Generalized FLIC: Learning with misclassification for Binary Classifiers

By
Arunabha Choudhury

Submitted to the graduate degree program in Electrical Engineering and Computer Science and the Graduate faculty of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Science.

Chairperson: Jerzy W. Grzymala-Busse

Swapan Chakrabarti

Bo Luo

Date Defended: 18th of November, 2014

The Thesis Committee for **Arunabha Choudhury**

certifies that this is the approved version of the following thesis:

Generalized FLIC: Learning with misclassification for Binary Classifiers

Chairperson: Jerzy W. Grzymala-Busse

Date Approved: 18th Nov, 2014

Generalized FLIC: Learning with misclassification for Binary Classifiers

Abstract

By

Arunabha Choudhury

Abstract

This work formally introduces a generalized fuzzy logic and interval clustering (FLIC) technique which, when integrated with existing supervised learning algorithms, improves their performance. FLIC is a method that was first integrated with neural network in order to improve neural network's performance in drug discovery using high throughput screening (HTS). This research strictly focuses on binary classification problems and generalizes the FLIC in order to incorporate it with other machine learning algorithms. In most binary classification problems, the class boundary is not linear. This pose a major problem when the number of outliers are significantly high, degrading the performance of the supervised learning function. FLIC identifies these misclassifications before the training set is introduced to the learning algorithm. This allows the supervised learning algorithm to learn more efficiently since it is now aware of those misclassifications. Although the proposed method performs well with most binary classification problems, it does significantly well for data set with high class asymmetry. The proposed method has been tested on four well known data sets of which three are from UCI Machine Learning repository and one from BigML. Tests have been conducted with three well known supervised learning techniques: Decision Tree, Logistic Regression and Naive Bayes. The results from the experiments show significant improvement in performance. The paper begins with a formal introduction to the core idea this research is based upon. It then discusses a list of other methods that have either inspired this research or have been referred to, in order to formalize the techniques. Subsequent sections discuss the methodology and the algorithm which is followed by results and conclusion.

Keyword: supervised learning, binary classification, fuzzy logic, clustering

To my parents and loving sister and all the dear friends without whose support, this would not have been possible

Acknowledgement

Dr. Jerzy Grzymala Busse, my advisor, for his support and valuable feedback at every step of my dissertation to help me successfully complete this work.

Dr. Swapan Chakrabarti, for introducing me to fuzzy logic, without whose support and teaching, I could not have come up with this novel method.

Dr. Bo Luo, for his valuable teaching in Database Management and Information Retrieval that helped me to be better equipped with various practical aspects of Intelligent Informatics.

Ghaith Shabsigh, for working with me to come up with the novel approach FLIC that is the core of this research work and at the same time being a wonderful colleague and friend during my stay at University of Kansas.

All the faculty members, whose teaching helped me at every step of my dissertation.

University of Kansas and Department of EECS, for awarding me full scholarship that helped me to be financially stable.

My parents and my sister, for their constant support, encouragement and curiosity that kept me motivated at every step to achieve my goals.

All my friends at University of Kansas and in India, who have always been there to help me during the entire process to make this work a success.

Contents

1-Introduction

The term "Fuzzy Logic" was first proposed by Lotfi A. Zadeh in 1965 [1]. Since then, fuzzy logic has been applied to numerous fields in machine learning and data mining with success. The use of fuzzy logic is not limited to set theory and artificial intelligence but also encompasses fields like control theory. Even though Zadeh formalized fuzzy logics in 1965, it had however been studied since the 1920s. Work of Lukasiewicz and Tarski [2] are worth a mention as they studied these logics as infinite-valued logics. Fuzzy logic is based upon the fuzzy set theory and we continue our discussion by formalizing fuzzy sets.

According to Zadeh in [1], a fuzzy set is a class of objects with a continuum of grades of membership. Such a set is characterized by a membership (characteristic) function which assigns to each object a grade of membership ranging between zero and one. Below is a formal definition of fuzzy set:

Let X be a space of points (objects), with a generic element of X denoted by x. A fuzzy set (class) A in X is characterized by a membership function $\mu_A(x)$ which associates with each point in X a real number in the interval [0, 1], with the value of $\mu_A(x)$ at x representing the "grade of membership" of x in A. Thus, nearer the value of $\mu_A(x)$ to unity, higher the grade of membership of x in A. In case of ordinary or crisp set, the membership function $\mu_A(x)$ takes only two values 0 and 1.

There are numerous ways of finding the membership function and in most cases it is problem dependent. In this research, the Euclidean Distance method has been considered for finding the membership function for each data points. Below is a formal definition for Euclidean distance between 2 arbitrary vectors in Cartesian coordinate system:

Let us consider 2 vectors,

$$p = (p_1, p_2, \dots, p_n) \; and \; p = (q_1, q_2, \dots, q_n)$$

which represent 2 points in Euclidean n-space. Then, the Euclidean distance (*d*) between point *p* and q is given by:

$$d(p,q) = d(q,p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \qquad (1)$$

At this point, we have all the necessary tools to build the membership function. We now slightly change the above definition for p and q. Let p be a fuzzy set in X and q be a point whose membership function $\mu_p(q)$ is to be determined with respect to *p*. Using Euclidean distance form given in (1)we first find $d(p,q)$ and finally,

$$\mu_p(q) = e^{-d(p,q)^2} \qquad (2)$$

1.1-Notations

The vector definition for fuzzy membership function is important in the context of data sets. Let us consider $X \in \mathbb{R}^d$ be a data set in *d* dimensional feature space. Let us also consider that X has *n* cases: x_i , $i = 1,2, \dots, n$. Then, each row of X, that is x_i, is a *d*-dimensional vector. Each element of x_i is a scalar and is denoted by x_{ij} where $j = 1,2, \dots, d$.

This work strictly focuses on binary classification problems. Hence, there is a binary decision value associated with every case in the data set. Let a vector,

$$y = (y_1, y_2, \dots, y_n)$$

be a decision vector for *n* cases and $y_i \in \{0,1\}$ a decision value for vector x_i where $i = 1,2, \dots, n$.

The mean of X , $i = 1,2, \dots, n$ is denoted by m_X which is also a vector. Each component of m_X is denoted by m_X^j where $j = 1,2, \dots, d$ and is calculated as:

$$m_X^j = \frac{\sum_{i=1}^{n} x_{ij}}{n} \qquad (3)$$

In case of binary classification, the mean of class label 1 and class label 0 are calculated separately and are denoted by a superscript, m_X^1 and m_X^0.

1.2-Supervised Learning

Stuart Russel and Peter norvig in [19] broadly categorizes the field of machine learning into three categories:

1. Supervised learning.
2. Unsupervised learning.
3. Reinforcement learning.

Supervised Learning can further be classified into 2 subcategories:

1. Inductive learning such as Decision Tree.
2. Statistical learning such as Logistic Regression and Naïve Bayes.

Supervised learning is a kind of machine learning technique that models a function from data set that has been labeled or has known decision values [3]. Supervised Learning generally consists of training and testing sets. The training set is used to train the model which is essentially the function that one wants to build. For this purpose, the training data set consists of cases with known decision values that we call labels. Each of these cases is a vector and every vector has a decision value associated with it as a case-decision pair. The training set is given as input to a typical supervised learning algorithm which analyzes the training data and builds a model. This model or function[1] can now be used to map new set of cases for which the decisions or labels[2] are not known. The goal of the entire process is to minimize the error in identifying unlabeled test cases.

1, 2-model, function and labels, decisions have been used interchangeably

Given a supervised learning problem, one can do the following:

Step 1:

The first step for the user is to decide what type of training samples the user wants to use to train a model. An example of this can be the analysis of handwriting. For example, in case of handwriting analysis, the user may include a single handwritten character, an entire handwritten word, or an entire line of handwriting.

Step 2:

After the decision has been made on what kind of training sample user wants to work with, it is time to collect the training samples. Effort should be made in order to make this training sample as close a representation of the actual population. For example, if the user wants to build a model to predict the letter based on handwriting, then he must make sure the training data is from a diverse collection of handwriting. A bad example is a training set with a lot of handwriting from only a handful of people.

Step 3:

In this step the user usually works on effective feature selection. This step is one of the most important steps as not all feature of the data set will contribute equally in building the model. Collecting a bad set of features can seriously degrade the performance and accuracy of the learning algorithm. Some of the very popular feature selection techniques are Information Gain, Expectation Maximization etc.

Step 4:

Once the user has collected a good set of features, it is time to decide on the supervised learning algorithm. Some of the popular choices are Decision Tree, Naïve Bayes, Logistic Regression, Support Vector Machine and Neural Network.

Step 5:

Once step one to four are completed, the training set is introduced to the learning algorithm and is allowed to train. Most supervised learning algorithm consists of control parameters that control the performance of learning algorithm. These parameters can be tweaked with, until a satisfactory training error is achieved. In order to avoid over fitting the model, the user may have an extra step called validation that validates the model's performance and give the user an idea of generalization error.

Step 6:

This is the last step of the entire cycle of supervised learning. Here the trained model is ready to be tested. The user now brings in a set of samples the learning function has never seen before. The learning function is then run on the testing set and different types of error values are measured. Multiple run on a number of test set can be done and an average error from multiple test run can also be reported for a more accurate measure of error.

A number of supervised learning algorithms are available, each with distinctive characteristics and each perform better than other algorithms in specific scenarios. This research explores the performance of Decision Tree, Naïve Bayes and Logistic Regression. A more formal definition of supervised learning technique is given below:

Let us say we have a data set X consisting of n training cases denoted by $(\mathrm{x}_1, \mathrm{y}_1), \ldots, (\mathrm{x}_\mathrm{n}, \mathrm{y}_\mathrm{n})$. x_i is the feature vector of the i^{th} case and y_i is the label (i.e. class) for case x_i. A supervised learning algorithm runs on this training sample and the target is to find a function g, such that,

$$\mathrm{g}: \mathrm{X} \rightarrow \mathrm{Y}$$

where the input space X is mapped onto the output space Y. If we consider the hypothesis space to be G then, $g \in G$. In a slightly different context, g can also be represented as a scoring function,

$$f: X \times \beta \to \mathbb{R}$$

where g is the returned value from the function f that gives the highest score, such that,

$$g(x) = arg\max_{\beta} f(x, \beta)$$

If we consider F to be the spacec of scoring functions then, $f \in F$. An example of such scoring function is the squared loss.

1.2.1-Inductive Learning

Inductive learning is basically learning from examples. According to [19] a more formal definition of inductive learning would be, let us consider that an example is a pair $(x, f(x))$, where x is the input and $f(x)$ is the output of the function applied to x. According to [19] the task of pure inductive inference is this:

Given a collection of examples of f, return a function h that approximates f.

The function h in this case is called a hypothesis. Learning is particularly difficult because it is not easy to tell if h is a good approximation of f. A sign of a good hypothesis is one that generalizes well which in other word means; the hypothesis is able to predict unseen cases. Two of the most common and widespread use of inductive learning technique is Rule Induction and Decision Tree learning. In this section we briefly introduce rule induction.

In rule induction, one generally has an input data set with attributes and decision values specified. According to [20], regularities that are hidden in the data set can be expression in terms of rules. From [20] rules can be defined as:

$$if\,(attribute-1, value-1)\ and\ (attribute-2, value-2)\ and\ \ldots and\ (attribute-n, value-n)\ then\ (decision, value)$$

Some popular rule induction techniques are LEM1, LEM2 and AQ [20]. Decision Tree which is another popular choice of Inductive Learning is discussed in subsequent section.

1.2.2-Statistical Learning

According to [19] statistical learning is a type of learning where learning is viewed as a form of uncertain reasoning from observation. The key concept in statistical learning is data and hypothesis [19]. Here, data is evidence which can be considered as instantiations of some or all of the random variables describing the domain. On the other hand, hypotheses are probabilistic theories of how the domain works, including logical theories as special case.

There are many aspects of statistical learning like Bayesian learning, regression analysis including linear and logistic regression, kernel methods and neural network. All these learning can be summarized under the characteristics of data and hypothesis described above. As an example of statistical learning we can consider Bayesian learning which simply calculates the probability of each hypothesis, given the data, and makes prediction on that basis [19]. On the other hand regression task like Logistic Regression takes a log-likelihood function as hypothesis. Statistical learning problem can be primarily formulated in two different ways:

1. Bayesian Learning.
2. Loss function based learning.

Bayesian learning framework is fundamentally based on Bayes theorem. The structure of Bayesian learning is primarily written as:

$$posterior \propto likelihood \times prior$$

Here, $\propto$ is the sign of proportionality and prior is a prior belief on the characteristics of the data. This belief is often regarded as a subjective belief. Likelihood function is the likelihood of each parameter given the data and finally posterior is the posterior probability of the parameter. A very common approximation- one that is usually adopted in science- is to make prediction based on a single most probable hypothesis- that is, a hypothesis that maximizes the posterior probability. This is often called maximum a posteriori (MAP) hypothesis.

On the other hand a non-Bayesian formulation is learning based on a loss function. A distribution function is used to build the hypothesis and then a loss function is formed. The most common use of the loss function is the square loss due to its convex nature. The problem then becomes an optimization problem where the loss function is minimized using various constrained or unconstrained linear and non-linear optimization techniques.

1.3-Binary Classification

Binary classification is a form of supervised learning where the label y_i (notation introduced in section 1.1) can only take two possible values 0 and 1. From [23] a binary classification task can be defined as;

Given:

1. An input space X
2. An unknown distribution D over $X \times \{0,1\}$

Compute: A function f minimizing $\mathbb{E}_{(x,y)\sim D}[f(x) \neq y]$

Here, the two classes are {0, 1} in the data set X. $\mathbb{E}_{(x,y)\sim D}$ is an error function that calculates the error between original class label y and predicted labels using function $f(x)$. The goal is to choose a function $f(x)$ such that the error $\mathbb{E}_{(x,y)\sim D}$ is minimized. Binary classification algorithms such as Decision Tree, logistic Regression and Naïve Bayes have been there for a while and been used in many fields. Despite their widespread use and numerous publications, following sections very briefly introduces these three algorithms.

1.3.1-Decision Tree

Learning with Decision Tree involves a tree like predictive model with branches that eventually conclude or infer a decision on the target value. Predictive analytics using Decision Tree has been a popular choice for learning and has been widely used as a data mining tool. The input to the model is a set of variables that are the feature vectors from the data set. The output is a Decision Tree model. The goal is to predict the target variable based on the input information.

Every Decision Tree consists of nodes that represent input variables. In regard to a data set, one can consider these input variables as the features. The structure of the tree basically starts with root node and branches out as more variables are added for classification purpose. Generally there are 2 types of Decision Trees:

1. Classification Tree
2. Regression Tree

The use of one of these trees is purely dependent on the type of problem one is looking at. If the decision value y is categorical, we use classification tree to make decisions. In case of real valued y, a regression tree is more appropriate.

Let us consider the following data set:

Outlook	Humidity	Wind	Decision
sunny	high	strong	no
sunny	high	weak	no
sunny	normal	strong	yes
sunny	normal	weak	yes
overcast	high	weak	no
overcast	high	strong	yes
rain	high	weak	yes

This is a data set where the decision is to whether to play tennis or not based on the weather condition. Figure 1.3.1 represents a Decision Tree model based on the sample data set:

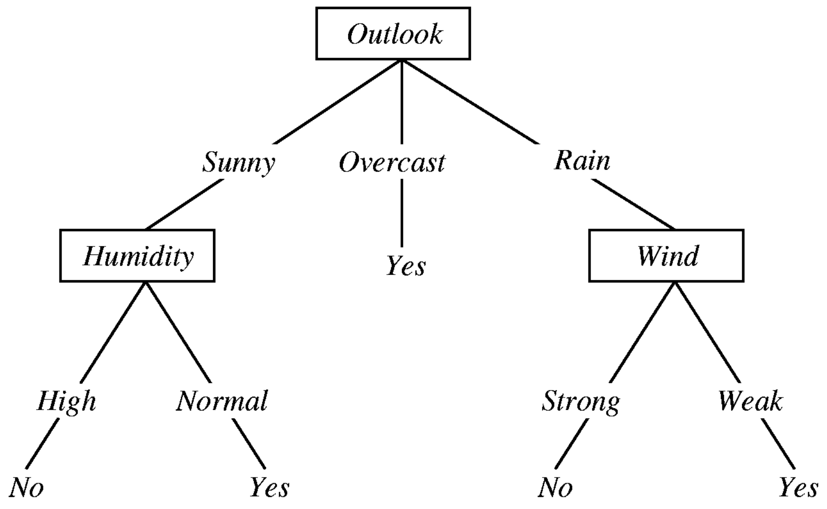

Fig 1.3.1: classification tree

Similarly, a regression tree may look like this:

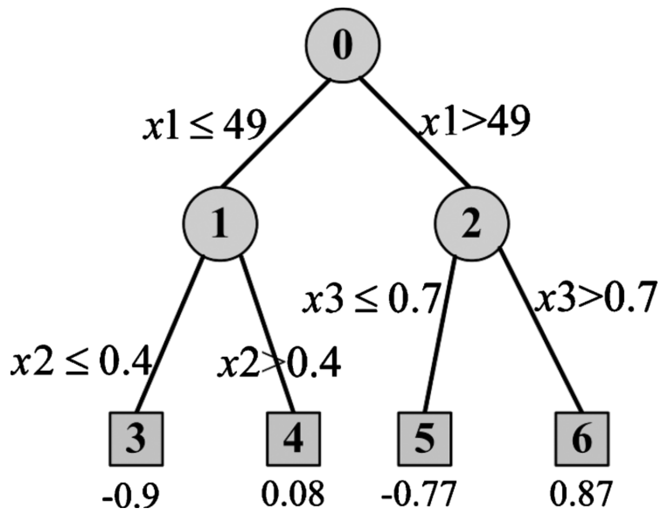

Fig 1.3.2: Regression Tree

A tree can be trained in many different ways. Some of the well established Decision Tree implementations are ID3, C4.5 etc.

1.3.2-Logistic Regression

Logistic Regression is a statistical classification model that is used for categorical classification. A logistic function plays the key role to describe the possible outcomes. The prediction is generally in terms of probability and this probability is then converted to a categorical decision value. A multinomial Logistic Regression model also allows you to classify multiple decisions and not restricted to binary classifications only. A more formal definition for Logistic Regression is given below:

Let us consider the binary classification model were the decision for ith feature vector y_i can only take two values; 0 or 1. From section 1.1 let X be a $n \times p$ matrix where x_i denotes a $p \times 1$ vector. Let β be the set of parameters and also a $p \times 1$ vector, and $y = [y_1, y_2, \dots, y_n]^T$ the set of decision variables. Then, we can describe the probability of success or failure (1 or 0) as a logistic function of the following form:

$$p(y_i = 1|x_i, \beta) = \frac{e^{\beta^T x_i}}{1 + e^{\beta^T x_i}}$$

And

$$p(y_i = 0|x_i, \beta) = 1 - p(y_i = 1|x_i, \beta) = \frac{1}{1 + e^{\beta^T x_i}}$$

From here, we can find the log-likelihood function and following which, we formulate an optimization problem where our goal is to minimize the log-likelihood function. Let us consider p a $n \times 1$ column vector, where

$$p = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix} = \begin{bmatrix} \frac{e^{\beta^T x_1}}{1 + e^{\beta^T x_1}} \\ \frac{e^{\beta^T x_2}}{1 + e^{\beta^T x_2}} \\ \vdots \\ \frac{e^{\beta^T x_n}}{1 + e^{\beta^T x_n}} \end{bmatrix}$$

And, W a $n \times n$ diagonal matrix, where

$$W = \begin{bmatrix} p_1(1-p_1) & 0 & \cdots & 0 \\ 0 & p_2(1-p_2) & & \vdots \\ \vdots & \vdots & & \vdots \\ 0 & \cdots & & p_n(1-p_n) \end{bmatrix}$$

These are results from solving the optimization problem that involves a second order partial derivative we talked about above. The value for β is found and updated in an iterative process using the following form:

$$\beta^{new} = arg\min_{\beta}(z - X\beta)^T W(z - X\beta)$$

where,

$$z = X\beta^{old} + W^{-1}(y - p)$$

The W associated with the regression function is sort of a weight and for this reason, each step of updating β in Logistic regression is often referred to as iterative weighted linear regression. We can use these β values on a testing set to find the probability associated with each x_i in the testing set. We can then assign decision value 0 or 1 depending on the probability of success.

1.3.3-Naïve Bayes

The Naïve Bayes classification is a classification model that is based on the Bayesian probability model and assumes complete independence of the features. From [18] let us consider an event can only occur only if one of the set of exhaustive and incompatible events $B_1, B_2, \dots, B_n$ occurs. The probabilities of these events

$$P(B_1), P(B_2), \dots, p(B_n)$$

corresponding to the total absence of any knowledge as to the occurrence or nonoccurrence of A, are known. We also know the conditional probabilities:

$$P(A|B_i); i = 1,2, \dots, n$$

for A to occur, assuming the occurrence of B_i. The question now we shall try and answer is, how does the probability of B_i change with additional information that A has actually happened. The answer to this question really amounts to finding the conditional probability of $P(B_i|A)$. In order to do this, we first set up few preliminaries. We know that the probability of compound event $P(AB_i)$ can be presented in two forms:

$$P(AB_i) = P(B_i)P(A, B_i)$$

$$P(AB_i) = P(A)P(B_i, A)$$

Equating the right-hand members, we derive the following expression for the unknown probability $P(B_i, A)$:

$$P(B_i, A) = \frac{P(B_i)P(A, B_i)}{P(A)}$$

Since the event A can materialize in the mutually exclusive forms,

$$P(AB_1), P(AB_2), \dots, P(AB_n)$$

By applying theorem of total probability, we get

$$P(A) = P(B_1)P(A|B_1) + P(B_2)P(A|B_2) + \cdots + P(B_n)P(A|B_n)$$

This probability is also called the marginal probability.

Hence the final expression for $P(B_i, A)$ is,

$$P(B_i, A) = \frac{P(B_i)P(A|B_i)}{P(B_1)P(A|B_1) + P(B_2)P(A|B_2) + \cdots + P(B_n)P(A|B_n)}$$

This formula is known as the Bayes Theorem.

The Naïve Bayes classifier is based upon this theorem where we call $P(B_i, A)$ the posterior probability. $P(B_i)$ is the prior probability of B_i and $P(A|B_i)$ is the likelihood of A given B_i. Considering the marginal probability $P(A)$ as a normalizing term only, we get:

$$posterior \propto likelihood \times prior$$

where again, $\propto$ is the sign of proportionality.

Below we look at a simple example of Naïve Bayes classifier (a modified version of a popular example from wikipedia):

Table 1.5.1 is a small data set that has height, weight and foot size information for different male and female. Our goal is, given information about a new person, the Naïve Bayes model should be able to identify whether the person is male (m) or female (f).

Table 1.5.1: Example Data set for Naïve Bayes

Sex	Height (feet)	Weight (lbs)	Foot size (inches)
m	6	185	11
m	5.9	195	12
m	5.5	175	12
m	5.8	160	10
f	5.1	105	5
f	5.4	155	7
f	5.3	135	6
f	5.7	150	8

We also assume that the classifier is created from the training set using a Gaussian distribution. To find the Gaussian distribution, we need to calculate the samples mean and variance.

Sex	Mean (height)	Variance (height)	Mean (weight)	Variance (weight)	Mean (foot size)	Variance (foot size)
m	5.8	0.047	178.75	222.92	11.25	0.917
f	5.37	0.062	136.25	506.25	6.5	1.7

The prior probability is generally based on our knowledge of the data. In this case let us consider the probability of the 2 classes male and female is equal and so $prior(m) = prior(f) = 0.5$. The prior probability can be based on factors like the frequency of male of female in the data set or it can also be our prior knowledge of a much larger population. Up until this stage, we have the training model ready for Naïve Bayes classifier.

Now we shall perform testing. Let us consider the testing case below:

Sex	Height (feet)	Weight (lbs)	Foot size (inches)
unknown	6	135	7

The final goal is to find the posterior probability for both male and female classes given the prior and the likelihood. The posterior for both male and female can be written as below:

$$posterior(m) = \frac{prior(m) \times likelihood(m)}{marginal\ probability}$$

And,

$$posterior(f) = \frac{prior(f) \times likelihood(f)}{marginal\ probability}$$

Now the prior is already known to us. The marginal probability is a normalizing constant and can be disregarded. The last piece is the likelihood function. Now,

$$likelihood(m) = p(height|m) \times p(weight|m) \times p(footsize|m)$$

And,

$$likelihood(f) = p(height|f) \times p(weight|f) \times p(footsize|f)$$

We find these values using the Gaussian distribution function as follows,

$$p(height|m) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(\frac{(6-\mu)^2}{2\sigma^2})} \approx 9.92e-04$$

$$p(weight|m) \approx 0.0018$$

$$p(footsize|m) \approx 9.42e-06$$

$$p(height|f) \approx 2.44e-022$$

$$p(weight|f) \approx 7.88e-004$$

$$p(footsize|f) \approx 0.2247$$

This finally results in,

$$likelihood(m) = 1.6820e - 011$$

And,

$$likelihood(f) = 4.3204e - 026$$

And hence,

$$posterior(m) \propto 8.4100e - 012$$

And,

$$posterior(f) \propto 2.1602e - 026$$

Since, the posterior for male is greater, the given test case is predicted as a male.

1.4-Standard Error Measures for Binary Classification

All classification algorithms have error associated with their performance. There are many different ways to look at the error measure depending on what kind of answers we are looking for. In binary classification, apart from mean absolute classification error, we also care about class asymmetry or in-class error. This paper talks about 4 different types of error:

1. Mean absolute error (MAE)
2. Precision
3. Recall
4. F-measure

In case of binary class prediction, if we consider the class label to be either 0 or 1, mean absolute error is calculated as follows:

$$MAE = \frac{\sum_{i=1}^{n} |\hat{y}_i - y_i|}{n}$$

where, $\hat{y}_i$ is the predicted outcome and y_i is the original class label. Both $\hat{y}_i \; and \; y_i \in \{0,1\}$. Mean absolute error in terms of nominal class labels can be written as:

$$MAE = 1 - \frac{\sum_{i=1}^{n}(\widehat{y_i} == y_i)}{n}$$

where, the logical operator "==" checks for equality of the 2 predicted vs. the original nominal class labels. If they are equal, 1 is returned otherwise 0.

Mean absolute error is not sufficient as an error measure for all binary classification problems. Binary classification problems often has class asymmetry which in other word means, number of cases for both classes are not equal. In this kind of problem, we also want to be able to measure the in-class error or how accurate the prediction is for each class. In order to deal with class asymmetry, we also talk about precision, recall and f-measure.

Before defining precision, recall and f-measure we would like to introduce the following terms associated with binary classification problems. Let us consider that the 2 class labels are positive (p) and negative (n). Then,

1. True Positive (TP): These are cases that are actually positive and have been identified as positive.
2. False Negative (FN): These are cases that are actually negative but have been identified as positive.
3. True Negative (TN): These are cases that are actually negative and have been identified as negative.
4. False Positive (FP): These are cases that are actually negative but have been identified as positive.

For positive cases, using the above definitions we can now define,

$$precision_p = \frac{TP}{(TP + FP)}$$

$$recall_p = \frac{TP}{(TP + FN)}$$

$$f_p^{score} = 2\frac{precision_p \times recall_p}{(precision_p + recall_p)}$$

Similarly for negative cases,

$$precision_n = \frac{TN}{(TN + FN)}$$

$$recall_n = \frac{TN}{(TN + FP)}$$

$$f_n^{score} = 2\frac{precision_n \times recall_n}{(precision_n + recall_n)}$$

Precision implies the accuracy of a predicted class label over all the predicted cases. So, if the model has identified for example 100 cases as positive and 70 out of those are truly positive (we know this because the training labels are known to us), then the precision is 70%. Precision gives us an idea of how accurate the model is in identifying a particular class label. In other word, out of all the cases that have been identified as let say positive, we are looking for the number of cases that are truly positive.

Recall on the other hand implies the relevance of the predicted class label over the population of the class label. This means. If for example, the model has identified 80 out of 100 total truly positives, then the recall is 80%. Recall gives us an idea of how relevant the precision is. In other word, whether the model has been able to identify a significant portion from the total population.

Both precision and recall are important and the goal is to get both a high precision as well as a high recall. This in practice is not always achievable and often times, a more realistic approach is to balance the precision and recall. F-measure or F-score which is a harmonic mean of the two quantities is sort of a balance between the two. F-measure gives equal weight to both precision and recall. Getting a good F-measure would mean that we are trying to build a model that balances the precision and recall rather than giving precision or recall different weights.

2-Literature Review

The related work to this research started back in December 2012 while working on a problem called Drug Discovery using High Throughput Screening (HTS). The authors of [8] started working on HTS in as early as in 2006, using a well known Machine Learning algorithm, Support Vector Machine. The goal was to identify compounds that can bind well with certain types of protein (called active compounds) and the same compounds then eventually go on to become potential anti-cancer drug. The data sets consisted of both active and non-active compounds and a sample from the data set was used to train the learning algorithm (SVM in this case). Once it was trained, the goal was to identify as many active compounds as it could, from a test set. In 2009, Dr. Swapan Chakrabarti et al. in their work [4] used Neural Network which performed superior to the work done in 2006. Despite an improvement, the performance of the predictor was still a major issue. In 2012, while researching on the same problem, the issue of high misclassification rate responsible for degrading the performance was addressed. While misclassification is an issue with all binary classifiers, in case of class asymmetry, the effect of this problem amplifies. This degrades the performance of the classifier and it is important to be able to inform the classifier about the misclassifications.

Several works on learning with misclassification have been done in various areas of Machine Learning and Data Mining. In [9] Michael Pazzani and et al. in their work explored algorithms for learning classification procedures that attempt to minimize the cost of misclassification examples. Their method called Reduced Cost Ordering algorithm creates a decision list (i.e., an ordered set of rules) that describe and compare a variety of inductive learning approaches. Their work was restricted to Decision Tree and no other binary classifiers were discussed. In [10] Shai Ben-David and et al. used surrogate loss functions to minimize the misclassification error rate for binary classification. They did not test their method on any data set and it was more of a theoretical set up for the surrogate loss function.

One of the major works in regards to improving classifier performance has been done using AdaBoost [11]. Boosting is a general method for improving the accuracy of any given learning algorithm. Boosting takes into account the idea that multiple weak learner's performance can be combined to get a strong learner. So the output of multiple weak learning algorithms is combined into a weighted sum. This weighted sum represents the final output of the classifier that has been boosted. AdaBoost is also adaptive. In its subsequent run, the weak learners in AdaBoost can be tweaked in favor of samples that were previously misclassified by the classifiers. One of the major drawbacks of AdaBoost is that it is very sensitive to outliers and data sets with inherent noise. If the numbers of outliers are large, the performance of AdaBoost can degrade significantly. The work in this paper particularly addresses these shortcomings of AdaBoost and subsequently shows that they can be overcome using the method discussed in this paper.

The idea behind integrating fuzzy logic with Neural Network to solve the problem of Drug Discovery using High Throughput Screening first surfaced in December 2012. At that time the classifier already being used to solve the given problem was Neural Network. Some work on fuzzy clustering technique can be found in [5] and [6]. These works however do not address classification problems. In [7], the author work on fuzzy labeling and apply his model to Decision Tree, fuzzy Bayesian estimation and linguistic FOIL algorithms. The drawback of this paper is that the method does not generalize well as the author converts the numeric quantities to linguistic expression. Although this aims towards better transparency, it does not scale well since not all Machine Learning algorithms can work with linguistic labeling.

In [12] James and et al. discuss the performance of K-Nearest Neighbor using fuzzy labeling. According to them, one of the difficulties that arise when utilizing K-Nearest Neighbor is that each of the labeled cases is given equal importance in deciding the class membership of the pattern to be classified, regardless of their "typicalness". They have shown that with fuzzy membership to the neighbors, the algorithm perform superior to the crisp counterpart. Their implementation however is limited to K-Nearest Neighbor and they use the inverse distance as the membership function. Their method does in a way identify the misclassifications but the misclassifications do not take part in learning.

In [13] Mansoor and et al. discuss a problem of pattern classification where misclassification costs from one class to the other class are not the same. In order to address this problem, they propose a method of designing a classification system based on fuzzy rule. In their work, in order to tune the rule-base, they use the rule-weight mechanism. In the proposed method they assume that the misclassification costs from one class to the other class are known and instead of minimizing the error rate they attempt to minimize the total cost of the classifier on the training data.

In [14] Massih-Reza and et al. discuss a special learning case where only a small set of labeled data is available together with a large set of unlabeled data. In their approach they make use of both unlabeled data and of a probabilistic misclassification model for those data. This is another approach of learning with misclassification but deals with a very specific case of semi-supervised learning where only a small set of labeled data is available. In their approach they have used a variant form of the classification Expectation Maximization algorithm.

A number of fuzzy rule based classification is also available in literature. In [16] Sushmita and Sankar discuss a self-organizing artificial neural network based on Kohenen's model of self-organization, which is capable of handling fuzzy input and providing fuzzy classification. In [15] Hans Roubos and et al. discuss an automatic design of rule-based classification systems based on labeled data. In [17] Margarita in her PhD thesis proposed a fuzzy semantic labeling method that uses confidence measures based on the orthogonal distance of an image block's feature vector to the hyper-plane constructed by a Support Vector Machine. In contrast to the proposed work, none of these papers talk about learning with fuzzy labels.

Multiclass problem is a domain that is also relevant to this work. In [21] the authors introduce techniques to solve multi-class problems where they reduce the problem into multiple binary classification problems. The authors then solve the problems using a "margin based binary learning algorithm". Out of various techniques to solve multi-class problems, one popular choice is to decompose multi-class problems into multiple binary problems. The two most common approach under this method is one-versus-all (OVA) and all-versus-all (AVA). According to [22], OVA and AVA are so simple that many people invented them independently. No one person can thus be attributed to the invention of the technique. Although a lot of work in OVA and AVA has been done thus far, no works have been done on taking a binary classification problem and solve it as a multi-class problem. In this research, one can consider multiple clusters as multiple classes. These classes are generated based on the misclassification and the learner is then trained with these misclassifications. One of the most important outcome of the discussion on decomposition of multi-class problem to multiple binary classification problems is that, now, one can solve each of these binary classification problems using the proposed FLIC method. This in turn allows one to scale to multiclass classification without worrying about any change in the proposed algorithm making it scalable to most existing methods.

3-Methodology

This section provides a more formal definition of the method called fuzzy logic and Interval Clustering (FLIC). The discussion in this section is divided into 2 sub-sections; the first section introduces the conversion of binary labels to fuzzy labels. The second section formalizes the idea and introduces Interval Clustering using the notations in section 1.1.

3.1-Introduction to Fuzzy Labeling

All binary classification models have outliers or misclassification that degrades the models' performance. FLIC helps learning algorithms to identify misclassification while learning, by adding fuzziness to the training labels. A supervised learning algorithm is always limited by the training samples it has. Hence, binary classification models strictly learn from the training samples and do not generalize very well. In order to improve the generalization error FLIC incorporates fuzzy labeling to the training set. In situations where one has a significant amount of outliers, one can refer to the data set as highly overlapping in feature space.

In order to make the learner more robust to the high overlapping data samples, this paper introduces fuzzy labeling for the training data set. In case of binary classification problems, dimension of the training data label is two. This work increases the label dimension by calculating fuzzy membership value for each case in the training set.

In order to formalize the proposed method, this section introduces the idea of fuzzy labeling as a replacement to binary labels for training data. For the purpose of clarity, this paper only considers the 1 and 0 binary labels. In practice they can be any nominal value such as "True" or "False", "yes" or "no" and so on. One can convert these values to 0 and 1 to find the fuzzy membership values for each training case. The notation for binary label has already been introduced in section 1.1.

Fuzzy label on the other hand is a membership function for each case in the data set. As discussed earlier, for a fuzzy set p in X and a point q, membership function of q with respect to p is given by $\mu_p(q)$. This section converts all the binary labels to fuzzy labels using the following algorithm:

Algorithm 1: Convert binary labels to fuzz labels

```
Input: Training data $X \in \mathbb{R}^d$ and class label $y$
Output: Fuzzy labels and Euclidean distance for both class 1 and 0 respectively
1. Divide training set as $X^1$ and $X^0$for label 1 and label 0 respectively
2. Find mean of both $X^1$ as $m_X^1$ and mean of $X^0$ as $m_X^0$
3. Find distances $d(m_X^1, x_i^1)$, $d(m_X^0, x_i^1)$, $d(m_X^1, x_i^0)$ and $d(m_X^0, x_i^0)$
4. Calculate membership values as:
   $\mu_m^1(x_i^1) \leftarrow e^{-d(m_X^1, x_i^1)}$, $\mu_m^1(x_i^0) \leftarrow e^{-d(m_X^0, x_i^1)}$, $\mu_m^0(x_i^1) \leftarrow e^{-d(m_X^1, x_i^0)}$ and
   $\mu_m^0(x_i^0) \leftarrow e^{-d(m_X^0, x_i^0)}$
5. Return $\mu_m^1(x_i^1), \mu_m^1(x_i^0), d(m_X^1, x_i^1), d(m_X^0, x_i^1), \mu_m^0(x_i^0), \mu_m^0(x_i^1), d(m_X^1, x_i^0)$ and $d(m_X^0, x_i^0)$
```

where,

$d(m_X^1, x_i^1)$ and $d(m_X^0, x_i^1)$ is the distance of class 1 samples to class 1 and class 0 respectively

$d(m_X^1, x_i^0)$ and $d(m_X^0, x_i^0)$ is the distance of class 0 samples to class 1 and class 0 respectively

$\mu_m^1(x_i^1)$ and $\mu_m^1(x_i^0)$ is the membership function of class 1 samples to class 1 and class 0 respectively

$\mu_m^0(x_i^1)$ and $\mu_m^0(x_i^0)$ is the membership function of class 0 samples to class 1 and class 0 respectively

3.2-Fuzzy Labeling with Hard Interval Clustering

Binary classification has two classes, 0 and 1 which means; each case either belongs to class 0 or class 1. For the purpose of discussion, this can be considered as two separate clusters. In terms of fuzzy boundary, one can also consider this to be fuzzification to level 0 or crisp boundary or no fuzzification. The cases belong to either of the class. This turns out to be a problem for data set where there are outliers. In case of outliers, a model cannot learn efficiently and we have the problem of misclassification.

The proposed method tries to identify the outliers before the model is trained. Following from section 3.1, the output from the fuzzy labeling algorithm is a fuzzy label for every training case in the data set. Hence for every x_i we have $\mu_m^y(x_i^1) \in [0,1]$ and $\mu_m^y(x_i^0) \in [0,1]$ where, $y \in \{0,1\}$. In other word, every case now belongs to both class 0 and class 1 but with a membership function. If membership function with class 0 is more than class 1, then the case belongs to class 0, otherwise class 1.

At this point we have n-clusters. The reason is; there are n training cases and each training case is unique due to unique membership function to both class 0 and class 1. Let us consider this to be fuzzification to level n. At this point, one can train the system with these membership functions. The problem is that, this scenario turns out to be over constrained for the system and the system cannot generalize for such diverse variation. Thus, this work proposes a novel approach called fuzzy membership based hard interval clusters which combine fuzzification with hard clustering to optimize the number of clusters. This in turn improves the performance of the algorithm.

In an ideal scenario, the membership function of class 1 cases with class 1 must be more than the membership function of class 1 cases with class 0. In other word, for a case in class 1:

$$\mu_m^1(x_i^1) > \mu_m^1(x_i^0)$$

On the other hand, for cases in class 0:

$$\mu_m^0(x_i^0) > \mu_m^0(x_i^1)$$

However in practice, for some cases from both the classes, this relationship is reversed. In this paper, such classes are called the weak class 0 and the weak class 1. These are considered to be the outliers in Euclidean d-space (since $X \in \mathbb{R}^d$).

The first task is to identify these outliers by taking the difference of their membership functions. We do:

$$D_1 = \mu_m^1(x_i^1) - \mu_m^1(x_i^0) \qquad (4)$$

$$D_0 = \mu_m^0(x_i^0) - \mu_m^0(x_i^1) \qquad (5)$$

where D_1 and D_0 are the differences in membership function for class 1 and class 0 respectively. In case of outliers, D_1 and D_0 have negative values. If we consider the outliers to be in a separate class of their own, we have four different clusters instead of two,

1. D_1>0 (strong class 1)
2. D_1<0 (weak class 1)
3. D_0>0 (strong class 0)
4. D_0<0 (weak class 0)

In order for the model to recognize weak class 1 and weak class 0, only the membership function value for the weak classes are changed by introducing a pseudo distance. The goal is to take these outliers and place them far from their opposite class as well as from their strong class counterpart. This should allow them to have a cluster of their own. In this work it is called: the rule of pseudo distance. For the purpose of discussion, let us consider a simple linear boundary for a binary classification model as following:

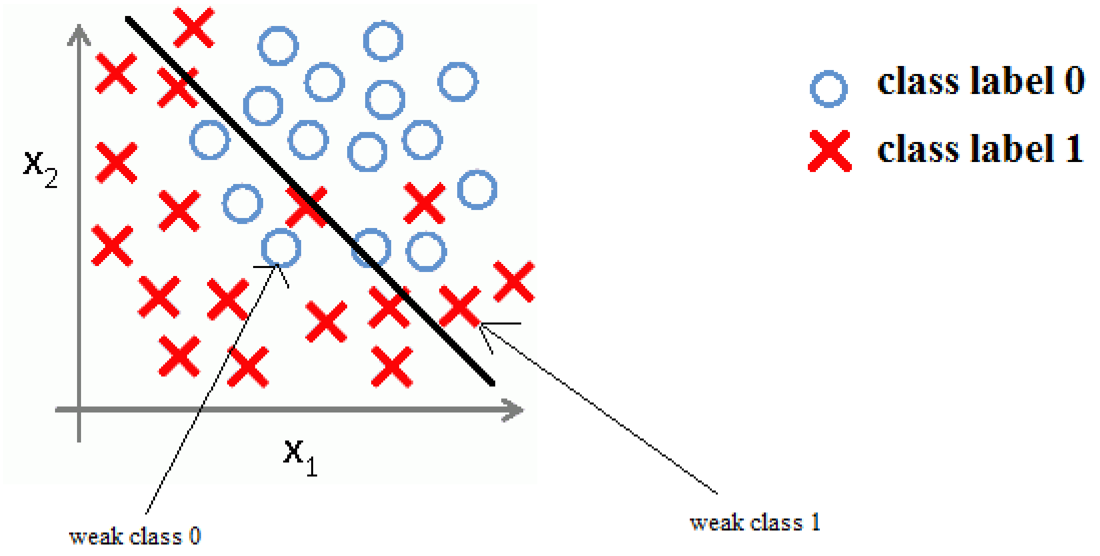

Fig 3.2.1: Weak class 0 and weak class 1 in Binary Classification.

The weak classes in Fig 1 are being shown by the arrow. This is a model for Euclidean 2-space. The 2 features are X_1 and X_2. Using the pseudo distance model, the outliers are assigned their own clusters. At this point pseudo distance model can be formally defined. Let us call this pseudo distance a constant c. We add this constant c to the current distance of the outliers from Algorithm 1. This new constant distance allows the outliers to have their own cluster and set them apart from the rest.

Before taking the discussion any further, let us look at different scenarios for the weak class 1 (we only consider class 1, the same situation applies to class 0 as well):

Scenario 1:

$$\mu_m^1(x_i^1) < \mu_m^1(x_i^0) \mathit{and}\ \mu_m^1(x_i^1) < 0.25$$

In this scenario the weak class 1 cases have very weak membership value with class 1 and relatively stronger membership to class 0. This is a situation where the outliers are very weak and the membership value of a weak class 1 to class 1 is very low.

Scenario 2:

$$\mu_m^1(x_i^1) < \mu_m^1(x_i^0) \mathit{and}\ 0.25 < \mu_m^1(x_i^1) < 0.5$$

In this scenario the weak class 1 cases have weak membership value with class 1 and relatively stronger membership to class 0. This is a slightly better scenario than scenario one but still, very weak outliers with membership value upper bounded by 0.5.

Scenario 3:

$$\mu_m^1(x_i^1) < \mu_m^1(x_i^0) \mathit{and}\ 0.5 < \mu_m^1(x_i^1) < 0.75$$

In this scenario the weak class 1 cases have moderately strong membership value with class 1 but even stronger membership to class 0. These are weak outliers doing better than scenario one and 2 as they have membership value more than 0.5.

Scenario 4:

$$\mu_m^1(x_i^1) < \mu_m^1(x_i^0) \mathit{and}\ \mu_m^1(x_i^1) > 0.75$$

In this scenario the weak class 1 cases have very strong membership value with class 1 but even stronger membership to class 0. These are not so weak outliers and have very high membership value with its own class.

Note: The same scenario applies to class 0 with membership values reversed.

These scenarios are important because based on these scenarios, one can decide on the number of clusters they want. For the simplest of case, where we do not care about the scenarios but only care about the following situation,

$$\mu_m^1(x_i^1) > \mu_m^1(x_i^0)$$

And,

$$\mu_m^0(x_i^0) > \mu_m^0(x_i^1)$$

We call it the 4-cluster situation. Below we formally define the 4-cluster situation and a method that finds the membership values in this situation:

3.2.1-Cluster 4

This is the simplest case where a constant distance c is added to all the weak cases from both class 0 and class 1.

$$f(x_i^1) = \begin{cases} d(m_X^1, x_i^1) + c, if D_1 < 0 \\ d(m_X^0, x_i^1) \quad , \ otherwise \end{cases} \tag{6}$$

And,

$$f(x_i^0) = \begin{cases} d(m_X^0, x_i^0) + c, if D_0 < 0 \\ d(m_X^1, x_i^0) \quad , \ otherwise \end{cases} \tag{7}$$

where, $f(x_i^1)$ and $f(x_i^0)$ are the new distances for the weak class 1 and weak class 0 respectively. This gives rise to 4 different clusters as we have discussed in section 3.2. As we can see, this only increases the distance of the weak classes with respect to its own class. The distance between the weak class and its opposite class remain unchanged. For example, if we have a case that is actually in class 1 but has higher membership with class 0, pseudo distance only add a constant term to $d(m_X^1, x_i^1)$ where as $d(m_X^0, x_i^1)$ remain unchanged. In case of nominal fuzzy labeling, let us consider the 4 clusters to be strong class 1, weak class 1, strong class 0 and weak class 0.

3.2.2-Cluster 6

In cluster 4, the fact that some of the weak cases despite having a high membership value (> 0.5) to its own class may be an outlier, is missed. They are different from the weak cases that have low membership value to its own class in the sense that, the latter can be considered as very weak. This allows one to go to a higher dimension of fuzzification. This is captured in cluster 6 where we do the following:

$$f(x_i^1) = \begin{cases} d(m_X^1, x_i^1) + c_1, if D_1 < 0 and\ \mu_m^1(x_i^1) \geq 0.5 \\ d(m_X^1, x_i^1) + c_2, if D_1 < 0 and\ \mu_m^1(x_i^1) < 0.5 \\ d(m_X^0, x_i^1) \quad , \quad otherwise \end{cases} \tag{8}$$

And,

$$f(x_i^0) = \begin{cases} d(m_X^0, x_i^0) + c_1, if D_1 < 0 and\ \mu_m^0(x_i^0) \geq 0.5 \\ d(m_X^0, x_i^0) + c_2, if D_1 < 0 and\ \mu_m^0(x_i^0) < 0.5 \\ d(m_X^1, x_i^0) \quad , \quad otherwise \end{cases} \tag{9}$$

where c_1 and c_2 are 2 constant terms for 2 different cases. In the first case, if the membership value of the weak class with its own class exceeds more than 0.5, we add c_1 else, we add c_2. The nominal fuzzy labeling for cluster 6 can be considered as: strong, weak and very weak for both class 1 and class 0.

3.2.3-Cluster 10

In section 3.2 we discussed the 4 scenarios where we see how $\mu_m^1(x_i^1)$ and $\mu_m^0(x_i^0)$ can fall under one of the four intervals. In cluster 10 we take under consideration all the scenarios. Each of these intervals has its own cluster. These clusters allow one to distinguish between strong and weak outliers thus allowing one to move to an even higher dimension. Hence, cluster 10 takes the following form:

$$f(x_i^1) = \begin{cases} d(m_X^1, x_i^1) + c_1, & if D_1 < 0 \; and \; \mu_m^1(x_i^1) \geq 0.75 \\ d(m_X^1, x_i^1) + c_2, & if \; D_1 < 0 \; and \; 0.5 < \mu_m^1(x_i^1) < 0.75 \\ d(m_X^1, x_i^1) + c_3, & if \; D_1 < 0 \; and \; 0.25 < \mu_m^1(x_i^1) < 0.5 \\ d(m_X^1, x_i^1) + c_4, & if D_1 < 0 and \; \mu_m^1(x_i^1) < 0.25 \\ d(m_X^0, x_i^1), & otherwise \end{cases} \tag{10}$$

And,

$$f(x_i^0) = \begin{cases} d(m_X^0, x_i^0) + c_1, & if D_0 < 0 \; and \; \mu_m^0(x_i^0) \geq 0.75 \\ d(m_X^0, x_i^0) + c_2, & if \; D_0 < 0 \; and \; 0.5 < \mu_m^0(x_i^0) < 0.75 \\ d(m_X^0, x_i^0) + c_3, & if \; D_0 < 0 \; and \; 0.25 < \mu_m^0(x_i^0) < 0.5 \\ d(m_X^0, x_i^0) + c_4, & if D_0 < 0 and \; \mu_m^0(x_i^0) < 0.25 \\ d(m_X^1, x_i^0), & otherwise \end{cases} \tag{11}$$

where c_1, c_2, c_3 and c_4 are 4 constant terms for 4 different cases. The nominal fuzzy labeling for cluster 6 can be considered as: strong, not strong, moderately weak, weak and very weak for both class 1 and class 0.

The algorithm for forming the Interval clusters is given below:

Algorithm 2: Clustering fuzzy labels to recognize outliers

```
Input: Training data $X \in \mathbb{R}^d$ and class label $y$ and cluster size $s$
Output: Fuzzy clustered labels $f(x_i^1)$ and $f(x_i^0)$
1. Call Algorithm 1 with $X$ and $y$ to get,
   $\mu_m^1(x_i^1), \mu_m^1(x_i^0), d(m_X^1, x_i^1), d(m_X^0, x_i^1), \mu_m^0(x_i^0), \mu_m^0(x_i^1), d(m_X^1, x_i^0)$ and $d(m_X^0, x_i^0)$
2. If $s = 4$ find $f(x_i^1)$ and $f(x_i^0)$ using equation 6 and 7
   Else if $s = 6$ find $f(x_i^1)$ and $f(x_i^0)$ using equation 8 and 9
   Else if $s = 10$ find $f(x_i^1)$ and $f(x_i^0)$ using equation 10 and 11
3. Return $f(x_i^1)$ and $f(x_i^0)$
```

In Fig. 3.2.1 we have identified the weak classes. The following figure illustrates the effect of pseudo distance on the weak classes or the outliers (for ease of understanding we apply linear separator but the same applies for non-linear separator).

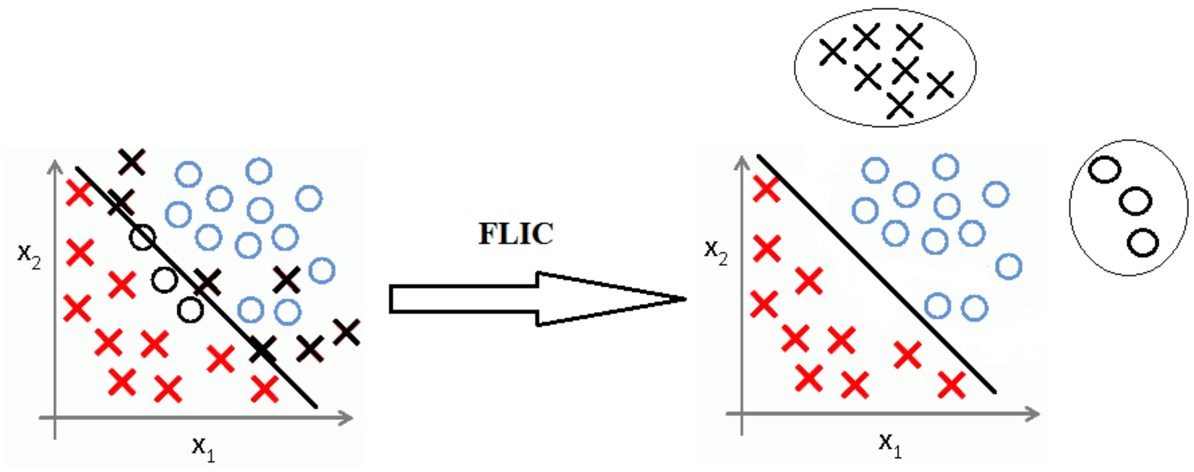

Fig 3.2.2: Outliers after applying FLIC

4-Results

Experiments were conducted on four data sets. Three of the four data sets- Internet Advertisement, Adult and Mushroom are from UCI Machine Learning repository. The last data Telecom Churn is from BigML. All the results are averages of 20 experiments. The graphs 4.1.1, 4.2.1, 4.3.1 and 4.4.1 in this chapter show both training and testing error for the four data sets. The error measures used in this work are the mean absolute error (MAE), precision, recall and F-measure (discussed independently in section 1.4). A detail analysis on the performance is given under the discussion section. The green line represents the general machine learning algorithms without FLIC whereas the red line is machine learning algorithm using FLIC. Some of the acronyms used are described below:

DT = Decision Tree

LR = Logistic Regression

NB = Naïve Bayes

FDT/FLR/FNB = Fuzzy (DT/LR/NB)

Along with the MAE graphs, this work also provides graphs (4.1.i to 4.4.i and i = 2 to 4) and tables (Appendix A) for precision, recall and F-measure for outputs from all the three methods- Decision Tree, Logistic Regression and Naïve Bayes. The error measures like precision, recall and F-measure are provided only from testing (no training precision, recall and F-measure have been provided). In order to read the statistics from the tables and the graphs, below are few definitions,

Sample size: The results are for varying number of training cases in the order of low to high. So sample size 1 means lowest number of training cases and corresponds to the left side of the graph and 10 means highest number of training cases and corresponds to the right side of the graph. In the tables in Appendix A, the sample size represents the same as described above. Each experiment involved different sample size and again for the table also, they are in order low to high.

Binary labels: For the binary labels letters 'A' and 'B' have been used. Other popular decision labels like 0 and 1 or true and false have not been used in order to make it more generic and avoid any bias from readers' point of view.

Fuzzy/Normal: This term identifies whether the results are from using FLIC. So, "N" stands for, no FLIC was used where as "F" stands for, FLIC was used.

Table 4.1 gives a summary of the data sets that have been used for training and testing.

Table 4.1: Data Sets

Properties / Data Set	No. of cases	No. of classes	No. of training cases	No. of testing cases	No. of class 'A' cases	No. of class 'B' cases	No. of attributes/ features
Internet Advertisement	1300	2	1000	300	865	435	1558
Adult	4000	2	2000	2000	3000	1000	122
Mushroom	7916	2	4000	3916	4000	3916	112
Telecom Churn	5000	2	2500	2500	4293	707	13

Table 4.2 is a table for testing parameters. The 3 parameters that were tweaked in order to get the best performance are nominal, normalized and cluster. Nominal stands for whether we used nominal class labels. Normalized stands for whether the training set and testing set were normalized before running the algorithm. Cluster stands for the number of interval cluster that were used to achieve the least error.

Table 4.2: Training and Testing Parameters

Parameters / Data Set	Nominal (order DT, NB and LR)	Normalized (order DT, NB and LR)	Cluster (order DT, NB and LR)
Internet Advertisement	False, True and True	False, False and False	10, 4 and 4
Adult	True, True and False	False, False and False	6, 4 and 10
Mushroom	False, True and False	False, False and False	4,10 and 4
Telecom Churn	True, True and False	True, False and False	6, 4 and 10

where, DT = Decision Tree, LR = Logistic Regression, NB = Naïve Bayes

4.1-Internet Advertisement

This data set has 1300 cases with 1558 features. 1000 cases have been used for training and 300 cases have been used for testing. According to UCI Machine Learning repository information, this data set represents a set of possible advertisements on Internet pages. The features encode the geometry of the image (if available) as well as phrases occurring in the URL, the image's URL and alt text, the anchor text, and words occurring near the anchor text. The task is to predict whether an image is an advertisement ("ad") or not ("nonad"). We call label ("nonad") class 'A' and label ("ad") class 'B'. Both normal and fuzzy training was done for this data set. The testing was then done using the normal model and the fuzzy model with the testing set. Figure 4.1.1 gives the mean absolute error (MAE) variation for the 3 methods, Decision Tree, Logistic Regression and Naïve Bayes.

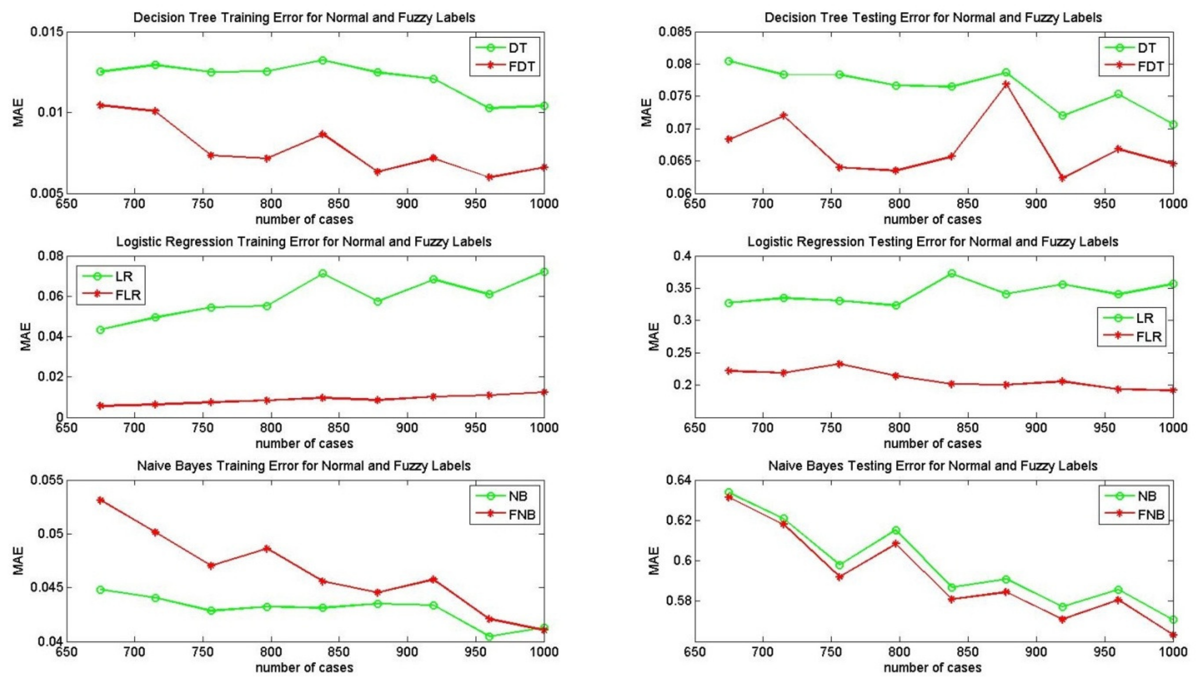

Fig 4.1.1: MAE vs. Number of Cases graph for DT, LR and NB (normal vs. fuzzy)

Figures 4.1.2, 4.1.3 and 4.1.4 give the precision, recall and f-measure for each of the methods and the 2 classes that is, class ‘A’ and ‘B’.

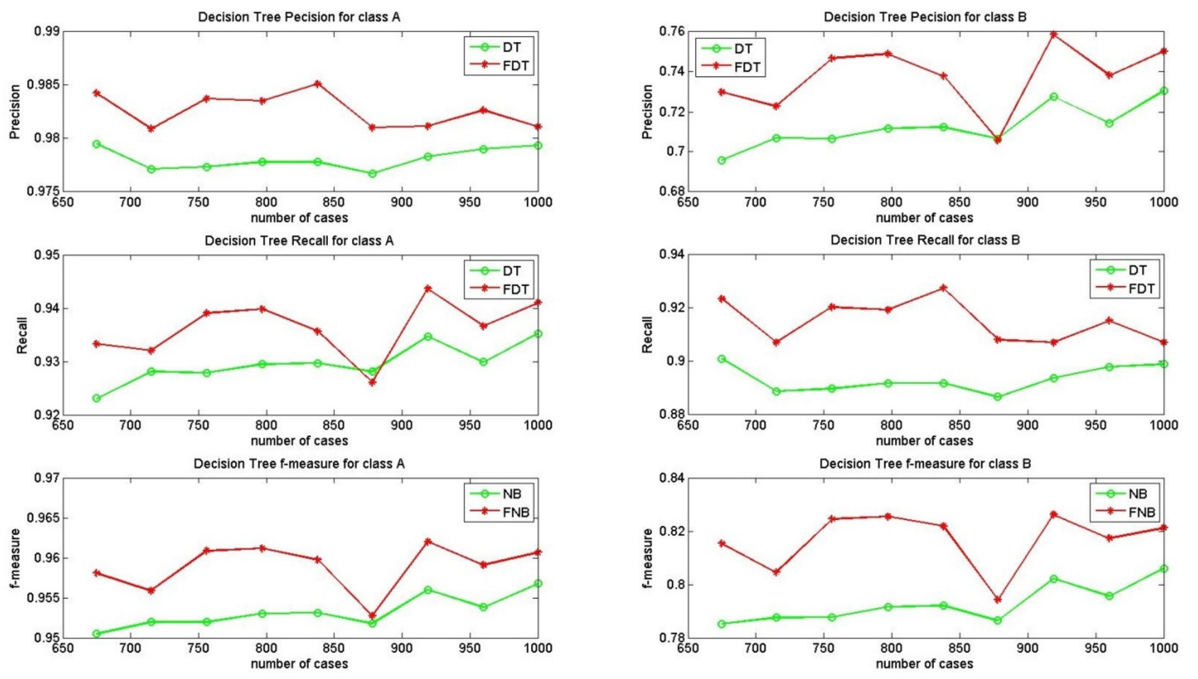

Fig 4.1.2: Precision, Recall and F-measure for Decision Tree

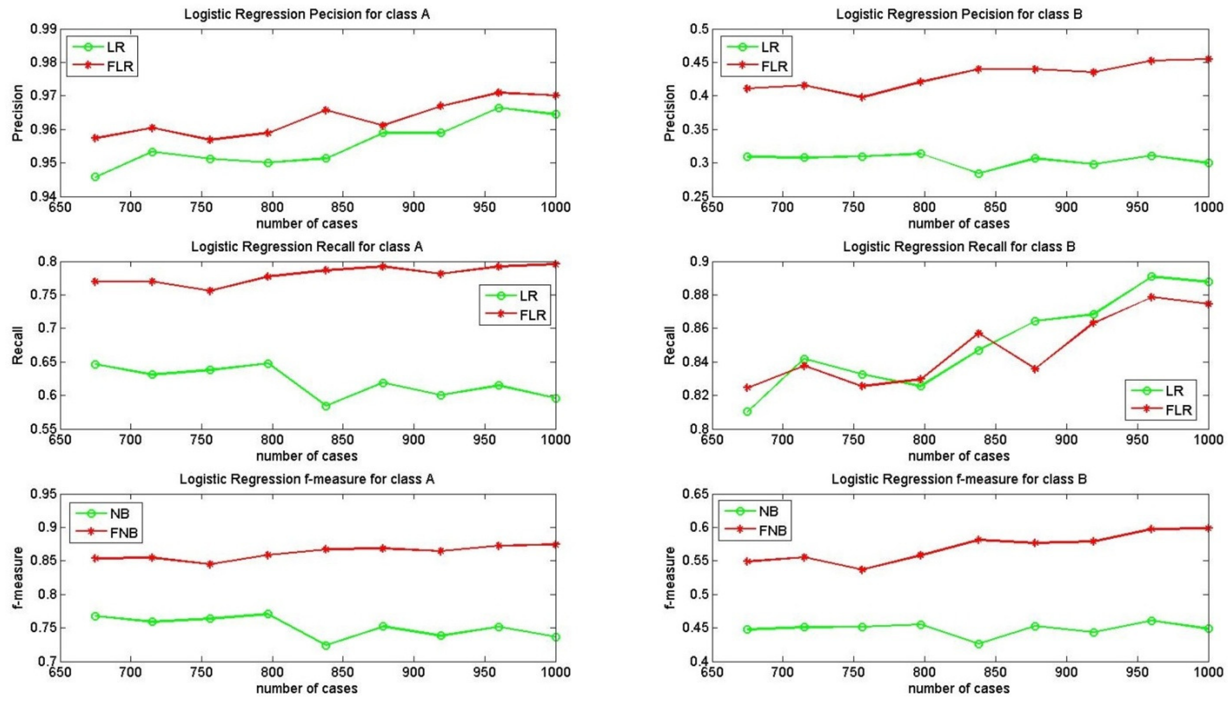

Fig 4.1.3: Precision, Recall and F-measure for Logistic Regression

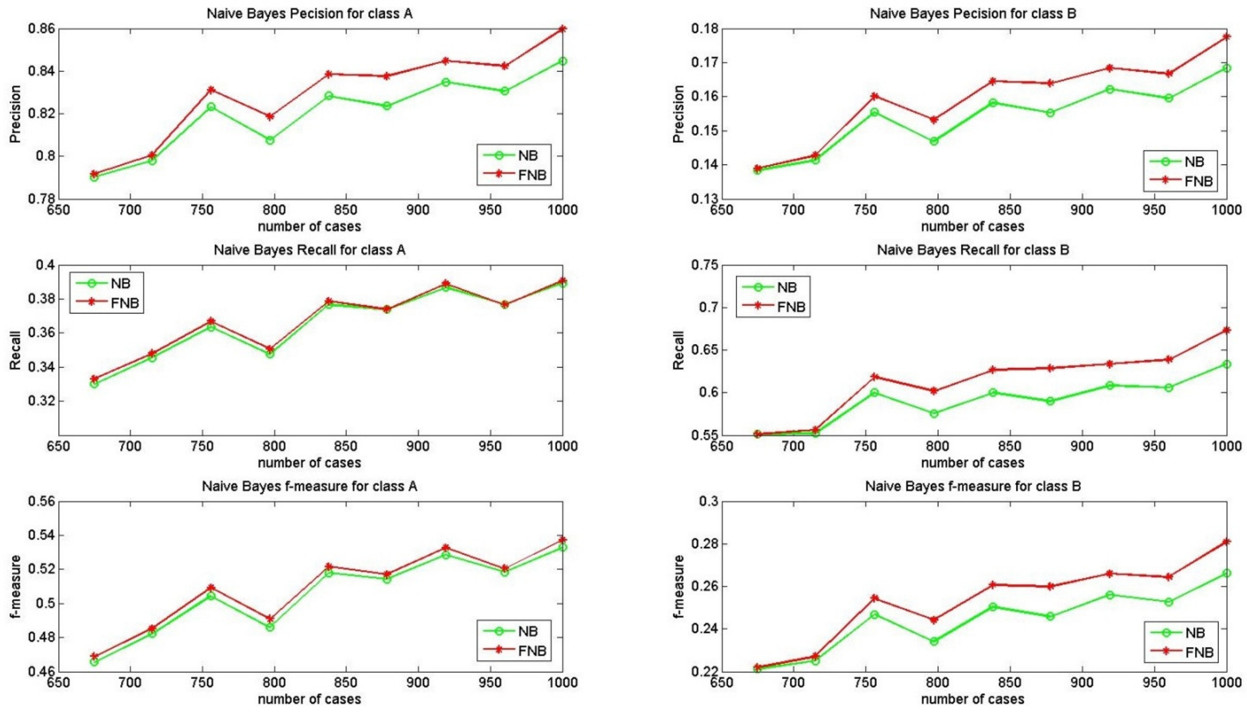

Fig 4.1.4: Precision, Recall and F-measure for Naïve Bayes

In Appendix-A Table 4.1.1, 4.1.2 and 4.1.3 lists the precision, recall and f-measure for the figures 4.1.2, 4.1.3 and 4.1.4. Due to the small size of the data set we have 9 iterations for this set instead of 10. A description of each table header is given in section 4.

4.2-Adult

This data set has 4000 cases and 122 features. 2000 cases were used for training and the remaining 2000 cases were used for testing. According to UCI Machine Learning repository data set information, the Extraction was done by Barry Becker from the 1994 Census database. A set of reasonably clean records was extracted using the following conditions: ((AAGE>16) && (AGI>100) && (AFNLWGT>1) && (HRSWK>0)). Prediction task is to determine whether a person makes over 50K a year. If one consider 0 as not making 50K a year and 1 as making 50K a year, class 'A' represents 0 and class "B" represents 1. Both normal and fuzzy training was done for this data set. The testing was then done using the normal model and the fuzzy model with the testing set. Figure 4.2.1 gives the mean absolute error (MAE) variation for the 3 methods, Decision Tree, Logistic Regression and Naïve Bayes:

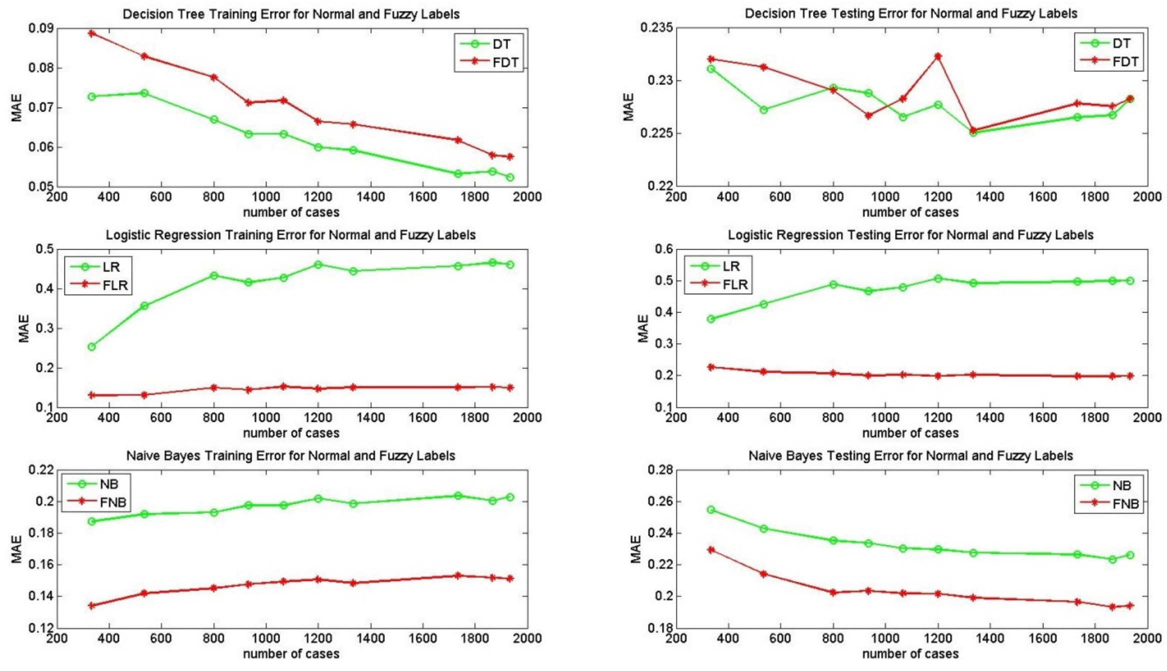

Fig 4.2.1: MAE vs. Number of Cases graph for DT, LR and NB (normal vs. fuzzy)

Figures 4.2.2, 4.2.3 and 4.2.4 give the precision, recall and f-measure for each of the methods and the 2 classes that is, class 'A' and 'B'.

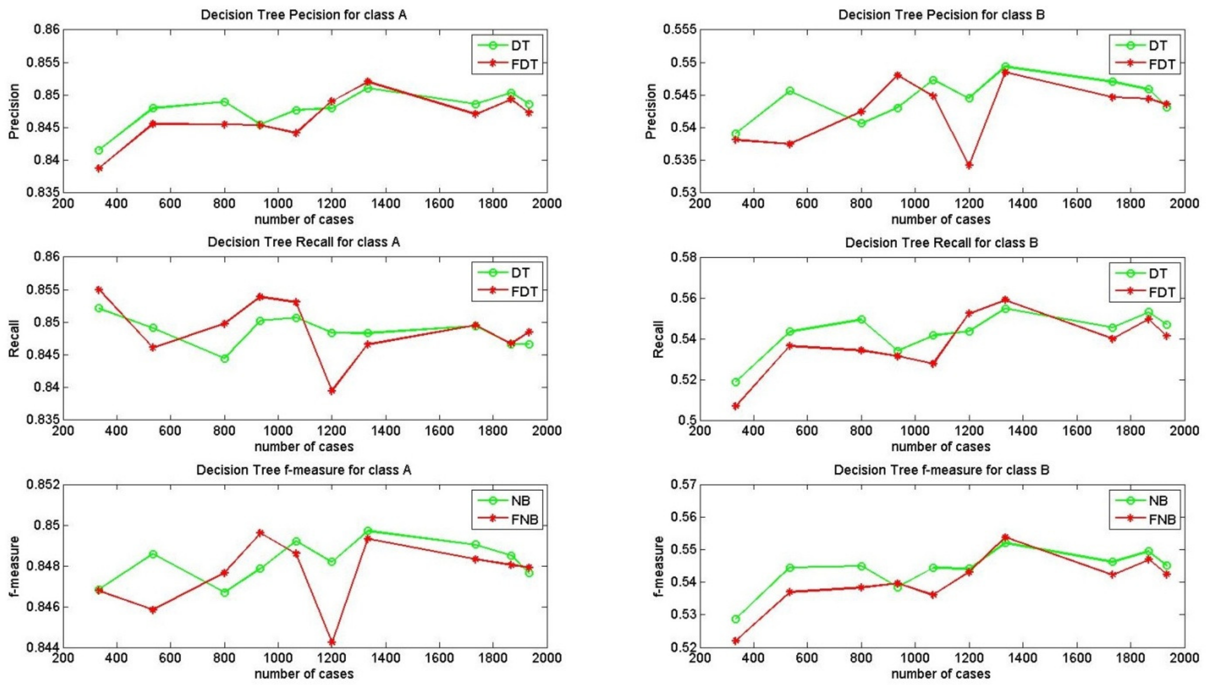

Fig 4.2.2: Precision, Recall and F-measure for Decision Tree

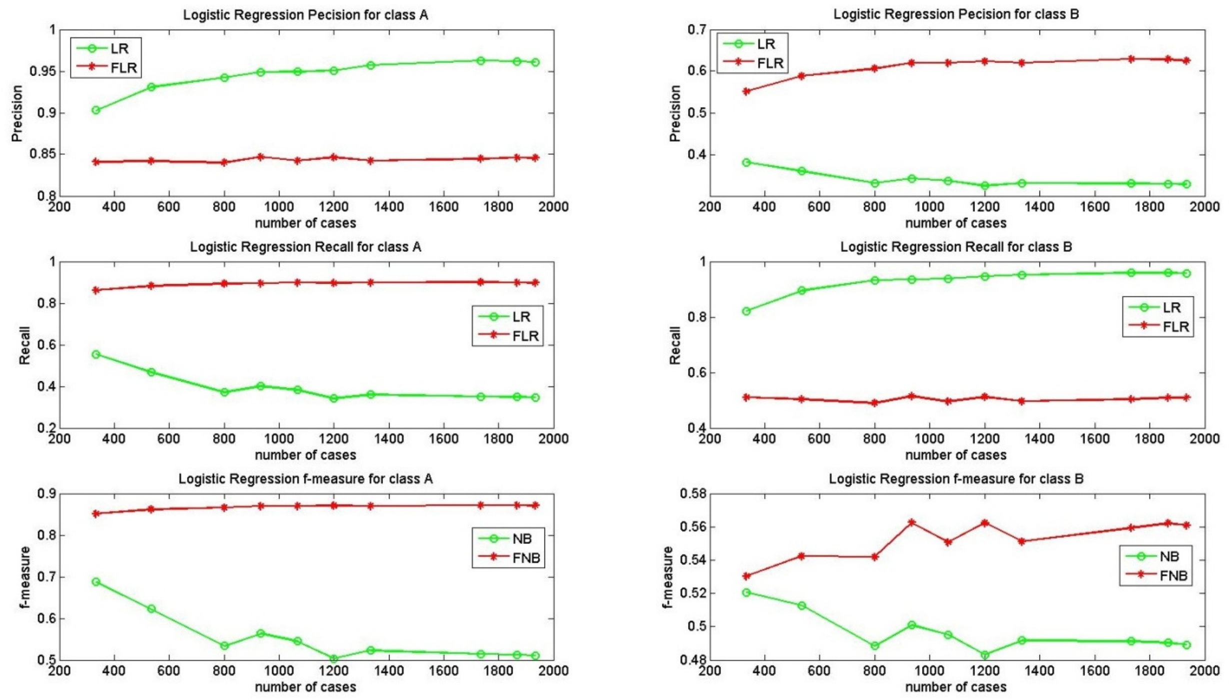

Fig 4.2.3: Precision, Recall and F-measure for Logistic Regression

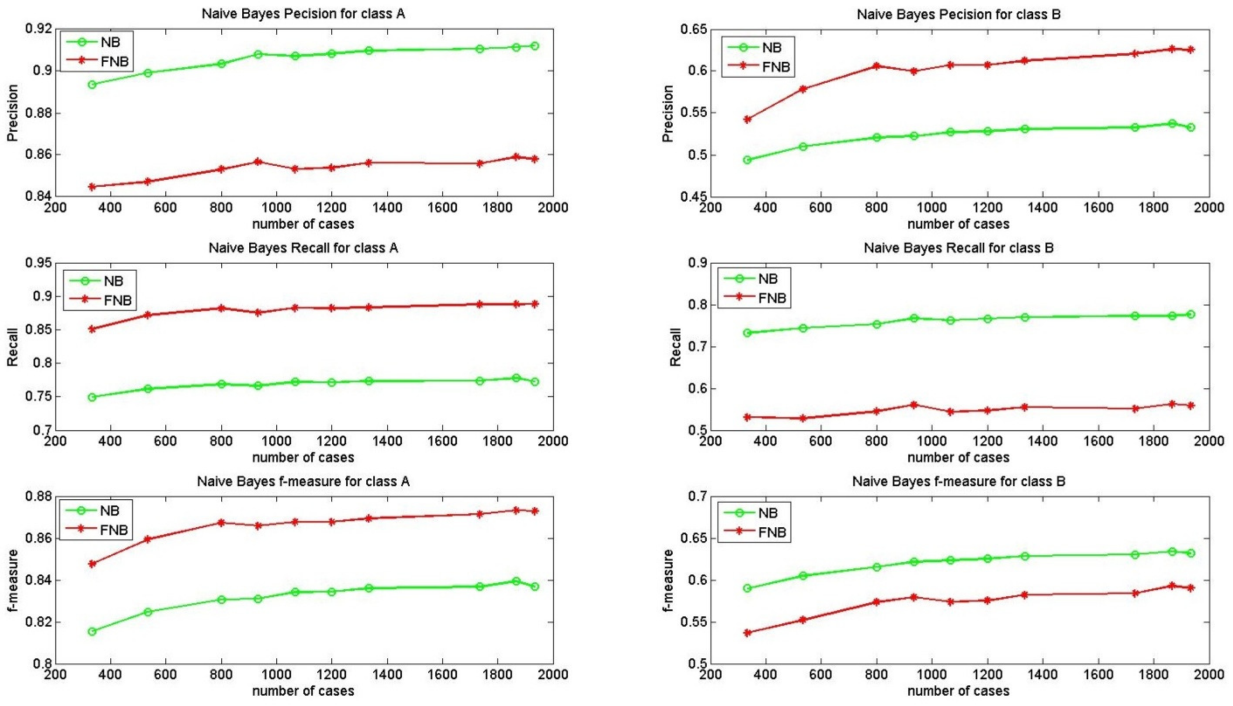

Fig 4.2.4: Precision, Recall and F-measure for Naïve Bayes

In Appendix-A Table 4.2.1, 4.2.2 and 4.2.3 lists the precision, recall and f-measure for the figures 4.2.2, 4.2.3 and 4.2.4. A description of each table header is given in section 4.

4.3-Mushroom

This data set has 7916 cases and 112 features. 4000 cases were used for training and the remaining 3916 cases were used for testing. According to UCI Machine Learning repository data set information, this data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family (pp. 500-525). Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one. The Guide clearly states that there is no simple rule for determining the edibility of a mushroom; no rule like ``leaflets three, let it be'' for Poisonous Oak and Ivy. This dataset was then taken by LIBSVM and they performed a preprocessing on the data. According to LIBSVM preprocessing information, Each nominal attribute was expanded into several binary attributes. The original attribute #12 had missing values and was not used. This research has used the LIBSVM version of the data set where the binary labels are 1 and 2. The other consideration was that, class 'A' represents 2 and class "B" represents 1. Both normal and fuzzy training was done for this data set. The testing was then done using the normal model and the fuzzy model with the testing set. Figure 4.3.1 gives the mean absolute error (MAE) variation for the 3 methods, Decision Tree, Logistic Regression and Naïve Bayes:

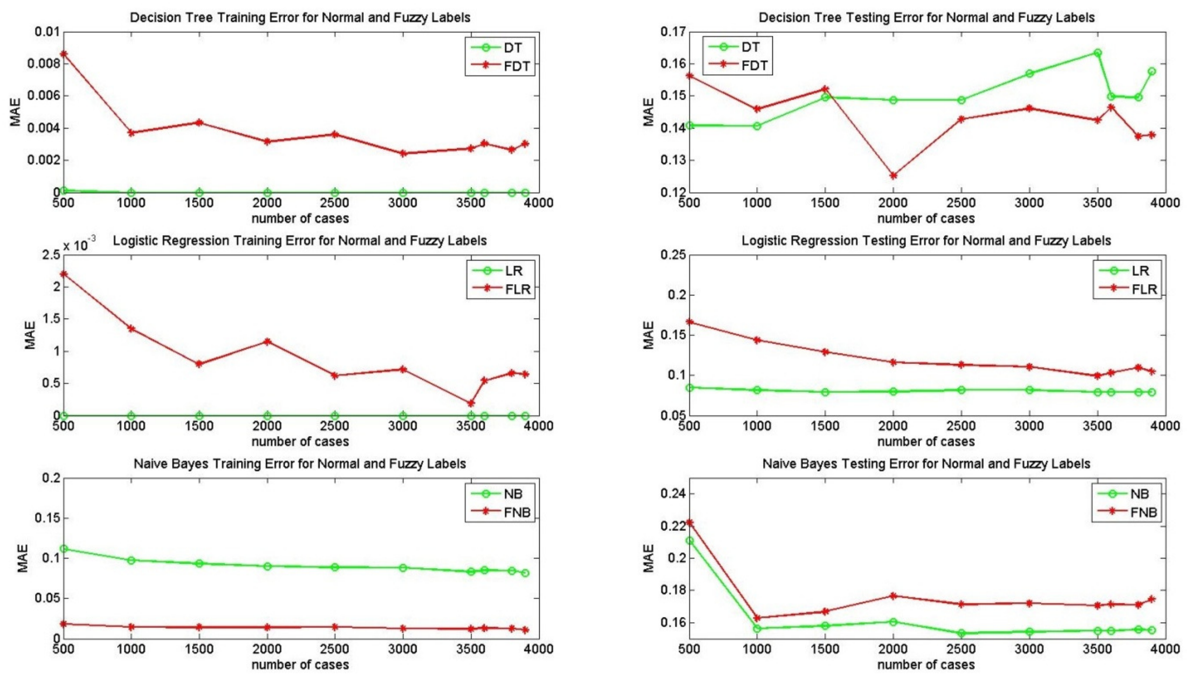

Fig 4.3.1: MAE vs. Number of Cases graph for DT, LR and NB (normal vs. fuzzy)

Figures 4.3.2, 4.3.3 and 4.3.4 give the precision, recall and f-measure for each of the methods and the 2 classes that is, class ‘A’ and ‘B’.

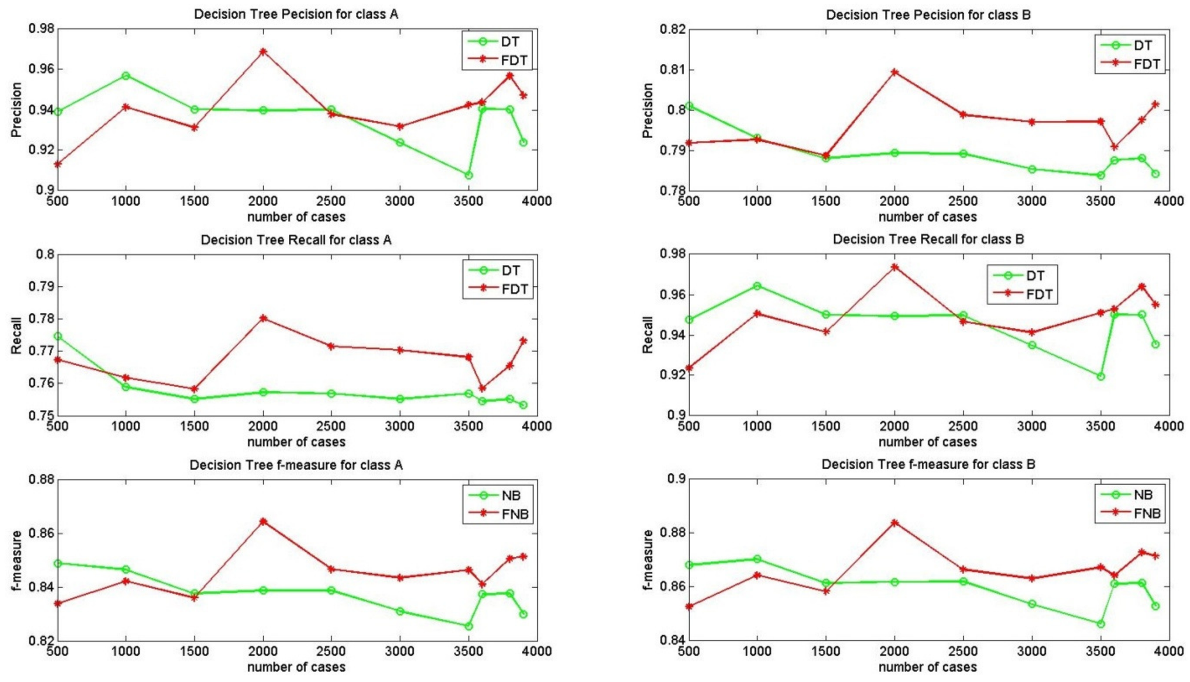

Fig 4.3.2: Precision, Recall and F-measure for Decision Tree

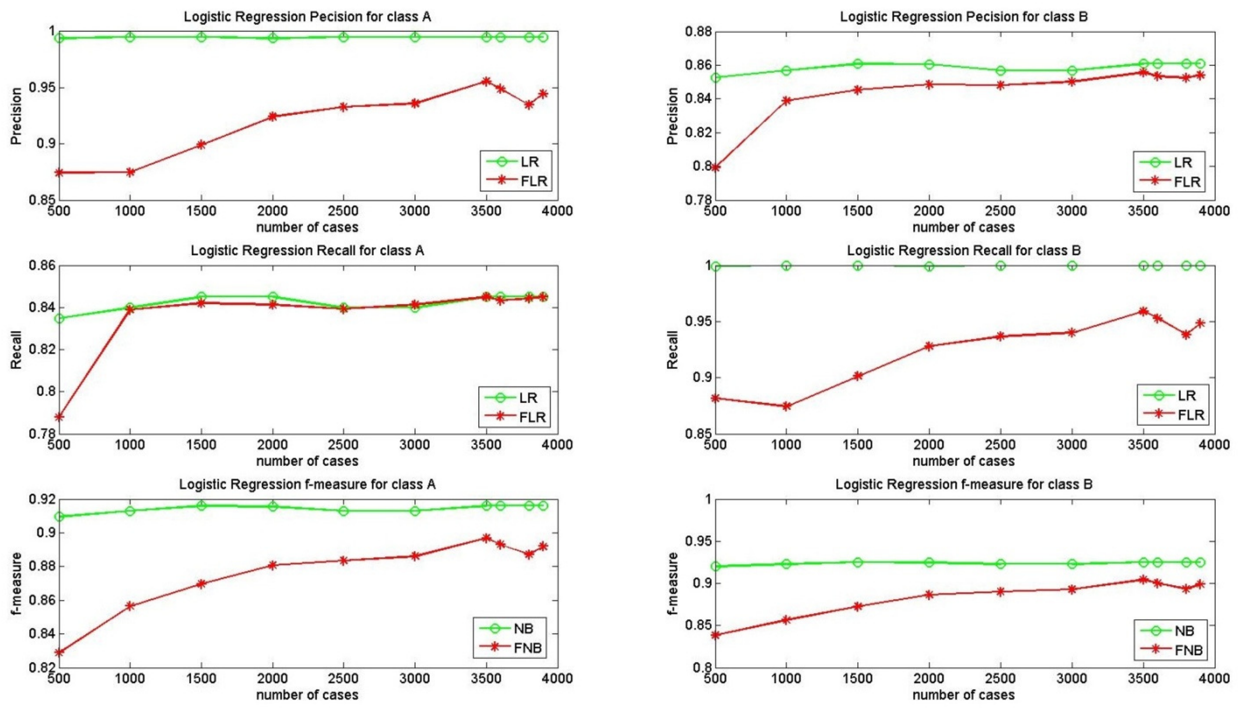

Fig 4.3.3: Precision, Recall and F-measure for Logistic Regression

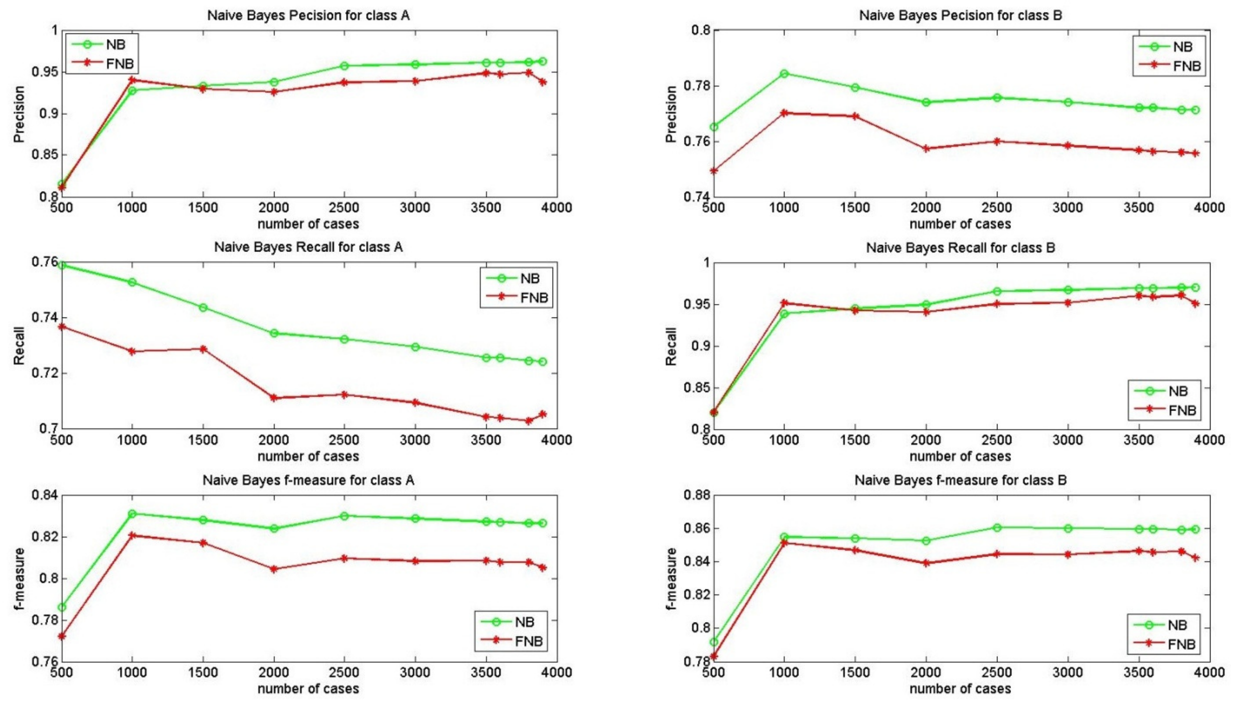

Fig 4.3.4: Precision, Recall and F-measure for Naïve Bayes

In Appendix-A Table 4.3.1, 4.3.2 and 4.3.3 lists the precision, recall and f-measure for the figures 4.3.2, 4.3.3 and 4.3.4. A description of each table header is given in section 4.

4.4-Telecom Churn

This data set has 5000 cases and 13 features. 2500 cases were used for training and the remaining 2500 cases were used for testing. This data set represents customers and their calling information such as number of night calls, day call and evening calls. Also information like international calls and average number of minutes used for all of the above for a period of one month. Some of these customers at some point discontinued the telecom service. Given new customer information, the goal is to predict how likely they are to discontinue the service at some point are. If one consider “False” as people who are still continuing the service and “True” as people who have discontinued the service, in this discussion we refer to “False” as class ‘A’ and “True” as class ‘B’. Both normal and fuzzy training was done for this data set. The testing was then done using the normal model and the fuzzy model with the testing set. Figure 4.1.1 gives the mean absolute error (MAE) variation for the 3 methods, Decision Tree, Logistic Regression and Naïve Bayes:

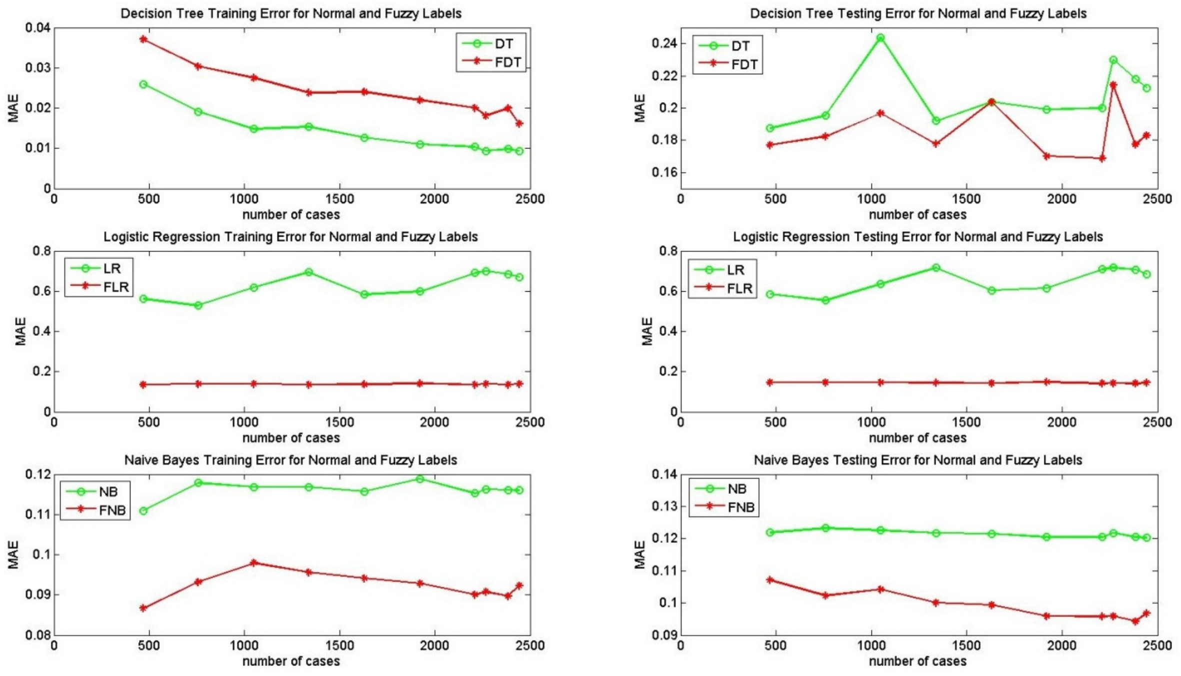

Fig 4.4.1: MAE vs. Number of Cases graph for DT, LR and NB (normal vs. fuzzy)

Figures 4.4.2, 4.4.3 and 4.4.4 give the precision, recall and f-measure for each of the methods and the 2 classes that is, class ‘A’ and ‘B’.

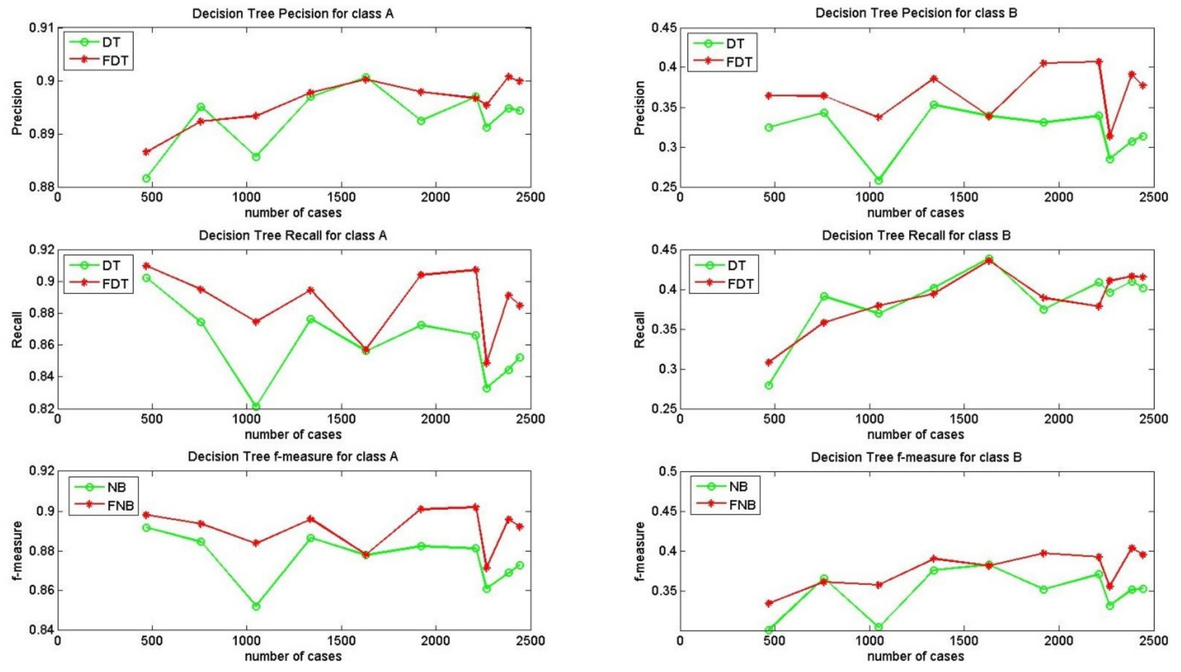

Fig 4.4.2: Precision, Recall and F-measure for Decision Tree

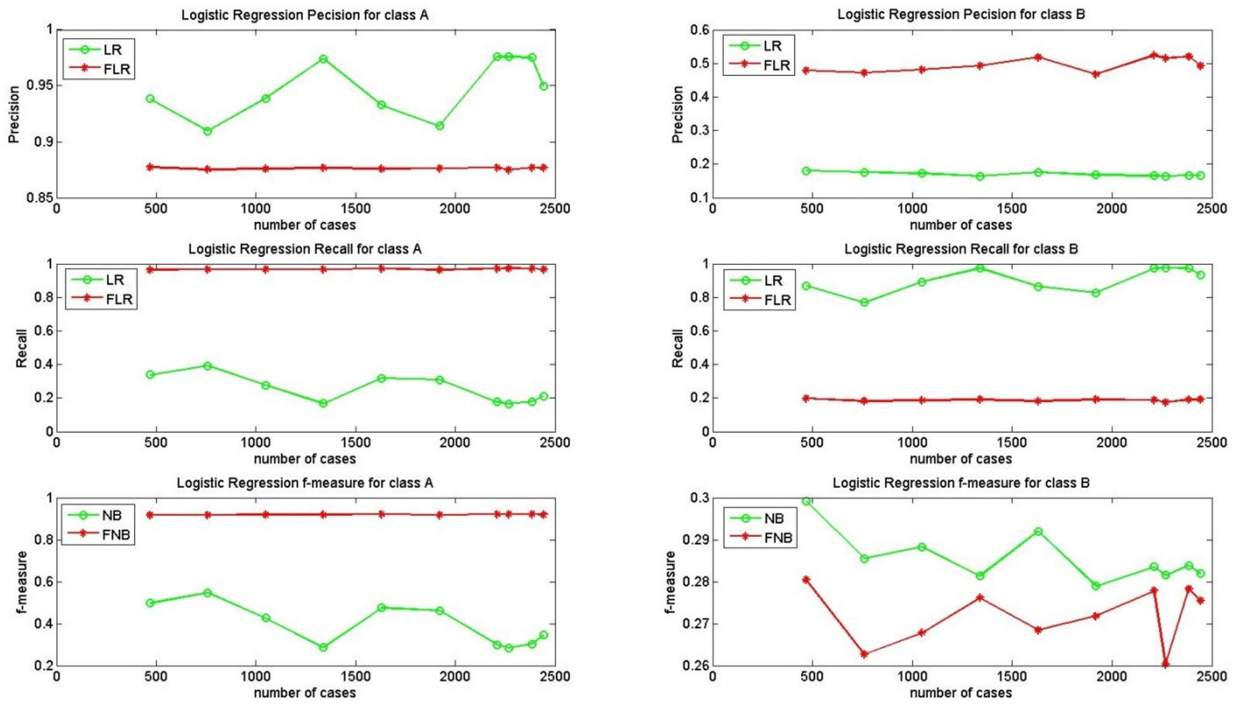

Fig 4.4.3: Precision, Recall and F-measure for Logistic Regression

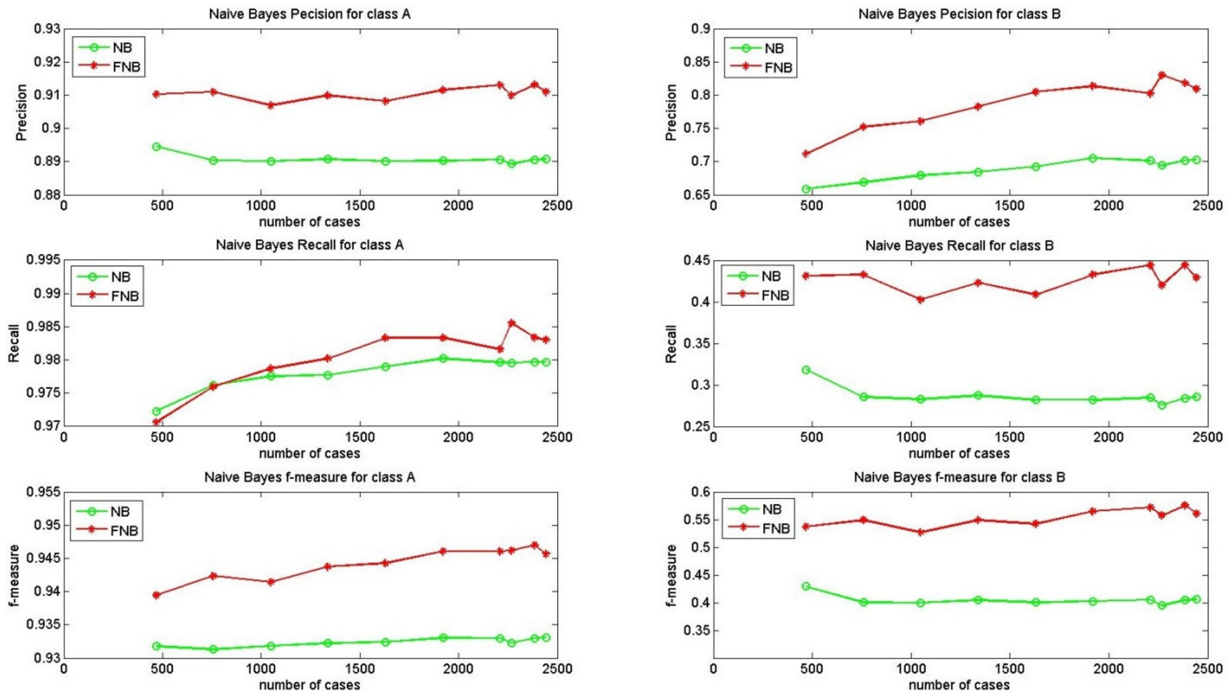

Fig 4.4.4: Precision, Recall and F-measure for Naïve Bayes

In Appendix-A Table 4.4.1, 4.4.2 and 4.4.3 lists the precision, recall and f-measure for the figures 4.4.2, 4.4.3 and 4.4.4. A description of each table header is given in section 4.

4.5-Result Summary

Table 4.5 shows the summary of accuracy for all the three classifiers.

Table 4.5: Summary of Accuracy (%)

Classifiers / Data Set	DT	FDT	LR	FLR	NB	FNB
Internet Advertisement	92.4	**93.3**	65.6	**79.1**	40.2	**40.8**
Adult	77.2	77.1	52.7	**79.6**	76.7	**79.64**
Mushroom	84.9	**85.7**	91.9	88.1	83.9	82.4
Telecom Churn	79.2	**81.5**	34.7	**85.6**	87.9	**90.1**

5-Discussion

Figure 4.1.1 represents the mean absolute error for the Internet advertisement data set. The green line shows the performance of the algorithms without using fuzzy labeling for the training set whereas the red line is the performance of the same algorithm after using fuzzy labeling. As one can see, testing error for all the 3 methods i.e. Decision Tree, Logistic Regression and Naive Bayes have gone down using the fuzzy logic technique. In figure 4.1.1, from the graphs, it is also clear that number of cases do not have an aggressive influence on the training error. Nevertheless, on average, the error rate decreases as the number of cases increases but exceptions can be observed with occasional spikes in the graph. Also, in figure 4.1.1, for Logistic Regression, the training error increases with increase in number of cases. This suggests that the model is finding it hard to predict the right set of parameters for the training set. This projects into the testing set and we can see a high error rate for testing compared to the training error. The testing error for Naïve Bayes is very high for both with or without FLIC. Although, use of FLIC lowers the error to some extent, overall, the error rate is about 50% on average. With the Internet Advertisement data set, Decision Tree performs the best with an error rate of only about 5% on average.

Tables 4.1.1, 4.1.2 and 4.1.3 in Appendix-A list precision, recall and f-measure for Internet Advertisement data set. The graphs for the two classes along with precision, recall and f-measure are also given in figure 4.1.2, 4.1.3 and 4.1.4. For Decision Tree in figure 4.1.2; all the three measure on average shows improvement after the use of FLIC (the red line). As far as Logistic Regression is concerned, for Internet Advertisement, precision and f-measure on average improves after using FLIC. Recall for class 'B' in Logistic Regression seems to fall slightly on average. This is an indication that, for the data set Internet Advertisement, even though FLIC with Logistic Regression classifies class 'B' more accurately, the number of class 'B' cases being classified falls. In other word, the significance of accuracy of class 'B' falls with increase in number of cases. This fall in recall however is not too significant as the f-measure on average improves; which suggests that the fall in recall is more than compensated by the improvement in precision for class precision 'B'. In case of Naïve Bayes even though fuzzy labeling does better than normal labeling for most of the cases the precision, recall and f-measure for this data set is significantly low.

Figure 4.2.1 represents the graph of MAE vs. Number of cases for the Adult data set. One can see that using fuzzy labeling for Logistic Regression and Naïve Bayes improves the error rate significantly whereas Decision Trees with fuzzy labeling even though improves the performance, the difference is not as significant as Logistic Regression. An interesting observation is; the training error for Decision Tree goes down as it sees more cases where as for

Logistic Regression and Naïve Bayes, the training error goes up with increase in number of cases. The testing error however steadily goes down with increase in number of training cases. This is an indication that the models have generalized well. The most interesting observation here perhaps is the Logistic Regression training and testing error curve. For this data set, Logistic Regression training and testing error goes up as the number of cases increases which is an indication that in general, Logistic Regression is not a good choice for this data set. But using FLIC, the error rate drops down significantly and makes FLR the best choice as a classifier among the 3 classifiers. This is a very positive indication for FLIC as this is a strong suggestion that FLIC is helping the classifier to learn with misclassification and generalize better.

In figure 4.2.2, one can observe that Decision Tree precision, recall and f-measure remains almost constant before and after using FLIC. For the Adult data set, FLIC does not have much effect on the performance of Decision Tree. In figure 4.2.3 i.e. for Logistic Regression, an interesting observation is, precision for class 'A' goes down whereas precision for class 'B' goes up. On the other hand, recall for class 'A' increases whereas recall for class 'B' falls. In other word, a lot more class 'A' cases are being identified from the entire population but the classification is not as accurate and on the other hand, although fewer class 'B' cases are being identified, the accuracy of those being identified are much higher. The f-measure is an interesting measure as it tells, this imbalance in precision and recall is being over-compensated and the overall performance after using FLIC is better than normal Logistic Regression. Also, for this data set, if one goes to table 4.1, one can see that this data set has a class asymmetry with only one fourth of all cases in class ‘B’. Class asymmetry is another problem that cannot always be improved using the normal method. This is also an indication that FLIC particularly does better in case of problems with class asymmetry. Looking at graph 4.2.4, one can say that precision, recall and f-measure for Naïve Bayes with FLIC does better than normal Naïve Bayes, but the improvement is not as significant as Logistic Regression. Looking at the graph, one can say that, using Naive Bayes, Class 'A' seems to be doing better than class 'B', particularly, for this Adult data set.

Figure 4.3.1 represents the graph of MAE vs. number of cases for the Mushroom data set. This data set has almost equal number of class ‘A’ and class ‘B’ cases. As one can see, the performance of Decision Tree using FLIC has improved but has some unusual spikes. This indicates the performance may not be stable. The performance of Logistic Regression and Naïve Bayes has gone down using FLIC. This data set is an example where FLIC has not generalized well. An interesting observation here is that, the training and testing error for all the methods are already significantly low. At this low error rate, FLIC may not have a lot to contribute. In other word, FLIC works well where the misclassification rate is significantly high. This data set is a good example where the models without FLIC have already learned to generalize. This is also indicated by the zero training error both for Decision Tree and Logistic Regression in figure 4.3.1. Figure 4.3.2, 4.3.3 and 4.3.4 shows the precision, recall and error rate for the 3 methods on Mushroom data set. As one can clearly see, the algorithms on average perform better without FLIC.

Figure 4.4.1 represents the MAE vs. number of cases for the Telecom Churn data set. Again as one can see the methods with FLIC has lower testing error than methods without FLIC. Logistic Regression has the most significant improvement in error rate which is from about 60% average error rate using normal method to only 12%-13% error using FLIC. Decision Tree and Naïve Bayes does equally well with Naïve Bayes giving one of the lowest error rate for the Telecom Churn data set.

The precision, recall and f-measure for the 3 methods on this data set are given in graph 4.4.2, 4.4.3 and 4.4.4. For Decision Tree the precision, recall and f-measure on average does slightly better when FLIC is used. For Logistic Regression, the precision for class ‘B’ increases but the precision for class ‘A’ decreases slightly. As far as recall is concerned, for Logistic Regression, one very interesting observation is, the recall for class ‘A’ increases significantly after using FLIC but the recall for class ‘B’ on the other hand decreases significantly. This means, even though the model with FLIC is able to identify class ‘B’ with higher precision, only a handful of class ‘B’ cases are being identified from the set of all class ‘B’ cases. This drop in recall for class 'B' is not compensated by precision and so the f-measure for class 'B' falls using FLIC. On the other hand, class 'A' does significantly better overall. In case of Naïve Bayes, the precision, recall and f-measure do better using FLIC for all cases. For the Telecom Churn data set, Naïve Bayes has the best overall performance.

6-Conclusion

In this research, it has been shown that Fuzzy Logic and Interval Clustering (FLIC) technique can be generalized and can be integrated with existing machine learning algorithms as a preprocessor to improve prediction accuracy. Results from the experiments show positive evidence of FLIC’s capability to improve the prediction accuracy for Decision Tree, Logistic Regression and Naive Bayes. This work also shows that FLIC performs particularly well for problems with class asymmetry which is a common issue in most binary classification problems. This method is independent of case size and also independent of number of features, which makes this method very flexible to implement. For data sets where the feature space is highly overlapping, FLIC works particularly well by learning with misclassification. This method requires no change in the existing machine learning algorithms and hence can be integrated with most existing machine learning system with ease. Overall, this research indicated that, FLIC can become a very useful and transparent tool for most existing popular machine learning techniques.

References

1. "Fuzzy sets", Information and Control 8 (3): 338–353 by Zadeh, L.A. (1965).
2. "Review of Mathematics of fuzzy logics in The Bulletin of Symbolic Logic, Vol. 6, No.3, (Sep. 2000), 342-346, JSTOR 421060", Francis Jeffry Pelletier.
3. "Foundations of Machine Learning", Mehryar Mohri, Afshin Rostamizadeh, Ameet Talwalkar (2012) The MIT Press; ISBN 9780262018258.
4. S. Chakrabarti, S. Svojanovsky, "Artificial Neural Network--Based Analysis of High-Throughput Screening Data for Improved Prediction", Journal of Biomolecular Screening.
5. Zengchang Qin, "Learning with Fuzzy labels", PhD dissertation University of Bristol.
6. M.-S. YANG, "A Survey of Fuzzy Clustering", Department of Mathematics, Chung Yuan Christian University.
7. Matjaž Juršič, Nada Lavrač, "Fuzzy Clustering of Documents", Department of Knowledge Discovery, Jozef Stefan Institute.
8. Jianwen Fang,1,2 Yinghua Dong,1 Gerald H. Lushington, "Support Vector Machines in HTS DataMining: Type I MetAPs Inhibition Study", Journal of Biomolecular Screening.
9. Michael Pazzani, Christopher Merz, Patrick Murphy Kamal Ali, Timothy Hume and Clifford Brunk, "Reducing Misclassification Costs".
10. Shai Ben-David, David Loker, Nathan Srebro, Karthik Sridharan, "Minimizing The Misclassification Error Rate Using a Surrogate Convex Loss".
11. Yoav Freund Robert E. Schapire, "A Short Introduction to Boosting", AT&T research labs.

12. James M. Keller, Michael R. Gray and James A. Givens, JR. "A Fuzzy K-Nearest Neighbor Algorithm".
13. Mansoor Zolghadri Jahromi, Mohammad Reza Moosavi, "Designing Cost-sensitive Fuzzy Classification Systems Using Rule-weight".
14. Massih-Reza Amini, Patrick Gallinari, "Semi-Supervised Learning with Explicit Misclassification Modeling"
15. Hans Roubos, Magne Setnes, and Janos Abonyi, "Learning Fuzzy Classification Rules from Data"
16. Sushmita Mitra, Sankar K. Pal, "Self-organizing Neural Network As A Fuzzy Classifier"
17. Margarita Carmen S. Paterno, "Fuzzy Semantic Labeling Of Natural Images"
18. J. V. Uspensky, "Introduction to Mathematical Probability"-chapter IV
19. Stuart Russel and Peter norvig, "Artificial Intelligence-A modern Approach"-chapter 18.1, 18.2, 20.1.
20. Jerzy W. Grzymala-Busse, Rule Induction.
21. Erin L. Allwein, Robert E. Schapire, Yoram Singer, "Reducing Multiclass to Binary:A Unifying Approach for Margin Classifiers".
22. Ryan Rifkin, "Multiclass Classification", 9.520 Class 06, 25 Feb 2008.
23. Hal Daume III, "A Course in Machine Learning"- chapter 5.1, page 69.

Appendix A: Tables for section 4

Internet Advertise

Table 4.1.1: Decision Tree

Sample size	Precision	Recall	F-measure	Binary labels	Fuzzy/Normal (F/N)
1	0.98	0.92	0.95	A	N
	0.70	0.90	0.79	B	
	0.98	0.93	0.96	A	F
	0.73	0.92	0.82	B	
2	0.98	0.93	0.95	A	N
	0.71	0.89	0.79	B	
	0.98	0.93	0.96	A	F
	0.72	0.91	0.80	B	
3	0.98	0.93	0.95	A	N
	0.71	0.89	0.79	B	
	0.98	0.94	0.96	A	F
	0.75	0.92	0.82	B	
4	0.98	0.93	0.95	A	N
	0.71	0.89	0.79	B	
	0.98	0.94	0.96	A	F
	0.75	0.92	0.83	B	
5	0.98	0.93	0.95	A	N
	0.71	0.89	0.79	B	
	0.99	0.94	0.96	A	F
	0.74	0.93	0.82	B	
6	0.98	0.93	0.95	A	N
	0.71	0.89	0.79	B	
	0.98	0.93	0.95	A	F
	0.71	0.91	0.79	B	
7	0.98	0.93	0.96	A	N
	0.73	0.89	0.80	B	
	0.98	0.94	0.96	A	F
	0.76	0.91	0.83	B	
8	0.98	0.93	0.95	A	N
	0.71	0.90	0.80	B	
	0.98	0.94	0.96	A	F
	0.74	0.92	0.82	B	
9	0.98	0.94	0.96	A	N
	0.73	0.90	0.81	B	
	0.98	0.94	0.96	A	F
	0.75	0.91	0.82	B	

Table 4.1.2: Logistic Regression

Sample size	Precision	Recall	F-measure	Binary labels	Fuzzy/Normal (F/N)
1	0.95	0.65	0.77	A	N
	0.31	0.81	0.45	B	
	0.96	0.77	0.85	A	F
	0.41	0.82	0.55	B	
2	0.95	0.63	0.76	A	N
	0.31	0.84	0.45	B	
	0.96	0.77	0.85	A	F
	0.42	0.84	0.56	B	
3	0.95	0.64	0.76	A	N
	0.31	0.83	0.45	B	
	0.96	0.76	0.84	A	F
	0.40	0.83	0.54	B	
4	0.95	0.65	0.77	A	N
	0.31	0.83	0.45	B	
	0.96	0.78	0.86	A	F
	0.42	0.83	0.56	B	
5	0.95	0.58	0.72	A	N
	0.28	0.85	0.43	B	
	0.97	0.79	0.87	A	F
	0.44	0.86	0.58	B	
6	0.96	0.62	0.75	A	N
	0.31	0.86	0.45	B	
	0.96	0.79	0.87	A	F
	0.44	0.84	0.58	B	
7	0.96	0.60	0.74	A	N
	0.30	0.87	0.44	B	
	0.97	0.78	0.86	A	F
	0.44	0.86	0.58	B	
8	0.97	0.61	0.75	A	N
	0.31	0.89	0.46	B	
	0.97	0.79	0.87	A	F
	0.45	0.88	0.60	B	
9	0.96	0.60	0.74	A	N
	0.30	0.89	0.45	B	
	0.97	0.80	0.87	A	F
	0.45	0.87	0.60	B	

Table 4.1.3: Naïve Bayes

Sample size	Precision	Recall	F-measure	Binary labels	Fuzzy/Normal (F/N)
1	0.79	0.33	0.47	A	N
	0.14	0.55	0.22	B	
	0.79	0.33	0.47	A	F
	0.14	0.55	0.22	B	
2	0.80	0.35	0.48	A	N
	0.14	0.55	0.23	B	
	0.80	0.35	0.49	A	F
	0.14	0.56	0.23	B	
3	0.82	0.36	0.50	A	N
	0.16	0.60	0.25	B	
	0.83	0.37	0.51	A	F
	0.16	0.62	0.25	B	
4	0.81	0.35	0.49	A	N
	0.15	0.58	0.23	B	
	0.82	0.35	0.49	A	F
	0.15	0.60	0.24	B	
5	0.83	0.38	0.52	A	N
	0.16	0.60	0.25	B	
	0.84	0.38	0.52	A	F
	0.16	0.63	0.26	B	
6	0.82	0.37	0.51	A	N
	0.16	0.59	0.25	B	
	0.84	0.37	0.52	A	F
	0.16	0.63	0.26	B	
7	0.83	0.39	0.53	A	N
	0.16	0.61	0.26	B	
	0.84	0.39	0.53	A	F
	0.17	0.63	0.27	B	
8	0.83	0.38	0.52	A	N
	0.16	0.61	0.25	B	
	0.84	0.38	0.52	A	F
	0.17	0.64	0.26	B	
9	0.84	0.39	0.53	A	N
	0.17	0.63	0.27	B	
	0.86	0.39	0.54	A	F
	0.18	0.67	0.28	B	

Adult

Table 4.2.1: Decision Tree

Sample size	Precision	Recall	F-measure	Binary labels	Fuzzy/Normal (F/N)
1	0.84	0.85	0.85	A	N
	0.54	0.52	0.53	B	
	0.84	0.86	0.85	A	F
	0.54	0.51	0.52	B	
2	0.85	0.85	0.85	A	N
	0.55	0.54	0.54	B	
	0.85	0.85	0.85	A	F
	0.54	0.54	0.54	B	
3	0.85	0.84	0.85	A	N
	0.54	0.55	0.54	B	
	0.85	0.85	0.85	A	F
	0.54	0.53	0.54	B	
4	0.85	0.85	0.85	A	N
	0.54	0.53	0.54	B	
	0.85	0.85	0.85	A	F
	0.55	0.53	0.54	B	
5	0.85	0.85	0.85	A	N
	0.55	0.54	0.54	B	
	0.84	0.85	0.85	A	F
	0.54	0.53	0.54	B	
6	0.85	0.85	0.85	A	N
	0.54	0.54	0.54	B	
	0.85	0.84	0.84	A	F
	0.53	0.55	0.54	B	
7	0.85	0.85	0.85	A	N
	0.55	0.55	0.55	B	
	0.85	0.85	0.85	A	F
	0.55	0.56	0.55	B	
8	0.85	0.85	0.85	A	N
	0.55	0.55	0.55	B	
	0.85	0.85	0.85	A	F
	0.54	0.54	0.54	B	
9	0.85	0.85	0.85	A	N
	0.55	0.55	0.55	B	
	0.85	0.85	0.85	A	F
	0.54	0.55	0.55	B	
10	0.85	0.85	0.85	A	N
	0.54	0.55	0.55	B	
	0.85	0.85	0.85	A	F
	0.54	0.54	0.54	B	

Table 4.2.2: Logistic Regression

Sample size	Precision	Recall	F-measure	Binary labels	Fuzzy/Normal (F/N)
1	0.90	0.56	0.69	A	N
	0.38	0.82	0.52	B	
	0.84	0.86	0.85	A	F
	0.55	0.51	0.53	B	
2	0.93	0.47	0.62	A	N
	0.36	0.90	0.51	B	
	0.84	0.88	0.86	A	F
	0.59	0.50	0.54	B	
3	0.94	0.37	0.53	A	N
	0.33	0.93	0.49	B	
	0.84	0.89	0.87	A	F
	0.61	0.49	0.54	B	
4	0.95	0.40	0.56	A	N
	0.34	0.94	0.50	B	
	0.85	0.89	0.87	A	F
	0.62	0.51	0.56	B	
5	0.95	0.38	0.55	A	N
	0.34	0.94	0.50	B	
	0.84	0.90	0.87	A	F
	0.62	0.50	0.55	B	
6	0.95	0.34	0.50	A	N
	0.32	0.95	0.48	B	
	0.85	0.90	0.87	A	F
	0.62	0.51	0.56	B	
7	0.96	0.36	0.52	A	N
	0.33	0.95	0.49	B	
	0.84	0.90	0.87	A	F
	0.62	0.50	0.55	B	
8	0.96	0.35	0.51	A	N
	0.33	0.96	0.49	B	
	0.84	0.90	0.87	A	F
	0.63	0.50	0.56	B	
9	0.96	0.35	0.51	A	N
	0.33	0.96	0.49	B	
	0.85	0.90	0.87	A	F
	0.63	0.51	0.56	B	
10	0.96	0.35	0.51	A	N
	0.33	0.96	0.49	B	
	0.85	0.90	0.87	A	F
	0.62	0.51	0.56	B	

Table 4.2.3: Naïve Bayes

Sample size	Precision	Recall	F-measure	Binary labels	Fuzzy/Normal (F/N)
1	0.89	0.75	0.82	A	N
	0.49	0.73	0.59	B	
	0.84	0.85	0.85	A	F
	0.54	0.53	0.54	B	
2	0.90	0.76	0.82	A	N
	0.51	0.74	0.60	B	
	0.85	0.87	0.86	A	F
	0.58	0.53	0.55	B	
3	0.90	0.77	0.83	A	N
	0.52	0.75	0.62	B	
	0.85	0.88	0.87	A	F
	0.61	0.54	0.57	B	
4	0.91	0.77	0.83	A	N
	0.52	0.77	0.62	B	
	0.86	0.88	0.87	A	F
	0.60	0.56	0.58	B	
5	0.91	0.77	0.83	A	N
	0.53	0.76	0.62	B	
	0.85	0.88	0.87	A	F
	0.61	0.54	0.57	B	
6	0.91	0.77	0.83	A	N
	0.53	0.77	0.63	B	
	0.85	0.88	0.87	A	F
	0.61	0.55	0.58	B	
7	0.91	0.77	0.84	A	N
	0.53	0.77	0.63	B	
	0.86	0.88	0.87	A	F
	0.61	0.55	0.58	B	
8	0.91	0.77	0.84	A	N
	0.53	0.77	0.63	B	
	0.86	0.89	0.87	A	F
	0.62	0.55	0.58	B	
9	0.91	0.78	0.84	A	N
	0.54	0.77	0.63	B	
	0.86	0.89	0.87	A	F
	0.63	0.56	0.59	B	
10	0.91	0.77	0.84	A	N
	0.53	0.78	0.63	B	
	0.86	0.89	0.87	A	F
	0.63	0.56	0.59	B	

Mushroom

Table 4.3.1: Decision Tree

Sample size	Precision	Recall	F-measure	Binary labels	Fuzzy/Normal (F/N)
1	0.94	0.77	0.85	A	N
	0.80	0.95	0.87	B	
	0.91	0.77	0.83	A	F
	0.79	0.92	0.85	B	
2	0.96	0.76	0.85	A	N
	0.79	0.96	0.87	B	
	0.94	0.76	0.84	A	F
	0.79	0.95	0.86	B	
3	0.94	0.76	0.84	A	N
	0.79	0.95	0.86	B	
	0.93	0.76	0.84	A	F
	0.79	0.94	0.86	B	
4	0.94	0.76	0.84	A	N
	0.79	0.95	0.86	B	
	0.97	0.78	0.86	A	F
	0.81	0.97	0.88	B	
5	0.94	0.76	0.84	A	N
	0.79	0.95	0.86	B	
	0.94	0.77	0.85	A	F
	0.80	0.95	0.87	B	
6	0.92	0.76	0.83	A	N
	0.79	0.93	0.85	B	
	0.93	0.77	0.84	A	F
	0.80	0.94	0.86	B	
7	0.91	0.76	0.83	A	N
	0.78	0.92	0.85	B	
	0.94	0.77	0.85	A	F
	0.80	0.95	0.87	B	
8	0.94	0.75	0.84	A	N
	0.79	0.95	0.86	B	
	0.94	0.76	0.84	A	F
	0.79	0.95	0.86	B	
9	0.94	0.76	0.84	A	N
	0.79	0.95	0.86	B	
	0.96	0.77	0.85	A	F
	0.80	0.96	0.87	B	
10	0.92	0.75	0.83	A	N
	0.78	0.94	0.85	B	
	0.95	0.77	0.85	A	F
	0.80	0.95	0.87	B	

Table 4.3.2: Logistic Regression

Sample size	Precision	Recall	F-measure	Binary labels	Fuzzy/Normal (F/N)
1	1.00	0.83	0.91	A	N
	0.85	1.00	0.92	B	
	0.87	0.79	0.83	A	F
	0.80	0.88	0.84	B	
2	1.00	0.84	0.91	A	N
	0.86	1.00	0.92	B	
	0.87	0.84	0.86	A	F
	0.84	0.87	0.86	B	
3	1.00	0.85	0.92	A	N
	0.86	1.00	0.93	B	
	0.90	0.84	0.87	A	F
	0.85	0.90	0.87	B	
4	1.00	0.85	0.92	A	N
	0.86	1.00	0.92	B	
	0.92	0.84	0.88	A	F
	0.85	0.93	0.89	B	
5	1.00	0.84	0.91	A	N
	0.86	1.00	0.92	B	
	0.93	0.84	0.88	A	F
	0.85	0.94	0.89	B	
6	1.00	0.84	0.91	A	N
	0.86	1.00	0.92	B	
	0.94	0.84	0.89	A	F
	0.85	0.94	0.89	B	
7	1.00	0.85	0.92	A	N
	0.86	1.00	0.93	B	
	0.96	0.85	0.90	A	F
	0.86	0.96	0.90	B	
8	1.00	0.85	0.92	A	N
	0.86	1.00	0.93	B	
	0.95	0.84	0.89	A	F
	0.85	0.95	0.90	B	
9	1.00	0.85	0.92	A	N
	0.86	1.00	0.93	B	
	0.93	0.84	0.89	A	F
	0.85	0.94	0.89	B	
10	1.00	0.85	0.92	A	N
	0.86	1.00	0.93	B	
	0.94	0.84	0.89	A	F
	0.85	0.95	0.90	B	

Table 4.3.3: Naïve Bayes

Sample size	Precision	Recall	F-measure	Binary labels	Fuzzy/Normal (F/N)
1	0.82	0.76	0.79	A	N
	0.77	0.82	0.79	B	
	0.81	0.74	0.77	A	F
	0.75	0.82	0.78	B	
2	0.93	0.75	0.83	A	N
	0.78	0.94	0.85	B	
	0.94	0.73	0.82	A	F
	0.77	0.95	0.85	B	
3	0.93	0.74	0.83	A	N
	0.78	0.94	0.85	B	
	0.93	0.73	0.82	A	F
	0.77	0.94	0.85	B	
4	0.94	0.73	0.82	A	N
	0.77	0.95	0.85	B	
	0.93	0.71	0.80	A	F
	0.76	0.94	0.84	B	
5	0.96	0.73	0.83	A	N
	0.78	0.97	0.86	B	
	0.94	0.71	0.81	A	F
	0.76	0.95	0.84	B	
6	0.96	0.73	0.83	A	N
	0.77	0.97	0.86	B	
	0.94	0.71	0.81	A	F
	0.76	0.95	0.84	B	
7	0.96	0.73	0.83	A	N
	0.77	0.97	0.86	B	
	0.95	0.70	0.81	A	F
	0.76	0.96	0.85	B	
8	0.96	0.73	0.83	A	N
	0.77	0.97	0.86	B	
	0.95	0.70	0.81	A	F
	0.76	0.96	0.85	B	
9	1.00	0.85	0.92	A	N
	0.86	1.00	0.93	B	
	0.93	0.84	0.89	A	F
	0.85	0.94	0.89	B	
10	0.96	0.72	0.83	A	N
	0.77	0.97	0.86	B	
	0.94	0.71	0.81	A	F
	0.76	0.95	0.84	B	

Telecom Churn

Table 4.4.1: Decision Tree

Sample size	Precision	Recall	F-measure	Binary labels	Fuzzy/Normal (F/N)
1	0.88	0.90	0.89	A	N
	0.32	0.28	0.30	B	
	0.89	0.91	0.90	A	F
	0.36	0.31	0.33	B	
2	0.90	0.87	0.88	A	N
	0.34	0.39	0.37	B	
	0.89	0.89	0.89	A	F
	0.36	0.36	0.36	B	
3	0.89	0.82	0.85	A	N
	0.26	0.37	0.30	B	
	0.89	0.87	0.88	A	F
	0.34	0.38	0.36	B	
4	0.90	0.88	0.89	A	N
	0.35	0.40	0.38	B	
	0.90	0.89	0.90	A	F
	0.39	0.39	0.39	B	
5	0.90	0.86	0.88	A	N
	0.34	0.44	0.38	B	
	0.90	0.86	0.88	A	F
	0.34	0.44	0.38	B	
6	0.89	0.87	0.88	A	N
	0.33	0.38	0.35	B	
	0.90	0.90	0.90	A	F
	0.41	0.39	0.40	B	
7	0.90	0.87	0.88	A	N
	0.34	0.41	0.37	B	
	0.90	0.91	0.90	A	F
	0.41	0.38	0.39	B	
8	0.89	0.83	0.86	A	N
	0.28	0.40	0.33	B	
	0.90	0.85	0.87	A	F
	0.31	0.41	0.36	B	
9	0.89	0.84	0.87	A	N
	0.31	0.41	0.35	B	
	0.90	0.89	0.90	A	F
	0.39	0.42	0.40	B	
10	0.89	0.85	0.87	A	N
	0.31	0.40	0.35	B	
	0.90	0.88	0.89	A	F
	0.38	0.41	0.40	B	

Table 4.3.2: Logistic Regression

Sample size	Precision	Recall	F-measure	Binary labels	Fuzzy/Normal (F/N)
1	0.94	0.34	0.50	A	N
	0.18	0.87	0.30	B	
	0.88	0.96	0.92	A	F
	0.48	0.20	0.28	B	
2	0.91	0.39	0.55	A	N
	0.18	0.77	0.29	B	
	0.88	0.97	0.92	A	F
	0.47	0.18	0.26	B	
3	0.94	0.28	0.43	A	N
	0.17	0.89	0.29	B	
	0.88	0.97	0.92	A	F
	0.48	0.19	0.27	B	
4	0.97	0.17	0.29	A	N
	0.16	0.97	0.28	B	
	0.88	0.97	0.92	A	F
	0.49	0.19	0.28	B	
5	0.93	0.32	0.47	A	N
	0.18	0.86	0.29	B	
	0.88	0.97	0.92	A	F
	0.52	0.18	0.27	B	
6	0.91	0.31	0.46	A	N
	0.17	0.83	0.28	B	
	0.88	0.96	0.92	A	F
	0.47	0.19	0.27	B	
7	0.98	0.18	0.30	A	N
	0.17	0.97	0.28	B	
	0.88	0.97	0.92	A	F
	0.52	0.19	0.28	B	
8	0.98	0.17	0.28	A	N
	0.16	0.98	0.28	B	
	0.87	0.97	0.92	A	F
	0.51	0.17	0.26	B	
9	0.98	0.18	0.30	A	N
	0.17	0.97	0.28	B	
	0.88	0.97	0.92	A	F
	0.52	0.19	0.28	B	
10	0.95	0.21	0.34	A	N
	0.17	0.93	0.28	B	
	0.88	0.97	0.92	A	F
	0.49	0.19	0.28	B	

Table 4.3.3: Naïve Bayes

Sample size	Precision	Recall	F-measure	Binary labels	Fuzzy/Normal (F/N)
1	0.89	0.97	0.93	A	N
	0.66	0.32	0.43	B	
	0.91	0.97	0.94	A	F
	0.71	0.43	0.54	B	
2	0.89	0.98	0.93	A	N
	0.67	0.29	0.40	B	
	0.91	0.98	0.94	A	F
	0.75	0.43	0.55	B	
3	0.89	0.98	0.93	A	N
	0.68	0.28	0.40	B	
	0.91	0.98	0.94	A	F
	0.76	0.40	0.53	B	
4	0.89	0.98	0.93	A	N
	0.68	0.29	0.40	B	
	0.91	0.98	0.94	A	F
	0.78	0.42	0.55	B	
5	0.89	0.98	0.93	A	N
	0.69	0.28	0.40	B	
	0.91	0.98	0.94	A	F
	0.80	0.41	0.54	B	
6	0.89	0.98	0.93	A	N
	0.71	0.28	0.40	B	
	0.91	0.98	0.95	A	F
	0.81	0.43	0.57	B	
7	0.89	0.98	0.93	A	N
	0.70	0.28	0.40	B	
	0.91	0.98	0.95	A	F
	0.80	0.44	0.57	B	
8	0.89	0.98	0.93	A	N
	0.69	0.28	0.39	B	
	0.91	0.99	0.95	A	F
	0.83	0.42	0.56	B	
9	0.89	0.98	0.93	A	N
	0.70	0.28	0.40	B	
	0.91	0.98	0.95	A	F
	0.82	0.44	0.58	B	
10	0.89	0.98	0.93	A	N
	0.70	0.29	0.41	B	
	0.91	0.98	0.95	A	F
	0.81	0.43	0.56	B	