GENERAL APPROACH FOR CHURN PREDICTION WITH GENETIC ALGORITHM OPTIMIZED K-NEAREST NEIGHBOR FRAMEWORK

Tennakoon Mudiyanselage Nipunika Priyadarshani Tennakoon

(118231G)

Dissertation submitted in partial fulfillment of the requirements for the degree Master of Science

Department of Computer Science & Engineering

University of Moratuwa

Sri Lanka

June 2015

Declaration

I declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: Date:

Name:

The above candidate has carried out research for the Masters Dissertation under my supervision.

Signature of the supervisor: Date:

Name of the supervisor:

Abstract

Customer churn has become one of the most significant topics in today's business. It has become a major challenge for a business with the evolving market and low barriers to switch between the service providers. It has identified that, retaining the old customers is more profitable for a company than acquiring new customers. That motivates business personnel for churn prediction. Service providers can get necessary measures to retain their customers if they could gain prior knowledge on the probable churns in the customer base. But, churn prediction is considered a difficult task.

Various attempts have been made in predicting churn and churn related information. Different data mining techniques had been used in developing churn models. Regression analysis, decision tree based methods and neural network based methods were among the most commonly used techniques. The most successful models suffered from low interpretability, which is a main consideration in a churn model while some of the models were domain specific.

K nearest neighbor classifier is one of the best algorithms to be used in classifications. But it has been rarely used in churn prediction. Genetic algorithms are considered an optimization technique which could be used in optimizing performance of other algorithms. Genetic Algorithm Optimized K Nearest Neighbor (gaKnn) is a framework that has tested for its high accuracy. Hence, we developed a Tool based on the gaKnn framework which could be used for churn prediction. We also incorporated two voting mechanisms; Bayesian weights and class confidence weights (*ccw*) to weight the prediction in order to address misclassification issues occur due to class skew.

Acknowledgements

I sincerely acknowledge the work carried out by the WEKA community and publishing it as an open source code.

I would like to dedicate my sincere thank to my supervisor Dr. Amal Shehan Perera, Senior Lecturer, Department of Computer Science and Engineering for his dedicated support for the success of this research. This would not have become a success without your support from the initial stage of the research.

I would like to thank the entire academic and the non academic staff of the Department of Computer Science and Engineering for their kindness extended to me in every aspect.

Last but not least, I thank my family and all my friends who supported me for the success of this piece of work. Your support was so precious.

Table of Contents

Declaration ii

Abstract iii

Acknowledgements iv

List of abbreviations viii

Table of figures ix

List of tables x

CHAPTER 1 – INTRODUCTION 1

1. Introduction 1

1.1. Background for Churn Prediction 1

1.2. Objectives 2

1.3. Statement of the Problem 3

1.4. About the gaKnn Churn Prediction Tool 4

CHAPTER 2- LITERATURE REVIEW 6

2. Literature Review 6

2.1. Churn Prediction 6

2.1.1. Past attempts on churn prediction 7

2.1.2. Analyzing existing churn models 10

2.2. Analyzing the Data Mining Models Related to the Framework 12

2.2.1. K nearest neighbor algorithm (KNN) 12

2.2.2. Genetic algorithm (GA) 14

2.2.3. Genetic algorithm as an optimizing technique 15

2.2.4. Genetic algorithm optimized k nearest neighbor (gaKnn) 17

2.3. Genetic Algorithm Optimized K-Nearest Neighbor Classification Framework 18

2.4. Data Pre-Processing 21

2.5. Classification vs. Class Skew 22

2.5.1. Mixture models 27

2.5.2. Expectation maximization (EM) algorithm 28

2.5.3. Bayesian networks 28

2.5.4. K2 algorithm 28

2.6. Classifier Evaluation 30

2.6.1. ROC curve/ AUC 31

2.6.2. Lift charts 31

2.6.3. Precision recall (PR) curve 32

CHAPTER 3 DESIGN 33

3 Design 33

3.1 Overview of Design 33

3.2 Pre-Processing Steps 35

3.2.1. Outlier handling 35

3.2.2. Missing value handling 38

3.2.3. Data discretization 38

3.3 Distance Function and Similarity Calculation 39

3.4 Data Skew 39

3.4.1. Class confidence weights 40

3.4.2. Naïve Bayesian probability weights 43

3.5 Testing and Validation Options 44

3.6 Graphical User Interface (GUI) 45
3.7 Saving the Model for Re-Use 47
CHAPTER 4- IMPLEMENTATION 48
4 Implementation 48
4.1 Implementation Environment and Process 48
4.1.1. R language 48
4.1.2. Process implementation 49
CHAPTER 5- TESING AND EVALUATION 56
5 Testing and Evaluation 56
5.1 Testing the Tool 56
5.1.1 Test goals 56
5.1.2 Testing process and results 57
5.2 Evaluation 67
CHAPTER 6- CONCLUSION AND FUTURE WORKS 75
6 Conclusion and Future Works 75
6.1. Conclusion 75
6.2. Future Works 76
REFERENCES 77
APPENDIX A – User Manual 81
APPENDIX B – Source Code 90

List of abbreviations

Abbreviation	Description
AUC	Area Under Curve
CCW	Class Confidence Weight
CDR	Call Detail Record
CSV	Comma Separated Values
DMEL	Data Mining by Evolutionary Learning
FN	False Negative
FP	False Positive
GA	Genetic Algorithm
gaKnn	Genetic Algorithm Optimized K nearest neighbor
GUI	Graphical User Interface
JGAP	Java Genetic Algorithms Package
KNN	K nearest Neighbor
NN	Neural Network
PR	Precision- recall
ROC	Receiver Operating Characteristic
TN	True Negative
TP	True Positive

Table of figures

Figure 2.1-1 Churn prediction process [21] 7
Figure 2.2-1 K nearest neighbor 13
Figure 2.3-1 Overall design of the framework 20
Figure 2.5-1 Pseudo Code for K2 algorithm [29] 29
Figure 3.1-1 Design overview of the Tool 34
Figure 3.6-1Activity flow 46
Figure 4.1-1 Graphical View of Data 51
Figure 4.1-2 Interface to feed data 52
Figure 4.1-3 Data Preprocessing Window 52
Figure 4.1-4 Visualize Results 53
Figure 4.1-5 Prediction window 53
Figure 4.1-6 Churn Results 54
Figure 4.1-7 Accuracy Measurements 54
Figure 5.2-1 AUC for different weighted votes 69
Figure 5.2-2 Change of Lift value 69
Figure 5.2-3 Change of Recall 70
Figure 5.2-4 Performance - Dataset 1 73
Figure 5.2-5 Performance – Dataset 3 73
Figure A-1 First Interface 82
Figure A-2 Data selection 82
Figure A-3 Data pre-processor 83
Figure A-4 Predictor 85
Figure A-5 Result analyzer 87
Figure A-6 Fitness distribution 88
Figure A-7 Model accuracy 89

List of tables

Table 5.1-1Tested Data 58
Table 5.1-2 Results for sample drawn from dataset 1 59
Table 5.1-3 Results for sample drawn from dataset 2 59
Table 5.1-4 Results for sample drawn from dataset 3 60
Table 5.1-5 Comparison between different training options 61
Table 5.1-6 Comparison on different test options 62
Table 5.1-7 Comparison outlier detection/handling methods- with Tool 63
Table 5.1-8 Comparison outlier detection/handling methods- with WEKA 64
Table 5.1-9 Results of training phase 64
Table 5.1-10 Results of evaluation with test data 65
Table 5.1-11 Results of training phase 65
Table 5.1-12 Evaluated with test data 66
Table 5.1-13 Results for missing value handled data 66

CHAPTER 1 – INTRODUCTION

1. Introduction

This chapter gives an overview of the requirement for churn prediction. It also describes the scope of our research. At the end of the chapter it gives a brief description about the Tool we developed using the gaKnn framework.

1.1. Background for Churn Prediction

Churn prediction is one of the most emerging topics in the today's market all over the world. Churn is a term that is used in the business world to describe the loss of customers or clients. It is also termed as customer turnover, defection or attrition. Most of the time churn is identified in two categories, namely voluntary churn and involuntary churn. Voluntary churn is due to the decision taken by the customer to switch to another service provider. Involuntary churn may due to various reasons like customer's relocation to a long term care facility, death, or relocation to a distant location. Involuntary churn does not involve with the decision taken by a client to quit the current service provider. For some businesses they identify another category as partial customer churn. That category represents the customers who are becoming less profitable and not perceiving services as they did before.

In most financial services, customer retention plays a very important role. In business everything is money. Therefore customer churn is a burden to a company. By all means, customer retention is cheaper than acquiring a new customer. Because the company lost the revenue it gained from the churned customer as well as the investment on that customer. Therefore it is very useful for companies to know the probable churners and reasons for churn. So that they can make suitable arrangements to prevent churn from happening in future and also companies can prevent from deteriorating their good

customers. Apart from that, if it could accurately predict the probable churns and reasons for churn, companies can improve resource planning and resource allocation efficiently.

Mobile telecom service providers have a special concern over churn predictions since it is more probable for churn situations due to the low barriers it has from switching from one service to another.[1][7]

That is, churn prediction is a vital factor for financial services in operation. To present, several attempts have been made to address the problem of churn prediction. With the aim of predicting churn and reasons for churn, different churn models have been developed taking various factors in to consideration.

1.2. Objectives

Our objective of this research was to find a more general approach for churn prediction. Churn models that have been developed so far were customized to each data set and company. So, as the data set differs it had to go for a new model. If there was any tool which could be used regardless of the dataset being used and the company, it is cost effective in every aspect.

The Genetic Algorithm Optimized K-Nearest Neighbor Framework was proved to have high prediction accuracy for normal predictions. It was only a framework with the least amount of features including basic mechanisms for data representation, optimized data manipulation and components for K-Nearest Neighbor implementation. And it was not a complete final solution for all aspects of Genetic Algorithm Optimized K-Nearest Neighbor Algorithm. So, our aim was to improve this framework so that it could be used generally in the churn prediction processes.

In the meantime, we wished to preprocess data before predicting churn. The result we get from the model was to be output in a more meaningful way. Churn models require another important feature that most of the other models may not require. For a churn model to be more useful, it should have a good interpretation of the results it produces

[3][21]. Therefore useful information could be gained about churning of a particular customer. Hence, the user would be given a visualization of the results in graphical manner. Apart from that, it would allow the user to change certain features and check for churn results.

1.3. Statement of the Problem

During our study we focused on the voluntary churners and we limited our scope to predict the binary relation where a customer would churn or not churn. We did not focused on the partial churners directly where it identifies the most probable churners in the future.

When developing a churn model it needs to take into consideration the various factors that are inherent to churn datasets. Churn data has a large quality. That is, data has a number of representations about the relevant object. If it considers customer data, it may include customers' personal information, usage data, behavioral data etc. Churn datasets in general are biased. Because, it is very rare that a natural dataset has churners and non churners equally distributed. This biased nature is one of the important factors to be considered during classification.

When predicting churn, basic principle is to predict churn based on past data. Hence, the larger the dataset the more accurate results it gives. Therefore, whatever the model that is developed should be capable of handling large datasets.

Apart from that, according to the history of churn prediction, it is observed that most churn models are data specific. But the churn models need to be more convenient and applicable; it is wise to make it suitable for general set of data.

Different churn models have been developed, but their accuracy depends on the size of the data set and the type of data it is using.

Lazy learning is widely used is classification and prediction in data mining tasks. Lazy methods are known to be more accurate in predicting regardless of time it takes. This accuracy is due to the fact that it uses several hypotheses to form the approximation to the target function. Hence, k- nearest neighbor approach is widely used in classification. Genetic algorithms are based on ideas of natural evolution and are used to evaluate fitness of other algorithms.

It has identified that, for certain applications k-nearest neighbor incorporated with genetic algorithm works well. But, since we are to develop the Tool by improving the gaKnn framework, it was taken in to consideration the factors that are limitations of the two algorithms; KNN and GA. Classification with KNN is not favorable when the training data is skewed and GA is not a better option when feature space is large.

1.4. About the gaKnn Churn Prediction Tool

gaKnn churn prediction tool which was developed based on the gaKnn framework (herein after referred to as Tool) is a set of integrated solutions that come in one pack enabling a user to accomplish churn prediction task in a more convenient way.

The Tool consists of modules for data pre processing, data analyzing, prediction, and data and results visualization in relation to the prediction tasks.

Though the word prediction is used, the principles of the Tool are based on the techniques for classification of data. In real world, the word prediction is used regardless of its technical meaning in data mining practice. Hence, to make the general user more sensitive towards the Tool, it has introduced in that manner.

This Tool works mainly in two steps; optimizing phase and evaluation phase. During the optimizing phase, the model is optimized using known data with GA, which gives an attribute level weight to each attribute. At the evaluation stage, instance level weight was assigned to the model which can be selected by the user at his choice. The best model can be selected by evaluating the accuracy measures. During the prediction, unknown

dataset can then be used with the evaluated model to get the predictions. The predictions are presented in a user friendly manner which the user can easily make use of the results.

CHAPTER 2- LITERATURE REVIEW

2. Literature Review

Literature review is mainly focused on three directions. One is to review the past attempts on churn prediction and the second is to analyze the relevant data mining models. The third is to identify and analyze the expected challenges when developing the Tool. It further explains the previous work carried out with genetic algorithm optimized k nearest neighbor approach and also review the work carried out with the use of two algorithms separately.

2.1. Churn Prediction

For a Business it is vital to retain their existing customers due to various reasons. The foremost reason is to prevent the loss of revenue on that customer and at the same time acquiring a new customer is too costly because of the saturated market. Churns involve a company's brand, referral, social and image management aspect would adversely affect if not handled properly [22].

If a company can have a prior identification about the probable churners they can organize retention campaigns targeted towards those high risk customers. Churn prediction models have been introduced making use of several technologies and targeting towards different aspects of churn. A good churn model should have several characteristics. Churn model should be a good model in technical point of view, should provide good insight into determinants of churn and should be easily be adoptable by the business users. That model is even better if it could provide a better insight of the reasons for churn. For a churn model to be more useful, it should have a good interpretability which is not just the accuracy.

Regardless of the model being used, the churn prediction process could be explained as in Figure 2.1.1 [21].

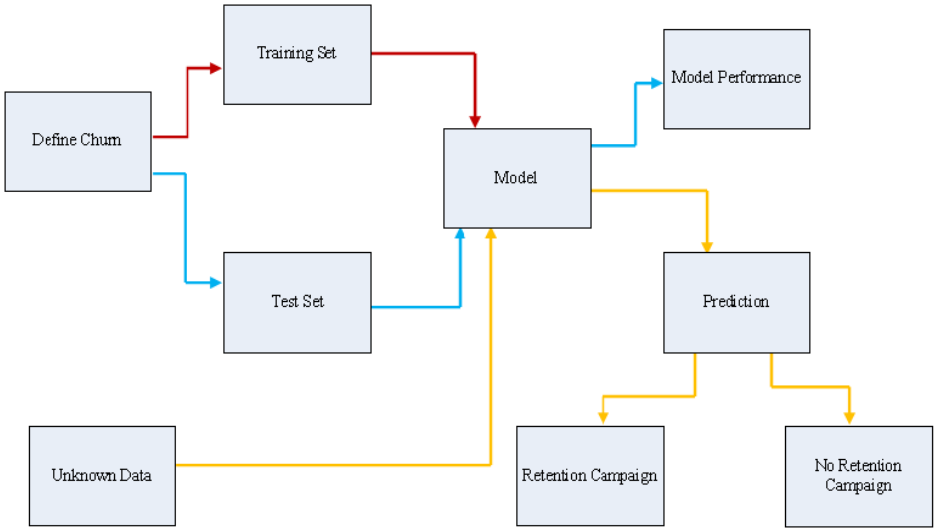

Figure 2.1.1 Churn prediction process [21]

2.1.1. Past attempts on churn prediction

Churn prediction has been one of main interests of Data Mining Researchers due to its value in various sectors. Therefore, there had been a number of attempts to predict churning customers as well as the reasons for churn. Because, those information can help companies in reducing their customer turn over.

This section gives an overview of some of the past attempts on predicting churn and the Section 2.1.2 explains about the models that have been used in such applications.

The Market Equations [9] have analyzed churn on a large automobile company in United States. They have come up with a neural network based approach for churn prediction. The expectation of the analysis was to identify the customer segments with diminished level of satisfaction with the company's product. They have come up with three main steps in detecting churn.

- Churn Trend Analysis- Aim was to understand trends in customer data
- Churn Profiling- Was to identify segments which are more prone to defect
- Churn Scoring and Segmentation- Assign each segment a churn score and with the use of a predictive model, find the most prone defectors

With the analysis of data, Market equations had tried to identify the factors for churn and reasons for a customer to leave or stay with a particular service provider and to find any demographic patterns or trends in customers.

According to Richter *et al.* [2], the most prevailing approach for churn prediction is to model individual customers and then derive the likelihood of churn using a predictive model. This approach focuses on identifying patterns that are uncommon to a particular customer which can relate for him to churn. For example, in relation to a mobile network, they monitor the number and length of calls by the customer, hold period etc. and assign a level of risk for uncommon patterns. But, it assumes that the customer will express his dissatisfaction prior to switching. They have suggested an approach which takes in to consideration the social factors. This approach first partition the network into a collection of small disjoint clusters, each representing a dense social group. Then with the use of a statistical model, it assigns a churn risk score to each group and later assigns an individual churn score to each subscriber based on the churn score of the group as well as personal characteristics.

The authors have claimed their new approach to be effective, but dataset they have tested on was small and they have not had access to any other information that may affect churn results like demographic details, individual mobile tariffs. But their approach claim to be giving accurate results even before any of the people has not churned in a social group.

And also they had come up with some conclusions from their research. First, as the prediction system gives the churn scores, companies work on preserving the customers who are most prone to churn. So to make it effective, churn prediction system should be

able to identify churners within its top predictions. Next, since only call data records (CDR) are used it has not needed to retrain the model as the data set varies. CDR represents date, time, duration, and location etc. of a phone call.

Junxiang[3], in their approach explain a survival analysis model for churn prediction where survival analysis is a statistical method of studying the occurrence and timing of events. In their study they have focused on the customer initiated churn and it has defined by cancel reason codes such as unacceptable call quality, more favorable competitor's pricing plan, misinformation given by sales, customer expectation not met, billing problem. A survival function and hazard function were used to describe the status of customer survival during the period of observation where survival function gives the probability of surviving beyond a certain time point *t* and hazard function describes the risk of event (in this case, customer churn) in an interval time after time *t*, conditional on the customer already survived to time *t.* They give two procedures for survival analysis:

1. LIFEREG -produces parametric regression models with censored survival data using maximum likelihood estimation
2. PHREG - is a semi-parametric regression analysis using partial likelihood estimation

By ranking the customers predicted survival probabilities in ascending order, the top two deciles capture 55-60% of churners and the top five deciles capture almost 90% of churners.

Most of the methods that are in use in relation to churn are mainly focused on predicting whether a customer would churn in the future. But when it comes to the industry, it is also important for the service provider to have knowledge of the customers who are most probable to churn. Au *et al.* [4] have proposed an approach to predict churn as well as the likelihood of the customer would churn. They have named their approach as Data Mining by Evolutionary Learning (DMEL) where it searches the possible rule space using an evolutionary approach.

Initial set of first order rules are generated using a probabilistic induction technique and based on the first order rules, the second order rules are generated. The third order rules are generated based on the second order rule set and generation of rules is continued until no more interesting rules in the current population can be identified. Likelihood of the predictions is estimated so that the customers can be ranked according to the likelihood to churn.

By representing the relationships among CDR, billing and demographics data of the customer, the rules are generated and those rules are used in predicting churn.

2.1.2. Analyzing existing churn models

Algorithms most commonly used in churn models are Decision Trees, logistic regression, support vector mechanism, Bayesian networks, survival analysis, self-organizing maps and relational classifiers. Decision Tree models are robust and have good interpretability but care should be given to class skew. Logistic regression models are well known and accepted for providing comparatively good results. Models using support vector mechanism accurate but are not easily interpreted. Survival analysis models are also good models which provide time to churn and interpretable results.

When analyzing the most common churn models in use, it can be observed that several prediction models have been used implementing churn prediction models. In a paper written by Lazarov and Capota[5], they have analyzed the most commonly used such models.

Regression analysis has been widely used in prediction tasks [3]. Typically, in a regression model, a standard error rate is calculated for each variable and the variables with most significance in respect to linear regressions for churn prediction are taken and construct the regression model. Since the prediction variable is binary (churner or non-

churner), logistic regression techniques are used in churn models. In contrast to logistic regression, in linear regression outcome may have infinite possible values. Regression analysis can determine a probability of churn as well as the likelihood of a customer being churned. These models do not have a better understandable method of interpreting the results.

Navie Bayesian models have also been used in churn prediction which uses the mathematics of Baye's theorem to make the predictions. [6] These models are simple and can perform faster. Navie Bayesian models have been tested with different types of data sets regardless of the data set is CDR, demographic data etc. But those have tested only for small data sets.

Decision tree based models have also been widely used in churn prediction history. [3][4] During the building phase of the tree, the data set is divided recursively until most of the records in each branch get an identical value. Branching is performed according to the value of the variable being tested. During the pruning phase, branches with noisy data are removed. Evaluation for a data set is performed by traversing through the tree until it reaches a leaf node. Then the record is assigned with the node label (churner or non-churner) for the test record. Decision trees are good in interpreting the results in a symbolic way. These models have high accuracy when the number of attributes in the dataset is less. Usage of these types of models is costly. Removal of a condition makes it to re-build the tree.

Neural networks based churn models are regarded the most successful churn models implemented so far [8][10][11]. In neural network based approaches, each variable is associated with a weight and combinations of weighted variables participate in the prediction task. During the prediction task, neural networks try to calculate the inputs and to output the probability of a particular customer is a churner. Hence, with neural network based models the likelihood of each classification made can also be determined.

Neural network based approaches need the data set to be larger and requires more time to give a reasonable weight to the predictor variables. These models do not give a good and easy understandable interpretation of the churn results as the model being too technical most of the time.

Apart from those models, in the approach suggested by Jadhav and Pawar [10], they have make use of a tool which is consisted of several classification models. In their analysis they have used back propagation mechanism which is derived from neural network models.

When considering churn models, apart from the accuracy it has to have a good interpretability of its results. For a company it is very vital that a model provide useful information which can support in focusing the customers who are more tend to churn.

Problem associated with most of the models is that those models do not have good interpretation of the churn results. The effectiveness of a model is also depending on the quality of the interpretation of the results giving the possible reasons for churn.

2.2. Analyzing the Data Mining Models Related to the Framework

This section focuses on the data mining models that have been used in the current framework, which we would enhance as a Tool for churn prediction. The framework has basically implemented with KNN and GA principles.

2.2.1. K nearest neighbor algorithm (KNN)

Nearest Neighbor Approach is considered one of the most popular algorithms for pattern recognition. In this approach, focusing on the nearest neighborhood, it compares a given test object with training objects that are similar to it. Each object is represented by a point in n-Dimensional space. The class label of the most closet object in the selected

neighborhood is given to the test object. Selection of the neighborhood is one of most critical task in this classification.

In k-nearest neighbor approach it searches for the k training objects that are closest to the unknown test object and create the k neighborhood. Depending on the sample variable type, distance to the unknown object is measured with Euclidean distance or with some other distant metrics. When k=1 the unknown object is assigned the class label of the closest object in the pattern space. When k is larger than one and need to return a real valued prediction, the unknown object is assigned the average value of the real valued labels associated with k nearest neighbors of the unknown object. For categorical attributes, it compares the corresponding object value with that of other objects. The most common class label is assigned to the unknown test object.

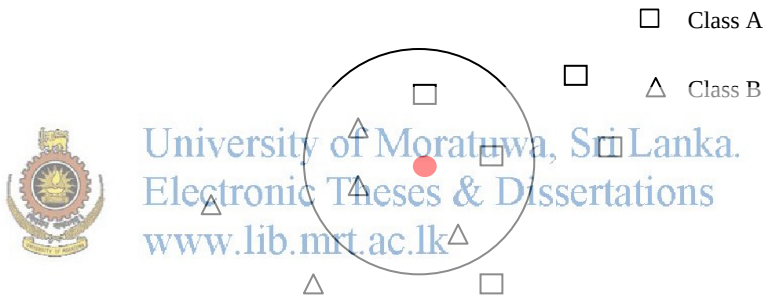

Figure 2.2-1 K nearest neighbor

Figure 2.2.1 illustrates the basic idea behind KNN approach. If it is to find the class label of the red colored dot, what KNN does is it predicts the class of that unknown dot based on the dots which are closer to it. In this case, it considers the most nearest k neighbors (in this example k=5) which are inside the circle. So here the most probable class label which can be given to the unknown dot would be B.

The goodness of classification highly depends on the k value. Selection of k depends on the data and generally for binary classification problems k being an odd number is helpful.

That is the unknown class gets the class label of the majority neighbors. But this method of majority voting can give erroneous results when the data set is highly skewed. Because the most frequent class label is more probable to be assigned to an unknown class. Several ways have been identified to overcome this issue. Most common methods are,

- Over sampling/ under sampling – where less class labeled objects are over sampled or more class labeled objects are under sampled
- Classification weighting – where each k nearest point is multiplied by the inverse of the distance from test point

2.2.2. Genetic algorithm (GA)

Genetic algorithms are based on ideas of natural evolution. The initial population is created with randomly generated rules where each of them is represented by a string of bits.[12] The new population is formed to consist with the fittest rules and their offsprings in the current population. The fitness of a rule is represented by its classification accuracy of training examples. And the offsprings are generated by crossover and mutation. New pairs of rules are generated in crossover by swapping substrings from pairs of rules in the current generation. In mutation, it inverts the randomly selected bits in a rule's string to generate a new child. This process continues until a population evolves and each rule in it satisfies a prescribed threshold.

If further explained, the operations evaluation, selection, crossover, and mutation are used to bias the search towards promising solutions.

GA is used in classification problems as well as in optimization problems. Apart from that GAs are used in evaluating the fitness of other algorithms.

2.2.3. Genetic algorithm as an optimizing technique

As it has stated before, GA has being tried using as an optimization mechanism by incorporating with other classification methods. According to Parag[8], they have tested GA with neural networks in predicting churn. In their approach GAs were used to learn connection weights of a neural network. GA uses the survival of the fittest heuristic to learn connection weights of a neural network. There they propose two GA based neural network (NN) models to predict customer churn in subscription of wireless services.

1. GA based NN model uses a cross entropy based criterion to predict customer churn

2. GA based NN model attempts to directly maximize the prediction accuracy of customer churn

In this approach, they have use the real-attribute representation for GA and then a fitness evaluation operator is applied to evaluate the fitness of each individual where the evaluation function can be either maximize the total number of correctly classified cases or maximize the log maximum likelihood hypothesis. Then the selection operator was applied to select the population members with higher fitness. It requires lower computational overhead and selects two parents based on the fitness probability where high fitness population members have higher chance of selection. Crossover and mutation operators were applied to these two parents with certain probabilities called crossover probability and mutation rate to generate two children. The process of selection, crossover and mutation were repeated so that the total number of children is equal to the population size. They have used arithmetic crossover which consists of generating children in such a way that every gene in a child is a convex combination (a linear combination of points with non-negative coefficients that sum up to 1) of genes from its two selected parents and has used a single gene mutation operator, where each gene in a child, with probability equal to mutation rate, is randomly changed with a random real number.

They have compared the results obtained from models which were based on GA with the results they obtained from other statistical models like z- score model, and have observed better performance in GA based approaches.

GA has being widely used in predictive modeling. Analysis carried out by Au *et al.* [4] states most common ways that GA has used in rule discovery. They have identified those on two broader categories.

1. Michigan approach
2. Pittsburgh approach

The techniques under Michigan approach have being identified with the following qualities.

- have modified the individual encoding method to use non binary representation
- do not encode the consequents of rules into the individuals
- use extended version of crossover and mutation operators suitable to their representations
- do not allow rules to be invoked as a result of the invocation of other rules
- define fitness functions in terms of some measures of classification performance
- can discover rules for a single class only

In the DMEL approach suggested by Au *et al.* [4], DMEL also make use of GAs. DMEL encodes a complete set of rules in one chromosome. Initial first order rules are generated with a probabilistic induction function and the higher order rules are based on the immediate lower order rules. Fitness function for DMEL is defined in terms of the probability that the attribute values of a record can be correctly determined with the encoded rules. With DMEL both positive and negative relationships could be found among attributes without any subjective input from the user.

2.2.4. Genetic algorithm optimized k nearest neighbor (gaKnn)

Though KNN is considered as one of the most popular method for pattern recognition, it has several limitations associated with the nature of the algorithm [13][14]. Calculation complexity is one limitation of KNN. In KNN it uses the whole sample in classification. Though it may improve the accuracy of prediction with large number of k neighbors, having noisy data in the sample can reduce the accuracy. At the same time, it has a high calculation complexity hence to find the k nearest neighbors it has to calculate all the similarities between samples. Another issue with traditional KNN is that the classification depends on the training set. Since the classifier is generated with the training sample, if there is a change in the dataset, it requires recalculation to generate the classifier. In traditional KNN it has no weight difference between samples. That is it does not carry any difference between samples according to the number of data the samples have. If it can give more weight to the near most neighbors it would improve the classification.

Researchers have proven that problems with KNN can be successfully addressed with GAs.

In the approach suggested by Suguna and Thanushkodi[13], GA is used in feature selection and the fitness value is calculated for N number of iterations. Unlike in traditional method which calculate the similarities between all the training and test samples and then choosing k-neighbors for classification, in this approach only k-neighbors are chosen at each iterations by using GA and the similarities are calculated, the test samples are classified with these neighbors and the accuracy is calculated.

Analoui and Amiri in their approach [15] have used GAs for feature reduction of nearest neighbor classifiers. There it reduces the dimension of features by performing simultaneous feature selection, extraction and classifier training using GA. GA optimizes a vector of feature weights, which are used to scale the individual features in the original pattern vectors. With feature extraction, the original features are transformed

to produce a new set of features. The number of features used in classifying can be finely reduced, hence reduce the cost of feature measurement, increase classifier efficiency, and allow higher classification accuracy.

Though it has used GA in optimizing the performance in KNN, all the attempts were application specific.

2.3. Genetic Algorithm Optimized K-Nearest Neighbor Classification Framework

Genetic algorithm optimized k-nearest neighbor (gaKnn) classification framework [16] [17] is the piece of work that we are interested in enhancing as a churn prediction Tool.

In gaKnn the data read in by the system is first evaluated with KNN with a predefined k value and selects the initial population to find the optimal parameters. The process of evaluating the optimal parameters is governed by the GA features in the framework. Then the optimal parameters along with their weights assigned to them are directed to evaluate the similarity.

Researchers have found several parameters that can increase the performance of KNN classification.

- Finding optimum value for K
- Finding the weight vector for attributes
- Finding voting power of neighbors
- Carrying out attribute selection
- Carrying out instance selection

During the work carried out by Dayaratne[16], the first two parameters were addressed. That framework consisted of features for data representation, data pre-processing, GA implementation, solution evaluation and similarity calculation.

Data representation module of this framework supports only the flat files nevertheless it can be extended to use other file formats as well. Flat files are data files that contain records with no structured relationships.

Data pre-processing module of the framework address only the incomplete data. But it is also extendable to other preprocessing aspects. Normalization is a data preprocessing method widely used for methods involving distance measurements so that those can be scaled to small specified ranges. When data normalization is required the framework supports the mechanism of dividing the complete column by the maximum value.

Figure 2.3.1 [16] depicts the overall design of the framework.

It has implemented the GA features with the JGAP (Java Genetic Algorithms Package) library which is a genetic algorithms and programming component provided as a java framework. JGAP [18] is highly modular so that it can be easily plugged-in customized genetic operators and other sub components.

The most important part in GA implementation is to find the best solution. To evaluate the best solutions a fitness function was defined. The solutions with higher fitness value are more likely to be chosen.

Similarity calculation is application specific. In this approach [16], similarity has measured as a distance function.

Framework mainly consisted of two parts.

1. KNN optimization
2. Classification with optimized KNN

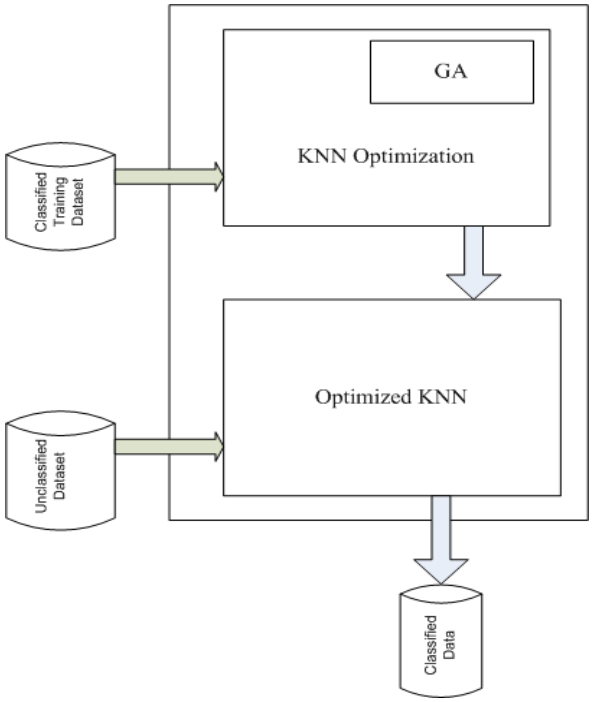

Figure 2.3-1 Overall design of the framework

KNN optimization

As data is read, attributes are identified. Then the read data is preprocessed.

KNN optimization module is consisted of three parts.

GA: implement with JGAP library

Solution evaluation module: evaluation of solutions made by GA (Fitness evaluation)

KNN algorithm: evaluate each solution

Classification with optimized KNN

As data is read to this module too, attributes are identified. Read data is pre processed

KNN algorithm is used to classify data.

As stated earlier, only the two optimization parameters, finding optimum value for K and finding the weight vector for attributes were considered.

gaKnn framework has been tested on several sizes of data sets and proven to be effective with its features of extensibility, adaptability, accuracy and scalability.

2.4. Data Pre-Processing

Presence of outliers in the training data is a major issue in classification. gaKnn framework had only two ways it could address the outliers and missing values. It had a solution which resolves the presence of missing values by replacing them with a default value (in this case it was NaN). The other way was to normalize the dataset and use it in the training process.

The presence of outliers can influence the model which may lead the classification to deviate from the actual results. Hence, outlier handling aims at reducing the influence of outliers on the training model. Outliers might be present in a dataset due to two main factors.

1. Due to errors in data which might be data entry errors
2. Due to inherent high variability of data

If it was determined that the outliers were due to errors in data, it could be removed from the training data. But if that was due to high variability in data, it should reduce the influence of those outliers on the model. Such solutions could be broadly categorized into two.

1 Transformation

2 Truncation

In transformation, it transforms the variable using an appropriate technique; for example take the logarithmic value of the variable. In truncation, it determines the highest (or lowest) values of the variable which are not detected as an outlier and raise (or reduce) the value of the outlier to that acceptable value(s).

The presence of missing values may also have adverse effect on the model. When dealing with skewed data it is very important that missing values are properly handled. If missing values are to be replaced with a default value, the value chosen may be vital for the model.

2.5. Classification vs. Class Skew

Classification involves a two-step process.

1. Learning Step/ Training Phase: where it builds a classifier based on a given set of classes or concepts
2. Evaluation Step/ Prediction Phase: where it classifies the unknown class labels

In the training phase it involves past data in order to build the classifier. Classification of the unknown sample is merely based on the built classifier. Hence, the data used in training phase plays an important role in the final result.

For better accuracy, the training data should be well balanced among all the classes and should well represent the application domain. But, when it analyses the data involved in churn calculations, those are highly imbalanced. Because, the two representative classes 'churner' and 'non churner' can rarely be having equal frequency in a churn dataset.

The imbalanced nature (skewness) of the dataset is one of the most challenging tasks in Data Mining. Researchers have found several mechanisms to overcome that issue. All these mechanisms comes under three main methods; data level, algorithmic level and combining or ensemble methods [23].

Data level methods include the re-sampling techniques. The most common two such techniques are undersampling and oversampling. In undersampling it drop out observations from the majority class and in oversampling, it duplicates the observations from the minority class to obtain a balanced set. But, oversampling may lead to over

fitting as it makes exact copies of the minority samples and undersampling maintain all the minority samples though it may discard useful majority samples.

Algorithmic level focuses on modifying the learning algorithms to address the impact of skewness. This may combine several learning algorithms. Ensemble methods include techniques which involve both data level and algorithmic level aspects.

Boosting and Bagging are two popular ensemble methods used with skewed data. Bagging stands for “Bootstrap Aggregating” and also called attribute bagging. It can be used with any learning algorithm. In bagging samples are drawn with replacement hence some observations may be repeated in another sample. Bagging may or may not improve the performance of a model; with 1-NN classifier bagging perform well while with stable KNN classifier it may mildly degrade the performance [24]. Boosting methods improve the performance of a classifier by reweighting the misclassified samples.

In general most classification algorithms have not designed to address the class skew. Therefore, measures have to be taken when dealing with such data.

Liu and Chawla [25] have proposed a quadratic mean based learning framework (QMLearn) which redefines the empirical risk function with concept of quadratic mean. They have applied that framework into logistic regression, SVM, linear regression, SVR, quantile regression and proven to get good results. They could get significantly better performance with the QMLearn on classification problems. It suggests that the extra computational cost spent on data sampling could be saved with the QMLearn and the weight vectors obtained from QMLearn were significantly better than those from existing methods for feature selection.

Akbani *et al.* [26] suggest a method for Support Vector Machines (SVM) to improve its results when dealing with skewed data. It presents a method which is a combination of two approaches. First it preprocess the data by oversampling or undersampling and then bias the classifier to get priority to the positive instances which could be done by

increasing the penalty associated with misclassifying the positive class relative to the negative class.

Liu and Chawla [27] have proposed a new algorithm oriented method for KNN weighting in order to handle skewed data. The proposed method introduces class confidence weights (*ccw*) which use the probability of attribute values given the class labels. The *ccw* on a training instance i was given as in Equation 2.5-1

$$\boldsymbol{w_i^{ccw} = p(x_i|y_i)} \qquad \textbf{Equation 2.5-1}$$

where x_i - attribute vector of instance i

y_i- class label of instance i

The *ccw* incorporated KNN classification rule could be given as in Equation 2.5-2

$$\boldsymbol{y_t^{'} = arg\,max_{c\in\{c_1,c_2\}} \sum_{x_i\in\emptyset(x_t)} I(y_i = c)\,.\,w_i^{ccw}} \qquad \textbf{Equation 2.5-2}$$

where $y_t^{'}$- predicted label

I(.) – an indicator function that returns 1 if its condition is true and 0 otherwise

Since the majority voting mechanism in traditional KNN could be given as in Equation 2.5-3, it has shown that, with the integration of *ccw* in that maximization method it could use conditional probabilities of classes given the values of k instances' feature vectors (Equation 2.5-4).

In traditional KNN,

$$y_t^{'} = \arg\max_{c\in\{c_1,c_2\}} \sum_{x_i\in\emptyset(x_t)} I(y_i = c)$$

$$=>\max\left\{\sum_{x_i\in\emptyset(x_t)} I(y_i = c_1)\ ,\sum_{x_i\in\emptyset(x_t)} I(y_i = c_2)\right\}$$

$$= \max\left\{\frac{\sum_{x_i\in\emptyset(x_t)} I(y_i=c_1)}{k}\ ,\frac{\sum_{x_i\in\emptyset(x_t)} I(y_i=c_2)}{k}\right\}$$

$$= \max\{\mathbf{p_t(c_1)}, \mathbf{p_t(c_2)}\} \quad \text{Equation 2.5-3}$$

In *ccw* integrated KNN,

$$y_t^{'} = \arg\max_{c\in\{c_1,c_2\}} \sum_{x_i\in\emptyset(x_t)} I(y_i = c)\,.\,p(x_i\,|y_i)$$

$$=>\max\left\{\frac{\sum_{x_i\in\emptyset(x_t)} I(y_i=c_1).p(x_i\,|y_i=c_1)}{k}, \frac{\sum_{x_i\in\emptyset(x_t)} I(y_i=c_2).p(x_i\,|y_i=c_2)}{k}\right\}$$

$$= \max\left\{p_t(c_1).p(x_i\,|y_i = c_1)_{x_i\in\emptyset(x_t)}, p_t(c_2).p(x_i\,|y_i = c_2)_{x_i\in\emptyset(x_t)}\right\}$$

$$= \max\{p_t(x_i, c_1)_{x_i\in\emptyset(x_t)}, p_t(x_i, c_2)_{x_i\in\emptyset(x_t)}\}$$

$$=\max\{\mathbf{p_t(c_1}\,|\mathbf{x_i})_{\mathbf{x_i}\in\emptyset(\mathbf{x_t})}, \boldsymbol{p_t(c_2}|\boldsymbol{x_i})_{\boldsymbol{x_i}\in\emptyset(\boldsymbol{x_t})}\} \quad \text{Equation 2.5-4}$$

where $p_t(c|x_i)_{x_i\in\emptyset(x_t)}$ - probability of x_t belonging to class c given the attribute values of all instances in $\emptyset(x_t)$.

Liu et al. justify the *ccw*-based KNN rule by implementing the same with likelihood ratio test. They have defined the null hypothesis (H_0) as "*x_t belonging to c_1*" and the alternative hypothesis (H_1) as "*x_t belonging to c_2*" assuming c_1 is the majority class. The first j neighbors of $\emptyset(x_t)$ were assumed to be from$c_1$and the other $k-j$ neighbors from c_2. Hence likelihood of H_0 (L_0) and H_1 (L_1) could be given as;

$$L_0 = \sum_{i=1}^{j} p(x_i|y_i = c_1)_{x_i\in\emptyset\,(x_t)}$$

$$L_1 = \sum_{i=j+1}^{k} p(x_i|y_i = c_2)_{x_i\in\emptyset\,(x_t)}$$

Then the likelihood ratio could be given as,

$$\Lambda = \frac{L_0}{L_1} = \frac{\sum_{i=1}^{j} p(x_i|y_i = c_1)_{x_i\in\emptyset\,(x_t)}}{\sum_{i=j+1}^{k} p(x_i|y_i = c_2)_{x_i\in\emptyset\,(x_t)}}$$

Since the majority class does not have higher priority than the minority in imbalanced data, $\Lambda = 1$ was taken as the rejection threshold. Hence they claim that use of *ccw*-based KNN rule in Equation 2.5-4 is equivalent to rejecting H_0(predict x_t to be c_2 when $\Lambda \leq 1$) or do not rejecting H_0(predictx_t to be c_1 when $\Lambda > 1$).

In order to estimate the *ccw* values, it has introduced two methods Mixture Modeling and Bayesian Networks (explained in section 2.5.1 and section 2.5.3). Mixture modeling deal with numeric features of the data while Bayesian networks handle nominal and categorical data.

In their approach to mixture modelling, they have assumed the training data is following a q-component finite mixture distribution with the probability density function (*pdf*);

$$\boldsymbol{p(x|\theta) = \sum_{m=1}^{q} \alpha_m \, p(x|\theta_m)} \qquad \textbf{Equation 2.5-5}$$

where, x - sample of training data whose pdf is demanded

α_m - mixing probabilities

θ_m - m^{th} Component

$\hat{\theta} = \{\theta_1, \ldots, \theta_q, \alpha_1, \ldots, \alpha_q\}$ is the complete set of parameters specifying the mixture model

Given training data Ω, the log-likelihood of a q-component mixture distribution was given as

$$\boldsymbol{log\, p(\Omega|\hat{\theta}) = log \prod_{i=1}^{n} p(x_i|\hat{\theta}) = \sum_{i=1}^{n} log \sum_{m=1}^{q} p(x_i|\theta_m)}$$

Using the Expectation Maximization Algorithm (explained in section 2.5.2) they have solved the Maximum Likelihood estimate($\hat{\theta}_{ML}$),

$$\boldsymbol{\hat{\theta}_{ML} = arg\, max_{\theta}\, log\ p(\Omega|\theta)} \qquad \textbf{Equation 2.5-6}$$

to find the $\widehat{\theta}$. The estimated $\widehat{\theta}$ has been applied in Equation 2.5-5 to calculate the pdf of each instance in the training data which would be taken as the *ccw* value of each instance.

In their approach of Bayesian Network, they have learnt and built a Directed Acyclic Graph (DAG) using the K2 search algorithm (explained in section 2.5.4).Learning the Bayesian Network has done in two steps;

i. Build the DAG over Ω

ii. Learn a set of conditional probability tables $\{p(\omega|pa(\omega)), \omega\epsilon\ \Omega\}$ where $pa(\omega)$ is the set of parents of ω in the ΩDAG

Then the joint probability distribution over Ω was calculated using the Equation 2.5-7

$$\boldsymbol{p(\Omega) = \prod_{i=1}^{d+1} p(\omega_i|pa(\omega_i))} \qquad \textbf{Equation 2.5-7}$$

Hence the *ccw* value of the training instance i was obtained from the Equation 2.5-8

$$\boldsymbol{w_i^{ccw} = p(x_i|y_i)\alpha \frac{p(\Omega)}{p(y_i)}} \qquad \textbf{Equation 2.5-8}$$

where $p(y_i)$is the proportion of class $y_i$among the entire training data

They have analysed the *ccw*–based KNN rules with datasets of different nature and have compared the results with several other classification techniques. It has shown that in most of the instances *ccw*-based approach performed well.

2.5.1. Mixture models

Mixture model is a statistical model which is generated from several different types of distributions. In a mixture model, the parameters of the overall distribution could be derived from the parameters of each distribution in the mixture.

In a mixture model, the probability density function of an instance could be denoted as in Equation 2.5-5. $\boldsymbol{\alpha_m}$ is the mixing probability which sums up to one. $\boldsymbol{p(x|\theta_m)}$ is the component density functions parameterized by the collection of model parameters $\hat{\theta}$.

Each instance is assumed to be from one of the q components with mean μ_q and variance σ_q. EM algorithms are used in estimating the mean and variance.

2.5.2. Expectation maximization (EM) algorithm

EM algorithm is extensively used in parameter estimation of mixture models and iterates in two alternate steps namely Expectation (E) step and Maximization (M) step. It estimates the maximum likelihood of the model parameters of a distribution. Maximum likelihood estimate $\widehat{\boldsymbol{\theta}}_{\boldsymbol{ML}}$(Equation 2.5.6) of a q-component mixture distribution with training data Ω, could be determined with EM algorithm. In the E-step expected values are calculated for the model parameters and in the M-step the results are maximized. The maximum values obtained during the M-step are assigned for the parameters.

2.5.3. Bayesian networks

Bayesian networks are also named as belief networks, Bayesian belief networks and probability networks. Bayesian network is defined by a DAG and a set of conditional probability tables (CPTs). In a DAG, each node represents a random variable and an arc represents a probabilistic dependence. For one variable it has one CPT depicting all the probabilistic dependencies it has to node connected by arcs.

DAG could be built with the use of a search algorithm and as the DAG is completed the learning process should be done. Learning a Bayesian network involves learning the CPTs for each node in the DAG.

2.5.4. K2 algorithm

K2 is a search algorithm which commonly used in leaning Bayesian Networks. Given a set of instances, this algorithm heuristically searches for the most probable belief network structure.

Pseudo code of the K2 algorithm is explained in Figure 2.5.1 [29]

Where,

π_i- set of parents of node x_i

```
1.  procedure K2;
2.  {Input: A set of n nodes, an ordering on the nodes, an upper bound u on the
3.           number of parents a node may have, and a database D containing m cases.}
4.  {Output: For each node, a printout of the parents of the node.}
5.  for i:= 1 to n do
6.      π_i := ∅;
7.      P_old := f(i, π_i); {This function is computed using Equation 20.}
8.      OKToProceed := true;
9.      While OKToProceed and |π_i| < u do
10.         let z be the node in Pred(x_i) - π_i that maximizes f(i, π_i ∪ {z});
11.         P_new := f(i, π_i ∪ {z});
12.         if P_new > P_old then
13.             P_old := P_new;
14.             π_i := π_i ∪ {z};
15.         else OKToProceed := false;
16.     end {while};
17.     write('Node: ', x_i, ' Parent of x_i: ',π_i);
18. end {for};
19. end {K2};
```

Equation 20 is included below:

$$f(i,\pi_i) = \prod_{j=1}^{q_i} \frac{(r_i-1)!}{(N_{ij}+r_i-1)!} \prod_{k=1}^{r_i} \alpha_{ijk}!$$

Figure 2.5-1 Pseudo Code for K2 algorithm [29]

$q_i = |\emptyset_i|$

$\emptyset_i$ – list of all possible instantiations of the parents of x_i in database D. That is, if $p_1, \dots, p_s$ are the parents of x_i then $\emptyset_i$ is the Cartesian product $\left\{v_1^{p_1}, \dots, v_{r_{p_1}}^{p_1}\right\} X \dots X \left\{v_1^{p_s}, \dots, v_{r_{p_s}}^{p_s}\right\}$ of all the possible values of attributes p_1 through p_s

$r_i = |V_i|$

V_i - list of all possible values of the attribute x_i

α_{ijk} - number of cases (i.e. instances) in D in which the attribute x_i is instantiated with its k^{th} value, and the parents of x_i in π_i are instantiated with the j^{th} instantiation in $\emptyset_i$

$N_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}$. That is, the number of instances in the database in which the parents of x_i in π_i are instantiated with the j^{th} instantiation in$\emptyset_i$.

The informal intuition here is that $f(i, \pi_i)$ is the probability of the database D given that the parents of x_i are π_i.

2.6. Classifier Evaluation

According to Parag[8], in comparing the classifiers they have suggested several criteria which include

1. prediction accuracy
2. top 10% deciles lift and
3. area under receiver operating characteristics (ROC) curve (AUC)

The accuracy is measured as either the total number of correct classifications or percentage of correct classifications. But, it may not be a good performance metric when the data sets are biased in that they have uneven distribution of examples belonging to any one particular class. Sensitivity and specificity are useful criteria when datasets are biased. When it is dealing with sensitivity and specificity, accuracy was predicted as follows.

Assume a binary classification problem with two classes, positives and negatives.

TP= correct classification belonging to the positive class

TN= correct classification belonging to the negative class

P=number of examples belonging to the positive class

N=number of examples belonging to the negative class

sensitivity=TP/ P

specificity=TN/N

accuracy=sensitivity*P/(P+N) + specificity*N/(P+N)

In churn predictions sensitivity is very important. Because identifying a non-churning customer as a churning customer is acceptable but identifying a churning customer as a non-churning customer is risky.

2.6.1. ROC curve/ AUC

ROC is drawn by plotting sensitivity (true positive rate: TPR) values on y-axis and false positive rate (FPR) (1-specificity) on x-axis. Each prediction result represents one point in the ROC space. ROC curve requires classifiers to generate continuous value attributes. If a classifier generates discrete output then ROC curve cannot be drawn, and only sensitivity and specificity ratios can be reported. Area under ROC curve is a measure of quality of the classification models. For a random classifier has an area under the curve of 0.5; AUC for a perfect classifier is equal to 1. The optimum curve is to be towards the upper left hand corner of the plot.

Top 10% decile lift performance metric has also been used to evaluate classifiers in churn studies. The higher value of top 10% decile lift is a hallmark of a good classifier. Decile lift measures the increase in the proportion of the labels of the 10% nearest neighbors of an item having the same class as the item over the proportion of the class across all items, except the tested point.

2.6.2. Lift charts

The ratio between the results from a model at the presence and in the absence of the model is denoted by the lift [31]. Its graphical representation is given by the lift chart. The lift chart is plotted with lift against Rate of positive predictions (RPP) and the higher the area under curve, the better the model is.

2.6.3. Precision recall (PR) curve

PR curve is an alternative to the ROC curve which is used when the differences in algorithms are not apparent. While ROC is plotted TPR against FPR, PR curves are plotted with Recall against Precision.

$$\text{Precision} = TP / (TP+FP)$$

$$\text{Recall} = \text{TPR} = TP/ (TP+FN)$$

Therefore, precision is also called Positive Predictive Value (PPV) and recall is also called sensitivity. Even though ROC curves are commonly used in presenting results for classification problems, it has observed that PR curve gives more useful information regarding the model when the datasets are highly skewed [30] and classification is binary [40]. Further it [30] has shown that an algorithm that optimizes the area under the ROC curve is not guaranteed to optimize the area under the PR curve. For different classifiers with same ROC curve, it can have different PR curves due to its high sensitivity towards the class skew. It was said that AUC of 0.5 of PR curve, shows a considerably good model.

CHAPTER 3 – DESIGN

3 Design

This chapter describes how the Tool was designed to meet the identified requirements. The chapter gives an overview to the design and illustrates each module separately.

3.1 Overview of Design

Our main objective was to customize the general purpose gaKnn framework as a comprehensive tool which could be used in churn prediction. As it was discussed in the previous Chapter, a classification task was to be carried out by the new tool mainly based on the K nearest neighbor and Genetic Algorithm concepts. The framework was accordingly designed to meet the principles of churn models while preserving the key concepts of KNN and GA.

We introduced new pre-processing steps to handle outliers and missing values. It also contains a module for data discretization.

We introduced an improvement to the existing similarity measurement to cater different data types specifically.

In order to address the class skew which is inherent to churn datasets, the approach suggested by Liu and Chawla [27] was incorporated in the tool. In another point of view, it added a weight to the vote for the probable class label of each instance. The idea of instance vote weighting is to improve the possibility of selecting as the class label providing more weight to the most probable class.

Apart from the later said approach, we suggest weighting instance votes with Naïve Bayesian Probabilities. Bayesian classifiers have proved to be more accurate theoretically but due to practical issues which are of major assumptions such as class conditional independence, Bayesian classification has led to reduced accuracy. But it was noted that Bayesian classifiers have proved to be useful in justifying other classifiers.

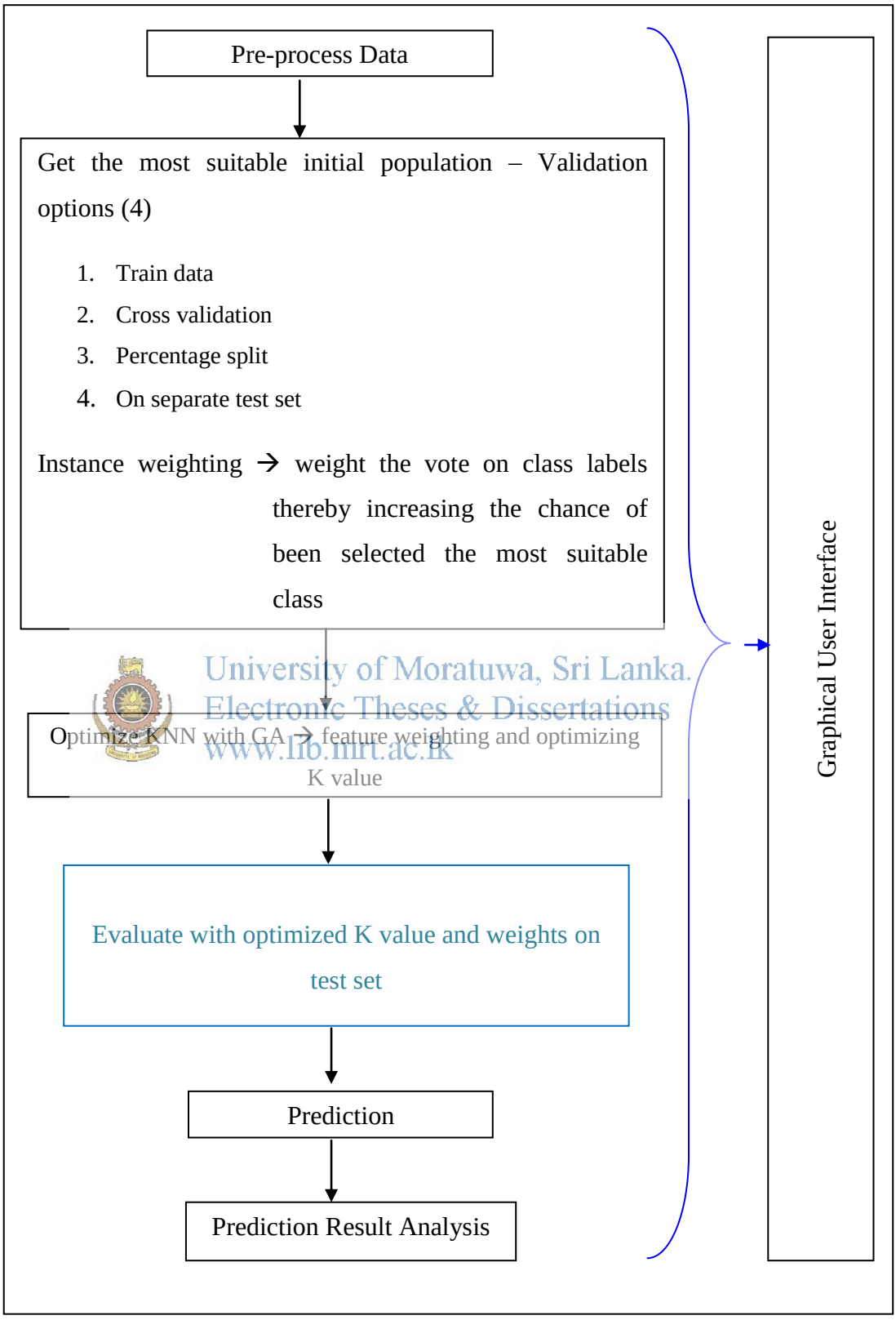

Figure 3.1-1 Design overview of the Tool

We introduce Naïve Bayesian Probabilities as votes to each individual instance. We expect to strengthen the probability of selecting the class label predicted by gaKnn with the Bayesian vote.

In the framework, validation was performed by splitting the training data in to portions of 2/3 as the train set and 1/3 as the test set. Similar to the methods used in WEKA[20], to improve the evaluation criterion we designed to add three other options for validation; to use the train set itself, use cross validation and to use a separate test set.

One of the most important aspects of a churn prediction tool is the interpretability of the data and its results. Hence, we designed a graphical user interface (GUI) of which the user was provided with the most important features of a churn prediction tool. It provides the user, the ability to visualize the data in terms of its distribution, presence of outliers and a statistical overview. The tool also provides the facility to visualize the churn results along with the prediction accuracies.

3.2 Pre-Processing Steps

It was observed that more pre-processing steps were required to streamline the current framework. Hence we designed methods to handle outliers and missing values.

3.2.1. Outlier handling

For a given dataset, certain few data points could be identified as outliers claiming (most often) that those data points fall beyond a legitimate range. If classification is performed by training a model with datasets which outliers are present, it is very often probable that the model compromise its accuracy, because of the influence of outliers on the other training data. Therefore, regardless whether those outliers are due to errors in data it may remove before classification. Because removal of such data being counted to few data points among the whole dataset and would not make a significant impact on the size representation of data.

As it was shown in the work carried out by Osborne and Overbay [28],removal of the outliers has a strong effect on the classification result. Therefore the two methods,

removal and truncation were used in designing strategies to handle outliers. The removal method was designed to remove the outliers which fell beyond some accepted extreme values and the truncation method was designed to replace the outliers with the accepted extreme values. In both the cases, accepted extreme values refer to the boundary values which are not identified as outliers.

A number of attempts have been made to detect and handle outliers and extreme values with different approaches. As a common practice in most of the outlier handlers, the values falling $1.5 \times IQR$ beyond the third quartile (Q_3) and below the first quartile (Q_1) were considered outliers;

where $Inter\ Quartile\ Range\ (IQR) = Q_3 - Q_1$

As stated by Songwon [32], work carried out by Tukey had suggested a similar method that can detect outliers. In his method he had introduced the concepts of inner fence and outer fence (box plot concept). Inner fences lie at a distance $1.5 \times IQR$ below Q_1 and $1.5 \times IQR$ above Q_3 while outer fences lie at a distance $3 \times IQR$ below Q_1 and $3 \times IQR$ above Q_3. A value fell between the inner and outer fences was considered a possible outlier and a value fell beyond the outer fence which is an extreme value was considered a probable outlier.

WEKA machine learning tool uses a similar approach to detect and handle outliers. It has defined lower and upper outlier thresholds as well as lower and upper extreme value thresholds.

$$lower\ outlier\ threshold = Q_1 - 3 \times IQR$$

$$upper\ outlier\ threshold = Q_3 + 3 \times IQR$$

$$lower\ extreme\ value\ threshold = Q_1 - 3 \times 2 \times IQR$$

$$upper\ extreme\ value\ threshold = Q_3 + 3 \times 2 \times IQR$$

Based on the technique in WEKA, we designed inner fences and outer fences to detect outlier and extreme value thresholds. In one of our methods, we designed to remove the values falling beyond and below the upper extreme value threshold and lower extreme value threshold respectively. In the other method it was designed to replace the values falling beyond the upper extreme value threshold with the upper outlier threshold value and the values falling below the lower extreme value threshold with lower outlier threshold value.

In the meantime, it was stated that [32] [33], those outlier detection methods might not be very suitable when data is skewed. Hubert and Vandervieren [33] suggest a method to improve standard boxplot to detect outliers in skewed data. In their adjusted boxplot they have defined the fences to be asymmetric around the box. They have used the concept of MedCouple (MC) [34] to define the new thresholds.

Given a dataset $X_n = \{x_1, x_2, \dots, x_n\}$ from a univariate distribution where it is sorted as $x_1 \leq x_2 \leq \cdots \leq x_n$, MC of the data could be defined as,

$$MC(x_1, x_2, \dots, x_n) = med_k \frac{(x_j - med_k) - (med_k - x_i)}{x_j - x_i}$$

Where med_k- median of X_n

$$x_i \leq med_k \leq x_j; \qquad x_i \neq x_j$$

The value of MC should lie between -1 and 1($-1 \leq MC \leq 1$)

when, $MC = 0$; distribution is symmetric

$MC > 0$; distribution is skewed right

$MC < 0$; distribution is skewed left

If $MC \geq 0$

lower outlier threshold: $Q_1 - 1.5 \times e^{-3.5MC} \times IQR$

upper outlier threshold: $Q_3 + 1.5 \times e^{4MC} \times IQR$

If $MC \leq 0$

lower outlier threshold: $Q_1 - 1.5 \times e^{-4MC} \times IQR$

upper outlier threshold: $Q_3 + 1.5 \times e^{3.5MC} \times IQR$

When $MC = 0$, it produces the standard box plot. In both the other cases, values that fell outside the threshold values were considered outliers.

We incorporated the same principles in our tool and based on the new detection criterion, outliers were handled by replacing and removing as it was done in the standard way.

3.2.2. Missing value handling

The missing value handler of the gaKnn framework replaced the missing values with the default value NaN. Replacement of all the missing values belonging to any data type with common NaN, led to other classification errors. Hence we introduced a missing value filter to handle the missing values. From that filter, instances with missing values were filtered out.

3.2.3. Data discretization

We decided that the theory introduced in the work carried out by Liu and Chawla [27] to be used in our tool (Section 3.4). In order to build and learn the DAG for Bayesian Network estimations, data required to be discrete. Experiments have shown that Naïve Bayesian classifiers work well with discretized features. Therefore a function was designed to discretize the data.

In data mining context, discretization refers to converting the continuous features into discrete features. Continuous values are reduced to ranges and each instance is given the label of each such range. Discretization could broadly categorize as supervised and

unsupervised. In supervised discretization, it uses information about the class labels while in unsupervised discretization, it does not. Binning, histogram analysis, entropy-based discretization are some of the popular discretizing mechanisms.

Equal width binning and equal frequency binning are two unsupervised binning methods. In equal width binning, it divides the data into intervals of equal size and in equal frequency binning, it divides the data into groups each of which contain equal number of values. Binning is the simplest method for discretization and in our tool we incorporated equal width binning as the discretization mechanism.

3.3 Distance Function and Similarity Calculation

Real world data are highly diverse hence can contain various types of data (numeric, nominal, string etc.). It was observed that the gaKnn framework was not well customized for different data types in similarity calculations. It addressed only nominal data separately while all the other variable types were considered in one category. Hence it was designed to streamline the similarity calculation method so that it could address more types of data separately.

Distance functions were designed for nominal, numerical attributes separately and a default function was designed to manage other data types. Decision was taken to address numeric data separately since, in a real world dataset most common types of data represent the two types nominal and numeric.

Similarity between two instances x_i and x_j of attribute n was calculated as;

$$sim(x_i, x_j) = \frac{1}{dist(x_i, x_j)}$$

3.4 Data Skew

Data skew is a major issue in classification. We introduced two instance weighting methods to address the biasing issue.

3.4.1. Class confidence weights

The concept suggested by Liu and Chawla [27] was incorporated in to our tool. The theory and concept was built as follows.

As it was stated in Section 2.5.1, mixture modeling was used to calculate *ccw* values of numeric data while Bayesian Networks were used to get *ccw* values of nominal data.

It was considered that the datasets which the tool has to handle follow a Gaussian Mixture Distribution. Then, to obtain the *ccw* values of each instance, *pdf* of the distribution is to be calculated. The *pdf* of the mixture distribution could be denoted as in Equation 2.5-5.

$$\boldsymbol{p(x|\theta) = \sum_{m=1}^{q} \alpha_m \, p(x|\theta_m)}$$

Where q - number of components

x - sample of training data

α_m - Mixing probabilities

$$\widehat{\theta} = \{\theta_1, \dots, \theta_q, \alpha_1, \dots, \alpha_q\}$$

Gaussian distribution of data could be given as in Equation 3.4-1

$$\boldsymbol{p(x|c_i) = \prod_{k=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_{c_i}} e^{\frac{-(x_k - \mu_{c_i})^2}{2\sigma_{c_i}^2}}} \qquad \textbf{Equation 3.4-1}$$

Where $\boldsymbol{c_i}$ – value of the class i

$\boldsymbol{\sigma_{c_i}}$ – standard deviation of class i

$\boldsymbol{\mu_{c_i}}$- mean of class i

$\boldsymbol{x_k}$ - value of the k^{th} attribute

In churn prediction context, we consider only the possibilities; churning and non-churning. Therefore, it is a two class problem. In applying the mixture model concept, we considered it a 2-component mixture. In this text, the 2 components have been numbered '1' and '2'. We consider that the parameters of component 1 category represent the churners and the parameters of component 2 category represent the non-churners.

Therefore, q = 2 and Equation 2.5-5 could be written as,

$$\boldsymbol{p(x|\theta) = \sum_{m=1}^{2} \alpha_m\, p(x|\theta_m)} \qquad \textbf{Equation 3.4-2}$$

$$= \alpha_1 p(x|\theta_m) + \alpha_2 p(x|\theta_m)$$

$$= \alpha_1 \left[\prod_{k=1}^{p} \frac{1}{\sqrt{2\pi}\sigma_{\theta_1}} e^{\frac{-(x_k-\mu_{\theta_1})^2}{2\sigma_{\theta_1}^2}} \right] + \alpha_2 \left[\prod_{k=1}^{p} \frac{1}{\sqrt{2\pi}\sigma_{\theta_2}} e^{\frac{-(x_k-\mu_{\theta_2})^2}{2\sigma_{\theta_2}^2}} \right]$$

p – number of attributes (features) of the train sample

Equation 3.4-2 could be applied to all the instances in the training data through all the p attributes.

The log-likelihood of a q-component mixture could be stated as;

$$\log p(\Omega|\hat{\theta}) = \log \prod_{i=1}^{n} p(x_i|\hat{\theta})$$

$$= \sum_{i=1}^{n} \log \sum_{m=1}^{q} \alpha_m p(x_i|\theta_m)$$

Therefore, for 2-component mixture,

$$\log p(\Omega|\hat{\theta}) = \sum_{i=1}^{n} \log \sum_{m=1}^{2} \alpha_m p(x_i|\theta_m)$$

$$= \sum_{i=1}^{n} \log[\alpha_1 p(x_i|\theta_1) + \alpha_2 p(x_i|\theta_2)]$$

Where n–number of instances in the train data

The log-likelihood was calculated using the EM algorithm.

E- Step:

During the E-step, the expected values (γ_q^i) are calculated for each $q = 2$ components.

Expected value of instance i when the class label is one, could be given as in Equation 3.4-3

$$\boldsymbol{\gamma_1^i = \frac{\alpha_1 p(x_i|\theta_1)}{\sum_{m=1}^{k} \alpha_m p(x_i|\theta_1)}} \quad \textbf{Equation 3.4-3}$$

$$= \frac{\alpha_1 p(x_i|\theta_1)}{\alpha_1 p(x_i|\theta_1) + \alpha_2 p(x_i|\theta_1)}$$

Where θ_1 represent the mean and variance of the respective attribute given the class label one

M-step:

During the M-step, the mean and variance were updated with weighted means and variances.

$$\boldsymbol{\mu_1 = \frac{\sum_{i=1}^{n} x_i \gamma_1^i}{\sum_{i=1}^{n} \gamma_1^i}} \quad \textbf{Equation 3.4-4}$$

$$\boldsymbol{\sigma_1^2 = \frac{\sum_{i=1}^{n} \gamma_1^i (x_i - \mu_1)^2}{\sum_{i=1}^{n} \gamma_1^2}} \quad \textbf{Equation 3.4-5}$$

$$\boldsymbol{\alpha_1 = \frac{\sum_{i=1}^{n} \gamma_1^i}{n}} \quad \textbf{Equation 3.4-6}$$

E-step and M-step has to be repeated for each instance in the train dataset, until γ_1^i is maximized for each instance.

The results obtained were to be substituted in Equation 3.4-2 in order to get the ccw_i of each instance.

Bayesian network was designed similar to that in WEKA, for the estimation of conditional probability values of nominal attributes. Then according to the breakdown in Section 2.5, the ccw_i values were calculated.

3.4.2. Naïve Bayesian probability weights

We designed naïve Bayesian probabilities to be added to the predicting vote in order to increase the voting power of the neighbors. Class conditional probabilities of each instance would be calculated based on the relevant prior probabilities obtained from the training set. It should be noted that we do not predict the class label with naïve Bayesian approach. We supposed to take only the calculated probability value which we would consider as a vote for the prediction made by GA optimized KNN classifier.

With naïve Bayesian, we could calculate posteriori probabilities of a variable, based on the prior probability of the given hypothesis.

To reduce the complexity of the calculation which increases with the number of attributes, class conditional independence was assumed. Thus, the posteriori probability could be given as in Equation 3.4-7

$$\boldsymbol{p(x|c_i) = \prod_{k=1}^{n} p(x_k|c_i)} \qquad \textbf{Equation 3.4-7}$$

Where $\boldsymbol{x}$–an instance of dataset

i–class label

$\boldsymbol{n}$ – number of attributes

The class conditional probabilities are to be computed, also considering the types of each attribute. If the attribute k is categorical,

$$\boldsymbol{p(x_k|c_i)} = \frac{number\ of\ instances\ of\ class\ c_i having\ the\ attribute\ value\ x_k}{number\ of\ instances\ of\ class\ c_i}$$

If the attribute k is continuous valued then, assuming the Gaussian distribution with mean μ and standard deviationσ, $\boldsymbol{p(x_k|c_i)}$could be defined as

$$\boldsymbol{p(x_k|c_i) = g(x_k, \mu_{c_i}, \sigma_{c_i})} \qquad \textbf{Equation 3.4-8}$$

Where$\boldsymbol{g\left(x_k, \mu_{c_i}, \sigma_{c_i}\right) = \frac{1}{\sqrt{2\pi}\sigma_{c_i}} e^{\frac{-(x_k-\mu_{c_i})^2}{2\sigma_{c_i}^2}}}$

Laplacian correction was incorporated to avoid the issue of zero probability.

Laplacian correction:*If zero probabilities were encountered it increases the count by one consequently increasing the denominator by one.*

Final value for probability is calculated as given in Equation 3.4-9 which would be used as the Naive Bayesian weight in the tool.

$$\boldsymbol{conditional\ probability = p(x|c_i).p(c_i)} \qquad \textbf{Equation 3.4-9}$$

3.5 Testing and Validation Options

The classifier was designed to validate classification under several categories. Three more options were introduced.

i. Use the training set

The dataset used for training the classifier was used as the validation set. Use of same set for training and evaluation may lead to over-fitting but a proper trained classifier would give 100% accuracy when the same set is used. This method evaluates how well the classifier can perform on data which it was trained. This method is used when there is only one data set which is not very large.

ii. Use cross validation

Cross fold validation was implemented as another mechanism to evaluate the goodness of the model. Cross validation is useful and proven to be accurate in validating a model when only one dataset is available and it is comparatively small.

In k-fold cross validation, the known dataset is divided into k folds. The entire dataset is divided into non overlapping k folds roughly of the same size and one such fold is taken as the test set and the remaining k-1 folds are taken as the training set. This entire process is repeated for k times hence each instance in the dataset is considered as a test instance only once and as training instance in k-1 times.

In gaKnn framework, GA algorithm has been used to optimize the k value and to prioritize the attributes by weighting them. The k value is decided by the fitness of each instance in the population. A repeated process is carried out to find the best fitted chromosome and thereby decide the k value as well as the weights obtained for each attribute.

The common practice for cross validation is to average the error calculated for each of the iteration of the k folds.

iii. Use a separate test set

In this method a separate dataset is used for evaluation independent of the train set which would be used to train the classifier. When the test data is different from the train set, the evaluation is more reliable. Preferably be used when two or more datasets are available which are drawn from the same domain.

3.6 Graphical User Interface (GUI)

Interpretability is one of the very important aspects in a churn prediction tool. We introduced a GUI to the framework in order to make it more user friendly and business oriented. Business personnel should be able to make business decisions with the support of the tool. GUIs are used to simplify the interaction between the computer program and

the user by representing them as visual elements. Thereby most of the technical details are masked to the user and he can freely interact with the program which is then attractive to non-technical people. Integration of GUIs could enhance the productivity as it facilitates multitasking.

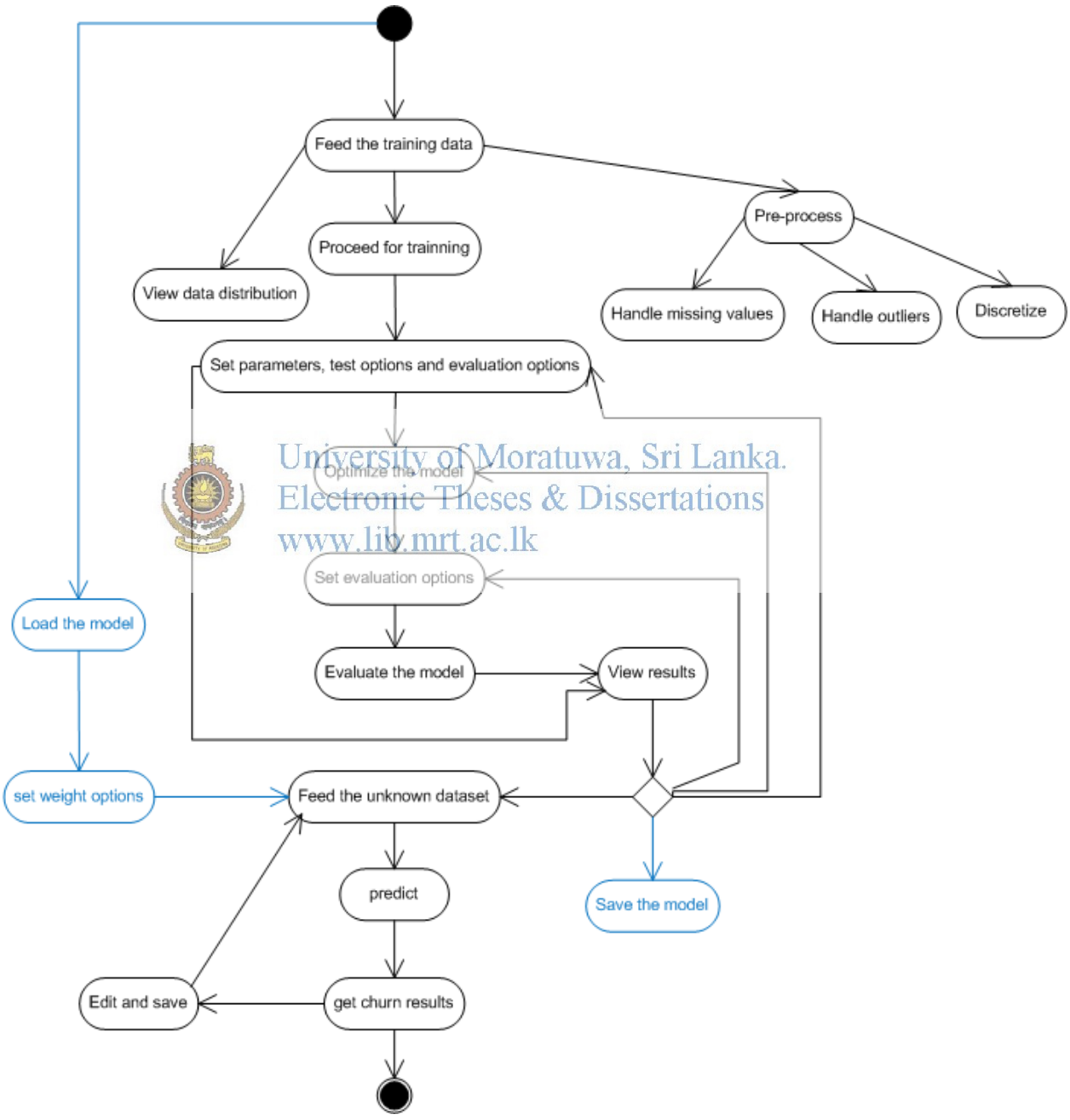

Figure 3.6-1Activity flow

The designed GUIs contain only the most important features of the system which a business user has to handle. But it could be extended with the addition of new features to the tool.

Figure 3.6-1 illustrates the basic functionality flow of the Tool. GUIs were designed in order to cater all those main aspects shown by the figure. Indicated in blue color are the processes which are optional to be carried out.

3.7 Saving the Model for Re-Use

By saving the model, we expect that the Tool could be used on similar test samples without training the Tool to obtain the optimized parameters at each run.

The optimized k value and the attribute vectors are stored in an xml file. But, such a model which is to be used in KNN classification requires information regarding the data which it was trained on. Therefore we designed to store information regarding the training data along with the optimized parameters.

Then the user can directly load the saved model which contains the optimized k value, weight vector and information regarding training data to predict the class label of a test sample of the same domain.

CHAPTER 4- IMPLEMENTATION

4 Implementation

This chapter elaborates the technologies used in implementing the design described in the previous chapter. The chapter also explains how the Tool could be used for churn prediction.

4.1 Implementation Environment and Process

The gaKnn framework was implemented in java and has used JGAP 3.3.3 library for GA implementation. Hence, additional enhancements as well as the Tool were implemented in the java environment using the technologies which are compatible with java.

The R language (Section 4.1.1) was used in implementing certain modules for statistical calculations and graphing. R 3.1.3 was configured in the Tool. After installing the R platform, rJava package has to be installed which has the three important libraries; JRIEngine.jar, JRI.jar and REngine.jar

The library opencsv-2.3.jar was used to handle CSV data files through the GUI. It is an open source CSV parser for java. When using *opencsv* library, it has to make sure its compatibility with java environment.

4.1.1. R language

R is a programming/ scripting language which has convenient features for statistical evaluations and graphing. It performs on variety of platforms including windows environment, UNIX based platforms and MacOS.

JRI is the Java to R interface and rJava is the R to java interface which facilitate communication with R and java. As rJava is installed, it adds-up the three important libraries *JRIEngine.jar*, *JRI.jar* and *REngine.jar*.

R code could be included within the java code or it can be written as R scripts which could be called from the java code. Both the methods have been used in our work.

4.1.2. Process implementation

Outlier handling

All the four outlier handling techniques were implemented with R language. Constraints of each technique were programmed into R scripts which were called from the java code.

Missing value handler

This was implemented with java.

Data discretization

Equal width binning technique was implemented with R language.

Similarity measurement/ distance function

Numeric representation of data was used for the distance and similarity calculations.

Distance function for numerical attributes was defined as a Euclidean distance function. Since the standard Euclidean distance may not perform well in high dimensional data, attribute weights were incorporated in the distance function as follows.

$$distance(x_k, y_k) = \sqrt{\sum_{k=1}^{n} w_k (x_k - y_k)^2}$$

Where x_k and y_k are data values of k^{th} attribute

n - number of attributes

w_k - weight of k^{th} attribute

Manhattan distance is an alternative to Euclidean distance with less computation cost. Hence, it was defined as the default distance function.

$$distance(x_k, y_k) = w_k|(x_k - y_k)|$$

The similarity value was obtained as

$$Similarity(x_k, y_k) = \frac{1}{\sum_{k=1}^{n} distance(x_k, y_k)}$$

Fitness function

The following fitness function was used.

$$fitness = \left(1 \Big/ \sqrt{\sum_{i=0}^{m-1} (1 - Conf_i)^2}\right) X\ 100$$

Where m is the number of instances in test data

$Conf_i = \frac{V(C_i)}{\sum_j^{k-1} V(C_i)}$ for k similarity measures of which $V(C_i)$ is the vote on class index C_i

Instance weighting

1. Class Confidence Weights

   The design in equations 3.4-1, 3.4-2, 3.4-3, 3.4-4, 3.4-5 and 3.4-6 were implemented with java. Equation 3.4-3 to 3.4-6 denote the relevant values when the class label is one, hence similar implementation was carried out for class label zero.

   For the Bayesian Network implementation [39], initially the conditional probability values were calculated and stored in a separate file which could be used while traversing through the instances.

2. Naïve Bayesian Weights

The design in equations 3.4-7, 3.4-8 and 3.4-9 were implemented with java. The values obtained by Equation 3.4-9 were used during KNN label prediction as a weight.

Training and validation options

The three techniques described in the Chapter 3 were implemented.

Data/Results analysis and graphing

The Tool provides a graphical overview of the data read to it. Thereby the user is provided with an overview of the distribution of the data (Figure 4.1.1).

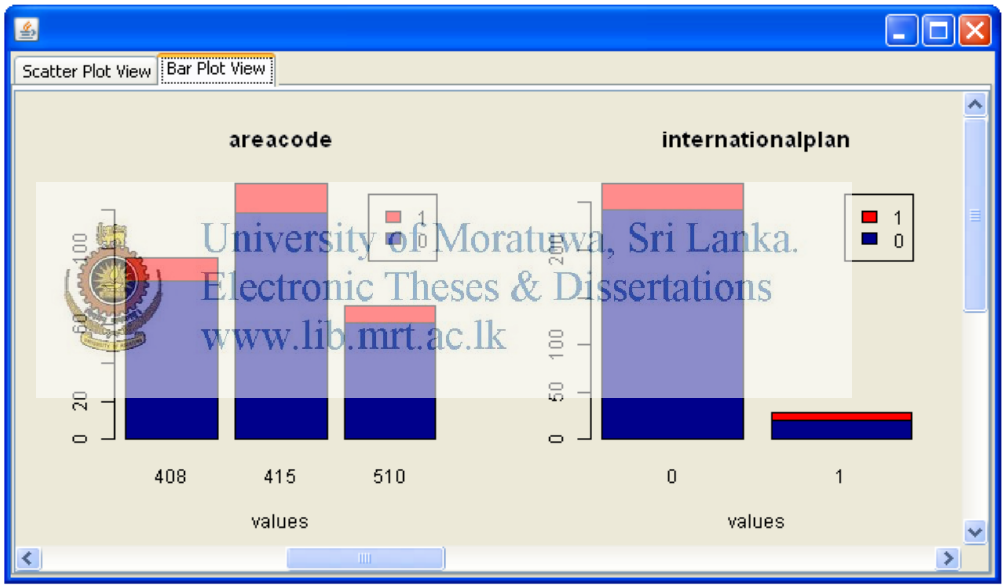

Figure 4.1-1 Graphical View of Data

The main Interface of the Tool is where the user could set necessary optimizing parameters as well as select the optimizing options (Figure 4.1.2).

Tool has simple GUIs for data preprocessing (Figure 4.1.3).

The Tool provides a better interpretation of the results of all the tests performed. The most important result from the Tool is the predicted churners (Figure 4.1.4, Figure 4.1.6).

Most of the graphs were sketched using R Language. Accuracy measurements were also implemented with R providing a graphical view (Figure 4.1.7).

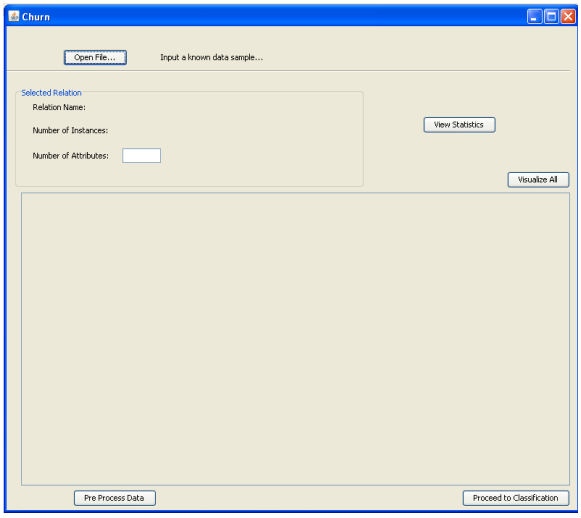

Figure 4.1-2 Interface to feed data

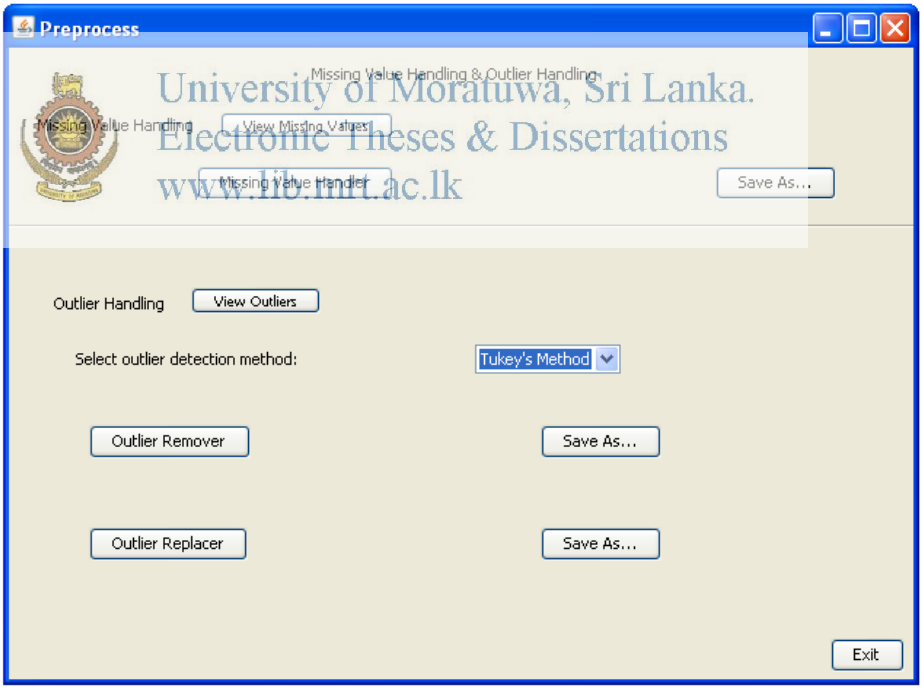

Figure 4.1-3 Data Preprocessing Window

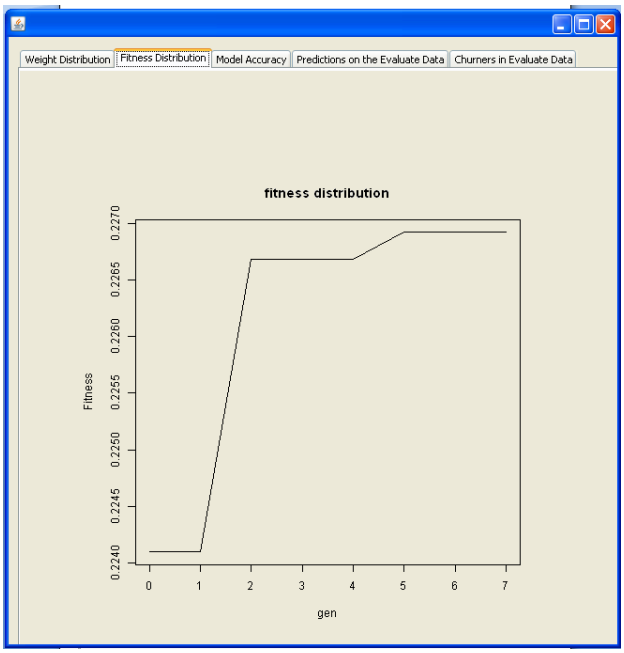

Figure 4.1-4 Visualize Results

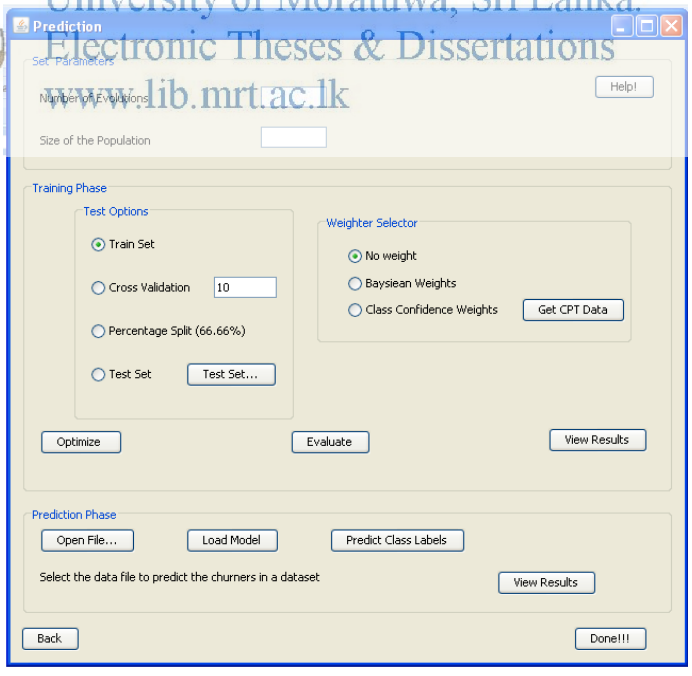

Figure 4.1-5 Prediction window

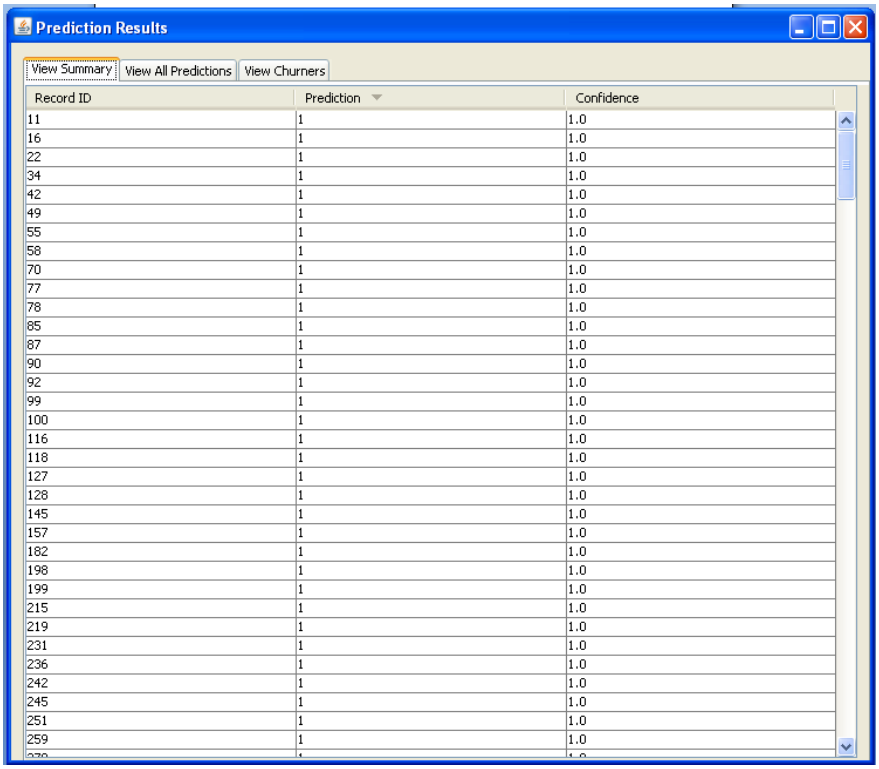

Figure 4.1-6 Churn Results

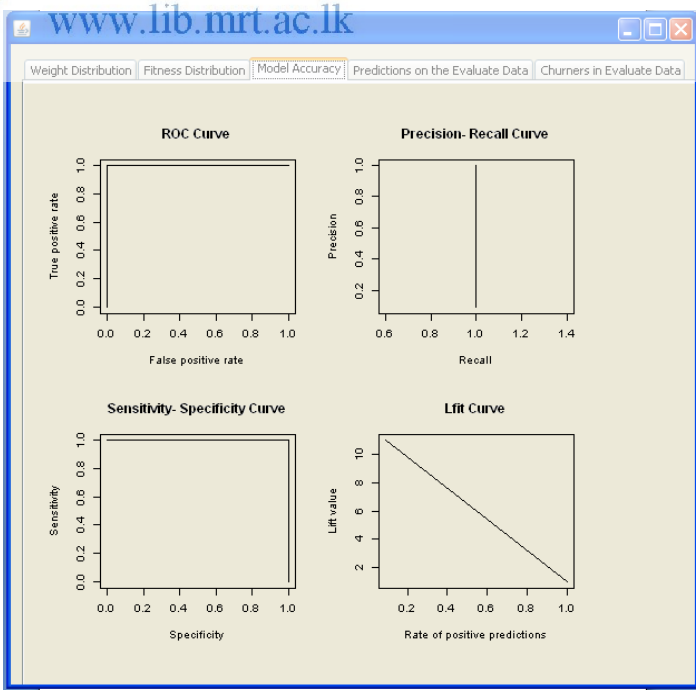

Figure 4.1-7 Accuracy Measurements

Saving the model for re-use

The model is saved as an xml file. That contains the optimum k value, weights on each attribute and information about the training dataset such as the name of the dataset.

KNN classification relies on classifying an unknown tuple based on some known data. That is, it requires training data during the classification of the class labels of the unknown data. Therefore when saving the model it has to consider the fact that information regarding the training data are also saved in the model since the training data is required during the classification.

The model is automatically saved with the completion of the optimization and training process. The model is saved as an xml file with the file extension *.prm* of which the name is the file name of train data file.

CHAPTER 5- TESING AND EVALUATION

5 Testing and Evaluation

This Chapter describes how the Tool was tested on different sets of data and the performance of the Tool in different situations.

5.1 Testing the Tool

In order to test the goodness of the Tool, test goals were set up based on our objectives. Then the designed tests were performed and the results were gathered to check whether the goals are met or any deviation thereof.

5.1.1 Test goals

The Tool was tested for the following features.

1. **Usability and adaptability**

This feature describes whether the tool can handle different churn datasets when different parameter values are provided.

2. **Accuracy**

This feature is to check whether the Tool provides a satisfactory level of accuracy.

The churn prediction Tool was built based on the gaKnn framework which was proved to provide a good level of accuracy for general datasets. Hence we have tested whether the Tool could also provide a good level of accuracy for churn prediction.

We have introduced two instance weighting mechanisms to the Tool and we tested the variation of accuracy when those weights are used against when no such weight has been applied during the prediction process.

The gaKnn framework has used the percentage split as the option for obtaining the training and validation sets during training and we have introduced another three

methods similar to WEKA. In this scenario we tested only the variation from the original option between the new methods.

Then, the four outlier handling mechanisms were also tested for accuracy. In this scenario, variation of accuracy when outliers were handled versus when outliers were not handled was tested.

3. Interpretability and user-friendliness

This feature checks three aspects of the Tool. They are,

i. whether the graphical user interface provides useful guidance in proceeding through the process
ii. whether the user can use any churn data set for churn prediction with minimal effort
iii. whether the churn results are presented in an effective manner

4. Suitability

This feature checks whether the Tool could be used effectively for the churn prediction tasks

5.1.2 Testing process and results

The Tool was tested for the features described in Section 5.1.1 with three data sets (Table 5.1-1).

Among the three datasets, Dataset1 has been widely used in researchers carried out for churn prediction [35][36].

Testing Environment

Processor: Intel(R) Core(TM) 2 Duo CPU, 2.00 GHz, 2.99 GB of RAM

Operating System: Microsoft Windows XP, professional version 2002, service pack 2

Supplementary Software: jdk 1.6, R i386 3.1.3

Dataset	Dataset 1 [19]	Dataset 2 [37]	Dataset 3 [38]
Name of the Dataset	sgi-churn data	churnReduced	unitedReducedChurn
Number of Instances	Train set: 3333 Test set: 1667	42,909	7344
Number of Attributes	19	6	3
Types of Attributes	Numeric, Nominal	Numeric, Nominal (only the class label)	Numeric, Nominal
Missing Values	No	No	Yes
Class Distribution	Positive: 483 Negative: 2850 (only train set)	Positive: 3153 Negative: 39,756	Positive: 3672 Negative: 3672

Table 5.1-1Tested Data

Testing Process

Initially, the three datasets were tested with WEKA and compared the results by testing the same data sets with the Tool. Then each data set was tested for accuracy of each voting method separately. Outlier handler and missing value handler were tested each with one data set.

Comparison to WEKA

Six random samples were drawn from the three datasets as train sets and test sets. Those sample data sets were tested with the Tool and with WEKA to compare the level of accuracy produced by the Tool.

Number of Instances in train set: 1199		
Distribution - Positive: 149 Negative: 1050		
	with Tool	**with WEKA**
	Evol: 4, pop: 15 Weight: no weight	iBk / k=5
AUC	0.651	0.644
Lift	2.834199	-
Precision	0.344828	0.389
Recall	0.410959	0.096
F1 Measure	0.375	0.153995876
TP	30	7
TN	470	516
FP	61	11
FN	58	66
Correctly Classified Instances	83.38%	87.17%

Table 5.1-2 Results for sample drawn from dataset 1

Number of Instances in train set: 1500		
Distribution - Positive: 96 Negative: 1404		
	with Tool	**with WEKA**
	Evol: 5, pop: 15 weight: Bayesian	iBk / k=5
AUC	0.515	0.588
Lift	2.901354	-
Precision	0.1818182	0
Recall	0.04255319	0
F1 Measure	0.068966	0
TP	2	0
TN	694	702
FP	9	1
FN	45	47
Correctly Classified Instances	92.80%	93.6%

Table 5.1-3 Results for sample drawn from dataset 2

Number of Instances in train set: 1200		
Distribution - Positive: 607 Negative: 593		
	with Tool	**with WEKA**
	Evol:4 pop:15 Weight:no weight	iBk / k=5
AUC	0.779	0.573
Lift	1.672175	-
Precision	0.8458418	0.539
Recall	**0.6869852**	**0.55**
F1 Measure	0.75818184	0.544444444
TP	417	322
TN	517	340
FP	76	275
FN	190	263
Correctly Classified Instances	77.83%	55.17%

Table 5.1-4 Results for sample drawn from dataset 3

Training set was used as the test option in both instances. In WEKA k value was set to 5 as Tool's initial k value was set to 5. Tests performed and results were recorded as denoted from Table 5.1-2 to Table 5.1-4

Three data sets were tested separately for accuracy of each voting method, outlier and missing value handling.

Dataset 1:

Random sample of 1199 from the train data were taken and tested under the 12 test cases denoted in Table 5.1-5 and the model was evaluated on a random sample of 600 taken from the test data. Evaluation results were recorded in Table 5.1-6

Class distribution of the train set: Positive – 149, Negative - 1050

Parameters:- Number of Evolutions: 20 Size of population: 20 Mutation: 1000

		Instance Weights		
		No Weight	**Bayesian Weight**	**CCW**
Train Option	**Train set**	**1)** AUC: 1.000	**2)**AUC: 1.000	**3)**AUC: 1.000
		Lift: 8.04698	Lift: 8.04698	Lift: 8.04698
		Time: 17 mins	Time: 14 mins	Time: 2 hr 38 mins
	CV	**4)** AUC: 1.000	**5)** AUC: 1.000	**6)** AUC: 0.570
		Lift: 8.040268	Lift: 8.040268	Lift: 3.654667
		Time: 2 mins	Time: 4 mins	Time: 40 mins
	Percentage Split	**7)** AUC: 1.000	**8)** AUC: 1.000	**9)** AUC: 1.000
		Lift: 9.068182	Lift: 9.068182	Lift: 9.068182
		Time: 4 mins	Time: 10 mins	Time: 1 hr 14 mins
	Test set	**10)** AUC: 0.499	**11)** AUC: 0.522	**12)** AUC: 0.523
		Lift: 0	Lift: 1.315068	Lift: 1.33284
		Time: 19 mins	Time: 26 mins	Time: 1 hr 35 mins

Table 5.1-5 Comparison between different training options

Train case No:	Weight option	AUC	Lift Value	Precision	Recall	F1 Measure	FN
1	no weight	0.518	**4.109589**	**0.5**	0.04109589	0.075949366	70
	bayes	**0.618**	1.730353	0.210526316	**0.493150685**	0.295081967	**37**
	ccw	0.54	1.761252	0.214285714	0.164383562	0.186046512	61
2	no weight	0.525	**5.479452**	**0.666666667**	0.054794521	0.101265824	69
	bayes	**0.623**	1.820704	0.221518987	**0.479452055**	0.303030303	**38**
	ccw	0.573	2.680167	0.326086957	0.205479452	0.25210084	58
3	no weight	0.565	1.967972	0.23943662	0.232876712	0.236111111	56

	bayes	0.565	1.967972	0.23943662	0.232876712	0.236111111	56
	ccw	**0.579**	**2.27608**	**0.276923077**	**0.246575342**	0.260869565	**55**
4	no weight	0.607	**6.921413**	**0.842105263**	0.219178082	0.347826087	57
	bayes	**0.621**	1.718133	0.209039548	**0.506849315**	0.296	**36**
	ccw	0.616	2.630137	0.32	0.328767123	0.324324324	49
5	no weight	0.554	**4.35133**	**0.529411765**	0.123287671	0.2	64
	bayes	**0.621**	1.797945	0.21875	**0.479452055**	0.300429185	**38**
	ccw	0.61	2.589604	0.315068493	0.315068493	0.315068493	50
6	no weight	0.565	**3.287671**	**0.4**	0.164383562	0.233009709	61
	bayes	**0.6**	1.71003	0.208053691	**0.424657534**	0.279279279	**42**
	ccw	0.599	2.538276	0.308823529	0.287671233	0.29787234	52
7	no weight	0.506	**1.643836**	**0.2**	0.02739726	0.048192771	71
	bayes	**0.541**	1.268392	0.154320988	**0.342465753**	0.212765958	**48**
	ccw	0.531	1.494396	0.181818182	0.164383562	0.172661871	61
8	no weight	**0.535**	**1.590809**	**0.193548387**	**0.164383562**	0.177777778	**61**
	bayes	**0.535**	**1.590809**	**0.193548387**	**0.164383562**	0.177777778	**61**
	ccw	0.523	1.369863	0.166666667	0.150684932	0.158273382	62
9	no weight	0.499	0	0	0	0	73
	bayes	**0.567**	**1.418787**	**0.172619048**	**0.397260274**	0.240663901	**44**
	ccw	0.522	1.393081	0.169491525	0.136986301	0.151515151	63
10	no weight	0.522	1.315068	0.16	0.164383562	0.162162162	61
	bayes	0.522	1.315068	0.16	0.164383562	0.162162162	61
	ccw	0.522	1.315068	0.16	0.164383562	0.162162162	61
11	no weight	0.523	1.33284	0.162162162	0.164383562	0.163265306	61
	bayes	0.523	1.33284	0.162162162	0.164383562	0.163265306	61
	ccw	0.523	1.33284	0.162162162	0.164383562	0.163265306	61
12	no weight	0.497	0	0	0	0	73
	bayes	**0.538**	**1.211247**	**0.147368421**	**0.383561644**	0.212927757	**45**
	ccw	0.511	1.157631	0.14084507	0.136986301	0.138888888	63

Table 5.1-6 Comparison on different test options

The Outlier handler was tested with the original train data set (the dataset with 3333 instances). The tests denoted in Table 5.1-7 were performed and the results were recorded. Along with the tests by the Tool outlier handled files were also tested with WEKA and the results were recorded in Table 5.1-8.

Both the tools were trained with the outlier handled data set and were evaluated on the test set.

Churn Prediction Tool was trained and tested with the following parameters:

Number of evolutions: 30 size of the population: 20 mutations: 1000

	Outlier Handling Method	Num. Instances in resultant file	AUC	Lift Value	TP	TN	FP	FN	Recall	Correctly Classified Instances
1	Original file	3333	0.532	3.720982	17	1426	17	207	0.07589	86.56%
2	Outlier Removal with Tukey's Method	3331	0.605	5.392728	50	1424	19	174	0.22321	88.42%
3	Outlier Removal with MedCouple	2511	**0.618**	**5.41234**	56	1422	21	168	0.25	**88.66%**
4	Outlier Replace with Tukey's Method	3333	0.577	1.993383	60	1279	164	**164**	**0.26786**	80.32%
5	Outlier Replace with MedCouple	3333	0.518	1.517294	21	1361	82	203	0.09375	82.90%

Table 5.1-7 Comparison outlier detection/handling methods- with Tool

WEKA was trained and tested with iBk classifier for k=5 and no distance weight was used.

	Outlier Handling Method	Num. Instances in resultant file	AUC	Lift Value	TP	TN	FP	FN	Recall	Correctly Classified Instances
1	Original file	3333	0.722	N/A	31	1415	28	**193**	0.13839	86.74%
2	Outlier Removal with Tukey's Method	3331	0.727	N/A	31	1415	28	**193**	0.13839	86.74%
3	Outlier Removal with MedCouple	2511	0.688	N/A	30	1421	22	194	0.13393	87.04%
4	Outlier Replace with Tukey's Method	3333	**0.913**	N/A	135	2824	26	348	**0.27950**	**88.78%**
5	Outlier Replace with MedCouple	3333	0.636	N/A	9	1433	10	215	0.04018	86.50%

Table 5.1-8 Comparison outlier detection/handling methods- with WEKA

Dataset 2:

Two samples were drawn from the dataset of 1500 instances and 750 instances where the first sample was taken as the train dataset (*ChurnReducedTrainSet_sample.arff*) and the later as the test dataset (*ChurnReducedTestSet_sample.arff*).

Class distribution of the train set: Positive – 96, Negative – 1404

The tests in Table 5.1-9 were performed and results were recorded

Parameters:- Number of Evolutions: 4 Size of population: 15 Mutation: 1000

		instance weight		
		no weight	**Bayesian weight**	**CCW**
Train Option	**Train set**	**1)** AUC: 1.000	**2)**AUC: 1.000	**3)**AUC: 1.000
		Lift: 15.625	Lift: 15.625	Lift: 15.625

Table 5.1-9 Results of training phase

Train case No:		AUC	Lift Value	Precision	Recall	FN
1	no weight	0.510	1.77305	0.11111	0.04255	45
	bayes	0.512	2.12766	0.13333	0.04255	45
	ccw	0.505	1.055038	0.06611	0.17021	39

Table 5.1-10 Results of evaluation with test data

Dataset 3:

The dataset was divided in to two parts as *unitedReducedChurn_train.arff* and *unitedReducedChurn_test.arff* where the first file contained 6000 instances and the later contained 1344 instances. The *unitedReducedChurn_train.arff* was used as the training dataset while *unitedReducedChurn_test.arff* was used as the test dataset.

Class distribution of the train set: Positive – 3022, Negative - 2978

The tests in Table 5.1-11 were performed and results were recorded

Parameters:- Number of Evolutions: 20 Size of population: 20 Mutation: 1000

		instance weight		
		no weight	**Bayesian weight**	**CCW**
Train Option	**Train set**	**1)** AUC: 0.755	**2)**AUC: 0.755	**3)** AUC: 0.773
		Lift: 1.685565	Lift: 1.685565	Lift: 1.663718
	CV	**4)** AUC: 0.801	**5)** AUC: 0.776	**6)** AUC: 0.786
		Lift: 1.824151	Lift: 1.613216	Lift: 1.679289

Table 5.1-11 Results of training phase

Train case No:		AUC	Lift Value	FN
1	no weight	0.515	1.042498	409
	Bayes	**0.523**	1.048095	**319**
	ccw	0.517	**1.056267**	438
2	no weight	0.515	1.042498	409
	Bayes	**0.523**	1.048095	**319**
	ccw	0.517	**1.056267**	438
3	no weight	0.515	1.042498	409
	bayes	**0.523**	1.048095	**319**
	ccw	0.517	**1.056267**	438
4	no weight	0.555	**1.135501**	385
	bayes	0.552	1.123793	376
	ccw	**0.572**	1.114078	**224**
5	no weight	0.524	1.051643	325
	bayes	0.52	1.039822	**302**
	ccw	**0.54**	**1.118151**	398

Table 5.1-12 Evaluated with test data

Since this dataset contained missing values, the missing value handler was tested with the training dataset (*unitedReducedChurn_train.arff*).

First, the missing values were removed by the handler and then the resultant file was trained to evaluate with the test dataset (*unitedReducedChurn_test.arff*)

The tests were performed and the results were recorded.

Parameters: Number of evolutions: 20 Size of population: 20 Mutation: 1000

	The file used	Num. Instances in resultant file	AUC	Lift Value	TP	TN	FP	FN	Recall	Correctly Classified Instances
1	Original file	6000	0.523	1.048095	331	372	322	319	0.509231	52.31%
2	Missing value filtered data set	5805	0.533	1.072737	331	387	307	319	0.509231	53.42%

Table 5.1-13 Results for missing value handled data

5.2 Evaluation

Test results were analyzed based on the features explained in Section 5.1.1. This section elaborates the results for each of the feature so explained.

1. Adaptability

The three different datasets could be read by the Tool and could present their distribution graphically. The Tool could also produce a statistical summary in terms of mean, median, minimum, maximum, first and third quartiles for numerical data and number of occurrences for nominal data.

The outlier detection module could handle all the three datasets. The Tool produced similar accuracy with WEKA for Tukey's method for outlier detection. For the MedCouple outlier detection method all the three datasets produced better outcome. It was observed that, removal methods scored better than outlier replacing methods in most instances.

Only one dataset had missing values and the missing value handler could handle that dataset properly.

2. Accuracy

Comparison to WEKA

Table 5.1-2, 5.1-3 and 5.1-4 illustrate the comparison of the accuracy of the Tool with WEKA. Table 5.1-2 shows that the AUC, Recall and the FN count are better in the Tool than those of WEKA. The Precision and the amount of total correct classifications are higher in WEKA. But the F1 measure shows better results with the Tool. The closer the F1 measure to 1, the better the model is. Results show that most important factors of a churn model were shown by the Tool.

Table 5.1-3 shows that the Precision and Recall are higher and the amount of FNs is less for the Tool compared to WEKA. Therefore the results indicate that the Tool is better as a churn tool than as a general classification tool like WEKA.

Table 5.1-4 shows that for all the metrics Tool produced better results for data set 3.

During the above comparison for Data set 1 and Data set 3 no weighting method was used. That is because even without any voting on the prediction it could provide a satisfactory solution.

Voting and training option

The Table 5.1-5 and Table 5.1-6 illustrate the results obtained at each train and test case to test the vote weighting method when the Dataset 1 is used. Table 5.1-9 and Table 5.1-10 shows the tests performed and results obtained for Dataset 2. Table 5.1-11 and Table 5.1-12 illustrate the relevant results obtained for Dataset 3.

According to Table 5.1-5, it produced the highest lift value when trained using percentage split and it produced the optimum AUC value at all the three training instances except when it used the test set during training. But when the model is evaluated on the test data, the model built by cross validation produced the most accurate results.

Comparisons of the results for different weights are shown in Table 5.1-6, Table 5.1-10 and Table 5.1-12. It was observed that the accuracy increases when the Bayesian Weights or CCW were added to the prediction vote.

Figure 5.2-1 shows the deviation of AUC for the top five AUC values when different weights were applied.

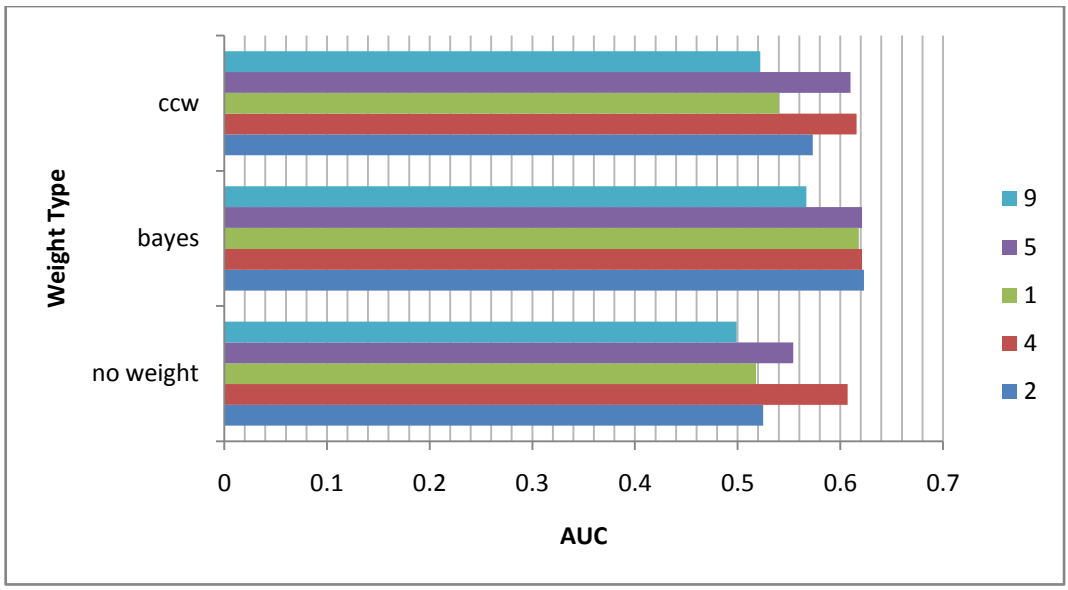

Figure 5.2-1 AUC for different weighted votes

When the Lift value is considered, the highest lift values has been produced when no weight is applied (Figure 5.2-2).

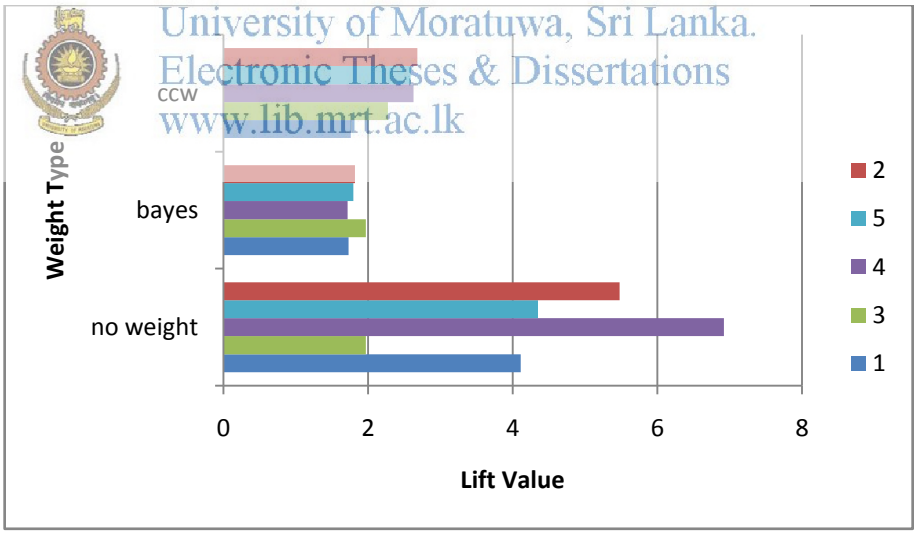

Figure 5.2-2 Change of Lift value

False Negatives and Recall

The most critical factor in churn prediction is the prediction of churning customers as non-churning customers. Because it avoids the service provider receiving any indication

regarding the churning customer, thereby avoiding the retention campaigns reaching them. Therefore the amount of False Negatives (FN) in a prediction plays an important role in a churn tool. The lower the FN the better the churn model is. Table 5.1-6 shows that Bayesian weight based predictions have the lowest FN values which were followed by the CCW based predictions.

Recall is important in a churn model as the model needs to minimize the amount of FNs. That indicates the amount of predicted churners out of the amount actual churners. Table 5.1-6 illustrates that Bayesian weight based predictions have high recall which were followed by the CCW based predictions.

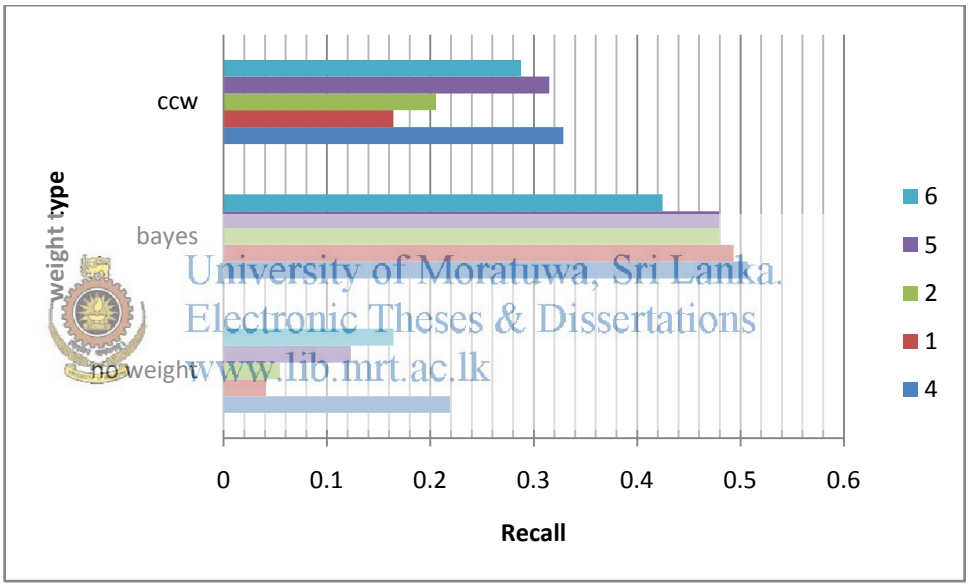

Figure 5.2-3 Change of Recall

Outlier handling

The four outlier handling methods were applied to the original dataset and four different output files were obtained. The files obtained were used for training the Tool and were evaluated on some test data. The results were recorded in Table 5.1-7 and Table 5.1-8.

Table 5.1-7 indicates the results obtained by evaluating the Churn Prediction Tool. It shows that the outlier detection method which uses the MedCouples has detected

considerable amount of outliers. MedCouple method was proven to be suitable for outlier detection when data set is skewed. Because, MedCouples are based on adjusted box plots.

Removal of outliers detected with MedCouples produced better results in terms of AUC when evaluated with the test data. At the same time, the MedCouple outlier handled files produced the lesser number of FNs compared to other methods. It must be noted that for a churn model, it is neither the amount of accuracy in terms of correct classification nor the AUC alone indicate the most suitable method.

Table 5.1-8 depicts the results obtained when the outlier handled files created by the Churn Prediction Tool were tested with WEKA. The results show that WEKA also produces similar results. Therefore it could be stated that the outlier detection method incorporated with the Tool is effective.

Missing value handling

The results in Table 5.1-13 shows that the AUC value and the Lift value have slightly improved when the missing values are handled.

3. Interpretability and usability

Tool was evaluated with several mechanisms and all such results are presented to the user graphically as well as textually through the GUI. As the dataset is read in to the Tool, the user is provided with the facility to view summary of the data graphically. A statistical evaluation in terms of minimum, maximum, median, mean and first and third quartile values for numeric data and number of occurrences for nominal data was also provided.

For each training effort and for each evaluation effort the evaluation metrics ROC curve, Precision Recall curve, Lift curve, area under ROC curve, precision, recall and lift value were recorded. These results could be accessed through the GUI. During the training

phase, by observing the patterns in fitness and other evaluation metrics the user could get the optimal model for evaluating.

The churn results were produced in an editable format so that the user could change data by analyzing the trends in the original data. Thereby the user can re-check the edited data for churn. The user was provided with the facility to view the trends in data while editing the results.

The most probable churners could be viewed at the top of the list so that they could be promptly targeted. Along with the list of churners, the probability each prediction is provided. Therefore the service provider can specifically target retention programs towards such customers.

The trained and optimized model could be saved and it could be used later. Thereby it reduces the time spent on training the model.

Refer Appendix A for the User Manual of the Tool.

4. Suitability

The Tool produced satisfactory results in terms of AUC, recall and FN counts which are most important measures in a churn model.

The Tool showed better accuracy as a churn tool when compared with WEKA.

Therefore it could be recommended that this Tool is suitable for the churn prediction task.

5. Performance

The time taken in completing the entire training and evaluating/ predicting tasks was highly depend on the number of instances in the training data set and the number of attributes in the dataset.

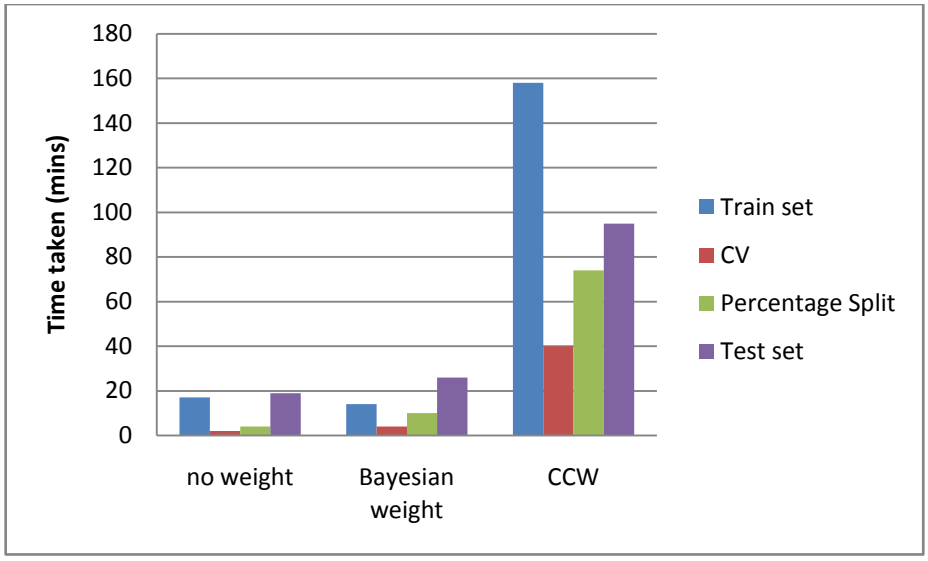

Figure 5.2-4 Performance - Dataset 1

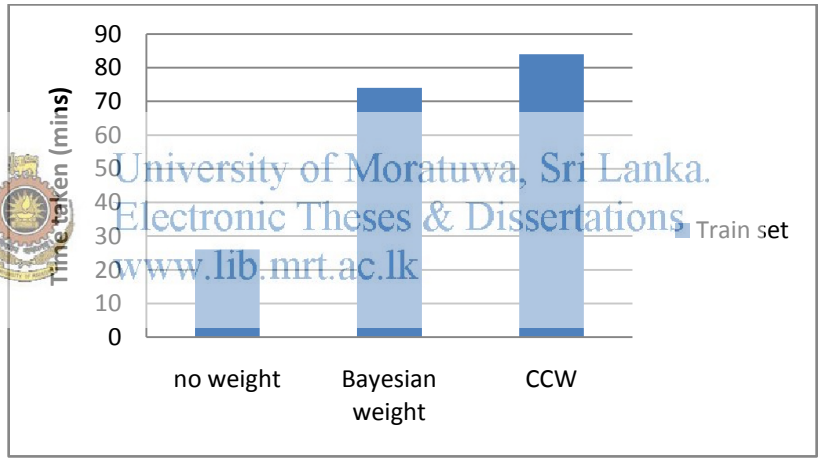

Figure 5.2-5 Performance – Dataset 3

Figure 5.2-3 and Figure 5.2-4 shows that time taken rapidly grows as weighting option changes from no weight, Bayesian weights to ccw. Time taken for the Dataset 1 was larger than the time taken for Dataset 3.

Number of instances in Dataset 1 was lesser than that of Dataset 3 but number of attributes in Dataset 1 was larger than the number in Dataset 3. When the number of attributes increases it takes much time for the process of genetic algorithm features. When the number of instances increases the computation of CCW takes much time.

Therefore when both the number of attributes and the number of instances increases it requires much computation time.

CHAPTER 6- CONCLUSION AND FUTURE WORKS

6 Conclusion and Future Works

This Chapter concludes the work carried out in implementing the Tool and its contribution for churn prediction tasks. Later part of the Chapter suggests extensions to the current Tool.

6.1. Conclusion

Incorporation of the Bayesian weights and class confidence weights as votes for prediction improved the accuracy of the classification. One of the most important factors of a churn model which requires having lesser number for false negatives was achieved by the incorporation of Bayesian weights and class confidence weights. That is mostly by addressing the class skew nature which is inherent to most churn data sets.

Proper handling of missing values and outliers play a key role in classification. Results also show improvement when those factors were addressed. In our scenario the datasets that were used contained comparatively a very low amount of missing values and outliers. Therefore the results obtained in the above tests improved slightly. But it could have a significant impact when used with very large datasets.

Similarity calculation is also an important part in the implementation. Improvements made to the similarity measurement had also contributed for better results.

Therefore, this Tool could be recommended as a comprehensive initiation for a churn Tool which is consisted with,

- ✓ Pre processing modules to handle missing values and outliers
- ✓ Data visualization facility
- ✓ Module for prediction with customizable value added feature
- ✓ Data editing facility for further predictions with minimal effort
- ✓ Presentation of churn results in human readable format

The Tool also makes room for future enhancements.

6.2. Future Works

The Tool is implemented with the basic principles of KNN rules and GA. In KNN classification, selection of k neighborhood plays a major role. With GA, an optimum value for k is calculated and the evaluation process was supported by instance weighting. For GA, definition of the stopping condition is an important task. In our Tool, the user has to specify the number of generations the GA should evolve before termination. Researchers have used different mechanisms for defining the stopping condition [4]. Hence, a new more improved criterion could be introduced for the stopping condition.

Our Tool provides a user about the future churners along with the confidence level assigned to each of the predictions. Therefore, the user can immediately attend to the most probable churners. This facility could be more extended to provide the user a time frame as to when a particular customer would churn.

The outlier handling mechanisms incorporated in the Tool address only the numeric data. A module could be introduced to handle outliers in nominal data.

Stratified 10-fold cross-validation is recommended over general k- fold cross validation for accuracy estimation [12]. Hence, the Tool's cross validation module could be improved for stratified cross validation.

The calculation of the class confidence weights (ccw) was a repetitive task. Time taken for calculations rapidly increases as the training data set grow. Therefore, the module may be re-engineered to optimize the calculation task.

GUI could be enhanced with features which are more attractive to non-technical people.

REFERENCES

[1] T. Au *et al.*, "Applying and Evaluating Models to Predict Customer Attrition Using Data Mining Techniques", *Journal of Comparative International Management*, Vol. 6, No: 1, Jan. 2003. ISSN 1718-0864, [Online]. Available: http://www.lib.unb.ca/Texts/JCIM/bin/get.cgi?directory=vol6_1/&filename=li.html. [Accessed: 04-Oct-2011].

[2] Y. Richter *et al.*, "Predicting customer churn in mobile networks through analysis of social groups," in *Proceedings of the 2010 SIAM International Conference on Data Mining (SDM 2010)*, 2010.

[3] L. Junxiang, "Predicting Customer Churn in the Telecommunications Industry – – An Application of Survival Analysis Modeling Using SAS®," in *SAS User Group International online proc.*, 2002, vol. Paper No.114–27.

[4] W. Au *et al.*, "A novel evolutionary data mining algorithm with applications to churn prediction," presented at the IEEE Trans. Evolutionary Computation, 2003, pp. 532–545.

[5] V. Lazarov and M. Capota, "Churn Prediction," Business Analytics Course, TUM Computer Science, 2007.

[6] S. V. Nath and R. S. Behara, "Customer Churn Analysis in the Wireless Industry: A Data Mining Approach," in *Annual Meeting of the Decision Sciences Instituite*, 2003, pp. 505–510.

[7] "Eleven Ways to Reduce Telecom Churn." [Online]. Available: http://www.tmcnet.com/usubmit/2008/01/29/3237095.htm. [Accessed: 04-Oct-2011].

[8] C. P. Parag, "Genetic algorithm based neural network approaches for predicting churn in cellular wireless network services," *Expert Systems with Applications*, vol. 36, no. 3, Part 2, pp. 6714–6720, Apr. 2009.

[9] "Outsourcing Customer Churn Analysis:: Customer Churn Analysis White Papers." [Online]. Available: http://www.marketequations.com/white-papers/customer-churn-analysis.html. [Accessed: 31-Oct-2011].

[10] R. J. Jadhav and U. T. Pawar, "Churn Prediction in Telecommunication Using Data Mining Technology," *(IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 2, No 2, pp. 17–19, Feb. 2011.

[11] T. Rashid, "Classification of Churn and non-Churn Customers for Telecommunication Companies," *International Journal of Biometrics and Bioinformatics (IJBB)*, vol. 3, no. 5, p. 82, 2009.

[12] J. Han and M. Kamber, "Classification & Prediction," in *Data Mining: Concepts & Techniques*, 2nd ed., Morgan Kaufmann Publishers, 2006, pp. 347– 355.

[13] N. Suguna and K. Thanushkodi, "An improved k-nearest neighbor classification using genetic algorithm," *IJCSI*, p. 18, 2010.

[14] H. Ahn *et al.*, "Global optimization of feature weights and the number of neighbors that combine in a case-based reasoning system," *Expert Systems*, vol. 23, no. 5, pp. 290–301, 2006.

[15] M. Analoui and M. F. Amiri, "Feature reduction of nearest neighbor classifiers using genetic algorithm," *Proceedings of world academy of science, engineering and tehchnology*, 2006.

[16] D. G. N. Dayaratne, "GeneticAlgorithm Optimized K-Nearest Neighbor Classification Framework (gaKnn)", M.Sc. Thesis, Dept. of Computer Science & Engineering, Univ. of Moratuwa, Sri Lanka, 2008.

[17] A. S. Perera *et al.*, "Evolutionary Nearest Neighbour Classification Framework," presented at the 18th International Conference on Software Engineering and Data Engineering, 2009, pp. 250–255.

[18] "JGAP: Java Genetic Algorithms Package." [Online]. Available: http://jgap.sourceforge.net/. [Accessed: 12-Jan-2012].

[19] "SGI - MLC++: Datasets from UCI." [Online]. Available: http://www.sgi.com/tech/mlc/db/. [Accessed: 26-Aug-2013].

[20] Mark Hall *et al.*, "The WEKA Data Mining Software: An Update; SIGKDD Explorations", Volume 11, Issue 1, 2009.

[21] "Mini Lecture: Churn Prediction: Analysis and Applications - YouTube." [Online]. Available: https://www.youtube.com/watch?v=6yCxbzDjBDc. [Accessed: 22-Feb-2015].

[22] "Turn data into revenue, faster!:Criticality of Predicting Customer Churn." [Online]. Available: http://www.infosysblogs.com/bigdata/2013/12/_my_personal_experience_of.html. [Accessed: 22-Feb-2015].

[23] B.W. Yap *et al.*, " An Application of Oversampling, Undersampling, Bagging and Boosting in Handling Imbalanced Datasets", *Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng-2013)*, Herawan,

T;Deris, M.M.; Abawajy, J. (Eds.) 2014,XXI, 730p.235 illus., Hardcover, ISBN: 978-981-4585-17-0

[24] L. Breiman, (1996). "Bagging predictors". Machine Learning 24 (2): 123–140. doi:10.1007/BF00058655. CiteSeerX: 10.1.1.121.7654

[25] W. Liu and S. Chawla, "A Quadratic Mean based Supervised Learning Model for Managing Data Skewness". In: *Proceedings of the Eleventh SIAM International Conference on Data Mining*, pp. 188–198 (2011)

[26] R. Akbani *et al.*, "Applying Support Vector Machines to Imbalanced Data Sets," Lecture Notes in Computer Science, vol. 3201, pp. 39-50, 2004.

[27] W. Liu and S. Chawla "Class confidence weighted knn algorithms for imbalanced data sets", *Proc. 15th Pacific-Asia Conf. Adv. Knowl. Discovery Data Mining,* pp.345 -356, 2011

[28] Jason W. Osborne and Amy Overbay (2004). The power of outliers (and why researchers should always check for them). Practical Assessment, Research & Evaluation, 9(6). [Online]. Available: http://PAREonline.net/getvn.asp?v=9&n=6 [Accessed: 11-Mar-2015].

[29] Prof. Carolina Ruiz, Department of Computer Science, WPI "Illustration of the K2 Algorithm for Learning Bayes Net Structures".

[30] J. Davis and M. Goadrich, "The Relationship between Precision-Recall and ROC Curves," *Proc. Int',l Conf. Machine Learning*, pp. 233-240, 2006

[31] "Lift Charts." [Online]. Available: http://www2.cs.uregina.ca/~dbd/cs831/notes/lift_chart/lift_chart.html. [Accessed: 15-Mar-2015].

[32] Seo, Songwon, "A Review and Comparison of Methods for Detecting Outliers in Univariate Data Sets", MSc thesis, Univ. of Pittsburgh, 2006

[33] M. Hubert and E. Vandervieren, "An adjusted boxplot for skewed distributions", Computational Statistics and Data Analysis 52 (2008) 5186–5201

[34] G. Brys *et al*., 2004. "A robust measure of skewness," Journal of Computational and Graphical Statistics 13, 996–1017.

[35] Amin, Adnan, et al. "Churn prediction in telecommunication industry using rough set approach." *New Trends in Computational Collective Intelligence*. Springer International Publishing, 2015. 83-95.

[36] Verbeke, Wouter, et al. "Building comprehensible customer churn prediction models with advanced rule induction techniques." *Expert Systems with Applications* 38 (2011): 2354-2364.

[37] "dme.churn/dataset/Balanced/FilledCFS at master · inquire/dme.churn · GitHub." [Online].Available:https://github.com/inquire/dme.churn/tree/master/dataset/Balanced/FilledCFS. [Accessed: 06-May-2015].

[38] "dme.churn/dataset/Balanced at master · inquire/dme.churn · GitHub." [Online]. Available: https://github.com/inquire/dme.churn/tree/master/dataset/Balanced. [Accessed: 06-May-2015].

[39] "Bayesian Network Classifiers in Weka for Version 3-5-7." [Online]. Available: http://www.cs.waikato.ac.nz/~remco/weka_bn/. [Accessed: 12-Oct-2014].

[40] T. Saito and M. Rehmsmeier, (2015). *The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets*. PLoS ONE, 10(3), e0118432. doi:10.1371/journal.pone.0118432

APPENDIX A – User Manual

gaKnn Churn Prediction Tool Version 1.0

User Manual

gaKnn Churn Prediction Tool is an easy way that you can get to know about the future churners of your product or service. This provides simple and easy to handle, user interfaces to get your work done. The Tool uses the key concepts of `k` nearest neighbor classification and genetic algorithms; supported by the concepts of Naïve Bayesian Weights and Class Confidence Weights (ccw).

----------System Requirements----------

- `JDK 1.6` or above
- `jgap` library
- `R 3.1.3` or above with `rJava` Package installed
- Other R libraries required: `foreign, ROCR, gplots, robustbase, infotheo, hexbin, colorspace, ggplot2`
- `opencsv 3.1.2` library (should be compatible with the `jdk` version)

----------Working with the Tool----------

Start the Tool

The Tool starts with the interface Figure A-1. Proceed with the instructions given at each step.

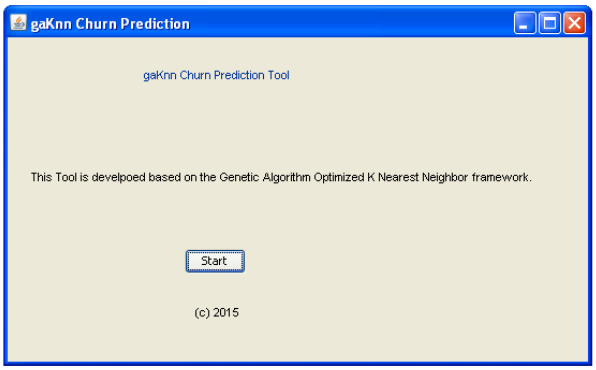

Figure A-1 First Interface

Selecting the Dataset

Select the dataset through window in Figure A-2. As the dataset is selected it provides a graphical as well as a statistical overview of the selected dataset.

To get the statistical data follow the `'View Statistics'` button.

Figure A-2 Data selection

Data Visualizing

You can get several views about the distribution of data through 'Visualize All' section.

Scatter plot view

Bar plot view

Data Pre-processing

If you proceeded to the preprocess module via preprocess button in the 'Churn' window, the window in Figure would be shown.

The 'Preprocess' window provides you the ability to handle missing values and outliers. Click on the button 'Missing Value Handler' and it will remove the missing values in the selected data file. The data instances which contain missing values could be viewed by clicking the button 'View Missing Values'.

If you need to use the missing value handled data file as your training data file, select that data file through the 'Churn' window.

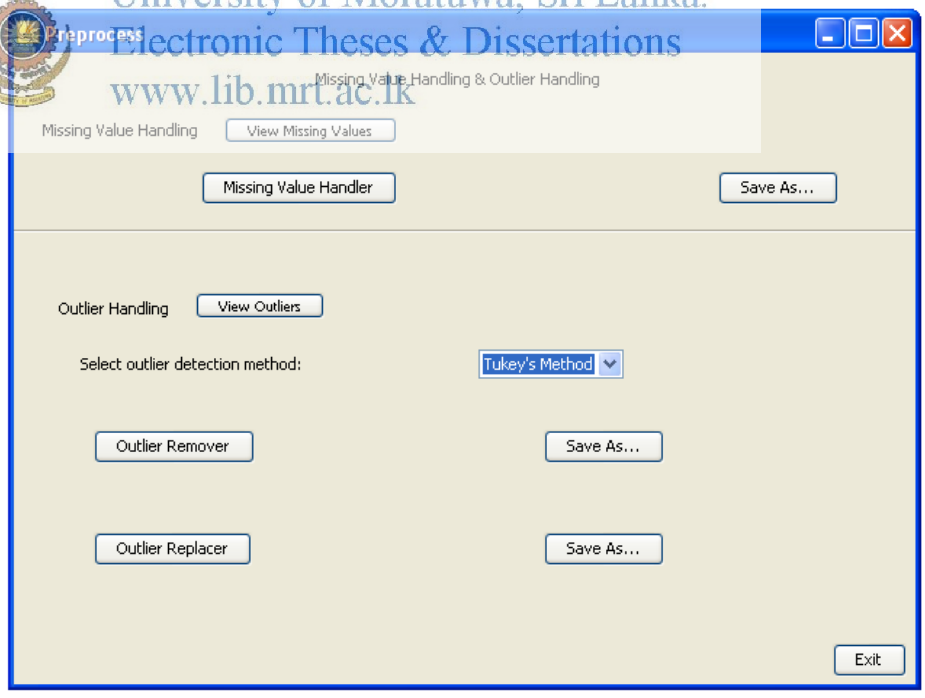

Figure A-3 Data pre-processor

Outliers in the selected data file are visualized as box plots. You can view them through the button 'View Outliers'.

The Tool provides two outlier detection methods;

1. Tukey's Method
2. MedCouples

Outlier handler provides two handling methods; removal and replacement.

First select the detection method from the dropdown menu and then click on the preferred handling method.

Please note that the visualized outliers are determined by the whiskers extend to the most extreme data point which is no more than range times the inter quartile range from the box.

Optimizing the KNN classifier

You can proceed to the optimizing phase via 'Proceed to Classification' button.

The Prediction window allows you to perform both the optimizing and prediction tasks.

The optimizing task involves genetic algorithm techniques. It is by this step the Tool finds an optimum value for k and a weight vector for the attributes considering their importance.

The number of times which genetic algorithm is required to iterate is read by the Number of Evolutions and the size of the population considered by genetic algorithm is read by Size of the Population fields in the Set Parameters section.

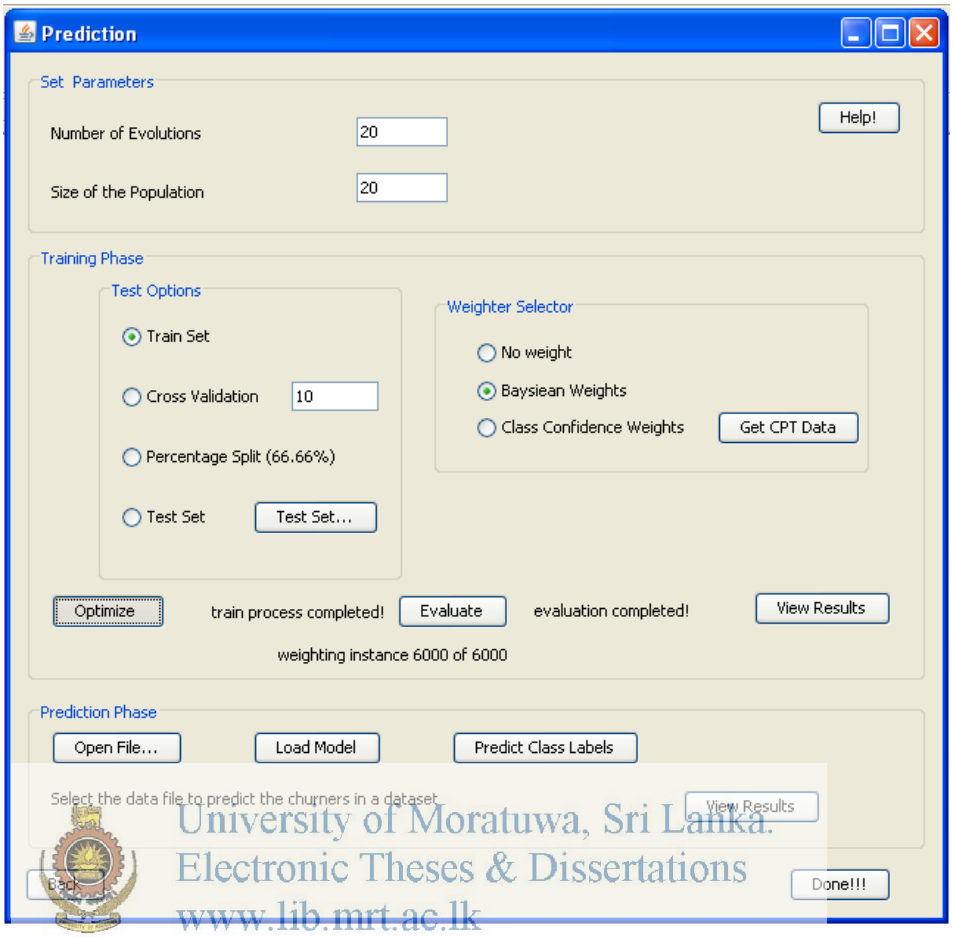

Figure A-4 Predictor

The population estimated with genetic algorithm is evaluated with KNN.

KNN classification relies on classification of unknown data based on known data. Therefore to test the KNN classifier it requires having two data sets; train set for training and test set for validating. In the `Training Phase` the Tool introduces four options to define the train set and test set; `Train Set`, `Cross Validation`, `Percentage Split` and `Test Set`.

If `Train Set` option is selected the dataset read into the Tool is used as both train set and test set. The `Cross Validation` option partitions the dataset, into specified `k` number of folds which would be used iteratively for `k` times where one fold is taken as

the test set while the remaining k-1 folds are used as the train set. The option Percentage Split partitions the dataset into two folds of which one fold contains two-third of the dataset and the other fold contains the remaining one-third. If you use the test set option then you are required to select a separate dataset which is from the same domain as the dataset already read into the Tool. The first dataset is used as the train set while the second fed is used as the test set.

To get a better result on prediction with KNN, instance voting is introduced. Two voting methods were introduced; Bayesian Weights, Class Confidence Weights. You may either perform the classification without either of the two methods by selecting 'No Weight' option. Bayesian Weights add a vote to each prediction based on the Bayesian probability of getting a particular class label. Class Confidence Weights add a vote to each prediction based on the probability of attribute values given the class labels.

After selecting the desired options from Test Options and Weight Selector, click on the Optimize button. It will evaluate the data read into the Tool with genetic algorithm and KNN to produce the optimum value for k as well as a weight vector for the attributes. Results obtained with relevant to the optimization process could be viewed through the View Results button in the Training Phase section.

The model is automatically saved as an xml file with the optimized k value and the weights for each attribute. The location of the file is the same as where the training dataset is located and with the same file name as the training dataset with *.prm* extension.

The classifier could be re-evaluated with the Train set or with a separate test set. If you need to re-evaluate the model with the Train set simply click on the Evaluate button without changing any other option. If you need to re-evaluate the model with a separate test set, select the Test set option from the Test Options and choose another

known set from the same domain as the initial dataset (Train Set). Results could be viewed through the `View Results` button in the `Training Phase` section.

Prediction

From the `Prediction Phase` section of the `Prediction` window choose the dataset with unknown class labels. You can predict the class labels through `Predict Class Labels` and view the churn results through `View Results` button.

View Results

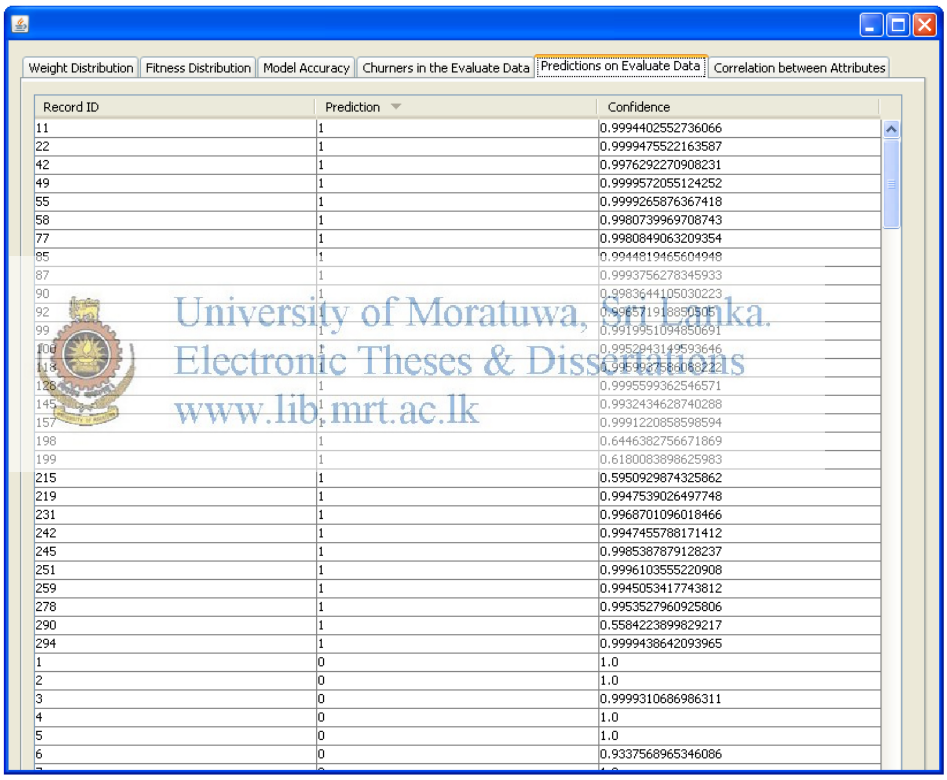

Figure A-5 Result analyzer

Results could be viewed for different stages of the process. Viewing the results under the `Training Phase` provide the distribution of attribute weights, fitness distribution and evaluated predictions. By selecting the required tab you can easily get the relevant results.

`Weight distribution` shows the importance that has on each attribute when training the classifier. Attributes with high weight values are the most contributing attributes for the final prediction.

`Fitness distribution` shows whether the fitness value is stabilized for the `Number of Evolutions` specified in `Prediction` window. The model provides better results if the fitness value is stabilized at a maximum value.

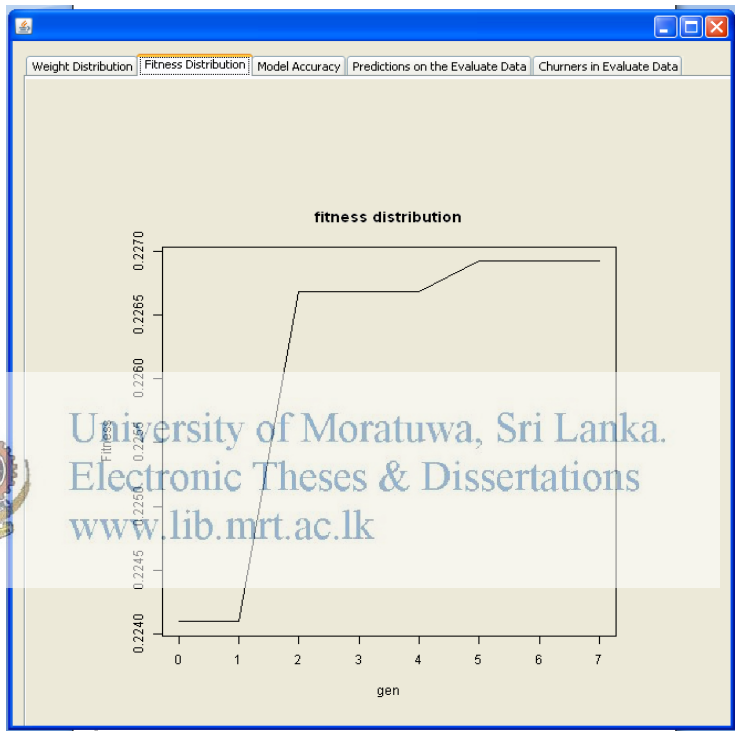

Figure A-6 Fitness distribution

Accuracy of the model could also be viewed in terms of four criteria (Figure A-7).

'`Churners in the evaluated data`' tab shows the predicted churners along with their details which were considered for the prediction. You can by examining the trend in data under '`Visualize All`' section, edit the values of attributes of the churners. Edited file could be saved and re-used for prediction.

`Prediction phase` allows you to load an existing model to predict churners. Select the Load Model button to select the saved model. The model loading should be saved as an xml file in *.prm* extension to be able to use with the Tool.

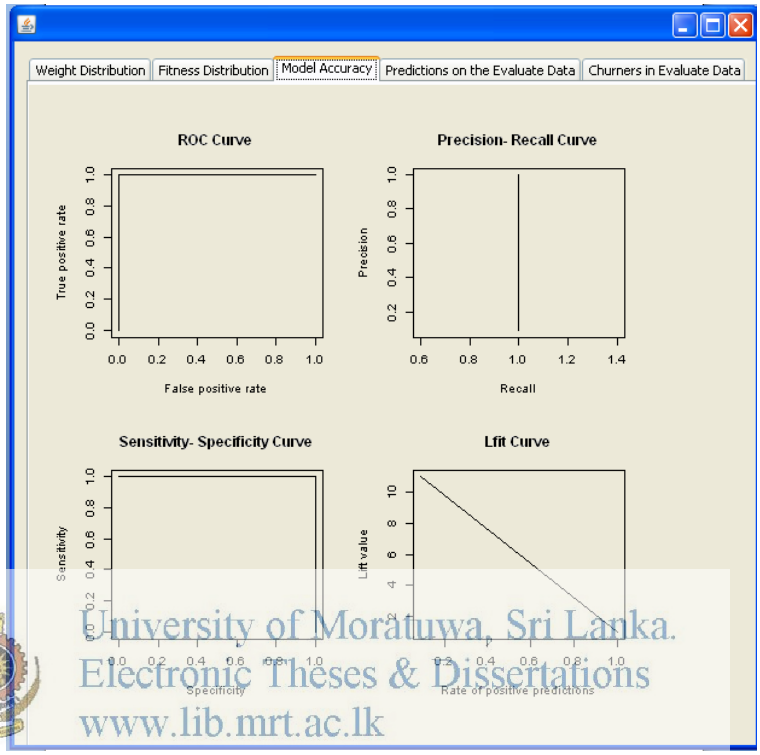

Figure A-7 Model accuracy

APPENDIX B – Source Code

The source code of the Tool could found in the CD attached with the title Appendix B.