b)$ Set $\mathrm{f_{mj}}(x) \leftarrow \frac{J-1}{J}(f_{mj}(x) - \frac{1}{J}\sum_{k=1}^{J} f_{mk}(x)), F_j(x) \leftarrow F_j(x) + f_{mj}(x)$
     $c)$ Update $\mathrm{p_j}(x) = \frac{e^{F_j(m)}}{\Sigma_{k=1}^{J} e^{F_k}(x)}$
3. Output the classifier $\mathrm{argmax}_j F_j(x)$.
```

The iterations are determined by using five fold cross validation. The data is split into training and test as five times. Logit Boost algorithm is run to a maximum number of iterations. Next, the error rates on test set are gathered then summed up over different folds. After that, the number of iterations with the lowest sum of errors is used to train the LogitBoost algorithm. This process is called the logistic regression model, at the root of the tree. The formulation of linear logistic regression is

$$\Pr(G = j \mid X = x) = \frac{e^{F_j(x)}}{\sum_{k=1}^{J} e^{F_k(x)}}, \sum_{\mathrm{k}=1}^{\mathrm{J}} F_k(x) = 0, \tag{4}$$

where Pr(G=j|X=x) is the pasterior class probability for J classes with functions x and $F_j(x) = \beta_j^T.x$. Therefore, considering binary splits on numerical attributes and multiway splits, on nominal attributes, by using C4.5 algorithm, a split of the data at the root is constructed. The LogitBoost algorithm is run on child node by starting with the committee $F_j(x)$, weights w_{ij} and probability estimates p_{ij} of the last iteration. The splitting continues as long as 15 instances are at a node. Finaly, the tree is pruned using CART pruning algorithm (Landwehr, et all, 2003).

2.4. Ripple Down Rules (Ridor)

RIpple-**DO**wn **R**ule or Ridor is a knowledge acquisition technique. The major feature of ripple down rules is, first, they can be added to a knowledge base faster than relational rules since the rules are added without modification. Secondly, since ridor is only used in context, they have less impact in damaging the knowledge base. With ridor, redundancy is the major problem because since knowledge is entered in context, the similar rule may end up being repeated in multiple contexts. Ripple Down Rules create exceptions to existing rules so changes are bounded in the context of the rule and will not affect other rules. Ripple Down Rules look like decision lists which are in the form if-then-else as new RDR rules are added by creating except or else branches to the existing rules. If a rule fires but produces an incorrect conclusion then for the new rule, an except branch is created. If no rule fires then an else branch is created for the new rule. The simplest Ridor pseudocode is shown in Algorithm 3.

Algorithm 3. Ridor Algorithm.

```
If a^b then c
except if d then e
else if f^g then h
```

The rule is implemented as, if a and b are true then we conclude c unless d is false. If d is false, we conclude e. If a and b are not true we continue with other rules that if f and g true then we conclude h. In order to create an exception to a rule, the algorithm should recover the word which caused the rule fired. In Figure 2, we can see the process of Ridor. The two black ellipses at the root are LAST_FIRED(0) rules. The process is considered one by one, left to right. To implement the fire, the prediction for either of these rules is “no other rule has fired”. The rule on the left is considered first. If it doesn’t fire, the next oldest LAST_FIRED(0) is considered. If a LAST_FIRED(0) rule does fire then in the next level, only the rules connected to it are applicants to fire. The newest against the oldest is added. Once one of these child rule is added then the only rules connected to it are net ones to fire (Compton, et all, 1990).

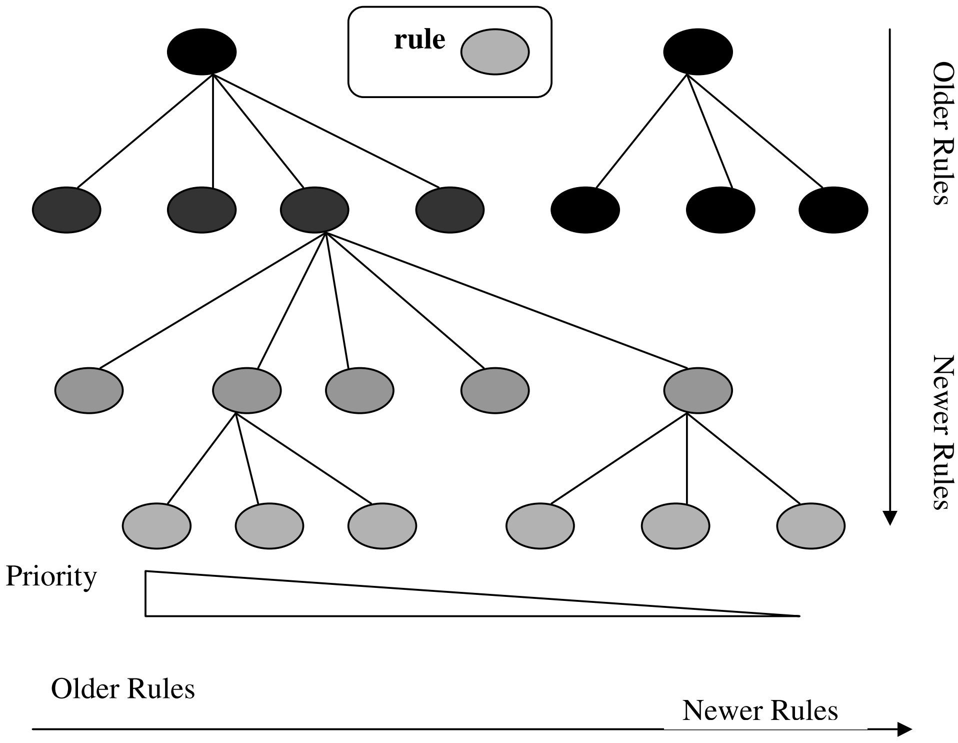


Figure 2. Representation of Ripple Down Rules.

In Industry, the variation of Ridor rules such as **M**ultiple **C**lassification of **R**ipple **D**own **R**ules (MCRDR) is developed. MCRDR incrementally and rapidly acquires and validates the knowledge which is on case by case criteria. Cases are used to validate the acquired knowledge (Richards, et all, 2002).

2.5. Nearest Neighbor with Generalization (Nnge)

Nearest Neighbor with Generalization (Nnge) is the Nearest Neighbor like algorithm, using non-nested generalized attributes. If the data is randomly spreaded, the algorithm employs the ratio “R” to the mean expected distance. This ratio R can be used for the three moments of the expected distance: the mean expected distance, the standard deviation of expected distance and the skewness of the frequency distribution of these distances. In k dimensional space, these three moments can be derived for a random dispersion of individuals. As the spherical volume of the sphere is given by

$$V_{sp} = \frac{\pi^{k/2} r^k}{\Gamma((k/2)+1)} \tag{5}$$

the probability of sphere that doesn't contain no other individual is

$$e - \frac{\rho \prod^{k/3} r^k}{\Gamma((k/2)+1)} \tag{6}$$

where ρ is the density of the population. This equation is also called the amount of distances to nearest neighbor, greater than r. Instead of nearest neighbor algorithm, Nnge obtains outputs from regular exemplars (Clark, et all, 1979). In (Pignotti, et all, 2004), Service Predictor and Alert System predicts the next service and checks whether time and user's location are appropriate or not. For the system, sequence rules do not provide time and location information to the user. By using NNGE, proximity rules which have time and location information to the user is generated.

2.6. Voting Feature Interval (VFI)

Voting Feature Interval (VFI) is the method for implementing the Voting Feature Interval classifier. VFI is faster than NaiveBayes algorithm. For VFI, each training example is represented as a vector of feature values with a label representing the class of the example. From the training examples, VFI constructs feature intervals for each feature. A feature interval represents a set of values of a given feature where the same subset of class values is observed. Two neighboring intervals contain different sets of intervals. The training process of VFI is given in Algorithm 4. In the training phase of VFI, the feature intervals for each feature dimension are constructed. The procedure find_end_points(TrainingSet,f,c) finds the lowest and the highest values for linear feature f from the examples of class c and each observed value for nominal feature f from the examples in TrainingSet. For each linear feature, 2k values are found. k is the number of classes. Next, the list of 2k end-points is sorted and each consecutive pair of points selects a fair interval. Each interval is represented by a vector of $\langle lower, \mathrm{count}_1, \ldots, count_k \rangle$ where lower is the lower bound of that interval and $count_i$ is the number of training instances of class i falling into that interval. The count_i values are computed by count_instances(i,c).

Algorithm 4. The Training Process of VFI Method.

```
train(TrainingSet):
begin
        for each feature f
                for each class c
                        EndPoints[f]=EndPoints[f] $\cup$ find_end_points(TrainingSet,f,c);
                sort(EndPoints[f]);
                for each class c
                        interval_class_count[f,i,c]=count_instances(f,i,c);
end
```

The classification phase of the VFI algorithm is given in Algorithm 5. The process starts by initializing the votes for each class by zero. e_f is the f value of the test example e. For each feature f, the interval on feature dimension f which e_f falls into, is searched. If e_f is missing, the corresponding feature gives a vote zero for each class. Thus, the features containing missing values are simply ignored. If e_f is known, the interval i which e_f falls into, is found. For each class c, feature f gives a vote equal to

$$feature\_vote[f,c] = \frac{interval\_class\_count[f,i,c]}{class\_count[c]} \qquad (7)$$

where *interval_class_count[f,i,c]* is the number of examples of class c which fall into interval i of feature dimension f. Each feature f collects its votes in an individual vote vector $\left\langle vote_{f,1},\ldots,vote_{f,k}\right\rangle$, where $vote_{f,c}$ is the individual vote of feature f for class c. k is the number of classes. Then, the individual vote vectors are summed up to get a total vote vector $\left\langle vote_1,\ldots,vote_k\right\rangle$. At last, the class with the highest total vote is predicted to be the class of the test instance (Demiroz, et all, 1997). In industry, VFI can be used for signal analysis such as noise analysis of network problems (Kalapanidas, et all, 2003).

Algorithm 5. The Classification Phase of VFI Method.

```
classify(e):
begin
    for each class c
        vote[c]=0
    for each feature f
        for each class c
            feature_vote[f,c]=0
        if $e_f$ value is known
            i=find_interval(f, $e_f$ )
            $feature\_vote[f,c] = \frac{interval\_class\_count[f,i,c]}{class\_count[c]}$
            normalize_feature_votes(f);
            for each class c
                vote[c]=vote[c]+feature_vote[f,c];
        return class c with highest vote[c];
end
```

2.7. Bootstrap Aggregating (Bagging)

Bootstrap Aggregating (Bagging) is the method of stacked generalization in combining models derived from different subsets of a training dataset by a single learning algorithm. Given a training dataset T of size N, standard batch bagging creates M base models. Each model is trained by calling the batch learning algorithm L_b on a bootstrap sample of size N which is created by drawing random samples with replacement from original data set. The pseudocode of bagging is given in Algorithm 6 (Oza, et all, 2005).

Algorithm 6. Bagging Algorithm.

```
Bagging(T, $L_b$ ,M)
        For each m $\in$ {1,2,...,M},
                $T_m$ = Sample_With_Replacement(T,|T|)
                $h_m = L_b(T_m)$
        Return{ $h_1, h_2, ..., h_m$ }
```

In order to combine predictions from different models derived from a single learning algorithm, majority vote is used. Bootstrap samples are randomly sampling L with replacement into K subsets of size N. Bootstrap samples are used in order to deliver training subsets. Let us we have a learning set $L = \{(y_n, x_n), n = 1, \ldots, N\}$ where y's are either class labels or numeric response. And let us call $\varphi(x, L)$ a predictor. If the input is x, we predict y by $\varphi(x, L)$. Now, let us we are given a sequence of learning sets $\{L_k\}$, each including of N independent observations from the same underlying distribution as L. Our aim is to use the L_k to get a better predictor than the single learning set predictor $\varphi(x, L)$. If y is numerical, then we can replace $\varphi(x, L)$ by the average of $\varphi(x, L_k)$ over k. If $\varphi(x, L)$ predicts a class $j \in \{1, \ldots, J\}$, then we can gather $\varphi(x, L_k)$ by voting. Let $N_j = \#$ $\{k; \varphi(x, L_k) = j\}$ and take $\varphi_A(x) = \arg\max_j N_j$. Let us take repeated bootstrap samples $\{L^{(B)}\}$ from L then form $\{\varphi(x, L^{(B)})\}$. If y is numerical, we take φ_B as $\varphi_B(x) = av_B \varphi(x, L^{(B)})$.. If y is a class label, let $\{\varphi(x, L^{(B)}\}$ vote to form $\varphi_B(x)$. This procedure is called "**B**ootstrap **Agg**regat**ing**" or **Bagging** (Breiman,et all., 1996).

2.8. Disjoint Aggregating (Dagging)

Dagging stands for **D**isjoint **Agg**regat**ing.** In Dagging, similar to bagging, it uses majority vote to combine multiple models, obtained from a single learning algorithm. Unlike bagging, dagging uses disjoint samples, rather than bootstrapping. The training set is partitioned into k subsets. A base classifier generates a hypothesis for each subset. The final prediction is done by plurality vote as same in bagging. Another difference is, dagging doesn't use extra resources since the same amount of examples are used as the training set (Tate, et all, 2005). Let us we have I output classes and let $p_{ki}(x)$ denote the probability that the k_{th} model assigns to the i_{th} class that is given the test instance x. The vector

$$P_{kn} = (pk1(x_n),\ldots, pki(x_n),\ldots, pkI(x_n)) \tag{8}$$

gives the k_{th} model's class probabilities for the n_{th} instance. At the end of the testing, data evaluated from the output of the K models is

$$\tilde{L} = \{(y_n, p_{1n},\ldots, p_{kn},\ldots, p_{Kn}), n = 1,\ldots, N'\} \tag{9}$$

This is called level-1 data. Using a learning algorithm that is called level-1 generalizer, we obtain a model $\tilde{M}$ called level-1 model that predicts the class from this level-1 data. In order to classify a new instance, the level-0 models M_k are used to produce a vector $(p_{11},\ldots, p_{1I},\ldots, p_{k1},\ldots, p_{kI},\ldots, p_{K1},\ldots, p_{KI})$ that is input to the $\tilde{M}$ and the output $\tilde{M}$ is the final classification result of it. Depending on the sampling strategy used to produce the data derived from level-0 models, we call this implementation Stacked Generalization Bag Stacking or Dag Stacking (Ting, et all, 1997).

2.9. Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples (Decorate)

Decorate, stands for **D**iverse **E**nsemble **C**reation by **O**ppositional **R**elabeling of **A**rtificial **T**raining **E**xamples. The combination of the output of several classifiers is useful if they disagree on some inputs. This disagreement is referred as diversity of the ensemble. For regression problems, generally, mean squared error is used to measure accuracy while the variance is used to measure diversity. The generalization error E of the ensemble can be expressed as $E = \overline{E} - \overline{D}$ where $\overline{E}$ and $\overline{D}$ are mean error and diversity of the ensemble, respectively. Let us call $C_i(x)$ as the prediction of the i_{th} classifier for the label of x. And let us call $C^*(x)$ as the prediction of the entire ensemble. The diversity of the i-th classifier is evaluated on example x as

$$d_i(x) = \begin{cases} 0: \text{ If } \mathrm{C_i}(x) = \mathrm{C}^*(x) \\ 1: \text{ Otherwise} \end{cases} \tag{10}$$

To compute the diversity of an ensemble of size n on a training set of size m, the above term is averaged:

$$\frac{1}{nm} = \sum_{i=1}^{n} \sum_{j=1}^{m} d_i(x_j) \tag{11}$$

In Decorate algorithm, an ensemble is generated iteratively. First, learning a classifier then adding it to the current ensemble is performed. In Algorithm 7, Decorate algorithm is shown. The classifiers in each successive iteration are trained on the original training data. In each iteration, artificial training examples are generated from the data distribution. R_{size} is the number of examples to be generated.

Decorate has much more accurate than Bagging and Boosting algorithms (Merville, et all, 2004).

Algorithm 7. Decorate Algorithm.

```
Input :
BaseLearn  : Base Learning algorithm.
T          : Set of m training examples $\langle (x_1, y_1), \ldots, (x_m, y_m) \rangle$ with labels $y_j \in Y$
$C_{size}$ : Desired ensemble size.
$I_{max}$  : Maximum number of iterations to build an ensemble.
$R_{size}$ : Factor that determines number of artificial examples for generation.
1) $i = 1$
2) trials = 1
3) $C_i$ = BaseLearn(T)
4) Initialize ensemble $C^* = \{C_i\}$
5) Compute ensemble error $\in = \frac{\sum_{xj \in T, C^*(x_j) \neq y_j} 1}{m}$
6) While $i < C_{size}$ and trials $< I_{max}$
7) Generate $R_{size} \times |T|$ training examples R based on distribution of training data
8) Label examples in R with probability of class labels inversely
        proportional to predictions of $C^*$
9) $T = T \cup R$
10) $C'$ = BaseLearn(T)
11) $C^* = C^* \cup \{C'\}$
12) $T = T - R$ , remove the artificial data
13) Compute training error $e'$ of $C^*$ as in step 5
14) If $e' \leq e$
15) $i = i + 1$
16) $e = e'$
17) otherwise,
18) $C^* = C^* - \{C'\}$
19) trials = trials + 1
```

2.10. Ada Boost M1

In order to improve the classification accuracy, AdaBoost algorithm has been theoretically proved to be an effective method. It focuses on minimization of training errors. However, when data is noisy, AdaBoost might have problem with overfitting (Jin, et all, 2003). AdaBoost.M1, which is the variation of AdaBoost algorithm can be used to improve the performance of a learning algorithm. Or it can be used to reduce the error of a classifier. The boosting algorithm AdaBoost.M1 takes a training set of m examples

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \tag{12}$$

as input. x_i is an instance drawn from some space x and it is a vector of attribute values. $y_i \in Y$ is the class label, associated with x_i. The boosting algorithm calls a learning algorithm, called WeakLearn, in many times. The aim of the weak learn algorithm is to find a value h_t which minimizes the training error. The disadvantage of AdaBoostM1 is, when the training error is greater than 0.5, algorithm cannot handle with weak values of h_t. In Algorithm 8, the process of AdaBoostM1 is shown (Freund,et all, 1996).

Algorithm 8. The Process of Ada Boost M1 Method.

1) Call weakLearn, providing distribution D_t,
2) Get back hypothesis $h_t \quad \mathrm{X} \rightarrow \mathrm{Y}$,
3) Calculate the error of h_t,
4) Update distribution D_t .

2.11. Sequential Minimal Optimization (SMO)

The **S**upport **V**ector **M**achine (SVM) algorithm is a classification technique. Given training vectors $x_i \in R^n, i = 1,\ldots,l$ in two classes and a vector $y \in R^l$ like $y_i \in \{1,-1\}$, SVM gives the following problem

$$\min \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \qquad \begin{array}{l} 0 \le \alpha_{\mathrm{i}} \le C, i = 1,\ldots,l, \\ y^T\alpha = 0 \end{array} \tag{13}$$

where e is the vector of all ones, C is the upper bound of all variables and Q is an *l* by *l* positive semi definite matrix (Lin,et all, 2000). Before we understand the Sequential Minimal Optimization (SMO) algorithm, let us examine the Support Vector Machine Quadratic Programming or SVM QP. SVM QP is known as chunking. The chunking algorithm uses the fact that if you remove the rows and the columns of the matrix corresponding to zero multipliers, the value of the quadratic form is the same. Thus, the large QP problem can be broken down into series of small QP problems whose main goal is to detect all of the non zero Lagrange multipliers and discard all of the zero Lagrange multipliers. Sequential Minimal Optimization is the simple algorithm that can rapidly solve the SVM QP problem without any extra matrix storage and without using numerical QP optimization steps at all. In Algorithm 9, the SMO algorithm is shown.

Algorithm 9. SMO Algorithm.

1) Divide QP problem into partitions,
2) Choose and solve smallest optimization on at every step,
3) Choose two Lagrange multipliers, find optimal values for these multipliers,
4) Update SVM.

SMO divides the overall QP problem into QP sub problems. Unlike other methods, SMO chooses to solve the smallest possible optimization problem at every step. At every step, SMO chooses two Lagrange multipliers to optimize and finds optimal values for these multipliers then updates the SVM to reflect the new optimal values. The advantage of SMO comes from easily solving two Lagrange problems, analytically. In addition, SMO does not require no extra matrix storage at all. SMO can be changed to solve fixed threshold SVM's. SMO updates individual Lagrange multipliers to be the minimum value of function Ψ along the corresponding dimension. The update rule is

$$\alpha_1^{new} = \alpha_1 + \frac{y_1 E_1}{K(\vec{x}_1 \vec{x}_1)} \tag{14}$$

where α is the Lagrange multiplier, E_i is the training error in i_{th} example K is the kernel x is input and y is the target. This update equation forces the output of SVM to be y_1 (Platt,et all,1998). In some areas, the implementation of SMO algorithm such as SMOBR which is written in C++ is used to work with data sets such as with data sets on hyperplanes (Kornienko, et all, 2005). In test results, SMOBR has shown better performance than other SVM methods.

2.12. Classification via Regression

This method uses regression methods C5.0, M5.0 and LR to perform classification. At the leaves, model trees execute the process of predicting continuous numeric values by using function approximation. For better understanding, let us consider we have a model at a leaf involved two attributes x and y with linear coefficients a and b and the model at the parent node involved two attributes y and z are

$$p = ax + by \tag{15}$$

and

$$q = cy + dz \tag{16}$$

In Classification via Regression, these two models are come together with formula

$$p' = \frac{na}{n+k}x + \frac{nb+kc}{n+k}y + \frac{kd}{n+k}z \tag{17}$$

where p' is the prediction passed up to the next higher node. p is the prediction passed to this node. q is the value predicted by the model at this node. n is the number of training instances which reach the node below and k is the constant. It continues until root gives a single smoothed linear model that will be used for the prediction (Frank, et all, 1998). Classification via Regression is used in biological areas for example in Langdon, et all, 2003, it is used in genetic programming to investigate the biochemical interactions with human P450 2D6 enzyme.

2.13. Locally Weighted Learning (LWL)

LWL stands for **L**ocally **W**eighted **L**earning. In LWL, linear regression model is fit to the data that is based on a weighting function and centered on the instance where a prediction is about to be generated. The resulting estimation is linear. To illustrate how locally weighted learning works, let us consider distance weighted averaging. Equation (18) is locally weighted regression because the local model is constant. A prediction $\hat{y}$ can be based on an average of n training values $\{y_1, y_2, \ldots, y_n\}$ is

$$\hat{y} = \frac{\sum y_i}{n} \tag{18}$$

this estimate minimizes a criterion:

$$C = \sum_i (\hat{y} - y_i)^2 \tag{19}$$

In the case where the training values $\{y_1, y_2, \ldots, y_n\}$ are taken under different conditions $\{x_1, x_2, \ldots, x_n\}$. In **L**ocally **W**eighted **R**egression LWR, local models are fit to nearby data. LWR can be derived by either weighting the training criterion for the local model or by directly weighting the data (Atkenson, et all,1997). Training of LWR is fast. It just requires adding new training data to the memory. When a prediction is needed for a query point x_q, the following weighted regression analysis is performed:

Algorithm 10. Locally Weighted Regression Algorithm.

```
Given:
        A query point $X_q$ and p training points $\{x_i, y_i\}$ in memory.
Compute Prediction:
a)Compute diagonal weight matrix W where
        $w_{ii} = \exp\left(-\frac{1}{2}(x_i - x_q)^T D (x_i - x_q)\right)$
b)Build matrix X and vector y such that
        $X = (\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_p)^T$ where $\tilde{x}_i = \left[(x_i - x_q)^T\right]^T$
        $y = (y_1, y_2, \ldots, y_p)^T$
c)Compute locally linear model
        $\beta = (X^T W X)^{-1} X^T W y$
d)The prediction for $x_q$ is thus
        $\hat{y}_q = \beta_{n+1}$
```

β_{n+1} denotes $(n+1)^{th}$ element of the regression vector β (Schaal,et all,2002). Locally weighted learning is critically dependent on the distance function. The three distance functions are Global Distance Functions, Query-based Local Distance Functions and Point- based local distance functions. Global distance functions are used in input space. For Query based local distance functions, the distance function, d() parameters are set on each query by an optimization process which typically minimizes the cross validation error. In point based local distance functions, the training criterion uses a different d() for each point x_i:

$$C(q) = \sum_i \left[(f(x_i, \beta) - y_i)^2 K(d_i(x_i, q)) \right] \tag{20}$$

the $d_i()$ can be selected either by a direct computation or by minimizing cross validation error. In locally weighted learning, it is possible to estimate the prediction error and gather confidence bounds in the predictions. LWL shows robust effects according to its most important parameter, the neighborhood size (Atkenson, et all, 1997).

2.14. Simple Logistic

Simple Logistic is the class for building a logistic regression model, using LogitBoost. The Logistic Regression Model Boosting is performed by applying a classification algorithm to reweighted type of training data and then giving a weighted majority vote of the sequence of classifiers. LogitBoost performs classification using a regression scheme as the base learner using additive logistic regression. The additive logistic regression has the form

$$F(x) = \sum_{j=1}^{p} f_j(x_j) \tag{21}$$

each component of f_j is a function belonging to small subset of variables x_j. The logistic regression model is used when the response variable of interest takes on two values (Friedman, et all, 2000). Let us define a response variable as y and denote event as y=1 when the subject has the characteristic of interest and y=0 when the subject does not. Also let us suppose the subject has a single predictor x that could be related to the response. The logistic regression defines the probability P(y=1) as,

$$P(y=1) = \frac{\exp(\beta_0 + \beta_1 x)}{1 + \exp(\beta_0 + \beta_1 x)} \tag{22}$$

and $P(y=0) = 1 - P(y=1)$. This model has an appropriate presentation for y=1 as

$$odds(y=1)=\frac{P(y=1)}{P(y=0)}=\exp(\beta\ +\beta_1 x) \tag{23}$$

meaning that the odds is simply the linear function $\beta_0+\beta_1 x$. This model has two parameters (Anderson, et all, 2002).

2.15. Back Propagation

Many neural network algorithms aim to adjust the weights to get better results in the performance. The gradient is evaluated using the technique, back propagation. The back propagation updates network weights and biases in the direction of performance function that decreases more rapidly.

$$X_{k+1}=X_k-\alpha_k g_k \tag{24}$$

In equation (24), X_k is a vector of current weights and biases, g_k is the current gradient and α_k is the learning rate. The gradient is evaluated as

$$g=J^T e \tag{25}$$

In equation (25), g is gradient, *J* is jacobian matrix which has the derivatives of network errors with respect to weights and biases, e is a vector of network errors. And the hessian matrix is shown as

$$H=J^T J \tag{26}$$

Moreover, similar approach to evaluate weights and biases;

$$X_{k+1}=X_k-\left[J^T J+\mu I\right]^{-1} J^T e \tag{27}$$

When μ is zero, the gradient descent reduces with a small step size (Yan,et all, 2003).

2.16. Multilayer Perceptron

Multilayer Perceptron is a method which uses backpropagation to classify instances. Except for when the class is numeric, the nodes in this network are all sigmoid. In this case the output nodes become unthresholded linear units. The architecture of Multilayer Perceptron is given in Figure 3.

Perceptrons

A typical multilayer perceptron network consists of three or more layers of processing nodes; an **input layer** that receives external inputs, one or more **hidden layers,** and an **output layer**.

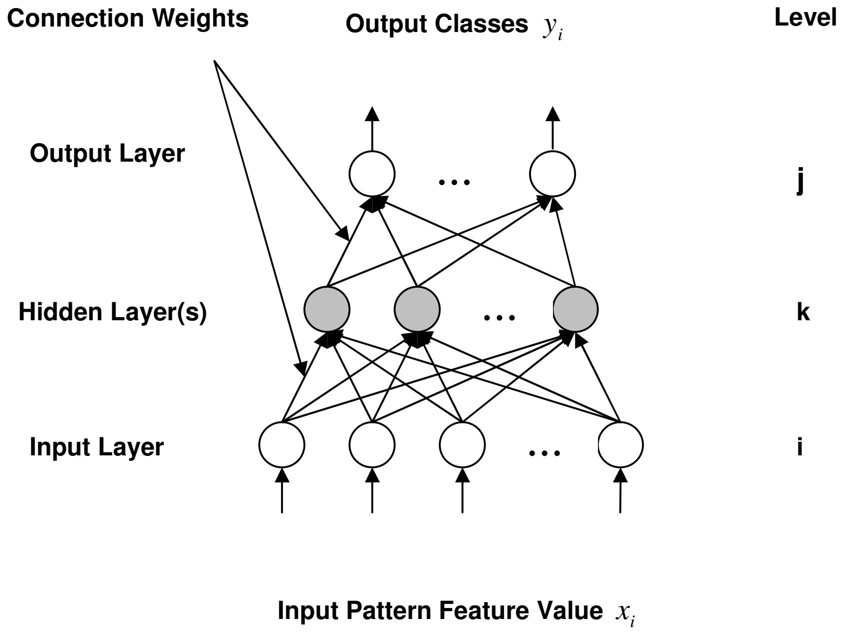


Figure 3. Architecture of a Multilayer Perceptron Network.

Process of Multilayer Perceptron

In the input layer, no process is done. When data are presented at the input layer, the network nodes perform calculations in the successive layers until an output value is obtained at each of the output nodes. This output signal should be able to indicate the appropriate class for the input data. That is, one can expect to have a high output value on the correct class node and low output values on all the rest. On Figure 4, a node in multilayer perceptron can be modeled as an artificial neuron which computes the weighted sum of the inputs at the presence of the bias, and passes this sum through the activation function as equation (28)

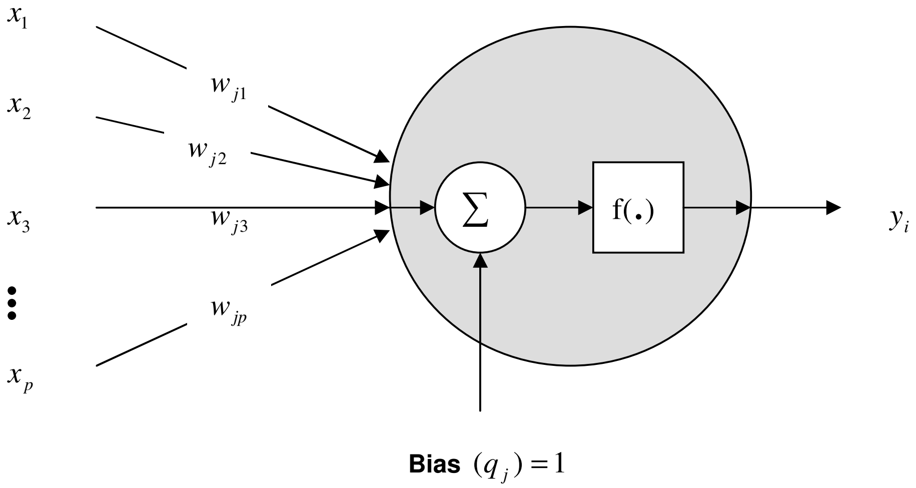


Figure 4. One Node of Multilayer Perceptron; an Artificial Neuron.

$$v_j = \sum_{i=1}^{p} w_{ji} x_i + \theta_j \qquad \mathrm{y_j} = f_j(v_j) \tag{28}$$

where v_j is the linear combination of inputs $x_1, x_2, ..., x_p$, θ_{j} is the bias, w_{ji} is the connection weight between the input x_i and the neuron j, and $f_j(v_j)$ is the activation function of the j_{th} neuron, and y_j is the output. The sigmoid function is a common choice of the activation function (Yan, et all, 2003);

$$f(a) = \frac{1}{1 + e^{-a}} \tag{29}$$

2.17. Radial Basis Function Neural Network (RBF NN)

Radial Basis Function Neural Network or RBF Neural Network is developed by using stochastic gradient descent method and a supervised clustering method. Unsupervised learning algorithms of RBF NN appear naturally suited. The structure of RBF Neural Networks is modular and can be easily implemented to hardware. RBF NN method is a normalized Gaussian **R**adial **B**asis **F**unction **N**etwork. RBF Network uses the k-means clustering algorithm to provide the basis functions and learns either a logistic regression for discrete class problems or linear regression for numeric class problems. RBF Network standardizes all numeric attributes to zero mean and unit variance. The RBF network has three layers: an input layer, a single layer for nonlinear processing of neurons and output layer (Tan, et all, 2001). In Figure 5, RBF Network with three layers is shown. For every input, every gaussian unit in hidden layer evaluates its output as a function of its center and width (variance).

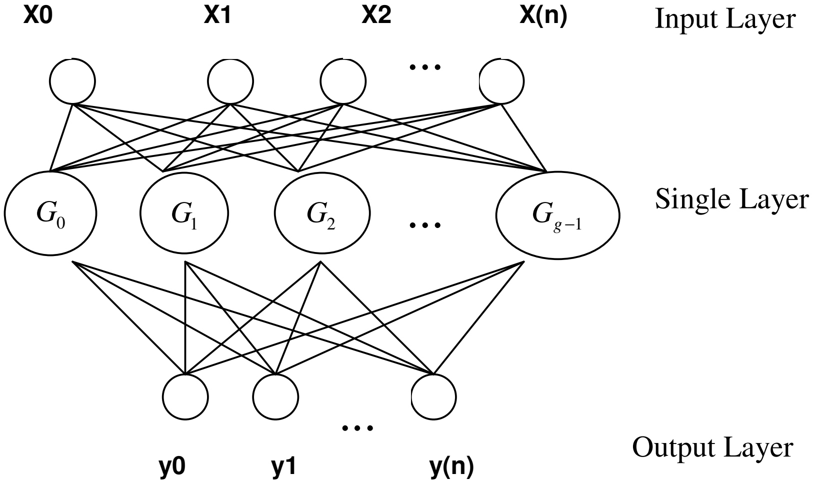


Figure 5. RBF Network with three Layers.

The input for RBF network is

$$G_j(x) = \frac{1}{\sqrt{2\pi b_j^{\,2}}} e^{-\frac{\|x - c_k\|}{2b^2{}_j}} \tag{30}$$

where x are inputs, $G_j(x)$ is the output of the j^{th} unit, c_j is the center of the unit j and $b_j^{\,2}$ is the variance of the j^{th} unit. The overall output is calculated as weighted sum of outputs of hidden units

$$\mathrm{y_i} = \mathrm{f_i(x)} = \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{w_{ik}} \varphi_{\mathrm{k}} (\mathrm{x}, \mathrm{c_k}) = \sum_{\mathrm{k=1}}^{\mathrm{N}} \mathrm{w_{ik}} \varphi_{\mathrm{k}} (\|\mathrm{x} - \mathrm{c_k}\|_2), \mathrm{i} = 1,2,\ldots,\mathrm{m} \tag{31}$$

where $x \in R^{n \times 1}$ is input vector, ϕ_k is the function from R^+ and $\|.\|_2$ denotes the euclidean norm, w_{ik} are the weights of output layer, N is the number of neurons in the hidden layer and $c_k \in R^{n \times 1}$ are RBF centers in input space. The process of RBF Network is given in Algorithm 11. The process starts by initializing the weights. For each neuron in hidden layer, the euclidean distance between its associated center and input to the network is computed. At last, the output of the network is computed as a weighted sum of hidden layer outputs. In addition, the weights are updated (Isa, et all, 2005).

Algorithm 11. The Process of RBF Network.

1) Initialize weigths,
2) For each neuron in hidden layer evaluate its euclidean distance,
3) Evaluate weighted sum of hidden layers,
4) Update weights,
5) Go to step 5.

3. Adaptive Neuro Fuzzy Inference System (ANFIS)

ANFIS stands for **A**daptive **N**euro **F**uzzy **I**nference **S**ystem. It is a fuzzy modeling procedure to learn about a data set and its aim is in order to compute the membership function parameters that best allow the corresponding fuzzy inference system to follow the given input - output data. Using a given input - output data set, ANFIS constructs a fuzzy inference system, FIS. Its membership function parameters are tuned using a backpropagation algorithm. This algorithm is single or with combination of least squares type of method. This allows fuzzy systems to learn the data they are modeling. In ANFIS, first, you hypothesize a parameterized model structure, relating inputs to membership functions, and to rules. Next, for training, you collect input - output data, that is in usable format by ANFIS. At last, the data which is delivered by modifying membership parameters, is trained by ANFIS. ANFIS is not available for all of the fuzzy inference system options. ANFIS only supports sugeno - type systems, having the following properties: First, it has the first order or zeroth order sugeno type systems. Second, it has a single output, obtained by weighted average defuzzification. All output functions must be in the same type or linear & constant. Third, it has no rule sharing. Different rules cannot share the same output membership function. And finally, ANFIS has unity weight for each rule. If any of these four rules is not accomplished, ANFIS gives errors. Similar to the neural network structure, a network structure mapping inputs to outputs through input membership functions and associated parameters, and then through output membership functions and associated parameters, can be used to implement the input - output map. A **gradient vector** measures how well the Fuzzy Inference System is modeling the input-output data for a set of parameters also reduces some error measure. ANFIS uses combination of least squares estimation or backpropagation for membership function parameter estimation. ANFIS can be expressed if – then rules like:

$$R_1 : IF\ x = A_1\ \&\ \mathrm{y} = \mathrm{B}_1\ \mathrm{THEN}\ \mathrm{z}_1 = f_1(x, y)$$
$$R_2 : IF\ \mathrm{x} = \mathrm{A}_2\ \&\ \mathrm{y} = \mathrm{B}_2\ \mathrm{THEN}\ \mathrm{z}_2 = f_2(x, y) \qquad (32)$$

where A and B are fuzzy sets, f_i is crisp function. The resulting network is shown in Figure 6. Each w_i is the output of each node in second layer. ANFIS algorithm is given in Algorithm 12 (Jang, et all, 1993), (Hernandez, et all, 2004), (Tang, et all, 2005).

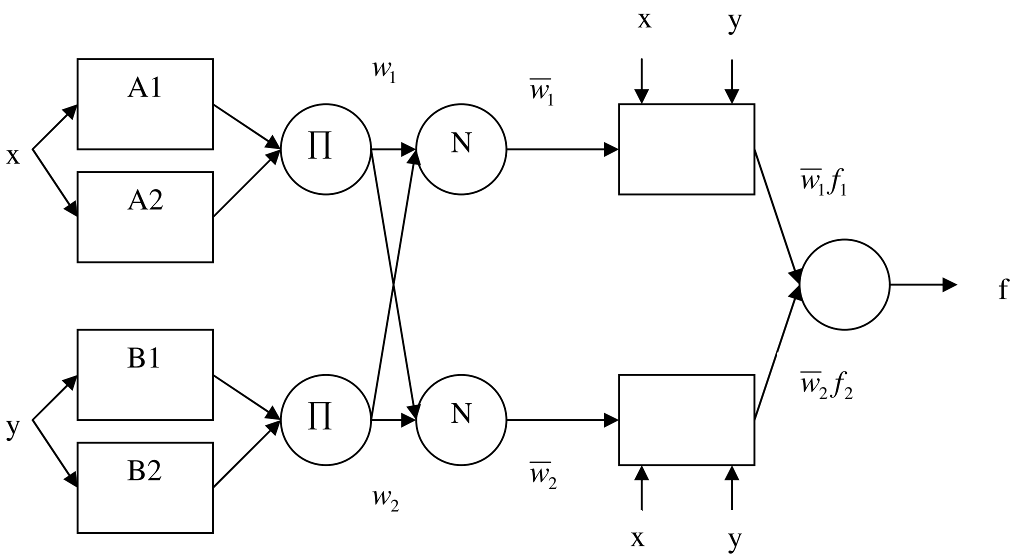


Figure 6. ANFIS Architecture.

Algorithm 12. ANFIS Algorithm.

1) Insert inputs to Fuzzy Procedure,

2) Multiply Incoming Signals,

3) Calculate Weights,

4) Apply Crisp Function to Weights,

5) Add final Weights.

What is Fuzzy?

Fuzzy means smoothed, piecewise and linear. Fuzzy Logic (FL) deals with complex and real systems. FL is a problem-solving methodology that implements itself ranging from simple, small, embedded microcontrollers to large, networked, multi channel computer based data acquisition and control systems. It can be implemented in hardware, software or both. FL provides a simple way to get a definite conclusion from uncertain, imprecise, noisy or missing input information. Fuzzy sets were first introduced into machine learning and data mining to facilitate the interpretation of rules in linguistic terms to avoid the partition instability and boundary bias problems. Fuzzy sets are represented as on conjunction and on implication operators (Serrurier, et all, 2007).

Neuro - Fuzzy System

Fuzzy logic and networks have been combined in many ways. Hybrid systems of neural networks and fuzzy logic are also called as fuzzy neural networks. Examples of these hybrid schemes are Fuzzy rule based systems with learning ability, Fuzzy rule-based systems represented by network architectures, Neural Networks for fuzzy reasoning and Fuzzified neural networks. The fuzzy rule based systems with learning ability are also called as neuro-fuzzy networks. Neural networks can learn from data but they cannot be explained. On the other hand, fuzzy systems consist of interpretable, explainable rules, however they could not learn. In order to identify data, we use learning algorithms to create fuzzy systems from data. Learning algorithms can learn fuzzy sets, fuzzy rules that leads Neuro-Fuzzy Systems (Tettey, et all, 2006).

Learning

The learning in the neural network is defined as the process that the input pattern is correctly mapped to corresponding desired output. The normalized actual output of the neural network at each training step can be written as a fuzzy set. We have supervised and unsupervised learning paradigms. In supervised learning, network has a kind of teacher that makes network aware about what is right or wrong considering input-output examples. This teacher feeds network with input and teaches the right output. In Unsupervised learning, network by itself recognizes its input and decides which neuron is to be trained, how they are trained. While the network is training, input is automatically classified. Network obtains the skill to decide the right output (Son, et all, 2007).

Model Validation

In order to see how well the FIS model predicts the data set, input vectors from input – output data sets are trained by FIS model. This whole process is called Model Validation. This is done by ANFIS GUI using testing data set. In the training, after a certain point, the model begins overfitting the data set. Until the overfitting begins, the model error of the checking data set tends to decrease. After that, model error of the checking data suddenly increases. In (Anderssen, et all, 2005), in order to validate variable selection procedure in ordinary least squares regression by using double loop process, a new model validation model called cross model validation is produced. In Algorithm 13 cross model validation procedure is shown.

Algorithm 13. Cross Model Validation.

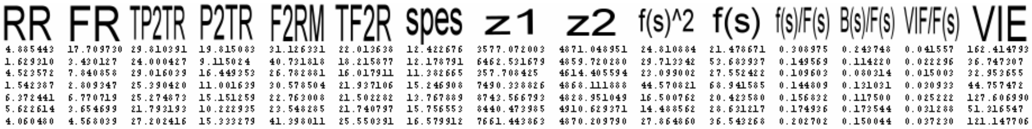
1) **Regression:** Estimate the parameters in a bi-linear regression model.
2) **Cross Validation and Jack-Knifing:** Estimate the optimal number of components for the regression, its predictive ability and its significance level of individual model parameters.
3) **Variable Selection**: From step 2 eliminate non-significant variables and recalculate the model using step 2 until variables that are significant at a chosen level remain in matrix X.
4) **Cross Model Validation**: Estimate the predictive ability of the optimized model by cross-model validation of the variable selection process.

4. METHODS

In the previous study (Aydin, et all, 2004) for the detection purpose of ES, fuzzy logic detection was used. After obtaining quadrature audio Doppler signals, discrete wavelet transform was applied to the Doppler signals. In order to characterize the signal type, fifteen types of parameters were obtained. Then, four of the threshold values which are different for each of the parameter were applied to the parameters. According to threshold values, each parameter was classified as AR, ES or DS signal. This is called Fuzzy Logic Detection. The detailed information about comparison result of fuzzy logic and other used DM techniques has been given in Conclusion section.

Preparing the Data Set

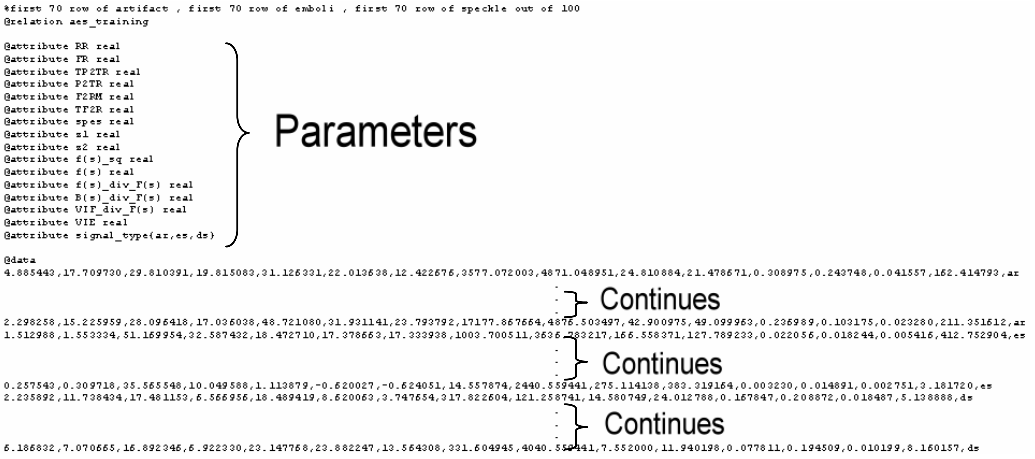
In this study, Data set 2 from the previous work has been used. Dedf_2 and Npar_2 sets have been combined (Aydin, et all, 2004). Some part of the combined data set has been shown in Figure 7. Each column in combined data set corresponds to a parameter.



RR	FR	TP2TR	P2TR	F2RM	TF2R	spes	z1	z2	f(s)^2	f(s)	f(s)/F(s)	B(s)/F(s)	VIF/F(s)	VIE
4.885443	17.709730	29.810391	19.815083	31.126331	22.013638	12.422676	3577.072003	4871.048951	24.810884	21.478671	0.308975	0.243748	0.041557	162.414793
1.629310	3.430127	24.000427	9.115024	40.731818	18.215877	12.178791	6462.531679	4859.720280	29.713342	53.683937	0.149569	0.114220	0.022296	36.747307
4.523572	7.840858	29.016039	16.449353	26.782881	16.017911	11.382665	357.708425	4614.405594	23.099002	27.552422	0.109603	0.080314	0.015003	32.953655
1.542387	2.809347	25.390420	11.001639	30.578504	21.937106	15.246908	7490.338826	4868.111888	44.570821	68.941585	0.144809	0.131031	0.030933	44.757472
6.372441	6.770719	25.274873	15.151259	22.763008	21.502282	13.767889	8743.566793	4828.951049	16.500762	20.423580	0.156832	0.117500	0.025222	127.606990
5.622614	3.654699	21.793193	10.222935	23.548285	21.740797	15.756553	8440.473985	4910.629371	14.488562	28.631217	0.174936	0.173544	0.031288	51.316547
4.060480	4.568039	27.202416	15.333279	41.398011	25.550391	16.579912	7661.443863	4870.209790	27.864860	36.543268	0.202702	0.150044	0.037230	121.147706

Figure 7. Data Set 2.

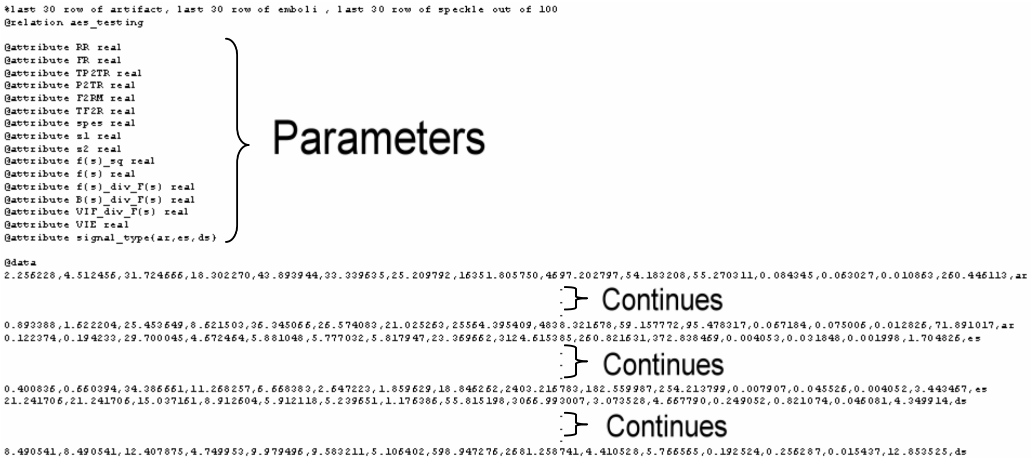
In the data set first 100 rows correspond to AR signal, next 100 rows correspond to ES and finally last 100 rows correspond to DS. In order to learn the data mining model and make comparison of the data mining efficiency and effectiveness, data set has been partitioned as 70% training set and 30% testing set. Data set has been converted to Weka data set format. The last row of new data set is one of AR, ES or DS which indicates that it is the output. The following image represents the Weka training data set called aes_training.arff. First 70 rows correspond to AR, next 70 rows correspond to ES and last 70 rows correspond to DS.



```
*first 70 row of artifact , first 70 row of emboli , first 70 row of speckle out of 100
@relation aes_training

@attribute RR real
@attribute FR real
@attribute TP2TR real
@attribute P2TR real
@attribute F2RM real
@attribute TF2R real
@attribute spes real
@attribute s1 real
@attribute s2 real
@attribute f(s)_sq real
@attribute f(s) real
@attribute f(s)_div_F(s) real
@attribute B(s)_div_F(s) real
@attribute VIF_div_F(s) real
@attribute VIE real
@attribute signal_type(ar,es,ds)

@data
4.885443,17.709730,29.810391,19.815083,31.126331,22.013638,12.422676,3577.072003,4871.048951,24.810884,21.478671,0.308975,0.243748,0.041557,162.414793,ar

2.298258,15.225959,28.096418,17.036038,48.721080,31.931141,23.793792,17177.867664,4876.503497,42.900975,49.099963,0.236989,0.103175,0.023280,211.351612,ar
1.512988,1.553334,51.169954,32.587432,18.472710,17.378663,17.333938,1003.700511,3636.783217,166.558371,127.789233,0.022056,0.018244,0.005416,412.752904,es

0.257543,0.309718,35.565548,10.049588,1.113879,-0.620027,-0.624051,14.557874,2440.559441,275.114138,383.319164,0.003230,0.014891,0.002751,3.181720,es
2.235892,11.738434,17.481153,6.566956,18.489419,8.620063,3.747654,317.822604,121.258741,14.580749,24.012788,0.167847,0.208872,0.018487,5.138888,ds

6.186832,7.070665,16.892346,6.922330,23.147768,23.882247,13.564308,331.604945,4040.559441,7.552000,11.940198,0.077811,0.194509,0.010199,8.160157,ds
```

Figure 8. Weka Training Data Set.

And following image represents the Weka testing data set called aes_testing.arff. First 30 rows correspond to AR, next 30 rows correspond to ES and last 30 rows correspond to DS.

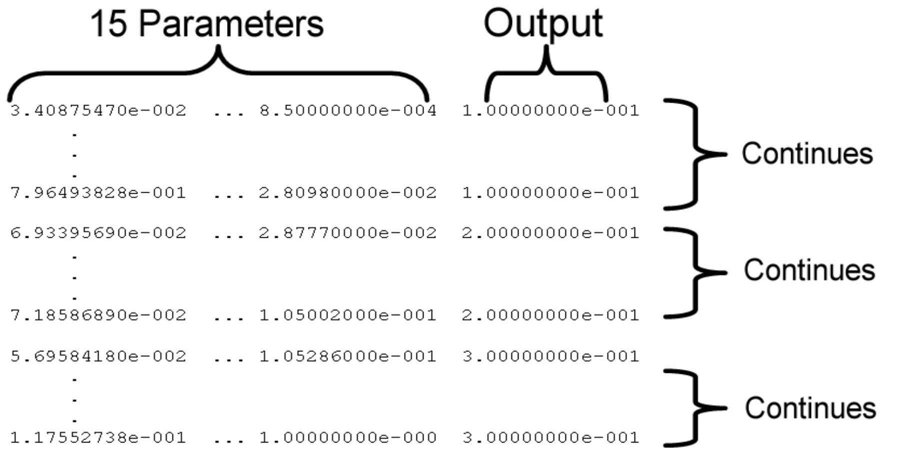


```
*last 30 row of artifact, last 30 row of emboli , last 30 row of speckle out of 100
@relation aes_testing

@attribute RR real
@attribute FR real
@attribute TP2TR real
@attribute P2TR real
@attribute F2RM real
@attribute TF2R real
@attribute spes real
@attribute s1 real
@attribute s2 real
@attribute f(s)_sq real
@attribute f(s) real
@attribute f(s)_div_F(s) real
@attribute B(s)_div_F(s) real
@attribute VIF_div_F(s) real
@attribute VIE real
@attribute signal_type(ar,es,ds)

@data
2.256228,4.512456,31.724666,18.302270,43.893944,33.339635,25.209792,16351.805750,4697.202797,54.183208,55.270311,0.084345,0.063027,0.010863,260.446113,ar

0.893388,1.622204,25.453649,8.621503,36.345066,26.574083,21.025263,25564.395409,4838.321678,59.157772,95.478317,0.067184,0.075006,0.012826,71.891017,ar
0.122374,0.194233,29.700045,4.672464,5.881048,5.777032,5.817947,23.369662,3124.615385,260.821631,372.838469,0.004053,0.031848,0.001998,1.704826,es

0.400836,0.660394,34.386661,11.268257,6.668383,2.647223,1.859629,18.846262,2403.216783,182.559987,254.213799,0.007907,0.045526,0.004052,3.443467,es
21.241706,21.241706,15.037161,8.912604,5.912118,5.239651,1.176386,55.815198,3066.993007,3.073528,4.667790,0.249052,0.821074,0.046081,4.349914,ds

8.490541,8.490541,12.407875,4.749953,9.979496,9.583211,5.106402,598.947276,2681.258741,4.410528,5.766565,0.192524,0.256287,0.015437,12.853525,ds
```

Figure 9. Weka Testing Data Set.

Next, sixteen data mining methods NBTree, LMT, Ridor, Nnge, VFI, Classification Via Regression, Ada Boost M1, Bagging, Dagging, Decorate, LWL, Simple Logistic, RBF Network, SMO, Multilayer Perceptron, and Naive Bayes have been trained for the detection purpose of three signals (AR, ES and DS). The training results in detail have been given in Results & Discussion section.

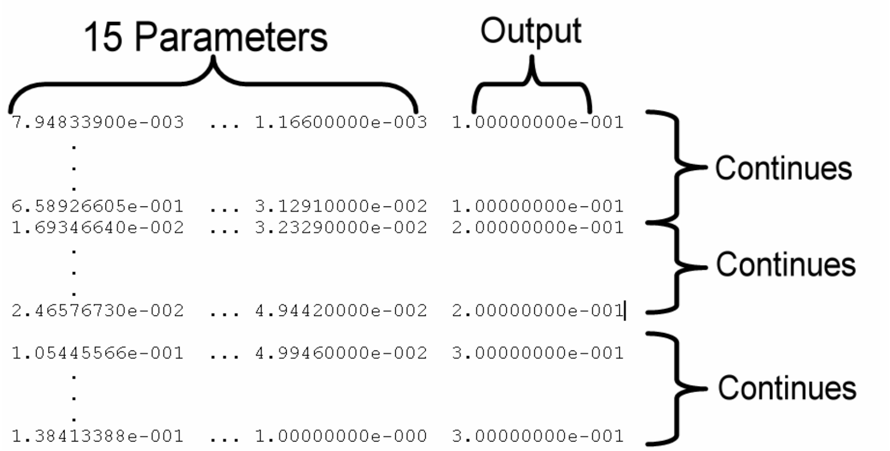
Afterward, data set 2 has been converted to matlab file format. The value 3.4e-001 indicates $3.4 * 10^{-1}$. As the output, 0.1 indicates AR, 0.2 indicates ES and 0.3 indicates DS. Again, data set has been split to training set and testing set. Training data set of ANFIS has been shown in Figure 10.


15 Parameters
Output
Continues
Continues
Continues

```
3.40875470e-002  ... 8.50000000e-004  1.00000000e-001
     .
     .
     .
7.96493828e-001  ... 2.80980000e-002  1.00000000e-001
6.93395690e-002  ... 2.87770000e-002  2.00000000e-001
     .
     .
     .
7.18586890e-002  ... 1.05002000e-001  2.00000000e-001
5.69584180e-002  ... 1.05286000e-001  3.00000000e-001
     .
     .
     .
1.17552738e-001  ... 1.00000000e-000  3.00000000e-001
```

Figure 10. ANFIS Training Data Set.

And testing data set of ANFIS has been shown in Figure 11.

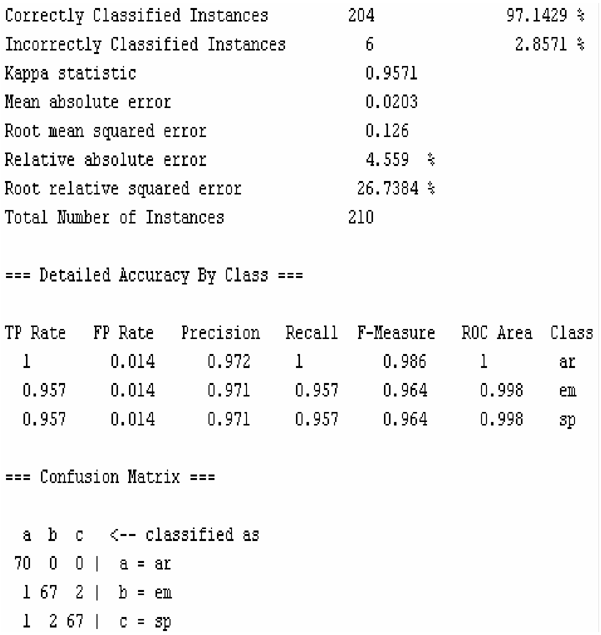

15 Parameters
Output
Continues
Continues
Continues

```
7.94833900e-003  ... 1.16600000e-003  1.00000000e-001
     .
     .
     .
6.58926605e-001  ... 3.12910000e-002  1.00000000e-001
1.69346640e-002  ... 3.23290000e-002  2.00000000e-001
     .
     .
     .
2.46576730e-002  ... 4.94420000e-002  2.00000000e-001
1.05445566e-001  ... 4.99460000e-002  3.00000000e-001
     .
     .
     .
1.38413388e-001  ... 1.00000000e-000  3.00000000e-001
```

Figure 11. ANFIS Testing Data Set.

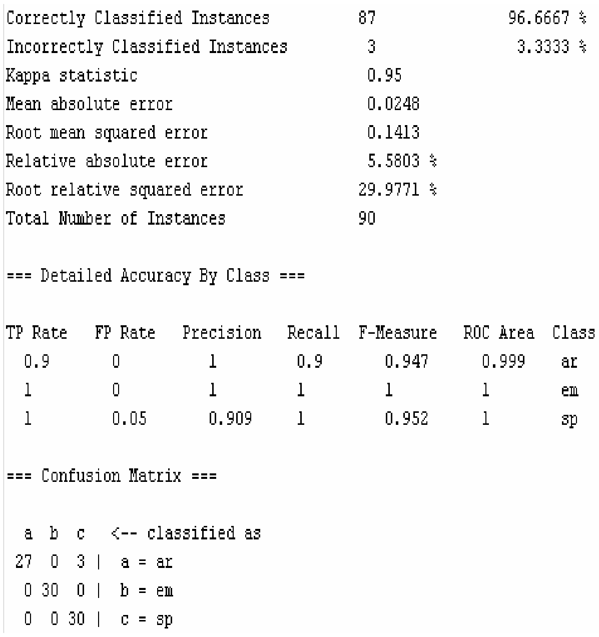
Screenshots of Weka Methods and ANFIS

Below, the screenshots of Weka have been given. Generally, for each method, the performance of normal result is better than the result of testing set. The test has been performed as follows. After opening Weka, aes_training.arff has been loaded. Then, in the classify tab, normal tests have been performed by selecting “Use training set”. After that, testing set has been loaded by clicking “Suppied test set”. Next, aes_testing.arff has been selected as testing set. Then, test with the testing set has been performed. Weka has been evaluated TP Rate, FP Rate, RMSE and Correctness values. Testing specificity has been considered as the detection result of three signals. By using formula, precision value has been evaluated. The screenshot of Naive Bayes method is given as follows.



```
Correctly Classified Instances         204               97.1429 %
Incorrectly Classified Instances         6                2.8571 %
Kappa statistic                          0.9571
Mean absolute error                      0.0203
Root mean squared error                  0.126
Relative absolute error                  4.559  %
Root relative squared error             26.7384 %
Total Number of Instances              210

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  1         0.014     0.972     1         0.986     1          ar
  0.957     0.014     0.971     0.957     0.964     0.998      em
  0.957     0.014     0.971     0.957     0.964     0.998      sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 70  0  0 |  a = ar
  1 67  2 |  b = em
  1  2 67 |  c = sp
```

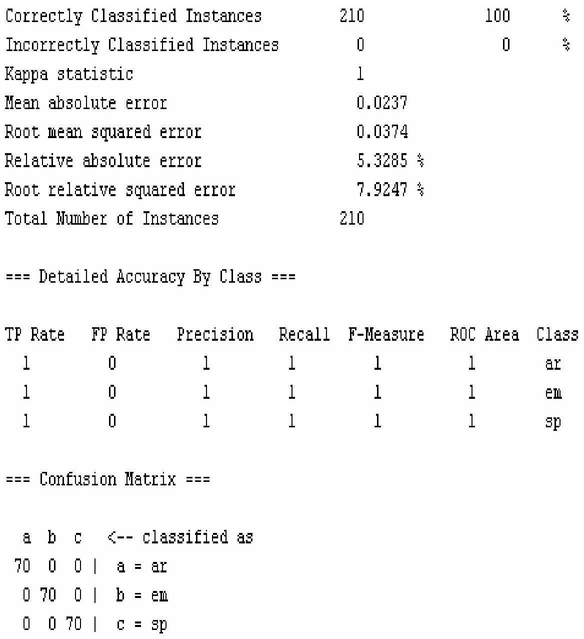
Figure 12. Naive Bayes Screenshot.



```
Correctly Classified Instances          87               96.6667 %
Incorrectly Classified Instances         3                3.3333 %
Kappa statistic                          0.95
Mean absolute error                      0.0248
Root mean squared error                  0.1413
Relative absolute error                  5.5803 %
Root relative squared error             29.9771 %
Total Number of Instances               90

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.9       0         1         0.9       0.947     0.999      ar
  1         0         1         1         1         1          em
  1         0.05      0.909     1         0.952     1          sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 27  0  3 |  a = ar
  0 30  0 |  b = em
  0  0 30 |  c = sp
```

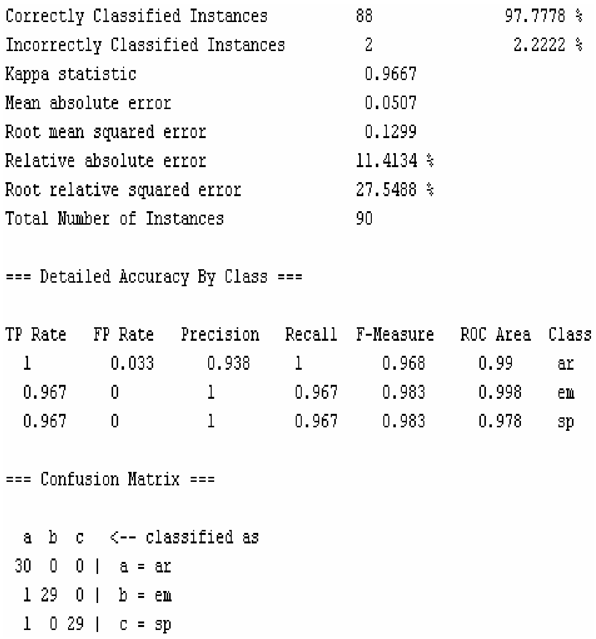
Figure 13. Naive Bayes with Testing Set.

The screenshot of NBTree is given below.



```
Correctly Classified Instances         210              100      %
Incorrectly Classified Instances         0                0      %
Kappa statistic                          1
Mean absolute error                      0.0237
Root mean squared error                  0.0374
Relative absolute error                  5.3285 %
Root relative squared error              7.9247 %
Total Number of Instances              210

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  1         0          1          1         1          1        ar
  1         0          1          1         1          1        em
  1         0          1          1         1          1        sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 70  0  0 |  a = ar
  0 70  0 |  b = em
  0  0 70 |  c = sp
```

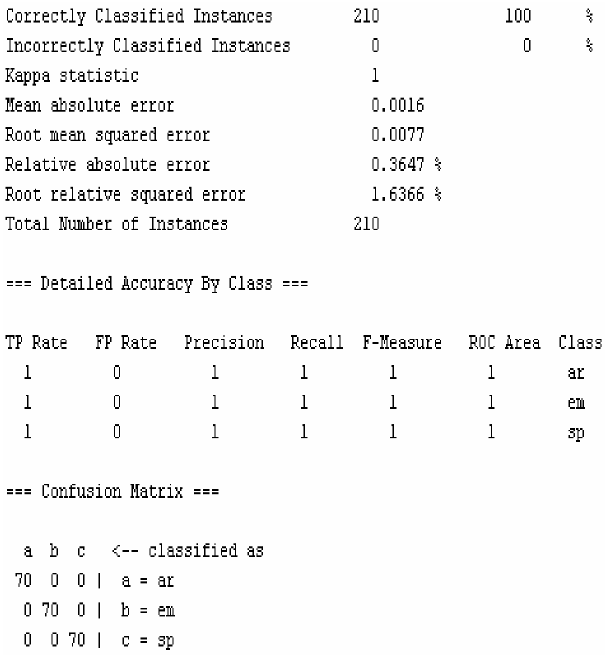
Figure 14. NBTree Screenshot.



```
Correctly Classified Instances          88               97.7778 %
Incorrectly Classified Instances         2                2.2222 %
Kappa statistic                          0.9667
Mean absolute error                      0.0507
Root mean squared error                  0.1299
Relative absolute error                 11.4134 %
Root relative squared error             27.5488 %
Total Number of Instances               90

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  1         0.033      0.938      1         0.968      0.99     ar
  0.967     0          1          0.967     0.983      0.998    em
  0.967     0          1          0.967     0.983      0.978    sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 30  0  0 |  a = ar
  1 29  0 |  b = em
  1  0 29 |  c = sp
```

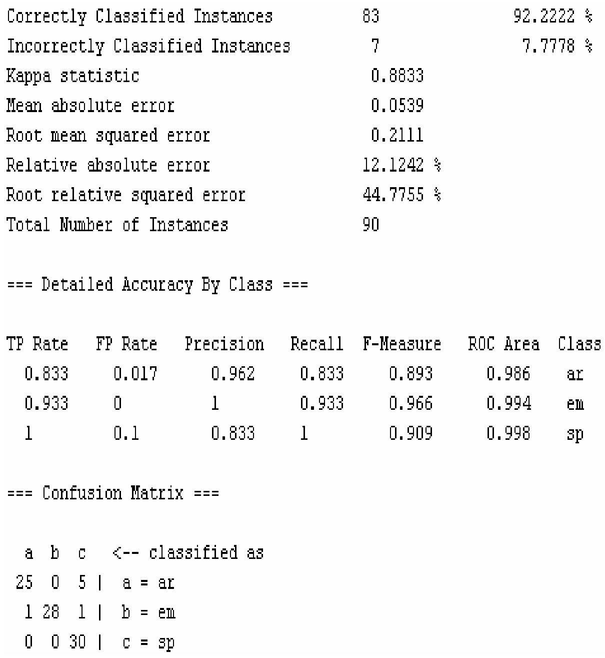
Figure 15. NBTree with Testing Set.

The screenshot of LMT is given as follows.



```
Correctly Classified Instances         210              100      %
Incorrectly Classified Instances         0                0      %
Kappa statistic                          1
Mean absolute error                      0.0016
Root mean squared error                  0.0077
Relative absolute error                  0.3647 %
Root relative squared error              1.6366 %
Total Number of Instances              210

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  1         0          1          1         1          1        ar
  1         0          1          1         1          1        em
  1         0          1          1         1          1        sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 70  0  0 |  a = ar
  0 70  0 |  b = em
  0  0 70 |  c = sp
```

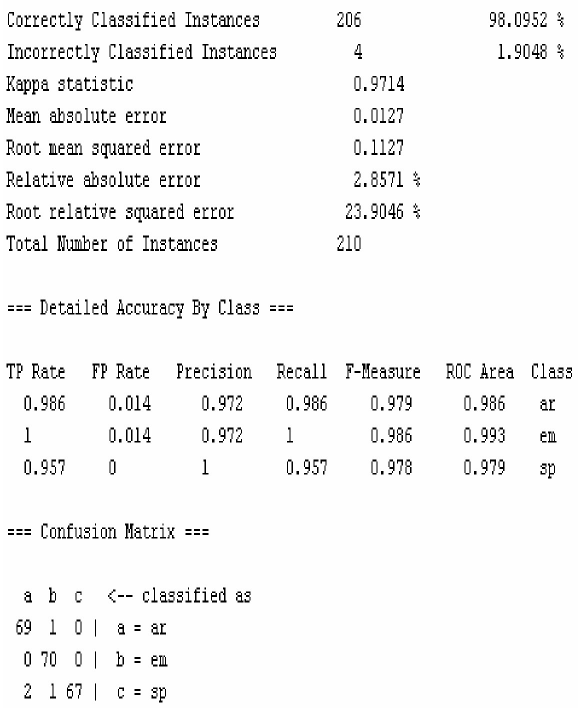
Figure 16. LMT Screenshot.



```
Correctly Classified Instances          83               92.2222 %
Incorrectly Classified Instances         7                7.7778 %
Kappa statistic                          0.8833
Mean absolute error                      0.0539
Root mean squared error                  0.2111
Relative absolute error                 12.1242 %
Root relative squared error             44.7755 %
Total Number of Instances               90

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.833     0.017      0.962      0.833     0.893      0.986    ar
  0.933     0          1          0.933     0.966      0.994    em
  1         0.1        0.833      1         0.909      0.998    sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 25  0  5 |  a = ar
  1 28  1 |  b = em
  0  0 30 |  c = sp
```

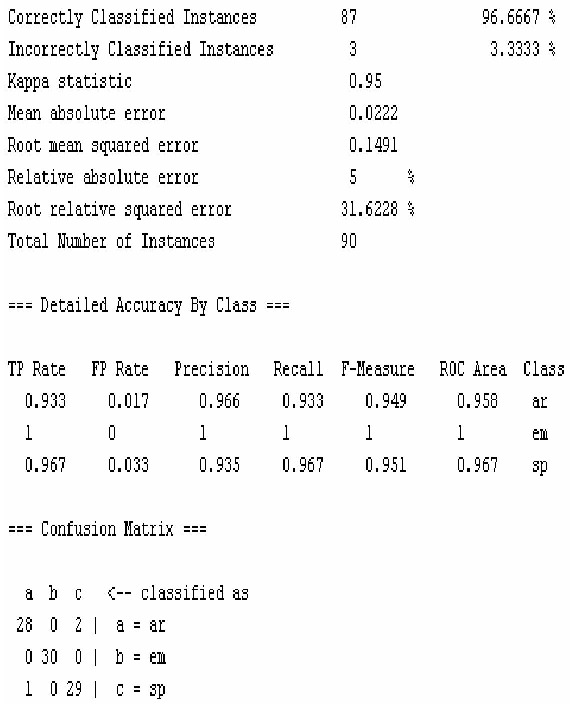
Figure 17. LMT with Testing Set.

The screenshot of Ridor is given below.



```
Correctly Classified Instances         206               98.0952 %
Incorrectly Classified Instances         4                1.9048 %
Kappa statistic                          0.9714
Mean absolute error                      0.0127
Root mean squared error                  0.1127
Relative absolute error                  2.8571 %
Root relative squared error             23.9046 %
Total Number of Instances              210

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.986     0.014      0.972     0.986     0.979      0.986    ar
  1         0.014      0.972     1         0.986      0.993    em
  0.957     0            1         0.957     0.978      0.979    sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 69  1  0 |  a = ar
  0 70  0 |  b = em
  2  1 67 |  c = sp
```

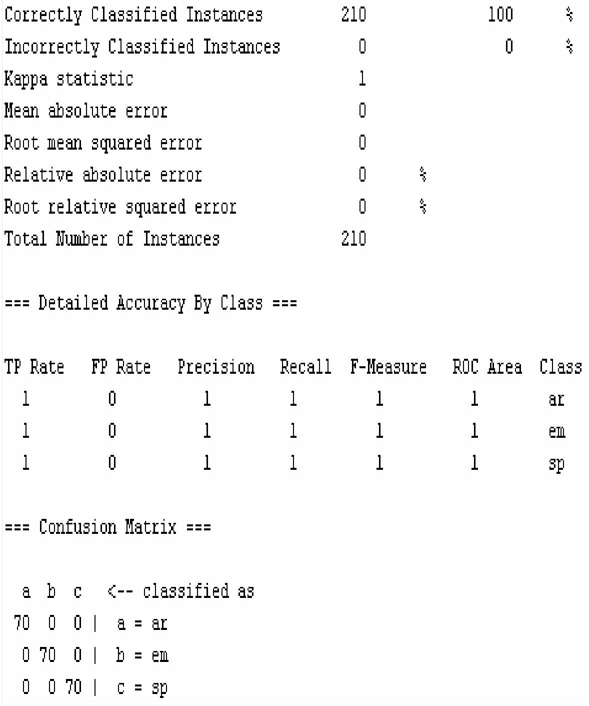
Figure 18. Ridor Screenshot.



```
Correctly Classified Instances          87               96.6667 %
Incorrectly Classified Instances         3                3.3333 %
Kappa statistic                          0.95
Mean absolute error                      0.0222
Root mean squared error                  0.1491
Relative absolute error                  5      %
Root relative squared error             31.6228 %
Total Number of Instances               90

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.933     0.017      0.966     0.933     0.949      0.958    ar
  1         0          1         1         1          1        em
  0.967     0.033      0.935     0.967     0.951      0.967    sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 28  0  2 |  a = ar
  0 30  0 |  b = em
  1  0 29 |  c = sp
```

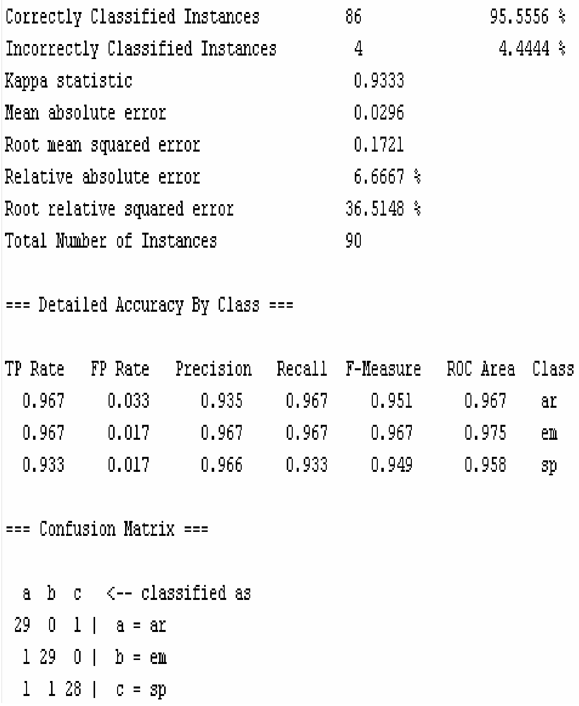
Figure 19. Ridor with Testing Set.

The screenshot of Nnge is given as follows.



```
Correctly Classified Instances         210              100      %
Incorrectly Classified Instances         0                0      %
Kappa statistic                          1
Mean absolute error                      0
Root mean squared error                  0
Relative absolute error                  0      %
Root relative squared error              0      %
Total Number of Instances              210

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  1         0          1         1         1          1        ar
  1         0          1         1         1          1        em
  1         0          1         1         1          1        sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 70  0  0 |  a = ar
  0 70  0 |  b = em
  0  0 70 |  c = sp
```

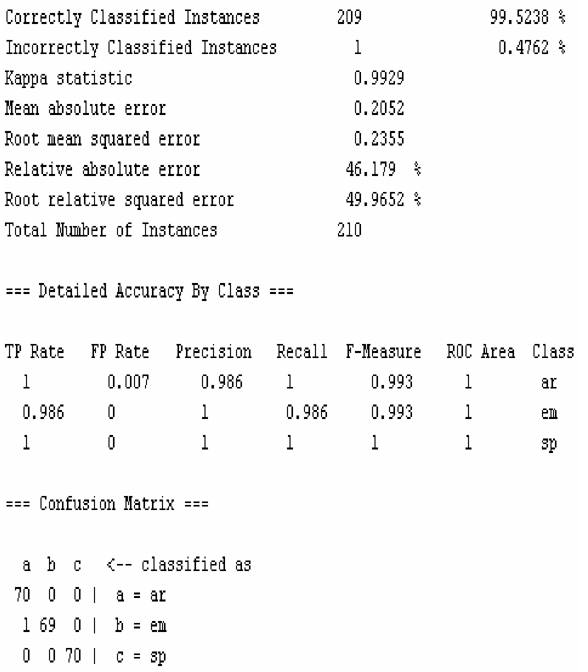
Figure 20. Nnge Screenshot.



```
Correctly Classified Instances          86               95.5556 %
Incorrectly Classified Instances         4                4.4444 %
Kappa statistic                          0.9333
Mean absolute error                      0.0296
Root mean squared error                  0.1721
Relative absolute error                  6.6667 %
Root relative squared error             36.5148 %
Total Number of Instances               90

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.967     0.033      0.935     0.967     0.951      0.967    ar
  0.967     0.017      0.967     0.967     0.967      0.975    em
  0.933     0.017      0.966     0.933     0.949      0.958    sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 29  0  1 |  a = ar
  1 29  0 |  b = em
  1  1 28 |  c = sp
```

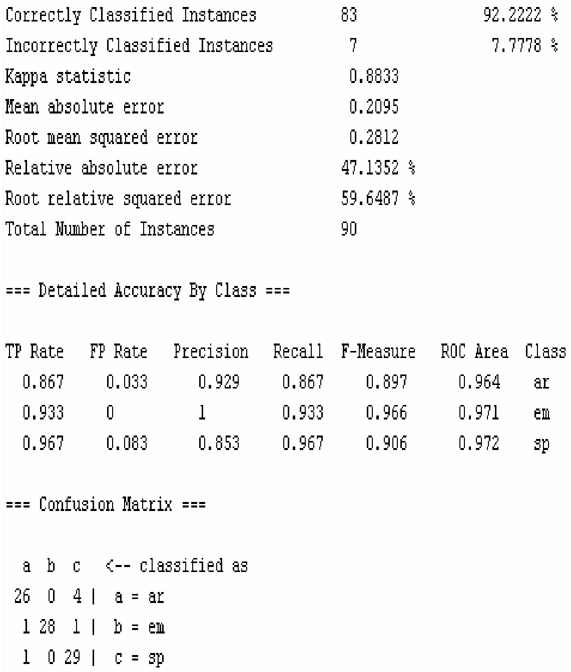
Figure 21. Nnge with Testing Set.

The screenshot of VFI is given below.



```
Correctly Classified Instances         209               99.5238 %
Incorrectly Classified Instances         1                0.4762 %
Kappa statistic                          0.9929
Mean absolute error                      0.2052
Root mean squared error                  0.2355
Relative absolute error                 46.179  %
Root relative squared error             49.9652 %
Total Number of Instances              210

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  1         0.007     0.986     1         0.993       1        ar
  0.986     0         1         0.986     0.993       1        em
  1         0         1         1         1           1        sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 70  0  0 |  a = ar
  1 69  0 |  b = em
  0  0 70 |  c = sp
```

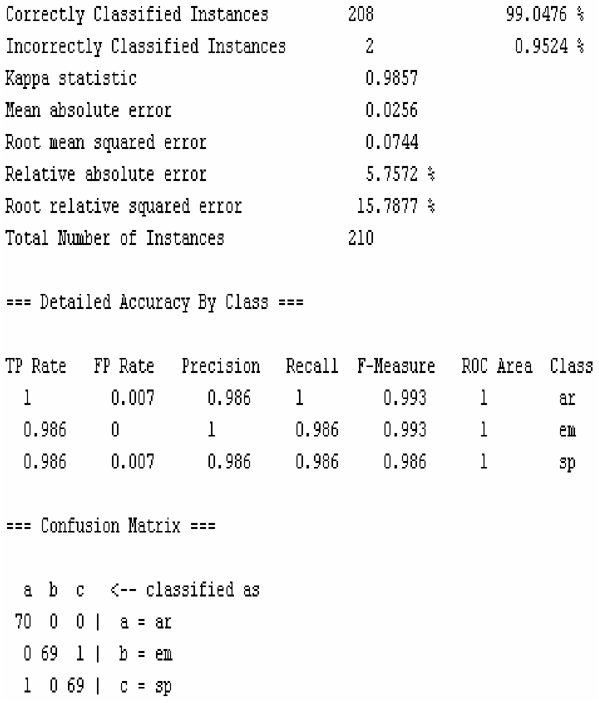
Figure 22. VFI Screenshot.



```
Correctly Classified Instances          83               92.2222 %
Incorrectly Classified Instances         7                7.7778 %
Kappa statistic                          0.8833
Mean absolute error                      0.2095
Root mean squared error                  0.2812
Relative absolute error                 47.1352 %
Root relative squared error             59.6487 %
Total Number of Instances               90

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.867     0.033     0.929     0.867     0.897       0.964    ar
  0.933     0         1         0.933     0.966       0.971    em
  0.967     0.083     0.853     0.967     0.906       0.972    sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 26  0  4 |  a = ar
  1 28  1 |  b = em
  1  0 29 |  c = sp
```

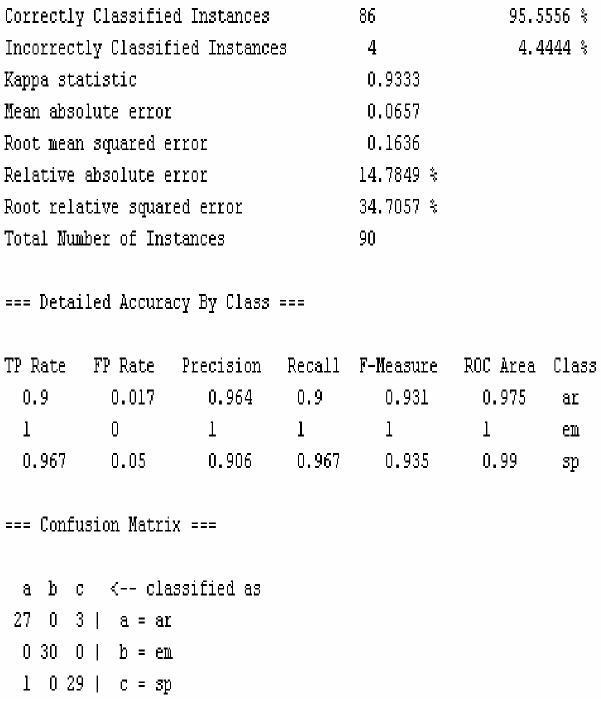
Figure 23. VFI with Testing Set.

The screenshot of Bagging is given as follows.



```
Correctly Classified Instances         208               99.0476 %
Incorrectly Classified Instances         2                0.9524 %
Kappa statistic                          0.9857
Mean absolute error                      0.0256
Root mean squared error                  0.0744
Relative absolute error                  5.7572 %
Root relative squared error             15.7877 %
Total Number of Instances              210

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  1         0.007     0.986     1         0.993       1        ar
  0.986     0         1         0.986     0.993       1        em
  0.986     0.007     0.986     0.986     0.986       1        sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 70  0  0 |  a = ar
  0 69  1 |  b = em
  1  0 69 |  c = sp
```

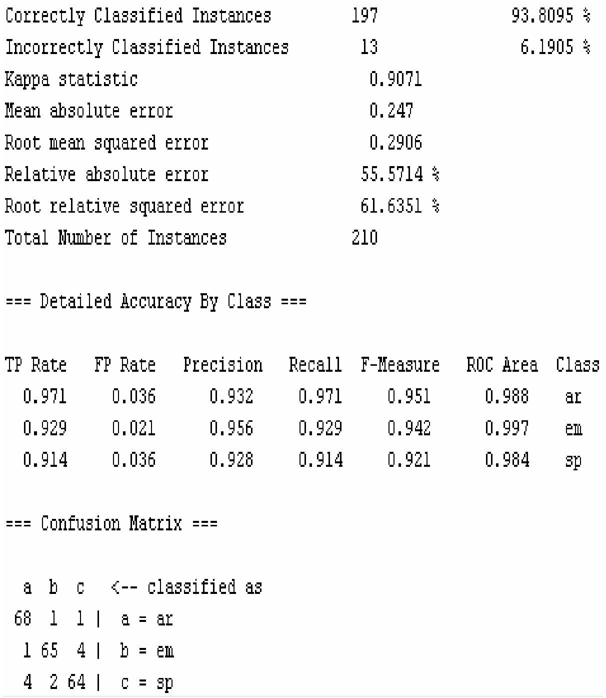
Figure 24. Bagging Screenshot.



```
Correctly Classified Instances          86               95.5556 %
Incorrectly Classified Instances         4                4.4444 %
Kappa statistic                          0.9333
Mean absolute error                      0.0657
Root mean squared error                  0.1636
Relative absolute error                 14.7849 %
Root relative squared error             34.7057 %
Total Number of Instances               90

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.9       0.017     0.964     0.9       0.931       0.975    ar
  1         0         1         1         1           1        em
  0.967     0.05      0.906     0.967     0.935       0.99     sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 27  0  3 |  a = ar
  0 30  0 |  b = em
  1  0 29 |  c = sp
```

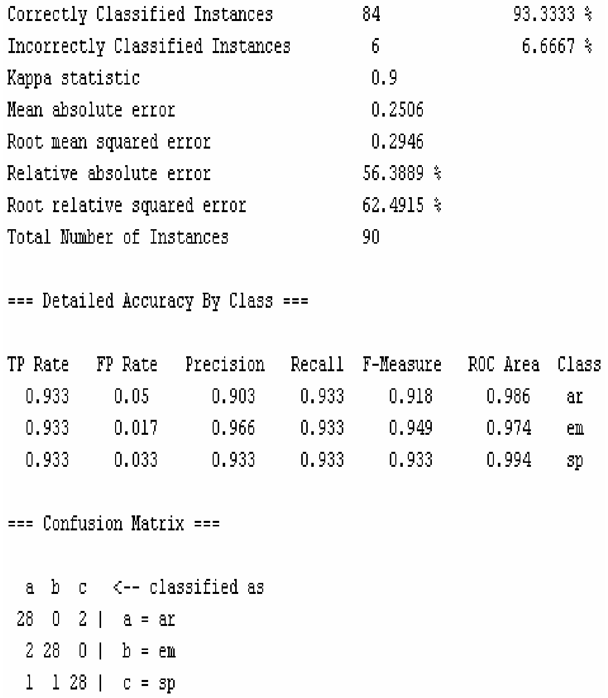
Figure 25. Bagging with Testing Set.

The screenshot of Dagging is given below.



```
Correctly Classified Instances         197               93.8095 %
Incorrectly Classified Instances        13                6.1905 %
Kappa statistic                          0.9071
Mean absolute error                      0.247
Root mean squared error                  0.2906
Relative absolute error                 55.5714 %
Root relative squared error             61.6351 %
Total Number of Instances              210

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.971     0.036      0.932     0.971     0.951      0.988    ar
  0.929     0.021      0.956     0.929     0.942      0.997    em
  0.914     0.036      0.928     0.914     0.921      0.984    sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 68  1  1 |  a = ar
  1 65  4 |  b = em
  4  2 64 |  c = sp
```

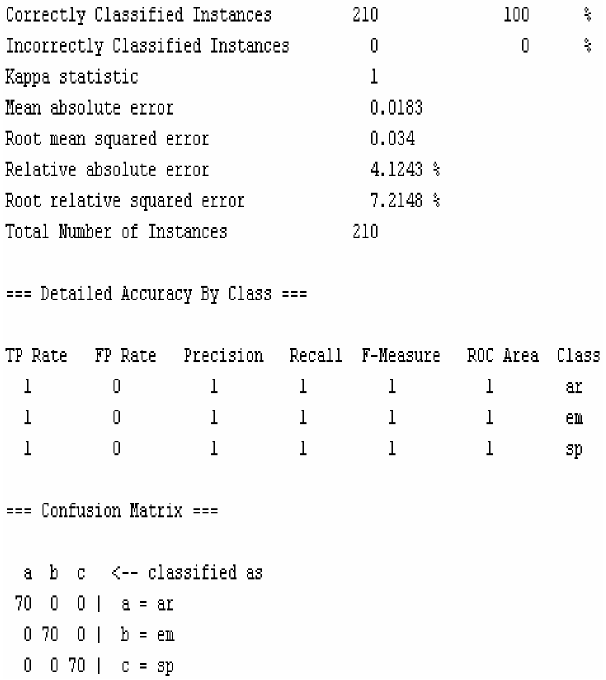
Figure 26. Dagging Screenshot.



```
Correctly Classified Instances          84               93.3333 %
Incorrectly Classified Instances         6                6.6667 %
Kappa statistic                          0.9
Mean absolute error                      0.2506
Root mean squared error                  0.2946
Relative absolute error                 56.3889 %
Root relative squared error             62.4915 %
Total Number of Instances               90

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.933     0.05       0.903     0.933     0.918      0.986    ar
  0.933     0.017      0.966     0.933     0.949      0.974    em
  0.933     0.033      0.933     0.933     0.933      0.994    sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 28  0  2 |  a = ar
  2 28  0 |  b = em
  1  1 28 |  c = sp
```

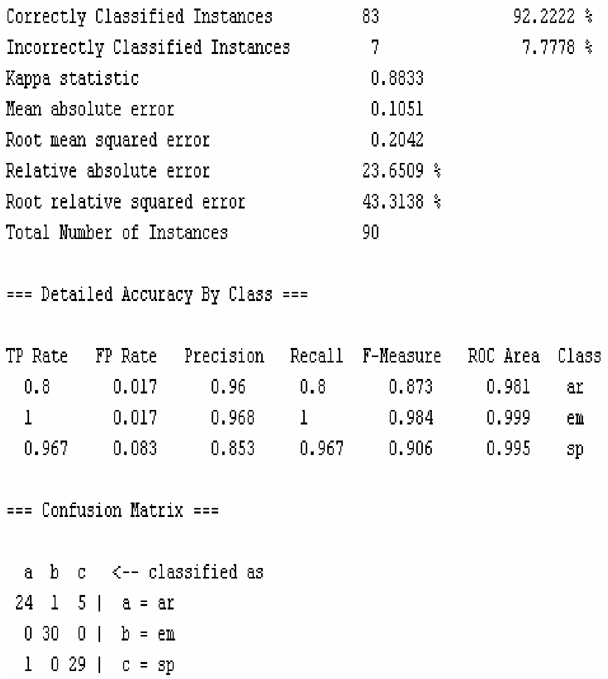
Figure 27. Dagging with Testing Set.

The screenshot of Decorate is given as follows.



```
Correctly Classified Instances         210              100      %
Incorrectly Classified Instances         0                0      %
Kappa statistic                          1
Mean absolute error                      0.0183
Root mean squared error                  0.034
Relative absolute error                  4.1243 %
Root relative squared error              7.2148 %
Total Number of Instances              210

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  1         0          1         1         1          1        ar
  1         0          1         1         1          1        em
  1         0          1         1         1          1        sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 70  0  0 |  a = ar
  0 70  0 |  b = em
  0  0 70 |  c = sp
```

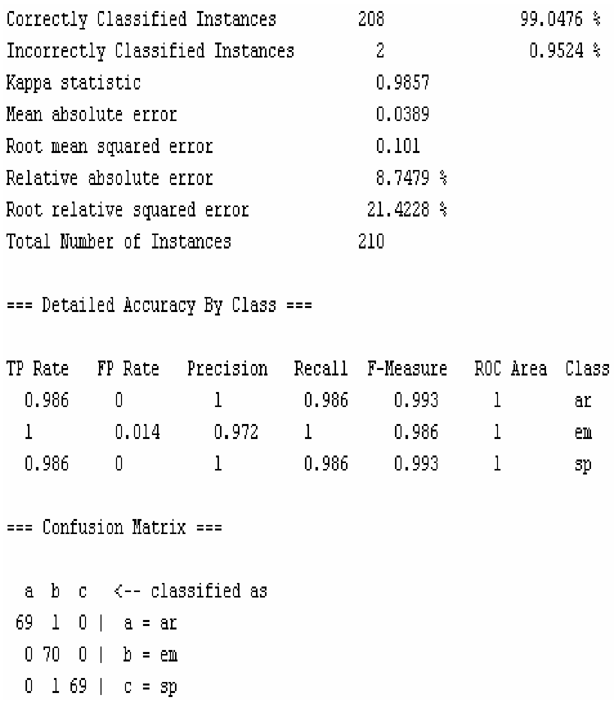
Figure 28. Decorate Screenshot.



```
Correctly Classified Instances          83               92.2222 %
Incorrectly Classified Instances         7                7.7778 %
Kappa statistic                          0.8833
Mean absolute error                      0.1051
Root mean squared error                  0.2042
Relative absolute error                 23.6509 %
Root relative squared error             43.3138 %
Total Number of Instances               90

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.8       0.017      0.96      0.8       0.873      0.981    ar
  1         0.017      0.968     1         0.984      0.999    em
  0.967     0.083      0.853     0.967     0.906      0.995    sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 24  1  5 |  a = ar
  0 30  0 |  b = em
  1  0 29 |  c = sp
```

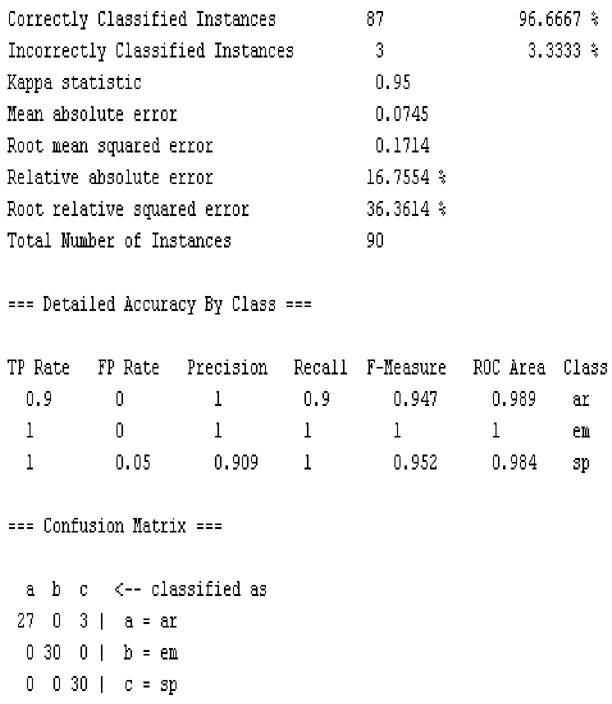
Figure 29. Decorate with Testing Set.

The screenshot of Ada Boost M1 is given below.



```
Correctly Classified Instances         208               99.0476 %
Incorrectly Classified Instances         2                0.9524 %
Kappa statistic                          0.9857
Mean absolute error                      0.0389
Root mean squared error                  0.101
Relative absolute error                  8.7479 %
Root relative squared error             21.4228 %
Total Number of Instances              210

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.986     0         1          0.986     0.993      1         ar
  1         0.014     0.972      1         0.986      1         em
  0.986     0         1          0.986     0.993      1         sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 69  1  0 |  a = ar
  0 70  0 |  b = em
  0  1 69 |  c = sp
```

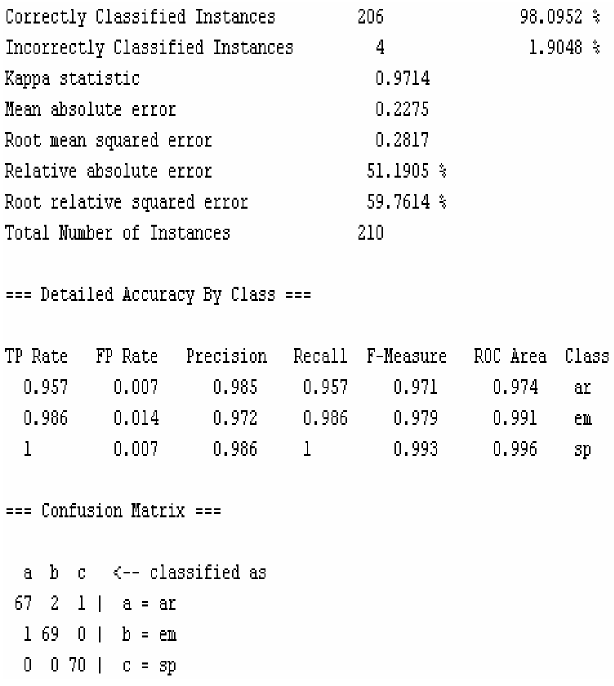
Figure 30. Ada Boost M1 Screenshot.



```
Correctly Classified Instances          87               96.6667 %
Incorrectly Classified Instances         3                3.3333 %
Kappa statistic                          0.95
Mean absolute error                      0.0745
Root mean squared error                  0.1714
Relative absolute error                 16.7554 %
Root relative squared error             36.3614 %
Total Number of Instances               90

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.9       0         1          0.9       0.947      0.989     ar
  1         0         1          1         1          1         em
  1         0.05      0.909      1         0.952      0.984     sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 27  0  3 |  a = ar
  0 30  0 |  b = em
  0  0 30 |  c = sp
```

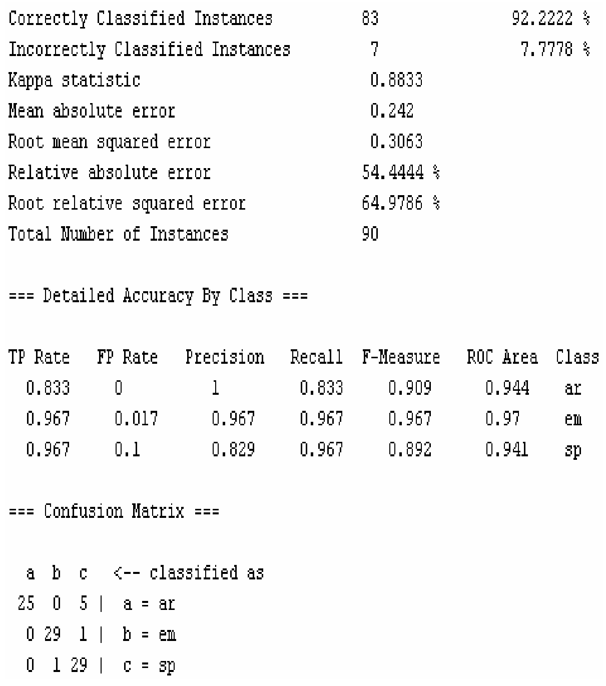
Figure 31. Ada Boost M1 with Testing Set.

The screenshot of SMO is given as follows.



```
Correctly Classified Instances         206               98.0952 %
Incorrectly Classified Instances         4                1.9048 %
Kappa statistic                          0.9714
Mean absolute error                      0.2275
Root mean squared error                  0.2817
Relative absolute error                 51.1905 %
Root relative squared error             59.7614 %
Total Number of Instances              210

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.957     0.007     0.985      0.957     0.971      0.974     ar
  0.986     0.014     0.972      0.986     0.979      0.991     em
  1         0.007     0.986      1         0.993      0.996     sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 67  2  1 |  a = ar
  1 69  0 |  b = em
  0  0 70 |  c = sp
```

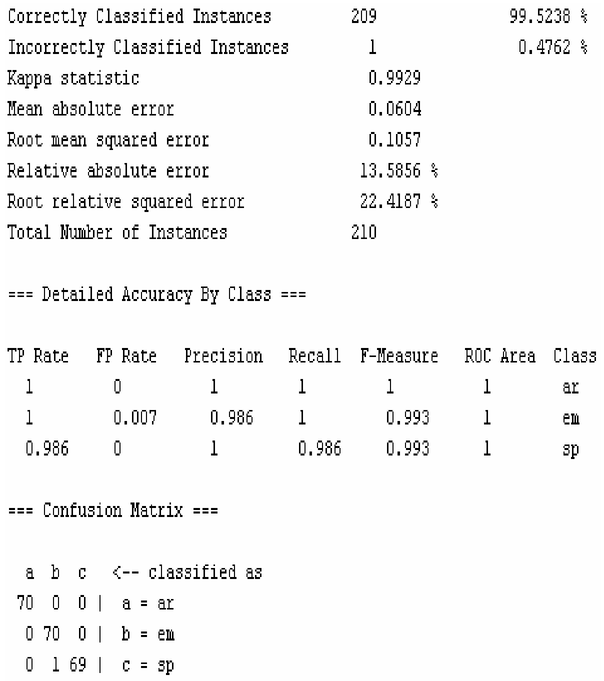
Figure 32. SMO Screenshot.



```
Correctly Classified Instances          83               92.2222 %
Incorrectly Classified Instances         7                7.7778 %
Kappa statistic                          0.8833
Mean absolute error                      0.242
Root mean squared error                  0.3063
Relative absolute error                 54.4444 %
Root relative squared error             64.9786 %
Total Number of Instances               90

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.833     0         1          0.833     0.909      0.944     ar
  0.967     0.017     0.967      0.967     0.967      0.97      em
  0.967     0.1       0.829      0.967     0.892      0.941     sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 25  0  5 |  a = ar
  0 29  1 |  b = em
  0  1 29 |  c = sp
```

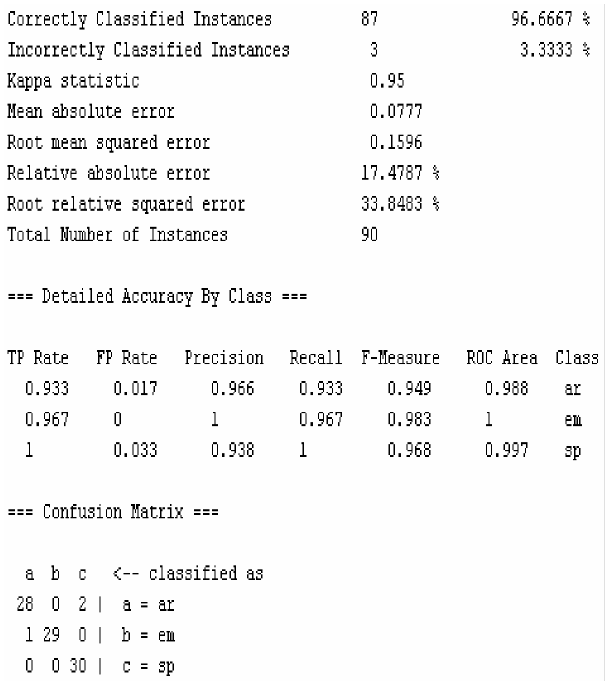
Figure 33. SMO with Testing Set.

The screenshot of Classification via Regression is given below.



```
Correctly Classified Instances         209               99.5238 %
Incorrectly Classified Instances         1                0.4762 %
Kappa statistic                          0.9929
Mean absolute error                      0.0604
Root mean squared error                  0.1057
Relative absolute error                 13.5856 %
Root relative squared error             22.4187 %
Total Number of Instances              210

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  1         0         1         1         1           1        ar
  1         0.007     0.986     1         0.993       1        em
  0.986     0         1         0.986     0.993       1        sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 70  0  0 |  a = ar
  0 70  0 |  b = em
  0  1 69 |  c = sp
```

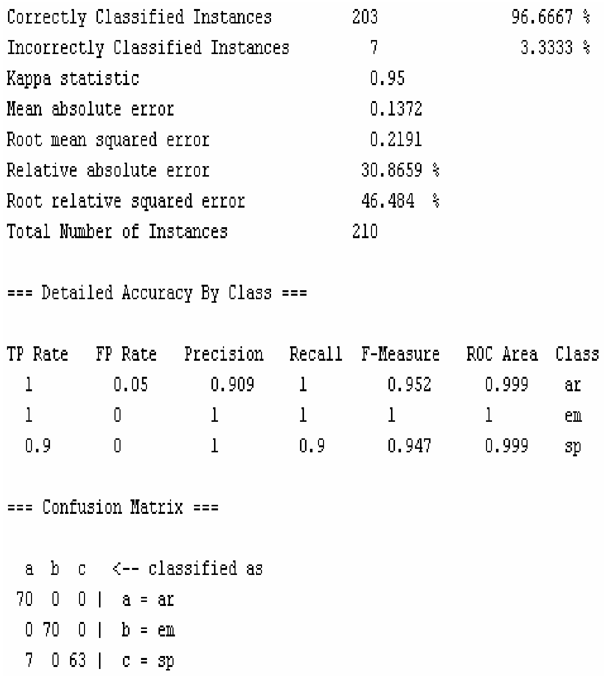
Figure 34. Classification Via Regression Screenshot.



```
Correctly Classified Instances          87               96.6667 %
Incorrectly Classified Instances         3                3.3333 %
Kappa statistic                          0.95
Mean absolute error                      0.0777
Root mean squared error                  0.1596
Relative absolute error                 17.4787 %
Root relative squared error             33.8483 %
Total Number of Instances               90

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.933     0.017     0.966     0.933     0.949       0.988    ar
  0.967     0         1         0.967     0.983       1        em
  1         0.033     0.938     1         0.968       0.997    sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 28  0  2 |  a = ar
  1 29  0 |  b = em
  0  0 30 |  c = sp
```

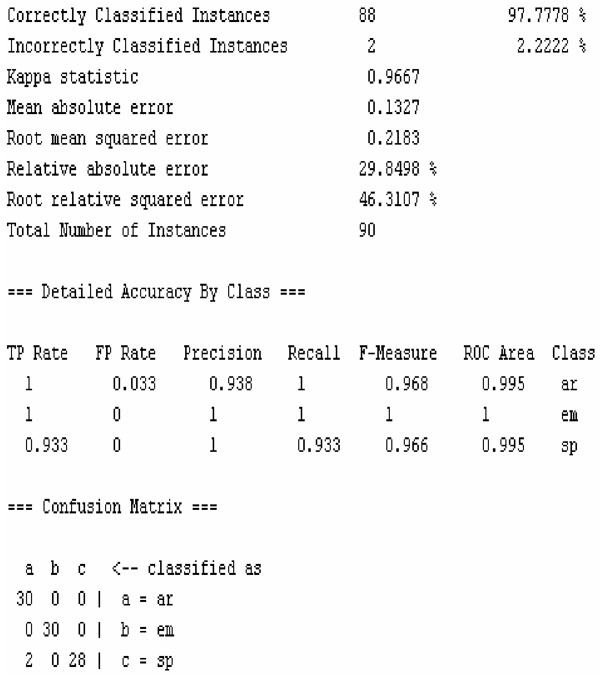
Figure 35. Classification Via Regression with Testing Set.

The screenshot of LWL is given as follows.



```
Correctly Classified Instances         203               96.6667 %
Incorrectly Classified Instances         7                3.3333 %
Kappa statistic                          0.95
Mean absolute error                      0.1372
Root mean squared error                  0.2191
Relative absolute error                 30.8659 %
Root relative squared error             46.484  %
Total Number of Instances              210

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  1         0.05      0.909     1         0.952       0.999    ar
  1         0         1         1         1           1        em
  0.9       0         1         0.9       0.947       0.999    sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 70  0  0 |  a = ar
  0 70  0 |  b = em
  7  0 63 |  c = sp
```

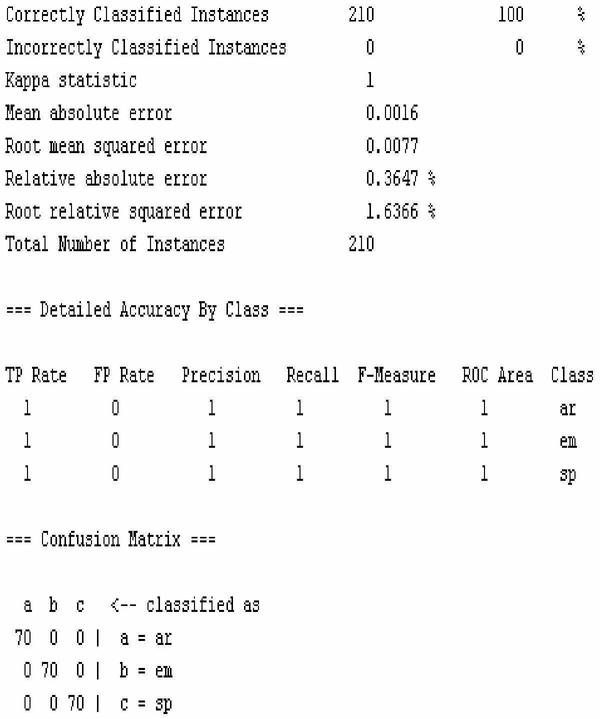
Figure 36. LWL Screenshot.



```
Correctly Classified Instances          88               97.7778 %
Incorrectly Classified Instances         2                2.2222 %
Kappa statistic                          0.9667
Mean absolute error                      0.1327
Root mean squared error                  0.2183
Relative absolute error                 29.8498 %
Root relative squared error             46.3107 %
Total Number of Instances               90

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  1         0.033     0.938     1         0.968       0.995    ar
  1         0         1         1         1           1        em
  0.933     0         1         0.933     0.966       0.995    sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 30  0  0 |  a = ar
  0 30  0 |  b = em
  2  0 28 |  c = sp
```

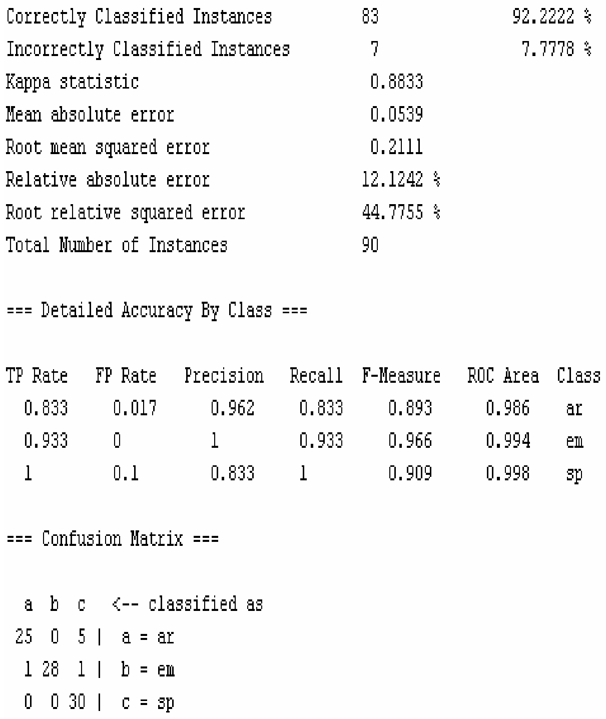
Figure 37. LWL with Testing Set.

The screenshot of Simple Logistic is given below.



```
Correctly Classified Instances         210              100      %
Incorrectly Classified Instances         0                0      %
Kappa statistic                          1
Mean absolute error                      0.0016
Root mean squared error                  0.0077
Relative absolute error                  0.3647 %
Root relative squared error              1.6366 %
Total Number of Instances              210

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  1         0          1          1        1          1        ar
  1         0          1          1        1          1        em
  1         0          1          1        1          1        sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 70  0  0 |  a = ar
  0 70  0 |  b = em
  0  0 70 |  c = sp
```

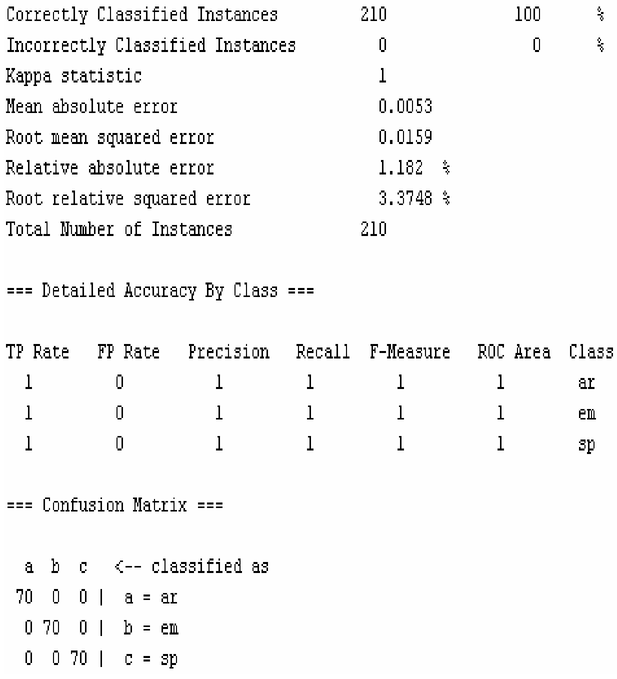
Figure 38. Simple Logistic Screenshot.



```
Correctly Classified Instances          83               92.2222 %
Incorrectly Classified Instances         7                7.7778 %
Kappa statistic                          0.8833
Mean absolute error                      0.0539
Root mean squared error                  0.2111
Relative absolute error                 12.1242 %
Root relative squared error             44.7755 %
Total Number of Instances               90

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.833     0.017      0.962     0.833     0.893      0.986    ar
  0.933     0          1         0.933     0.966      0.994    em
  1         0.1        0.833     1         0.909      0.998    sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 25  0  5 |  a = ar
  1 28  1 |  b = em
  0  0 30 |  c = sp
```

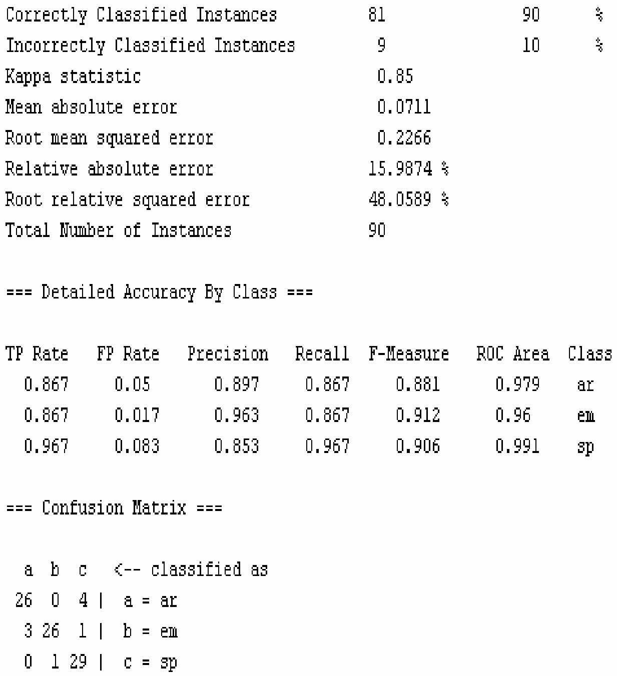
Figure 39. Simple Logistic with Testing Set.

The screenshot of Multilayer Perceptron is given as follows.



```
Correctly Classified Instances         210              100      %
Incorrectly Classified Instances         0                0      %
Kappa statistic                          1
Mean absolute error                      0.0053
Root mean squared error                  0.0159
Relative absolute error                  1.182  %
Root relative squared error              3.3748 %
Total Number of Instances              210

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  1         0          1          1        1          1        ar
  1         0          1          1        1          1        em
  1         0          1          1        1          1        sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 70  0  0 |  a = ar
  0 70  0 |  b = em
  0  0 70 |  c = sp
```

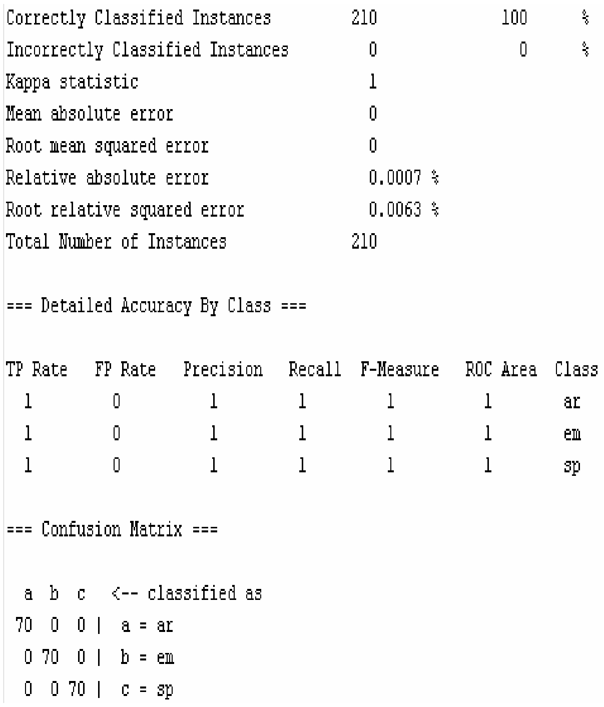
Figure 40. Multilayer Perceptron Screenshot.



```
Correctly Classified Instances          81               90      %
Incorrectly Classified Instances         9               10      %
Kappa statistic                          0.85
Mean absolute error                      0.0711
Root mean squared error                  0.2266
Relative absolute error                 15.9874 %
Root relative squared error             48.0589 %
Total Number of Instances               90

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.867     0.05       0.897     0.867     0.881      0.979    ar
  0.867     0.017      0.963     0.867     0.912      0.96     em
  0.967     0.083      0.853     0.967     0.906      0.991    sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 26  0  4 |  a = ar
  3 26  1 |  b = em
  0  1 29 |  c = sp
```

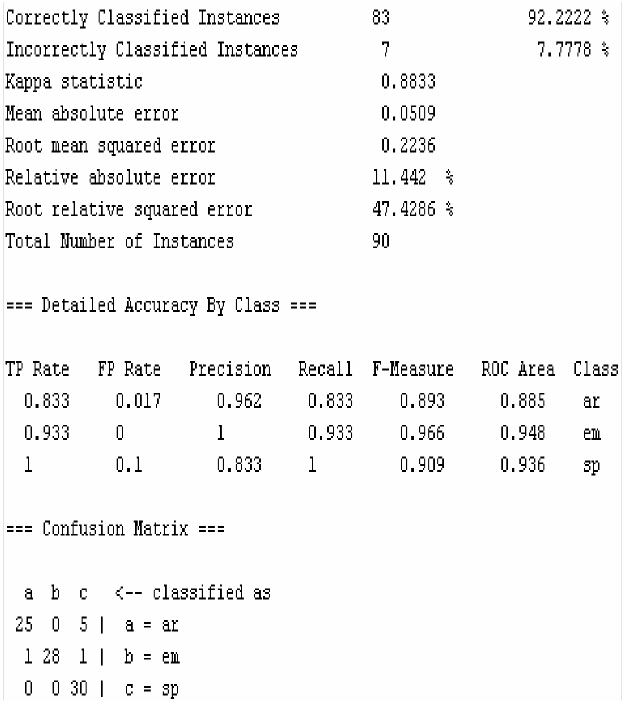
Figure 41. Multilayer Perceptron with Testing Set.

The screenshot of RBF Network is given below.



```
Correctly Classified Instances         210              100     %
Incorrectly Classified Instances         0                0     %
Kappa statistic                          1
Mean absolute error                      0
Root mean squared error                  0
Relative absolute error                  0.0007 %
Root relative squared error              0.0063 %
Total Number of Instances              210

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  1         0          1          1        1          1        ar
  1         0          1          1        1          1        em
  1         0          1          1        1          1        sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 70  0  0 |  a = ar
  0 70  0 |  b = em
  0  0 70 |  c = sp
```

Figure 42. RBF Network Screenshot.



```
Correctly Classified Instances          83               92.2222 %
Incorrectly Classified Instances         7                7.7778 %
Kappa statistic                          0.8833
Mean absolute error                      0.0509
Root mean squared error                  0.2236
Relative absolute error                 11.442  %
Root relative squared error             47.4286 %
Total Number of Instances               90

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.833     0.017      0.962     0.833     0.893      0.885    ar
  0.933     0          1         0.933     0.966      0.948    em
  1         0.1        0.833     1         0.909      0.936    sp

=== Confusion Matrix ===

  a  b  c   <-- classified as
 25  0  5 |  a = ar
  1 28  1 |  b = em
  0  0 30 |  c = sp
```

Figure 43. RBF Network with Testing Set.

For ANFIS, test has been performed as follows. As the output, 0.1 has been selected as AR, 0.2 has been selected as ES and 0.3 has been selected as DS. For the experiment, Sub Clustering method has been used. The range of influence has been chosen as 0.5, squash factor has been selected as 1.25, accept ratio has been chosen as 0.5 and rejection ratio has been selected as 0.15. First, training and testing data sets aes_trn.dat and aes.tstng.dat have been loaded, respectively. Next, classification of training data has been performed by selecting "Plot against: Training data". Then, classification of testing data has been performed by selecting "Plot against: Testing data." As a result, Figure 44 and Figure 45 have been obtained. Red dots correspond to FIS output. By grouping red dots, sensitivity, specificity and precision values of training data and testing data have been computed. This process is called cross validation technique. RMSE value has been obtained from the ANFIS editor.

Considering Figure 44, training specificity for AR, ES and DS has been evaluated as 100%.

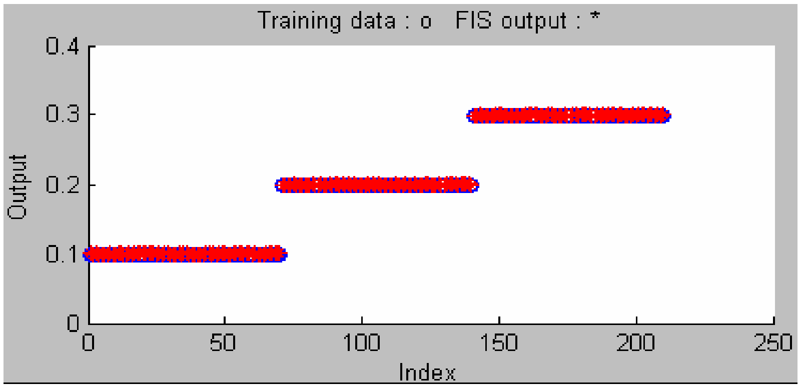


Figure 44. ANFIS Classification of Training Data.

Considering Figure 45, testing specificity of AR, ES and DS has been evaluated as 98%, 98% and 97%, respectively. Again, testing specificity has been considered as the detection result of three signals. At the end, 98% of the data has been correctly classified, averagelly.

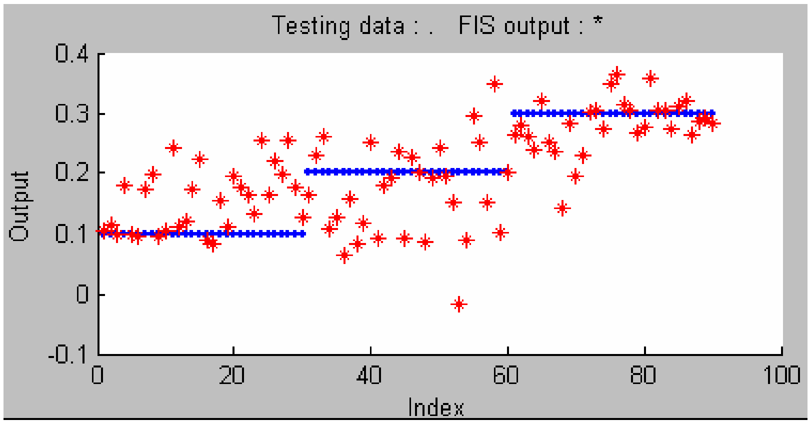


Figure 45. ANFIS Classification of Testing Data.

Fuzzy inference diagram shows the rules computed by inputs. As shown in Figure 46 and Figure 47, ANFIS has been created 795 rules. When the red vertical line is on peak value of curve or nearby the peak value of curve, it can be argued that this rule has been predicted output successively.

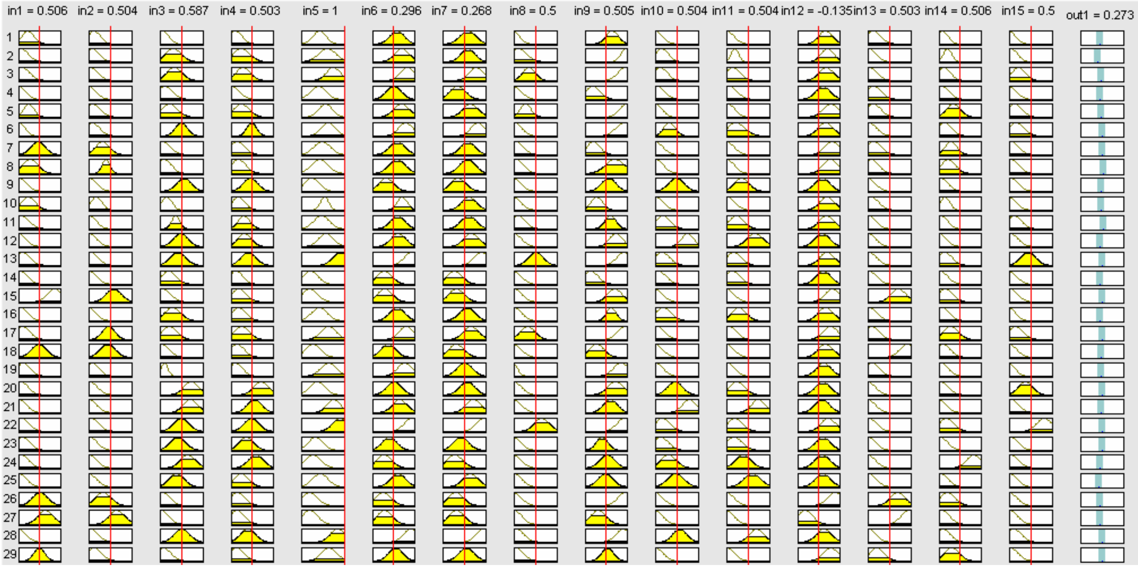


Figure 46. Fuzzy Inference Diagram.

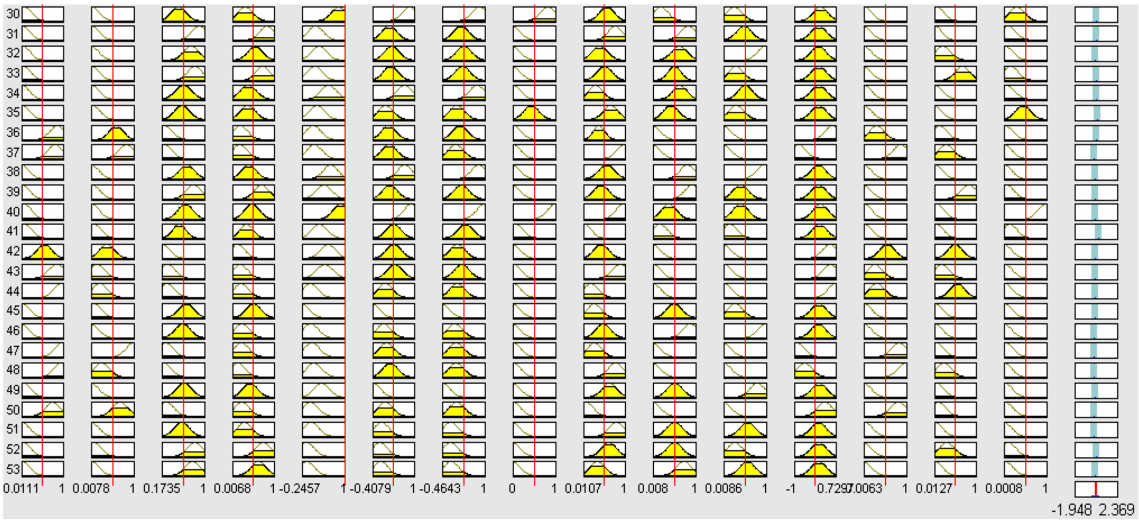


Figure 47. Fuzzy Inference Diagram (continued).

The FIS structure view gives information about Fuzzy Inference System. As seen in Figure 48, 15 input parameters have been given as input. ANFIS has been produced 795 input membership functions. Then, by Sub-Clustering method, ANFIS has been generated 795 rules and 795 output membership functions.

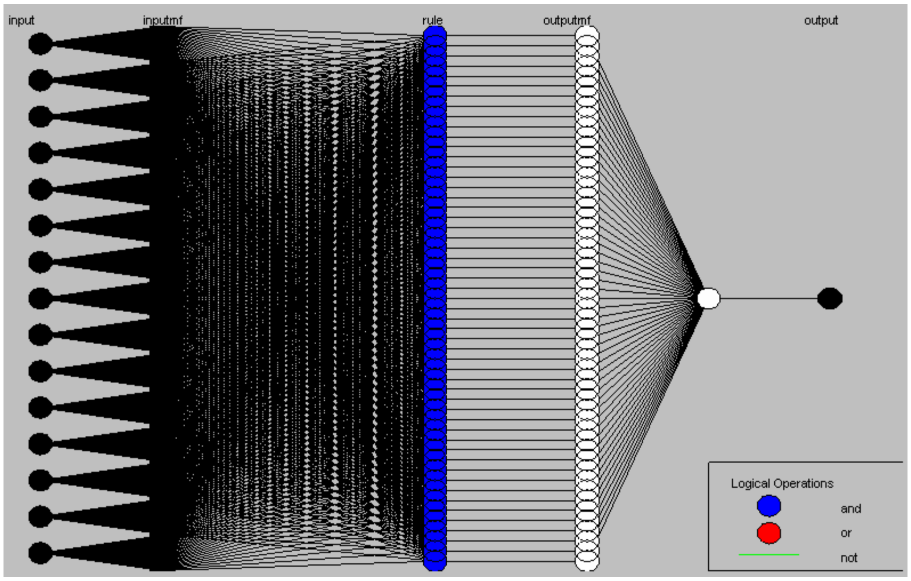


Figure 48. FIS Structure of the System.

4.1. Comparison of Data Mining Techniques

Data mining techniques have been classified with respect to sensitivity and specificity, as given below.

TP, TN, FP, FN, Sensitivity, Specificity, TP Rate, FP Rate, Precision

TP means True Positive items correctly classified. TN means True Negative items correctly classified. FP means False Positive items correctly classified and FN means False Negative items correctly classified. Sensitivity is the fraction of positive instances. It is also defined as TP Rate. TP Rate is defined as Sensitivity and evaluated as

$$\text{TP RATE} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{33}$$

Specificity is the fraction of the samples predicted as positives which are truly positives. It is also defined as 1- FP Rate. Specificity is computed as

$$\text{SPECIFICITY} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{34}$$

FP Rate is evaluated as

$$\text{FP RATE} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{35}$$

Precision or accuracy is the fraction of correctly classified samples. It is computed as (Tang, et all, 2005).

$$\text{PRECISION} = \frac{\text{TN} + \text{TP}}{\text{TN} + \text{TP} + \text{FN} + \text{FP}} \tag{36}$$

4.2. Receiver Operating Characteristics Curve

Receiver **O**perating **C**haracteristic (ROC) curve is used to view the performance of classification. ROC curve is drawn as FP Rate (1-specificity) vs TP Rate (sensitivity). The perfect model for ROC curve is a line between the points [0, 0], [0, 1] and a line [0, 1], [1, 1] which look like symbol Γ. Any signal which is close as those lines gives more reliable information. ROC curve for Weka has been evaluated as follows. First, the test with testing set has been completed. Then, in the result list, the tested method, for example "bayes.NaiveBayes" has been clicked. The option "visualize threshold curve->AR" has been chosen. Next, ROC curve graph for AR has been obtained. After that, "Save" button has been clicked. The code has been saved in somewhere. Then, the saved code has been opened with a text editor. Finally, the sixth and seventeenth parameters of code have been obtained as FP Rate and TP Rate, respectively. Using the same procedure, for ES and DS, FP Rate and TP Rate values have been computed. At the end, average values of FP Rate and TP Rate have been calculated. For ANFIS, by using cross validation technique, FP Rate and TP Rate values have been computed. In order to evaluate them, ANFIS classification of testing data (Figure 45) has been used. As a result, ROC curve has been drawn considering final FP Rate and TP Rate values. In Figure 49, ROC Curve for Naive Bayes, Classification via Regression, LWL, RBF Network, Multilayer Perceptron and ANFIS has been drawn. The two algorithms Naive Bayes and Classification via Regression are more reliable than the others. According to the curve, the fastest algorithms are Naive Bayes, Classification Via Regression and Multilayer Perceptron, respectively (Tang, et all, 2005).

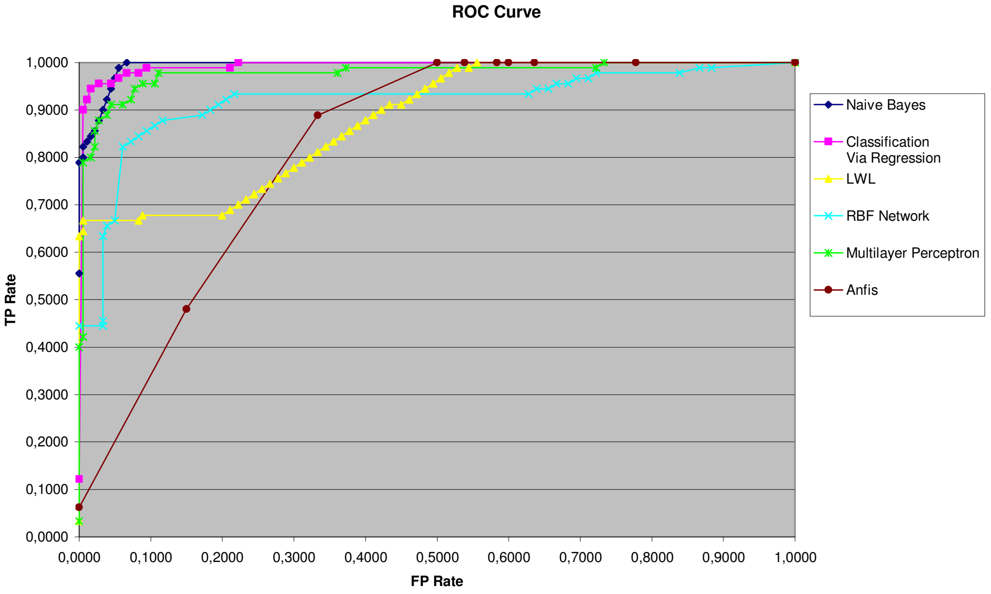


Figure 49. ROC Curve for Naive Bayes, ClassificationVia Regression, LWL, RBF Network, Multilayer Perceptron, and ANFIS.

5. RESULTS & DISCUSSION

In the tests; NBTree, LMT, Ridor, Nnge, VFI, Classification Via Regression, AdaBoostM1, Bagging, Dagging, Decorate, LWL, Simple Logistic, RBF Network, SMO, Multilayer Perceptron, Naive Bayes and ANFIS have been used. In Weka, all methods have been selected and accepted which have 90% and above correctness values each. In Table 2, NBTree, NaiveBayes, Ridor, Classificatin Via Regression, Bagging, AdaBoostM1 and Nnge have RMSE values below 20% and it can be argued that these algorithms are more reliable. Sensitivity and specificity results are average values of AR, ES and DS signals. Precision, RMSE and Correct results have been obtained by performing the test with testing set. Testing specificity has been considered as detection result of signals. Detection result of signals and comparison of methods has been given in Table 4 in Conclusion section.

Table 2. Average test Results of the Used Methods sorted as ascending order of RMSE.

Method	Type	Training Data Sensitivity	Training Specificity	Testing Data Sensitivity	Testing Specificity	Precision	RMSE	Correct
NBTree	Trees	100%	100%	98%	99%	0,9793	0,1299	98%
NaiveBayes	Bayes	97%	99%	97%	98%	0,9696	0,1413	97%
Ridor	Rules	98%	99%	97%	98%	0,9670	0,1491	97%
Classification via Regression	Meta	100%	100%	97%	98%	0,9680	0,1596	97%
Bagging	Meta	99%	99%	96%	98%	0,9566	0,1636	96%
Ada Boost M1	Meta	99%	100%	97%	98%	0,9696	0,1714	97%
Nnge	Rules	100%	100%	96%	98%	0,9560	0,1721	96%
Decorate	Meta	100%	100%	92%	96%	0,9270	0,2042	92%
LMT	Trees	100%	100%	92%	96%	0,9316	0,2111	92%
Simple Logistic	Functions	100%	100%	92%	96%	0,9316	0,2111	92%
LWL	Lazy	97%	98%	98%	99%	0,9793	0,2183	98%
RBF Network	Functions	100%	100%	92%	96%	0,9316	0,2236	92%
Multilayer Perceptron	Functions	100%	100%	90%	95%	0,9043	0,2266	90%
ANFIS	Matlab	100%	100%	96%	98%	0,9734	0,2664	96%
VFI	Misc	100%	100%	92%	96%	0,9273	0,2812	92%
Dagging	Meta	94%	97%	93%	97%	0,9340	0,2946	93%
SMO	Functions	98%	99%	92%	96%	0,9320	0,3063	92%

Evaluation of Testing Specificity

In Weka, testing specificity has been evaluated using the formula

$$\text{Specificity=1-FP Rate} \tag{37}$$

In ANFIS, using cross-validation techique, testing specificity of AR, ES and DS have been evaluated as 0.99, 0,98 and 0.97, respectively. In Table 3, evaluation of testing specificity has been shown.

Table 3. Evaluation of Testing Specificity.

Method	**Testing Specificity =(1-FP Rate)**		**Average Testing Specificity =(AR+ES+DS)/3**
Naive Bayes	**AR**	1-0=1	(1+1+0.95)/3=0.98
	ES	1-0=1	
	DS	1-0.05=0.95	
NBTree	**AR**	1-0.033=0.97	(0.97+1+1)/3=0.99
	ES	1-0=1	
	DS	1-0=1	
LMT	**AR**	1-0.017=0.98	(0.98+1+0,9)/3=0.96
	ES	1-0=1	
	DS	1-0.1=0.9	
Ridor	**AR**	1-0.017=0.98	(0,98+1+0,97)/3=0.98
	ES	1-0=1	
	DS	1-0.033=0.97	
Nnge	**AR**	1-0.033=0.97	(0.97+0.98+0.98)/3=0.98
	ES	1-0.017=0.98	
	DS	1-0.017=0.98	
VFI	**AR**	1-0.033=0.97	(0.97+1+0.92)/3=0.96
	ES	1-0=1	
	DS	1-0.083=0.92	
Bagging	**AR**	1-0.017=0.98	(0.98+1+0.95)/3=0.98
	ES	1-0=1	
	DS	1-0.05=0.95	
Dagging	**AR**	1-0.05=0.95	(0.95+0.98+0.97)/3=0.97
	ES	1-0.017=0.98	
	DS	1-0.033=0.97	
Decorate	**AR**	1-0.017=0.98	(0.98+0.98+0.92)/3=0.96
	ES	1-0.017=0.98	
	DS	1-0.083=0.92	
Ada Boost M1	**AR**	1-0=1	(1+1+0.95)/3=0.98
	ES	1-0=1	
	DS	1-0.05=0.95	

Table 3. Evaluation of Testing Specificity (continued).

SMO	**AR** **ES** **DS**	1-0=1 1-0.017=0.98 1-0.1=0.9	(1+0.98+0.9)/3=0.96
Classification via Regression	**AR** **ES** **DS**	1-0.017=0.98 1-0=1 1-0.033=0.97	(0.98+1+0.97)/3=0.98
LWL	**AR** **ES** **DS**	1-0.033=0.97 1-0=1 1-0=1	(0.97+1+1)/3=0.99
Simple Logistic	**AR** **ES** **DS**	1-0.017=0.98 1-0=1 1-0.1=0.9	(0.98+1+0.9)/3=0.96
Multilayer Perceptron	**AR** **ES** **DS**	1-0.05=0.95 1-0.017=0.98 1-0.083=0.92	(0.95+0.98+0.92)/3=0.95
RBF Network	**AR** **ES** **DS**	1-0.017=0.98 1-0=1 1-0.1=0.9	(0.98+1+0.9)/3=0.96
ANFIS	**AR** **ES** **DS**	0.98 0.98 0.97	(0.98+0.98+0.97)/3=0.98

In Figure 50, methods have been sorted considering ascending order of RMSE. NBTree, Naive Bayes, Ridor, Classification via Regression, Bagging, Ada Boost M1 and Nnge have RMSE values under 0.2 which means they are more reliable methods than others.

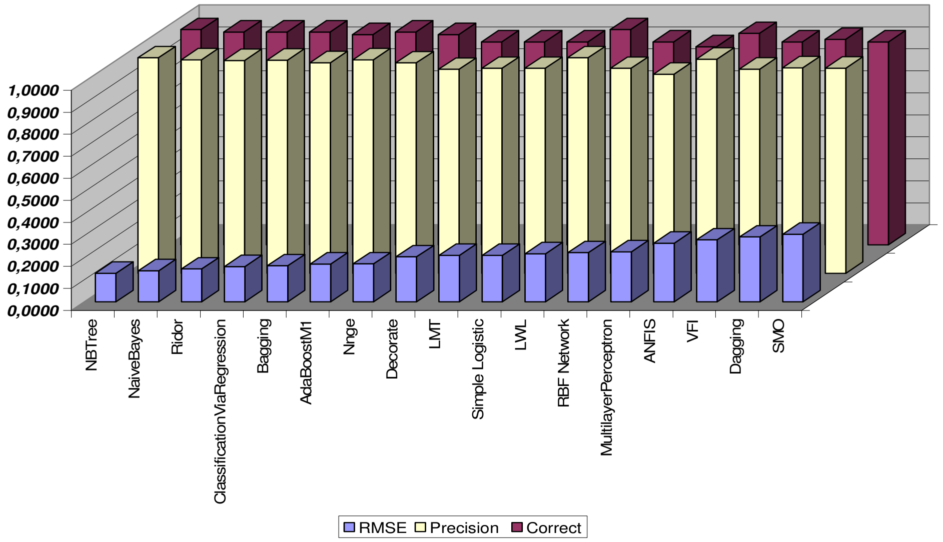


Figure 50. RMSE, Precision and Correctness Chart, sorted by ascending order of RMSE.

In Figure 51, methods have been sorted considering descending order of precision. NBTree, LWL, ANFIS, Naive Bayes, Ada Boost M1, Classification via Regression, Ridor, Bagging and Nnge have been given above 95% precision.

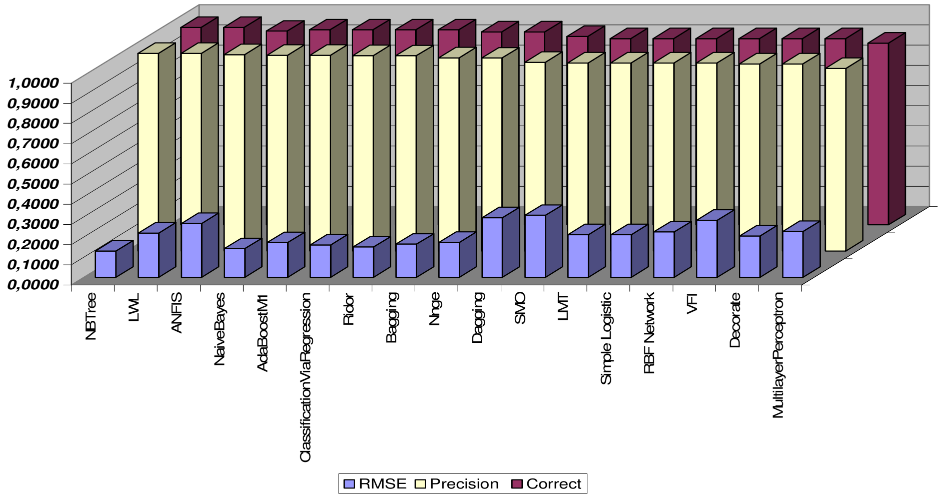


Figure 51. RMSE, Precision and Correctness Chart, sorted by descending order of Precision.

In Figure 52, algorithms have been sorted considering descending order of Correctness. NBTree, LWL, Ridor, Classification via Regression, Naive Bayes, Ada Boost M1, ANFIS, Nnge and Bagging have been given 96% and above correctness.

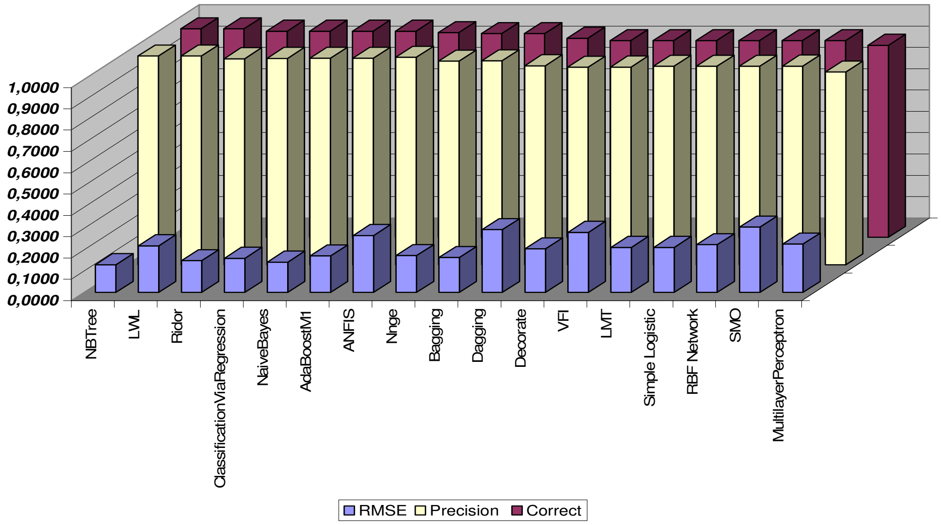


Figure 52. RMSE, Precision and Correctness Chart, sorted by descending order of Correctness.

In Figure 53, methods have been sorted considering descending order of training sensitivity. NBTree, Classification via Regression, ANFIS, Nnge, Decorate, VFI, LMT, Simple Logistic, RBF Network and Multilayer Perceptron have been given 100% training sensitivity.

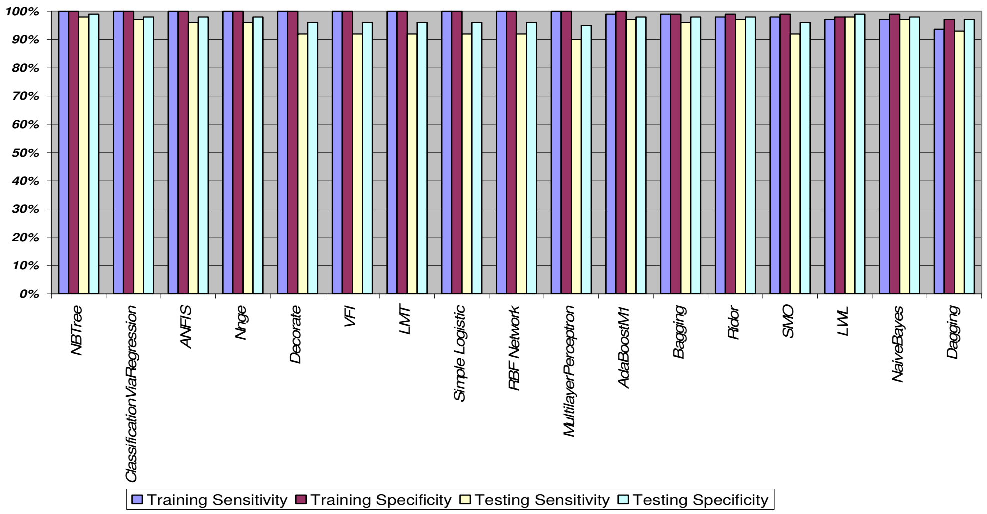


Figure 53. Sensitivity and Specificity Measures, sorted by descending order of Training Sensitivity.

In Figure 54, methods have been sorted according to descending order of training specificity. Except LWL and Dagging, the rest of the algorithms have been given 99% and above training specificity.

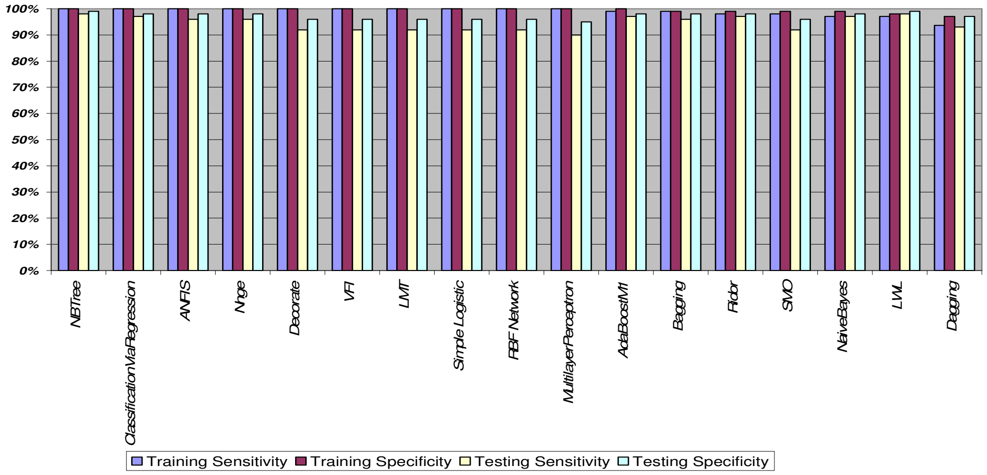


Figure 54. Sensitivity and Specificity Measures, sorted by descending order of Training Specificity.

In Figure 55, methods have been sorted considering descending order of testing sensitivity. NBTree, LWL, Ada Boost M1, Naive Bayes, Classification via Regression, Ridor, ANFIS, Bagging and Nnge have been given 96% and above testing sensitivity.

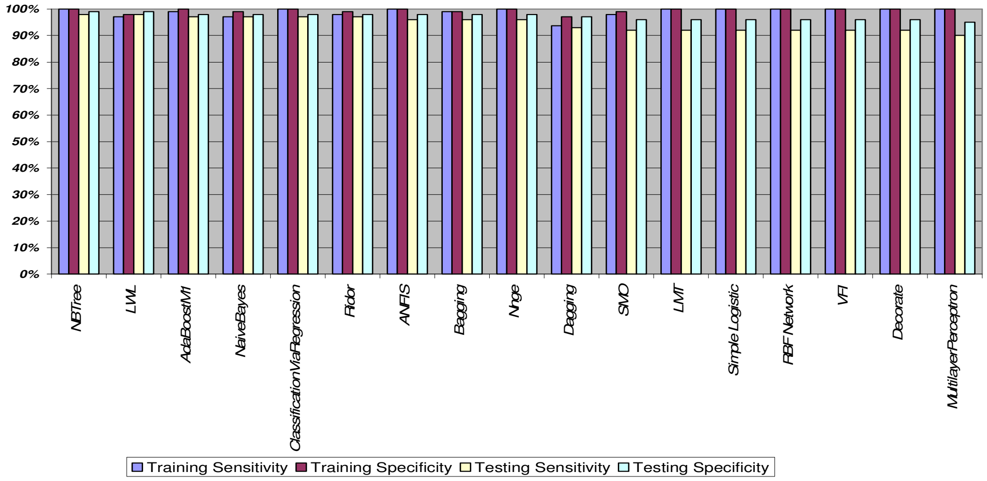


Figure 55. Sensitivity and Specificity Measures, sorted by descending order of Testing Sensitivity.

In Figure 56, algorithms have been sorted according to descending order of testing specificity. NBTree, LWL, Ada Boost M1, Naive Bayes, Classification via Regression, Ridor, ANFIS, Bagging and Nnge have been given 98% and above testing specificity.

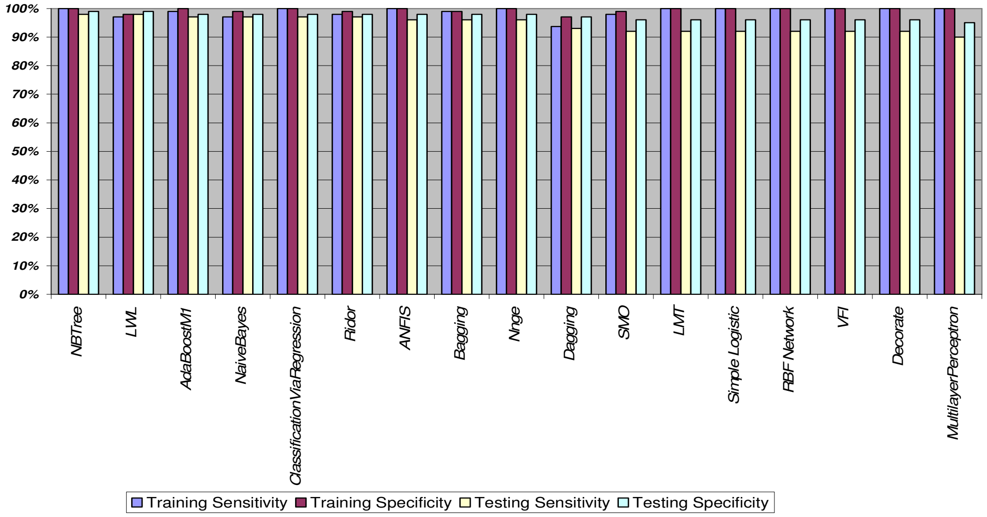


Figure 56. Sensitivity and Specificity Measures, sorted by descending order of Testing Specificity.

6. CONCLUSION

In this study, the data mining techniques for classification of Doppler signals have been studied. The aim was to increase the sensitivity and specificity of the detection system. For this reason, data mining methodology has been used. For the comparison purpose, testing specificity has been considered as the detection result of signals. Classification results which are obtained in this study were almost same or more accurate than the previous system. Table 4 gives the comparison of fuzzy logic detection and the used DM methods. Because of the data set 2 in Aydin, et all, 2004 has been used, detection results of signals from data set 2 have been taken. If we compare the fuzzy logic with other DM methods, for AR signals, Naïve Bayes, Ada Boost M1, SMO and ANFIS have shown better performance. Considering ES signal, all of the DM methods have given better performance than fuzzy logic. Finally, considering SP signal, fuzzy logic has shown an average performance. Fuzzy logic has been better than VFI, Decorate, Multilayer Perceptron, LMT, Simple Logistic, RBF Network and SMO. Overall, according to AR signal, Naive Bayes, Ada Boost M1 and SMO performed well. For the ES signal; Naive Bayes, Ada Boost M1, Ridor, Classification via Regression, Bagging, LMT, Simple Logistic, RBF Network, NBTree, LWL and VFI were good. And finally for the DS signal, NBTree and LWL performed well.

Table 4.Comparision of Fuzzy Logic Rule with used DM Methods.

Method	Detection Result of Signal		
	AR	ES	DS
Fuzzy Logic (results from 2nd data set)	98%	95%	95%
Naive Bayes	100%	100%	95%
NBTree	97%	100%	100%
LMT	98%	100%	90%
Ridor	98%	100%	97%
Nnge	97%	98%	98%
VFI	97%	100%	92%
Bagging	98%	100%	95%
Dagging	95%	98%	97%
Decorate	98%	98%	92%
Ada Boost M1	100%	100%	95%
SMO	100%	98%	90%
Classification via Regression	98%	100%	97%
LWL	97%	100%	100%
Simple Logistic	98%	100%	90%
Multilayer Perceptron	95%	98%	92%
RBF Network	98%	100%	90%
ANFIS	98%	98%	97%

7. REFERENCES

[1] N. Aydin, F. Marvasti, H. S. Markus, “Embolic Doppler Ultrasound Signal Detection Using Discrete Wavelet Transform,” IEEE Transactions on Information Technology in Biomedicine, vol.8, no.2, pp. 182-190, June-2004.

[2] C. Apte, "Data Mining: An Industrial Research Perspective," IEEE Computational Science and Engineering, vol. 04, no. 2, pp. 6-9, April/June-1997.

[3] I.M.Mullins, M.S.Siadaty, J. Lyman,K. Scully, C.T.Garrett, “Data Mining and Clinical Data Repositories, Insights from 667,000 Patient Data Set,” Computers in Biology and Medicine, vol.36, no.12, pp. 1351-1377, December-2006.

[4] A.Ouali,A.Ramdane-Cherif, M.O. Krebs, “Delimiting Cut-Off of Age at Onset in Schizophrenia Using Bayesian Network,” The 4th IEEE International Conference in Cognitive Informatics, University of California, Irvine, USA, pp. 276-284, August-2005.

[5] R. Kohavi, “Scaling up the Accuracy of Naive-Bayes Classifiers: a Decision Tree Hybrid,” In Proc. of the 2nd Int. Conf. on Knowledge Discovery and Data Mining, pp. 202–207, 1996.

[6] M. J. Fonseca, J. A. Jorge, “Indexing High-Dimensional Data for Content-Based Retrieval in Large Databases,” Technical report, pp.267-274, March-2003.

[7] Z. Xie, “A Study of Selective Neighborhood-Based Naive Bayes for Efficient Lazy Learning,” Tools with Artificial Intelligence, 16th IEEE International Conference, pp.758-760, November-2004.

[8] N. Landwehr, M. Hall, E. Frank, “Logistic Model Trees,” Machine Learning, vol.59, no.1-2, pp. 161-205, May-2005.

[9] P.J. Compton, R. Jansen, "A Philosophical Basis for Knowledge Acquisition," vol.2, no.3, pp. 241-257, September-1990.

[10] D. Richards, "Reverse Engineering Ontologies from Performance Systems," Proc. of 22nd Annual International Conference of the British Computer Society's Specialist Group on Artificial Intelligence, pp 1-14, 2002.

[11] P.J. Clark and F.C. Evans, “Generalization of a Nearest Neighbor Measure of Dispersion for Use in K Dimensions,” Ecology, vol.60, no2, pp.316-317 1979.

[12] E. Pignotti, P. Edwards, G. A. Grimnes, “Context-Aware Personalised Service Delivery,” Proceedings of the Sixteenth European Conference on Artificial Intelligence - ECAI-2004, Valencia-Spain, pp. 1077-1078, IOS Press, Amsterdam, 2004.

[13] G. Demiroz, A. Guvenir, “Classification by Voting Feature Intervals,” Lecture Notes In Computer Science, Proceedings of the 9th European Conference on Machine Learning, vol. 1224, pp. 85 - 92, 1997.

[14] E. Kalapanidas, N. Avouris, M.Craciun and D.Neagu, “Machine Learning Algorithms: A Study on Noise Sensitivity,” Proc. 1st Balcan Conference in Informatics, pp. 356-365, Thessaloniki, November-2003.

[15] L. Breiman, “Bagging Predictors,” Machine Learning, vol.24, no.2, pp.123-140, 1996.

[16] N.C. Oza, “Online Bagging and Boosting,” Conference on Systems, Man and Cybernetics, 2005 IEEE International Conference, pp.2340-2345, vol.3, October-2005.

[17] L.Tate, “Using Subset Training on noisy Data,” University of Texas at San Antonio, pp 18-19, 2006.

[18] W. K. Ting, I. Witten, “Stacking Bagged and Dagged Models,” Proc. of ICML'97, Morgan Kaufmann, San Francisco CA, pp.367-375, 1997.

[19] P. Melville, R. Mooney, “Creating Diversity in Ensembles Using Artificial Data,” Journal of Information Fusion, Special Issue on Diversity in Multiple Classifier Systems, vol.6, no.1, pp. 99-111, 2004.

[20] Y. Freund, R. E. Schapire, “Experiments with a New Boosting Algorithm in Machine Learning,” Proc. of the Thirteenth International Conference, pp. 148-156, 1996.

[21] R. Jin, L. Si, Y. Liu, A.Hauptmann, J. Carbonell, “A New Boosting Algorithm Using Input Dependent Regularizer,” 20th International Conference on Machine Learning (ICML'03), Washington, DC, pp.1-5, August-2003.

[22] J.C.Platt, “Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines,” Technical Report, Microsoft Research, pp.1-8, 1998.

[23] C.J. Lin, “On the Convergence of the Decomposition Method for Support. Vector Machines,” IEEE Trans. Neural Networks, vol.11, no.4, pp.1288–1298, 2001.

[24] L. Kornienko, D. W. Albrecht, D. L. Dowe, “A Preliminary MML Linear Classifier Using Principal Components for Multiple Classes,” Australian Conference on Artificial Intelligence, vol.3809, pp.922-926, December-2005.

[25] E. Frank, Y. Wang, S. Inglis, G. Holmes, I.H. Witten, “Using Model Trees for Classification,” Machine Learning, vol.32, no.1, pp.63-76, 1998.

[26] W. B. Langdon, S. J. Barrett, B. F. Buxton, “Predicting Biochemical Interactions - Human P450 2D6 Enzyme Inhibition,” Proceedings of the 2003 Congress on Evolutionary Computation, IEEE Press, pp.807-814, 2003.

[27] C.G. Atkenson, A.W.Moore, S. Schaal, “Locally Weighted Learning,” AI Review Journal, vol.73, pp.1-10, 1997.

[28] S. Schaal, C.G. Atkeson, S. Vijayakumar, “Scalable Techniques from Nonparameteric Statistics for Real-Time Robot Learning,” Applied Intelligence, vol.17, no.1, pp.49-60, 2002.

[29] J. Friedman, T. Hastie, R. Tibshirani, “Additive Logistic Regression: a Statistical View of Boosting,” The Annals of Statistics, vol.38, no.2, pp.337-374, April-2000.

[30] J. E. Anderson, “Logistic Regression Model,” University of Minnesota, Division of Biostatistics, pp.1-14, November-2002.

[31] H. Yan, J. Zheng, Y. Jiang, C. Peng, Q. Li, “Development of a Decision Support System for Heart Disease Diagnosis using Multilayer Perceptron,” vol.5, pp.709-712, 2003.

[32] J. Isa, “CBID – Clustering based 1D Learner,” Charles University-Prague, pp.1-5, 2005.

[33] Y. Tan, J. Wang, J.M. Zurada, "Nonlinear Blind Source Separation using a Radial Basis Function Network," IEEE Transactions on Neural Networks, vol.12, no.1, pp.124-134, 2001.

[34] J.S.R. Jang, “ANFIS: Adaptive Network –based Fuzzy Inference System,” IEEE Trans. Syst. Man-Cybern., vol.23, no.3, May/June-1993.

[35] D. Hemandez, D.M. Saez, "Neuro-Fuzzy Method for Automated Defect Detection in Aluminium Castings," vol.3212, pp.826-833, 2004.

[36] Y.Tang, “Granular Support Vector Machines Based on Granular Computing, Soft Computing and Statistical Learning,” Georgia State University, Department of Computer Science, pp.1-27, December-2005.

[37] M.Serrurier, D.Dubois, H. Prade, T. Sudkamp, “Learning Fuzzy Rules with their Implication Operators,” Data & Knowledge Engineering, vol.60, no.1, pp.71-89, January-2007.

[38] T. Tettey, T. Marwala, “Controlling Interstate Conflict using Neuro-Fuzzy Modeling and Genetic Algorithms,” Intelligent Engineering Systems, Proceedings International Conference INES '06, pp.30-34, June-2006.

[39] C.Son, “Correlation between Learning (Probability of Success) and Fuzzy Entropy in Control of Intelligent Robot's Part Macro-Assembly Tasks with Sensor Fusion Techniques,” Robotics and Computer-Integrated Manufacturing, vol.23, no.1, pp.47-62, February-2007.

[40] E.Anderssen, K.Dyrstad, F.Westad, H.Martens, “Reducing Over-Optimism in Variable Selection by Cross-Model Validation,” Chemometrics and Intelligent Laboratory Systems, vol.84, no.1, pp.69-74, December-2006.

[41] S. Nebuya, M. Noshiro, B.H. Brown, R.H. Smallwood, P. Milnes, “Detection of Emboli in Vessels Using Electrical Impedance Measurements - Phantom and Electrodes,” Physiological Measurement, vol.26, no.2, pp.111-118, April-2005.

[42] A.M. Demchuk, M. Saqqur, A.V. Alexandrov, “Transcranial Doppler in Acute Stroke,” Neuroimaging Clinics of North America, vol.15, no.3, pp.473-+, August-2005.

[43] E. Chung, L.K. Fan, C. Degg, D.H. Evans, “Detection of Doppler Embolic Signals: Psychoacoustic Considerations,” Ultrasound in Medicine and Biology, vol.31, no.9, pp.1177-1184, September-2005.

[44] A. Okamura, H. Ito, K. Iwakura, S. Kawano, K. Inoue, Y. Maekawa, T. Ogihara, K. Fujii, “Detection of Embolic Particles with the Doppler Guide Wire During Coronary Intervention in Patients with Acute Myocardial Infarction - Efficacy of Distal Protection Device,” Journal of the American College of Cardiology, vol.45, no.2, pp.212-215, January-2005.

[45] J.M. Girault, M. Biard, D. Kouame, A. Bleuzen, F. Tranquart, “Spectral Correlation of the Embolic Blood Doppler Signal,” Acoustics, Speech and Signal Processing, ICASSP 2006 Proceedings, IEEE International Conference, vol.2, pp.1200-1203, May-2006.

[46] D. Kouame, M. Biard, J.M. Girault, A. Bleuzen “Adaptive AR and Neurofuzzy Approaches: Access to Cerebral Particle Signatures,” IEEE Transactions on Information Technology in Biomedicine, vol.10, no.3, pp.559-566, July-2006.

[47] P. Palanchon, A. Bouakaz, J.Klein, N. de Jong "Multifrequency Transducer for Microemboli Classification and Sizing ," IEEE Transactions on Biomedical Engineering, vol.52, no.12, pp.2087-2092, December-2005.

[48] A.D. Mackinnon, R. Aaslid, H.S. Markus "Ambulatory Transcranial Doppler Cerebral Embolic Signal Detection in Symptomatic and Asymptomatic Carotid Stenosis," Stroke, vol.36, no.8, pp.1726-1730, August-2005.

[49] J. Cowe, J. Gittins, A.R. Naylor, D.H. Evans "Rf Signals Provide Additional Information on Embolic Events Recorded During TCD

Monitoring,", Ultrasound In Medicine And Biology, vol.31, no.5, pp.613-623, May-2005.

[50] F. Kilicaslan, A. Verma, E. Saad, A. Rossillo, D.A. Davis, S.K. Prasad, O. Wazni, M.F. Marrouche, L.N. Raber, J.E. Cummings, S. Beheiry, S. Hao, J.D. Burkhardt, W. Saliba, R.A. Schweikert, D.O. Martin, A. Natale "Transcranial Doppler Detection of Microembolic Signals During Pulmonary Vein Antrum Isolation: Implications for Titration of Radiofrequency Energy," Journal of Cardiovascular Electrophysiology, vol.17, no.5, pp.495-501, May-2006.

[51] R.A. Rodriguez, F. Rubens, C.D. Rodriguez, H.J. Nathan "Sources of Variability in the Detection of Cerebral Emboli with Transcranial Doppler During Cardiac Surgery," Journal of Neuroimaging, vol.16, no.2, pp.126-132, April-2006.

[52] R. Dittrich, M.A. Ritter, M. Kaps, M. Siebler, K. Lees, V. Larrue, D.G. Nabavi, E.B. Ringelstein, H.S. Markus, D.W. Droste "The Use of Embolic Signal Detection in Multicenter Trials to evaluate Antiplatelet Efficacy - Signal Analysis and Quality Control Mechanisms in the caress (Clopidogrel and Aspirin for Reduction of Emboli in Symptomatic Carotid Stenosis) Trial," Stroke, vol.37, no.4 ,pp.1065-1069, April-2006.

[53] C.I. Chen, Y. Iguchi, Z. Garami, M.D. Malkoff, R.W. Smalling, M.S. Campbell, A.V. Alexandrov "Analysis of Emboli During Carotid Stenting with Distal Protection Device," Cerebrovascular Diseases, vol.21, no.4, pp.223-228, 2006.

[54] M. Rosenkranz, J. Fiehler, W. Niesen, C. Waiblinger, B. Eckert, O. Wittkugel, T. Kucinski, J. Rother, H. Zeumer, C. Weiller, U. Sliwka "The Amount of Solid Cerebral Microemboli During Carotid Stenting Does Not Relate to the Frequency of Silent Ischemic Lesions," American Journal Of Neuroradiology, vol.27, no.1, pp.157-161, January-2006.

8. VITA

Türkalp Kucur was born in Ankara. He received his B.S. degree in Computer Engineering from the Işık University in 2004. During high school, he has had training period in technical service department in TRT Genel Müdürlüğü, Ankara, at year 1999. During university year, he also has had training period on technical service department of Casper Bilgisayar Sistemleri A.Ş. at year 2002 and software department of Yage Bilgisayar LTD at year 2003. His interests are SQL server, visual basic and java.