

CUSTOMER CHURN PREDICTION FOR TELECOMMUNICATIONS INDUSTRY

YABAŞ, UTKU

JANUARY 2014

CUSTOMER CHURN PREDICTION FOR TELECOMMUNICATIONS INDUSTRY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF
NATURAL AND APPLIED SCIENCES OF
IZMIR UNIVERSITY OF ECONOMICS

BY
YABAŞ, UTKU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE
IN THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

JANUARY 2014

M.S. THESIS EXAMINATION RESULT FORM

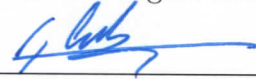
Approval of the Graduate School of Natural and Applied Sciences



Prof. Dr. Cüneyt Güzeliş
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.


Prof. Dr. Turhan Tunalı
Head of Department

We have read the thesis entitled “**Customer Churn Prediction for Telecommunications Industry**” completed by **YABAŞ, UTKU** under supervision of **Assoc. Prof. Dr. Türker İnce** and **Asst. Prof. Dr. Hakkı Candan Çankaya** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.


Asst. Prof. Dr. Hakkı Candan Çankaya
Co-Supervisor


Assoc. Prof. Dr. Türker İnce
Supervisor

Examining Committee Members

Date: 23/01/2014

Assoc. Prof. Dr. Türker İnce
Electrical and Electronics Eng. Dept., IUE

Prof. Dr. Cüneyt Güzeliş (Chair)
Electrical and Electronics Eng. Dept., IUE

Assoc. Prof. Dr. Süleyman Ünlütürk
Software Eng. Dept., IUE

Asst. Prof. Dr. Hakkı Candan Çankaya
Computer Engineering Dept., SMU, USA

Asst. Prof. Dr. Berna Özbek
Electrical and Electronics Eng. Dept., IYTE






ABSTRACT

CUSTOMER CHURN PREDICTION FOR TELECOMMUNICATIONS INDUSTRY

YABAŞ, UTKU

MS in Intelligent Engineering Systems

Graduate School of Natural and Applied Sciences

Supervisor: Assoc. Prof. Dr. Türker İnce

Co-Supervisor: Asst. Prof. Dr. Hakkı Candan Çankaya

January 2014

Customer churn is a concern for telecommunication service providers due to its associated costs. In this thesis, we analysed state-of-the-art data mining algorithms and developed novel methods to accurately predict customers who will change and turn to another provider for the same or similar service. We extensively evaluated performance of our proposed approach using a public and real dataset compiled by Orange Telecom for the Knowledge Discovery and Data Mining (KDD) 2009 Competition. This dataset has 100,000 instances with 230 attributes, which makes it a “big data”. IBM achieved the highest score on this dataset requiring significant amount of computational resources. We aimed to find alternative methods that can match or improve the recorded highest score with more efficient use of resources. In our study, we focus on ensemble of classifiers techniques. We compared performance of single, powerful classifiers to state-of-the-art ensemble methods for churn detection problem. Additionally, we showed that these results can be further improved by combining selected subset of well performing classifiers by a voting classifier. Overall, the results with our proposed approach were similar to the official top scorers of the competition. We believe that our proposed approach can be valuable for solving other challenging machine learning problem domains (such as “big data” problems) rather than churn prediction. Also, we performed experiments using the selected datasets from the UCI Machine Learning repository. Our proposed approach outperforms the single powerful algorithms contained in the ensemble for most of the datasets tested.

Keywords: churn prediction, ensemble classifier, voting classifier, data mining, machine learning.

ÖZ

TELEKOMÜNİKASYON SERVİSLERİNDEN ABONELİKLERİNİ İPTAL EDECEK MÜŞTERİLERİ ÖNCE DEN TAHMİN ETMEK

YABAŞ, UTKU

Akıllı Mühendislik Sistemleri, Yüksek Lisans

Fen Bilimleri Enstitüsü

Tez Danışmanı: Doç. Dr. Türker İnce

İkinci Tez Danışmanı: Yrd. Doç. Dr. Hakkı Candan Çankaya

Ocak 2014

Müşteri kaybetmek, telekom firmaları açısından kaybettirdiği para bakımından önemli bir endişedir. Bu tez çalışmasında, en son veri madenciliği yöntemlerini analiz ederek, servislerden ayrılacak veya başka bir firmanın servisini kullanmayı düşünen müşterileri tahmin etmek için yeni metotlar geliştirdik. Önerdiğimiz yaklaşımın performansını yoğun bir şekilde değerlendirdik. Bu değerlendirmeyi yapmak için Orange Telecom tarafından “Knowledge Discovery and Data Mining 2009”(KDD) yarışması için sunduğu gerçek ve kullanıma açık bir veri kümesi kullandık. Bu veri kümesinde toplam 100.000 örnek ve 230 değişken bulunmaktadır. Bu yüzden veri kümesi “büyük veri” kapsamına girmektedir. IBM bu yarışmada birinci olmuştur, ancak önemli ölçüde bilişimsel kaynak kullanmaktadır. Biz alternatif metotlar ve daha uygun kaynaklar kullanarak, yarışmadaki en yüksek skorlara ulaşmayı hedefledik. Bu çalışmada, toplu sınıflandırıcı teknikleri üzerine yoğunlaştık. Tek ve güçlü sınıflandırıcılar ile en son toplu sınıflandırıcıları “müşteri ayrılma” problemi için karşılaştırdık. Ayrıca, bu metotların performanslarını arttırmak için iyi performans gösteren sınıflandırıcıları seçerek; bunları oylayıcı sınıflandırıcı ile birleştirdik. Genel olarak, elde ettiğimiz sonuçlar, yarışmanın en yüksek sonuç alan resmi yarışmacıları ile yakındı. Önerdiğimiz yaklaşımın, “müşteri ayrılması tahmini” dışındaki başka zorlayıcı otomatik öğrenme problem alanları için de değerli olabileceğine inanıyoruz. Yöntemimizin doğruluğunu onaylamak için, UCI Machine Learning kütüphanesinden topladığımız veri kümeleri ile deneyler yaptık. Bu deneyler sonucunda çoğu veri kümesinde yöntemimiz, içinde bulunan toplu sınıflandırıcıdaki bütün algoritmalarından daha iyi sonuçlar elde etmiştir.

Anahtar Kelimeler: müşteri kayıp tahmini, çoklu sınıflandırıcı, oylayan sınıflandırıcı, otomatik öğrenme, veri madenciliği.

ACKNOWLEDGEMENT

I wish to express my deepest gratitude to my supervisors, Assoc. Prof. Türker İnce and Assist. Prof. Hakkı Candan Çankaya for their guidance, advice, criticism, insight throughout the research.

I must state my thanks to my supervisor Dr. Türker İnce for his thoughts, guidance and comments especially on the field of Machine Learning.

I must express my thanks to Assist. Prof. Hakkı Candan Çankaya for his support, never ending encouragement and advices, even from the overseas.

I am thankful to my family; Uğur Toprak Yabaş, Günay Yabaş and Muhip Yabaş for their encouragement and believing in me all my life.

I would also like to thank to my dearest wife Selen Yalçın Yabaş for her all supports during the process of this thesis.

TABLE OF CONTENTS

Front Matter	i
Abstract	iii
Öz	iv
Acknowledgement	vi
Table of Contents	ix
1 Introduction	1
2 Churn Prediction	4
2.1 What is Churn?	4
2.2 Churn Prediction Problem	6
2.2.1 Churn Prediction as a Classification Problem	6
2.2.2 Challenges in Churn Prediction	7
2.2.3 Evaluating Churn Prediction System	8
3 Technical Background	16
3.1 Literature Survey	16

3.2	Classifiers	21
3.2.1	Decision Tree	22
3.2.2	Logistic Regression	22
3.2.3	Random Forest	24
3.3	Machine Learning and Programming Tools	25
3.3.1	Weka	25
3.3.2	Java	26
3.3.3	Regular Expressions and Text Editors	26
3.3.4	Rapidminer	26
4	Methods	27
4.1	The Proposed Approach	27
4.1.1	Dataset	28
4.1.2	Feature Selection and Preprocessing	33
4.1.3	The Meta-Classifier	35
4.1.4	The Proposed Ensemble Classifier Approach	36
5	Experimental Evaluation and Results	40
5.1	Results on Churn Prediction Datasets	41
5.2	Test Framework	45
5.2.1	Multi-processing	48
5.2.2	Experimental Results from UCI Benchmark Datasets	48

5.3	Limitations	62
6	Conclusions	64
A	Numeric Variables in KDD 2009 Dataset	66
B	Evaluations with Number of Algorithms	78

LIST OF TABLES

4.1	Statistics of Some Numeric Variables	30
4.2	Statistics of Some Categorical Variables	32
4.3	Preprocessing - Classifier pairs : Meta-Classifiers	36
5.1	Preprocessing - Classifier Pairs and ROC Scores	42
5.2	ROC Area of the Final Model	43
5.3	Comparison of Results with Other Methods for KDD 2009 Dataset	43
5.4	Best Meta-Classifiers for SGI	45
5.5	UCI Datasets Used in Tests	47
5.6	Algorithms Used in Tests	47
5.7	ROC Results on UCI datasets	52
5.8	ROC Results on UCI datasets <i>continued</i>	53
5.9	F-measure Results on UCI datasets	55
5.10	F-measure Results on UCI datasets <i>continued</i>	56
5.11	Accuracy Results on UCI datasets	58
5.12	Accuracy Results on UCI datasets <i>continued</i>	59

A.1	Statistics of All Numeric Variables	67
B.1	Effects of Number of Algorithms, ROC Area	79
B.2	Effects of Number of Algorithms, ROC Area <i>continued</i>	80
B.3	Effects of Number of Algorithms, F-measure	81
B.4	Effects of Number of Algorithms, F-measure <i>continued</i>	82
B.5	Effects of Number of Algorithms, Accuracy	83
B.6	Effects of Number of Algorithms, Accuracy <i>continued</i>	84

LIST OF FIGURES

2.1	Churn Prediction as Data Mining Problem	7
2.2	Confusion Matrix	9
2.3	Fit, Under-fit and Over-fit Model [32]	11
2.4	Three-Fold Cross Validation [40]	13
2.5	ROC Graph with 5 Discrete Classifiers	14
2.6	A Sample ROC Curve [24]	15
4.1	The Proposed Churn Prediction System Design	29
4.2	Histograms of some variables	32
4.3	High Level Design of Meta-Classifiers	36
4.4	High Level Design of a Voting-Classifier	38
5.1	Class Diagram related with Classifiers	46
5.2	Effects of Number of Algorithms, ROC Area	61
5.3	Effects of Number of Algorithms, ROC Area 2	61
5.4	Effects of Number of Algorithms, F-measure	62
5.5	Effects of Number of Algorithms, F-measure 2	62

5.6	Effects of Number of Algorithms, Accuracy	63
5.7	Effects of Number of Algorithms, Accuracy 2	63

Chapter 1

Introduction

Rapidly changing technology in wireless infrastructure, like LTE, variety of hand-held smart devices, and innovative apps in Mobile communication world make service subscriber retention a competitive effort. Subscriber churn is a concern of customer care management for most of the mobile and wireless service providers and operators due to its associated costs. Especially, in saturated wireless communications market, there are incumbent service providers and small operators offering promotions and packages for subscribers who would like to change-and-turn (churn) to their own services. Wireless service providers have to have strategies and counter promotions for potential churners as it is more expensive earning a subscriber back once s/he unsubscribes the service. Therefore, subscriber churn awareness lands on the roadmaps of many wireless and mobile service providers for their survival.

In this study, we concentrate on evaluation and analysis of performance of different machine learning and data mining methods for accurate churn prediction. We analysed state-of-the-art data mining algorithms and developed novel methods to accurately and efficiently predict subscribers who will change-and-turn (churn) to another provider for the same or similar service. In 2009, the French telecom company Orange sponsored a competition in Knowledge Discovery and Data Mining (KDD), and posted three datasets [1] compiled from real

customers. One of the dataset was churn analysis and prediction in wireless service providers' domain. They provided a sample dataset from their wireless and mobile subscribers filtered transactions to be used in the competition. In our study, we developed and tested various algorithms over this dataset and used it in our experiments. Although the competition is over, platform is still open for submissions and data is still available for further research towards improvements and other interpretations. Evaluations for the ranking consider the area under the Receiver Operating Characteristic (ROC) curve. ROC area is a standard scoring for the problems where classes are highly skewed. It is a scoring that is independent from the class ratios. IBM achieved high scores on this dataset requiring a significant amount of computing resources [10].

In this study, we aimed to develop alternative data mining methods that can match or improve the recorded high scores with more efficient and practical use of resources. We have concentrated on ensemble classification methods which encompass single methods to improve the solution to the churn prediction problem. Then we propose our own approach to solve churn prediction problem. Our approach considers different classifier - preprocessing pairs. It is based on the various single powerful classifier algorithms together with own preprocessing methods, which are called meta-classifiers. We build meta-classifier models from different algorithms. We choose best performing meta-classifiers for our final voting classifier. The results from preliminary application of our proposed approach to churn prediction problem were promising [42, 43].

The first place in the KDD 2009 Competition in churn problem solutions is still held by IBM [3]. Our proposed approach scored close to the top scores of the KDD 2009 Competition with motivating results and less computational resources. We have further investigated our proposed approach on different benchmark datasets gathered from UCI Machine Learning dataset repository [41]. These datasets represent real-world problems with low-dimensional and limited training sets as opposed to large and high dimensional data of churn prediction problem. Results are analysed and interesting conclusions have been found after intensive benchmark tests.

This thesis is organized as follows; Chapter 2 briefs about the definition of churn, churn prediction problem, methods used and challenges. Chapter 3 provides some background on related or similar studies on churn prediction, followed by the tools we have used. In Chapter 4, KDD 2009 dataset is analysed and we explain our proposed approach for churn prediction problem. Chapter 5 illustrates our benchmark tests we did for the evaluation of our method on different datasets. Our method is compared and contrasted with other methods. We conclude the thesis in Chapter 6 by giving a direction for our future studies, and improvements for our system.

Chapter 2

Churn Prediction

2.1 What is Churn?

Over the last two decades mobile telecommunications become the dominant communication medium. The market has reached a degree of saturation where each new customer must be won over the competitors. Since the cost of winning a new customer is far greater than the cost of preserving an existing one. Churn for telecommunication can be defined as, subscriber who has a contract with a service provider quits the contract and makes a contract with another service provider for a same or similar service because of a better deal or some other reason.

There are three types of churn which are:

1. **Active:** the customer decides to quit
2. **Rotational:** the customer quits contract without the aim of switching to a competitor.
3. **Passive:** the company discontinuous the contract itself.

We are interested in the first two types of the churn. The basis of the churn

detection is historical data containing information about past churners. A comparison is made between these churners and existing customers. As likely churners are identified customers for which the classification suggests similarly to prior churners.

The dataset used in churn prediction includes:

1. **Customer behaviour:** identifies the parts of service a customer is using and their frequencies. For example; the provider can track number and length of calls, period between calls, the usage of the network for data exchange, etc.
2. **Customer perceptions:** Customer perceptions are defined as the way a customer apprehends the service. They can be measured with customer surveys and include data like overall satisfaction, quality of service, problem experience, satisfaction with problem handling, pricing, locational convenience, image/reputation of the company, customer perception of dependency to the vendor, etc.
3. **Customer demographics:** are some of the most used variables for churn prediction. They include age, gender, level of education, social status, geographical data, etc.
4. **Macro-environment variables:** identify changes in the world, different experiences of customers, which can affect the way they use a service. For example; people who have survived a natural disaster and could rely on their mobile phones during the event are most likely to continue using the service.

The size of gathered data is usually very large, with large number of attributes (features) which results in high dimensionality, hence making it a complex and challenging "Big Data" problem. One approach is to use data reduction (such as principal component analysis) and feature selection (such as mutual information feature selection) techniques.

The methods used in churn prediction are regression analysis, Naïve Bayes classifier, Decision Trees, Artificial Neural Networks, Support Vector Machines and many more. These algorithms are detailed in Mitchell’s book on “Machine Learning” [35]. We describe some of the specific algorithms we used in Chapter 3 and Chapter 4.

2.2 Churn Prediction Problem

Churn prediction problem can be defined as follows; customer data on past calls for each mobile subscriber between a specific time period, together with all personal and business information is maintained by the carrier. In addition, for the training phase, labels are provided in the form of a list of churners together with their corresponding churn dates. Given this data consist of previous churners and non-churners, patterns and similarities between them are tried to be discovered. These patterns are applied on current customers to predict future churners so that they can be contacted before they churn.

2.2.1 Churn Prediction as a Classification Problem

Problem of churn prediction is a data mining problem. To predict churners in telecommunications services, first we need previous customer records of churners and non-churners. Dataset needs to be gathered from these records. Data gathering is not in the scope of this work. We assume that we have the following dataset. Each customer i , should be represented with a vector x_i with n attributes; $x_i^1, x_i^2, x_i^3, \dots, x_i^n$, labeled by a value y which in our case; $y = \{churner, non - churner\}$. y is a Boolean valued target function. Data is used to build prediction models using a machine learning algorithm. Prediction model is developed to predict the current customers if s/he is a future churner or not. Prior to the training stage, data may need preprocessing to help machine learning algorithms to train models. Pre-processing step we applied is explained in Chapter 4. The model is tested using a separate independent dataset in which

classes of each instance are known. Test dataset can also be gathered by splitting the training dataset. Model is applied to the test dataset to predict the churners in this dataset. We know the labels of the test dataset. Therefore, we can see how well the model does perform. The test dataset has to be different than the training set. Otherwise the performance of the model would be unrealistic. Model will definitely perform better on the dataset that it had been developed and trained on than the completely different dataset. This process can be seen in Figure 2.1. There is another testing method called 10-fold cross validation which is used to test our proposed method on other well known datasets from UCI. The k-fold cross validation is explained in Section 2.2.3.1.

Churn Prediction is a 2-class classification problem with a boolean-valued target function: cherner (1) and non-cherner (0). There are many machine learning algorithms available which can be applied to churn prediction problem. However, there is no single golden method that is better than others in all indicators, because accuracy and concision cannot appear in a single method simultaneously. Therefore, we have focused on ensemble of classifiers. Our proposed approach is described in detail in Chapter 4.

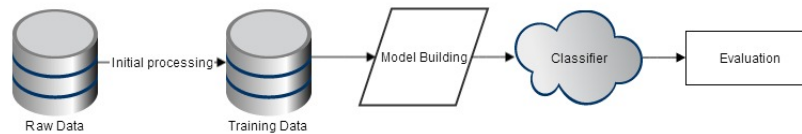


Figure 2.1: Churn Prediction as Data Mining Problem

2.2.2 Challenges in Churn Prediction

Data needed for the training is gathered from telecommunication companies. Records of the users are usually very big. Hundreds of millions of transactions are recorded everyday. These transactions and user information are stored in databases consist of thousands of tables and millions of rows. This makes churn prediction challenging task. Not all of these data is relevant with the churn.

Database has to be processed and relevant tables should be gathered. However, data will still be huge with large number of features (columns) and instances (rows). This makes churn classification problem using classical methods very expensive in computing resources. Therefore, churn prediction problem which consists of analysing a very large and high dimensional data is a real "Big Data" problem where special, more efficient data mining algorithms should be used.

Another important problem with churn prediction is that churn data is usually skewed or imbalance. Skewed data means, majority classes (non-churner) are way higher in number than minority classes (churner). In our case ratio between the minority and majority classes are only less than 1 : 10. Less than 1 in every 10 customers churn. This makes evaluation harder. Algorithms used on classification are biased through the majority classes due to class imbalanced. There is very small information on minority classes. Non-churner patterns must be learned from these small number of non-churner instances. Some preprocessing techniques can be used for this situation. Resampling method is a simple and effective method to bias the classifier through minority class. Resampling used in our approach explained in Chapter 4. Without resampling, some classifier models classify all instances as non-churner. If the algorithm classifies all instances as "non-churner", classifier would have more than 90% accuracy. Classifying all instances as non-churner result in high accuracy, but obviously it is useless. Classifier cannot find any churners which is our main purpose. Different evaluation metric is needed for skewed data which is explained in Section 2.2.3.

2.2.3 Evaluating Churn Prediction System

In complex data mining tasks like churn prediction, researchers cannot just choose their favourite classifier and apply to the problem. Not all algorithms can be applied to all problem domains. Many machine learning algorithms should be executed with different settings to build many different models (classifiers). After these models are build, they needed to be compared and contrast to choose the best one for the prediction task. We need a score metric that can tell us how well the algorithm is predicting and compare the models. Methods to compare and

evaluate the classifiers will be discussed but first some performance metrics need to be clarified.

When a new instance in the test dataset comes, the classifier predicts it. If the prediction is correct (same value as the label), it is counted as true. If the prediction is wrong, it is counted as false. Prediction is counted as **true negative** if the prediction is negative (non-churner) and correct, **true positive** if the prediction is positive (churner) and correct, **false negative** if the prediction is negative but the actual class is positive and **false positive** or false alarm if the prediction is positive but the actual class is negative. Two by two confusion matrix can be constructed from a classifier and a set of test instances as in Figure 2.2. This matrix is basis for many common metrics.

	Actual Churners	Actual Non-churners
Predicted Churners	True Positive	False Positive
Predicted Non-churners	False Negative	True Negative

Figure 2.2: Confusion Matrix

The definitions of most common performance metrics based on the confusion matrix which are used to evaluate machine learning algorithms are given below:

- P =Number of Actual Positives
- N =Number of Actual Negatives
- TP =True Positives
- TN =True Negatives
- FP =False Positives
- FN =False Negatives
- $fprate$ = False positive rate = $\frac{FP}{N}$

- $tprate = \text{true positive rate} = \frac{TP}{P}$
- $Precision = \frac{TP}{TP+FP}$
- $Recall = \frac{TP}{TP+FN}$
- $Accuracy = \frac{(TP+TN) \times 100}{TP+FP+TN+FN}$
- $F - measure = \frac{2 \times precision \times recall}{precision + recall}$

Some measures that can be used to evaluate different classifiers includes accuracy, F-measure, lift chart, ROC area etc. Each represents a different approach to evaluating the performance. Before we choose a measure, we first need to clarify what the performance means to our application. Performance changes according to the data properties and problem domain. For example, there are two systems that are predicting forest fires. In a time span of a month, forest is observed. There were 5 fires. First system predicted 8 fires, 4 of them being true positives and 4 of them being false positives. Second System predicted 15 fires, 5 of them being true positives, 10 of them being false positives. Which system is better? System 1 is better if we measure with accuracy but it misses a fire and a cost of a fire is too high. We cannot risk missing a fire. 10 false alarms can be tolerated for correct prediction of a 1 more fire.

There are two other practical issues that need to be explained and we will see some occurrences of these in later chapters. Over-fitting and under-fitting are encountered often, not just in churn prediction but in all classification tasks. Over-fitting happens when the classifier specializes too much on the training dataset [33]. This usually happens when the classifier model or structure is overly complex hence it is severely affected by the noise in data due to over-parameterization. Therefore, it performs poor on the test or the new dataset. Patterns it discovers are too specific and complex. It discovers patterns that do not actually in the data. Classifier may adjust to very specific random features of the training data; features that have no causal relation to the target function. Therefore, classifier's performance is quite well on seen examples (training dataset) but poor on new data (test dataset). The other issue is under-fitting

which is just the opposite of over-fitting. Under-fit classifier model cannot fit to data. Because, it is too general or simple even for training data. Under-fitting is easier to discover because, classifier performs poor on training data. The k-fold cross validation can reduce the problem of over-fitting. Figure 2.3 is an example for these issues. Figure show the data points (a) of the dataset, an under-fit model (b), fit model (c) and over-fit model (d).

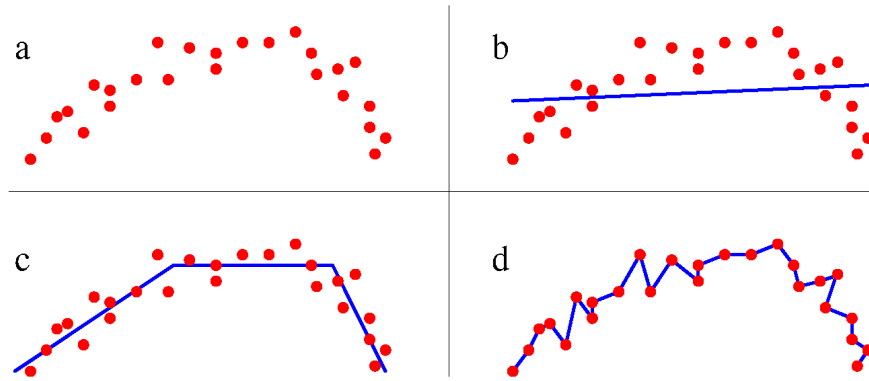


Figure 2.3: Fit, Under-fit and Over-fit Model [32]

In churn domain, we should be able to predict as much churners as possible and we must not misclassify many non-churners as churners. There will be some cost to get a churner back. If system, classifies non-churner as churner (false positive), cost of a retaining customer back will be paid for non-churning customer. On the other hand, if a churner is classified as non-churner (false negative), the company will not try to get the customer back. Hence, company will loose the customer resulting in more cost for the company. For a telecom company, the cost of retaining a current customer is lower than acquiring a new customer.

True positive rate is the ratio between the number of true churners which are classified as churners (true positives) and number of actual churners ($Positives = TP + FN$). False positive rate is the ratio between the number of predicted churners that are actually non-churners (false positives) and the number of actual non-churners ($Negatives = FP + TN$). $tprate$ and $fprate$ are not individually enough. **Precision** and **recall** are two measures that can be used in imbalanced datasets. **Precision** is the portion of the customers correctly classified as churners to all customers that are classified as churners. **Recall** is the portion of the

customers correctly classified as churners to all churners. **Precision** answers the following question: “Among the customers that are classified as churners, how many of them are actually churners” and **Recall** answers “Among all churners, how many of them the classifier could predict correctly”. There is always a trade-off between precision and recall. High precision results in low recall. **F-measure** considers both precision and recall. Therefore, it is a better measurement for problems that suffer from class imbalance. For the churn prediction domain and our dataset, the most appropriate measure is the *ROC area* which will be discussed on the next subsection. It is widely used in telecommunications because this measure considers both true positive and false positive rates. ROC area score is independent from class ratios. The KDD 2009 competition which have used the dataset we used also uses ROC curve for scoring. Therefore, we have chosen the same metric for evaluating our models to solve our main problem in addition to other common metrics.

2.2.3.1 Cross Validation

Cross validation is about selecting a hypothesis out of a set of hypotheses each resulting from an application of a learning algorithm to set of training data. The common approach is to test all hypotheses on a separate (independent) set of data and select the hypothesis which gives lowest error. According to the k -fold cross-validation technique, the original sample is randomly partitioned into k equal size sub-samples. Of the k sub-samples, a single sub-sample is retained as the validation data for testing the model, and the remaining $k-1$ sub-samples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k sub-samples used exactly once as the validation data. The k results from the folds can then be averaged (or otherwise combined) to produce a single estimation. The advantage of this method over repeated random sub-sampling is that all observations are used for both training and validation, and each observation is used for validation exactly once [38]. We used 10 as k which is also the most common value for k used in the literature [28, 40]. Cross-validation is analysed very well by Refaeilzadeh et al [40]. They show three-fold cross validation with a figure, see Figure 2.4.

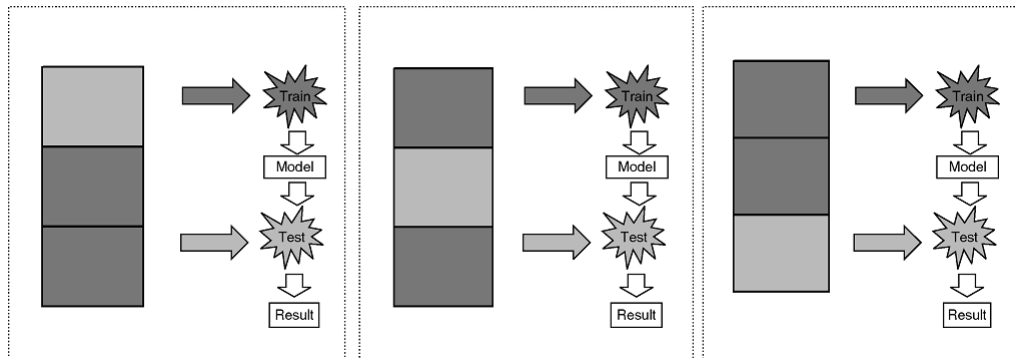


Figure 2.4: Three-Fold Cross Validation [40]

Procedure of the ten-fold cross validation can be defined as follows;

- Let n be the number of instances in the data.
- Break data into 10 sub-samples of size $\frac{n}{10}$.
- Train on 9 sub-samples and test on the remaining 1.
- Repeat the process 10 times and take the mean of the evaluation.

Ten-Fold cross validation technique is used in our experiments, see Chapter 5. It is widely used in data mining community to compare models or evaluate the performances of classifiers, especially when data is limited.

2.2.3.2 ROC Area Evaluation

ROC stands for a Receiver Operating Characteristics graph. ROC is a technique for visualising, organizing and selecting a classifier based on their performance [24, 26, 34]. ROC area is very useful evaluation method because it has properties that make it useful for skewed class distribution. ROC graphs are two-dimensional graphs in which $tprate$ is plotted on the Y axis and $fprate$ is plotted on the X axis. A classifier's $tprate$ and $fprate$ is plotted on this graph. An example with 5 classifiers of this graph can be seen on Figure 2.5.

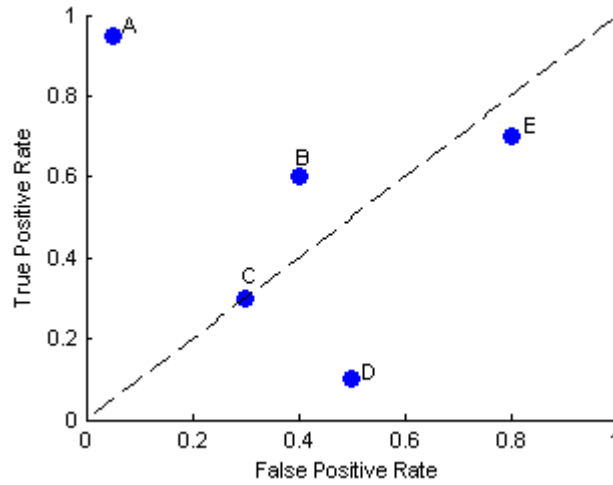


Figure 2.5: ROC Graph with 5 Discrete Classifiers

There are some important points to mark. If a classifier predicts all instances as non-churner (negative) then, it will be on the point $(0, 0)$. If a classifier predicts all instances as churner then, it will be on the point $(1, 1)$. $(0, 1)$ would be a perfect classifier. Classifier is better than another one, if it is on the north west of the other classifiers [26]. The diagonal line $y = x$ represents the random classifier. A classifier is expected to do better than a random classifier. The classifier A is the best classifier in Figure 2.5. It predicts most of the positive instances correctly without making too many false positives. Classifier E also predicts most of the positives correctly, but it predicts too many instances as positive which results in an increase in false positive rate.

Discrete classifiers have only one confusion matrix so they yield to a single point on the graph. Probabilistic classifiers outputs a probability which represents the instance's membership score to the positive class. If this score is higher than the threshold then classifier outputs the positive class. By varying the threshold, different confusion matrix would be generated, each of them representing a point on the graph. Each discrete classifier also produces a value that represents the degree of membership to target class. Degree of membership values might not be strict probabilities. For example, decision trees can output its confidence by looking at how many of positive and negative instances appear on the leaf node on training set. Higher the probability value represents higher confidence to

the membership to the class. If these probability values are normalized, we can have a probabilistic classifier. Now, it is possible to vary the threshold and plot the classifiers on the ROC graph. From these points a curve is drawn and the area under this curve is the area under the ROC curve score or area under the curve (AUC). Main advantage of the ROC graphs in churn prediction problem is that they enable visualizing of the classifier performance without regard to class distributions [26]. An example ROC Curve is given in Figure 2.6. Each step on the graph is a point formed by varying the threshold. The figure is taken from [24].

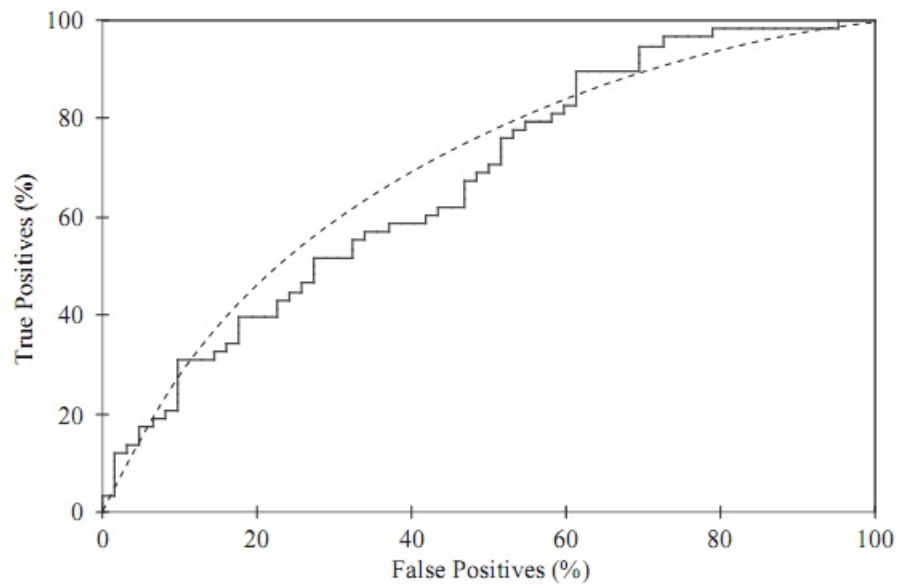


Figure 2.6: A Sample ROC Curve [24]

Chapter 3

Technical Background

3.1 Literature Survey

Since the topic of the thesis is a relatively new area, there are a limited number of good quality academic papers written on churn prediction in telecommunications industry. Most of the literature about churn prediction is about different fields and do not cover the problem we face in this thesis.

Lazarov et al. [22] explain the steps and methods of the churn prediction in a very nice and neat way. It tries to give a basic idea how the churn prediction is done. The paper begins by discussing the importance of the churn prediction in a world of ever growing competition on the market. The companies are trying to find churners because gaining a new customer is more expensive than the keeping the one that is going to leave. Paper continues on introduction by summarizing Van Del Poel and Larivieres work on economic value of customer retention. Not all customers bring the same profit by retention. Some customers are more worthy than others. Value of a customer is summarized as below:

- lowering the need to seek new and potentially risky customers, which allows focusing on the demands of existing customers;

- long-term customers tend to buy more;
- positive word of mouth from satisfied customers is a good way for new customers' acquisition;
- long-term customers are less costly to serve, because of a larger database of their demands;
- long-term customers are less sensitive to competitors' marketing activities;
- losing customers results in less sales and an increased need to attract new customers, which is five to six times more expensive than the money spent for retention of existing customers;
- people tend to share more often negative than positive service experience with friends, resulting in negative image of the company among possible future customers.

Paper is finalized by description of the structure of the Customer Relationship Management tools used today. Churn prediction should answer who?, why? and when? questions then actions should be taken for the profitable customers that are going to churn. They claim that churn prediction analysis is complex because data gathered is very large. Paper continues with describing how the quality of the predictor is measured. Paper discuss the most popular learners in churn prediction with their advantages and disadvantages, they are regression analysis, Naïve Bayes, Decision Trees and Artificial Neural Networks.

“Predicting Customer Churn in Mobile Networks through Analysis of Social Groups”, Yossi Richter, Elad Yom-Tov, and Noam Slonim [12] describe their method to predict churners which also take into consideration of the effect of a social group on a person in that group. The paper begins by providing information on how a good churn prediction system should be. Then, it briefly describes existing solutions and notes that all of them take a person as one but there can be a social group that can affect him/her and justifies this approach. Later in the introduction they describe the churn prediction and how it is done in real business. Their approach which is group-first approach is described. They built

a social network graph, nodes represent customers and edges are relations among customers. Later they weight these edges and these weights represent the strength of the link between customers. This information helps them to find the social groups and leaders of the groups. If the leader churns, s/he might persuade whole group to churn. Therefore, they calculate the customer value of possible churners so that mobile operator can take action on more valuable customers. Because, not all churners can be contacted on time. Paper continues with their experiments and the results. First, they describe their data. They had a 28 days data from a major mobile operator. Later, they indicate how they decide that the group will churn. They claim that possibility to churn for small groups are 2.7 times higher than for larger groups. On last section authors describe the possible reasons for churn which they found with their experiment. They conclude the paper with their results and execution times. They claim that their results were good. However, they did not provide much information with their results. Their execution time is worse than the traditional churn prediction methods. Authors conclude paper by supporting that this method can also be used for other goals in mobile operators other than churn prediction.

An Experimental Study on Four Models of Customer Churn Prediction - Zhu et. al. [15] try to experiment and compare four models used for churn prediction. Methods like Decision Tree, Artificial Neural Networks and Logistic Regression have been improved for many times and are difficult to become better. Their study tries to choose some novel models which can be improved and solve more complex customer churn problems. These four models are Bayesian networks, support vector machines, rough set and survival analysis. Paper first briefly describes each method with their advantages and disadvantages. Later on the paper authors explain, how these methods can be applied to this specific problem, churn prediction. The process applied in each method and their results are explained later on. For the experiments, authors describe how they compared the results. They compared the results in four different metrics. These metrics are; ROC curve, accuracy, captured response and lift value evaluation which are the main indicators to evaluate preciseness of the models. They claim that in overall Bayesian network performs better than all others by supporting this by results

and different evaluation metrics.

Wei et al. [13] try to predict potential churners at the contract level for a specific prediction time-period. They have a dataset from one of the largest service providers in Taiwan. They are using two types of data for churn prediction which are contractual data and call details of the subscribers. They do not consider the demographics data. Authors claim that Decision Tree and Decision Rule approaches are better than neural network on churn prediction. They've used Decision Tree over Decision Rule based on its popularity. Authors clarify that data is highly skewed. They use multi-classifier class-combiner approach to address this problem. They create S randomly partitioned training data subsets from the majority class examples. Minority class examples are replicated among all these subsets. Ratio of minority and majority instances are the same for each subset. Minority examples are the same in all subsets. S is decided by the desired ratio among the majority and minority classes. A Decision Tree is built from the each training subset. Now, number of Decision Trees are also S . These S classifiers perform a weighted voting for the prediction.

Ensemble methods are widely used in Data-mining community as well as for churn prediction systems. Verbeke et al. [28] experiments machine learning algorithms consist of both single and ensemble algorithms in a set of churn prediction datasets compiled from telecommunication service providers including KDD 2009 dataset. They propose a profit based evaluation function to choose best performing classifier. They use small number of variables and report that results are better than the classical evaluation methods. We have also compared our score on KDD 2009 and SGI datasets with their work.

Idris et. al. [29] propose an intelligent churn prediction system for telecom. They report that system uses Random Forest, Rotation Forest, RotBoost and DECORATE ensembles in combination with minimum redundancy and maximum relevance (mRMR), Fisher's ratio and F-score. System is tested on the KDD 2009 dataset and their results show that the system performs very well on it. Almost as well as IBM's score.

Another example of ensemble algorithms used applied to a different application area is by Coussement et. al. [30]. They apply some state-of-the-art ensemble algorithms for churn prediction in gambling industry. They claim that churn prediction is a valuable strategy to retain customers at risk. Many experiments reported on this paper. Methods they've applied are CART Decision Tree, generalized additive models, Random Forest and GAMens. They use real world dataset from gambling industry. Evaluation metrics used are top-decile lift and the lift index. They conclude their paper by referring that churn prediction to retain customers in gambling industry is highly profitable.

Rodriguez et al. [27] propose a ensemble method called "Rotation Forest". Rotation Forest is a meta-classifier. It randomly splits the dataset in to subsets and apply Principal Component Analysis in order to preserve the variability information. Decision Trees algorithm is used for model building. They compare Rotation Forest with Bagging, AdaBoost and Random Forest on a set of UCI datasets. We have also used Rotation Forest for our experiments in Chapter 5.

The studies most similar to ours are the studies on KDD 2009 Competition. Top scorers on this competition written a report that is published by the competition [2]. We will now briefly review their methods. IBM has their success from the ensemble selection. After some pre-processing, they've added more features with binning and prediction of Decision Tree with one or two features. Then, they built a library of classifiers. Many classifiers are put into this library even if their test performance was not good. Classifier library includes 500 to 1000 classifiers. They note that best performing classifiers in this library are boosted trees, followed by Logistic Regression and Random Forest. We also verified and got similar results with these classifiers. Once the library is generated, Ensemble Selection builds an ensemble by selecting from the library of classifiers that best performs on area under the ROC curve. Bagging is applied to the final models to reduce the variation. Chances of over-fitting increase if there are large number of classifiers to choose. Their method is classical data mining approach without needing of human expert to interpret. Main problem of this approach is significant amount of computer resources needed because of the large number of classifiers in the library. Our main goal is to propose a new data mining method

to improve IBM's method in terms of computational resources required while achieving comparable or close prediction accuracies. They claim that each fold of the cross validation took little more than one day with a cluster of nine dual Opteron nodes(2 GHz).

Second best scoring team on KDD 2009 Competition were ID Analytics [11]. They have used TreeNet algorithm which is also known as Classification and Regression Trees. Variable pre-processing is done in a very standard way. They've down-sampled the negative examples because of the huge number of negative examples in the dataset. Feature selection is applied with the TreeNet algorithm. Feature selection is done by removing variables one by one considering the variable importance from Breiman. They combine bagging and boosting with TreeNet as their final model to improve their results.

Third best scoring team were University of Melbourne [4]. Their approach needed fair amount of computational resources that can be easily found on today's personal computers. They ran their algorithms on 2.66 GHz Intel Core 2 Duo processor with 2 Gb of RAM. They have used shallow Decision Trees with boosting. Their primary reference is from the work of Freidman [17].

3.2 Classifiers

In this study, we have used variety of state-of-the-art powerful classifiers. Number of classification algorithms used are around 20 and total number of models trained are around 250. However, we include, in this manuscript, the subset we used in final model of our proposed ensemble classifier in Chapter 4. The selected subset of single classifiers are Decision Trees, Random Forest and Logistic Regression. Other significant algorithms include Artificial Neural Networks, SVM (Support Vector Machines), Rotation Forest, Naïve Bayes etc. ANN and SVM had a common disadvantage of long training time (due to large number of training examples) hence being very slow in model building [24, 35]. Their performance were rather poor in comparison to the other algorithms explained below and we

had to eliminate them from consideration during our studies. In Analysis of KDD 2009 [3], we saw that either top scoring teams have not used them or these algorithms did not perform well. The algorithms we have used in our final model are described in the following subsections.

3.2.1 Decision Tree

The ID3 and its successor C4.5 algorithms are the most popular decision tree algorithms and are widely used in machine learning [16,18]. The C4.5 algorithm recursively builds sub-trees by putting instances into tests and sum of all these sub-trees forms a single decision tree. The C4.5 demonstrates an improvement of ID3 decision tree algorithm from Quinlan [16,18,24,35]. Decision Tree has been widely used in churn prediction practices. For pruned decision trees, confidence factor can be parametrized. When deciding the classification for the leaf in the tree, confidence factor determines how many misclassified instances on the training set that leaf can tolerate. We have applied confidence factors of 0.25, 0.50, 0.60, 0.75 where higher confidence factors decrease the performance significantly. We have chosen to use confidence factors of 0.25 and 0.50 for pruned decision trees as the best practice. We also built un-pruned decision tree where its own results were good on the training set but, performed significantly worse on the test set compared to pruned ones. This shows that un-pruned tree over-fits the test set. In this problem single decision tree was not predictive enough. On the other hand, our observation is that ensemble of decision trees performs significantly better than using a single decision tree.

3.2.2 Logistic Regression

Linear Regression is a type of regression analysis, where data are modelled using linear predictor functions and unknown model parameters are estimated from the data [24]. It attempts to model the relationship between relevant predictive features and the target feature by fitting a linear equation to the observed data. Linear regression is usually fitted using least squares approach. For $j = 1, 2, 3, \dots, m$

features and $i = 1, 2, 3, \dots, n$ instances, linear regression can be expressed as:

$$y^i = w_0 + w_1 a_1^i + w_2 a_2^i + \dots + w_n a_n^i \quad (3.1)$$

where:

- y^i is the target label for the instance i
- w_0 is the constant parameter
- w_j is the weight of the feature j
- a_j^i represents the value of the feature j of instance i

Linear regression was the first regression analysis that has been studied and used in practice extensively. This is because models which depend linearly on their unknown parameters are easier to fit. Linear regression is best for the regression tasks where target feature is continuous. Linear regression can be used for churn prediction problem with a little modification; however, it is not a good algorithm for the churn prediction problem because it cannot predict categorical targets. Logistic regression [24] is a modified version of the linear regression for prediction tasks where target feature is categorical like in churn problem; namely, churner and non-churner.

By using the logistic regression, we modelled the relationship between the target feature and other features. Logistic regression replaces the target feature with; $P(1|a_1, a_2, \dots, a_k)$

Logistic regression builds a linear model based on transformed target feature. The final model is;

$$P(1|a_1, a_2, \dots, a_k) = \frac{1}{1 + e^{-w_0 - w_1 a_1 - \dots - w_k a_k}} \quad (3.2)$$

Similar to linear regression, logistic regression tries to find weights that minimize the training error in the dataset. Linear regression measures the weights by

using squared error and logistic regression uses the log-likelihood of the model. The log-likelihood of the model is;

$$\sum_{i=1}^n (1-x^i) \log(1 - P(1|a_1^i, a_2^i, \dots, a_k^i)) + (x^i) \log(P(1|a_1^i, a_2^i, \dots, a_k^i)) \quad (3.3)$$

Logistic regression technique tries to maximize the equation above by fitting the weights and is widely used for the churn prediction problem.

3.2.3 Random Forest

Random Forest is also another powerful machine learning algorithm from Breiman [9]. It builds a given number of decision trees with random features. At each split of a single decision tree, m number of random features are chosen and the best split on these m random features is used to split the node. Each single tree votes for the prediction. Most popular prediction among many decision trees is the output of the forest. It is the best performing base algorithm in our tests. It runs significantly faster than adaBoost algorithm and performs better. Some disadvantages of the Random Forest are that they sometimes over-fit and it is hard for a human to interpret.

The original Random Forest algorithm [23], grows many classification trees. To classify a new object from an input vector, the algorithm puts the input vector down on the each tree in the forest. Each tree determines a classification, and votes for that class. The forest chooses the classification that has the most votes over all the trees in the forest. Each tree grows as follows:

- If the number of cases in the training set is N , sample N cases at random with replacement, from the original data. This sample will be the training set for growing the tree.

- If there are M input features, a number $m \ll M$ is specified such that at each node, m features are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.
- Each tree is grown to the largest extent possible. There is no pruning.

Our implementation differs from the original random forest. We limit the depth of the trees by 15, 20 and 25: hence, did not grow the trees to the largest extent. In some cases Random Forest over-fits our dataset. We applied this limit to overcome the over-fitting problem. Limiting the depth of the trees also reduces the need for memory. Smaller trees need less space in memory.

3.3 Machine Learning and Programming Tools

3.3.1 Weka

(Waikato Environment for Knowledge Analysis) is a popular suite of machine learning software written in Java, developed at the University of Waikato, New Zealand. Weka is open source software issued under the GNU General Public License [7]. Weka can be downloaded for free from its website [6]. Weka supports standard data mining tasks, more specifically, data preprocessing, clustering, classification, regression, visualization, and feature selection. Weka is a standalone packet and can also be used with Java programming language. Weka normally do not support multi-threading. Therefore it can be very slow with big datasets. All the basic machine learning algorithms and most of the preprocessing methods we used have been implemented under this tool. Additionally, we have implemented some of our own preprocessing methods and voting classifiers using Weka's class structure. Our study includes the interpretation and the evaluation of the algorithms that are already implemented in Weka.

3.3.2 Java

All implementation had been done with Oracle's Java [36] programming language using Weka's libraries with Eclipse IDE [37]. Java's collections, string and concurrency libraries are used intensively.

3.3.3 Regular Expressions and Text Editors

When importing data to the Weka environment, data has to be in ".arff" format. We have done it with some regular expressions and text editors. Some of the preprocessing like replacing missing values is done with regular expressions.

3.3.4 Rapidminer

Rapid miner is a data mining and data analysis toolkit which includes a rich GUI [8]. Data-mining process can be generated by building pipelines and flows. We have not used this tool for development. However, we have used it for some analysis and data format converting operations. Rapidminer can generate rich analysis. Through these analysis, we have decided which variables to keep or remove.

Chapter 4

Methods

4.1 The Proposed Approach

In this study, we applied data mining algorithms for subscriber churn analysis and prediction; namely, data gathering and preparation, prediction model building, and model evaluation. Our algorithms were developed using the data available from the KDD 2009 Competition website [1] and we also performed testing on the SGI [39] churn dataset. The KDD 2009 dataset is the largest real, publicly available datasets for churn related problems. The data were collected from Orange Telecom wireless customer transactions. The properties of the dataset features are explained in the next subsection.

In order to develop our proposed methods and perform some experiments we had to make some necessary preprocessing of the data. KDD 2009 dataset is challenging. Preprocessing had to be done to convert the raw data to training data for classification. This processing step is explained on the next subsection. In usual settings a machine learning algorithm can be directly applied to this dataset for training. However, we propose to use a different approach based on meta-classifier to improve performance of individual, powerful classifiers. Each meta-classifier we propose encompasses both a preprocessing method and a model

built from the preprocessed dataset. Preprocessing which is applied by the meta-classifier may vary depending on the model built. After a set of meta-classifiers is tested, a combination is used in our final proposed model. To come up with a good performing meta-classifier, we search for a well-matched preprocessing and algorithm pairs according to their performance. In classical ensemble techniques, preprocessing applied to the dataset is same for all classifiers used. While a preprocessing technique can increase the performance of some algorithms, it may decrease the performance of others. Our approach uses the suitable preprocessing technique or techniques for each algorithm. In our proposal, the final classification is achieved by using a voting classifier technique. Classifier used in the voting classifier is chosen among the best performing meta-classifiers that were developed and tested. The decision can be made by applying a performance threshold. Alternative for that approach is to train each meta-classifier and include the top n meta-classifier to the voting classifier by looking at their performance scores. The procedures (pseudo-codes) of the above-mentioned two alternative approaches are given as Algorithm 2 and Algorithm 3. High level design of the proposed churn prediction system can be seen in Figure 4.1.

4.1.1 Dataset

The KDD 2009 dataset consists of a very large and high dimensional data which makes it very challenging for data mining application. It has 100,000 examples with 230 variables and is equally divided into training and test sets. Test set's labels have been removed. ROC area results of the classifier on the test set can be obtained by submitting predictions to the website. It was not possible to use domain expertise because customers' data was encrypted and variables' names were hidden. 190 variables are numeric and 40 variables are nominal. Dataset is suffering from class imbalance. Ratio of positive and negative instances is only less than 1 : 10. While some algorithms are not affected from this situation, some of them are highly biased through positive class. Resampling is a preprocessing technique to help these algorithms.

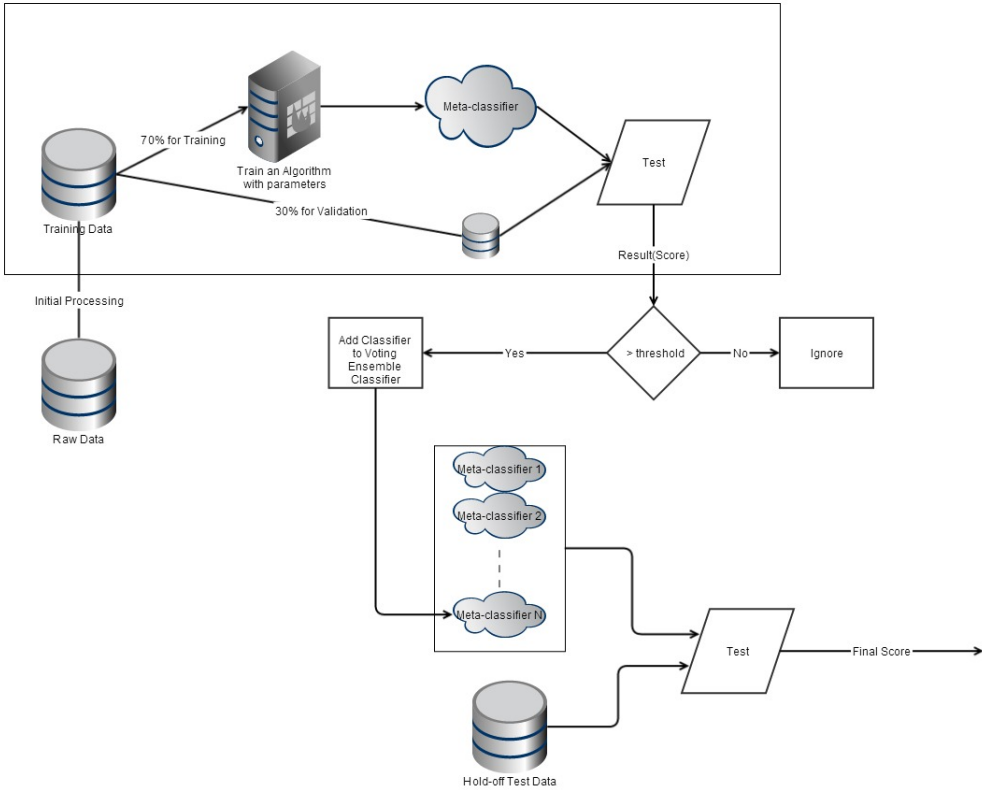


Figure 4.1: The Proposed Churn Prediction System Design

4.1.1.1 Numeric Variables

There were many problems with the numeric variables. There are 190 numeric variables in the dataset and all had been multiplied by a factor. None of the numeric variables was normalized with different dynamic ranges. They were not on the same scale. Some variables are scaled between 0 to 1, while others take value between 1 to 1000000. Example for this case can be seen between *Var138* and *Var5* on Table 4.1.

Table 4.1: Statistics of Some Numeric Variables

Name	Min	Max	Mean	Std. Dev.	Distinct	Missing
Var1	0	680	11.487	40.71	18	99%
Var2	0	5	0.00	0.14	2	98%
Var5	0	12325590	392606	928090	534	97%
Var8	NaN	NaN	NaN	NaN	NaN	100%
Var11	8	40	8.626	2.87	5	98%
Var25	0	13168	96.83	214.32	271	10%
Var32	NaN	NaN	NaN	NaN	NaN	100%
Var43	0	625	4.16	20.57	20	98%
Var50	0	18	0.09	0.88	4	98%
Var73	4	264	66.64	52.86	131	0%
Var87	0	28	5.42	5.25	5	99%
Var95	0	5640330	98671	180633	20002	45%
Var112	0	10352	66.22	157.64	230	10%
Var138	0	2	0	0.05	2	97%

Variables are polluted by high number of missing values and outliers. There are many attributes that have more than 90% of their values missing. There are some variables that are completely empty. Some variables only take few distinct values or all values are gathered around only one value, see Figure 4.2. Problems of numeric variables are summarized below:

- 19 attributes have all their values missing.
- 40 attributes have 99% of their values missing.
- 45 attributes have 98% of their values missing.
- 43 attributes have 97% of their values missing.

- 148 attributes have more than 90% of their values missing.
- 3 attributes contain only one value over all instances.
- Only one attribute have less than 1% of its values missing.
- 52 attributes have less than 10 distinct values.
- All numeric variables have different dynamic ranges.

Table 4.1 summarizes some of the statistics of the numeric values. Please refer to Table A.1 in Appendix A for full statistics of the numeric variables. Some examples from the Table 4.1 are that *Var8* and *Var32* have all of their values missing. Therefore, they are useless variables. Standard deviations are very large for the scales of the variables. *Var138* take only 2 distinct values over more than thousand instances which means that *Var138* is more like a nominal variable.

4.1.1.2 Nominal Variables

Nominal or categorical variables also have many issues. There are total of 40 nominal variables. Nominal variables do not have as much missing values as in numerical variables. There are only 6 variables with more than 90% of their values missing or that only take one value.

Some categorical variables have quite large vocabulary. Table 4.2 show an overview of some of the statistics of nominal variables. Table 4.2 shows the number of instances that have the value of the mode for that variable, number of distinct values and percentage of missing values for some categorical variables. As seen on the Table 4.2, *Var201* and *Var205* have more than 10,000 different categories. Problematic issues of the total 40 nominal variables are as follows:

- 17 variables have less than 10 categories, 3 of them take only 1 category.
- 9 variables have more than 10 less than 100 categories.

- 2 variables have more than 100 categories.
- 9 variables have more than 1000 categories.

Table 4.2: Statistics of Some Categorical Variables

Name	# of Instances at Mode	Distinct	Missing
Var191	1083	1	98%
Var199	955	5073	0%
Var201	12777	2	74%
Var205	31962	3	0%
Var214	73	15415	51%

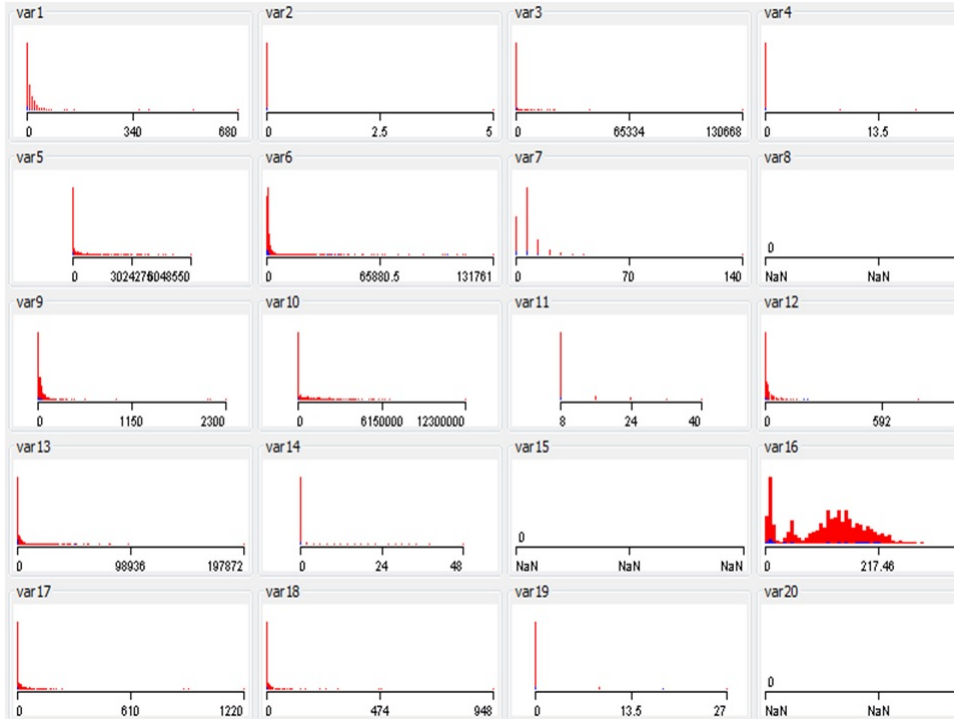


Figure 4.2: Histograms of some variables

We have also tried our method on a public artificial churn dataset from SGI. This dataset is much more structured than the KDD 2009 dataset and is not as challenging as KDD. It has 5000 instances and 21 attributes. 1700 instances were kept as held off instances for test. 3300 instances were used for training. Best performing models were chosen for the final model and tested on the hold off set of 1700 instances. Four attributes were removed from the dataset because they were 100% correlated with an another attribute.

4.1.2 Feature Selection and Preprocessing

There are two preprocessing steps in our approach. The first one is applied to convert raw data into usable training data. This preprocessing will be called “initial processing” or “initial pre-processing” on the rest of the thesis. Output of this initial preprocessing is a dataset that is used by all meta-classifiers. In this section, we explain initial preprocessing step. The initial preprocessing includes, removal of redundant variables, handling of missing values, aggregation of the categorical values and normalization. Because of the large number of variables on KDD 2009 dataset we had to filter some of them. We have removed the variables which have more than 98% of their values missing. We also removed the features which take only one value over all instances. Information is carried over only if there is a change in the pattern of the value of the variable. Because, if there is no change then there is no information to process.

This process left us with 168 variables. Missing values in numeric variables are replaced with mean and missing values in categorical variables are assigned to a new value as “*missing*”. They are treated as a new value and new category. There can be a reason for these values to be missing. Therefore, missingness might be predictive. All values in numerical values are normalized between 0 and 1. Some categorical values have large number of items in their vocabulary. These variables are problematic because some classifiers, like neural networks, can only handle numeric variables. To use these categorical variables with such classifiers that accepts only numeric variables, we have to apply binarization. In binarization each value in a categorical variable is represented by a binary variable. This variable is assigned 1 if the instance belongs to that category and all other variables created from binarization is assigned 0. This approach was not useful for our problem because some categorical variables have more than 1000 distinct values creating additional 1000 new variables which significantly increase model building time complexity.

To circumvent this expansion in number of variables, we keep the 10 most frequent categories and group the rest in a category called “*Others*”. Therefore, there are at most 11 new variables generated. Some algorithms cannot handle

numeric values. We have applied discretization to numeric variables for these algorithms. Preprocessing we applied can be listed as follows:

- Remove the variables that have more than 98% of their values missing.
- Remove the variables that take only one value or category.
- Replace missing values in numerical variables with mean.
- Aggregate the missing values in nominal variables in a category called “*missing*”.
- Keep 10 most frequent categories in nominal variables and aggregate others in a category called “*others*”.
- Apply binarization to the nominal variables for the algorithms that cannot handle nominal values.
- Apply discretization to the numeric variables for the algorithms that cannot handle numeric values.

SGI dataset was already preprocessed and most of the methods were not applicable that were on KDD 2009 dataset.

4.1.2.1 Bagging with Decision Trees

Bagging was first introduced by Breiman in 1996 [20]. Bagging, see Algorithm 1, repeatedly samples instances with replacement from a dataset according to a uniform probability distribution. If an instance is chosen for the sample, it is not removed from the original set. For the next iteration it has same probability to be chosen again. Therefore, some instances maybe chosen more than once in the same dataset, and some of them may not be chosen at all. This is called *sampling with replacement*. In bagging, each sample is equally sized. A decision tree is built for each of these samples. Each of these trees votes for the prediction for the test set instance. The class that is voted for most of the time is the output of

the ensemble classifier. Bagging may be applied to other algorithms as well but bagging with decision trees is the most popular practice.

We have applied bagging algorithm for Decision Trees. Bagging technique significantly improved the performance of our Decision Trees. For these results please refer to Table 5.1. AdaBoost can also be applied to the Decision Trees but Bagging performs better and faster.

Algorithm 1 Bagging Algorithm with Decision Trees

- 1: k is the number of samples with replacement
 - 2: **for** $i = 0$ to k **do**
 - 3: Create a sample of size N , D_i
 - 4: Train a decision tree C_i on the sample D_i
 - 5: **end for**
 - 6: $C^*(x) = \operatorname{argmax} \sum_i \sigma(C_i(x) = y)$
 $\sigma(\cdot) = 1$ if its argument is true and 0 otherwise.
-

4.1.3 The Meta-Classifier

In this study, we built and used our own meta-classifiers for the churn analysis and prediction problem. For each meta-classifier, we match a preprocessing method or a set of methods that best fit the base classifier and increase its performance. In this study, we decided to use ensemble of meta-classifiers because of the fact that there is no single preprocessing technique that is compatible with all classifier algorithms. Therefore, we apply different preprocessing techniques where they fit. Each meta-classifier we propose encompasses both a preprocessing method and a model built from the preprocessed dataset, see Figure 4.3. The dataset is pre-processed with the preprocessing technique chosen then the classification model is developed with the preprocessed data. This method might look like suffering from excessive computations of subsets; however, in practice there are only limited number of preprocessing techniques for each classifier. For example, ANNs cannot handle categorical values; therefore, there is no need to apply discretization methods to it. Pruned decision trees are biased through frequent class. We can apply re-sampling for pruned decision trees. With a little training and testing with a small subset of the data, correct matchings can be determined. On Table

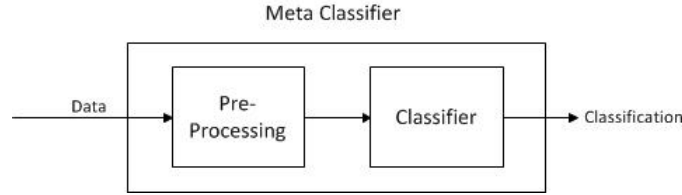


Figure 4.3: High Level Design of Meta-Classifiers

4.3 the effects of preprocessing techniques to decision trees(DT), Logistic Regression(LR) and Random Forest(RF) are given. While selection of attributes with information gain metric, significantly increases performance of decision trees, it decreases the performance of Logistic Regression while having nearly no effect on Random Forest.

Table 4.3: Preprocessing - Classifier pairs : Meta-Classifiers

Name	DT ^a	DT pruned ^b	LR ^c	RF ^d
No preprocessing	0.52	0.50	0.67	0.67
Re-sample	0.47	0.53	0.67	0.68
Discretization	0.62	0.68	0.69	0.69
Info gain selection	0.63	0.64	0.61	0.68

^a Decision Trees

^b Decision Trees with post-pruning algorithm

^c Logistic Regression

^d Random Forest

4.1.4 The Proposed Ensemble Classifier Approach

As explained previously, in order to improve the performance of single, powerful classifiers contained within, we propose an ensemble classifier approach to address the churn prediction problem over large and high-dimensional datasets. We formed our ensemble classifier using the “voting classifier” method described in the following subsection. The participating meta-classifiers in the voting classifier of our final model are determined by a threshold. In our experiments the threshold is 0.69. If classifier performance exceeds or equals the threshold ROC area score, then we include it in the final model. This score is chosen by inspecting the other classifiers’ performances. We have tried to include good performing

classifiers to our final model. Thresholds between 0.65 to 0.70 were empirically turned out to be reasonable choices. We tried all of them and best result was achieved with threshold of 0.69. There were 3 classifiers with ROC area larger than 0.69. For the KDD 2009 dataset we choose the best classifiers by looking at their ROC area performance. However, the metric used to choose classifiers can be different depending on the dataset and characteristics of the problem. Different metrics for choosing algorithms can be F-measure, recall, precision, accuracy, etc. which were used during our tests shown in Chapter 5. Algorithm 2 is the procedure of the whole process of our proposed approach. Alternative evaluation functions can be implemented easily. For example, as shown in Algorithm 3, top n number of algorithms can be used for selection. It would not be easy to decide on the number n as well as the threshold. However, it can be experimented using different values. Large values of n or small values of threshold will cause more algorithms to be selected and can result in over-fitting. Each problem domain will require different values of n or threshold.

Algorithm 2 Algorithm of Our Approach

Require: *threshold*

Require: Meta-classifiers $MC = mc_1, mc_2, \dots, mc_n$

```

1: vc is voting classifier
2: for Each  $mc_i \in MC$  do
3:   Train Meta-classifier  $mc_i$ 
4:    $score \leftarrow$  Evaluate  $mc_i$ 
5:   if  $score > threshold$  then
6:     add  $mc_i$  to vc
7:   else
8:     ignore
9:   end if
10: end for
11: train vc

```

4.1.4.1 Voting Classifier

Voting classifier, as explained in Algorithm 4, is also a meta-classifier. Voting classifiers include many trained classifiers in them. Each classifier in the ensemble, classifies instances and votes for a class(churner or non-churner). These votes

Algorithm 3 Alternative Algorithm of Our Approach**Require:** $n \leftarrow$ number of algorithms that will be used in the voting classifier**Require:** Meta-classifiers $MC = mc_1, mc_2, \dots, mc_m$

- 1: vc is voting classifier
- 2: $E = e_1, e_2, \dots, e_n$ are evaluations of meta-classifiers
- 3: **for** Each $mc_i \in MC$ **do**
- 4: Train Meta-classifier mc_i
- 5: $e_i \leftarrow$ Evaluate mc_i
- 6: **end for**
- 7: Add top n , mc to vc
- 8: train vc

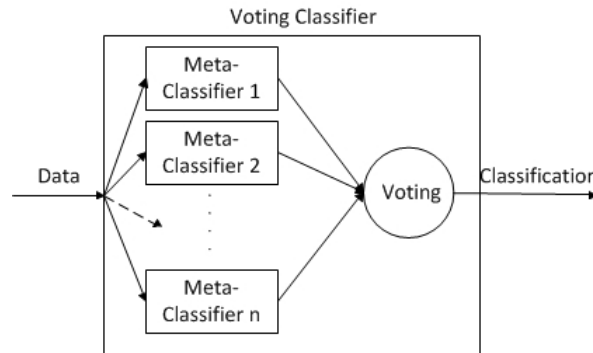


Figure 4.4: High Level Design of a Voting-Classifier

might be probabilities or pure votes. There are some experimented and practised ways of combining different classifiers and meta-classifier. Some of them use majority voting and some use product of probabilities or average probability. With majority voting, each individual classifier votes for the prediction. As the name suggests, the class which gets the maximum vote simply becomes the output of the ensemble classifier. With product of probabilities, each classifier's probability contributes to the class selection and class with a higher probability has stronger influence for the output of the ensemble classifier. Another method to combine classifiers takes the average of the probabilities from classifiers and decides the output. Algorithm 4 presents pseudo-code of a sample voting classifier. The high-level design of a voting classifier is also illustrated in Figure 4.4. In this study we have used average of probabilities method to determine final prediction.

Algorithm 4 Sample Voting Classifier Algorithm

- 1: Train n classifiers C_1, C_2, \dots, C_n
 - 2: D is the un-labeled test instance to be classified
 - 3: Prediction of the ensemble classifier $C^*(D)$ is:
 - 4: $C^*(D) = \operatorname{argmax} \sum_i^n \sigma(C_i(D) = y)$
 $\sigma(\cdot) = 1$ if its argument is true and 0 otherwise.
-

Chapter 5

Experimental Evaluation and Results

The method we have tried to solve the churn prediction problem performed well on the KDD 2009 and SGI datasets. After these results, we wanted to further experiment our algorithm on well studied datasets. We did extensive experiments to discover our approach and, its weaknesses and strength. Datasets we have used in the experiments are given in Table 5.5 along with some of their statistics. All the datasets in Table 5.5 are classification tasks. These datasets are very neat, therefore we did not apply any preprocessing on most of them. For benchmark tests, we have chosen 10 well known machine learning algorithms. They are given on the Table 5.6. These algorithms also form our library of classifiers for our proposed approach. We tried to inspect the performance of the algorithms in Table 5.6 executed on the datasets in Table 5.5 along with our proposed approach. What we want to show is if the algorithms can perform better when used with our approach compared to running individually. Results were very promising. In most of the cases our approach performed equal or better than the best performing algorithm in the Table 5.6.

5.1 Results on Churn Prediction Datasets

Our proposed model is built out of meta-classifiers that have performance exceeding or equal to the ROC area score of 0.69. There are three meta-classifiers that qualify for this requirement, see in Table 5.1. These classifiers are chosen from library of more than 250 classifiers. Logistic Regression based meta-classifier was built with discretization. Decision trees use bagging. For Random Forest based meta-classifiers 300 trees and a depth limit of 20 is used. Preprocessing applied to the Random Forest based meta-classifiers were supervised discretization, re-sampling and the feature selection of 60 features with information gain metric. As seen from the Table 5.2, the base classifier performance significantly increases with proposed ensemble of models with voting procedure.

Table 5.1: Preprocessing - Classifier Pairs and ROC Scores

	DT pruned	DT	Log. Reg.	RF 100 trees	RF 200 trees	RF 300 trees	Bagging
No Preprocessing	0.5000	0.5205	0.6767	0.6593	0.6762	0.6823	0.6933(1)
Resample	0.4527	0.4699	0.6774	0.6744	0.6855	0.6885	0.4801
Sup. Disc.	0.6816	0.6524	0.6674	0.6812	0.6938	0.6984(2)	0.6730
Discretize	0.6114	0.6212	0.6929(3)	0.6647	0.6765	0.6804	0.6147
InfoGain 60 attributes	0.6366	0.6311	0.6670	0.6732	0.6836	0.6863	0.6876

Table 5.2: ROC Area of the Final Model

	Classifier	Score
1	Bagging	0.6933
2	RF with 300 trees	0.6984
3	Log. Reg.	0.6929
	Final Model with (1),(2),(3)	0.7230

The results of our proposed approach are still below the top three scorers of the KDD 2009 competition. On the other hand, it performs better than all the algorithms experimented by Verbeke et al [28]. Our best performing algorithms are similar to the top 3 of the KDD 2009. Bagging, Random Forest and Logistic Regression were reported to perform well which can be seen in Table 5.3. However, Verbeke’s experiments do not show the same.

Table 5.3: Comparison of Results with Other Methods for KDD 2009 Dataset

Technique	ROC Score	Variables	Reference
Ensemble selection	0.7651	100 to 5000	[10]
TreeNet	0.7614	7595	[11]
Gradient Boosting Machine	0.7493	198	[4]
Our Meta-classifiers with voting	0.7230	168	
Bayes Network	0.714	20	[28]
Neural Networks	0.712	20	[28]
k-Nearest Neighbour	0.700	20	[28]
Naïve Bayes	0.688	20	[28]
CART	0.658	20	[28]

Our observations indicate that model building is significantly faster than IBM’s reported methods [10]. The runtime for the proposed final classifier was approximately 2 hours using standard desktop computing platform with dual core 2.66 GHz processor and a 4 Gb of RAM. The final results were very encouraging in terms of efficiency with higher scores and faster runtime. Our method also has some limitations. Human expertise is needed to choose the appropriate preprocessing methods to use with each classifier. An expert in the field can easily limit the set of preprocessing methods for each classifier. An expert also needs to determine the threshold to choose classifiers for final model. For each new problem, this threshold may vary. Therefore, a prior work has to be done to determine the threshold before automating the process. As a future work, we will try to

automate both preprocessing method selection and final threshold selection steps.

Table 5.3 show the comparison of our results with KDD 2009 Competition leaders and other known algorithms tested on this dataset by Verbeke et al. For results of other algorithms, refer to Verbeke’s paper [28]. Our method is close to leaders of the competition. It cannot get a better score. On the other hand, it performs better than the all of the state-of-the-art algorithms tested on this dataset. In terms of computational resources it is better than IBM’s methodology. Our experiments are promising that with less resources, very high scores can be achieved.

For the SGI dataset, F-measure of the models were very close to each other. Therefore, it was not very good indicator to compare different models in terms of performance. ROC area scores show the distinction between the models. Preprocessing methods did not help the algorithms very much. Most of the time models without preprocessing are better than the ones with the preprocessing techniques. We can conclude that this is a normal result. Because, dataset is artificial and it is already in an organized form. There aren’t any missing values and attributes are normalized. But we have seen that preprocessing techniques significantly increased the results on KDD dataset, since it is a real data collection from customer. Through KDD 2009 website, only ROC area results are calculated and published. We didn’t have a chance to see and publish the accuracy and F-measure.

We have tried variety of Random Forest with different depths and trees. Results were almost the same with the different number of depth limits. We have used depth limits of 0, 5, 10, 15, 20 and results were the same. This is because the limited number of attributes. Random Forest converges with a depth limit of 5, therefore increasing the depth limit does not help. Situation is the same with the number of trees, too. To summarize, random forest with 300 trees and 5 depth had ROC area of 0.864. Bagging with decision trees had ROC area of 0.838. They had the highest ROC score in our models except the Random Forest. We have included those two in our final ensemble voting classifier. This final model had 0.876 ROC area score on the test set. We observed that results are

significantly increase when our approach is used. Accuracy has also increased with 0.06 points. However, this score is not significant. F-measure scores stayed the same with the best performing classifier. We were concentrated on increasing ROC area, which is a distinctive metric for skewed class problems.

Table 5.4: Best Meta-Classifiers for SGI

Technique	Accuracy	F-measure	ROC Area
Random Forest	88.18	0.877	0.864
Bagging with DT	88.66	0.889	0.838
Final Model	88.72	0.889	0.876

Our approach improved the base classifier results significantly. SGI dataset that is artificial were already had some pre-processing techniques applied. Therefore pre-processing techniques did not help our method. As a result we have built a normal voting classifier. Methods in Table 5.4 are obtained without preprocessing techniques but initial preprocessing is applied. Test has been run 10 times and mean scores have been reported.

Our experiment with SGI dataset shows that the ROC area scoring is the most distinctive one among others. Therefore, a good method to evaluate classifiers for churn prediction problem. In addition, our method performed well on another churn dataset which is very promising. Further experiments with our proposed approach are given and explained on the next chapter.

5.2 Test Framework

We used Weka libraries for our tests. Test framework is written in JAVA. All tests reported in this section are done with 10 times 10-fold cross validation. 10-fold cross validation is explained in Chapter 2. 10 times 10-fold cross validation is executing 10-fold cross validation 10 times. The data is re-shuffled and re-stratified before each round. For each run performances are recorded. Average performances and standard deviations for these 10 runs is computed. Weka has its own evaluation class for 10-fold cross validation. To apply methods from this class to our own classifier, we had to integrate classifier to Weka environment.

10-fold evaluation method in Weka expects a classifier that is implemented from *Weka.Classifier* interface. The *Weka.Classifier* interface want classifiers to implement only one method called *buildClassifier()*, see Figure 5.1. As the name suggests, Weka wants new classifiers to declare how it is built. The *buildClassifier()* method expects only one parameter called *Instances*. *Instances* is a Weka class that represents instances. Classifier gets build with these instances. Instances are training examples. Weka has another class called *Evaluation*. It has a method, *evaluateModel*. Evaluation class has attributes like ROC area, F-measure, precision, recall and accuracy. When *evaluateModel* method is called, the attributes of the *Evaluation* class is set. *Evaluation* class's *evaluateModel* method can be called by providing the classifier, dataset and number of folds. Weka environment takes care of the rest.

Tests are done on 17 UCI datasets with 10 algorithms. Each algorithm trained and tested on each dataset for 10 folds. This process is done 10 times. Total of 170 results, 1700 runs and 17000 folds were gathered. We also experimented our algorithm by varying the number of algorithms used in our final model. This test is also done on each dataset for 10 folds and 10 times. Each of these test were done in a multi-threaded environment. For details please see Section 5.2.1. Class diagram for Weka's *Classifier*, *Evaluation* and our method when integrated to Weka can be seen on Figure 5.1.

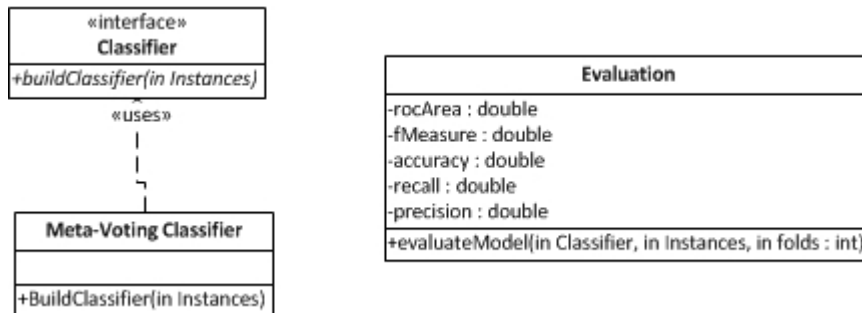


Figure 5.1: Class Diagram related with Classifiers

Table 5.5: UCI Datasets Used in Tests

Dataset	Class	Instance	Disc. features	Cont. Features
anneal	6	898	32	6
autos	7	205	10	16
breast-cancer	2	286	10	0
glass	7	214	0	9
hearth-statlog	2	270	0	13
hepatitis	2	155	13	6
iris	3	150	0	4
labor	2	57	8	8
lymph	4	148	15	3
primary-tumor	22	239	17	0
segment	7	2310	0	19
soybean	19	683	35	0
splice	3	3190	60	0
vehicle	4	846	0	18
vote	2	435	16	0
wine	3	178	0	13
zoo	7	101	16	1

Table 5.6: Algorithms Used in Tests

	Algorithm	acronym
1	Decision Trees with 0.25 confidence factor	DT
2	Decision Trees with 0.50 confidence factor	DT
3	Naïve Bayes	NB
4	Logistic Regression	LR
5	Bagging with Decision Trees	Bagging
6	AdaBoost	
7	Rotation Forest	
8	Random Forest with 10 trees	RF
9	Random Forest with 20 trees	RF
10	Nearest Neighbour	IBK

5.2.1 Multi-processing

Weka GUI does not support multi-threading by default. However, by using the Weka through Java code itself, each classifier and each test can be trained in a different thread. This makes use of the different cores of the processor. Our benchmark tests were done on a 8-core processor. By multi-threading, tests were done approximately 8 times faster. We have used 8 threads for experiments which made optimum use of 8 cores. Multi-threading was implemented with Java's *Executor* class. *Executor* classes thread pool was invoked. Thread pool can be created by providing number of threads to be used in Java. As mentioned in Section 5.2, each test case was run 10 times. All of these tasks are runnable tasks. They were executed on thread pool. Thread pool queues each of these tasks. Thread pool takes a task from the task queue. If there is a waiting thread, task is invoked on this thread. There are at most 8 tasks running at the same time, because task pool was created with 8 threads. Multi-threading is essential in benchmark tasks. It significantly reduces the time that an experiment take.

Our approach is also done by multi-threading. Our method includes training of meta-classifiers in the voting classifier. Each classifier is trained on a different thread, therefore on a different core of the CPU. Multi-threading increases the performance of the classifier if the processor is multi-cored.

5.2.2 Experimental Results from UCI Benchmark Datasets

Our tests have been executed in various different settings. For KDD 2009 dataset, we have chosen best performing meta-classifiers to our final ensemble model according to their ROC area scores. From now on, we will call our algorithm selection evaluation function as metric. Such metrics are ROC area, F-measure, recall, precision, etc. ROC area is a good metric for skewed datasets. However, for some datasets F-measure and accuracy might be more predictive. Even if they are not, they might be better for the purpose of the classification problem. In the experiments we have tried both ROC area, F-measure and accuracy for the algorithm selection for the ensemble model. ROC area and F-measure scores are

weighted average values of the metrics. Accuracy cannot be weighted. It is the percentage of the correct classifications over all classifications. Weighted average is calculated as follows;

- Suppose we have a dataset D with the label y which can take n number of values(classes)
- Each class is defined as;
 $y = c_1, c_2, \dots, c_n.$
- Number of instances labeled as each class are;
 $m_1, m_2, \dots, m_n.$
- Evaluation function for class i is defined as;
 $E(c_i, D)$
- Total number of instances is;
 $N = \sum_{i=1}^n (m_i)$
- Weighted average evaluation of the classifier is;

$$\sum_{i=1}^n \left(\frac{E(c_i, D) \times m_i}{N} \right) \quad (5.1)$$

With weighted average score, each classes score contributes to the final score according to their proportion of the number of instances belong to that class. Evaluation function can easily be made ROC area evaluation or F-measure evaluation. Tables below are the evaluation results. Note that the tables were too big, therefore they were split into two tables. Table 5.7 and Table 5.8 forms one table, Table 5.9 and 5.10 forms another table, Table 5.11 and Table 5.12 forms the last table.

Table 5.7 and Table 5.8 are the ROC area results of the 10 experimented algorithms versus our approach. Results show the ROC area scores of the algorithms in each dataset. The *Max* column in Table 5.8 shows the score of the best performing algorithms out of 10 algorithms we have experimented. The last column

shows the comparison of our method with other algorithms. If the value on the last column is *better*, our method is significantly better than all algorithms used in regards to significance factor. If it is *same*, our method performs same as the algorithm that performs best among others, therefore better than all others. If it is *worse*, our method is not as good as the best performing algorithm, but this doesn't mean that our method is the worst. It is just worse than the best one when chosen ROC area metric. Table 5.9, Table 5.10 and Table 5.11, and Table B.6 structured same as the ROC Tables. Table 5.9 and Table 5.10 are F-measure results of the algorithms and our method when F-measure chosen as algorithm selection metric. Table 5.11 and Table 5.12 is accuracy results of the algorithms and our method when accuracy chosen as algorithm selection metric.

First, we compare the performance of algorithms used in these experiments. When we apply the ROC area evaluation for the algorithms declared in Table 5.6, scores were very close to each other on the same datasets. Rotation Forest [27] performs best on 8 datasets. Surprisingly, Naïve Bayes classifier is the best over 6 datasets and it is significantly better than others. We would not expect Naïve Bayes to out perform others, because other algorithms include some state-of-the-art techniques. There is no similarity between the attributes of the datasets that an algorithm performs well on, see Table 5.5 and Table 5.7, Table 5.8. Unlike for KDD 2009 dataset, Random Forest, Logistic Regression, AdaBoost and Bagging did not perform very well over the selected UCI datasets. Bagging did not top on any of the datasets, AdaBoost top only one dataset but sharing the score with Rotation Forest, and Random Forest tops 4 datasets sharing the score of the 2 with Rotation Forest. Rotation Forest over-performed the Bagging, AdaBoost and Random Forest on all datasets. Overall, Rotation Forest and Naïve Bayes had the best ROC area scores.

For ROC area results, our algorithm is slightly better than the all other methods on most datasets. We applied significance factor of 0.005. In most of the dataset, results are not improved more than the significance factor therefore our method did not significantly improved the results. On the dataset “primary tumor” our method is better than all of the other methods except Naïve Bayes.

It scores very close to Naïve Bayes but slightly worse. For the “hepatitis, lymphography, vehicle” datasets, our method performs significantly better than all algorithms. ROC area metric is not distinctive on these datasets in respect to other metrics. These datasets do not suffer from class imbalance. For relatively simple datasets that do not suffer from class imbalance like the ones in UCI, ROC area is not a very good evaluation. Algorithms reported in Table 5.6 already tops the ROC area scores for the datasets used. Using more algorithms do not help the results. ROC area is not a good evaluator for UCI datasets since all scores are very close to each other. Algorithms have score of more than 0.99 on 8 datasets which is very hard to improve. However, our method is still at least as good as the other methods we have used with lower standard deviation. We can conclude that our method do not provide a much improvement if the evaluation is done by the ROC area, and ROC area is not very predictive for the chosen datasets.

Table 5.7: ROC Results on UCI datasets

Datasets	Algorithms						
	DT	DT pruned	NB	LR	Bagging	AdaBoost	
anneal	0.988 \pm 0.019	0.991 \pm 0.014	0.959 \pm 0.019	0.994 \pm 0.010	0.996 \pm 0.011	0.997 \pm 0.009	
autos	0.906 \pm 0.052	0.906 \pm 0.053	0.835 \pm 0.063	0.900 \pm 0.045	0.951 \pm 0.035	0.957 \pm 0.038	
breast-cancer	0.587 \pm 0.116	0.595 \pm 0.119	0.703 \pm 0.108	0.630 \pm 0.112	0.662 \pm 0.111	0.639 \pm 0.114	
Glass	0.805 \pm 0.072	0.806 \pm 0.071	0.775 \pm 0.056	0.824 \pm 0.053	0.898 \pm 0.052	0.911 \pm 0.048	
heart-statlog	0.786 \pm 0.094	0.782 \pm 0.091	0.902 \pm 0.054	0.905 \pm 0.054	0.881 \pm 0.066	0.860 \pm 0.065	
hepatitis	0.668 \pm 0.184	0.717 \pm 0.188	0.857 \pm 0.128	0.840 \pm 0.152	0.813 \pm 0.145	0.827 \pm 0.122	
iris	0.964 \pm 0.041	0.966 \pm 0.039	0.995 \pm 0.012	0.982 \pm 0.028	0.984 \pm 0.026	0.971 \pm 0.040	
labor-neg-data	0.723 \pm 0.223	0.718 \pm 0.245	0.969 \pm 0.076	0.989 \pm 0.037	0.873 \pm 0.211	0.946 \pm 0.113	
lymphography	0.790 \pm 0.119	0.792 \pm 0.120	0.918 \pm 0.071	0.807 \pm 0.098	0.908 \pm 0.077	0.915 \pm 0.077	
primary-tumor	0.721 \pm 0.049	0.717 \pm 0.047	0.847 \pm 0.034	0.790 \pm 0.038	0.770 \pm 0.045	0.736 \pm 0.050	
segment	0.987 \pm 0.006	0.986 \pm 0.006	0.971 \pm 0.006	0.989 \pm 0.004	0.997 \pm 0.003	0.999 \pm 0.001	
soybean	0.977 \pm 0.013	0.972 \pm 0.015	0.995 \pm 0.003	0.990 \pm 0.005	0.994 \pm 0.005	0.993 \pm 0.007	
splice	0.956 \pm 0.013	0.949 \pm 0.013	0.994 \pm 0.003	0.941 \pm 0.010	0.988 \pm 0.005	0.990 \pm 0.005	
vehicle	0.855 \pm 0.030	0.853 \pm 0.030	0.771 \pm 0.029	0.906 \pm 0.018	0.920 \pm 0.018	0.925 \pm 0.020	
vote	0.979 \pm 0.025	0.979 \pm 0.025	0.974 \pm 0.019	0.987 \pm 0.021	0.984 \pm 0.020	0.988 \pm 0.016	
wine	0.948 \pm 0.046	0.948 \pm 0.046	0.999 \pm 0.003	0.982 \pm 0.025	0.994 \pm 0.014	0.994 \pm 0.019	
zoo	0.976 \pm 0.031	0.976 \pm 0.031	1.000 \pm 0.003	0.983 \pm 0.025	0.987 \pm 0.023	0.988 \pm 0.025	

Table 5.8: ROC Results on UCI datasets *continued*

Datasets	Algorithms							
	Rotation Forest	RF	RF	RF	IBK	MAX	Ours	
anneal	0.996 \mp 0.008	0.999 \mp 0.006	0.999 \mp 0.005	0.985 \mp 0.019	0.999	0.999	0.999	same
autos	0.961 \mp 0.031	0.952 \mp 0.035	0.958 \mp 0.029	0.840 \mp 0.060	0.961	0.962	0.962	same
breast-cancer	0.675 \mp 0.109	0.647 \mp 0.108	0.654 \mp 0.104	0.604 \mp 0.082	0.703	0.698	0.698	same
Glass	0.919 \mp 0.044	0.919 \mp 0.050	0.931 \mp 0.044	0.795 \mp 0.057	0.931	0.935	0.935	same
heart-statlog	0.896 \mp 0.059	0.876 \mp 0.062	0.887 \mp 0.057	0.760 \mp 0.085	0.905	0.906	0.906	same
hepatitis	0.843 \mp 0.133	0.825 \mp 0.133	0.847 \mp 0.122	0.678 \mp 0.139	0.857	0.867	0.867	better
iris	0.996 \mp 0.010	0.982 \mp 0.029	0.984 \mp 0.027	0.966 \mp 0.036	0.996	0.996	0.996	same
labor-neg-data	0.981 \mp 0.068	0.943 \mp 0.108	0.954 \mp 0.101	0.844 \mp 0.162	0.989	0.990	0.990	same
lymphography	0.923 \mp 0.069	0.907 \mp 0.068	0.923 \mp 0.066	0.817 \mp 0.085	0.923	0.939	0.939	better
primary-tumor	0.794 \mp 0.037	0.774 \mp 0.038	0.788 \mp 0.038	0.641 \mp 0.039	0.847	0.838	0.838	worse
segment	0.999 \mp 0.001	0.998 \mp 0.002	0.999 \mp 0.001	0.983 \mp 0.007	0.999	0.999	0.999	same
soybean	0.996 \mp 0.004	0.993 \mp 0.006	0.995 \mp 0.005	0.945 \mp 0.019	0.996	0.997	0.997	same
splice	0.992 \mp 0.004	0.972 \mp 0.010	0.987 \mp 0.004	0.833 \mp 0.015	0.994	0.995	0.995	same
vehicle	0.939 \mp 0.012	0.916 \mp 0.018	0.925 \mp 0.015	0.797 \mp 0.025	0.939	0.945	0.945	better
vote	0.991 \mp 0.012	0.988 \mp 0.016	0.989 \mp 0.015	0.923 \mp 0.041	0.991	0.993	0.993	same
wine	0.999 \mp 0.003	0.998 \mp 0.005	0.999 \mp 0.003	0.965 \mp 0.031	0.999	0.999	0.999	same
zoo	0.993 \mp 0.020	0.993 \mp 0.016	0.998 \mp 0.005	0.980 \mp 0.031	1.000	0.999	0.999	same

Significance Factor 0.005

F-measure results are more distinctive than the ones in the ROC area. We used significance factor of 0.005 for F-measure like in ROC area evaluation. F-measure considers both precision and recall. Normally, when precision is increased, recall decreased. There is a trade-off between them. F-measure is a good evaluator when both precision and recall are important. Naïve Bayes, Logistic Regression, Rotation Forest and AdaBoost are among the best scoring algorithms. Naïve Bayes and Rotation Forest top 3 of the datasets. Logistic Regression and AdaBoost top 4 datasets each. Our method is better on many of the datasets with F-measure metric, see Table 5.9 and Table 5.10. Our method is better even when the comparison is “same” on the table, however it is not significantly better. On “iris” dataset, our method performs slightly worse than the best of other algorithms. It performs better on “glass,hearth-statlog, hepatitis, labor, lymphography, primary tumor, wine and zoo”. The datasets that our approach performs well on have a common thing. They all have 2 or 3 values for the class variable, except zoo and primary tumor, as seen in Table 5.5. Zoo dataset has 7 classes and Primary - tumor dataset has 22 classes. If the evaluation is done with F-measure we can easily say that our method performs better or same on almost all datasets.

Table 5.9: F-measure Results on UCI datasets

Datasets	Algorithms						
	DT	DT pruned	NB	LR	Bagging	AdaBoost	
anneal	0.989 \pm 0.011	0.989 \pm 0.011	0.886 \pm 0.027	0.995 \pm 0.007	0.990 \pm 0.010	0.997 \pm 0.006	
autos	0.780 \pm 0.096	0.780 \pm 0.097	0.561 \pm 0.113	0.758 \pm 0.087	0.796 \pm 0.086	0.841 \pm 0.082	
breast-cancer	0.675 \pm 0.077	0.657 \pm 0.076	0.714 \pm 0.080	0.656 \pm 0.072	0.675 \pm 0.078	0.664 \pm 0.078	
Glass	0.662 \pm 0.096	0.667 \pm 0.099	0.453 \pm 0.102	0.614 \pm 0.099	0.718 \pm 0.093	0.736 \pm 0.079	
heart-statlog	0.779 \pm 0.076	0.772 \pm 0.074	0.834 \pm 0.061	0.835 \pm 0.065	0.804 \pm 0.068	0.784 \pm 0.072	
hepatitis	0.777 \pm 0.097	0.781 \pm 0.100	0.841 \pm 0.092	0.830 \pm 0.082	0.785 \pm 0.084	0.812 \pm 0.082	
iris	0.947 \pm 0.054	0.947 \pm 0.054	0.955 \pm 0.051	0.972 \pm 0.044	0.946 \pm 0.051	0.943 \pm 0.053	
labor-neg-data	0.761 \pm 0.181	0.782 \pm 0.181	0.934 \pm 0.105	0.939 \pm 0.103	0.800 \pm 0.169	0.832 \pm 0.171	
lymphography	0.761 \pm 0.114	0.770 \pm 0.116	0.825 \pm 0.100	0.790 \pm 0.109	0.787 \pm 0.103	0.830 \pm 0.110	
primary-tumor	0.365 \pm 0.065	0.374 \pm 0.066	0.453 \pm 0.062	0.379 \pm 0.067	0.395 \pm 0.070	0.367 \pm 0.064	
segment	0.969 \pm 0.012	0.969 \pm 0.012	0.778 \pm 0.021	0.960 \pm 0.012	0.975 \pm 0.010	0.982 \pm 0.009	
soybean	0.920 \pm 0.029	0.916 \pm 0.031	0.927 \pm 0.030	0.925 \pm 0.028	0.925 \pm 0.030	0.928 \pm 0.030	
splice	0.945 \pm 0.013	0.937 \pm 0.013	0.954 \pm 0.012	0.904 \pm 0.015	0.952 \pm 0.012	0.953 \pm 0.012	
vehicle	0.717 \pm 0.043	0.718 \pm 0.042	0.407 \pm 0.052	0.793 \pm 0.039	0.739 \pm 0.042	0.752 \pm 0.041	
vote	0.966 \pm 0.026	0.966 \pm 0.026	0.901 \pm 0.039	0.956 \pm 0.031	0.963 \pm 0.026	0.955 \pm 0.030	
wine	0.931 \pm 0.061	0.931 \pm 0.061	0.975 \pm 0.038	0.969 \pm 0.045	0.951 \pm 0.056	0.964 \pm 0.043	
zoo	0.898 \pm 0.086	0.898 \pm 0.086	0.947 \pm 0.062	0.934 \pm 0.076	0.905 \pm 0.087	0.917 \pm 0.086	

Table 5.10: F-measure Results on UCI datasets *continued*

Datasets	Algorithms						
	Rotation Forest	RF	RF	RF	IBK	MAX	Ours
anneal	0.987 \pm 0.012	0.993 \pm 0.009	0.996 \pm 0.007	0.990 \pm 0.012	0.997	0.998	same
autos	0.815 \pm 0.088	0.811 \pm 0.085	0.822 \pm 0.086	0.741 \pm 0.097	0.841	0.845	same
breast-cancer	0.684 \pm 0.076	0.676 \pm 0.073	0.674 \pm 0.078	0.676 \pm 0.074	0.714	0.718	same
Glass	0.727 \pm 0.096	0.747 \pm 0.090	0.769 \pm 0.091	0.692 \pm 0.085	0.769	0.776	better
heart-statlog	0.825 \pm 0.072	0.801 \pm 0.070	0.810 \pm 0.065	0.760 \pm 0.085	0.835	0.849	better
hepatitis	0.831 \pm 0.088	0.813 \pm 0.085	0.820 \pm 0.076	0.799 \pm 0.093	0.841	0.850	better
iris	0.953 \pm 0.051	0.942 \pm 0.052	0.942 \pm 0.051	0.953 \pm 0.049	0.972	0.961	worse
labor-neg-data	0.911 \pm 0.124	0.853 \pm 0.158	0.858 \pm 0.160	0.839 \pm 0.166	0.939	0.949	better
lymphography	0.815 \pm 0.101	0.794 \pm 0.101	0.808 \pm 0.099	0.808 \pm 0.090	0.830	0.865	better
primary-tumor	0.398 \pm 0.063	0.392 \pm 0.064	0.398 \pm 0.063	0.340 \pm 0.066	0.453	0.462	better
segment	0.980 \pm 0.010	0.977 \pm 0.009	0.980 \pm 0.010	0.971 \pm 0.012	0.982	0.985	same
soybean	0.946 \pm 0.027	0.915 \pm 0.032	0.923 \pm 0.031	0.897 \pm 0.035	0.946	0.945	same
splice	0.954 \pm 0.012	0.879 \pm 0.030	0.911 \pm 0.024	0.760 \pm 0.021	0.954	0.965	better
vehicle	0.778 \pm 0.035	0.735 \pm 0.049	0.746 \pm 0.042	0.690 \pm 0.040	0.793	0.792	same
vote	0.963 \pm 0.027	0.959 \pm 0.027	0.962 \pm 0.026	0.923 \pm 0.039	0.966	0.968	same
wine	0.977 \pm 0.035	0.971 \pm 0.039	0.976 \pm 0.032	0.951 \pm 0.044	0.977	0.982	better
zoo	0.913 \pm 0.090	0.880 \pm 0.115	0.911 \pm 0.095	0.957 \pm 0.066	0.957	0.980	better
Significance Factor 0.005							

Accuracy results in Table 5.11 and Table 5.12 are computed from percentage of the number of correct classified instances over total number of instances. Significance factor used is 0.005. Like the F-measure evaluations, adaBoost and Naïve Bayes are the best performing classifiers. Each one tops 4 datasets in the list. The datasets they top are the same with the ones with F-measure evaluations. It is expected that results will be correlated with other measures. Our method still performs significantly better. It is worse on “breast cancer”, “iris” and “soybean” datasets, same on “anneal” and “glass” datasets, and better on all other datasets. As mentioned above it is very hard to do better on “anneal” dataset, because like in other experiments results are very high. Almost all algorithms scores above 99%.

Table 5.11: Accuracy Results on UCI datasets

Datasets	Algorithms						
	DT	DT pruned	NB	LR	Bagging	AdaBoost	
anneal	98.998 ± 0.915	99.009 ± 0.893	86.590 ± 3.310	99.577 ± 0.608	99.132 ± 0.874	99.778 ± 0.500	
autos	78.898 ± 9.055	78.805 ± 9.291	57.414 ± 10.774	76.588 ± 8.459	80.657 ± 7.963	84.726 ± 7.738	
breast-cancer	70.502 ± 7.116	67.067 ± 7.606	72.697 ± 7.737	67.771 ± 6.918	70.498 ± 7.024	67.223 ± 8.153	
Glass	67.626 ± 9.312	68.043 ± 9.568	49.446 ± 9.498	63.368 ± 9.628	73.504 ± 9.080	75.152 ± 7.587	
heart-statlog	78.148 ± 7.417	77.407 ± 7.323	83.593 ± 5.985	83.667 ± 6.428	80.593 ± 6.723	78.593 ± 7.152	
hepatitis	79.221 ± 9.567	79.037 ± 9.996	83.808 ± 9.702	83.888 ± 8.121	80.725 ± 8.179	82.379 ± 8.014	
iris	94.733 ± 5.301	94.733 ± 5.301	95.533 ± 5.019	97.267 ± 4.352	94.667 ± 5.014	94.333 ± 5.222	
labor-neg-data	78.433 ± 16.028	80.233 ± 16.061	93.567 ± 10.274	94.067 ± 10.012	82.200 ± 14.481	84.767 ± 15.288	
lymphography	76.586 ± 10.845	77.462 ± 11.058	83.129 ± 8.890	79.571 ± 10.109	79.410 ± 9.873	83.243 ± 10.589	
primary-tumor	41.185 ± 6.568	41.715 ± 6.723	49.707 ± 6.462	39.476 ± 7.034	43.686 ± 7.011	41.480 ± 6.454	
segment	96.892 ± 1.160	96.887 ± 1.183	80.048 ± 1.809	96.000 ± 1.211	97.481 ± 1.038	98.208 ± 0.927	
soybean	92.299 ± 2.838	91.846 ± 2.998	92.942 ± 2.917	92.620 ± 2.730	92.739 ± 2.875	93.032 ± 2.885	
splice	94.476 ± 1.259	93.693 ± 1.282	95.408 ± 1.184	90.357 ± 1.487	95.216 ± 1.255	95.323 ± 1.225	
vehicle	72.282 ± 4.320	72.283 ± 4.230	44.683 ± 4.588	79.324 ± 3.897	74.479 ± 4.010	75.589 ± 3.991	
vote	96.571 ± 2.561	96.571 ± 2.561	90.024 ± 3.912	95.648 ± 3.115	96.272 ± 2.632	95.512 ± 3.047	
wine	93.199 ± 5.901	93.199 ± 5.901	97.520 ± 3.772	96.902 ± 4.421	95.219 ± 5.444	96.451 ± 4.215	
zoo	91.627 ± 7.183	91.627 ± 7.183	94.973 ± 5.860	93.445 ± 7.774	92.318 ± 7.085	93.309 ± 7.090	

Table 5.12: Accuracy Results on UCI datasets *continued*

Datasets	Algorithms						
	Rotation Forest	RF	RF	IBK	MAX	Ours	
anneal	98.854 \pm 1.017	99.38 \pm 0.747	99.62 \pm 0.596	99.13 \pm 1.055	99.778	99.766	same
autos	82.333 \pm 8.291	81.75 \pm 8.261	83.01 \pm 8.133	74.55 \pm 9.399	84.726	84.537	same
breast-cancer	72.186 \pm 6.616	69.5 \pm 7.135	69.6 \pm 7.337	68.59 \pm 7.524	72.697	72.762	same
Glass	74.617 \pm 9.182	76.17 \pm 8.684	78.12 \pm 8.579	69.95 \pm 8.434	78.119	78.084	same
heart-statlog	82.704 \pm 6.985	80.41 \pm 6.829	81.26 \pm 6.313	76.15 \pm 8.461	83.667	84.926	better
hepatitis	84.283 \pm 8.366	83.13 \pm 7.989	83.82 \pm 6.891	81.4 \pm 8.554	84.283	85.226	better
iris	95.333 \pm 5.059	94.27 \pm 5.105	94.27 \pm 5.016	95.4 \pm 4.803	97.267	96.067	worse
labor-neg-data	91.7 \pm 11.13	86.9 \pm 13.47	87.2 \pm 13.98	84.3 \pm 16.24	94.067	94.912	better
lymphography	82.571 \pm 9.154	80.74 \pm 8.879	82.28 \pm 8.805	81.54 \pm 8.483	83.243	86.622	better
primary-tumor	45.026 \pm 6.412	42.13 \pm 6.594	42.86 \pm 6.34	34.64 \pm 7.072	49.707	50.206	same
segment	97.957 \pm 0.985	97.75 \pm 0.914	97.98 \pm 0.969	97.13 \pm 1.21	98.208	98.476	same
soybean	94.787 \pm 2.511	91.8 \pm 3.015	92.56 \pm 2.953	90.17 \pm 3.293	94.787	94.583	same
splice	95.361 \pm 1.214	88.13 \pm 2.807	91.29 \pm 2.262	75.97 \pm 2.068	95.408	96.483	better
vehicle	78.343 \pm 3.407	74.08 \pm 4.681	75.19 \pm 3.989	69.59 \pm 3.77	79.324	79.421	same
vote	96.341 \pm 2.721	95.95 \pm 2.649	96.23 \pm 2.576	92.23 \pm 3.949	96.571	96.805	same
wine	97.68 \pm 3.514	97.13 \pm 3.886	97.63 \pm 3.223	95.12 \pm 4.336	97.68	98.09	same
zoo	92.718 \pm 7.587	90.98 \pm 8.571	93.47 \pm 7.058	96.55 \pm 5.337	96.545	97.624	better

Significance Factor 0.005

All classifiers had a hard time when trying to classify “primary tumor” dataset. This is due to the high number of nominal values that “primary tumor” has for the class label, which is 22. Our classifier cannot do better on iris dataset. Naïve Bayes performs very good on some datasets, however it performs poor on the other datasets that other algorithms perform well on like; “autos”, “glass” and “segment”. On average our proposed approach is significantly better on most of the datasets with all metrics. These experiments gave us some motivating results.

5.2.2.1 Effects of Number of Algorithms

Our method decides which algorithms to be used for the final voting by providing it number of algorithms or a threshold score. In this section, we will examine the effects of number of algorithms used in the final model. This experiment has same configuration and framework with the previous experiment. The effects of number of classifiers will be discussed separately for each metric, just like the experiment above. X-axis of the graphs in this section represents the number of algorithms used in the final model. Number of algorithms used is varied between 1 to 10. Y-axis shows the corresponding result. Each line represents a dataset which are described on legend of the graph. All graphs are drawn from the results shown in Table B.1 and Table B.2; Table B.3 and Table B.4; and Table B.5 and Table B.6. All results can also from seen from these tables in Appendix B

As described in the previous section ROC area evaluation is not distinctive for these datasets. The number of classifiers effects the scores of the final model, however depending on the dataset it increases or decreases. It is hard to see the relation between number of algorithms and ROC area score, see the Figure 5.2. Different number of algorithms performs better on different datasets. It is hard to find a global number. However, most of the algorithms have their best performance around 3, 4 and 5 number of algorithms. If more than 3, 4, 5 algorithms are used, performance decreases. On hard classification tasks like the “primary tumor”, best performance is found by using only one algorithm. This is actually the same as using only the best performing algorithm, and voting

classifier does not have any effect because there is only one voter. Algorithms training on this datasets seem to over-fit faster. Datasets do not contain easily detectable pattern and final model gets complex with more than one algorithm. Final model tend to discover patterns that do not exists.

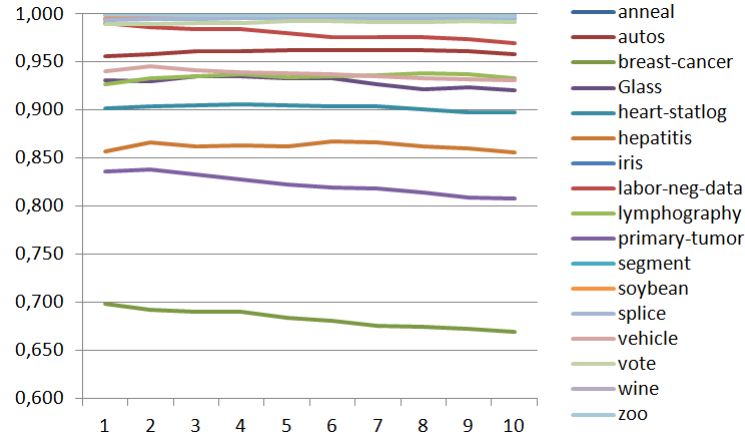


Figure 5.2: Effects of Number of Algorithms, ROC Area

With most of the datasets score increases with the number of algorithms with some exceptions. At some point it starts to decrease because of over-fitting. It is easier to see this when we remove the exceptions from the graph and zoom it, see Figure 5.3 and Figure 5.7.

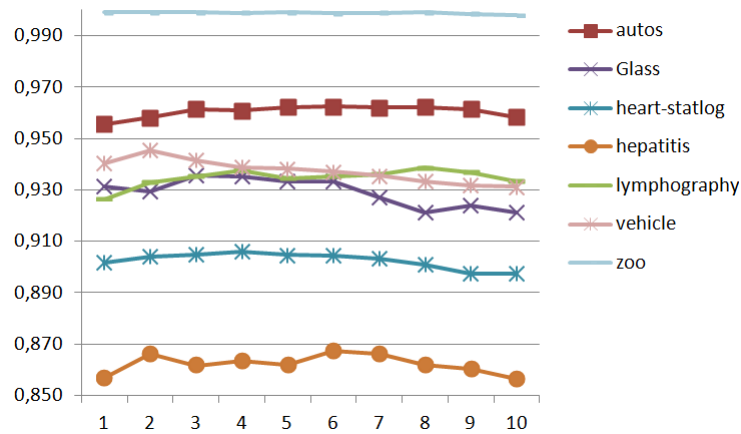


Figure 5.3: Effects of Number of Algorithms, ROC Area 2

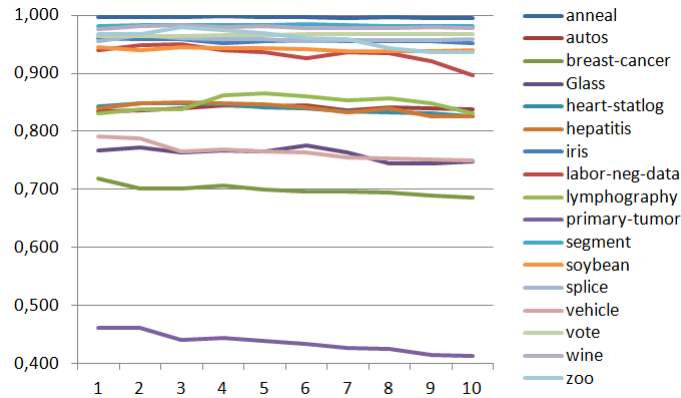


Figure 5.4: Effects of Number of Algorithms, F-measure

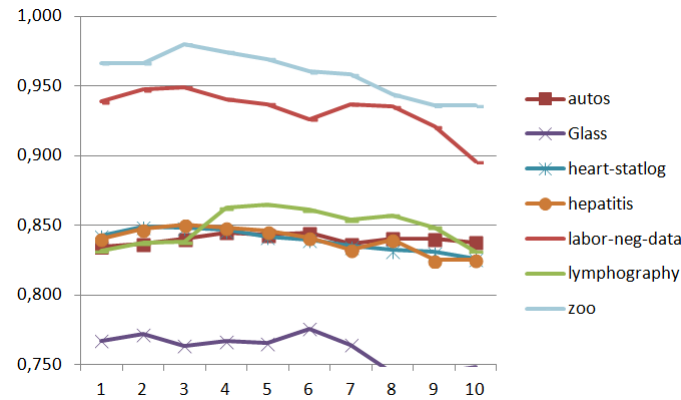


Figure 5.5: Effects of Number of Algorithms, F-measure 2

5.3 Limitations

One limitation of our proposed approach is the need for human expert intervention, which prevents it from being fully automatic. Number of algorithms or threshold score should be decided upon for algorithm selection. There has to be a human expert to decide number of algorithms to be used. Brute force can be used to select the best number by trying all numbers. However it would be very expensive if algorithm library is too large in means of computational resources. Optimum number of algorithms would be different for each dataset and it should be experimented. There is no good interval for it. This is experimented on the previous section. However, all machine learning algorithms should be experimented by tuning parameters in data mining tasks and different tests should be

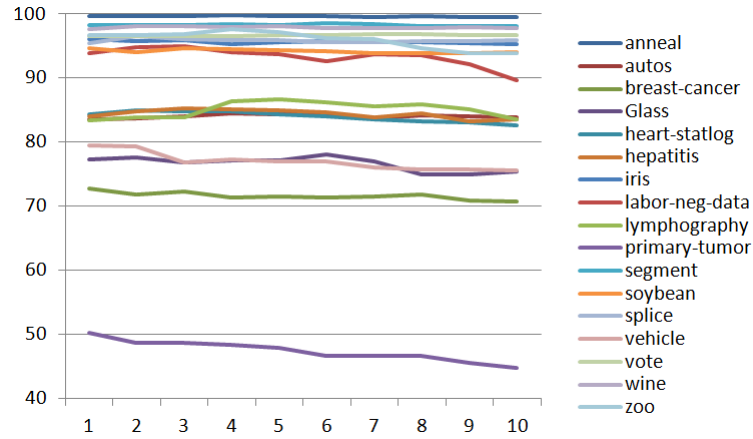


Figure 5.6: Effects of Number of Algorithms, Accuracy

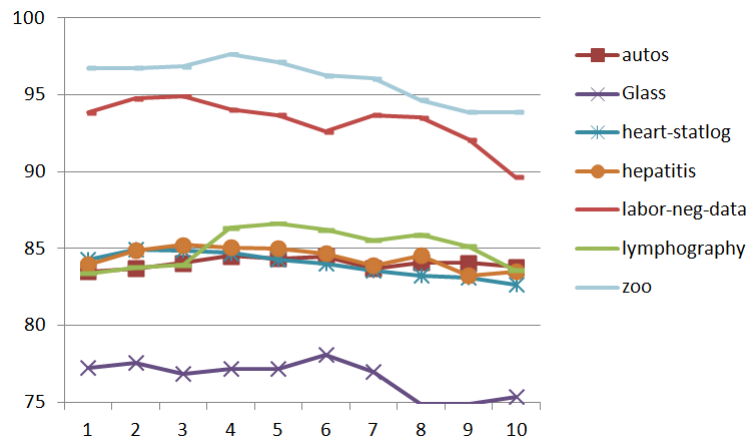


Figure 5.7: Effects of Number of Algorithms, Accuracy 2

executed. Best way to decide number of algorithms is by experimenting different values. As seen on the graphs above, final method performance tend to increase for a while and after some point it decreases but not always. In some cases, this heuristic will stuck in local optima. It is not possible to fully automate the process. This is the major limitation we currently see with our proposed approach.

Chapter 6

Conclusions

In this thesis, we analysed various powerful single classifier algorithms and their combinations as ensemble classifiers and proposed a new ensemble classifier model with improved performance for churn prediction problem in Mobile and Wireless Communication Services. We worked on KDD 2009 dataset and tried to improve or match highest scoring teams on the KDD 2009 Competition. Earlier in the study we have observed that preprocessing technique has a significant impact on the performance of a classifier. Single preprocessing technique will not perform well on all classifiers. In this study, we have built an ensemble voting classifier encompassing a set of well performing meta-classifiers. Meta-classifiers consists of preprocessing and classifier pairs. Best meta-classifiers are chosen from a library of classifiers. These classifiers vote for the prediction. The results demonstrated that the performance of churn prediction can significantly improve on a single computer with a practical time window for obtaining the results. We could not score as well as the winners of the competition. However our scores were very promising. Our score was close to top scores with less computational resources.

We have further investigated our algorithm on 17 UCI datasets. We compared 10 well studied algorithms and our approach on UCI datasets. Our approach scored better or same on almost all datasets in both ROC area, F-measure and accuracy.

We experimented the number of algorithms used in the final model. However, we cannot find a global value or a routine to calculate the value for number of algorithms. This stay as limitation to our approach. Number of algorithms is still parametrized. A human expert needs to experiment the algorithm to find a optimum or near optimum value for this.

Our approach is integrated with Weka libraries. Therefore, one can use it on his/her own application with Java programming language when made publicly available. It is also implemented by multi-threading. Each algorithm in the final model and each fold of the validation process can be executed in different threads.

Our future work is to find a logical value or way to find the number of algorithms to be used in the final model. This value will be different for each dataset and algorithms used. Therefore; rather than a common value, we will try to find a way or formula to calculate the optimum number of classifiers. If we can achieve this, all process can be fully automated. Our experiments in these thesis shows that it will be very complex to formulate that. Within the time given for this graduate study, we observed that the proposed technique is promising and it needs further study to overcome its limitations.

Appendix A

Numeric Variables in KDD 2009 Dataset

Table A.1: Statistics of All Numeric Variables

Variable	Min	Max	Mean	Std. Dev.	# missing	% missing	Unique	Distinct
1	0.00	680.00	11.49	40.71	49298	99	8	18
2	0.00	5.00	0.00	0.14	48759	98	1	2
3	0.00	130668.00	425.30	4270.19	48760	98	104	146
4	0.00	27.00	0.13	1.28	48421	97	1	4
5	0.00	6048550.00	238793.33	644125.91	48513	97	563	571
6	0.00	131761.00	1326.44	2685.69	5529	11	533	1486
7	0.00	140.00	6.81	6.33	5539	11	2	8
8	NaN	NaN	NaN	NaN	50000	100	0	0
9	0.00	2300.00	48.15	154.78	49298	99	44	100
10	0.00	12300000.00	392588.45	927868.40	48513	97	523	534
11	8.00	40.00	8.63	2.87	48760	98	0	5
12	0.00	1184.00	16.07	64.19	49442	99	9	22
13	0.00	197872.00	1249.69	2794.95	5539	11	832	2634
14	0.00	48.00	0.74	3.71	48760	98	6	19
15	NaN	NaN	NaN	NaN	50000	100	0	0
16	0.00	434.92	120.24	72.39	48513	97	392	597
17	0.00	1220.00	11.39	49.49	48421	97	13	37

Continued on next page

Table A.1 – continued from previous page

Variable	Min	Max	Mean	Std. Dev.	# missing	% missing	unique	Distinct
18	0.00	948.00	7.22	34.42	48421	97	10	26
19	0.00	27.00	0.25	1.78	48421	97	1	4
20	NaN	NaN	NaN	NaN	50000	100	0	0
21	0.00	36272.00	234.52	565.56	5529	11	293	734
22	0.00	45340.00	290.25	704.49	5009	10	294	735
23	0.00	1555.00	7.54	49.45	48513	97	12	29
24	0.00	494.00	4.51	9.93	7230	14	31	93
25	0.00	13168.00	96.83	214.32	5009	10	97	271
26	0.00	9.00	0.07	0.53	48513	97	1	4
27	0.00	4.00	0.03	0.25	48513	97	1	3
28	-66.88	5158.56	224.51	98.52	5011	10	1857	4167
29	0.00	2.00	0.02	0.21	49298	99	0	2
30	0.00	95.00	7.44	8.85	49298	99	3	13
31	NaN	NaN	NaN	NaN	50000	100	0	0
32	NaN	NaN	NaN	NaN	50000	100	0	0
33	0.00	1090000.00	127021.01	496141.03	49153	98	294	298
34	0.00	56.00	1.17	4.38	48759	98	1	6
35	0.00	110.00	0.72	3.00	5009	10	2	13
36	0.00	2419600.00	159553.85	327715.06	48759	98	525	531

Continued on next page

Table A.1 – continued from previous page

Variable	Min	Max	Mean	Std. Dev.	# missing	% missing	unique	Distinct
37	0.00	11635020.00	648522.15	1382224.80	48421	97	534	550
38	0.00	18846900.00	2579106.93	3010076.46	5009	10	28820	30832
39	NaN	NaN	NaN	NaN	50000	100	0	0
40	0.00	2648.00	13.96	81.41	48759	98	8	27
41	0.00	728.00	26.65	61.85	49298	99	15	36
42	NaN	NaN	NaN	NaN	50000	100	0	0
43	0.00	625.00	4.16	20.57	48759	98	7	20
44	0.00	135.00	0.17	1.63	5009	10	3	8
45	0.00	60553.40	7256.13	8798.18	49656	99	342	343
46	0.00	1668.00	16.80	70.82	48759	98	21	50
47	0.00	168.00	2.19	10.19	49298	99	5	11
48	NaN	NaN	NaN	NaN	50000	100	0	0
49	0.00	18.00	0.09	0.88	48759	98	2	4
50	0.00	1660.00	35.69	127.03	49298	99	30	63
51	0.00	347806.40	43652.01	71229.49	46253	93	3540	3561
52	NaN	NaN	NaN	NaN	50000	100	0	0
53	0.00	12500000.00	654346.51	1379534.77	49298	99	387	397
54	0.00	32.00	4.79	5.83	48759	98	1	5
55	NaN	NaN	NaN	NaN	50000	100	0	0

Continued on next page

Table A.1 – continued from previous page

Variable	Min	Max	Mean	Std. Dev.	# missing	% missing	unique	Distinct
56	0.00	3012970.00	87964.67	273885.56	49354	99	192	195
57	0.00	7.00	3.51	2.03	0	0	10742	25614
58	0.00	9676800.00	164061.33	645780.54	49298	99	238	244
59	0.00	5443200.00	414596.52	767371.19	49180	98	562	566
60	0.00	1215.00	9.54	49.18	48513	97	17	47
61	0.00	4624.00	40.46	190.75	49153	98	13	39
62	0.00	441.00	5.10	24.41	49442	99	6	12
63	0.00	2952.00	40.20	155.74	49306	99	19	48
64	0.00	223728.30	26526.47	32764.47	49762	100	231	233
65	9.00	180.00	14.87	10.13	5539	11	1	15
66	0.00	4600.00	96.84	311.25	49306	99	44	100
67	0.00	5.00	0.02	0.29	48513	97	0	2
68	0.00	11921.00	85.60	397.06	48759	98	31	84
69	0.00	1870000.00	3531710.24	4937147.40	48513	97	651	706
70	0.00	6189690.00	400340.56	809681.50	48513	97	510	521
71	0.00	14106.00	137.06	541.15	48871	98	49	119
72	3.00	24.00	4.19	2.30	22380	45	0	8
73	4.00	264.00	66.64	52.86	0	0	1	131
74	0.00	142156.00	103.66	766.71	5539	11	115	371

Continued on next page

Table A.1 – continued from previous page

Variable	Min	Max	Mean	Std. Dev.	# missing	% missing	unique	Distinct
75	0.00	70.00	6.50	8.63	48759	98	2	13
76	0.00	19353600.00	1490153.84	1853693.47	5009	10	27898	29743
77	0.00	666.00	10.41	41.69	49298	99	8	23
78	0.00	39.00	0.53	2.14	5009	10	1	13
79	NaN	NaN	NaN	NaN	50000	100	0	0
80	0.00	3637080.00	54421.04	188329.20	48513	97	394	400
81	0.00	1814403.00	103084.05	106272.13	5529	11	42298	43042
82	0.00	9.00	2.43	2.30	48421	97	0	4
83	0.00	6335.00	20.02	88.13	5009	10	83	195
84	0.00	4160.00	42.53	243.24	48760	98	59	96
85	0.00	1178.00	8.46	20.62	5009	10	53	149
86	0.00	2425840.00	286892.97	458778.44	49298	99	441	448
87	0.00	28.00	5.42	5.25	49298	99	0	5
88	0.00	5105.00	68.95	226.47	48917	98	29	88
89	0.00	300.00	5.52	17.21	49354	99	6	16
90	0.00	7.00	0.02	0.37	49298	99	0	2
91	0.00	9404.00	91.37	360.77	48871	98	49	119
92	0.00	1865367.00	170679.44	222820.57	49829	100	167	169
93	2.00	8.00	2.13	0.62	48513	97	0	4

Continued on next page

Table A.1 – continued from previous page

Variable	Min	Max	Mean	Std. Dev.	# missing	% missing	unique	Distinct
94	0.00	5640330.00	98671.07	180633.30	22380	45	15958	20002
95	0.00	3629172.00	109771.41	360087.86	48759	98	263	267
96	0.00	498.00	4.65	21.23	48759	98	13	33
97	0.00	36.00	0.92	3.40	48513	97	2	7
98	0.00	1245440.00	21295.06	83413.16	49442	99	112	115
99	0.00	2848.00	26.28	94.60	48421	97	16	47
100	0.00	28.00	0.89	3.15	49298	99	1	5
101	0.00	1755.00	19.80	83.99	49127	98	12	28
102	0.00	279458.10	28765.80	33844.79	49549	99	441	445
103	0.00	2002.00	18.78	85.56	48513	97	18	39
104	0.00	10152.00	100.79	379.77	49180	98	17	62
105	0.00	6768.00	67.19	253.18	49180	98	17	62
106	0.00	2274345.00	38164.13	160059.60	48421	97	255	261
107	0.00	1932.00	5.09	52.12	48513	97	12	24
108	0.00	7257600.00	193779.52	517817.76	49298	99	334	341
109	0.00	7456.00	60.89	141.25	7230	14	76	209
110	6.00	30.00	6.68	2.71	49298	99	1	5
111	0.00	3628806.00	284824.23	497265.56	48871	98	790	794
112	0.00	10352.00	66.22	157.64	5009	10	90	230

Continued on next page

Table A.1 – continued from previous page

Variable	Min	Max	Mean	Std. Dev.	# missing	% missing	unique	Distinct
113	-9803600.00	9932480.00	-153278.61	761372.98	0	0	47975	48511
114	0.00	4493460.00	608132.61	932057.05	48759	98	639	643
115	0.00	5337.00	37.75	194.75	49180	98	10	35
116	0.00	3.00	0.06	0.43	49298	99	0	2
117	0.00	2073600.00	129837.29	288764.75	48421	97	640	656
118	3.00	3.00	3.00	0.00	49829	100	0	1
119	0.00	105060.00	916.11	2165.43	5529	11	598	1487
120	0.00	2502.00	31.64	122.80	48513	97	27	64
121	0.00	672.00	6.95	30.77	49298	99	13	33
122	0.00	6.00	0.04	0.38	48759	98	1	3
123	0.00	13086.00	60.19	221.55	5009	10	135	298
124	0.00	9676800.00	223079.15	744424.06	48421	97	340	347
125	0.00	5436045.00	27887.63	90128.38	5539	11	4869	10505
126	-32.00	68.00	-0.55	22.53	13920	28	0	51
127	0.00	2480.00	28.09	124.35	48917	98	14	39
128	0.00	7147.00	96.53	317.06	48917	98	29	88
129	0.00	636.00	10.69	36.85	49298	99	22	45
130	0.00	3.00	0.50	1.11	48760	98	0	2
131	0.00	44200000.00	4064785.37	25309532.27	49298	99	149	151

Continued on next page

Table A.1 – continued from previous page

Variable	Min	Max	Mean	Std. Dev.	# missing	% missing	unique	Distinct
132	0.00	184.00	3.52	9.99	5009	10	2	19
133	0.00	15009900.00	2273571.92	2438599.59	5009	10	36193	37603
134	0.00	5735340.00	437340.38	604341.47	5009	10	31335	33181
135	0.00	710.43	199.92	132.54	48421	97	492	679
136	0.00	1209602.00	110690.91	189493.92	49306	99	531	534
137	0.00	204.00	3.83	12.43	49298	99	7	19
138	0.00	2.00	0.00	0.05	48421	97	1	2
139	0.00	2700560.00	181541.37	393358.34	48513	97	666	674
140	0.00	520545.00	1381.26	3990.51	5539	11	892	2648
141	NaN	NaN	NaN	NaN	50000	100	0	0
142	0.00	12.00	1.13	1.97	49298	99	0	4
143	0.00	18.00	0.06	0.64	5009	10	0	4
144	0.00	81.00	11.73	11.72	5529	11	0	10
145	0.00	6126.00	58.44	228.10	48421	97	29	88
146	0.00	96.00	2.78	7.88	48513	97	2	10
147	0.00	8.00	1.70	1.48	48513	97	0	5
148	0.00	18808.00	140.49	633.37	48513	97	49	119
149	0.00	16934400.00	294920.80	656894.63	7230	14	16227	18652
150	0.00	6048000.00	157687.62	413611.16	48421	97	582	600

Continued on next page

Table A.1 – continued from previous page

Variable	Min	Max	Mean	Std. Dev.	# missing	% missing	unique	Distinct
151	0.00	1200.00	11.06	53.04	49153	98	6	19
152	0.00	66.00	8.31	9.10	48421	97	2	12
153	0.00	13907800.00	6181967.17	4348926.09	5009	10	31995	36397
154	0.00	15000000.00	1538157.99	2279848.25	49298	99	379	388
155	0.00	35.00	0.80	3.04	48421	97	1	8
156	0.00	8050.00	169.47	544.69	49306	99	44	100
157	0.00	5440.00	33.07	187.53	48871	98	24	64
158	0.00	87.00	1.90	6.33	49127	98	10	18
159	0.00	99.00	4.71	11.03	48759	98	1	10
160	0.00	4862.00	38.80	99.50	5009	10	157	402
161	0.00	81.00	3.37	8.58	48421	97	1	9
162	0.00	10900000.00	336038.68	973436.64	48759	98	462	471
163	0.00	14515200.00	486077.96	848863.81	5009	10	20618	22957
164	0.00	138.00	1.75	8.10	48421	97	8	19
165	0.00	1209600.00	28842.36	115786.91	49127	98	200	204
166	0.00	2261.00	22.55	96.21	48513	97	24	48
167	NaN	NaN	NaN	NaN	50000	100	0	0
168	0.00	1270.48	332.94	102.88	49298	99	344	453
169	NaN	NaN	NaN	NaN	50000	100	0	0

Continued on next page

Table A.1 – continued from previous page

Variable	Min	Max	Mean	Std. Dev.	# missing	% missing	unique	Distinct
170	0.00	957.00	3.04	27.92	48759	98	9	18
171	0.00	5443200.00	367451.72	604237.74	48917	98	731	746
172	0.00	119.00	9.74	10.40	48513	97	4	13
173	0.00	6.00	0.01	0.13	5009	10	0	4
174	0.00	1156.00	7.04	36.19	48421	97	10	29
175	NaN	NaN	NaN	NaN	50000	100	0	0
176	0.00	892.00	4.72	31.00	48760	98	9	28
177	0.00	8554350.00	618888.39	1306030.28	48759	98	436	443
178	0.00	1345.00	16.69	59.02	49354	99	12	30
179	0.00	890.00	3.14	28.21	48421	97	6	15
180	0.00	1430000.00	3777073.87	3786347.29	49298	99	501	518
181	0.00	49.00	0.61	2.50	5009	10	2	7
182	0.00	11994780.00	1416638.10	2279786.37	48421	97	800	819
183	0.00	3048400.00	77773.80	201618.77	48759	98	366	374
184	0.00	1200.00	8.46	46.97	48759	98	14	31
185	NaN	NaN	NaN	NaN	50000	100	0	0
186	0.00	102.00	3.30	8.78	49298	99	4	13
187	0.00	910.00	16.54	60.22	49298	99	27	57
188	-6.42	628.62	167.37	113.98	48759	98	393	535

Continued on next page

Table A.1 – continued from previous page

Variable	Min	Max	Mean	Std. Dev.	# missing	% missing	unique	Distinct
189	6.00	642.00	270.14	86.71	28978	58	9	97
190	0.00	230427.00	22007.05	29085.15	49667	99	326	328

Appendix B

Evaluations with Number of Algorithms

Table B.1: Effects of Number of Algorithms, ROC Area

Datasets	Number Of Classifiers				
	1	2	3	4	5
anneal	0.999 \mp 0.001	0.999 \mp 0.001	0.999 \mp 0.000	0.999 \mp 0.001	0.999 \mp 0.001
autos	0.956 \mp 0.006	0.958 \mp 0.005	0.961 \mp 0.005	0.961 \mp 0.005	0.962 \mp 0.004
breast-cancer	0.698 \mp 0.006	0.692 \mp 0.007	0.690 \mp 0.007	0.690 \mp 0.008	0.684 \mp 0.010
Glass	0.931 \mp 0.003	0.929 \mp 0.004	0.935 \mp 0.005	0.935 \mp 0.003	0.933 \mp 0.004
heart-statlog	0.902 \mp 0.003	0.904 \mp 0.003	0.905 \mp 0.004	0.906 \mp 0.003	0.905 \mp 0.003
hepatitis	0.857 \mp 0.005	0.866 \mp 0.009	0.862 \mp 0.009	0.864 \mp 0.011	0.862 \mp 0.011
iris	0.994 \mp 0.000	0.996 \mp 0.001	0.996 \mp 0.001	0.995 \mp 0.001	0.995 \mp 0.002
labor-neg-data	0.990 \mp 0.005	0.986 \mp 0.006	0.984 \mp 0.007	0.984 \mp 0.006	0.980 \mp 0.008
lymphography	0.926 \mp 0.012	0.933 \mp 0.011	0.935 \mp 0.009	0.937 \mp 0.008	0.934 \mp 0.008
primary-tumor	0.836 \mp 0.001	0.838 \mp 0.001	0.833 \mp 0.004	0.827 \mp 0.004	0.822 \mp 0.004
segment	0.999 \mp 0.000	0.999 \mp 0.000	0.999 \mp 0.000	0.999 \mp 0.000	0.999 \mp 0.000
soybean	0.996 \mp 0.001	0.997 \mp 0.000	0.997 \mp 0.000	0.997 \mp 0.000	0.997 \mp 0.000
splice	0.994 \mp 0.000	0.995 \mp 0.000	0.995 \mp 0.000	0.995 \mp 0.000	0.995 \mp 0.000
vehicle	0.940 \mp 0.003	0.945 \mp 0.002	0.941 \mp 0.002	0.939 \mp 0.003	0.938 \mp 0.003
vote	0.989 \mp 0.002	0.989 \mp 0.002	0.990 \mp 0.002	0.990 \mp 0.002	0.993 \mp 0.001
wine	0.998 \mp 0.001	0.999 \mp 0.000	0.999 \mp 0.000	0.999 \mp 0.000	0.999 \mp 0.000
zoo	0.999 \mp 0.000	0.999 \mp 0.000	0.999 \mp 0.000	0.999 \mp 0.001	0.999 \mp 0.001

Table B.2: Effects of Number of Algorithms, ROC Area *continued*

Datasets	Number Of Classifiers						Max
	6	7	8	9	10		
anneal	0.999 \mp 0.001	0.999 \mp 0.001	0.999 \mp 0.001	0.999 \mp 0.001	0.999 \mp 0.001	0.999 \mp 0.001	0.999
autos	0.962 \mp 0.004	0.962 \mp 0.004	0.962 \mp 0.004	0.961 \mp 0.004	0.958 \mp 0.004	0.958 \mp 0.004	0.962
breast-cancer	0.680 \mp 0.010	0.676 \mp 0.011	0.674 \mp 0.014	0.672 \mp 0.015	0.669 \mp 0.015	0.669 \mp 0.015	0.698
Glass	0.933 \mp 0.004	0.927 \mp 0.004	0.921 \mp 0.004	0.924 \mp 0.004	0.921 \mp 0.005	0.921 \mp 0.005	0.935
heart-statlog	0.904 \mp 0.003	0.903 \mp 0.002	0.901 \mp 0.003	0.897 \mp 0.004	0.897 \mp 0.004	0.897 \mp 0.004	0.906
hepatitis	0.867 \mp 0.007	0.866 \mp 0.006	0.862 \mp 0.007	0.860 \mp 0.007	0.856 \mp 0.007	0.856 \mp 0.007	0.867
iris	0.995 \mp 0.002	0.995 \mp 0.002	0.995 \mp 0.002	0.995 \mp 0.002	0.995 \mp 0.002	0.995 \mp 0.002	0.996
labor-neg-data	0.976 \mp 0.009	0.975 \mp 0.007	0.975 \mp 0.007	0.974 \mp 0.006	0.969 \mp 0.007	0.969 \mp 0.007	0.990
lymphography	0.935 \mp 0.007	0.936 \mp 0.007	0.939 \mp 0.006	0.937 \mp 0.007	0.933 \mp 0.008	0.933 \mp 0.008	0.939
primary-tumor	0.819 \mp 0.004	0.819 \mp 0.004	0.814 \mp 0.005	0.809 \mp 0.006	0.808 \mp 0.006	0.808 \mp 0.006	0.838
segment	0.999 \mp 0.000	0.999 \mp 0.000	0.999 \mp 0.000	0.999 \mp 0.000	0.999 \mp 0.000	0.999 \mp 0.000	0.999
soybean	0.997 \mp 0.000	0.997 \mp 0.000	0.997 \mp 0.000	0.996 \mp 0.000	0.997 \mp 0.000	0.997 \mp 0.000	0.997
splice	0.995 \mp 0.000	0.995 \mp 0.000	0.995 \mp 0.000	0.995 \mp 0.000	0.995 \mp 0.000	0.995 \mp 0.000	0.995
vehicle	0.937 \mp 0.003	0.935 \mp 0.003	0.933 \mp 0.002	0.932 \mp 0.002	0.931 \mp 0.002	0.931 \mp 0.002	0.945
vote	0.993 \mp 0.001	0.992 \mp 0.001	0.992 \mp 0.001	0.992 \mp 0.001	0.992 \mp 0.001	0.992 \mp 0.001	0.993
wine	0.999 \mp 0.000	0.999 \mp 0.000	0.999 \mp 0.000	0.999 \mp 0.000	0.999 \mp 0.001	0.999 \mp 0.001	0.999
zoo	0.999 \mp 0.000	0.999 \mp 0.000	0.999 \mp 0.000	0.999 \mp 0.000	0.998 \mp 0.000	0.998 \mp 0.000	0.999

Table B.3: Effects of Number of Algorithms, F-measure

Datasets	Number Of Classifiers				
	1	2	3	4	5
anneal	0.997 \mp 0.002	0.997 \mp 0.002	0.997 \mp 0.002	0.998 \mp 0.002	0.997 \mp 0.002
autos	0.835 \mp 0.018	0.837 \mp 0.014	0.840 \mp 0.012	0.845 \mp 0.011	0.843 \mp 0.010
breast-cancer	0.718 \mp 0.008	0.702 \mp 0.009	0.701 \mp 0.011	0.706 \mp 0.013	0.700 \mp 0.013
Glass	0.767 \mp 0.019	0.772 \mp 0.018	0.764 \mp 0.014	0.767 \mp 0.012	0.766 \mp 0.010
heart-statlog	0.843 \mp 0.006	0.849 \mp 0.005	0.848 \mp 0.008	0.847 \mp 0.006	0.842 \mp 0.006
hepatitis	0.840 \mp 0.006	0.847 \mp 0.005	0.850 \mp 0.006	0.848 \mp 0.010	0.846 \mp 0.010
iris	0.961 \mp 0.007	0.958 \mp 0.005	0.959 \mp 0.004	0.952 \mp 0.003	0.955 \mp 0.004
labor-neg-data	0.939 \mp 0.014	0.948 \mp 0.015	0.949 \mp 0.016	0.941 \mp 0.014	0.937 \mp 0.014
lymphography	0.832 \mp 0.012	0.837 \mp 0.016	0.838 \mp 0.011	0.862 \mp 0.014	0.865 \mp 0.013
primary-tumor	0.462 \mp 0.009	0.462 \mp 0.009	0.441 \mp 0.008	0.445 \mp 0.007	0.440 \mp 0.009
segment	0.982 \mp 0.001	0.982 \mp 0.001	0.983 \mp 0.001	0.983 \mp 0.001	0.983 \mp 0.001
soybean	0.945 \mp 0.006	0.939 \mp 0.003	0.945 \mp 0.005	0.944 \mp 0.005	0.944 \mp 0.005
splice	0.954 \mp 0.001	0.965 \mp 0.001	0.960 \mp 0.001	0.959 \mp 0.001	0.958 \mp 0.002
vehicle	0.792 \mp 0.006	0.789 \mp 0.008	0.766 \mp 0.013	0.769 \mp 0.011	0.766 \mp 0.010
vote	0.965 \mp 0.002	0.965 \mp 0.002	0.965 \mp 0.002	0.965 \mp 0.002	0.966 \mp 0.002
wine	0.976 \mp 0.007	0.981 \mp 0.006	0.982 \mp 0.005	0.979 \mp 0.005	0.980 \mp 0.006
zoo	0.967 \mp 0.006	0.967 \mp 0.006	0.980 \mp 0.007	0.974 \mp 0.005	0.969 \mp 0.006

Table B.4: Effects of Number of Algorithms, F-measure *continued*

Datasets	Number Of Classifiers						Max
	6	7	8	9	10		
anneal	0.996 \mp 0.001	0.994 \mp 0.001	0.996 \mp 0.001	0.995 \mp 0.001	0.994 \mp 0.001	0.998	
autos	0.845 \mp 0.016	0.837 \mp 0.013	0.841 \mp 0.016	0.840 \mp 0.016	0.838 \mp 0.012	0.845	
breast-cancer	0.696 \mp 0.015	0.696 \mp 0.016	0.695 \mp 0.017	0.689 \mp 0.014	0.686 \mp 0.015	0.718	
Glass	0.776 \mp 0.016	0.765 \mp 0.011	0.744 \mp 0.009	0.744 \mp 0.009	0.748 \mp 0.013	0.776	
heart-statlog	0.840 \mp 0.008	0.835 \mp 0.007	0.832 \mp 0.006	0.831 \mp 0.009	0.826 \mp 0.008	0.849	
hepatitis	0.841 \mp 0.010	0.833 \mp 0.012	0.840 \mp 0.010	0.825 \mp 0.015	0.826 \mp 0.014	0.850	
iris	0.957 \mp 0.005	0.956 \mp 0.005	0.955 \mp 0.006	0.955 \mp 0.007	0.952 \mp 0.006	0.961	
labor-neg-data	0.926 \mp 0.021	0.937 \mp 0.014	0.935 \mp 0.014	0.921 \mp 0.016	0.896 \mp 0.017	0.949	
lymphography	0.861 \mp 0.011	0.854 \mp 0.011	0.857 \mp 0.011	0.849 \mp 0.010	0.831 \mp 0.016	0.865	
primary-tumor	0.434 \mp 0.010	0.427 \mp 0.013	0.426 \mp 0.013	0.414 \mp 0.012	0.414 \mp 0.009	0.462	
segment	0.985 \mp 0.001	0.983 \mp 0.001	0.981 \mp 0.002	0.981 \mp 0.002	0.981 \mp 0.002	0.985	
soybean	0.941 \mp 0.005	0.938 \mp 0.005	0.938 \mp 0.005	0.939 \mp 0.005	0.940 \mp 0.005	0.945	
splice	0.955 \mp 0.002	0.956 \mp 0.002	0.957 \mp 0.002	0.958 \mp 0.002	0.959 \mp 0.001	0.965	
vehicle	0.764 \mp 0.011	0.756 \mp 0.007	0.753 \mp 0.010	0.752 \mp 0.008	0.750 \mp 0.009	0.792	
vote	0.967 \mp 0.002	0.968 \mp 0.003	0.968 \mp 0.002	0.967 \mp 0.002	0.967 \mp 0.002	0.968	
wine	0.977 \mp 0.005	0.978 \mp 0.003	0.978 \mp 0.005	0.979 \mp 0.004	0.978 \mp 0.007	0.982	
zoo	0.960 \mp 0.010	0.958 \mp 0.009	0.944 \mp 0.010	0.936 \mp 0.008	0.936 \mp 0.008	0.980	

Table B.5: Effects of Number of Algorithms, Accuracy

Datasets	Number Of Classifiers				
	1	2	3	4	5
anneal	99.710 \mp 0.188	99.710 \mp 0.188	99.710 \mp 0.188	99.766 \mp 0.169	99.688 \mp 0.164
autos	83.512 \mp 1.861	83.707 \mp 1.417	84.049 \mp 1.196	84.537 \mp 1.047	84.341 \mp 0.987
breast-cancer	72.762 \mp 0.789	71.818 \mp 1.004	72.203 \mp 0.992	71.399 \mp 1.379	71.503 \mp 1.469
Glass	77.243 \mp 1.916	77.570 \mp 1.846	76.869 \mp 1.406	77.196 \mp 1.197	77.196 \mp 1.020
heart-statlog	84.296 \mp 0.554	84.926 \mp 0.470	84.852 \mp 0.819	84.704 \mp 0.598	84.259 \mp 0.556
hepatitis	83.935 \mp 0.933	84.839 \mp 0.595	85.226 \mp 0.609	85.032 \mp 0.632	84.968 \mp 0.959
iris	96.067 \mp 0.696	95.800 \mp 0.521	95.933 \mp 0.359	95.200 \mp 0.267	95.533 \mp 0.427
labor-neg-data	93.860 \mp 1.414	94.737 \mp 1.569	94.912 \mp 1.655	94.035 \mp 1.404	93.684 \mp 1.404
lymphography	83.378 \mp 1.098	83.784 \mp 1.570	83.919 \mp 1.038	86.351 \mp 1.378	86.622 \mp 1.345
primary-tumor	50.206 \mp 0.720	48.673 \mp 0.904	48.702 \mp 0.860	48.407 \mp 0.652	47.847 \mp 0.344
segment	98.199 \mp 0.133	98.247 \mp 0.134	98.307 \mp 0.143	98.320 \mp 0.124	98.290 \mp 0.097
soybean	94.583 \mp 0.559	93.982 \mp 0.337	94.583 \mp 0.472	94.407 \mp 0.462	94.378 \mp 0.525
splice	95.436 \mp 0.106	96.483 \mp 0.084	96.009 \mp 0.108	95.881 \mp 0.150	95.831 \mp 0.162
vehicle	79.421 \mp 0.613	79.338 \mp 0.766	76.844 \mp 1.215	77.234 \mp 1.056	77.009 \mp 0.930
vote	96.460 \mp 0.211	96.460 \mp 0.211	96.460 \mp 0.211	96.506 \mp 0.200	96.598 \mp 0.200
wine	97.640 \mp 0.655	98.090 \mp 0.573	98.034 \mp 0.453	97.865 \mp 0.490	98.034 \mp 0.628
zoo	96.733 \mp 0.634	96.733 \mp 0.634	96.832 \mp 0.741	97.624 \mp 0.485	97.129 \mp 0.533

Table B.6: Effects of Number of Algorithms, Accuracy *continued*

Datasets	Number Of Classifiers						Max
	6	7	8	9	10		
anneal	99.610 \mp 0.143	99.432 \mp 0.105	99.588 \mp 0.141	99.532 \mp 0.130	99.454 \mp 0.078	99.766	
autos	84.488 \mp 1.555	83.707 \mp 1.313	84.098 \mp 1.650	84.049 \mp 1.619	83.805 \mp 1.191	84.537	
breast-cancer	71.259 \mp 1.178	71.538 \mp 0.901	71.748 \mp 1.661	70.909 \mp 1.562	70.769 \mp 1.573	72.762	
Glass	78.084 \mp 1.584	76.963 \mp 1.025	74.907 \mp 0.837	74.907 \mp 0.837	75.374 \mp 1.306	78.084	
heart-statlog	84.037 \mp 0.767	83.556 \mp 0.707	83.259 \mp 0.593	83.111 \mp 0.910	82.630 \mp 0.819	84.926	
hepatitis	84.645 \mp 1.072	83.871 \mp 1.117	84.516 \mp 1.040	83.226 \mp 1.443	83.484 \mp 1.419	85.226	
iris	95.733 \mp 0.533	95.600 \mp 0.533	95.533 \mp 0.600	95.467 \mp 0.653	95.200 \mp 0.581	96.067	
labor-neg-data	92.632 \mp 2.046	93.684 \mp 1.404	93.509 \mp 1.370	92.105 \mp 1.617	89.649 \mp 1.655	94.912	
lymphography	86.216 \mp 1.139	85.541 \mp 1.011	85.878 \mp 1.066	85.135 \mp 0.956	83.581 \mp 1.512	86.622	
primary-tumor	46.637 \mp 0.829	46.637 \mp 1.248	46.578 \mp 1.212	45.457 \mp 0.973	44.779 \mp 0.756	50.206	
segment	98.476 \mp 0.100	98.329 \mp 0.146	98.091 \mp 0.189	98.095 \mp 0.193	98.056 \mp 0.168	98.476	
soybean	94.143 \mp 0.468	93.880 \mp 0.531	93.865 \mp 0.509	93.895 \mp 0.512	94.026 \mp 0.453	94.583	
splice	95.495 \mp 0.200	95.602 \mp 0.181	95.690 \mp 0.174	95.771 \mp 0.182	95.900 \mp 0.122	96.483	
vehicle	76.903 \mp 1.024	75.993 \mp 0.688	75.697 \mp 0.916	75.638 \mp 0.752	75.579 \mp 0.819	79.421	
vote	96.736 \mp 0.200	96.805 \mp 0.261	96.805 \mp 0.191	96.736 \mp 0.225	96.667 \mp 0.212	96.805	
wine	97.697 \mp 0.467	97.809 \mp 0.303	97.809 \mp 0.467	97.921 \mp 0.439	97.809 \mp 0.686	98.090	
zoo	96.238 \mp 0.970	96.040 \mp 0.886	94.653 \mp 1.010	93.861 \mp 0.863	93.861 \mp 0.863	97.624	

BIBLIOGRAPHY

- [1] 2009 Knowledge Discovery and Data Mining Competition, <http://www.kddcup-orange.com>, *Last Accessed in May 2013*
- [2] I. Guyon, V. Lemaire, M. Boulle, G. Dror, D. Vogel(2009). “The 2009 Knowledge Discovery in Data Competition” Challenges in Machine Learning, Volume 3
- [3] I. Guyon, V. Lemaire, M. Boulle, G. Dror, D. Vogel(2009). “Analysis of the KDD Cup 2009: Fast Scoring on a Large Customer Database”, JMLR: Workshop and Conference Proceedings 7, 1-22.
- [4] H. Miller, S. Clarke, S. Lane, A. Lonie, D. Lazaridis, S. Petrovski, O. Jones(2009). “Predicting customer behaviour: The University of Melbournes KDD Cup report”, JMLR: Workshop and Conference Proceedings 7, 4555.
- [5] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten(2009). “The WEKA Data Mining Software: An Update”, SIGKDD Explorations, Volume 11, Issue 1.
- [6] <http://www.cs.waikato.ac.nz/ml/weka/>, *Last Accessed in December 2013*
- [7] GNU General Public License <http://www.gnu.org/licenses/gpl.html>, *Last Accessed in December 2013*
- [8] Mierswa, Ingo and Wurst, Michael and Klinkenberg, Ralf and Scholz, Martin and Euler, Timm(2006). “YALE: Rapid Prototyping for Complex Data Mining Tasks”, Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-06).

- [9] Leo Breiman(2001). “Random Forests”, *Machine Learning* 45, 5-32.
- [10] Alexandru Niculescu-Mizil, Claudia Perlich, Grzegorz Swirszcz, Vikas Sindhwani, Yan Liu, Prem Melville, Dong Wang, Jing Xiao, Jianying Hu, Moninder Singh, Wei Xiong Shang, Yan Feng Zhu(2009). “Winning the KDD Cup Orange Challenge with Ensemble Selection”, *JMLR: Workshop and Conference Proceedings* 7, 1-16.
- [11] Jianjun Xie, Viktoria Rojkova, Siddharth Pal, Stephen Coggeshall(2009). “A Combination of Boosting and Bagging for KDD Cup 2009 Fast Scoring on a Large Database”, *JMLR: Workshop and Conference Proceedings* 7, 35-43.
- [12] Yossi Richter, Elad Yom-Tov, Noam Slonim(2010). “Predicting Customer Churn in Mobile Networks through Analysis of Social Groups”, *SDM 2010*, 732-741.
- [13] Chih-Ping Wei, I-Tang Chiu(2002). “Turning Telecommunications Call Details to Churn Prediction: a Data Mining Approach”, *Expert Systems with Applications*, 23, 103-112.
- [14] SAS Institute (2000). “Best Practice in Churn Prediction”, SAS Institute White Paper.
- [15] Chao Zhu, Jiayin Qi, Chen Wang(2009). “An Experimental Study on Four Models of Customer Churn Prediction”, *Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics*, 3199 - 3204.
- [16] R. Quinlan(1993). “C4.5: Programs for Machine Learning”, Morgan Kaufmann, San Mateo.
- [17] J. H. Friedman(2001). “Greedy function approximation: a gradient boosting machine.” *Annals of Statistics*, 29(5), 1189-1232.
- [18] R.Quinlan(1986). “Induction of Decision Trees”, *Machine Learning* 1, 81-106.
- [19] S. Le Cessie, J. C. van Houwelingen(1992). “Ridge Estimators in Logistic Regression”, *Applied Statistics* 41, No 1, 192-201.

- [20] Breiman, Leo(1996). “Bagging predictors”, *Machine Learning* 24 (2), 123140.
- [21] Yoav Freund, Robert E. Schapire(1996), “Experiments with a new boosting algorithm.”, *Thirteenth International Conference on Machine Learning*, San Francisco, 148-156.
- [22] Vladislav Lazarov, Marius Capota “Churn Prediction”.
- [23] <http://www.stat.berkeley.edu/~breiman/RandomForests/>, *Last Accessed in December 2013*
- [24] I. H. Witten, E. Frank, M. Hall(2011). *Data Mining and Practical Machine Learning Tools and Techniques 3rd Edition*, San Francisco: Morgan Kaufmann.
- [25] Stuart Russell, Peter Norvig(2010). “Artificial Intelligence: A Modern Approach”, Pearson
- [26] Tom Fawcett(2006). “An introduction to ROC analysis”, *Pattern Recognition Letters* 27, 861 - 874.
- [27] Juan J. Rodriguez, Ludmila I. Kuncheva, Carlos J. Alonso(2006). “Rotation Forest: A New Classifier Ensemble Method”, *IEEE Transactions On Pattern Analysis and Machine Intelligence*, Vol.28, No.10, 1619 - 1630.
- [28] Wouter Verbeke, Karel Dejaeger, David Martens, Joon Hur, Bart Baesens(2012). “New Insights into Churn Prediction in the Telecommunication Sector: A Profit Driven Data Mining Approach”, *European Journal of Operational Research*, 218, 211-229.
- [29] Adnan Idris, Asifullah Khani, Yeon Soo Lee(2013). “Intelligent churn prediction in telecom: employing mRMR feature selection and RotBoost based ensemble classification”, *Applied Intelligence*, Volume 39, Issue 3, 659-672.
- [30] Coussement, K., De Bock, K.W.(2013). “Customer churn prediction in the online gambling industry: The benecial effect of ensemble learning”, *Journal of Business Research*, Volume 66, Issue 9, 1629-1636

- [31] Zhen-Yu Chen , Zhi-Ping Fan, Minghe Sun(2012). “A Hierarchical Multiple Kernel Support Vector Machine for Customer Churn Prediction Using Longitudinal Behavioral Data”, *European Journal of Operational Research*, 223, 461-472.
- [32] Yunjin Lee, Seungyong Lee, Ioannis Ivrissimtzis, and Hans-Peter Seidel(2006). “Overfitting Control for Surface Reconstruction”, In *Proc. fourth Eurographics Symposium on Geometry Processing 2006*, 231-235, Sardinia, Italy.
- [33] <http://en.wikipedia.org/wiki/Overfitting>, *Last Accessed in December 2013*
- [34] Pang-Ning Tan, Michael Steinbach and Vipin Kumar(2005). *Introduction to Data Mining 1st Edition*, Addison-Wesley.
- [35] Tom M. Mitchell(1997). *Machine Learning*, McGraw-Hill, 1997.
- [36] <http://www.oracle.com/us/technologies/java/overview/index.html>, *Last Accessed in December 2013*
- [37] <http://www.eclipse.org/>, *Last Accessed in December 2013*
- [38] [http://en.wikipedia.org/wiki/Cross-validation_\(statistics\)](http://en.wikipedia.org/wiki/Cross-validation_(statistics)), *Last Accessed in December 2013*
- [39] <http://www.sgi.com/tech/mlc/db/>, *Last Accessed in December 2013*
- [40] Payam Refaeilzadeh, Lei Tang, Huan Liu: “Cross Validation”, Arizona State University http://www.cse.iitb.ac.in/~tarung/smt/papers_ppt/ency-cross-validation.pdf, *Last Accessed in December 2013*
- [41] Bache, K., Lichman, M. (2013). *UCI Machine Learning Repository* <http://archive.ics.uci.edu/ml>. Irvine, CA: University of California, School of Information and Computer Science. *Last Accessed in December 2013*
- [42] U. Yabas, H. C. Cankaya, T. Ince(2012). “Customer Churn Prediction for Telecom Services”, 2012 IEEE 36th Annual Computer Software and Applications Conference - COMPSAC 2012, 358-359.

- [43] U. Yabas, H. C. Cankaya(2013). “Subscriber Churn Awareness in Telecommunications Services”, To appear in the proceedings of The 5th IEEE International Workshop on Management of Emerging Networks and Services.