Chapter 3

3 Classification of Microarray Cancer Data Using Ensemble Approach

In this chapter a method referred to as *SD-EnClass* is presented, for combining classifiers from different classification families into an ensemble, based on a simple estimation of each classifier's class performance. The experimental comparison of the base classifiers J48, NB, IBK and the ensemble method are performed on nine microarray cancer datasets. The results show that the proposed model improves classification accuracy as opposed to simply selecting the best classifier in the combination. A comparison of the performance of the ensemble model is also done as a comparison with Bagging, Boosting and Stack Generalization. In the second stage, a meta-ensemble is constructed by combining the results of the proposed method with the results of Boosting, Bagging and Stacking using the combining method proposed, to obtain results which are significantly better than using Boosting, Bagging or Stacking alone.

3.1 Introduction

Many methods have been proposed in microarray classification, including subspace clustering [231,232] and ensemble methods such as Bagging and Boosting [42,76], for building classifiers from microarray gene expression data to be used for classifying unknown data. An ensemble of classifiers is a set of classifiers whose individual predictions are combined in some way to classify new examples, with an intention of improving classification accuracy over an average classifier. Since it is not known *apriori* which classifier is best for a particular classification problem, an ensemble reduces the risk of selecting a poorly performing classifier.

In view of the significant improvement in the classification accuracy through combining classifiers, Breiman [42] introduced *Bagging*, which combines outputs from decision tree models generated from bootstrap samples (with replacement) of a training dataset. Models are combined by simple voting. Fruend and Schapire [53] introduced *Boosting*, an iterative process of weighing more heavily the incorrectly classified cases by decision tree models, and then combining all the models generated during the process. Arcing [54] is a form of boosting that, like the original boosting, weighs incorrectly classified cases more heavily, but instead of the Fruend and Schapire's [53] formula for weighing, weighted random samples are drawn from the training data. Wolpert [47] used regression to combine neural network models which was later known as *Stacking*. These are just a few of the well known algorithms currently described in the literature, and many more methods have been developed by researchers as well. A survey by Kiliç and Tan [233] provides an insight to algorithms that can handle binary classification problems.

3.2 Background

A classifier h is a function that maps a vector of attribute values x (also called *example*) to classes in $C= \{C_1, C_2, \ldots, C_n\}$. An ensemble classifier consists of a set of classifiers $H = \{h_1, h_2, \ldots\ldots, h_k\}$ whose output is dependent on the outputs of the constituent classifiers [234].

The performance of an ensemble mostly depends on the individual performance of the classifiers present in the ensemble. Two key requirements of the classifiers forming the ensemble are:

- diversity of classifiers in nature
- accuracy in the classifier predictions

Similar classifiers usually make similar errors, so forming an ensemble with similar classifiers would not improve the classification rate. Also, presence of a poorly performing classifier may cause performance deterioration in the overall performance. Similarly, presence of a classifier that performs much better than all of the other available base classifiers may cause degradation in the overall performance. Another important factor is the amount of correlation among the incorrect classifications made by each classifier. If the consistent classifiers tend to misclassify the same instances, then combining their results will have no benefit. In contrast, a greater amount of independence among the classifiers can result in errors by individual classifiers being overlooked when the results of the ensemble are combined.

3.3 Construction of Ensembles

The task of constructing an ensemble can be broken down into two subtasks: (i) selection of a diverse set of base level models or classifiers with consistently acceptable performance, and (ii) appropriate combination of their predictions with due weightage. Next, these two subtasks along with some other important factors are discussed.

3.3.1 Classifier Selection

In supervised classification, classifiers are trained to become experts in some local area of the total feature space. For each example (say x), a classifier is identified which is most likely to produce the correct classification label, as shown in *Figure* 3.1. The output of the classifiers identified as the best for a given classification problem, is selected. Usually, the input sample space is partitioned into smaller areas and each classifier (say h_i) learns the example in each area. It is similar to the *divide and conquer* approach. Here, multiple local experts may be nominated to make the decision. Finally, a subset of classifiers performing consistently well with high classification accuracy for several real-life datasets is selected as the base classifiers.

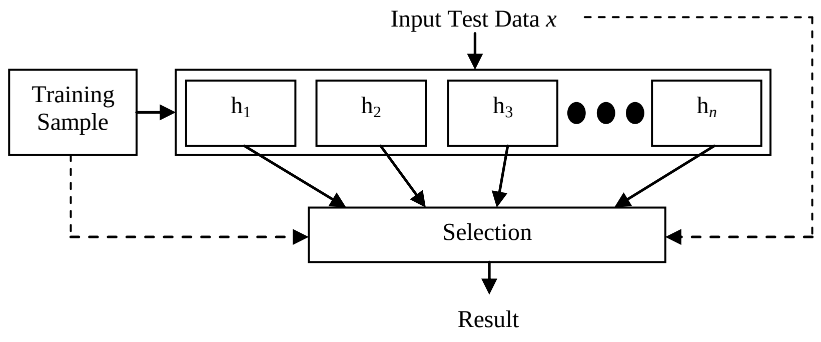


Figure 3.1: Classifier Selection

3.3.2 Fusion

Here, the outputs of many different classifiers are mixed instead of extracting a single best classifier. Each classifier in the ensemble has some knowledge of the entire feature space and tries to solve the same classification problem using different methods based on different training sets, classifiers or parameters. The final output is determined by fusing the decisions of the individual classifiers as shown in *Figure* 3.2. The fusion can be at the data level as well as the decision level. All the classifiers in the ensemble are trained over the entire feature space.

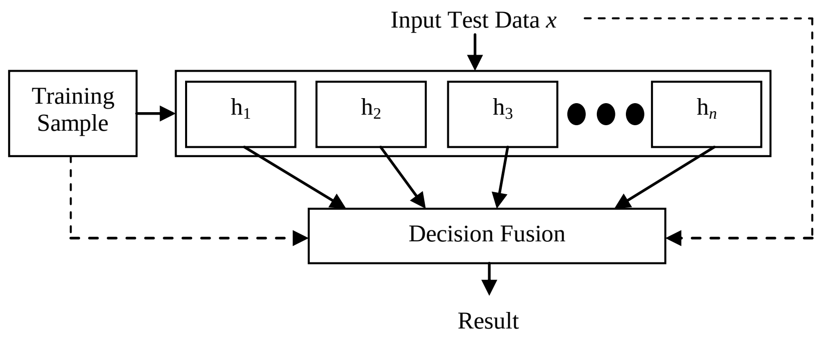


Figure 3.2: Fusion

3.4 Existing Methods

Several methods have been developed for the construction of ensembles. Some methods are general and they can be applied to any learning algorithm whereas others are specific to particular algorithms. Some of the basic approaches for the construction of ensembles are as follows.

3.4.1 Different Classifier Models

For effectiveness, several types of learning algorithms from different backgrounds e.g., decision tree, neural network and nearest neighbour can be used. However, the same classifier can also be used with a slight change in the user-defined parameters, leading to significant variation in classification results.

3.4.2 Different Feature Subsets

Classifiers are built using different subsets of features of the training dataset. It works only when some redundancy in features is present on the training dataset. Both the *deterministic* and *random* approaches can be used for selecting different feature subsets of input data. In the *deterministic* approach a prior knowledge about the input data is required, whereas, the *random* approach uses random subspace method for selecting the different feature subsets. Here, each classifier is selected on a random space, and a feature in the subset is selected with a probabilistic approach. Random forests, which uses decision trees, is an example of this method. However, a common problem with this random method is that some subspaces lack information and as a result may not give good performance.

3.4.3 Different Training Sets

One learning algorithm is run on different random sub-samples of training data to produce different classifiers. It works well for unstable learners, i.e., output classifier undergoes major changes given only small changes in training data. Random sub-samples of the training data can be generated using *re-sampling* and *re-weighing*. Bagging and wagging use re-sampling, while Boosting and Arcing uses re-weighing of the training dataset.

3.4.4 Different Combination Schemes

Different classifiers are trained on the training data and their outputs are combined using different combination schemes like majority voting, algebraic combiners, etc. to get the final output.

3.5 Ensemble Combination Methods

An ensemble of classifiers can be trained simply on different subsets of the training data, different parameters of the classifiers, or even with different subsets of features as in random subspace models. The classifiers can then be combined using one of the combination methods as shown in *Figure* 3.3.

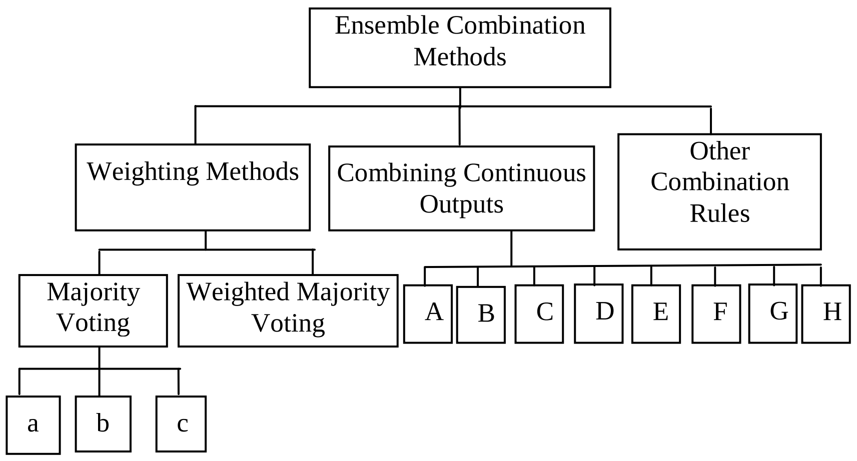


Figure 3.3: Ensemble Combination Methods

3.5.1 Weighting Methods

When classifiers are combined by assigning weights to their classification, the weight determines the contribution made by the classifier to the final ensemble. Let us denote the decision of classifier h_t on class c_j as $d_{t,j}$ such that $d_{t,j}$ is 1 if h_t selects c_j and 0 otherwise. The ensemble then chooses class J that receives the largest total vote.

Majority Voting: In this approach each classifier has equal vote and the most popular classification is the one chosen by the ensemble as a whole [235]. The most basic variant is the plurality vote, where the class with the most votes wins.

$$\sum_{t=1}^{T} d_{t,J}(x) = max_{j=1}^{C} \sum_{t=1}^{T} d_{t,j} \qquad (3.1)$$

where:

J = class selected
$d_{t,j}(x)$ = support given by the t^{th} classifier to the j^{th} class for the instance x.
T = total number of classifiers.

In general, there are three versions of majority voting as shown below.

a) One on which all classifiers agree (unanimous voting).
b) Predicted by at least one more than half the number of classifiers (simple majority).
c) One that receives the highest number of votes, whether or not the sum of those votes exceeds 50% (plurality voting or just majority voting).

However, none of these versions is free from the common limitations of majority voting.

Weighted Majority Voting: If one has evidence that certain experts are more qualified than others, weighing the decisions of those qualified experts more heavily may further improve the overall performance. Further, it is assumed that the classifiers are assigned weights according to their performances such that classifier h_t is assigned weight w_t.

The total weights received by a class is the sum of the product of the weights the classifiers, w_t and their respective decisions, $d_{t,j}$. The class receiving the highest weighted vote is selected as the final decision of the ensemble. Thus, according to this assumption an ensemble will select class J if equation (3.2) holds [71]:

$$\sum_{t=1}^{T} w_t\, d_{t,J}(x) = max_{j=1}^{C} \sum_{t=1}^{T} w_t\, d_{t,j} \qquad (3.2)$$

where:
w_t = weight of the t^{th} classifier.

3.5.2 Combining Continuous Outputs

Continuous output is interpreted as the degree of support given to a class by a classifier, usually accepted as the value of the posterior probability for that class [235]. Algebraic combiners are used to merge the decisions of the classifiers in continuous output format, following the convention as given above and $\mu_j(x)$ = total support for the j^{th} class for instance x.

A. ***Sum Rule***: The total support for a class is calculated as the sum of the supports given to that class by all the classifiers. After calculating the total supports for all the classes, the class with the highest support is selected as the final output. The sum rule has found to be useful in reducing noise in large sets of weak classifiers [236,237].

$$\mu_j(x) = \sum\nolimits_{t=1}^{T} d_{t,j}\,(x) \tag{3.3}$$

B. ***Mean Rule***: This rule is similar to the sum rule but the total support is normalized by $\frac{1}{T}$ (T=number of classifiers) [238]

$$\mu_j(x) = \frac{1}{T}\sum\nolimits_{t=1}^{T} d_{t,j}\,(x) \tag{3.4}$$

C. ***Weighted Sum Rule***: By this rule, the support for a class is calculated by the sum of the product of the classifiers weight and their respective supports [239].

$$\mu_j(x) = \sum\nolimits_{t=1}^{T} w_t d_{t,j}\,(x) \tag{3.5}$$

D. ***Product Rule***: Here, supports provided by the classifiers to a particular class are multiplied to obtain the final support for that class. This rule is very sensitive to the pessimistic classifiers as a low support can remove any chance of getting selected by that class [240,241]. The rule assumes noise free and reliable confidence estimates. It fails if these estimates may be accidentally zero or very small.

$$\mu_j(x) = \prod_{t=1}^{T} d_{t,j}\,(x) \tag{3.6}$$

E. ***Maximum Rule***: As the name suggests, this rule selects the maximum of all the supports of the different classifiers for a particular class. The maximum rule, however, fails for simple classifiers that are not sensitive to small differences that are detected by more complicated classifiers [242].

$$\mu_j(x) = max_{t=1}^{T}\{d_{t,j}(x)\} \tag{3.7}$$

F. ***Minimum Rule***: This rule selects the minimum of all the supports of the different classifiers for a particular class [242].

$$\mu_j(x) = min_{t=1}^{T}\{d_{t,j}(x)\} \tag{3.8}$$

G. ***Median Rule***: This rule selects the median of the supports of the different classifiers for a particular class [243].

$$\mu_j(x) = median_{t=1}^{T}\{d_{t,j}(x)\} \tag{3.9}$$

H. ***Generalized Mean Rule***: Many of the above rules are in fact special cases of the generalized mean rule, which is as follows [71]

$$\mu_{j,r}(x) = \left[\frac{1}{T}\sum_{t=1}^{T} d_{t,j}\,(x)^r\right]^{\frac{1}{r}} \tag{3.10}$$

From the above discussion it is apparent that only under very strict conditions a fixed rule is really the best combination. They will certainly be sub-optimal if the base classifiers generate unreliable confidences (e.g. caused by a small training set or by overtraining). But if the available set of objects is sufficiently large, then an improved result may be found by carefully training the combining classifier.

The methods of combination are summarized in *Table* 3.1.

3.6 Related Work

In the past decade, researchers have devoted their efforts to the study of ensemble decision tree methods for microarray classification. Ensemble decision tree methods combine decision trees generated from multiple training datasets by re-sampling the training dataset. Bagging, Boosting and Stacking are some of the well-known ensemble methods in the machine learning field. How well an ensemble works is dependent on the performance of the individual classifiers present in the ensemble basket. In order to obtain an improvement the base classifiers need to be accurate (better than chance) and diverse from each other [244].

Table 3.1: Summary of rules of Combination Methods

Approach	Method	Formula	Effectiveness
Class Label Combinati on	Majority Voting	$\sum_{t=1}^{T} d_{t,J}(x) = max_{j=1}^{C} \sum_{t=1}^{T} d_{t,j}$	Gives average performance when majority does not give accurate prediction
	Weighted Majority Voting	$\sum_{t=1}^{T} w_t\, d_{t,J}(x) = max_{j=1}^{C} \sum_{t=1}^{T} w_t\, d_{t,j}$	Performs well only if the weights of the classifiers are assigned precisely
Continuou s Output Combinati on	Sum Rule	$\mu_j(x) = \sum_{t=1}^{T} d_{t,j}(x)$	Gives average performance when majority does not give accurate prediction
	Weighted Sum Rule	$\mu_j(x) = \sum_{t=1}^{T} w_t d_{t,j}(x)$	Performs well only if the weights of the classifiers are assigned precisely
	Mean Rule	$\mu_j(x) = \frac{1}{T}\sum_{t=1}^{T} d_{t,j}(x)$	Performance is significantly affected by outliers
	Product Rule	$\mu_j(x) = \prod_{t=1}^{T} d_{t,j}(x)$	Sensitive to low probability value
	Maximum Rule	$\mu_j(x) = max_{t=1}^{T}\{d_{t,j}(x)\}$	Chooses the most optimistic value
	Minimum Rule	$\mu_j(x) = min_{t=1}^{T}\{d_{t,j}(x)\}$	Performance is significantly affected by outliers
	Median Rule	$\mu_j(x) = median_{t=1}^{T}\{d_{t,j}(x)\}$	Performance is significantly affected by outliers
	Generalized Rule	$\mu_{j,r}(x) = \left[\frac{1}{T}\sum_{t=1}^{T} d_{t,j}(x)^r\right]^{\frac{1}{r}}$	Affected by Outliers

where:

$d_{t,j}(x)$ = support given by the t^{th} classifier to the j^{th} class for the instance x.

w_t = weight of the t^{th} classifier.

T = total number of classifiers.

$\mu_j(x)$ = total support for the j^{th} class for instance x.

Diversity is necessary because if a classifier makes a misclassification, there may be another classifier that may compensate for it by correctly classifying the misclassified sample.

Many researchers have also made an objective comparison of ensemble methods to demonstrate their advantage using different parameters [61] on microarray data.

Ge and Wong [62] made a comparison of a single classifier of decision trees with six ensemble methods, *viz*. random forests, stacked generalization, bagging, Adaboost, LogitBoost, and Multiboost using three different feature selection schemes. Statnikov *et al.* [245] compared random forests with SVM for microarray-based cancer classification across 22 datasets. Analysis of gene microarray data at the pathway level [246] is also a very active research topic. Bertoni *et al.* [247] investigated the use of random subspace ensembles of SVMs. Re and Valentini [248] showed that simple ensemble methods can obtain results comparable with state-of-the-art data integration methods for gene function prediction.

An ensemble framework based on the support vector machine that integrates diverse datasets in the context of the Gene Ontology hierarchy was developed by Guan *et al.* [249] to confirm functions for a mitochondrial protein of Saccharomyces cerevisiae. Cesa-Bianchi *et al.* [250] experimentally showed that the key factors for the success of hierarchical ensemble methods are the integration among multilabel hierarchical, data fusion, and cost-sensitive approaches, as well as the strategy of selecting negative examples for the purpose of gene functional inference.

3.7 Base Classifiers

A main task of microarray classification is to build a classifier from historical microarray gene expression data, and then use the classifier to classify future coming data. Many methods have been used in Microarray classification, and typical methods are Support Vector Machines (SVMs) [251,252], k-nearest neighbour classifier [253], C4.5 decision tree [254,255], rulebase classification method [253] and ensemble methods, such as bagging and boosting [42,76].

The classifiers selected for our ensemble are J48 (decision tree), IBK (instance based learner) and Naive Bayes (probabilistic) and they have been selected as base classifiers based on the following grounds.

- all these classifiers performed consistently well over several real-life UCI datasets
- they are from three different classification algorithms families
- they belong to three different states i.e., stable, unstable and probabilistic.

3.7.1 J48

Decision tree J48 implements Quinlan's C4.5 algorithm [18] for generating a pruned or unpruned C4.5 tree. Decision trees are built from a set of labelled training data using the concept of information entropy. Based on each attribute of the data, a decision can be arrived at by splitting the data into smaller subsets.

J48 examines the normalized information gain (difference in entropy) that results from choosing an attribute for splitting the data. To arrive at a decision, the attribute with the highest normalized information gain is used. Then the algorithm recursively moves on to the smaller subsets and the splitting procedure stops if all instances in a subset belong to the same class. Next, a leaf node is created in the decision tree indicating the class. In case none of the features give any information gain, then J48 creates a decision node higher up in the tree using the expected value of the class.

J48 can handle both continuous and discrete attributes, training data with missing attribute values and attributes with differing costs. It also provides an option for pruning trees after creation.

3.7.2 IBk

Instance-based classifiers [20] such as the *k*NN classifier operate on the assumption that classification of unknown instances can be done by relating the unknown to the known according to some distance/similarity function. The probability of two instances far apart in the *instance space* (as defined by the appropriate *distance function*) belonging to the same class is less likely than two closely situated instances.

The algorithm computes the *k* closest neighbours of an instance of an unknown class and the class is assigned by voting among those neighbours. To prevent ties, the value of *k* is taken as an odd number for binary classification. Plurality voting or majority voting is used for multiple classes but the latter can sometimes result in no class being assigned to an instance, while the former can result in classifications being made with very low support from the neighbourhood. Another option is to weight each neighbour by an inverse function of its distance to the instance being classified.

3.7.3 Naïve Bayes (NB)

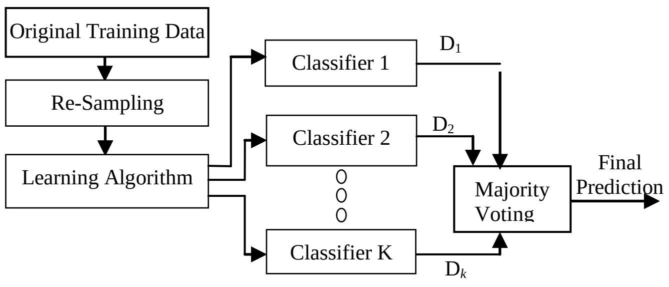
Naïve Bayes classifier [21] is a probabilistic classifier based on the Bayes theorem assuming a strong (Naïve) independence of attributes. The classifier is termed naive since it relies on two important simplifying assumptions, (i) the predictive attributes are conditionally independent given the class, and (ii) no hidden or latent attributes influence the prediction process. The method is designed for use in supervised classification tasks in which the performance goal is to accurately predict the class of the test instances in which the training instances include class information. Naïve Bayes classifiers take into consideration that all attributes (features) independently contribute to the probability of a certain decision. It analyses independently all the attributes of the data with equal importance and hence Naïve Bayes classifiers can be trained very efficiently in a supervised learning setting.

3.8 Ensemble Methods

The ensemble approach has become a leading technique in supervised or unsupervised analysis problems, due to its ability to improve the results obtained from base classifiers and simple clustering algorithms. It has been very effective in combining independent, diversified models for the purpose of improving accuracy in prediction. The combination process integrates information from all partitions in the ensemble, so that possible errors of the individual algorithms could be compensated. That way the consensus obtained from a set of partitions of the same dataset may represent a better solution.

3.8.1 Bagging

Bagging [42] is an ensemble method where the same learning algorithm is used for different training datasets to obtain different classifiers. The diversity amongst the training datasets is achieved by a bootstrap technique used to re-sample the training dataset. As shown in *Figure* 3.4, each classifier is then trained on a re-sample of instances, which then assigns a predicted class to this set of instances. The individual classifiers' predictions (having equal weightage) are then combined by taking majority voting.

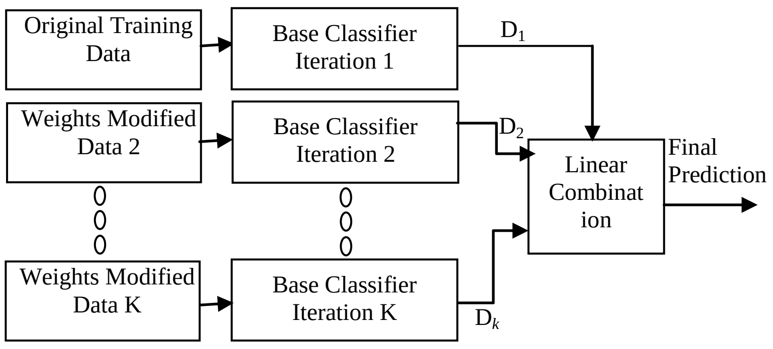


D_i :- Class Prediction by the i^{th} Classifier

Figure 3.4: Block diagram of multiple classifier systems based on Bagging

3.8.2 Boosting

Boosting [53] uses a re-sampling technique different from Bagging. In this case a new training dataset is generated according to its sample distribution. The first classifier is constructed from the original dataset where every sample has an equal weight (*Figure* 3.5). In the succeeding training dataset, the weight is reduced if the sample has been correctly classified otherwise it is increased if the samples are misclassified. In the committee decision, a weighted voting method is used so that a more accurate classifier is given greater weightage than a less accurate classifier.



D_i : - Class Prediction by the ith Classifier

Figure 3.5: Block diagram of multiple classifier systems based on Boosting

3.8.3 Stacked Generalization

Stacked generalization (or stacking) proposed by Wolpert [47], performs its task in two phases (*Figure* 3.6): (i) the *layer*-1 base classifiers are trained using bootstrapped samples of the *level*-0 training dataset and (ii) the outputs of *layer*-1 are then used to train a *layer*-2 *meta classifier*. The purpose is to check whether the training data have been properly learned. For example, if a particular classifier incorrectly learned a certain region of the feature space, and hence consistently misclassifies instances coming from that region, then the *Level-2* classifier may be able to learn this behavior, and along with the learned behaviors of other classifiers, it can correct such improper training. Polikar [235] proposed to use class probabilities rather than class labels as the output in the *level*-1 dataset, so as to improve the stacking performance.

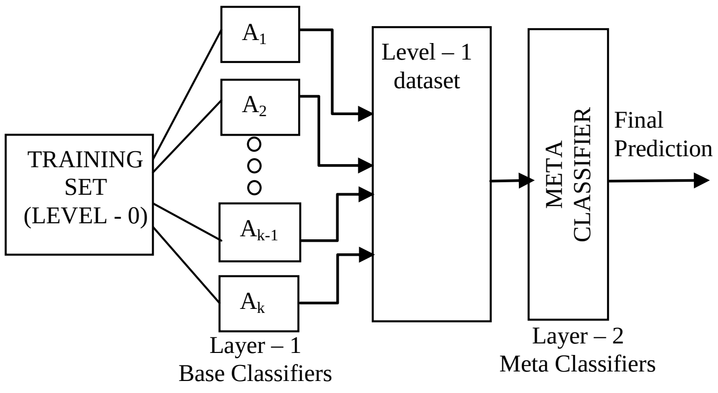


Figure 3.6: Multiple classifiers based on Stack Generalization

A summary of the existing ensemble methods is summarized in *Table* 3.2

Table 3.2: Summary of the Existing Ensemble Approaches

Method	Training Approach	Classifiers	Decision Fusion	Classifier Priority	Input Parameter	Pros	Cons
Bagging	ReSampling	Unstable learner trained over re-sampled sets outputs different models	Majority Voting	No	Training data, no of classes, no of dimensions, no of iterations	Simple and easy to understand and implement	Accuracy value lower than other ensemble approaches
Boosting	ReSampling	Weak learner re-weighted in every iteration	Weighted Majority Voting	No	Training data, no of classes, no of dimensions, no of iterations	Performance of the weak learner boosted manifold	Degrades with noise
Stack Generalization	ReSampling and k-folding	Diverse base classifiers	Meta-classifier	No	Training data, no of classes, no of dimensions, no of iterations	Good performance	Storage and time complexity

3.9 Motivation

Based on the empirical study on these various combination methods and limited experimental study, it has been observed that:

- most classifiers are application dependent and inconsistent in performance over majority of datasets
- an ensemble method with an appropriate set of base classifiers are found to perform better than the individual classifiers
- appropriate training samples can improve the performance of the classifiers significantly
- performance of the majority of combination methods are affected by the presence of outliers
- the sum rule or the weighted majority voting give good results
- a significant improvement in classification accuracy becomes possible with a meta-ensemble i.e., ensemble of ensembles, if the base ensemble methods perform consistently well and the combination method is carefully selected

Algorithms from different classification families can be used with appropriate combination method to form an effective ensemble.

- To reduce error rates, it is a necessary condition to combine the relatively uncorrelated output predictions. With highly correlated output predictions, there is little scope for the reduction in error, as the *committee of experts* has no diversity to draw from.
- A strong reason for combining models across different algorithm families can be stated as - different algorithms will provide uncorrelated output estimates because of their varied classification functions.
- Abbott [256] showed considerable differences in classifier performance class by class - information that is clear, once classifier is obscure to another. Since it is difficult to know a priori which algorithm(s) will produce the lowest error for each domain (on unseen data), combining models across algorithm families mitigates that risk by including contributions from all the families.

The motivation is to devise a cost effective ensemble method, SD-EnClass, not influenced by the biasness of the base classifiers and which shows consistently improved detection rates compared to the base classifiers in the combination. Hence,

- The selection of the base classifiers has been done based on the fact they are diverse in nature and have been established over the 2 class problem.
- To eliminate the biasness of individual classifiers, the ensemble method is adopted so that the overall classification accuracy is improved and also the errors of one classifier are averaged out by the correct classification of another classifier.
- Bagging and Boosting of each of the base classifiers is done so as to derive the maximum benefit in terms of classification accuracy.
- The individual ensemble approaches are also not totally free from their limitations, so an approach of creating an ensemble of ensembles [257] is found to be suitable to maximise classification accuracy.

3.10 Experimental Design Methodology

Tenfold cross-validation is used in this experiment where the dataset is partitioned into ten sets of equal size. Nine of these sets are combined and used for training while the remaining one is used for testing. Then the process is repeated with nine different sets combined for training and so on until all the ten individual partitions have been used for testing. The final accuracy of an algorithm will be the average of the ten trials. The datasets were obtained from Kent Ridge Biological Dataset Repository [258]. *Table* 3.3 shows the summary of the characteristics of the nine datasets. Since determining how much data is needed for training and testing is one of the key issues of data mining, we have worked around this issue by using varying proportions of training and test datasets to avoid over-fitting, but without compromising on accuracy. Therefore, the proportion of the training data was consistently kept below 60% in the experiments conducted.

The performances tabulated in *Table* 3.5 are an average of the experiments using tenfold cross-validation with varying proportions of training and test datasets.

Table 3.3: Basic information of the datasets used

Dataset	Training Instances	Test Instances	Total Instances	No of Attributes
Leukemia	38	34	72	7129
Colon	32	30	62	2000
CNS	33	27	60	7129
Ovarian	144	109	253	15154
Prostate	75	61	136	12601
PCO	21	15	36	12601
Lung Cancer	105	76	181	12533
Breast Cancer	51	46	97	24481
Lymphoma	50	46	96	4026

3.11 Software Used For Comparison

All the algorithms were executed in WEKA (Weka 3.6.2) package which is available online (http://www.cs.waikato.ac.nz/ml/weka/) with their default parameter settings in a high-end workstation with 3.33 GHz Intel Xeon processor and 8 GB RAM and the proposed model was implemented in the same environment using Java (jdk1.6). Default settings are used for all compared ensemble methods as they showed a high accuracy on an average.

3.12 The Proposed SD-EnClass

The model for combining classifiers into an ensemble is based on a simple estimation of each classifier's expertise (accuracy) on class prediction. This technique is a simple combining method which can use any '*n*' stronger learners as base classifiers to build an effective ensemble. Predictions of the '*n*' classifiers are combined to obtain the best prediction for a given test instance. In the experimental setup, *decision trees, Bayesian classifier* and *k-nearest neighbour learners* are selected as the base learners based on their consistent performance over real-life datasets such as UCI, as mentioned in *Section* 3.6.

The model utilizes the expertise of a classifier in classifying a part of the problem better than the other classifiers in the ensemble. It explores the necessary property of the diversity among the base classifiers to have a good performing ensemble. It is known that different classifiers handle the problem of classification at hand differently. So an attempt has been made to combine the results of such diverse classifiers to obtain a better classification result than using a single classifier alone.

The architecture of the proposed model is shown in the *Figure* 3.7. It is a *two-layer* model, where each layer is dedicated with a definite functionality. The output of the *Layer-1* is used as input by the *Layer-2*. *Layer-1* deals with the training and selection of the base classifiers, while *Layer-2* deals with the combination of the predictions of the selected base classifiers.

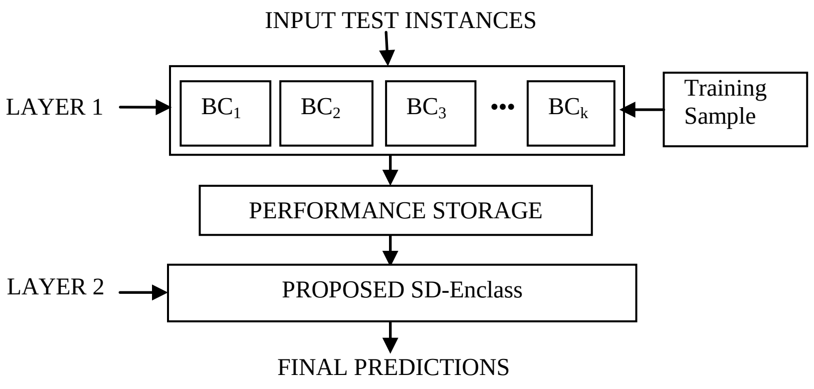


Figure 3.7: Architecture of the Proposed SD-EnClass

3.12.1 Distinct Training Sample Selection

The proposed SD-EnClass uses a *2-step* technique, referred as *SD-Prune-Redundant*, to select a distinct subset of training samples by discarding the redundant training instances. This forms the input to the tenfold cross-validation. Since the performance of classifiers is dependent on its training, so effort is made to create a training set which is complete in nature *i.e,* the training set should hold instances that would represent the entire domain space. Steps of *SD-Prune-Redundant* are given in *Figure* 3.8.

S1. Convert training dataset D_{Train} into market-basket form D_{MB} using *Algorithm* 1;
S2. Filter D_{MB} using *Algorithm* 2 to remove duplicate training instances.

```
ALGORITHM 1:MB-Representation Algorithm
Input: Training Dataset $D_{Train}$
Output: Market-Basket representation of $D_{Train}$ i.e., $D_{MB}$
Step-1: Read the raw training dataset , say $D_{Train}$
Step-2: For each attribute $a_i$ of $D_{Train}$
Step-3: Compute no. of intervals, say $\alpha$, for $a_i$ adaptively on the distribution of data
Step-4: Assign $\alpha$ no of bits for $a_i$ towards market-basket representation of $D_{Train}$
Step-5: Next i
Step-6: End
```

(a)

```
ALGORITHM 2:Find-Distinct
Input: $D_{MB}$
Output: $D_{MB}`$
Step-1: Read $D_{MB}$;
Step-2: For each pair of instances from $D_{MB}$
Step-3: Compute similarity $S_{ij}$ by counting number of agreements, i.e. no of positions
        attribute values of $i^{th}$ and $j^{th}$ instances match
Step-4: if $S_{ij} = N$ i.e. the total number of dimensions of $D_{MB}$ then select mean of these
        two instances
Step-6: Next pair
Step-7: End
```

(b)

Figure 3.8: (a) Market Basket Conversion (b) Distinct training instance selection

3.12.2 Working of the SD-Enclass

Since determining how much data is needed for training and testing is one of the key issues of data mining, a way to worked around this issue is by using varying proportions of training and test datasets so as to avoid over-fitting, without compromising on accuracy. For this purpose, tenfold cross-validation is used in this experiment. In tenfold cross-validation, a data set is equally divided into 10 folds (partitions) with the same distribution. In each test 9 folds of data are used for training and one fold is for testing (unseen data set). The test procedure is repeated 10 times. The final accuracy of an algorithm is the average of the 10 trials.

Initially, the base classifiers are trained with the distinct training dataset. Next, the performance of the base classifiers using the test dataset is evaluated. The classifier with the highest class performance for a certain class out of the base classifiers becomes the expert of that class. The class-specific performance of a classifier is calculated as:

Class specific accuracy = (*Total no of correctly predicted instances for a class*) / (total no *of predicted instances of that class*).

To evaluate the class performance of a classifier, a confusion matrix as shown in *Table* 3.4 is used. The elements in this table characterize the classification behaviour of a given classifier. The sum of the row elements represent the number of total instances present in each class, whereas the sum of the column elements gives the total number of instances predicted as that class.

Table 3.4: Confusion Matrix

Classes	**Predicted A**	**Predicted B**	**Predicted C**	***Class-wise no of Instances***
Actual *A*	x	y	z	$x+p+l$ (for A)
Actual *B*	p	q	r	$y+q+m$ (for B)
Actual *C*	l	m	n	$z+r+n$ (for C)
No of Predicted Class Instances	$x+y+z$ (for A)	$p+q+r$ (for B)	$l+m+n$ (for C)	

As shown in *Table* 3.4 the total number of instances present for class A is ($x+p+l$), and the total number of instances predicted for class A is ($x+y+z$). Here, the number of classes in the table is taken to be three, for more number of classes the rows and columns will increase respectively.

Example 3.1: Let, the total number of predicted instances for class A be 100, and total number of correctly predicted instances for class A is 90, then the (CSA) for class A for that classifier is 90/100 i.e., 0.9.

The *class specific accuracy* (CSA) for each base classifier is computed for each class and the accuracy values are stored using a *2-D link list structure*. For the experimental setup the number of base classifiers is chosen to be three (*n=3*). In the *link list structure*, the nodes in the rows correspond to the number of base classifiers in the ensemble and the column nodes correspond to the number of classes in the dataset. From this data structure, the class expert for a given class is easily found. During classification of an instance, the instance is first classified by the base classifiers and the individual predictions of the base classifiers are combined as follows:

S1. For a given instance, if all the classifiers predict the same class then the ensemble goes by the same decision.

S2. If the predictions of majority classifiers (2 out of 3) match, then any of the following situations may arise:

[a] C3 is an expert in the class it predicts, whereas C1 and C2 are not, then the prediction given by C3 is taken as the decision of the ensemble.

[b] C3 is an expert in the class it predicts and anyone of the classifiers, C1 and C2, is also an expert in its predictions then the ensemble looks for the class probabilities of the respective classifiers and selects the one with the highest value. If a further tie exists between the probability values then the ensemble goes with the majority.

S3. If the predictions of all the classifiers disagree then any of the following situations may arise:

[a] One of the classifiers could be an expert in its prediction then the ensemble goes by that classifier’s decision.

[b] Two classifiers could be experts in its class predictions. In such a case, the decision of the classifier which has a higher class probability is taken as the final decision.

[c] All the three classifiers could be experts in its class predictions. In that case, the decision of the classifier which has the highest class probability is taken as the final decision.

In this manner, the class predictions of the base classifiers are combined to get the final prediction.

3.12.3 Performance Analysis of the Proposed Model

Table 3.5: Performance Analysis of the Proposed Model

Datasets	**J48**	**NB**	**IBK**	**SD_EnClass**
Leukemia	91.18	88.24	73.53	**92.45**
Colon	**98.39**	66.13	91.94	94.40
CNS	90.00	75.00	83.33	**90.15**
Ovarian	**98.91**	94.07	98.42	98.89
Prostate	26.47	26.47	**52.94**	**52.94**
PCO	85.71	42.85	**90.48**	**90.48**
Lung Cancer	81.88	95.30	**97.97**	81.88
Breast Cancer	**73.68**	36.84	68.42	73.49
Lymphoma	91.67	88.54	91.67	**93.22**

In this phase the performance of the proposed model is tested. Our model directly uses the outputs of the base classifiers and also improves the quality of the training data. This is done by discarding redundant training instances and retaining only a distinct subset of training samples. The outputs of J48, IBK and Naïve Bayes classifiers are combined with the combination rule that has proposed earlier and the results are depicted in *Table* 3.5.

It is observed that the proposed model has been successful in increasing the prediction accuracy in 3 out of 9 datasets while in 2 datasets it has been at par with the best performing classifier. This has been mainly due to the improvement in the quality of the training data achieved by our algorithm. Thus in 5 i.e. (3+2) datasets the proposed model has shown a good performance while in 4 datasets the performance has decreased. Nonetheless, the proposed model has been successful in more than 55% of the cases.

The graph in *Figure* 3.9 shows the performance of the proposed ensemble.

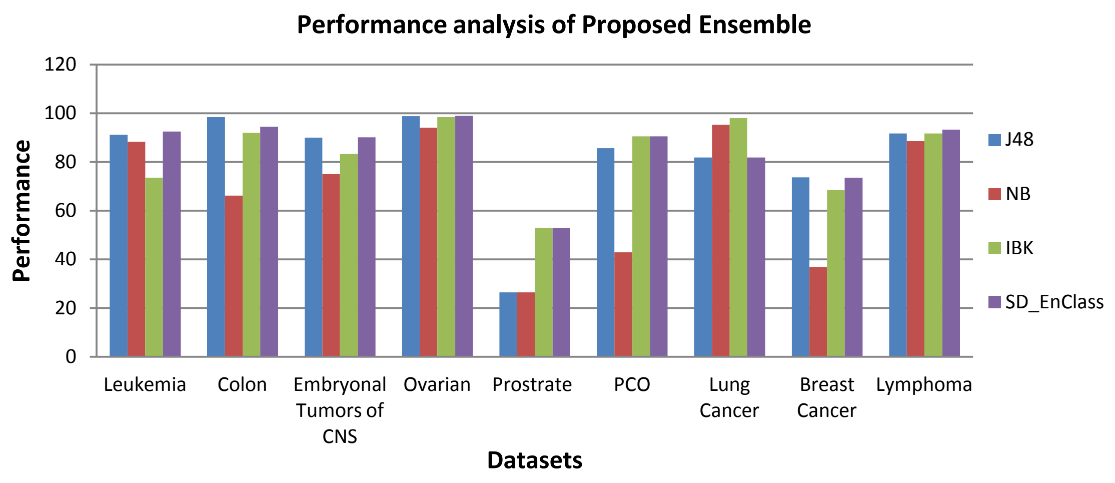


Figure 3.9: Performance Analysis of the Proposed Model

Next, the three basic ensemble approaches i.e. *Bagging, Boosting* and *Stacking* is applied to each of the base classifiers and their performance is analyzed. For Stack Generalization, the three classifiers are used as the base learners and these three base learners are used as Meta learners one by one and their performances are compared. It was noticed that Bagging, Boosting and Stacking improved the performance of J48 and NB, while Stacking with IBK slightly improved the performance of IBK at the expense of computation time. This is because Bagging and Boosting mainly improves the performances of unstable learners; IBk being a stable learner its performance was not affected much.

While comparing the proposed model with the existing ensembles it was seen from *Table* 3.6 that the proposed model has not been able to outperform the existing ensembles in most of the cases. The prediction accuracy is slightly less than the best performing ensemble.

Table 3.6: Performance Analysis of the Proposed Ensemble with Existing Ensembles

Datasets	**Bagging J48**	**Bagging NB**	**Bagging IBK**	**Boosting J48**	**Boosting NB**	**Boosting IBK**	**Stacking J48**	**Stacking NB**	**Stacking IBK**	**SD_EnClass**
Leukemia	**94.12**	88.23	73.53	91.18	88.24	73.53	91.18	91.18	91.18	92.45
Colon	95.16	66.13	90.32	**98.39**	87.10	91.94	**98.39**	93.59	93.59	94.40
CNS	81.67	81.67	83.33	90.00	75.00	83.33	58.33	10.00	51.67	**90.15**
Ovarian	98.02	92.89	98.81	98.81	**100.00**	98.42	98.81	98.81	98.81	98.89
Prostate	26.47	26.47	38.24	26.47	26.47	**52.94**	26.47	26.47	**52.94**	**52.94**
PCO	**90.48**	66.67	85.71	85.71	42.86	**90.48**	61.90	14.29	66.67	**90.48**
Lung Cancer	91.28	96.64	97.32	81.88	95.30	**97.99**	**97.99**	**97.99**	96.64	81.88
Breast Cancer	**78.95**	36.84	68.42	68.42	36.84	68.42	68.42	68.42	68.42	73.49
Lymphoma	92.71	83.33	88.54	**95.83**	88.54	91.67	86.46	25.00	94.79	93.22

Figure 3.10 gives a visual comparison of the proposed model with the existing ensembles.

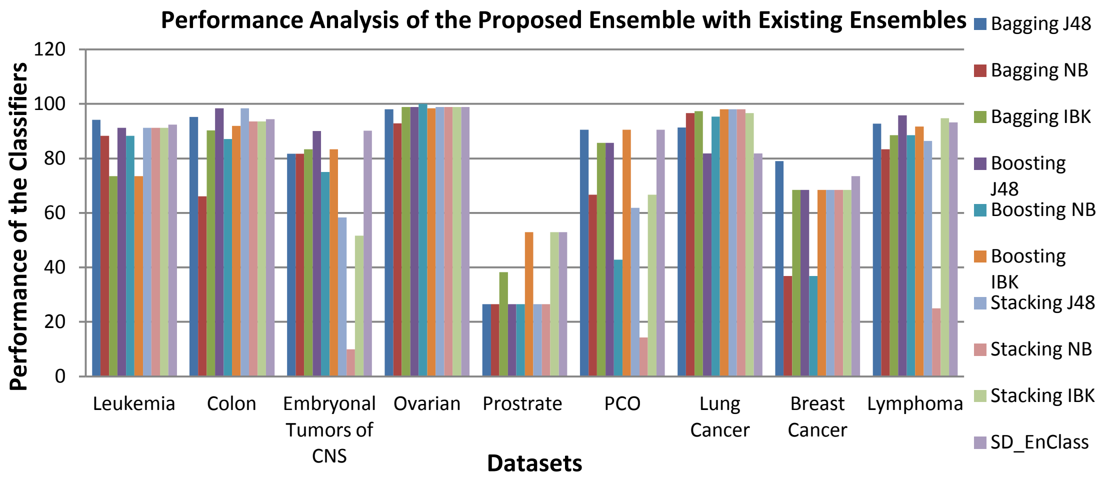


Figure 3.10: Performance Analysis of the Proposed Model with Existing Ensembles

Notably, the new model is able to have the highest accuracy in 1 dataset, in 2 datasets it is at par with the best existing ensemble and in 3 datasets it has been the second best performer. To sum up, it cannot be concluded that the proposed model has been the best performer but it has shown an average performance. To improve the prediction accuracy further, the Meta-ensemble is thus proposed.

3.13 Meta-Ensemble

From the results of the previous section it is clear that combining the outputs of different classifiers improves classification accuracy than the best single classifier in the combination, but it doesn’t perform as well as boosting. The advantage of boosting acts directly to reduce the error cases, whereas combining works indirectly. As the proposed model works well to get the best output from the combination, so this method is used to combine the results of the ensemble with the results of boosting, stacking and bagging and form a Meta-Ensemble, the architecture is shown in *Figure* 3.11.

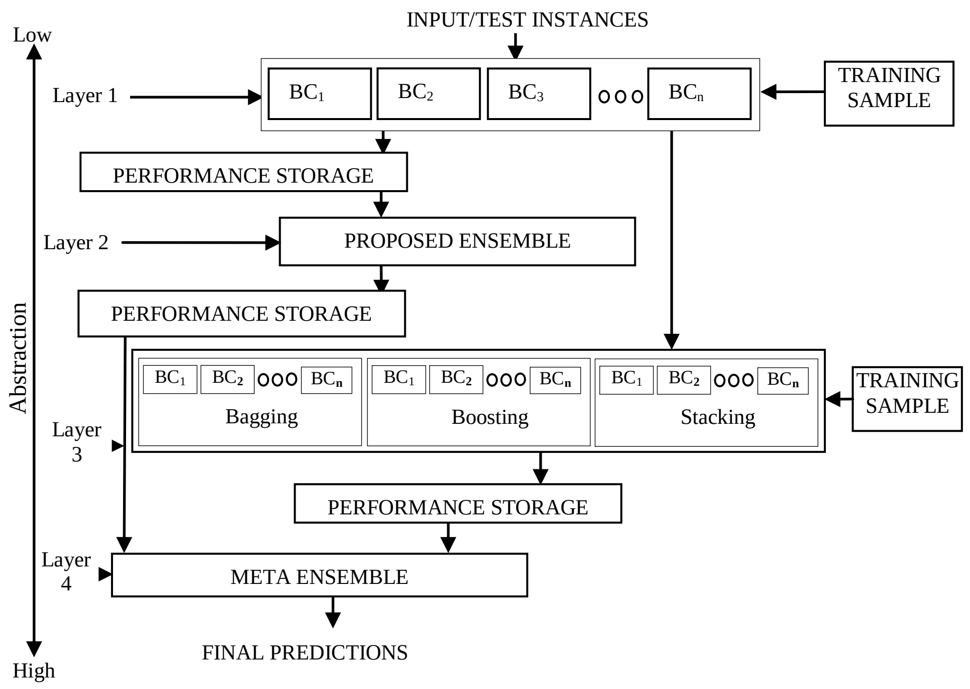


Figure 3.11: Architecture of the Meta-ensemble

The new Meta-Ensemble is a *four-layer* model where each layer has a definite functionality and the output of the lower layers is used by the higher layers. More summarized results are presented while traversing towards the higher layers.

In *layer*-1, any *n* classifier models are generated. Here *n* could be any integer, for this case *n* is set to 3. *Layer*-1 could be further improved by training a larger number of classifier models and selecting a small set of good performing classifier models out of them. That would definitely increase the overall accuracy of the proposed model.

Layer-2 is the ensemble layer where the individual outputs of the consistent base classifiers are combined as described in *Section* 3.12.2. The output of the ensemble is stored in a file which is then fed in *layer*-4.

Layer-3 creates a pool of classifier ensembles. Popular methods such as Bagging, Boosting and Stack generalization (Stacking) are implemented with the classifier models which are selected in *layer*-1 and their performances recorded. Stacking is implemented with the three classifiers as base learners and taking one of them at a time as a meta-learner. Two best performing ensembles are selected for *layer*-4 along with the proposed ensemble model.

Layer-4 finally combines the output of *layer*-2 and *layer*-4 i.e., the ensemble model and the two best performing ensembles from *layer*-3 to give the final prediction.

Table 3.7: Performance Analysis of the Meta-ensemble

Datasets	**Bagging J48**	**Bagging NB**	**Bagging IBK**	**Boosting J48**	**Boosting NB**	**Boosting IBK**	**Stacking J48**	**Stacking NB**	**Stacking IBK**	**SD_En Class**	**Meta Ensemble**
Leukemia	**94.12**	88.23	73.53	91.18	88.24	73.53	91.18	91.18	91.18	92.45	**94.12**
Colon	95.16	66.13	90.32	98.39	87.10	91.94	98.39	93.59	93.59	94.40	**99.21**
CNS	81.67	81.67	83.33	90.00	75.00	83.33	58.33	10.00	51.67	90.15	**90.19**
Ovarian	98.02	92.89	98.81	98.81	**100.00**	98.42	98.81	98.81	98.81	98.89	99.95
Prostate	26.47	26.47	38.24	26.47	26.47	**52.94**	26.47	26.47	**52.94**	**52.94**	**52.94**
PCO	**90.48**	66.67	85.71	85.71	42.86	**90.48**	61.90	14.29	66.67	**90.48**	**90.48**
Lung Cancer	91.28	96.64	97.32	81.88	95.30	**97.99**	**97.99**	**97.99**	96.64	81.88	**97.99**
Breast Cancer	78.95	36.84	68.42	68.42	36.84	68.42	68.42	68.42	68.42	73.49	**79.87**
Lymphoma	92.71	83.33	88.54	95.83	88.54	91.67	86.46	25.00	94.79	93.22	**96.13**

A comparative analysis of the Meta-Ensemble with the existing ensembles is presented in the *Table* 3.7. For each of the datasets, Boosting, Bagging and Stacking performed differently. The outputs of the two best performing methods out of boosting, bagging and stacking for each of the classifiers are combined with the result given by the proposed model.

The results in *Table* 3.7 show that out of the 9 datasets the method works well for 8 of them, while performing average in one of the datasets. Out of the 8 datasets it performed well, it improved classification accuracy in 4 of them significantly while in rest 4 it performed as well as the best classifier in the combination.

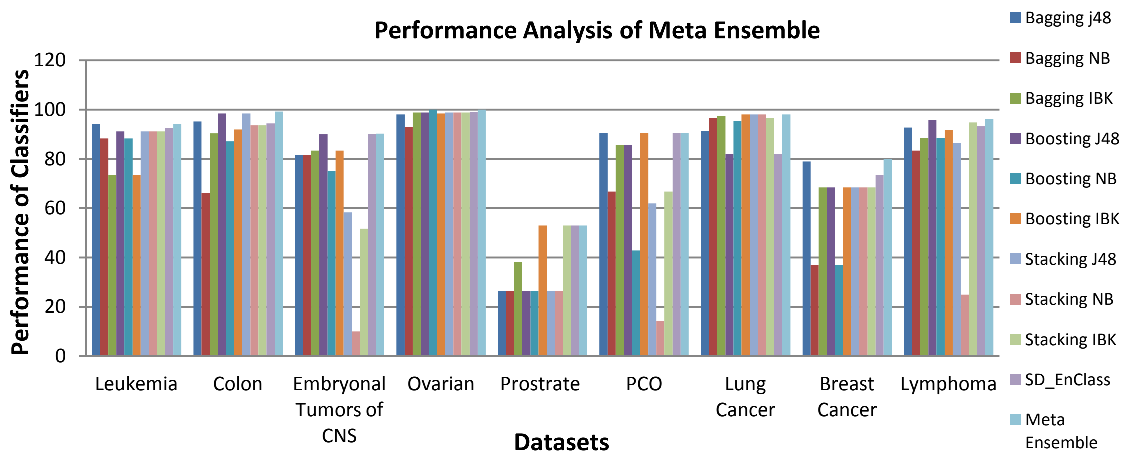


Figure 3.12: Performance Analysis of the Meta-ensemble

Thereby the results suggest that given a set of classifiers, the proposed model combines the prediction of the classifiers to obtain a better prediction result, which in most of the cases is better than selecting the best classifier in the combination of classifiers. This is visually depicted in *Figure* 3.12.

3.14 Statistical Significance of Classifiers

To show that the improvements made in classification of the microarray cancer data by meta-ensemble are statistically significant, a *paired t-test* is conducted. The paired t-test is used since a measure is needed between the before and after comparisons of classification accuracy using ensemble classifiers and by using Meta-Ensemble. Hence a paired t-test of the Meta-Ensemble with the rest of the ensemble classifiers used for classification in the experiment conducted is carried out.

Table 3.8 shows the results of the paired t-test. It is noticed that the t-test values are positive, indicating an improvement in the classification of the cancer data using the Meta-Ensemble. Moreover the low *p-values* suggest that there is strong evidence of a mean increase in the classification accuracy between using Meta-Ensemble classifier and the bagging, boosting and stacking versions of the classifiers J48, NB and IBK. This is an indication that the classification of cancer data using Meta-Ensemble provides better results.

The *mean difference* (observed difference between the classifiers) is an indicator of the mean increase in the classification accuracy. Here it is noticed that the smallest mean difference is between Meta-Ensemble and SD-EnClass (3.65) which indicates that Meta-Ensemble classification shows an improvement over SD-EnClass. This improvement in classification accuracy is even more pronounced when we compare with the mean differences obtained with the other classifiers. The next closest mean difference is between Meta-Ensemble and Bagging J48 (5.78), followed by Meta-Ensemble and Boosting IBK (5.79). The worst case scenario is between Meta-Ensemble and Stacking NB (30.57), since Stacking NB could not achieve good prediction results for CNS and Prostate Cancer (PCO) datasets.

Table 3.8: Paired t-test results

Paired Classifier		paired t-test	df	p-value	95% confidence Interval	Mean difference
Meta-Ensemble	Bagging J48	2.0908	8	0.034972	0.692194	5.782
	Bagging NB	3.7951	8	0.002637	9.180846	18.001
	Bagging IBK	4.0007	8	0.001973	4.558686	8.578
	Boosting J48	2.3433	8	0.023534	1.472499	7.132
	Boosting NB	3.0681	8	0.007696	7.026124	17.837
	Boosting IBK	2.5398	8	0.017366	1.552306	5.796
	Stacking J48	2.897	8	0.009992	4.493482	12.548
	Stacking NB	2.626	8	0.151871	8.922482	30.574
	Stacking IBK	2.1698	8	0.030923	1.368973	9.574
	SD-EnClass	2.1211	8	0.033364	0.451814	3.664

The results of the paired t-test are shown in **Error! Reference source not found.** and Mean difference (observed difference between the classifiers) is plotted in **Error! Reference source not found.** below.

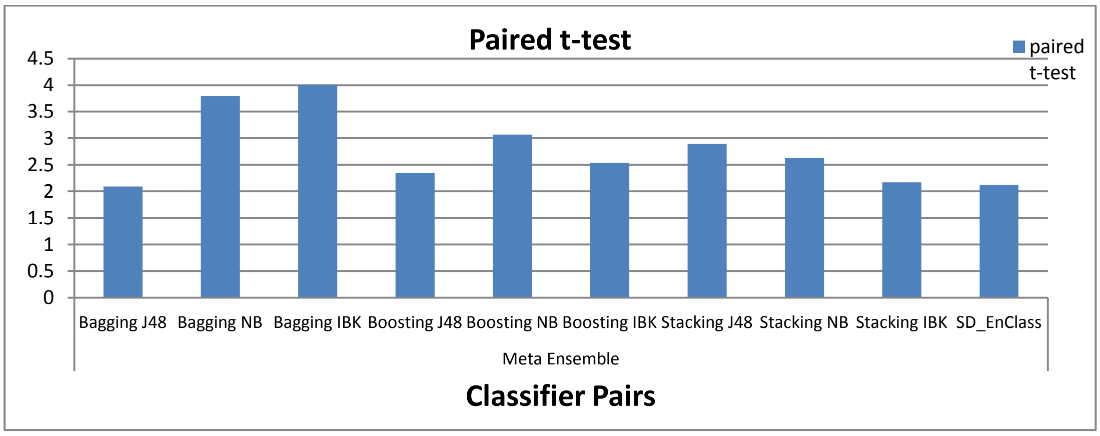


Figure 3.13: Paired t-test comparison of the Classifiers

We notice that the *p-value* and *mean difference* for Stacking NB is comparatively much higher than the rest of the ensemble classifiers.

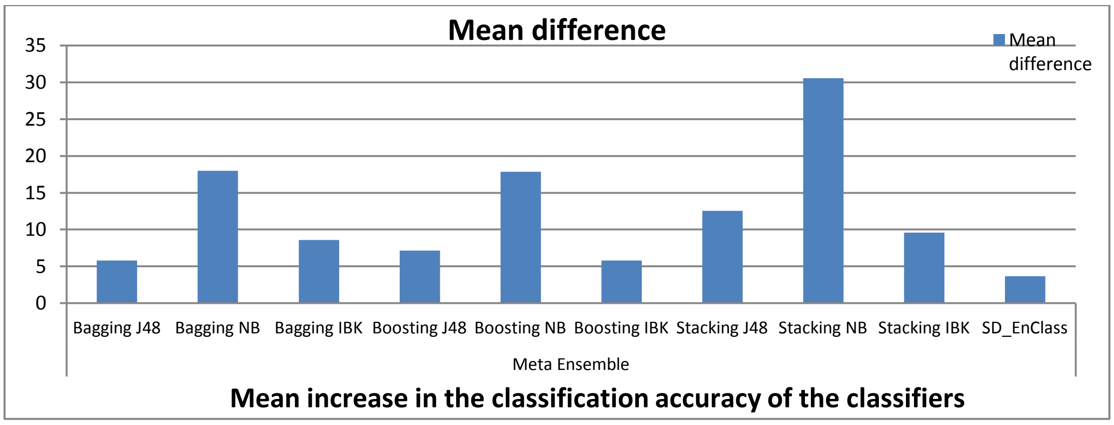


Figure 3.14: Classification accuracy of the classifiers as compared to Meta-Ensemble

This may be due to the fact that Stacking NB could not achieve good prediction values (10 and 14.29 respectively), for CNS and Prostate Cancer (PCO) datasets as reported in *Table* 3.7, when compared to its competitors.

3.15 Discussion

In this chapter the performances of popular ensemble methods like Bagging, Boosting and Stack Generalization are analyzed. It has been found that on an average, Boosting and Stack Generalization using unstable learners (decision trees) and probabilistic classifiers (Naïve Bayes) works better than applying them to stable learners (nearest neighbour classifiers). It has also been learnt that Bagging of classifiers performs better than Boosting and Stack generalization for most of the cancer datasets used in the experiments.

An effective way of combining the outputs of the classifiers in the ensemble has been proposed, based on the class performance of each of the classifiers in the combination and experimental results show that the ensemble performs better than the best single classifier in the combination in majority of the cases. The model has been mostly tested on the two class datasets for which it performed well; it has also tested on one multi-class dataset for which it gave good performance and in future the combination rules can be extended so that the ensemble model works even better for multi-class problem.

While combining models across the different algorithm families, an improvement of the performance in the classification accuracy was seen as compared to the best single model in the combination, but when compared to Bagging, its performance was average. So in the next stage the results of the proposed ensemble was combined with the results of the best performing ensemble methods for the datasets, using the proposed combining method and obtained results which were significantly better than using Boosting, Bagging or Stacking alone.

In the next chapter, the combination method developed is used for the purpose of combining the output of several clustering algorithms with an objective to improve the accuracy of the clustering results of cancer datasets.