An Empirical Comparison Of Supervised Learning Processes

Sanjeev Manchanda* smanchanda@thapar.edu
School of Mathematics and Computer Applications,
Thapar University, Patiala-147004 (INDIA).
**Corresponding author*

Mayank Dave mdave67@yahoo.com
Department of Computer Engg.,
National Instt. Of Technology, Kurukshetra, India.

S. B. Singh sbsingh69@yahoo.com
Department of Mathematics,
Punjabi University, Patiala, India.

Abstract

Data mining as a formal discipline is only two decades old, but it has registered phenomenal development and has become a mature discipline in this short span. In this paper, we present an empirical study of supervised learning processes based on empirical evaluation of different classification algorithms. We have included most of the supervised learning processes based on different pre pruning and post pruning criteria. We have included ten datasets, collected from internationally renowned agencies. Different specific models are presented and results are generated. Issues related to different processes are analyzed suitably. We also present a comparison of our study with benchmark results of different datasets and classification algorithms. We have presented results of all algorithms with fifteen different performance measures out of a set of twenty three calculated measures, making it a comprehensive study.

Keywords: Data Mining, Knowledge Discovery in Databases, Supervised learning algorithms, Stacking, Classification, Regression etc.

1. Introduction

Knowledge discovery in databases (KDD) is the theme of many discussions for last two decades. A large number of techniques and algorithms have been developed for mining the knowledge from large databases. Supervised learning techniques are usually used for the solution of classification problems. Usually a general process is recommended for supervised learning. But practical implementation of a general process becomes difficult, when we need to implement this general process for some specific problem solving. There are possibly many processes that are used for supervised learning. Problem arises with finding a suitable process for extracting knowledge for problem at hand. This type of dilemma motivated us to analyze the environmental factor that affect the selection of a suitable process and to handle potential issues involved in such processing.

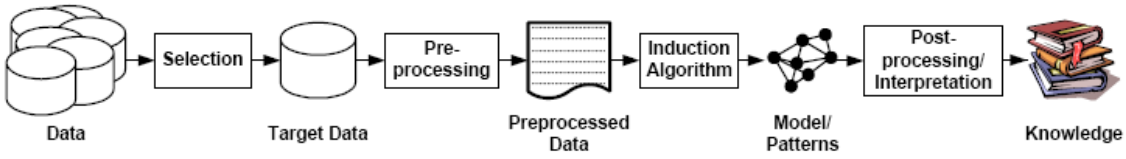

Figure 1: The KDD process (Fayyad et al. [10])

Present work is mainly motivated through following three objectives. First of all, supervised learning processes can vary from simple to very complex processing. No single process can fulfill all needs and suitability of any process depends upon many environmental factors. So, there is a need to analyze different processes by identifying different environmental factors. Secondly, Different techniques and algorithms are used to extract knowledge from data. These algorithms involve certain criteria to extract knowledge. Different techniques and algorithms are suitable for different types of problems. There is no unique technique/algorithm to solve all types of problems. So, there is a need to analyze suitability of different techniques/algorithms with specific domain of problems. Thirdly, Different performance metrics are considered appropriate for different domains, e. g. Precision/Recall measures are preferred metrics for information retrieval, ROC curves/area is preferred metric for the problems related to medical domain, Lift is preferred for marketing tasks etc. Each metric is dedicated to some specific nature of algorithm evaluation. No individual metric may be used for all domains. So, there is a need to test different learning algorithms based on a large set of metrics. Overall present paper is an effort to explore relationship between types of problems with specific technique/algorithm as well as with type of processing required for extracting knowledge based on different metrics. Experiments are performed through many suitable processes on a variety of supervised learning techniques and algorithms. Results are presented for fifteen different metrics out of generated results for twenty three metrics. Output of these experiments is compared with the results obtained from direct experimentation of classification algorithms and the results obtained through cross validations. Results are also compared with the available benchmark results of the problems involved for study. This paper includes a comprehensive study of different possible supervised learning processes. Internationally renowned datasets are chosen for evaluating six most important processes for study. These datasets are applied on these processes and comprehensive results are presented.

Rest of the content of this paper is organized in following manner. Second section includes the literature review of related work. Third section includes the description of various processes for supervised learning. Fourth section includes the description of different techniques and algorithms included for study. Fifth section explains methodology of study. Sixth section includes experimental results of present study. Seventh section includes a comparison of present study with other studies. Eighth section concludes the study with future directions. Last but not the least, Ninth section lists the references used during present study.

2. Literature Review

Data mining has originated just two decades back. Within this short span, data mining has grown up as a mature discipline. Large numbers of techniques and algorithms have been developed for extraction of knowledge. Out of these algorithms, majority of algorithms are developed for

supervised learning. Supervised learning is mostly performed for classification tasks. Data mining itself has emerged from other disciplines like Machine Learning, Artificial Intelligence and Statistics etc., so it is obvious to get initial references related to this study from its parent disciplines. Many researches were being performed before the time data mining was coined as a separate discipline for study.

In a study, the results of a point awarding approach were compared with the results obtained by the linear discriminant (Fahrmeir et al. [9]). One study reported that back-propagation outperformed nearest neighbour for classifying sonar targets (Gorman et al. [13]), whereas some Bayes algorithms were shown to be better on other tasks (Shadmehr et al. [30]). A symbolic algorithm, ID3 (Kirkwood et al. [16]) was developed, which performed better than discriminant analysis for classifying the gait cycle of artificial limbs.

The CART (Classification and Regression Trees) method (Breiman et al. [6]) was used to analyze consumer credit granting (Hofmann [14]). It concluded that CART had major advantages over discriminant analysis and emphasized the ability of CART to deal with mixed datasets containing both qualitative and quantitative attributes. However, on different tasks other researchers found that a higher order neural network (HONN) performed better than ID3 (Spikvoska et al. [32]) and back-propagation did better than CART (Atlas et al., [1]).

A study was conducted for a coordinated comparison of many algorithms on the MONK's problem (Mitchell et al. [20]). A diverse set of statistical methods, neural networks, and a decision tree classifier was compared on the Tsetse fly data (Ripley [28]). After many small comparative studies, STATLOG is known as first comprehensive study that analyzed different data mining algorithms (King et al. [15]). Another research work compared several learning algorithms (including SVMs) on a handwriting recognition problem using three performance criteria: accuracy, rejection rate, and computational cost (LeCun et al. [18]). One other study evaluated nearly a dozen learning methods on a real medical data set using both accuracy and an ROC-like metric (Cooper et al. [8]). In one other study, an impressive empirical analysis was presented about different ensemble methods such as bagging and boosting (Bauer et al. [3]). An empirical comparison of decision trees and other classification methods was performed using accuracy as the main criterion (Lim et al. [19]). An empirical study conducted comparison between decision trees and logistic regression (Perlich et al. [23]). One study examined the issue of predicting probabilities with decision trees, including smoothed and bagged trees (Provost et al. [25]). One research work presented the comparison of different tools and techniques of data mining (Witten et al. [33]). Recently, one study was conducted to rank different many learning algorithms (Caruana et al. [7]). Present research work is dedicated to analyze all type of classification techniques and algorithms on a variety of problems and to compare the results with earlier studies.

3. Various Processes for Supervised Learning

Supervised learning processes can vary from simple processing to very complex processing. Different techniques and algorithms are used to extract knowledge from data. These algorithms involve certain criteria to extract knowledge. Different techniques and algorithms are suitable for different types of problems. There is no unique technique/algorithm to solve all types of problem. Supervised learning involves training set to train algorithm for the creation of a model and then this model is applied on test set to generate and compare results. Different supervised learning processes are as follows:

3.1 Simple Supervised Learning: In its simplest form input data is applied to classification algorithm to generate a model, model is applied test data and result is generated. Such experimentation suffers with over fitting and under fitting of model and results may not fulfill the reliability criteria. So there is a need for preprocessing and post-processing of data.

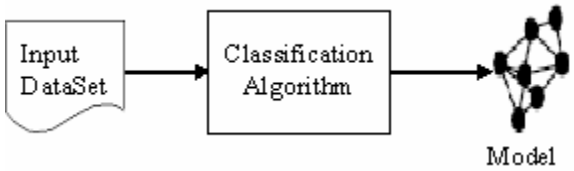

Figure 2: Simple Supervised Learning Process.

3.2 Preprocessing of the data: A data set collected is not directly suitable for induction (knowledge acquisition), it comprises in most cases noise, missing values, inconsistent data, data set is too large, and so on. Therefore, we need to minimize the noise in data, choose a strategy for handling missing (unknown) attribute values, use any suitable method for selecting and ordering attributes (features) according to their informativity (so-called attribute mining), discretize/fuzzify numerical (continuous) attributes, validating part of training data to be used for creating model and eventually process continuous classes.

3.2.1 Attribute Transformation: Input data may be nominal or numerical. Few classification algorithms like ID3 and Naïve Bayes operate only on discrete data, whereas regression based algorithms operate only on numerical data. So there may a requirement of transformation of data from one form to another to match the data with algorithmic requirements.

3.2.1.1 Categorical Attribute Transformation: Nominal or categorical data may be transformed into binary or scale values as follows:

(a) Categorical to Binary: Problems having more than two categories of class attribute are converted into Binary class problems. We have converted our datasets into binary class treating first half of class categories as negative class and last half as positive class.

(b) Dual Scaling: Dual scaling (Nishisato [22]) is a multivariate method for assigning scale values to the rows and columns of a table of data, with certain optimal properties.

3.2.1.2 Continuous Attribute Transformation: Continuous or real number based attributes may be transformed into discrete attributes as follows:

(a) Class-based discretization: Class-Attribute relationship is used to define discretization of any attribute, each attribute is discretized independently. Such discretization is useful for small number of attributes, but becomes complex for large number of attribute.

(b) Fixed-bin discretization: All the attributes to be discretized are considered collectively and a fixed number of bins are used for discretizing all attributes. We have used fixed bin discretization, so that the future researchers can utilize the results of this paper for their comparative analysis and it also helps in maintaining consistency of experimentation.

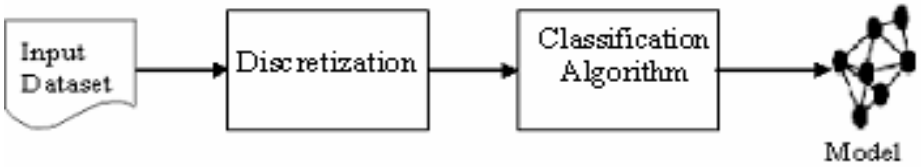

Figure 3: Discretized Supervised Learning Process.

(c) Principle component analysis: Principal component analysis is a useful tool for categorization of data, it separates the dominating features in the data set.

3.2.2 Data Sampling

(a) Progressive sampling: Progressive Sampling (PS) (Provost et al. [27]) incrementally constructs a training set from a larger dataset without decreasing the classification performance and without altering the initial format of the examples

(b) Random sampling: Samples are selected randomly for experimentation. Such a sampling makes the experimentation results to be unreliable as different sampling algorithms may select samples differently and results may vary significantly.

(c) Stratified sampling: Stratified sampling is based on re-sampling the original datasets in different ways: under-sampling the majority class or over-sampling the minority class.

3.2.3 Validation:

(a) Fixed Split Validation: Simplest form of experimentation is to divide dataset into two fixed length datasets of training set and test set to perform experiment directly. This kind of experimentation is meant for simple testing of algorithms. Biggest problem with fixed split is over fitting of training data e.g. tree based techniques may have too many branches that may reflect anomalies and result in poor accuracy of unseen samples. To overcome the limitations of fixed split two approaches used are prepruning and post pruning. Prepruning is performed through cross-validation, whereas many calibration methods have been proposed for post pruning. Following sections include the discussion about these methods.

(b) Cross Validation: To evaluate the robustness of the classifier, the normal methodology is to perform cross validation on the classifier. Ten fold cross validation has been proved to be statistically good enough in evaluating the performance of the classifier (Witten et al. [33]). For present study datasets are divided into training and test sets. Then training set is equally divided into 10 different subsets for ten fold cross validation. Nine out of ten of the training subsets are used to train the learner and the tenth subset is used as the test set. The procedure is repeated ten times, with a different subset being used as the test set. In this way cross validation is performed to calibrate the models and select the best parameters and then models are applied on the large Final test set.

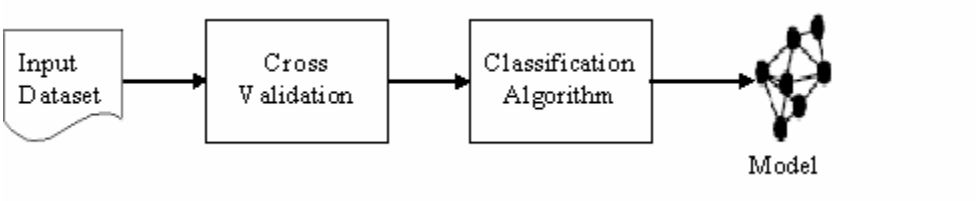

Figure 4: Cross-Validated Supervised Learning Process.

3.3 Post processing of the derived knowledge: The pieces of knowledge extracted in the previous step could be further processed. One option is to simplify the extracted knowledge. Also, we can evaluate the extracted knowledge, visualize it, or merely document it for the end user. They are various techniques to do that. Next, we may interpret the knowledge and incorporate it into an existing system, and check for potential conflicts with previously induced knowledge.

3.3.1 Calibration: Many learning algorithms do not predict probabilities. For example the outputs of an SVM are normalized distances to the decision boundary, whereas naive bayes models are known to predict poorly calibrated probabilities, because of the unrealistic independence assumption.

A number of methods have been proposed for mapping predictions to posterior probabilities. Platt Scaling (Platt [24]) is used for transforming SVM predictions to posterior probabilities by passing them through a sigmoid. Platt scaling also works well for boosted trees and boosted stumps (Niculescu et al.[21]). A sigmoid is also not the correct transformation for all learning algorithms.

Second method used for calibration is Logistic regression. Logit Boost algorithm is used for performing additive logistic regression. This algorithm performs classification using a regression scheme as the base learner, and can handle multi-class problems (Friedman et al. [11]) and it can also do efficient internal cross-validation to determine appropriate number of iterations.

Other method generally used for calibration is Isotonic Regression (Zadrozny et al. [35,36]; Robertson et al. [29]). It is used to calibrate predictions from SVMs, naive bayes, boosted naive bayes, and decision trees. Isotonic Regression is more general method, but its only restriction is that the mapping function used is isotonic (monotonically increasing). A standard algorithm for Isotonic Regression that finds a piecewise constant solution in linear time, is the pair-adjacent violators (PAV) algorithm (Ayer et al. [2]).

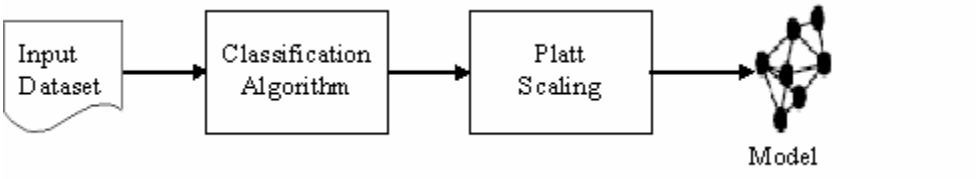

Figure 5: Post-Processed Supervised Learning Process.

3.3.2 Thresholding: The minimum acceptable value which, in the user's judgment, is necessary to satisfy the need. If threshold values are not achieved, program performance is seriously degraded, the program may be too costly, or the program may no longer be timely.

3.3.2.1 Class Probability Estimators (CPE) Thresholding: For a decision maker to act optimally it is necessary to estimate the probability of success. Because training information is costly, we would like to reduce the cost of inducing an estimation model that will render decisions of a given quality. One approach to reducing the cost of learning accurate CPEs is via traditional active learning methods, which are designed to improve the model's average performance over the instance space. if the probability of a successful outcome exceeds the threshold.

3.3.2.2 Regression Thresholding: Threshold regression refers to first-hitting-time models with regression structures that accommodate covariate data. The parameters of the process, threshold state and time scale may depend on the covariates.

3.4 Stacking: Stacking combines the output of a number of classifiers. Stacked Generalization, also known as Stacking in the literature, is a method that combines multiple classifiers by learning the way that their output correlates with the true class on an independent set of instances. At a first step, N classifiers Ci, i = 1..N are induced from each of N data sets Di, i = 1..N. Then, for every instance e_j , j = 1..L of an evaluation set E, independent of the Di data sets, the output of the classifiers $Ci(e_j)$ along with the true class of the instance $class(e_j)$ is used to form an instance m_j , j = 1..L of a new data set M, which will then serve as the meta-level training set. Each instance will be of the form: $C1(e_j)$, $C2(e_j)$, . . . , $CN(e_j)$, $class(e_j)$. Finally, a global classifier GC is induced directly from M. If a new instance appears for classification, the output of all local models is first calculated and then propagated to the global model, which outputs the final result. Any algorithm suitable for classification problems can be used for learning the Ci and GC classifiers. Independence of the actual algorithm used for learning Ci, is actually one of the advantages of Stacking, as not every algorithm might be available for each data set and not the same algorithm performs best for every data set. We have applied stacking of isotonic regression with other classification algorithms.

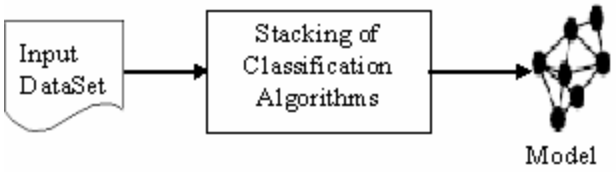

Figure 6: Stacked Supervised Learning Process.

3.5 Complex Processing: Different preprocessing, Post-processing and stacking of different algorithms may be combined to extract knowledge from databases. Such complex criteria may involve parallel processing of different algorithms as well. No encouraging results have been generated through such processing.

4. Description of Techniques and Algorithms used for study

Different techniques included for study with their specific algorithms are as follows:

4.1 Classification Techniques and Algorithms: A variety of classification algorithms were used for study. These techniques/algorithms are broadly described as follows:

4.1.1 Decision Trees: Tree-shaped structures that represent sets of decisions. These decisions generate rules for the classification of a dataset. Decision trees represent a series of IF…THEN type rules which are linked together and can be used to predict properties for observations based upon the values of various features. These are able to produce human-readable descriptions of trends in the underlying relationships of a dataset and can be used for classification and prediction tasks. The algorithms used for experimentation were Decision Stump and REPTree etc. Different parameters were set as follows: maximum tree depth was allowed to be infinite, minimum number of instance per leaf were set to 2, Confidence threshold was set to 0.25 and numbers of trees were allowed to be infinite.

4.1.2 Support Vector Machine: These are methods for creating functions from a set of labeled training data. These functions can be a classification function (the output is binary: is the input in a category) or the function can be a general regression function. For classification, SVMs operate by finding a hyper-surface in the space of possible inputs. This hyper-surface will attempt to split the positive examples from the negative examples. The split will be chosen to have the largest distance from the hyper-surface to the nearest of the positive and negative examples. Intuitively, this makes the classification correct for testing data that is near, but not identical to the training data. We have included LibSVM algorithm for study. Different parameters were set as follows: Different types of kernel functions were tried like linear, polynomial, radial basis function etc., Degree of kernel function set to 3 and Tolerance parameter set to 0.001.

4.1.3 Genetic Algorithms: Optimization techniques that use process such as genetic combination, mutation, and natural selection in a design based on the concepts of evolution. Genetic algorithms should be used, when no other option is left. We have not included any genetic algorithm, but learning processes are always based on genetic processing, so indirect contribution of genetic processing can not be neglected.

4.1.4 Neural Networks: Inspired by the structure of the brain, a neural network consists of a set of highly interconnected entities, called *nodes* or *units*. Each unit is designed to mimic its biological counterpart, the neuron. Each accepts a weighted set of inputs and responds with an output. An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. We have included Multi Layer Perceptron algorithm for study. Different parameters were set as follows: Learning rate of back propagation set to be 0.3, Momentum rate 0.2 etc.

4.1.5 K-nearest neighbor: Among the various methods of supervised statistical pattern recognition, the Nearest Neighbor rule achieves consistently high performance, without *a priori* assumptions about the distributions from which training examples are drawn. It involves a training set of both positive and negative cases. A new sample is classified by calculating the distance to the nearest training case; sign of that point then determines the classification of the sample. The IBk classifier included in present study extends this idea by taking the *k* nearest points and assigning the sign of the majority. It is common to select *k* small and odd to break ties (typically 1, 3 or 5). Larger *k* values help reduce the effects of noisy points within the training data set, and the choice of *k* is often performed through cross-validation. Different parameters were set as follows: Different values for k were tried ranging 1 to 10.

4.1.6 Rule Induction: The extraction of useful if-then rules from data based on statistical significance. We have included Decision Table and ZeroR algorithms for study. Different parameters were set as follows: Confidence threshold set to 0.25.

4.2 Boosting/Bagging: These methods create a set or ensemble of classifiers from a given dataset. Each classifier is generated with a different training set obtained from the original using re-sampling techniques. The final output is obtained by voting.

Boosting: The idea of Boosting is to combine simple rules to form an ensemble such that the performance of the single ensemble member is improved i.e. Boosted. AdaBoostM1 algorithm was used for boosting trees (Yoav et al. [34]). Different parameters were set as follows: Number of iterations allowed was 10 and 100 percentage of weight mass being used.

Bagging (**B**ootstrap A**gg**regat**ing**): It produces replications of the training set by sampling with replacement. Each replacement of the training set has the same size as the original set, but some examples can appear more than once while other don't appear at all. A classifier is generated from each replication. All classifiers are used to classify each sample from the test set using a vote scheme (Breiman [6]). We have applied Bagging and Boosting on Decision Stump and REPTree algorithms. Experimental results of both Boosting and Bagging are really enthusiastic. Different parameters were set as follows: Size of each bag being set to 100 and number of iterations allowed were 10.

5. Methodology

5.1 Datasets: Present study compares supervised learning algorithms on ten binary classification problems. ADULT, COV_TYPE, LETTER, PEN_DIGITS, SHUTTLE, SATELLITE and TIC2000 are the problems from UCI repositories (Blake et al. [5]). COV_TYPE has been converted to a binary problem by treating the largest four classes as positives and the rest three as negatives. LETTER is converted by replacing alphabets A-M as negatives and N-Z as positives. PEN_DIGITS is converted by replacing top five digits (5 to 9) into positive class whereas lower five into negative class (0 to 5). SATELLITE and SHUTTLE are the problems from STATLOG. SHUTTLE has been converted to a binary problem by treating largest two classes as positives and rest three classes as negatives. SATELLITE conversion is treated by converting largest three classes (i. e. 4, 5, 7) as positives (class 6 was absent), whereas smallest three classes as negatives (i.e. 1, 2, 3). ACC_CELE and ACC_DROSO are biological sequence datasets (Sonnenburg et al.[31]). DS1_100 is outcome of biological and chemistry experiments (Komarek et al. [17]). Table 1 includes the description about the datasets.

Problem	Number of Attributes	Size of Datasets		
		Train Set	Test Set	Total
ADULT	14	9768	39074	48842
COV_TYPE	54	10000	40000	50000
ACC_CELE	141	10000	40000	50000
ACC_DROSO	141	10000	40000	50000
DS1_100	100	10000	16734	26734
LETTER	16	10000	10000	20000
PEN_DIGITS	16	5000	5992	10992
SATELLITE	36	3000	3435	6435
SHUTTLE	9	10000	40000	50000
TIC2000	85	5000	4822	9822

Table 1: Description of Problems

5.2 Experimentation: Experimentation is the most important part of any empirical study. We have included all the ways of experimentation developed so far for supervised learning. In this study Pre-processing through Fixed split validation and Cross validation have been performed, whereas three calibration methods viz. Platt scaling, Logit Boost and Additive Regression have been used for experimentation and Isotonic Regression has been applied through Stacking of algorithms. Discretization has been applied for ID3 and Naïve Bayes algorithms.

5.3 Metrics for evaluation: Learning techniques and algorithms are used in a variety of domains. Different performance metrics are considered appropriate for different domains, e. g. Precision/Recall measures are preferred metrics for information retrieval, ROC curves/area is preferred metric for the problems related to medical domain, Lift is preferred for marketing tasks etc. Each metric is dedicated to some specific nature of algorithm evaluation. No individual metric may be used for all domains. So, there is a need to test different learning algorithms based on a large set of metrics. Metrics used for testing algorithms are broadly categorized as follows (Same metric may belong to more than one broader category depending on their nature belonging to multiple categories):

5.3.1 Confusion Matrix Based Metrics: Outcome of all classification tasks produces four types of output i.e. two from each instance is mapped to one element of the set { Positive, Negative} from actual positive and negative class labels, whereas other two labels {Positive, Negative} from the class predictions produced by a model. Different statistics like Accuracy, Precision, Recall, Fallout, F-measure, Margin etc. are directly derived from the confusion matrix (Provost et al. [26,27]), whereas Lift, AUC are derived from it.

		Actual Class	
		Positive	**Negative**
Predicted Class	**Positive**	**T**rue **P**ositives	**F**alse **P**ositives
	Negative	**F**alse **N**egatives	**T**rue **N**egatives

Table 2: A contingency table for a binary class problem

5.3.2 Threshold metrics: The threshold metrics are accuracy, F-score and Lift (Giudici, [12]). A fixed threshold 0.5 is used for Accuracy and F-Score. For lift, percent p of cases is predicted as positive and the rest as negative, for present study p is selected to be 25%. Predictions may have a significant distance from these thresholds.

5.3.3 Rank Metrics: The rank metrics used are **A**rea **U**nder the RO**C** curve (i. e. AUC) (Provost et al. [26]), Average Precision and Recall.

5.3.4 Errors: Different types of errors have been involved in this study. Absolute Error, Relative Error, Root Mean Squared Error, Squared Error and Fallout etc. have been calculated for all the algorithms and problems involved for present study. Classification error has been omitted from the table because it can be calculated from the accuracy measure by subtracting accuracy from one.

5.3.5 Probability Metrics: Probability metrics, Root mean squared error and Mean crossed-entropy, interpret the predicted value of each case as a conditional probability of that case being in the positive class.

5.3.6 Other Metrics: Other metrics like kappa and correlation are calculated**.** The kappa coefficient measures pair wise agreement among a set of coders making category judgments, correcting for expected chance agreement (Berry [4]), whereas correlation calculates the degree of relationship between attributes.

6. Experimental Results

This section includes the experimental results of present study. Experimental results are divided into two categories viz. Major study and Minor study.

6.1 Major Study: Major study includes twenty two algorithms and experiments are performed over fixed split, cross validation and platt scaling. For Fixed Split validation original dataset is divided into train set and test set, then experiments are performed. For Cross Validation dataset is again divided into Train set and test set, Train set is further divided into ten fold datasets, Experiments are performed over one fold with the help of others and dataset with minimum squared error is selected for testing the performance over test set. For Platt Scaling, Cross validated model is passed through a sigmoid and probabilities based predictions are performed.

6.1.1 Performance by Problem: Table 3 includes accuracy of twenty two algorithms involved for study and are ranked in descending order based on their average performances. Random Forest algorithms have topped the chart, whereas J48, PART, Multi Layer Perceptron, IBk, REPTree and ADTree algorithms are close to the top positions. Fixed Split has performed better than Cross Validated and Platt Scaling preprocessing and post processing algorithms. As Cross validation restricts the over fitting of algorithms, so the performance over cross validation and platt scaling is the corrected performance of the algorithms. Even for Cross validated and Platt scaling results random forest algorithms perform far better than other algorithms. Bagging (**B**ootstrap A**gg**regat**ing**) has performed better than alone algorithm and with boosting. Fro ADTree Boosting seems to perform better than others and has enhanced the performance rapidly. ZeroR has performed very badly and has secured lowest positions as compared to others.

6.1.2 Performance by metrics: Table 4 includes averages of fifteen metrics involved in study and are positioned in descending order according to average accuracy. For few metrics output was generated to be NaN (**N**ot **a N**umber), for such metrics averages are calculated over remaining values, excluding the count of such values. These values are pointed with an asterisk (*) and if all the values (for all ten problems) are NaN, such values are represented by NaN*.

6.2 Minor Study: Minor study includes five algorithms and experiments are performed over other calibration methods like Additive Regression, Logistic Regression and Isotonic Regression. Limitation for regression based methods is that these require fully numeric values, so all the datasets are converted into numeric values except for the Logit Boost algorithm (i. e. Logistic Regression). On Additive regression ten fold cross validation has been applied and study is performed through meta classifiers. Logit Boost involves internal cross validation, so fixed split experimentation is performed. For Isotonic regression, stacking is done in conjunction with other algorithms and ten fold cross validation is performed.

6.2.1 Performance by Problem: Table 5 includes accuracy of five algorithms for all ten datasets that are involved for study and are ranked in descending order, based on their average accuracy. Five algorithms used for study are IBk, Decision Stump, Decsion Table, LibSVM and ZeroR across six dimensions i. e. Fixed Split, Cross Validation, Platt Scaling, Additive Regression, Logit Boost and Isotonic Regression. Results indicate the better performances through Logit Boost calibration, followed by Additive Regression. Isotonic Regression has degraded the performances of the algorithms. IBk and Decision Table has topped the chart with calibration and individually. Additive Regression has enhanced the performance of ZeroR algorithm and has uplifted its performance significantly.

6.2.2 Performance by metrics: Table 6 includes averages of fifteen metrics involved in study and are positioned in descending order according to average accuracy. For few metrics output was generated to be NaN (**N**ot **a N**umber) and Infinity, for such metrics averages were calculated over remaining values, excluding the count of such values. These values are pointed with an asterisk (*) for NaN, a plus sign (+) for Infinity and if all the values (for all ten problems) are NaN or Infinity, such values are represented by NaN* or Inf+.

6.3 Graphical Comparison: A graphical comparison involving ROC and Precision/Recall graphs of algorithms is prepared for Cross Validated experiments on Adult dataset.

6.3.1 ROC Curves: An ROC graph is a technique for visualizing, organizing and selecting classifiers based on their performance. ROC graphs are two-dimensional graphs in which True Positive rate is plotted on the Y axis and False Positive rate is plotted on the X axis. An ROC graph is compared on the basis of behavior of the curve in graph. A curve sharply rising towards Y axis is considered to be better than the diagonal or a curve sharply bending towards X axis. Clearly, in figure 7 better performing algorithms like random forests, boosted decision stumps etc. have their curves rising towards Y-axis marking better performance for them, SMO Decision Stump etc. are rising diagonally indicate average performance.

6.3.2 Precision/Recall Curves: Precision is the ratio of True Positives to the Sum of True Positives and False Positives, Recall is the ratio of True Positives to the Sum of True Positives and False Positives. A Precision/Recall curve bending towards origin is considered to be worst performances, whereas a curve rising away from origin towards 1 for X and Y axis collectively, is considered to be better performances. Clearly, algorithms like Random Forests, ADTrees etc. are rising away from origin indicate better performances, whereas diagonal curves for algorithms like SMO, ID3 etc. indicate average performances. These graphs are indicating the scenario of Adult problem in figure 8, curves can dramatically change for other problems depending upon their results.

Algorithm	Validation/ Calibration	Adult	CovType	Celegans	Droso	DS1_100	Letter	PenDigits	Satellite	Shuttle	Tic2000	Average
Random-Forest-Bagging	Fixed Split	0.8436	0.9804	0.9421	0.9832	0.9796	0.9632	0.9938	0.9485	0.9998	0.9287	0.9563
Random-Forest-Boosting	Fixed Split	0.8343	0.9788	0.9452	0.9832	0.9799	0.9671	0.9902	0.9517	0.9997	0.9243	0.9554
Random-Forest	Fixed Split	0.8328	0.9774	0.9446	0.9832	0.9788	0.9519	0.9903	0.9412	0.9997	0.9247	0.9525
J48	Fixed Split	0.8533	0.9789	0.9589	0.9832	0.9751	0.9183	0.9718	0.9191	0.9992	0.9384	0.9496
PART	Fixed Split	0.8452	0.9758	0.9579	0.9760	0.9785	0.9192	0.9813	0.9301	0.9997	0.9081	0.9472
MultiLayerPerceptron	Fixed Split	0.8064	0.9755	0.9775	0.9856	0.9786	0.8906	0.9887	0.9360	0.9986	0.9231	0.9460
IB-k	Fixed Split	0.7889	0.9789	0.9304	0.9765	0.9727	0.9715	0.9968	0.9426	0.9991	0.9007	0.9458
REPTree	Fixed Split	0.8409	0.9741	0.9579	0.9830	0.9775	0.8901	0.9631	0.9185	0.9982	0.9405	0.9444
ADTree-Boosting	Fixed Split	0.8509	0.9691	0.9617	0.9810	0.9776	0.8415	0.9786	0.9275	0.9998	0.9399	0.9428
Random-Forest-Bagging	Platt	0.8463	0.9823	0.9420	0.9839	0.9776	0.9597	0.9830	0.8210	0.9994	0.9305	0.9426
Random-Forest-Bagging	Cross Val	0.8463	0.9822	0.9420	0.9839	0.9774	0.9594	0.9823	0.8221	0.9995	0.9305	0.9426
Random-Forest-Boosting	Cross Val	0.8209	0.9836	0.9447	0.9839	0.9754	0.9613	0.9791	0.8282	0.9994	0.9270	0.9403
Random-Forest	Cross Val	0.8449	0.9803	0.9431	0.9839	0.9774	0.9452	0.9783	0.8169	0.9996	0.9274	0.9397
Random-Forest	Platt	0.8446	0.9809	0.9437	0.9839	0.9797	0.9468	0.9821	0.8070	0.9995	0.9195	0.9388
Random-Forest-Boosting	Platt	0.8209	0.9835	0.9447	0.9839	0.9789	0.9613	0.9815	0.8282	0.9994	0.8946	0.9377
MultiLayerPerceptron	Cross Val	0.8170	0.9760	0.9775	0.9856	0.9740	0.8847	0.9825	0.8489	0.9981	0.9324	0.9377
J48	Cross Val	0.8525	0.9807	0.9583	0.9839	0.9724	0.9141	0.9548	0.7991	0.9989	0.9382	0.9353
J48	Platt	0.8525	0.9807	0.9582	0.9839	0.9724	0.9141	0.9548	0.7991	0.9989	0.9382	0.9353
Decision-Table	Fixed Split	0.8516	0.9777	0.9421	0.9832	0.9773	0.8487	0.9196	0.9019	0.9989	0.9405	0.9341
PART	Cross Val	0.8187	0.9742	0.9571	0.9787	0.9765	0.9113	0.9678	0.8378	0.9991	0.9123	0.9333
PART	Platt	0.8184	0.9742	0.9571	0.9787	0.9765	0.9113	0.9678	0.8378	0.9991	0.9123	0.9333
IB-k	Cross Val	0.7851	0.9761	0.9306	0.9761	0.9656	0.9706	0.9890	0.8256	0.9990	0.9058	0.9324
ADTree-Boosting	Cross Val	0.8550	0.9801	0.9637	0.9821	0.9752	0.8314	0.9676	0.8236	0.9994	0.9384	0.9316
REPTree	Cross Val	0.8371	0.9767	0.9569	0.9826	0.9721	0.8869	0.9556	0.8084	0.9990	0.9390	0.9314
REPTree	Platt	0.8361	0.9767	0.9569	0.9826	0.9721	0.8860	0.9556	0.8084	0.9990	0.9390	0.9312
MultiLayerPerceptron	Platt	0.8166	0.9760	0.9655	0.9220	0.9796	0.8847	0.9825	0.8489	0.9980	0.9367	0.9311
ADTree-Bagging	Fixed Split	0.8515	0.9678	0.9634	0.9832	0.9789	0.7568	0.9307	0.9319	0.9996	0.9405	0.9304
IB-k	Platt	0.7851	0.9761	0.9161	0.9691	0.9651	0.9703	0.9890	0.8242	0.9990	0.8946	0.9289
ADTree	Fixed Split	0.8517	0.9678	0.9581	0.9832	0.9788	0.7404	0.8900	0.8961	0.9997	0.9405	0.9206
ADTree-Bagging	Cross Val	0.8538	0.9856	0.9637	0.9842	0.9742	0.7680	0.8979	0.8215	0.9996	0.9390	0.9187
ADTree-Bagging	Platt	0.8526	0.9848	0.9546	0.9839	0.9754	0.7614	0.8975	0.8358	0.9985	0.9390	0.9184
ADTree-Boosting	Platt	0.8535	0.9801	0.9235	0.9063	0.9490	0.8324	0.9658	0.8306	0.9994	0.9384	0.9179
Decision-Table	Cross Val	0.8515	0.9379	0.9420	0.9839	0.9795	0.8318	0.9087	0.8041	0.9982	0.9386	0.9176
Decision-Table	Platt	0.8518	0.9382	0.9420	0.9839	0.9795	0.8275	0.9062	0.8073	0.9982	0.9386	0.9173
SimpleLogistic	Fixed Split	0.8503	0.9733	0.9772	0.9853	0.9808	0.7321	0.8418	0.9258	0.9585	0.9405	0.9166
SMO	Fixed Split	0.8470	0.9725	0.9682	0.9800	0.9805	0.7358	0.8460	0.9269	0.9564	0.9405	0.9154
ADTree	Cross Val	0.8522	0.9848	0.9601	0.9836	0.9702	0.7568	0.8773	0.8026	0.9996	0.9392	0.9126
ADTree	Platt	0.8507	0.9848	0.9601	0.9839	0.9713	0.7416	0.8730	0.8082	0.9985	0.9390	0.9111
Decision-Stump-Boosting	Fixed Split	0.8420	0.9556	0.9567	0.9832	0.9713	0.6992	0.8518	0.9004	0.9980	0.9405	0.9098
SimpleLogistic	Cross Val	0.8491	0.9819	0.9745	0.9850	0.9824	0.7237	0.8289	0.8370	0.9594	0.9388	0.9061
ID3	Fixed Split	0.7967	0.9741	0.9411	0.9712	0.9637	0.8343	0.7937	0.8719	0.9990	0.9054	0.9051
BayesNetGenerator	Platt	0.8534	0.9789	0.9459	0.9464	0.9796	0.7705	0.8211	0.7907	0.9937	0.9370	0.9017
Decision-Stump-Boosting	Cross Val	0.8421	0.9781	0.9578	0.9831	0.9582	0.6962	0.8408	0.8122	0.9973	0.9390	0.9005
BayesNetGenerator	Cross Val	0.8515	0.9811	0.9781	0.9772	0.9786	0.7605	0.8233	0.7907	0.9931	0.8675	0.9002
BayesNetGenerator	Fixed Split	0.8308	0.9385	0.9782	0.9780	0.9765	0.7703	0.8296	0.8725	0.9918	0.8345	0.9001
SMO	Platt	0.8038	0.9334	0.9686	0.9814	0.9830	0.7299	0.8306	0.8565	0.9469	0.9390	0.8973
Decision-Stump-Boosting	Platt	0.8417	0.9781	0.9372	0.9704	0.9581	0.6962	0.8358	0.8148	0.9984	0.9390	0.8970
SimpleLogistic	Platt	0.8269	0.9792	0.9379	0.9577	0.9775	0.7243	0.8263	0.8151	0.9182	0.9380	0.8901
SMO	Cross Val	0.8038	0.9334	0.9686	0.9814	0.9830	0.7299	0.8306	0.8565	0.8565	0.9390	0.8883
Decision-Stump-Bagging	Fixed Split	0.7608	0.9220	0.9421	0.9832	0.9699	0.6712	0.7176	0.8789	0.9266	0.9405	0.8713
Decision-Stump	Fixed Split	0.7608	0.9220	0.9421	0.9832	0.9699	0.6712	0.7101	0.8789	0.9266	0.9405	0.8705
Naïve-Bayes-Simple	Fixed Split	0.8332	0.9382	0.9783	0.9775	0.9430	0.7157	0.7762	0.8771	0.8956	0.7553	0.8690
Decision-Stump	Cross Val	0.7596	0.9524	0.9420	0.9839	0.9581	0.6678	0.7063	0.8230	0.9279	0.9390	0.8660
Decision-Stump	Platt	0.7596	0.9524	0.9420	0.9839	0.9581	0.6678	0.7063	0.8230	0.9279	0.9390	0.8660
Decision-Stump-Bagging	Platt	0.7596	0.9524	0.9420	0.9839	0.9572	0.6678	0.7063	0.8230	0.9279	0.9390	0.8659
Decision-Stump-Bagging	Cross Val	0.7596	0.9524	0.9420	0.9839	0.9581	0.6678	0.7063	0.8105	0.9279	0.9390	0.8647
Naïve-Bayes-Simple	Cross Val	0.8273	0.9811	0.9782	0.9767	0.9540	0.7040	0.7547	0.8160	0.8974	0.7553	0.8645
ID3	Platt	0.7864	0.9770	0.9439	0.9659	0.9475	0.6435	0.8059	0.8082	0.8279	0.9231	0.8629
Naïve-Bayes-Simple	Platt	0.8273	0.9788	0.9451	0.9449	0.9713	0.7040	0.7552	0.8160	0.8974	0.7721	0.8612
LibSVM	Fixed Split	0.7609	0.9671	0.9421	0.9832	0.9815	0.9713	0.5045	0.5525	0.9482	0.9399	0.8551
ID3	Cross Val	0.7866	0.9774	0.9439	0.9659	0.9476	0.5029	0.8059	0.8082	0.8279	0.9204	0.8487
LibSVM	Platt	0.7597	0.9809	0.9420	0.9839	0.9831	0.9717	0.5113	0.3584	0.9427	0.9390	0.8373
LibSVM	Cross Val	0.7597	0.9809	0.9420	0.9839	0.9831	0.9717	0.5113	0.3584	0.3584	0.9390	0.7788
ZeroR	Fixed Split	0.7608	0.8286	0.9421	0.9832	0.9699	0.5013	0.5045	0.5525	0.7887	0.9405	0.7772
ZeroR	Cross Val	0.7596	0.0765	0.9420	0.9839	0.9713	0.4971	0.5113	0.3584	0.7885	0.9390	0.6828
ZeroR	Platt	0.7596	0.0765	0.9420	0.9839	0.9713	0.4971	0.5113	0.3584	0.7885	0.9390	0.6828

Table 3: Accuracy of all algorithms over ten problems and their mean performances in descending order

Algorithm	Val/Cali	Abs_Err	Rel_Err	RMSE	Sqr_Err	Corr.	Pre_Avg	AUC	Margin	Kappa	Preci.	Recall	LIFT	Fallout	F_Mea.	Acc.
Random-Forest-Bagging	FixedSplit	0.078	0.078	0.166	0.034	0.738*	0.202	0.934	0.033	0.583	0.836*	0.578	6.655*	0.016	0.758*	0.956
Random-Forest-Boosting	FixedSplit	0.047	0.047	0.183	0.043	0.682*	0.202	0.896	0.020	0.595	0.843*	0.587	7.589*	0.016	0.689*	0.955
Random-Forest	FixedSplit	0.076	0.076	0.174	0.038	0.667*	0.200	0.892	0.000	0.581	0.834*	0.574	7.506*	0.018	0.676*	0.952
J48	FixedSplit	0.069	0.069	0.192	0.044	0.688*	0.206	0.824	0.001	0.616	0.768*	0.623	5.964*	0.026	0.721*	0.950
PART	FixedSplit	0.062	0.062	0.194	0.046	0.655	0.212	0.851	0.000	0.652	0.730	0.661	7.239	0.029	0.688	0.947
MultiLayerPerceptron	FixedSplit	0.057	0.057	0.187	0.046	0.686	0.197	0.927	0.000	0.676	0.784	0.675	9.462	0.022	0.709	0.946
IB-k	FixedSplit	0.056	0.056	0.199	0.054	0.613	0.219	0.819	0.000	0.612	0.658	0.641	5.350	0.033	0.647	0.946
REPTree	FixedSplit	0.080	0.080	0.197	0.046	0.685*	0.203	0.878	0.006	0.603	0.820*	0.608	9.243*	0.028	0.711*	0.944
ADTree-Boosting	FixedSplit	0.077	0.077	0.186	0.043	0.645	0.217	0.931	0.002	0.643	0.712	0.667	7.755	0.041	0.760*	0.943
Random-Forest-Bagging	Platt	0.095	0.095	0.191	0.046	0.684*	0.190	0.922	0.021	0.533	0.804*	0.550	7.732*	0.034	0.708*	0.943
Random-Forest-Bagging	Cross Val	0.094	0.094	0.188	0.045	0.683*	0.190	0.922	0.018	0.532	0.804*	0.548	7.722*	0.033	0.706*	0.943
Random-Forest-Boosting	Cross Val	0.063	0.063	0.209	0.058	0.613*	0.182	0.879	0.000	0.519	0.831*	0.527	8.732*	0.029	0.613*	0.940
Random-Forest	Cross Val	0.091	0.091	0.194	0.048	0.617*	0.189	0.874	0.000	0.535	0.797*	0.551	8.157*	0.035	0.635*	0.940
Random-Forest	Platt	0.091	0.091	0.194	0.048	0.569	0.198	0.874	0.000	0.550	0.702	0.573	7.147	0.042	0.655*	0.939
Random-Forest-Boosting	Platt	0.069	0.069	0.211	0.060	0.633*	0.188	0.879	0.001	0.545	0.812*	0.558	8.148*	0.033	0.646*	0.938
MultiLayerPerceptron	Cross Val	0.067	0.067	0.205	0.055	0.655	0.193	0.921	0.000	0.651	0.713	0.688	9.310	0.039	0.692	0.938
J48	Cross Val	0.082	0.082	0.215	0.058	0.653*	0.199	0.809	0.002	0.583	0.733*	0.623	6.507*	0.046	0.697*	0.935
J48	Platt	0.082	0.082	0.215	0.058	0.653*	0.199	0.809	0.002	0.583	0.732*	0.624	6.496*	0.046	0.697*	0.935
Decision-Table	FixedSplit	0.091	0.091	0.215	0.056	0.760*	0.199	0.784	0.013	0.524	0.892*	0.541	6.731*	0.038	0.810*	0.934
PART	Cross Val	0.074	0.074	0.219	0.061	0.611	0.184	0.819	0.000	0.600	0.713	0.615	7.903	0.034	0.642	0.933
PART	Platt	0.074	0.074	0.219	0.061	0.611	0.184	0.819	0.000	0.600	0.713	0.615	7.900	0.034	0.642	0.933
IB-k	Cross Val	0.070	0.070	0.225	0.068	0.575	0.212	0.804	0.000	0.572	0.612	0.632	5.612	0.052	0.617	0.932
ADTree-Boosting	Cross Val	0.088	0.088	0.205	0.054	0.608	0.197	0.915	0.003	0.600	0.717	0.627	8.891	0.049	0.648	0.932
REPTree	Cross Val	0.090	0.090	0.214	0.057	0.658*	0.201	0.866	0.006	0.587	0.729*	0.631	8.442*	0.051	0.705*	0.931
REPTree	Platt	0.090	0.090	0.214	0.057	0.659*	0.202	0.866	0.006	0.588	0.728*	0.633	8.434*	0.052	0.706*	0.931
MultiLayerPerceptron	Platt	0.099	0.099	0.219	0.057	0.640	0.201	0.921	0.001	0.624	0.670	0.736	6.983	0.046	0.666	0.931
ADTree-Bagging	FixedSplit	0.155	0.155	0.216	0.058	0.735*	0.212	0.928	0.043	0.578	0.868*	0.604	7.704*	0.054	0.789*	0.930
IB-k	Platt	0.091	0.091	0.229	0.068	0.580	0.217	0.805	0.015	0.577	0.604	0.651	5.415	0.056	0.625	0.929
ADTree	FixedSplit	0.160	0.160	0.226	0.063	0.703*	0.197	0.909	0.033	0.554	0.854*	0.574	7.540*	0.047	0.766*	0.921
ADTree-Bagging	Cross Val	0.164	0.164	0.233	0.067	0.629*	0.212	0.915	0.027	0.549	0.769*	0.611	10.975*	0.078	0.681*	0.919
ADTree-Bagging	Platt	0.171	0.171	0.238	0.067	0.636*	0.212	0.915	0.041	0.551	0.768*	0.624	10.670*	0.078	0.683*	0.918
ADTree-Boosting	Platt	0.138	0.138	0.232	0.063	0.591	0.216	0.915	0.019	0.572	0.625	0.720	5.406	0.066	0.625	0.918
Decision-Table	Cross Val	0.109	0.109	0.243	0.071	0.616*	0.193	0.786	0.008	0.480	0.739*	0.542	6.739*	0.057	0.669*	0.918
Decision-Table	Platt	0.109	0.109	0.243	0.071	0.616*	0.197	0.786	0.008	0.480	0.736*	0.546	6.733*	0.061	0.670*	0.917
SimpleLogistic	FixedSplit	0.163	0.163	0.251	0.080	0.682*	0.203	0.894	0.051	0.605	0.828*	0.627	11.102*	0.056	0.745*	0.917
SMO	FixedSplit	0.085	0.085	0.265	0.085	0.674*	0.203	0.791	0.000	0.601	0.794*	0.639	9.390*	0.057	0.740*	0.915
ADTree	Cross Val	0.166	0.166	0.240	0.070	0.555	0.207	0.898	0.016	0.541	0.716	0.604	9.097	0.077	0.610	0.913
ADTree	Platt	0.179	0.179	0.247	0.073	0.704*	0.218	0.898	0.036	0.486	0.791*	0.571	5.199*	0.092	0.794*	0.911
Decision-Stump-Boosting	FixedSplit	0.134	0.134	0.230	0.065	0.627*	0.182	0.891	0.015	0.484	0.858*	0.507	7.824*	0.042	0.683*	0.910
SimpleLogistic	Cross Val	0.133	0.133	0.237	0.068	0.583	0.194	0.904	0.002	0.572	0.741	0.615	10.959	0.072	0.646	0.906
ID3	FixedSplit	0.148	0.148	0.342	0.147	0.527	0.188	0.758	0.000	0.522	0.636	0.557	4.928	0.044	0.590	0.905
BayesNetGenerator	Platt	0.154	0.154	0.255	0.078	0.587	0.208	0.911	0.009	0.561	0.686	0.691	8.212	0.082	0.635	0.902
Decision-Stump-Boosting	Cross Val	0.142	0.142	0.245	0.075	0.578*	0.180	0.884	0.010	0.511	0.702*	0.561	8.087*	0.062	0.647*	0.900
BayesNetGenerator	Cross Val	0.113	0.113	0.249	0.080	0.621	0.215	0.911	0.000	0.615	0.690	0.715	8.804	0.087	0.693	0.900
BayesNetGenerator	FixedSplit	0.112	0.112	0.251	0.076	0.630	0.239	0.919	0.000	0.625	0.685	0.740	7.883	0.087	0.702	0.900
SMO	Platt	0.108	0.108	0.293	0.102	0.609*	0.184	0.767	0.006	0.535	0.736*	0.603	9.765*	0.069	0.674*	0.897
Decision-Stump-Boosting	Platt	0.182	0.182	0.263	0.079	0.587*	0.190	0.884	0.030	0.519	0.655*	0.623	6.355*	0.069	0.658*	0.897
SimpleLogistic	Platt	0.188	0.188	0.266	0.079	0.546	0.192	0.904	0.020	0.523	0.655	0.644	7.325	0.079	0.602	0.890
SMO	Cross Val	0.112	0.112	0.308	0.112	0.594*	0.207	0.763	0.000	0.521	0.717*	0.610	9.509*	0.083	0.668*	0.888
Decision-Stump-Bagging	FixedSplit	0.188	0.188	0.282	0.091	0.522*	0.179	0.810	0.074	0.309	0.756*	0.402	2.503*	0.073	0.771*	0.871
Decision-Stump	FixedSplit	0.187	0.187	0.285	0.094	0.623*	0.175	0.774	0.082	0.308	0.759*	0.397	2.508*	0.070	0.767*	0.871
Naïve-Bayes-Simple	FixedSplit	0.143	0.143	0.303	0.105	0.557	0.238	0.900	0.000	0.545	0.629	0.702	6.546	0.108	0.640	0.869
Decision-Stump	Cross Val	0.211	0.211	0.305	0.106	0.516*	0.161	0.742	0.073	0.300	0.697*	0.376	5.283*	0.078	0.633*	0.866
Decision-Stump	Platt	0.225	0.225	0.308	0.107	0.516*	0.161	0.742	0.099	0.300	0.697*	0.376	5.283*	0.078	0.633*	0.866
Decision-Stump-Bagging	Platt	0.225	0.225	0.307	0.106	0.516*	0.161	0.756	0.098	0.300	0.696*	0.376	5.240*	0.078	0.633*	0.866
Decision-Stump-Bagging	Cross Val	0.212	0.212	0.304	0.105	0.514*	0.164	0.756	0.075	0.298	0.692*	0.379	5.268*	0.081	0.632*	0.865
Naïve-Bayes-Simple	Cross Val	0.145	0.145	0.303	0.110	0.540	0.219	0.890	0.000	0.527	0.631	0.674	7.458	0.114	0.626	0.864
ID3	Platt	0.137	0.137	0.342	0.137	0.398	0.123	0.683	0.000	0.372	0.593	0.404	4.895	0.038	0.450	0.863
Naïve-Bayes-Simple	Platt	0.183	0.183	0.305	0.106	0.482	0.218	0.890	0.008	0.456	0.657	0.641	8.310	0.113	0.555	0.861
LibSVM	FixedSplit	0.145	0.145	0.319	0.145	0.474*	0.084	0.652	0.000	0.325	0.761*	0.308	8.570*	0.004	0.557*	0.855
ID3	Cross Val	0.337	0.337	0.518	0.337	0.401*	0.104	0.669	0.000	0.343	0.559*	0.370	5.225*	0.034	0.447*	0.849
LibSVM	Platt	0.163	0.163	0.328	0.163	0.661*	0.175	0.654	0.000	0.326	0.770*	0.413	8.217*	0.104	0.646*	0.837
LibSVM	Cross Val	0.221	0.221	0.384	0.221	0.620*	0.259	0.618	0.000	0.246	0.663*	0.440	7.596*	0.204	0.593*	0.779
ZeroR	FixedSplit	0.279	0.279	0.348	0.139	NaN*	0.100	0.500	0.221	0.000	0.501*	0.100	1*	0.100	0.668*	0.777
ZeroR	Cross Val	0.307	0.307	0.366	0.157	NaN*	0.300	0.500	0.250	0.000	0.311*	0.300	1*	0.300	0.445*	0.683
ZeroR	Platt	0.307	0.307	0.366	0.157	NaN*	0.300	0.500	0.250	0.000	0.311*	0.300	1*	0.300	0.445*	0.683

Table 4: Average performances for each learning algorithm by metric (average over ten problems)

Algorithm	Val./Cal.	Adult	CovType	Celegans	Droso	DS1_100	Letter	PenDigits	Satellite	Shuttle	Tic2000	Average
IB-k	FixedSplit	0.7889	0.9789	0.9304	0.9765	0.9727	0.9715	0.9968	0.9426	0.9991	0.9007	0.9458
Decision Table	LogitBoost	0.8449	0.9797	0.9670	0.9837	0.9762	0.9156	0.9594	0.8032	0.9798	0.9349	0.9344
Decision-Table	FixedSplit	0.8516	0.9777	0.9421	0.9832	0.9773	0.8487	0.9196	0.9019	0.9989	0.9405	0.9341
Ibk	LogitBoost	0.7833	0.9765	0.9304	0.9765	0.9672	0.9717	0.9898	0.8294	0.9992	0.9054	0.9330
Ibk	AddReg	0.7834	0.9673	0.9383	0.9813	0.9690	0.9710	0.9908	0.8250	0.9992	0.9036	0.9329
IB-k	CrossVal	0.7851	0.9761	0.9306	0.9761	0.9656	0.9706	0.9890	0.8256	0.9990	0.9058	0.9324
IB-k	PlattScaling	0.7851	0.9761	0.9161	0.9691	0.9651	0.9703	0.9890	0.8242	0.9990	0.8946	0.9289
Decision-Table	CrossVal	0.8515	0.9379	0.9420	0.9839	0.9795	0.8318	0.9087	0.8041	0.9982	0.9386	0.9176
Decision-Table	PlattScaling	0.8518	0.9382	0.9420	0.9839	0.9795	0.8275	0.9062	0.8073	0.9982	0.9386	0.9173
Decision Stump	LogitBoost	0.8519	0.9784	0.9567	0.9841	0.9700	0.7383	0.8650	0.8125	0.9984	0.9390	0.9094
Decision-Stump	FixedSplit	0.7608	0.9220	0.9421	0.9832	0.9699	0.6712	0.7101	0.8789	0.9266	0.9405	0.8705
Decision-Stump	CrossVal	0.7596	0.9524	0.9420	0.9839	0.9581	0.6678	0.7063	0.8230	0.9279	0.9390	0.8660
Decision-Stump	PlattScaling	0.7596	0.9524	0.9420	0.9839	0.9581	0.6678	0.7063	0.8230	0.9279	0.9390	0.8660
Decision Table	AddReg	0.7802	0.9437	0.9487	0.9839	0.9704	0.6745	0.7934	0.6885	0.9276	0.9382	0.8649
LibSVM	FixedSplit	0.7609	0.9671	0.9421	0.9832	0.9815	0.9713	0.5045	0.5525	0.9482	0.9399	0.8551
LibSVM	PlattScaling	0.7597	0.9809	0.9420	0.9839	0.9831	0.9717	0.5113	0.3584	0.9427	0.9390	0.8373
LibSVM	LogitBoost	0.7595	0.9743	0.9432	0.9832	0.9813	0.9772	0.5117	0.3584	0.9490	0.9224	0.8360
Decision Stump	AddReg	0.7888	0.9249	0.9420	0.9839	0.9713	0.5215	0.6225	0.8277	0.7885	0.9390	0.8310
LibSVM	AddReg	0.7596	0.9235	0.9420	0.9839	0.9713	0.5029	0.5113	0.6416	0.7885	0.9390	0.7964
ZeroR	AddReg	0.7596	0.9235	0.9420	0.9839	0.9713	0.5029	0.5113	0.6416	0.7885	0.9390	0.7964
Decision Stump	IsoReg	0.7596	0.9235	0.9420	0.9839	0.9713	0.5029	0.5113	0.6416	0.7885	0.9390	0.7964
Decision Table	IsoReg	0.7596	0.9235	0.9420	0.9839	0.9713	0.5029	0.5113	0.6416	0.7885	0.9390	0.7964
Ibk	IsoReg	0.7596	0.9235	0.9420	0.9839	0.9713	0.5029	0.5113	0.6416	0.7885	0.9390	0.7964
LibSVM	IsoReg	0.7596	0.9235	0.9420	0.9839	0.9713	0.5029	0.5113	0.6416	0.7885	0.9390	0.7964
ZeroR	IsoReg	0.7596	0.9235	0.9420	0.9839	0.9713	0.5029	0.5113	0.6416	0.7885	0.9390	0.7964
LibSVM	CrossVal	0.7597	0.9809	0.9420	0.9839	0.9831	0.9717	0.5113	0.3584	0.3584	0.9390	0.7788
ZeroR	FixedSplit	0.7608	0.8286	0.9421	0.9832	0.9699	0.5013	0.5045	0.5525	0.7887	0.9405	0.7772
ZeroR	CrossVal	0.7596	0.0765	0.9420	0.9839	0.9713	0.4971	0.5113	0.3584	0.7885	0.9390	0.6828
ZeroR	PlattScaling	0.7596	0.0765	0.9420	0.9839	0.9713	0.4971	0.5113	0.3584	0.7885	0.9390	0.6828
ZeroR	LogitBoost	0.7596	0.0765	0.9420	0.9839	0.9713	0.4971	0.5113	0.3584	0.7885	0.9390	0.6828

Table 5: Accuracy of selected algorithms across Fixed Split, Cross Validation and all four types of Calibration methods over ten problems and their mean performances in descending order

Algorithm	Val/Cal.	Abs_Err	Rel_Err	RMSE	Sqr_Err	Corr.	Pre_Avg	AUC	Margin	Kappa	Preci.	Recall	LIFT	Fallout	F_Mea.	Acc.
IB-k	FixedSplit	0.056	0.056	0.199	0.054	0.613	0.219	0.819	0.000	0.612	0.658	0.641	5.350	0.033	0.647	0.9458
Decision Table	LogitBoost	0.089	0.089	0.205	0.050	0.622	0.208	0.917	0.002	0.604	0.755	0.621	9.641	0.047	0.649	0.9344
Decision-Table	FixedSplit	0.091	0.091	0.215	0.056	0.760*	0.199	0.784	0.013	0.524	0.892*	0.541	6.731*	0.038	0.810*	0.9341
Ibk	LogitBoost	0.068	0.068	0.223	0.067	0.577	0.211	0.800	0.000	0.574	0.615	0.631	5.682	0.051	0.619	0.9330
Ibk	AddReg	0.068	Inf +	0.220	0.066	0.570	0.211	0.000	1.000	0.552	0.564	0.624	5.108	0.069	0.565	0.9329
IB-k	CrossVal	0.070	0.070	0.225	0.068	0.575	0.212	0.804	0.000	0.572	0.612	0.632	5.612	0.052	0.617	0.9324
IB-k	PlattScaling	0.091	0.091	0.229	0.068	0.580	0.217	0.805	0.015	0.577	0.604	0.651	5.415	0.056	0.625	0.9289
Decision-Table	CrossVal	0.109	0.109	0.243	0.071	0.616*	0.193	0.786	0.008	0.480	0.739*	0.542	6.739*	0.057	0.669*	0.9176
Decision-Table	PlattScaling	0.109	0.109	0.243	0.071	0.616*	0.197	0.786	0.008	0.480	0.736*	0.546	6.733*	0.061	0.670*	0.9173
Decision Stump	LogitBoost	0.138	0.138	0.238	0.071	0.534	0.187	0.893	0.005	0.516	0.730	0.562	10.203	0.062	0.583	0.9094
Decision-Stump	FixedSplit	0.187	0.187	0.285	0.094	0.623*	0.175	0.774	0.082	0.308	0.759*	0.397	2.508*	0.070	0.767*	0.8705
Decision-Stump	CrossVal	0.211	0.211	0.305	0.106	0.516*	0.161	0.742	0.073	0.300	0.697*	0.376	5.283*	0.078	0.633*	0.8660
Decision-Stump	PlattScaling	0.225	0.225	0.308	0.107	0.516*	0.161	0.742	0.099	0.300	0.697*	0.376	5.283*	0.078	0.633*	0.8660
Decision Table	AddReg	0.122	Inf +	0.238	0.067	0.600	0.215	0.000	1.000	0.274	0.204	1.000	1.000	1.000	0.304	0.8649
LibSVM	FixedSplit	0.145	0.145	0.319	0.145	0.474*	0.084	0.652	0.000	0.325	0.761*	0.308	8.570*	0.004	0.557*	0.8551
LibSVM	PlattScaling	0.163	0.163	0.328	0.163	0.661*	0.175	0.654	0.000	0.326	0.770*	0.413	8.217*	0.104	0.646*	0.8373
LibSVM	LogitBoost	0.166	0.166	0.311	0.142	0.439*	0.179	0.695	0.024	0.336	0.698*	0.426	7.067*	0.107	0.445*	0.8360
Decision Stump	AddReg	0.167	Inf +	0.260	0.079	0.542	0.223	0.000	1.000	0.108	0.204	1.000	1.000	1.000	0.304	0.8310
LibSVM	AddReg	0.485	Inf +	0.526	0.341	1.38E-07*	0.078	0.000	1.000	0.000	0.204	1.000	1.000	1.000	0.304	0.7964
ZeroR	AddReg	0.485	Inf +	0.526	0.341	1.38E-07*	0.078	0.000	1.000	0.000	0.204	1.000	1.000	1.000	0.304	0.7964
Decision Stump	IsoReg	0.263	Inf +	0.366	0.157	1.02E-07*	0.271	0.000	1.000	0.000	0.204	1.000	1.000	1.000	0.304	0.7964
Decision Table	IsoReg	0.263	Inf +	0.366	0.157	1.02E-07*	0.271	0.000	1.000	0.000	0.204	1.000	1.000	1.000	0.304	0.7964
Ibk	IsoReg	0.263	Inf +	0.366	0.157	1.02E-07*	0.271	0.000	1.000	0.000	0.204	1.000	1.000	1.000	0.304	0.7964
LibSVM	IsoReg	0.263	Inf +	0.366	0.157	1.02E-07*	0.271	0.000	1.000	0.000	0.204	1.000	1.000	1.000	0.304	0.7964
ZeroR	IsoReg	0.263	Inf +	0.366	0.157	1.02E-07*	0.271	0.000	1.000	0.000	0.204	1.000	1.000	1.000	0.304	0.7964
LibSVM	CrossVal	0.221	0.221	0.384	0.221	0.620*	0.259	0.618	0.000	0.246	0.663*	0.440	7.596*	0.204	0.593*	0.7788
ZeroR	FixedSplit	0.279	0.279	0.348	0.139	NaN*	0.100	0.500	0.221	0.000	0.501*	0.100	1*	0.100	0.668*	0.7772
ZeroR	CrossVal	0.307	0.307	0.366	0.157	NaN*	0.300	0.500	0.250	0.000	0.311*	0.300	1*	0.300	0.445*	0.6828
ZeroR	PlattScaling	0.307	0.307	0.366	0.157	NaN*	0.300	0.500	0.250	0.000	0.311*	0.300	1*	0.300	0.445*	0.6828
ZeroR	LogitBoost	0.307	0.307	0.366	0.157	NaN*	0.300	0.500	0.250	0.000	0.311*	0.300	1*	0.300	0.445*	0.6828

Table 6: Average performances for selected learning algorithm by metric, across Fixed Split, Cross Validation and all four types of Calibration methods, (average over ten problems)

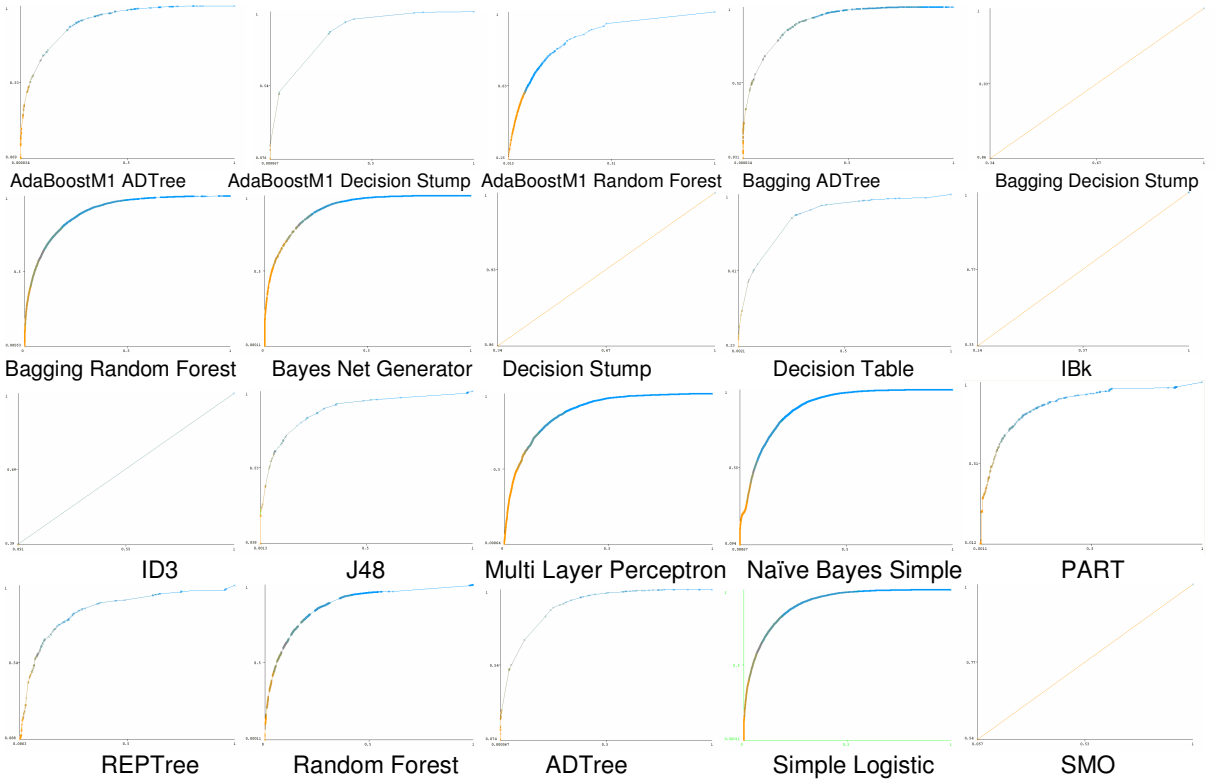

Figure 7: ROC graphs of twenty algorithms for Adult problem (X Axis-False Positive Rate, Y-Axis True Positive Rate).

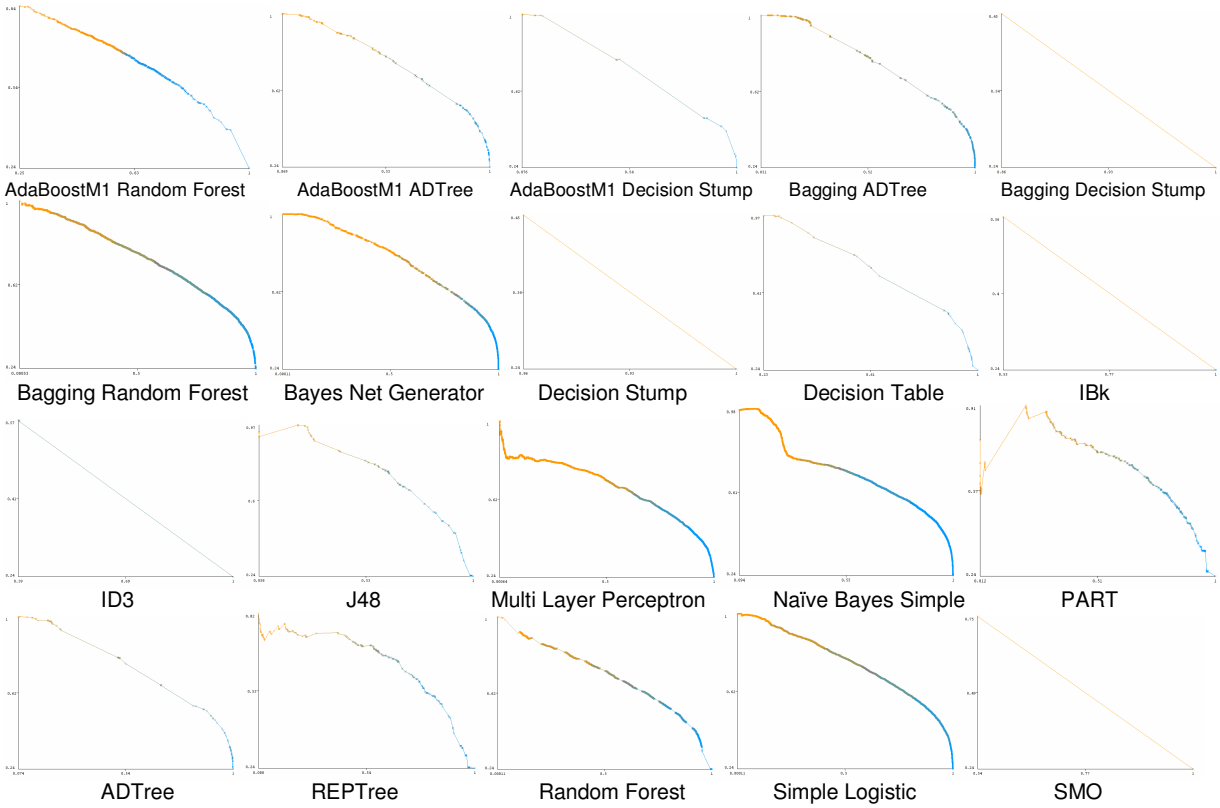

Figure 8: Precision/Recall graphs of twenty algorithms for Adult problem (X Axis-Recall, Y Axis-Precision).

7. Comparison of results

Results of supervised learning techniques depend upon many things like type of dataset, number of instances in dataset, algorithm used for testing, process used for producing output etc. Data mining is a study of knowledge discovery in large datasets. First of all we present the comparison of different datasets based on average accuracy through different processes followed by the performance of different algorithm based on their average accuracy over three processes of major study. Figure 9 includes average performance of different problems from major study. Droso problem has performed best with an average accuracy of 97.78%, whereas Letter problem has performed worst with average performance with 80.57% average accuracy.

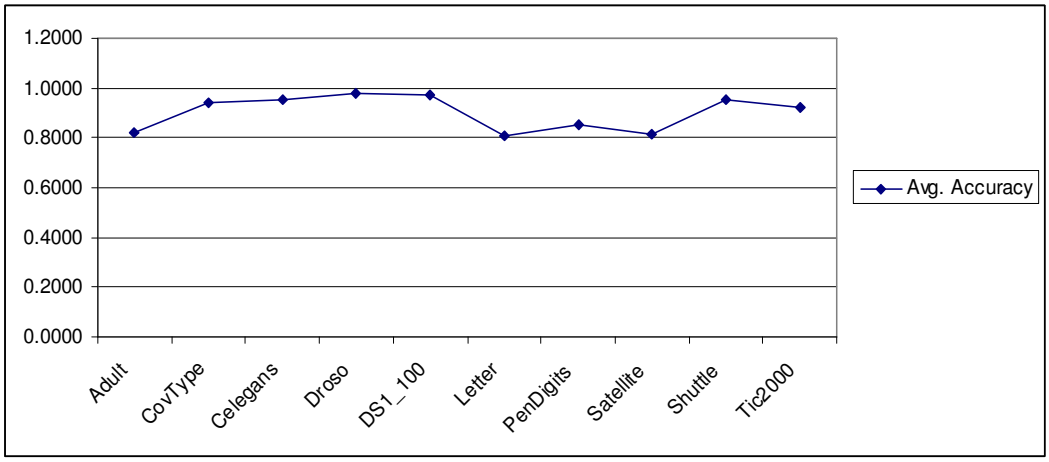

Figure 9: Average performance of different processes for different problems

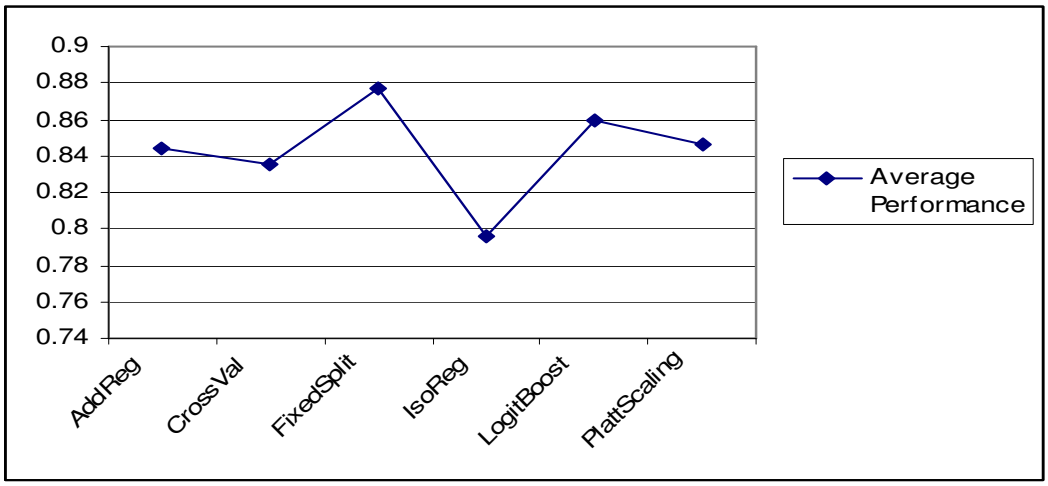

Figure 10: Average of average performances from five algorithms over all ten datasets included in minor study.

Figure 10 includes the average performance of all the processes from the average of five algorithms over all ten datasets included in minor study. Fixed split is highest performer, but its performance is over fitted, whereas post pruning through isotonic regression is least performer with 79.64% performance. Cross Validation and other three post pruning methods have pruned the models more appropriately. Among these four LogitBoost has performed best, whereas cross validation has performed least. Reason behind low performance of cross validation is exclusion of one tenth of training dataset while processing final model.

Other dimension of comparison includes two comprehensive studies that have been performed yet. First one is Statlog (King et al. [15]) and other is recent one (Caruana et al. [7]). One of the major differences between earlier two studies and current study, is about the selection of datasets i. e. earlier studies were mostly based on small datasets, whereas present study includes most of the datasets that are bigger in size and simple rule of probability states that increasing number of instances produces more accurate results and minimizes the chances of deviation. When Statlog study was conducted, algorithms like

Random Forest etc. were not being developed and data mining was in its initial phase of development. During last two decades data mining field has become mature enough. Statlog study presented the results for individual datasets. We compiled and processed the data for comparison and found that piecewise linear classifier DIPOL92 to be performing best for their tests, whereas Decision Tree was ranked second followed by the Back Propagation and kNN (k Nearest Neighbor) etc. Clearly, the absence of better algorithms like Random Forest at that time kept the high quality performances far away from current standards. Today, we have far better results than the results presented in Statlog.

Other recently conducted study (Caruana et al. [7]) presented the results that Boosted decision tree with platt scaling algorithm is the best performer, whereas Random Forest with platt scaling is the second best performer. Bagging and Boosting was not applied upon Random Forest. Experiments were performed through cross validation, Platt Scaling and Isotonic Regression. Top performers were Boosted Decision Tree, Random Forest, Bagged decision tree, SVM (Support Vector Machine), ANN (Artificial Neural Network) etc. Results of our study have marked Bagged Random Forest to be the best performer followed by J48, PART, Multi Layer Perceptron and IBk etc.

Finally results of present study are compared with the best known results ever claimed for problems included in study. For adult dataset best possible result is claimed for FSS Naïve Bayes in the description of datasets of UCI repositories (Blake et al. [5]) having 85.95% accuracy, where 32561 instances were used for training and 16281 instances for testing. Present study has used 9768 instances for training and 39074 instances for testing and best result is 85.50% for ADTree-Boosting algorithm with cross validation, which confirms our claim that training with twenty percent training instances for large datasets achieve significant maturity in results. For other problems as well results are up to the mark with best possible results ever being obtained.

8. Conclusion and Future Directions

Data mining has marked substantial progress in last two decades. Learning methods such as boosting, random forests, bagging and IBk etc. have achieved excellent performance that would have been difficult to obtain just fifteen years ago. Calibration with either Platt's method, Logit Boost, Additive Regression or Isotonic Regression is remarkably effective at obtaining excellent performance on the probability metrics from learning algorithms that performed well on the ordering metrics. Calibration dramatically improves the performance of Random Forests, ADTree, Decision stumps and Naive Bayes etc. and provides a noticeable improvement for random forests. With excellent performance over all fifteen metrics, calibrated Random Forest trees were the best learning algorithms overall. ADTree, IBk, J48 and MultiLayer Perceotron were quite close to it. Algorithm ZeroR has registered worst performance, but has registered a little improvement through Additive Regression based calibration. As the environmental factors like type of problems, size of dataset etc. may affect the performance of the algorithm, even better algorithms sometimes may result in bad results. Even after having a significant margin between best and worst performances, there exist chances for improvement. Authors will continue to work for the improvement of the processing environment of badly performing algorithms and for the improvement of the best algorithms as well as for the development of new algorithms for the field.

9. References

1. Atlas L., Connor J., and Park D. *"A performance comparison of trained multi-layer perceptrons and trained classification trees"*. In *Systems, man and cybernetics: proceedings of the 1989 IEEE international conference*, pages 915–920, Cambridge, Ma. Hyatt Regency, 1991

2. Ayer M., Brunk H., Ewing G., Reid W. & Silverman E. *"An empirical distribution function for sampling with incomplete information"*. Annals of Mathematical Statistics, 5, 641-647, 1955

3. Bauer E. and Kohavi R. *"An empirical comparison of voting classification algorithms: Bagging, boosting, and variants"*. Machine Learning, 36, 1999

4. Berry C. C. *"The kappa statistic". Journal of the American Medical Association, Linguistics (COLING-90),* volume 2, pages 251-256, 1992

5. Blake C. and Merz C., UCI repository of machine learning databases, 1998

6. Breiman L., Friedman J. H., Olshen R. A. and Stone C. J. *"Classification and Regression Trees"*. Wadsworth and Brooks, Monterey, CA., 1984

7. Caruana Rich and Niculescu-Mizil Alexandru. *"An Empirical Comparison of Supervised Learning Algorithms".* Proceedings of the 23 rd International Conference on Machine Learning, Pittsburgh, PA, 2006

8. Cooper G. F., Aliferis C. F., Ambrosino R., Aronis J., Buchanan B. G., Caruana R., Fine M. J., Glymour C., Gordon G., Hanusa B. H., Janosky J. E., Meek C., Mitchell T., Richardson T. and Spirtes P. *"An evaluation of machine learning methods for predicting pneumonia mortality".* Artificial Intelligence in Medicine, 9, 1997

9. Fahrmeir, L., Haussler, W., and Tutz, G. *"Diskriminanz analyse".* In Fahrmeir, L. and Hamerle, A., editors, *Multivariate statistische Verfahren*. Verlag de Gruyter, Berlin, 1984

10. Fayyad U., Piatetsky-Shapiro G. and P. Smyth. *"The KDD process for extracting useful knowledge from volumes of data".* CACM 39 (11), pp. 27-34, 1996

11. Friedman J., Hastie T. and Tibshirani R. *"Additive Logistic Regression: a Statistical View of Boosting"*. Stanford University,1998

12. Giudici P. *"Applied data mining".* John Wiley and Sons. New York, 2003

13. Gorman R. P. and Sejnowski T. J. *"Analysis of hidden units in a layered network trained to classify sonar targets"*. Neural networks, 1 (Part 1):75–89, 1988

14. Hofmann H. J. *"Die anwendung des cart-verfahrens zur statistischen bonitatsanalyse von konsumentenkrediten".* Zeitschrift fur Betriebswirtschaft, 60:941–962, 1990

15. King R., Feng C. and Shutherland A. *"Statlog: comparison of classi_cation algorithms on large real world problems".* Applied Artificial Intelligence, 9, 1995

16. Kirkwood C., Andrews B. and Mowforth P. *"Automatic detection of gait events: a case study using inductive learning techniques"*. Journal of biomedical engineering, 11(23):511–516, 1989

17. Komarek P., Gray A., Liu T. and Moore A. *"High Dimensional Probabilistic Classification for Drug Discovery"*, Biostatics, COMPSTAT, 2004

18. LeCun Y., Jackel L. D., Bottou L., Brunot A., Cortes C., Denker J. S., Drucker H., Guyon I., Muller U. A., Sackinger E., Simard P. and Vapnik V. *"Comparison of learning algorithms for handwritten digit recognition"*. International Conference on Artificial Neural Networks (pp. 53{60).Paris, 1995

19. Lim T. S., Loh W.-Y. and Shih Y. S. *"A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms"*. Machine Learning, 40, 203-228, 2000

20. Mitchell T., Buchanan B., DeJon G., Dietterich T., Rosenbloom P. and Waibel A. *"Machine Learning".* Annual Review of Computer Science, vol. 4, pp. 417-433, 1990

21. Niculescu-Mizil A. and Caruana R. *"Predicting good probabilities with supervised learning"*. Proc. 22nd International Conference on Machine Learning (ICML'05), 2005

22. Nishisato S. *"Analysis of Categorical Data: Dual Scaling and its Applications"*. University of Toronto Press, Toronto, 1980

23. Perlich C., Provost F. and Simono J. S. *"Tree induction vs. logistic regression: a learning-curve analysis".* J. Mach. Learn. Res., 4, 211-255, 2003

24. Platt J. *"Probabilistic outputs for support vector machines and comparison to regularized likelihood methods"*. Adv. in Large Margin Classifiers, 1999

25. Provost F. and Domingos P. *"Tree induction for probability-based rankings"*. Machine Learning, 2003

26. Provost Foster J. and Kohavi Ron, *"On Applied Research in Machine Learning"*. Machine Learning 30 (2-3): 127-132, 1998

27. Provost F., Jensen D. and Oates T. *"Efficient progressive sampling"*. Fifth ACM SIGKDD, International Conference on Knowledge Discovery and Data Mining. San Diego, USA. 1999

28. Ripley B. *"Statistical aspects of neural networks"*. Chaos and Networks - Statistical and Probabilistic Aspects. Chapman and Hall, 1993

29. Robertson T., Wright F. and Dykstra R. *"Order restricted statistical inference"*. John Wiley and Sons, New York, 1988

30. Shadmehr R. and D'Argenio Z. *"A comparison of a neural network based estimator and two statistical estimators in a sparse and noisy environment"*. In *IJCNN-90: proceedings of the international joint conference on neural networks*, pages 289–292, Ann Arbor, MI. IEEE Neural Networks Council, 1990

31. Sonnenburg S, Rätsch G. and Schäfer C. *"Learning interpretable SVMs for biological sequence classification".* Research in Computational Molecular Biology, Springer Verlag, pages 389-407, 2005

32. Spikovska L. and Reid M. B., *"An empirical comparison of id3 and honns for distortion invariant object recognition*". In TAI-90: tools for artificial intelligence: proceedings of the 2nd international IEEE conference, Los Alamitos, CA. IEEE Computer Society Press, 1990

33. Witten I. H. and Frank E. *"Data Mining: Practical machine learning tools and techniques with java implementations"*. Morgan Kaufmann, 2000

34. Yoav Freund, Robert E. Schapire. *"Experiments with a new boosting algorithm"*. Thirteenth International Conference on Machine Learning, San Francisco, 148-156, 1996

35. Zadrozny B. and Elkan C. *"Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers"*. ICML, 2001

36. Zadrozny B. and Elkan C. *"Transforming classifier scores into accurate multi-class probability estimates"*. KDD, 2002