Volume 2, No. 3, May-June 2011
ISSN No. 0976-5697
International Journal of Advanced Research in Computer Science
RESEARCH PAPER
Available Online at www.ijarcs.info

Performance of Classification Algorithms in Heart Disease Data

P.Santhi*
Lecturer, Computer Science and engineering
Paavai Engineering College
Namakkal, India
mssanthiboopathicse@gmail.com

Dr..V.Murali Bhaskaran
Principal
Paavai College of Engineering
Namakkal, India
Murali66@gmail.com

Abstract: The healthcare industry is one of the world largest and fastest growing industries. In healthcare industries having the large amounts of data, it maintains general health in populations and communities through the promotion of healthy behavior and prevention of disease. In healthcare industry the heart disease is most challenging problem. In this paper proposes the classification algorithms to evaluate the performance of the classifiers by using heart disease prediction data. We evaluate the performance of the classifiers of Bayes (BayesNet, Naïve Bayes, Naïve Bayes updateable), functions (Logistics, Multilayer Perception, RBF Network, SMO, simple Logistics), Lazy (IB1, IBK, Kstar, LWL), Meta(AdaBoost, Attribute Selected Classifier, Bagging, CVParameter Selection, Classification via Regression, Decorate, Filtered Classifier, Grading, LogitBoost, Multi BoostAB, Multiclass Classifier, Multischeme, Ordinal class classifier, Raced Incremental Logit Boost, Random Committ, Stacking, StackingC), Misc(Hyper Pipes, VFI), Rule(conjunctive rule, Decision Table, JRip, OneR, NNge, PART, Ridor, ZeroR), trees(Decision stump, J48, NB Tree, REP Tree, Random Forest, Random Tree).The prediction accuracy of the classifiers are evaluated using 10 folds cross validation. The final result shows the performance of the classifiers based on the prediction Accuracy.
Keywords: Classification, Bayes, Lazy, Rule, Meta, Tree.

I. INTRODAUCTION

In healthcare industry the heart disease is most challenging problem in the real world [4]. It is having the large amounts of data. Heart disease is a general name for a wide variety of diseases, disorders and conditions that affect the heart and sometimes the blood vessels as well. Heart disease is the number one killer of women and men. Symptoms of heart disease vary depending on the specific type of heart disease. A classic symptom of heart disease is chest pain. However, with some forms of heart disease, such as atherosclerosis, there may be no symptoms in some people until life-threatening complications develop. Any of a number of conditions that can be affects the heart. Some examples include coronary heart disease, heart attack, cardiovascular disease, pulmonary heart disease and high blood pressure. Heart disease is a big problem in today society because of lifestyle issues such as poor diet, lack of exercise and smoking. This paper evaluates the performance of the classifiers by using the heart disease data [1]. The performance will be evaluated by using the classifiers of functions, Lazy, Meta, Rule and Tree. In final, the result shows the performance of classifiers in increasing data set size. The prediction accuracy of the classifiers is evaluated based on the 10 folds cross validation.

II. EVALUATION STEPS

In this system having the following steps for evaluating the performance of the classifiers:

A. Data set Collection
B. Data Preprocessing
C. Classification
D. Performance Evaluation

A. *Data Source*

In this paper the performance of the classifiers are evaluated by using the heart disease data. In this data have been collected from Knowledge discovery data set in the Switzerland data base. It contains the 107 Instances and 14 attributes [4]. The following Visualization shows the ranges of each attributes in heart disease data.

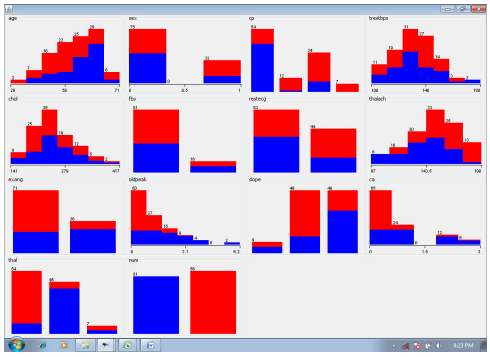

Fig.1. Visualization of Attributes

B. *Data Preprocessing*

Data pre-processing is an often neglected but important step in the data mining process[3]. The phrase "Garbage In, Garbage Out" is particularly applicable to data mining and machine learning projects. Data gathering methods are often loosely controlled, resulting in out-of-range values, impossible data combinations, missing values, etc. Analyzing data that has not been carefully screened for such problems can produce misleading results. Thus, the representation and quality of data is first and foremost before running an analysis.

If there is much irrelevant and redundant information present or noisy and unreliable data, then knowledge discovery during the training phase is more difficult. Data preparation and filtering steps can take considerable amount of processing time. Data pre-processing includes cleaning, normalization, transformation, feature extraction and selection, etc. The product of data pre-processing is the final training set [1]. In preprocessing the data cleaning is the process of removing the noise, irrelevant data and removing the inconsistency. It is also

having the data integration, data transformation and data reduction.

C. ***Classification***

The classification is the method of supervised learning [6]. It is the task of generalizing known structure to apply to new data [6]. In this classification contains the classifiers of Bayes, functions, Lazy, Meta, Misc and Tree classifiers [7].

1. *Building Bayes Classifiers Algorithm*

a. *BayesNet:* Bayes Network learning using various searches algorithms and quality measures.

OPTIONS

BIFFile -- Set the name of a file in BIF XML format. A Bayes network learned from data can be compared with the Bayes network represented by the BIF file. Statistics are calculated the number of missing and extra arcs.

Debug -- If set to true, classifier may output additional info to the console.

Estimator -- Select Estimator algorithm for finding the conditional probability tables of the Bayes Network.

Search Algorithm -- Select method used for searching network structures.

Use ADTree -- When ADTree (the data structure for increasing speed on counts, not to be confused with the classifier under the same name) is used learning time goes down typically. However, because ADTrees are memory intensive, memory problems may occur. Switching this option off makes the structure learning algorithms slower, and run with less memory. By default, ADTrees are used.

b. *Naïve Bayes:* The Class is used for a Naive Bayes classifier using estimator classes. Numeric estimator precision values are chosen based on analysis of the training data. For this reason, the classifier is not an UpdateableClassifier (which in typical usage are initialized with zero training instances) -- if you need the Updateable Classifier functionality, use the NaiveBayes Updateable classifier [8]. The Naive Bayes Updateable classifier will use a default precision of 0.1 for numeric attributes when build Classifier is called with zero training instances.

OPTIONS

Debug -- If set to true, classifier may output additional info to the console.

Use Kernel Estimator -- Use a kernel estimator for numeric attributes rather than a normal distribution.

Use Supervised Discretization -- Use supervised discretization to convert numeric attributes to nominal ones.

c. *Naïve Bayes Uptateable*: The Class is used for a Naive Bayes classifier using estimator classes. This is the updateable version of Naïve Bayes. This classifier will use a default precision of 0.1 for numeric attributes when build Classifier is called with zero training instances.

OPTIONS

Debug -- If set to true, classifier may output additional info to the console.

Use KernelEstimator -- Use a kernel estimator for numeric attributes rather than a normal distribution.

Use SupervisedDiscretization -- Use supervised discretization to convert numeric attributes to nominal ones.

2. *Building Function Classifiers Algorithm*

a. *Logistic:* The Class is used for building and using a multinomial logistic regression model with a ridge estimator. If there are k classes for n instances with m attributes, the parameter matrix B to be calculated will be an m*(k-1) matrix. In order to find the matrix B for which L is minimized, a Quasi-Newton Method is used to search for the optimized values of the m*(k-1) variables. Note that before we use the optimization procedure, we 'squeeze' the matrix B into a m*(k-1) vector. Although original Logistic Regression does not deal with instance weights, we modify the algorithm a little bit to handle the instance weights.

OPTIONS

Debug -- Output debug information to the console.

Max Its -- Maximum number of iterations to perform.

Ridge -- Set the Ridge value in the log-likelihood.

b. *Multi Layer Perception* : A Classifier uses the back propagation to classify the instances. It is used for test the neural network.

c. *RBF Network:* The Class that implements a normalized Gaussian radial basis function network. It uses the k-means clustering algorithm to provide the basis functions and learns either a logistic regression (discrete class problems) or linear regression (numeric class problems) on top of that.

d. *SMO:* It Implements the John Platt's sequential minimal optimization algorithm for training a support vector classifier. This implementation globally replaces all missing values and transforms nominal attributes into binary ones. It also normalizes all attributes by default. (In that case the coefficients in the output are based on the normalized data, not the original data --- this is important for interpreting the classifier.) Multi-class problems are solved using pair wise classification.

To obtain proper probability estimates, use the option that fits logistic regression models to the outputs of the support vector machine. In the multi-class case the predicted probabilities are coupled using Hastie and Tibshirani's pairwise coupling method.

OPTIONS

Build LogisticModels -- Whether to fit logistic models to the outputs (for proper probability estimates).

c -- The complexity parameter C.

Cache Size -- The size of the kernel cache (should be a prime number). Use 0 for full cache.

Debug -- If set to true, classifier may output additional info to the console.

Epsilon -- The epsilon for round-off error (shouldn't be changed).

Exponent -- The exponent for the polynomial kernel.

Feature Space Normalization -- Whether feature-space normalization is performed (only available for non-linear polynomial kernels).

Filter Type -- Determines how/if the data will be transformed.

Gamma -- The value of the gamma parameter for RBF kernels.

Lower Order Terms -- Whether lower order polyomials are also used (only available for non-linear polynomial kernels).

Num Folds -- The number of folds for cross-validation used to generate training data for logistic models (-1 means use training data).
Random Seed -- Random number seed for the cross-validation.
Tolerance Parameter -- The tolerance parameter (shouldn't be changed).
Use RBF -- Whether to use an RBF kernel instead of a polynomial one.

e. *Simple Logistic*: The Classifier is for building linear logistic regression models. LogitBoost with simple regression functions as base learners is used for fitting the logistic models. The optimal number of LogitBoost iterations to perform is cross-validated, which leads to automatic attribute selection.
OPTIONS
Debug -- If set to true, classifier may output additional info to the console.
Error On Probabilities -- Use error on the probabilties as error measure when determining the best number of LogitBoost iterations. If set, the number of Logit Boost iterations is chosen that minimizes the root mean squared error (either on the training set or in the cross-validation, depending on useCrossValidation).
Heuristic Stop -- If heuristicStop > 0, the heuristic for greedy stopping while cross-validating the number of LogitBoost iterations is enabled. This means LogitBoost is stopped if no new error minimum has been reached in the last heuristicStop iterations. It is recommended to use this heuristic, it gives a large speed-up especially on small datasets. The default value is 50.
Max BoostingIterations -- Sets the maximum number of iterations for LogitBoost. Default value is 500, for very small/large datasets a lower/higher value might be preferable.
Num BoostingIterations -- Set fixed number of iterations for LogitBoost. If >= 0, this sets the number of LogitBoost iterations to perform. If < 0, the number is cross-validated or a stopping criterion on the training set is used (depending on the value of useCrossValidation).
Use CrossValidation -- Sets whether the number of LogitBoost iterations is to be cross-validated or the stopping criterion on the training set should be used. If not set (and no fixed number of iterations was given), the number of LogitBoost iterations is used that minimizes the error on the training set (misclassification error or error on probabilities depending on errorOnProbabilities).

3. *Building Lazy Classifiers Algorithm*

a. *IB1:* Nearest-neighbour classifier. Uses normalized Euclidean distance to find the training instance closest to the given test instance, and predicts the same class as this training instance. If multiple instances have the same (smallest) distance to the test instance, the first one found is used.
OPTIONS
Debug -- If set to true, classifier may output additional info to the console.
b. *IBK:* Storing and using specific instances improves the performance of several supervised learning algorithms. These include algorithms that learn decision trees, classification rules, and distributed networks. However, no investigation has analyzed algorithms that use only specific instances to solve incremental learning tasks [8]. In this paper, we describe a framework and methodology, called instance-based learning that generates classification predictions using only specific instances. Instance-based learning algorithms do not maintain a set of abstractions derived from specific instances.

This approach extends the nearest neighbor algorithm, which has large storage requirements. We describe how storage requirements can be significantly reduced with, at most, minor sacrifices in learning rate and classification accuracy. While the storage-reducing algorithm performs well on several real-world databases, its performance degrades rapidly with the level of attribute noise in training instances. Therefore, we extended it with a significance test to distinguish noisy instances. This extended algorithm's performance degrades gracefully with increasing noise levels and compares favorably with a noise-tolerant decision tree algorithm.

C. . *KStar:* **K*** is an instance-based classifier, that is the class of a test instance is based upon the class of those training instances similar to it, as determined by some similarity function. It differs from other instance-based learners in that it uses an entropy-based distance function. The use of entropy as a distance measure has several benefits[7]. Amongst other things it provides a consistent approach to handling of symbolic attributes, real valued attributes and missing values. The approach of taking all possible transformation paths is discussed. We describe K*, an instance-based learner which uses such a measure and results are presented which compare favorably with several machine learning algorithms.
d. *LWL*

It uses an instance-based algorithm to assign instance weights which are then used by a specified Weighted Instances Handler. It is used for classification (e.g. using naive Bayes) or regression (e.g. using linear regression).

Option	Description
KNN	How many neighbours are used to determine the width of the weighting function (<= 0 means all neighbours).
classifier	The base classifier to be used.
debug	If set to true, classifier may output additional info to the console.
Nearest NeighbourSearchAlgorithm	The nearest neighbour search algorithm to use (Default: LinearNN).
Weighting Kernel	Determines weighting function. [0 = Linear, 1 = Epnechnikov,2 = Tricube, 3 = Inverse, 4 = Gaussian and 5 = Constant. (default 0 = Linear)].

4. *Building Lazy Classifiers Algorithm*

a. *AdaBoost:* The Class is used for boosting a nominal class classifier using the Adaboost M1 method. Only nominal class problems can be tackled. Often dramatically improves performance, but sometimes overfits.
OPTIONS
Classifier -- The base classifier to be used.
Debug -- If set to true, classifier may output additional info to the console.
Num Iterations -- The number of iterations to be performed.
Seed -- The random number seed to be used.
Use Resampling -- Whether resampling is used instead of reweighting.
Weight Threshold -- Weight threshold for weight pruning.

b. *Attribute Selected Classifier:* Dimensionality of training and test data is reduced by attribute selection before being passed on to a classifier[8].
OPTIONS
Classifier -- The base classifier to be used.
Debug -- If set to true, classifier may output additional info to the console.
Evaluator -- Set the attribute evaluator to use. This evaluator is used during the attribute selection phase before the classifier is invoked.
Search -- Set the search method. This search method is used during the attribute selection phase before the classifier is invoked.
c.*Bagging:* The Class for bagging classifier is used to reduce variance. We can do classification and regression depending on the base learner.

d. *CV Parameter Selection:* This Class is used performing the parameter selection by cross-validation for any classifier.

e. *Classification via Regression:* This Class is used for doing classification using regression methods. Class is binarized and one regression model is built for each class value.
OPTIONS
Classifier -- The base classifier to be used.
Debug -- If set to true, classifier may output additional info to the console.

f. *Decorate:* DECORATE is a meta-learner for building diverse ensembles of classifiers by using specially constructed artificial training examples. Comprehensive experiments have demonstrated that this technique is consistently more accurate than the base classifier, Bagging and Random Forests. Decorate also obtains higher accuracy than Boosting on small training sets, and achieves comparable performance on larger training sets.

g. *Filtered Classifier:* This Class is used for running an arbitrary classifier on data that has been passed through an arbitrary filter. Like the classifier, the structure of the filter is based exclusively on the training data and test instances will be processed by the filter without changing their structure.
OPTIONS
Classifier -- The base classifier to be used.
Debug -- If set to true, classifier may output additional info to the console.
Filter -- The filter to be used.

h. *Grading:* It Implements Grading. The base classifiers are "graded".
OPTIONS
Classifiers -- The base classifiers to be used.
Debug -- If set to true, classifier may output additional info to the console.
Meta Classifier -- The meta classifiers to be used.
Num Folds -- The number of folds used for cross-validation.
Seed -- The random number seed to be used.

i. *Logit boost:* This Class is used for performing additive logistic regression. This class performs classification using a regression scheme as the base learner, and can handle multi-class problems.

j. *MultiBoostAB:* This Class is used for boosting a classifier using the Multi Boosting method. Multi Boosting is an extension to the highly successful AdaBoost technique for forming decision committees. Multi Boosting can be viewed as combining AdaBoost with wagging. It is able to harness both Ada Boost's high bias and variance reduction with wagging's superior variance reduction. Using C4.5 as the base learning algorithm, Multi-boosting is demonstrated to produce decision committees with lower error than either AdaBoost or wagging significantly more often than the reverse over a large representative cross-section of UCI data sets. It offers the further advantage over AdaBoost of suiting parallel execution.

K. *Multi class classifier:* A meta classifier is used for handling multi-class datasets with 2-class classifiers. This classifier is also capable of applying error correcting output codes for increased accuracy.

OPTIONS
Classifier -- The base classifier to be used.
Debug -- If set to true, classifier may output additional info to the console.
Method -- Sets the method to use for transforming the multi-class problem into several 2-class ones.
Random Width Factor -- Sets the width multiplier when using random codes. The number of codes generated will be thus number multiplied by the number of classes.
Seed -- The random number seed to be used.

l. *Multischeme:* This Class is for selecting a classifier from among several using cross validation on the training data or the performance on the training data. Performance is measured based on percent correct (classification) or mean-squared error (regression).
OPTIONS
Classifiers -- The classifiers to be chosen from.
Debug -- Whether debug information is output to console.
Num Folds -- The number of folds used for cross-validation (if 0, performance on training data will be used).
Seed -- The seed used for randomizing the data for cross-minChunkSize and grow twice as large for as many times as they are less than or equal to the maximum size.

Min ChunkSize -- The minimum number of instances to train the base learner with.
Pruning Type -- The pruning method to use within each committee. Log likelihood pruning will discard new models

m. *Stacking:* It combines the several classifiers using the stacking method.
OPTIONS
Classifiers -- The base classifiers to be used.
Debug -- If set to true, classifier may output additional info to the console.
Meta Classifier -- The meta classifiers to be used.
Num Folds -- The number of folds used for cross-validation.
Seed -- The random number seed to be used.

n. *StackingC:* It requires meta classifier to be a numeric prediction scheme.

OPTIONS
Classifiers -- The base classifiers to be used.
Debug -- If set to true, classifier may output additional info to the console.
Meta Classifier -- The meta classifiers to be used.

5. *Building Rule Classifiers*

a. *Conjunction Rule:* This class implements a single conjunctive rule learner that can predict for numeric and nominal class labels. A rule Consists of antecedents "AND"ed together and the consequent (class value) for the classification/regression. In this case, the consequent is the distribution of the available classes (or mean for a numeric value) in the dataset. If the test instance is not covered by this rule, then it's predicted using the default class distributions/value of the data not covered by the rule in the training data [6]. This learner selects an antecedent by computing the Information Gain of each antecedent and prunes the generated rule using Reduced Error Pruning (REP) or simple pre-pruning based on the number of antecedents. For classification, the Information of one antecedent is the weighted average of the entropies of both the data covered and not covered by the rule.
For regression, the Information is the weighted average of the mean-squared errors of both the data covered and not covered by the rule.
In pruning, weighted average of the accuracy rates on the pruning data is used for classification while the weighted average of the mean-squared errors on the pruning data is used for regression.
OPTIONS
Debug -- If set to true, classifier may output additional info to the console.
Exclusive -- Set whether to consider exclusive expressions for nominal attribute splits.
Folds -- Determines the amount of data used for pruning. One fold is used for pruning, the rest for growing the rules.
Min No -- The minimum total weight of the instances in a rule.
Num Antds -- Set the number of antecedents allowed in the rule if pre-pruning is used. If this value is other than -1, then pre-pruning will be used, otherwise the rule uses reduced-error pruning.

b. *Decision Table:* The Class for building and using a simple decision table majority classifier. It evaluates feature subsets using best-first search and can use cross-validation for evaluation. There is a set of methods that can be used in the search phase (E.g.: Best First, Rank Search, Genetic Search) and we may also use LBk to assist the result. In this experiment, we
Choose the cross Val = 1; search Method = Best First and useIBk = False

c. *JRip:* This class implements a propositional rule learner, Repeated Incremental Pruning to Produce Error Reduction (RIPPER), which was proposed by William W. Cohen as an optimized version of IREP.

d. *NNge:* Nearest-neighbor-like algorithm using non-nested generalized exemplars (which are hyperrectangles that can be viewed as if-then rules).

e. *OneR:* Class for building and using a 1R classifier; in other words, uses the minimum-error attribute for prediction, discretizing numeric attributes.

f. *PART:* The Class for generating a PART decision list. It uses separate-and-conquer. Builds a partial C4.5 decision tree in each iteration and makes the "best" leaf into a rule.

g. *Ridor:* It is the implementation of a RIpple-DOwn Rule learner. It generates a default rule first and then the exceptions for the default rule with the least (weighted) error rate. Then it generates the "best" exceptions for each exception and iterates until pure. Thus it performs a tree-like expansion of exceptions.
The exceptions are a set of rules that predict classes other than the default. IREP is used to generate the exceptions.

h. *ZeroR:* The Class is for building and using a 0-R classifier. It predicts the mean (for a numeric class) or the mode (for a nominal class).
OPTIONS
debug -- If set to true, classifier may output additional info to the console.

6. *Building Tree Classifiers Algorithm*

a. *J48:* The Class is for generating a pruned or un pruned C4.5 decision tree.
OPTIONS
Binary Splits -- Whether to use binary splits on nominal attributes when building the trees.
Confidence Factor -- The confidence factor used for pruning (smaller values incur more pruning).
Debug -- If set to true, classifier may output additional info to the console.
Min NumObj -- The minimum number of instances per leaf.
Num Folds -- Determines the amount of data used for reduced-error pruning. One fold is used for pruning, the rest for growing the tree.
Reduced ErrorPruning -- Whether reduced-error pruning is used instead of C.4.5 pruning.
Save Instance Data -- Whether to save the training data for visualization.

Seed -- The seed used for randomizing the data when reduced-error pruning is used.
Sub tree Raising -- Whether to consider the subtree raising operation when pruning.
Un pruned -- Whether pruning is performed.
Use Laplace -- Whether counts at leaves are smoothed based on Laplace.

b. *LMT:* Classifier for building 'logistic model trees', which are classification trees with logistic regression functions at the leaves. The algorithm can deal with binary and multi-class target variables, numeric and nominal attributes and missing values.

c. *NB Tree:* The Class is for generating a decision tree with naive Bayes classifiers at the leaves.
OPTIONS
Debug -- If set to true, classifier may output additional info to the console.

d. *Random Forest:* The Class is used for constructing a forest of random trees.
OPTIONS
Debug -- If set to true, classifier may output additional info to the console.
Num Features -- The number of attributes to be used in random selection (see RandomTree).
Num Trees -- The number of trees to be generated.
Seed -- The random number seed to be used.

e. *Random Tree:* The Class for constructing a tree that considers K randomly chosen attributes at each node. It performs no pruning.
OPTIONS
K Value -- Sets the number of randomly chosen attributes.
Debug -- Whether debug information is output to the console.
Min Num -- The minimum total weight of the instances in a leaf.
Seed -- The random number seed used for selecting attributes.

f. *Decision Stump:* The Class for building and using a decision stump. It is used in conjunction with a boosting algorithm. Missing is treated as a separate value.
OPTIONS
Debug -- If set to true, classifier may output additional info to the console.M5P

g. *REP Tree:* It is a Fast decision tree learner.
Option Description
Debug: If set to true, classifier may output additional info to the console.
Max Depth: The maximum tree depth (-1 for no restriction).
Min Num: The minimum total weight of the instances in a leaf.
minVariance Prop: The minimum proportion of the variance on all the data that needs to be present at a node in order for splitting to be performed in regression trees.
no Pruning: Whether pruning is performed.
Num Folds: Determines the amount of data used for pruning. One fold is used For pruning, the rest for growing the rules.
Seed: The seed used for randomizing the data.

D. *Performance Evaluation*

The performance of the classifier will be evaluated by using the heart disease data that contains the 107 instances and 10 folds cross validation [7]. The best classifier have evaluated by using prediction accuracy [2]. The following table contains the Prediction accuracy, Correctly Classified and Incorrectly Classified Instance for each classifier algorithm and the result of F_measures of the classes for each classifier algorithm. It contains the Sick and buff classes having the separate F-measure. The F-measure is calculated by using the precision and Recall.

The Evaluation graph shows the performance of classifiers based on prediction accuracy. The classifiers of Naïve Bayes and Naïve Bayes updatable in Bayes classifiers, SMO in Function Classifiers, IB1and IBK in Lazy Classifiers, MultiBoostAB, and Multi Class Classifier in Meta Classifier,

Decision Table in Rule Classifiers, NB Tree in Tree Classifiers is having the better performance comparing to other classifier algorithms.

Table I. Prediction Accuracy of Classifiers

Classifier Category	Classifier Algorithms	Measur<es		
		Correctly Classified Instance	In correctly Classified Instance	Prediction Accuracy
Bayes	BayesNet	93	14	86.9
	Naïve Bayes	95	12	88.8
	Naïve Bayes Updateable	95	12	88.8
Function	Logistics	90	17	84.1
	Multilayer Perception	87	20	81.3
	RBF Network	91	16	85.0
	SMO	92	15	86.0
	Simple Logistics	91	16	85.0
Lazy	IB1	89	18	83.2
	IBK	89	18	83.2
	KStar	87	20	81.3
	LWL	87	20	81.3
Misc	HyperPipes	65	42	60.7
	VFI	90	17	84.1
Meta	AdaBoost	87	20	81.3
	Attribute selected classifier	88	19	82.2
	Bagging	88	19	82.2
	CV Parameter Selection	56	51	52.3
	Classification Via Regression	89	18	83.2
	Decorate	85	22	79.4
	Filtered Classifier	83	24	77.6
	Grading	56	51	52.3
	Logit Boost	87	20	81.3
	MultiBoostAB	90	17	84.1
	Multi Class Classifier	90	17	84.1
	MultiScheme	56	51	52.3
	Ordinal Class Classifier	83	24	77.6
	Raced Incremental Logit Boost	56	51	52.3
	Random Committ	86	21	80.4
	Stacking	56	51	52.3
	StackingC			
Rule	Conjunctive Rule	89	18	83.2
	Decision Table	95	12	88.8
	JRip	91	16	85.0
	NNge	90	17	84.1
	OneR	86	21	80.4
	PART	88	19	82.2
	Ridor	85	22	79.4
	ZeroR	56	51	52.3
Tree	Decision Stump	89	18	83.2
	J48	83	24	77.6
	LMT	90	17	84.1

	NB Tree	97	10	90.7
	REP Tree	85	22	79.4
	Random Forest	88	19	82.2
	Random Tree	83	24	77.6

Table II. F_Measures of Sick and Buff Classes

Classifier Category	Classifier Algorithms	F-Measures for Sick Class	F-Measures for Buff Class
Bayes	BayesNet	85.4	88.1
	Naïve Bayes	88	89.5
	Naïve Bayes Updateable	88	89.5
Function	Logistics	82.8	85.2
	Multilayer Perception	80	82.5
	RBF Network	83.7	86.2
	SMO	85.1	86.7
	Simple Logistics	84	86
Lazy	IB1	82.7	83.6
	IBK	82.7	83.6
	KStar	79.6	82.8
	LWL	78.7	83.3
Misc	HyperPipes	69.1	46.2
	VFI	83.5	84.7
Meta	AdaBoost	80.4	82.1
	Attribute selected classifier	81.2	83.2
	Bagging	81.2	83.2
	CV Parameter Selection	0	68.7
	Classification Via Regression	82	84.2
	Decorate	76.6	81.7
	Filtered Classifier	75.5	79.3
	Grading	0	68.7
	Logit Boost	80.4	82.1
	MultiBoostAB	82.8	85.2
	Multi Class Classifier	82.8	85.2
	MultiScheme	0	68.7
	Ordinal Class Classifier	76	78.9
	Raced Incremental Logit Boost	0	68.7
	Random Committ	79.6	81.1
	Stacking StackingC	0	68.7
Rule	Conjunctive Rule	81.3	84.7
	Decision Table	88	89.5
	JRip	84	86
	NNge	83.2	85
	OneR	78.8	81.7
	PART	80.8	83.5
	Ridor	77.6	81
	ZeroR	0	68.7
Tree	Decision Stump	81.3	84.7
	J48	76	78.9
	LMT	82.8	85.2
	NB Tree	90	91.2
	REP Tree	78	80.7
	Random Forest	81.2	83.2
	Random Tree	75.5	79.3

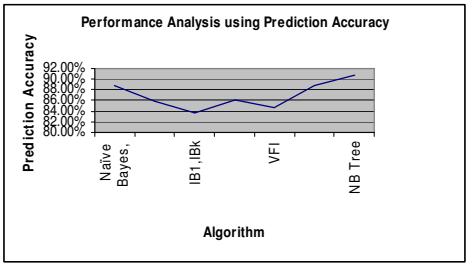

Fig 2. Evaluation Graph

Table III. Prediction Accuracy of Classifiers

Classifiers Category	Classifiers Algorithm	Prediction Accuracy
Bayes	Naïve Bayes, Naïve Bayes Updateable	88.8%
Function	SMO	86%
Lazy	IB1,IBk	83.6%
Meta	MultiBoostAB, Multi Class Classifier	86.1%
Misc	VFI	84.7%
Rule	Decision Table	88.8%
Tree	NB Tree	90.7%

III. CONCLUSION

In medical industry having the large amount of useful data .In this data is used for many purposes, here the heart attack prediction data is used for find the performance of classifiers. In final result shows the performance of classifier algorithm using prediction accuracy. In this result shows the evaluation of F-measure for two classes of sick and buff classes and the prediction accuracy of classifier s. The comparison result shows that, the NB Tree having the highest prediction Accuracy. In Future this research will be expanded in to numeric class. The next expansion of this research is to use the clustering algorithm in heart disease prediction data and find the performance of this clustering algorithm.

IV.ACKNOWLEDGMENT

I Express my sincere thanks to my guide Dr.V.MuraliBhaskaran,M.E.,(Ph.D), for his continuous support and co-operation for doing my research.

I extend my thanks to Dr.R.Nedunchezian, M.E.,(Ph.D) and Dr.T.Purusothaman, M.E.,(Ph.D) for their encouragement of my research work.

I thank to my HOD and all my friends of Computer science and Engineering Department for their encouragement.

Finally, i place my humble accolates to my family members for their moral support

V.REFERENCES

[1] Varun Kumar, Nisha Rathee,"Knowledge Discovery from Database using an Integration of clustering and Classification", IJACSA, vol 2 No.3,PP. 29-33,March 2011.

[2] Ritu Chauhan, Harleen Kaur, M.Afshar Alam, "Data Clustering Method for Discovering Clusters in Spatial Cancer Databases", International Journal of Computer Applications (0975 – 8887) Volume 10– No.6, November 2010.

[3] G.Karraz,G.Magenes,"Automatic Classification of Heart beats using Neural Network Classifier based on a Bayesian Frame Work", IEEE, Vol 1,2006.

[4] N.A.Setiawan,A.F.M.Hani,"Missing Attribute Value Prediction Based on Artificial Neural Network and Rough Set Theory", IEEE, Vol 1, pp.306-310,2008.

[5]] Sellappan Pandian, Rafigh Awang,"Heart Disease Prediction System using Data Mining Techniques",IEEE Computer, Vol 7, PP.295-304, August 2008.

[6] Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005).

[7] Ian H.Witten, et al, "Weka: Practical Machine Learning Tools and Techniques with Java implementations," Working Paper 99/11, Department of Computer.

[8] Weka – Data Mining Machine Learning Software, http://www.cs.waikato.ac.nz/ml/.