Chapter 1

Introduction

Indian Telecom has emerged as one of the greatest economic success stories, registering a consistent overall growth rate of more than 35 percent over the past decade in terms of subscribers. Telecom consumers are an ideal choice for Knowledge Discovery in Database (KDD), since their usage statistics are well recorded, and have ample documentation. People are given a huge choice of offers and different service providers to decide upon. Recently, the mobile telecommunication market has changed from a rapidly growing market, into a state of saturation and fierce competition. The focus of telecommunication companies has therefore shifted from building a large customer base into keeping customers in house. For that reason, it is valuable to know which customers are likely to switch to a competitor in the near future. If a customer terminates a membership agreement with one company and become a customer of another competitor, this customer is called as lost customer or Churn customer. Since acquiring new customers is more expensive than retaining existing customers, churn prevention can be regarded as a popular way of reducing the company’s costs.

Churn prediction is currently a relevant subject in data mining and has been applied in the field of banking, mobile telecommunication, life insurances, and others. In fact, all companies who are dealing with long term customers can take advantage of churn prediction methods. The goal is to distinguish churners from non-churners as much as possible. When new customers are offered, the churn prediction model attempts to predict to which class each customer belongs.
Chapter 2

System Study

2.1 Project Domain

In this world of social networking and on-line purchases, need for better services, amenities, products reaches new dimensions. People start expressing themselves through social media, they start comparing products and services, share their views. The need for better understanding of consumers is necessary. The one who has better understanding of consumer turns out to be the market winner. So need for analysing consumer behavior is of prime importance. As a result of which several OLAP tools where developed. But as you all know the world is growing and the shared data as well so now it has reached a point where, an SQL based system can no longer handle such huge inflow of consumer emotion, so here we present before you an unique tool that uses NoSQL database, HBase to store consumer information. HBase is proven of storing large chunks of raw data (facebook uses HBase to store data). Here we analyze the so called 'garbage' to plot out the unbelievable and exciting patterns in consumer behavior.

2.2 Existing System

Established literature on customer churn uses various data mining technologies, such as Neural Networks, Clustering, Decision Tree, Regression, Support Vector Machine and ensemble of hybrid methods, to provide predictions. ID3 is one such Decision Tree based Algorithm. Conventionally data mining is done using DBMiner.

2.2.1 ID3 Algorithm

ID3 (Iterative Dichotomiser 3) is an algorithm used to generate a decision tree from a dataset. Some attributes split the data up more purely than others. That means that their values correspond more consistently with instances that have particular values of the target attribute (the one we want to predict) than those of another attribute. The ID3 algorithm begins with the original set S as the root node. On each iteration of the algorithm, it iterates through every unused attribute of the set S and calculates the information gain of that attribute. Then selects the attribute which has the largest information gain value. The set S is then split by the selected attribute to produce subsets of the data. The algorithm continues to recurse on each subset, considering only attributes never selected before.
Entropy

The entropy of a dataset can be considered to be how disordered it is. It is related to information, in the sense that the higher the entropy, or uncertainty, of some data, then the more information is required in order to completely describe that data. In building a decision tree, we aim to decrease the entropy of the dataset until we reach leaf nodes at which point the subset that we are left with has zero entropy and represents instances all of one class (all instances have the same value for the target attribute). The entropy of a dataset, $S$, with respect to the target attribute, with the following calculation:

$$\text{Entropy}(S) = \sum_{i=1}^{C} p_i \log_2 p_i$$

where $P_i$ is the proportion of instances in the dataset that take the $i$th value of the target attribute, which has $C$ different values.

Information Gain

Information gain is the reduction in entropy (Gain in information) that would result in splitting the data on an attribute, $A$. Gain$(S,A) = \text{Entropy}(S) - \sum_v \frac{|S_v|}{|S|} \text{Entropy}(S_v)$ where $v$ is a value of $A$, $|S_v|$ is the subset of instances of $S$ where $A$ takes the value $v$, and $|S|$ is the number of instances.

Nodes in the tree represent features, with branches representing possible values connecting the features. A leaf representing the class terminates a series of nodes and branches. Initially, the method starts to search an attribute with best information gain at root node and divide the tree into sub-trees. Similarly, each sub-tree is further separated recursively following the same rule. The partitioning stops if the leaf node is reached or there is no information gain. Once the tree is created, rules can be obtained by traversing each branch of the tree.

Disadvantages

- Data may be over-fitted or over-classified, if a small sample is tested
- only one attribute is tested at a time for making a decision
- Classifying continuous data may be computationally expensive, as many trees must be generated to see where to break the continuum.
Chapter 3

Software Requirement Specification

3.1 Introduction

3.1.1 Purpose

This document is in purpose of explaining what the project OLAP on Consumer Behaviour using Big Data Techniques is about. This project is an attempt to design and implement an application that can take Consumer Records as input and give Consumer churn prediction details as output. It will enable the user to know in advance about the valuable consumer who are about to churn. The aim of the project is to make an application that will give the user a more user friendly interface which can be used by even a normal business man with non technical background. Since the customer services centres of the carrier only have a fixed number of staff available to contact a small fraction of all subscribers, it is important for it to distinguish subscribers with high probability of churning from those with low probability so that, given the limited resources, the high probability churners can be contacted first.

3.1.2 Scope

With the advent of large volume of data, in this era of information explosion, there has been a growing need for handling this large pile of unrelated data. It is necessary to process these data in less amount of time. In this scenario BigData Technique comes into picture. The core concept of this project lies on the fact that BigData technique such as Apache HBase has the capability to process such vast amount of unrelated data in a swift manner. This project would be a milestone in the current computing world. Also the usage of the new data mining algorithm called Data Mining using Evolutionary Learning (DMEL) has enhanced the efficiency and results with around 95 percent accuracy can be obtained. With time, the application can be enhanced with more and more features that will make it very user-friendly and also will support the user.

3.1.3 Definitions and Abbreviations

- OLAP: OLAP stands for Online analytical processing. In computing OLAP is an approach to answering multi-dimensional analytical (MDA) queries swiftly. OLAP tools enable users to analyze multi-dimensional data interactively from multiple perspectives.

- HBase: HBase is an open source, non-relational, distributed database and is written in Java and runs on top of Hadoop Distributed File System.
• ID3 : Iterative Dichotomiser (ID3) Algorithm is an algorithm invented by Ross Quinlan used to generate a decision tree from a dataset. ID3 is the precursor to the C4.5 algorithm, and is typically used in the machine learning and natural language processing domains.

• DMEL : DMEL stands for Data Mining by Evolutionary Learning. It is a new data mining algorithm to handle classification problems of which the accuracy of each predictions made has to be estimated. In performing its tasks, DMEL searches through the possible rule space using an evolutionary approach.

3.1.4 Overview
The remainder of the document contains full description of the project for the better understanding of what is being intended to do. It lists all the functions performed by the system and the concerns in detail for each of the system functions and actions for the software developer’s assistance. The following sections are cross referenced by topic to increase understanding of any user group that intends to use the Application which performs OLAP on consumer behaviour using Big Data techniques.

3.2 General Description

3.2.1 Product Perspective
Whether it be business reporting for sales, marketing, management reporting, business process management (BPM), budgeting and forecasting, has to be handled by a businessman. Our project aims to simplify this job. By merely giving the Mobile consumer data records as input, user can get the desired consumer behaviour pattern, which is churn prediction, as output. The additional information derived from analysis of a single large set of related data helps to serve the purpose. The system is built using Apache Hbase.

3.2.2 Product Features

• Ability to provide ”just-in-time” information: Provide managers with the information they need to make effective decisions about an organization’s strategic directions.

• Diverse pattern Generation: Diverse hidden patterns can be generated from input data which otherwise has no meaning and is unordered.

• Reduced restrictions on building query: Since data analysis is a complex process which requires a multifaceted processing of available user data, complex querying is required which is supported throughout this project.

• Multidimensional Views of Data: A multidimensional view of data provides more than the ability to ”slice and dice”; it provides the foundation for analytical processing through flexible access to information.

• Complex Calculations: User possesses unrelated data which is difficult to analyse and process. Complex calculations are performed in this project for analysis of these unrelated data which otherwise would have been a difficult job for the user as generating a pattern from unrelated data is a tedious process.
• **Ability to model real business problems and more efficient use of people resources:** By providing the ability to model real business problems and a more efficient use of people resources, this project enables the organization as a whole to respond more quickly to market demands. Market responsiveness, in turn, often yields improved revenue and profitability.

• **Data storage reliability:** Storing data in hbase cluster so we have replica of each and every info and in the event of failure we can we replica and no data loss.

### 3.2.3 User Characteristics

The user of this system must be from a telecom company. He need not have a technical background. Knowledge about basic computer concepts and previous experience of data entry are necessary to use this product. User just needs to provide the consumer data records as input to get the desired consumer behaviour pattern as output.

### 3.2.4 Operating System Environment

**Software**
- Apache HBase
- Ubuntu or Any other Debian based System
- Java compiler

**Hardware**
- 32 /64 bit systems (if using cdh3)
- 64 bit system (if using cdh4)

### 3.3 Functional Requirements

#### 3.3.1 Input

- The Training dataset
- Select attributes and set their importance factor
- Test data

#### 3.3.2 Output

- The Rules generated by studying the training data
- the resultant predicted value of target attribute obtained on applying those rules on the test data
3.3.3 Processing

User on uploading the training set is asked to select required attributes from those present in the uploaded Database. Once selected then they are asked to set the importance value of those selected attributes by adjusting slider. This is followed by the Rule generation process where rules get stored. These rules are then applied on the Test Data which is uploaded next.

3.4 External Interface Requirements

3.4.1 User Interface

The starting screen enables the user to upload the training dataset and test dataset. A list of attributes in the data set are displayed on the screen. The user can select the required attributes by checking the check-box displayed against each attribute. In the next screen, the user can set the importance factor of each selected attribute by adjusting the slider. The final screen displays the customer churn list.

3.5 Software System Features

The Project has been divided into the following modules:

3.5.1 Java Interface Module

This module helps the user to input the customer data (training data and test data). After inputting the files, a list of attributes in the data set are displayed. The user can select the required attributes from the list and also set their importance factor, which are to be used for rule generation. The training and test data are then stored in HBase.

3.5.2 Rule Generation Module

DMEL algorithm is applied on the training data set. The evolutionary process begins with the generation of an initial set of first-order rules using a probabilistic induction technique and based on these rules, rules of higher order are obtained iteratively. When identifying interesting rules, an objective interestingness measure is used. The fitness of a chromosome is defined in terms of the probability that the attribute values of a record can be correctly determined using the rules it encodes.

3.5.3 Rule Application Module

The rules discovered by mining the training dataset are applied on the testing dataset to predict the target attribute values. The customer churn list are then displayed to the user.

3.5.4 Query Engine

Query engine is used for retrieving required data from HBase. Querying functions are called in the Rule generation and Rule application modules.
3.6 Other Non-Functional Requirements

3.6.1 Performance Requirements

The project requires multiple processors working in sync with each other. Hence, a high-speed multi-core machine is ideal for the operation. RAM of the range of 4 GB would further increase the performance. The whole system should be Linux based.
Chapter 4

System Design

System design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. System design could see its a the application of systems theory to product development. There is an overlap with the disciplines of system analysis, system architecture and system engineering.

4.1 Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). The commonly used notations in DFD are:

Figure 4.1: Symbols Used
4.1.1 Level 0 DFD

Figure 4.2: Level 0 DFD
4.1.2 Level 1 DFD

Figure 4.3: Level 1 DFD
4.1.3 Level 2 DFD

Figure 4.4: Level 2 DFD
4.2 UML Diagram

4.2.1 Use case Diagram

A use case diagram is a diagram that shows a set of use cases and actors and their relationships. It is used to model the dynamic aspect of a system. Hence a use case diagram commonly contain:

- Use cases
- Actors
- Dependency, generalization and association relationships

![Use Case Diagram](image-url)

Figure 4.5: Use case Diagram
4.2.2 Sequence Diagram

A sequence diagram emphasizes the time ordering of messages. The Objects that participate in the interaction are placed across the X axis. Typically the objects time initiate the interaction are placed at the left. The messages that these objects send and receive are placed along Y axis, in order of increasing from top to bottom. A sequence diagram has:

- **Object lifeline:** It is a vertical dashed line that represents the existence of an object over a period of time.
- **Focus of control:** It is a tall thin rectangle that shows the period of time during which an object is performing an action.

![Sequence Diagram](image.png)

**Figure 4.6: Sequence Diagram**
4.2.3 Activity Diagram

An activity diagram is essentially a flowchart that shows the flow from activity to activity. It usually contains:

- Activity states and action states
- Transitions
- Objects

Figure 4.7: Activity Diagram
Chapter 5

Detailed Design

The project has been divided into 4 modules. They are:

1. User Interface
2. Query engine
3. Rule generation
4. Rule application

A brief description of the modules are given below.

**User interface:**
This module is the primary interface with the user and integrates all the modules into a single runnable application. The user inputs data into this module.

**Query engine:**
This module is the interface between UI and HBase. The input data is exported to and imported from HBase using this module.

**Rule generation:**
This module is responsible for generating rules based on the training data.

**Rule application:**
This module is responsible for applying the generated rules to the actual data in which prediction has to be done.
5.1 User Interface

The user interface is the part of the application with which the user interacts. The project is designed in such a way that it is a standalone java application with hbase running in the background. All that the user sees is an application window which accepts the users input files and displays the predicted behaviour. The user interface is divided into the following components.

- Start page
- File browser
- Attribute selector
- Window for setting importance factor
- Output page

Start page

The start page essentially consists of two buttons

- Button for selecting the training data for the program
- Button for selecting the data on which the prediction has to be performed

The pressing of button 1 launches a file browser, which can then be used to select the text file which contains the training data. This training data is then used to generate rules using dmel algorithm.

Attribute selector

Once the file is selected, the program reads the contents of the file. The attributes are identified. A popup window is displayed which prompts the user to select the attributes that should be used for generating the rules. The attributes that are unchecked are discarded.

Window for setting importance factor

After selecting the attributes, the user is prompted with another window which displays a slider for each attribute selected in the previous step. The slider ranges from 0 to 1 in steps of 0.1. Once this step is complete, the selected attributes and their corresponding importance factors are then stored into attribute instances of the attribute class, each of which contains a variable named importance factor. Once this step is complete, the program returns to the starting page. Once the user is at the starting page again, he clicks button 2 to browse for the actual database file on which the prediction is to be done. For this the file browser is used again.

File browser

The file browser is invoked whenever the user has to select a file. The file browser is a file explorer window that prompts the user to browse to a file and return its absolute path to the program.
Output page

Once the user enters the database file on which prediction is to be done, the prediction algorithm is invoked. A set of rules are generated on the basis of the training data that has been previously selected and these rules are then applied on the database on which prediction is to be done. The target attribute value is then updated for every row in the database. The output page then provides options for displaying the set of rules as it is generated, and also for displaying the entire database, and also for displaying only the target attribute field.

5.2 Query Engine

The Query engine is developed as a java platform, which helps in connecting analytic engine with the consumer database. The Consumer Database is stored in Apache HBase. The efficiency of the whole system revolves around the efficiency of query engine. The query engine is used wherever the program has to write some data to HBase or read some data from HBase. Following are the instances where the query engine has a role.

- Read the training data file and store it into HBase in a suitable format as soon as the user browses into the file using the file explorer.
- Retrieve the training data from HBase and use it for generating the rules.
- Read the database file on which prediction is to be done and store it into HBase in a suitable format as soon as the user browses into the file using the file explorer.
- Retrieve the database from HBase and use it for applying the rules.
- For each entry in the database, store the value of the target attribute in HBase

The following are the instruction to setup Apache HBase, an inevitable component of query engine.

Make sure JAVA JDK 1.7.x is installed in all the nodes.

    extract the zookeeper tar.gz file
    tar -zxvf zookeeper-3.3.6.tar.gz

Create a file with any name zoo.cfg in the conf folder and give the following data in the file you have created (change the ip accordingly)

    sudo nano conf/zoo.cfg

    tickTime=2000
dataDir=/var/zookeeper
clientPort=2181
initLimit=5
syncLimit=2
server.1=localhost:2888:3888

It is required that a file named myid be created in the path specified in dataDir which contains just one entry which is of the server id of the specific system.
Here server id is 1

    sudo mkdir /var/zookeeper
    sudo nano /var/zookeeper/myid

Update the /etc/hosts file on each machine to add the host names being used in the zookeeper configuration

    sudo nano /etc/hosts

eg: 192.168.0.54 zoo1
      192.168.0.55 zoo2
      192.168.0.56 zoo3

This completes the configuration part, next cd to the zookeeper home and start the cluster by running

    sudo bin/zkServer.sh start

Check your cluster by

    sudo bin/zkServer.sh status

**Installing Cloudera CDH4.3 Hadoop**

Download the CDH4-repository

    sudo dpkg -i < Path to CDH4-repository-file >
    sudo apt-get update
    sudo apt-get install hadoop-0.20-mapreduce-jobtracker
    sudo apt-get install hadoop-hdfs-namenode
    sudo apt-get install hadoop-hdfs-secondarynamenode
    sudo apt-get install hadoop-0.20-mapreduce-tasktracker hadoop-hdfs-datanode
    sudo apt-get install hadoop-client
    sudo apt-get update

    sudo gedit /etc/hadoop/conf/core-site.xml

===> Add this

    <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:8020</value>
    </property>

    sudo gedit /etc/hadoop/conf/hdfs-site.xml

===> Add this

    #==============================================================================

    <property>
    <name>dfs.permissions.superusergroup</name>
    <value>hadoop</value>

21
<property>
  <name>dfs.namenode.name.dir</name>
  <value>/data/1/dfs/nn,/nfsmount/dfs/nn</value>
</property>

<property>
  <name>dfs.datanode.data.dir</name>
  <value>/data/1/dfs/dn,/data/2/dfs/dn,/data/3/dfs/dn</value>
</property>

#========================================
sudo mkdir -p /data/1/dfs/nn /nfsmount/dfs/nn
sudo mkdir -p /data/1/dfs/dn /data/2/dfs/dn /data/3/dfs/dn /data/4/dfs/dn
sudo chown -R hdfs:hdfs /data/1/dfs/nn /nfsmount/dfs/nn /data/1/dfs/dn /data/2/dfs/dn /data/3/dfs/dn /data/4/dfs/dn
sudo chmod 700 /data/1/dfs/nn /nfsmount/dfs/nn
sudo chmod go-rx /data/1/dfs/nn /nfsmount/dfs/nn
sudo -u hdfs hadoop namenode -format

sudo gedit /etc/hadoop/conf/mapred-site.xml

#========================================

<property>
  <name>mapred.job.tracker</name>
  <value>localhost:8021</value>
</property>

<property>
  <name>mapred.local.dir</name>
  <value>/data/1/mapred/local,/data/2/mapred/local,/data/3/mapred/local</value>
</property>

#========================================
sudo mkdir -p /data/1/mapred/local /data/2/mapred/local /data/3/mapred/local /data/4/mapred/local
sudo chown -R mapred:hadoop /data/1/mapred/local /data/2/mapred/local /data/3/mapred/local /data/4/mapred/local
sudo -u hdfs hadoop fs -mkdir /tmp
sudo -u hdfs hadoop fs -chmod -R 1777 /tmp
sudo -u hdfs hadoop fs -mkdir -p /var/lib/hadoop-hdfs/cache/mapred/mapred/staging
sudo -u hdfs hadoop fs -chmod 1777 /var/lib/hadoop-hdfs/cache/mapred/mapred/staging
sudo -u hdfs hadoop fs -chown -R mapred /var/lib/hadoop-hdfs/cache/mapred
sudo -u hdfs hadoop fs -ls -R /
sudo -u hdfs hadoop fs -mkdir /tmp/mapred/system
sudo -u hdfs hadoop fs -chown mapred:hadoop /tmp/mapred/system
sudo service hadoop-hdfs-namenode start
sudo service hadoop-hdfs-datanode start
sudo service hadoop-0.20-mapreduce-tasktracker start
sudo service hadoop-0.20-mapreduce-jobtracker start

**Installing HBase**

sudo apt-get install hbase
dpkg -L hbase

sudo gedit /etc/security/limits.conf
#=================================
hdfs - nofile 32768
hbase - nofile 32768
#=================================

To apply the changes

sudo gedit /etc/pam.d/common-session
#=================================
session required pam_limits.so
#=================================

sudo gedit /etc/hadoop/conf/hdfs-site.xml
#=================================
<property>
  <name>dfs.datanode.max.xcievers</name>
  <value>4096</value>
</property>
#=================================

sudo apt-get install hbase-master
sudo service hbase-master start
In your browser: http://localhost:60010

    sudo apt-get install hbase-rest
    sudo gedit /etc/hbase/conf/hbase-site.xml

    #===================
    <property>
    <name>hbase.rest.port</name>
    <value>60050</value>
    </property>
    #===================

    sudo -u hdfs hadoop fs -mkdir /hbase
    sudo -u hdfs hadoop fs -chown hbase /hbase
    sudo apt-get install hbase-regionserver
    sudo gedit /etc/hosts

    Change ip to 127.0.0.1 in both the instances
    sudo service hbase-master start
    sudo service hbase-regionserver start
    sudo service hbase-rest start
    sudo -u hdfs hbase shell

    Just check if you are able to create a hbase table:
    create 'cr','fam'

Successfully installed Hadoop cdh4.3 from cloudera, Apache HBase, Zookeeper.

The query engine retrieves data from hbase and delivers it to the analytics engine.
5.3 Rule Generation

Rule Generation module is where the rules for predicting customer churn get generated. A data mining algorithm, called DMEL, is used to mine rules in databases. DMEL is a kind of Genetic Algorithm which searches through huge rule spaces effectively using an evolutionary approach.

Algorithm starts with a set of solutions (represented by chromosomes) called population. The initial population is generated using probabilistic induction technique. Solutions from one population are taken and used to form a new population. This is motivated by a hope, that the new population will be better than the old one. Solutions which are selected to form new solutions (offspring) are selected according to their fitness - the more suitable they are the more chances they have to reproduce. DMEL evaluates the fitness of a chromosome using a function defined in terms of the probability that the attribute values of a record can be correctly determined using the rules it encodes. This is repeated until some condition (for example number of populations or improvement of the best solution) is satisfied.

Hence results in generation of best population which consist of the rules to predict the customer churn accurately. These rules are stored in file for future use in the Rule Application Module.

Outline of the basic Genetic Algorithm:

1. [Start] Generate random population of n chromosomes (suitable solutions for the problem)
2. [Fitness] Evaluate the fitness f(x) of each chromosome x in the population
3. [Newpopulation] Create a new population by repeating following steps until the new population is complete
   (a) Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
   (b) [Crossover] With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.
   (c) [Mutation] With a mutation probability mutate new offspring at each locus (position in chromosome).
   (d) [Accepting] Place new offspring in a new population
4. [Replace] Use new generated population for a further run of algorithm
5. [Test] If the end condition is satisfied, stop, and return the best solution in current population
6. [Loop] Go to step 2
5.3.1 DMEL Algorithm

Algorithm : DMEL

\( R_1 \leftarrow \{1\text{-st order rules obtained by probabilistic induction}\}; \)
\( l \leftarrow 2; \)
\( \textbf{while } R_{l-1} \neq \emptyset \textbf{ do} \)
\( \textbf{begin} \)
\( \quad t \leftarrow 0; \)
\( \quad \text{population}[t] \leftarrow \text{initialize}(R_{l-1}); \)
\( \quad \text{fitness}(\text{population}[t]); \)
\( \quad \textbf{while } \text{not } \text{terminate} (\text{population}[t]) \textbf{ do} \)
\( \quad \textbf{begin} \)
\( \quad \quad t \leftarrow t + 1; \)
\( \quad \quad \text{population}[t] \leftarrow \text{reproduce}(\text{population}[t - 1]); \)
\( \quad \quad \text{fitness}(\text{population}[t]); \)
\( \quad \textbf{end} \)
\( \quad R_l \leftarrow \text{decode} (\text{the fittest individual in } \text{population}[t]); \)
\( \quad l \leftarrow l + 1; \)
\( \textbf{end} \)
\( \text{Rules} \leftarrow \bigcup_{i} R_i; \)
Algorithm: INITIALIZE

population initialize($R_{l-1}$)
begin
    $\mathcal{R} \leftarrow \{ \text{all conjuncts in the antecedent of all } r \in R_{l-1} \}$;
    $i \leftarrow 1$;
    while $i \leq \text{popsize}$ do
        begin
            $j \leftarrow 1$;
            while $j \leq \text{nalleles}$ do
                begin
                    $\text{chrom}_i.\text{allele}_j \leftarrow \text{rand}(\mathcal{R})$;
                    $j \leftarrow j + 1$;
                end
            end
            $i \leftarrow i + 1$;
        end
    return $\bigcup_{i} \text{chrom}_i$;
end
Algorithm : REPRODUCE

\textbf{population} reproduce(population[t - 1])
\begin{algorithmic}
\State $chrom_1 \leftarrow \text{select}(\text{population}[t - 1])$;
\State $chrom_2 \leftarrow \text{select}(\text{population}[t - 1])$;
\State $nchrom_1, nchrom_2 \leftarrow \text{crossover}(chrom_1, chrom_2)$;
\State $\text{mutation}(nchrom_1)$;
\State $\text{mutation}(nchrom_2)$;
\State $\text{population} \leftarrow \text{steady-state}(\text{population}[t - 1], nchrom_1, nchrom_2)$;
\State $\text{return } \text{population}$;
\end{algorithmic}
\end{algorithm}

Algorithm : MUTATION

\textit{mutation}(nchrom)
\begin{algorithmic}
\begin{algorithmic}
\State $\mathcal{R} \leftarrow \{\text{all conjuncts in the antecedent of all } r \in R_{t-1}\}$;
\State $j \leftarrow 1$;
\While{$j \leq nalleles$}
\State $\text{while } j \leq nalleles$ do
\State $\text{begin}$
\State $\text{if } \text{random} < p_{mutation}$ then
\State $\text{begin}$
\State $k = \text{random}(1, l)$;
\State $\text{nchrom.allele}_j.\text{rule}_k \leftarrow \text{hill-climb}(\mathcal{R})$;
\State $\text{end}$
\State $j \leftarrow j + 1$;
\State $\text{end}$
\State $\text{end}$
\State $\text{end}$
\State $\text{end}$
\end{algorithmic}
\end{algorithmic}
5.4 Rule Application

Rule Application Module applies the rule generated in the Rule Generation Module on the test data uploaded by the user. Purpose is to distinguish churners from non-churners as much as possible. This module predicts to which class each customer belongs, a churner or non-churner.

Each of the rules generated in the previous module is applied on each record of the test dataset. A weight is calculated during the application of each rule. Depending on this calculated weight, the appropriate target value is assigned to each record, i.e. "churn" or "not churn". The result is stored in a separate table. Finally, the customer churn list is passed to the User Interface module.
Chapter 6

Developer Tools and External Libraries Used

6.1 HBase

Since a random, realtime read/write access to the Data is needed, hence Apache HBase is used which is an open source, non-relational, distributed database. Analysis can be done on one day’s data or on one week’s or on maximum of one month’s customer data. Data is flushed from the HBase periodically, so data older than one month is not available for analysis. An HBase system comprises a set of tables. Each table contains rows and columns, much like a traditional database. Each table has an element defined as a Primary Key, and all access attempts to HBase tables by using this Primary Key. An HBase column represents an attribute of an object. HBase allows for many attributes to be grouped together into what are known as column families, such that the elements of a column family are all stored together. This is different from a row-oriented relational database, where all the columns of a given row are stored together. The table schema of HBase is predefined and the column families are specified. However, its very flexible in that new columns can be added to families at any time, making the schema flexible and therefore able to adapt to changing application requirements.

The important features of HBase are:

- Linear and modular scalability.
- Strictly consistent reads and writes.
- Automatic and configurable sharding of tables
- Automatic failover support between RegionServers.
- Convenient base classes for backing Hadoop MapReduce jobs with Apache HBase tables.
- Easy to use Java API for client access.
- Block cache and Bloom Filters for real-time queries.
- Query predicate push down via server side Filters
- Thrift gateway and a REST-ful Web service that supports XML, Protobuf, and binary data encoding options
- Extensible jruby-based (JIRB) shell
- Support for exporting metrics via the Hadoop metrics subsystem to files or Ganglia; or via JMX
6.2 Eclipse

Eclipse is an integrated development environment (IDE). It contains a base workspace and an extensible plug-in system for customizing the environment. Written mostly in Java, Eclipse can be used to develop applications. The Eclipse software development kit (SDK), which includes the Java development tools, is meant for Java developers. Released under the terms of the Eclipse Public License, Eclipse SDK is free and open source software.

6.3 Ubuntu

Ubuntu is a Linux flavour built on top of Debian system. So in the proposed system, Ubuntu can be replaced by any Debian based system. Ubuntu is composed of many software packages, the majority of which are free software.

6.4 Java

Java is a computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java applications are typically compiled to bytecode (class file) that can run on any Java virtual machine (JVM) regardless of computer architecture. Java is, as of 2014, one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems (which has since merged into Oracle Corporation) and released in 1995 as a core component of Sun Microsystems’ Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

The features of Java Programming Language are:

Simple

- Looks familiar to existing programmers: related to C and C++:
  - Omits many rarely used, poorly understood, confusing features of C++, like operator overloading, multiple inheritance, automatic coercions, etc.
  - Contains no goto statement, but break and continue
  - Has no header files and eliminated C preprocessor
  - Eliminates much redundancy (e.g. no structs, unions, or functions)
  - has no pointers

Object-Oriented

Java is an object-oriented language, which means that you focus on the data in your application and methods that manipulate that data, rather than thinking strictly in terms of procedures. In an object-oriented system, a class is a collection of data and methods that operate on that data. Taken together, the data and methods describe the state and behavior of an object. Classes are arranged in a hierarchy, so that a subclass can inherit behavior from its superclass.
Java comes with an extensive set of classes, arranged in packages, that you can use in your programs.

**Distributed**
- It has a spring-like transparent RPC system
- Now uses mostly tcp-ip based protocols like ftp and http

**Interpreted**
The Java compiler generates byte-codes, rather than native machine code. To actually run a Java program, you use the Java interpreter to execute the compiled byte-codes. Java byte-codes provide an architecture-neutral object file format. The code is designed to transport programs efficiently to multiple platforms.
- rapid turn-around development
- Software author is protected, since binary byte streams are downloaded and not the source code

**Robust**
Java has been designed for writing highly reliable or robust software:
- language restrictions (e.g. no pointer arithmetic and real arrays) to make it impossible for applications to smash memory (e.g overwriting memory and corrupting data)
- Java does automatic garbage collection, which prevents memory leaks
- extensive compile-time checking so bugs can be found early; this is repeated at runtime for flexibility and to check consistency

**Secure**
Security is an important concern, since Java is meant to be used in networked environments. Without some assurance of security, you certainly wouldn’t want to download an applet from a random site on the net and let it run on your computer. Java’s memory allocation model is one of its main defenses against malicious code (e.g can’t cast integers to pointers, so can’t forge access). Furthermore:
- access restrictions are enforced (public, private)
- byte codes are verified, which copes with the threat of a hostile compiler

**Architecture-Neutral**
- compiler generates bytecodes, which have nothing to do with a particular computer architecture
- easy to interpret on any machine
**Portable**

Java goes further than just being architecture-neutral:

- no "implementation dependent" notes in the spec (arithmetic and evaluation order)
- standard libraries hide system differences
- the Java environment itself is also portable: the portability boundary is POSIX compliant

**High-Performance**

Java is an interpreted language, so it will never be as fast as a compiled language as C or C++. In fact, it is about 20 times as slow as C. However, this speed is more than enough to run interactive, GUI and network-based applications, where the application is often idle, waiting for the user to do something, or waiting for data from the network.

**Multithreaded**

Java allows multiple concurrent threads of execution to be active at once. This means that you could be listening to an audio clip while scrolling the page and in the background downloading an image. Java contains sophisticated synchronization primitives (monitors and condition variables), that are integrated into the language to make them easy to use and robust. The java.lang package provides a Thread class that supports methods to start, run, and stop a thread, and check on its status.

**Dynamic**

Java was designed to adapt to an evolving environment:

- Even after binaries have been released, they can adapt to a changing environment
- Java loads in classes as they are needed, even from across the network
- It defers many decisions (like object layout) to runtime, which solves many of the version problems that C++ has

### 6.5 Package javax.swing

Swing is the primary Java GUI widget toolkit. It is part of Oracle’s Java Foundation Classes (JFC) an API for providing a graphical user interface (GUI) for Java programs. Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT). Swing provides a native look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists. Unlike AWT components, Swing components are not implemented by platform-specific code. Instead they are written entirely in Java and therefore are platform-independent. The term "lightweight" is used to describe such an element.
Chapter 7

Testing

7.1 Introduction

Software testing is critical element of software quality assurance and represents the ultimate review of specification design and coding. Software testing is one of the broader topics and often referred to as verification to all the activities that endure the software built is traceable to use requirements. Software testing can be taken as one of the many processes in software development. It provides the opportunity to correct the errors in the developed system. Testing is the process of executing a program with the intend of finding error.

7.2 Goals and Objectives

The objective is to decide the test that systematically uncover the different classes of errors and to do so with a minimum amount of time and effort. A good test is one that has high probability of finding the yet undiscovered errors. The product has been implemented by the software developer itself. Testing has been carried out according to the test plan and test procedures stated in this test specification. This document gives a general description of the test specification of the system.

7.3 Scope

The test case is a document that describes an input an action or event and an expected response to determine if a feature of an application is working correctly. So the project entitled MEC Archive also requires various test cases like login test, upload test, mentor allocation test, ranking test, search test etc. Testing begins at the middle level and works toward the integration of web based system. Testing and debugging are different activities. But, any testing strategies include debugging strategies. in the test phase the testing of the developed system is done along with the system data.

7.4 Test Plan

It describes the overall testing strategy and the project management issues that are required to properly execute effective tests.
7.5 Test Strategy

The overall strategy for software testing is described.

7.5.1 Unit Testing

In unit testing different modules are tested against the specifications produced during the design of the module. Unit testing is essential for the verifications of the code produced during the coding phase and hence the goal is to test the internal logic of the modules. The testing is carried out during the programming itself. After designing and coding each form, they are run to see whether there are any anomalies. Some of the various test cases used to test the system are as follows:

- Test cases are given for testing against requirements of the unit being tested.
- Test case for the path or branch covering.
- Test case with classes of bad data.

7.5.2 Integration Testing

Integration test exposes defects in the interface and the interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of architectural design are integrated and tested until the software works as a system. Here each module such as searching mentor allocation, file upload, ranking and management is combined together to check whether any error occurs or not.

7.5.3 System Testing

System Testing test a completely integrated system to verify that it meets its requirement. In this test method we will combine several types of testing. We will test for several different conditions by following several different test methods. It is for checking whether a system is working in a perfect way or not and also check it meet all requirements of the user or not.

7.5.4 Validation Testing

At the culmination of the integration testing, the software was completely assembled as a package, interfacing errors have been uncovered and final series of software validation testing began. Here we test the system function in a manner that can be reasonably expected by customer, the system was tested against system requirement specification. Different unusual inputs that the user may use where assumed and the outputs were verified for such unprecedented inputs. This test is performed to validate the software. In this the entire software will be created and will test all the components of the software together. Here we are checking the required file validator and regular expression validator that is, In required field validator check whether all the fields are filled or not and in regular expression what we do is check all the fields are filled with a correct value or not.
7.6 Test Cases

7.6.1 Sample Input Text

The input is given in the form of a text file containing customer information (training data set).

<table>
<thead>
<tr>
<th>District</th>
<th>House_Type</th>
<th>Income</th>
<th>International_Plan</th>
<th>Churn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suburban</td>
<td>Detached</td>
<td>High</td>
<td>No</td>
<td>NO</td>
</tr>
<tr>
<td>Suburban</td>
<td>Detached</td>
<td>High</td>
<td>Yes</td>
<td>NO</td>
</tr>
<tr>
<td>Rural</td>
<td>Detached</td>
<td>High</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Urban</td>
<td>Semi-detached</td>
<td>High</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Urban</td>
<td>Semi-detached</td>
<td>Low</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Urban</td>
<td>Semi-detached</td>
<td>Low</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Rural</td>
<td>Semi-detached</td>
<td>Low</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Suburban</td>
<td>Terrace</td>
<td>High</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Suburban</td>
<td>Semi-detached</td>
<td>Low</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Urban</td>
<td>Terrace</td>
<td>Low</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Suburban</td>
<td>Terrace</td>
<td>Low</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Rural</td>
<td>Terrace</td>
<td>High</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Rural</td>
<td>Detached</td>
<td>Low</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Urban</td>
<td>Terrace</td>
<td>High</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Figure 7.1: Training Data set
7.6.2 Intermediate Output

A sample rule set is generated using the training data set and is stored in a text file.

<table>
<thead>
<tr>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>International Plan = No AND Income = Low AND District = Rural→Yes</td>
</tr>
<tr>
<td>International Plan = No AND Income = Low→Yes</td>
</tr>
<tr>
<td>Income = Low AND District = Suburban→Yes</td>
</tr>
<tr>
<td>District = Rural AND House Type = Semi-detached→Yes</td>
</tr>
<tr>
<td>International Plan = No→Yes</td>
</tr>
<tr>
<td>International Plan = Yes→No</td>
</tr>
<tr>
<td>Income = High→No</td>
</tr>
<tr>
<td>Income = Low→Yes</td>
</tr>
<tr>
<td>House Type = Detached→No</td>
</tr>
<tr>
<td>House Type = Terrace→No</td>
</tr>
<tr>
<td>House Type = Semi-detached→Yes</td>
</tr>
<tr>
<td>District = Rural→Yes</td>
</tr>
<tr>
<td>District = Suburban→No</td>
</tr>
<tr>
<td>District = Urban→No</td>
</tr>
</tbody>
</table>

Figure 7.2: Rule set
### 7.6.3 Output

Final output is a list of churned mobile users.

<table>
<thead>
<tr>
<th>Suburban</th>
<th>Detached</th>
<th>High</th>
<th>No→ No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suburban</td>
<td>Detached</td>
<td>High</td>
<td>Yes→ No</td>
</tr>
<tr>
<td>Rural</td>
<td>Detached</td>
<td>High</td>
<td>No→ No</td>
</tr>
<tr>
<td>Urban</td>
<td>Semi-detached</td>
<td>High</td>
<td>No→ No</td>
</tr>
<tr>
<td>Urban</td>
<td>Semi-detached</td>
<td>Low</td>
<td>No→ Yes</td>
</tr>
<tr>
<td>Urban</td>
<td>Semi-detached</td>
<td>Low</td>
<td>Yes→ Yes</td>
</tr>
<tr>
<td>Rural</td>
<td>Semi-detached</td>
<td>Low</td>
<td>Yes→ Yes</td>
</tr>
<tr>
<td>Suburban</td>
<td>Terrace</td>
<td>High</td>
<td>No→ No</td>
</tr>
<tr>
<td>Suburban</td>
<td>Semi-detached</td>
<td>Low</td>
<td>No→ Yes</td>
</tr>
<tr>
<td>Urban</td>
<td>Terrace</td>
<td>Low</td>
<td>No→ Yes</td>
</tr>
<tr>
<td>Suburban</td>
<td>Terrace</td>
<td>Low</td>
<td>Yes→ No</td>
</tr>
<tr>
<td>Rural</td>
<td>Terrace</td>
<td>High</td>
<td>Yes→ No</td>
</tr>
<tr>
<td>Rural</td>
<td>Detached</td>
<td>Low</td>
<td>No→ Yes</td>
</tr>
<tr>
<td>Urban</td>
<td>Terrace</td>
<td>High</td>
<td>Yes→ No</td>
</tr>
</tbody>
</table>

Figure 7.3: Final Output
Chapter 8

Screenshots

Figure 8.1: Start Page
Figure 8.2: Select Training data file
Figure 8.3: Select Test data file
Figure 8.4: Attribute Selection
Figure 8.5: Set importance factor
Figure 8.6: Output
Figure 8.7: Output
Chapter 9

Conclusion

Developed a system for churn prediction in telecom industry. It helps to distinguish churners from non-churners as much as possible using DMEL algorithm. The churn prediction model attempts to predict to which class each customer belongs. A carrier can then choose to provide special personalized offer and services to those subscribers who are predicted with higher likelihood to churn. The use of HBase increased the speed of data retrieval. The experimental results on the subscriber database also showed that DMEL is robust in a way that it is able to discover rules hidden in the database and to predict the churns of subscribers under different churn rates. Since the churn rates of different subscribers are different and the churn rate of a specific carrier varies from time to time, robustness is necessary to an effective churn predictor.
Chapter 10

Future Scope

- The system can be further extended for providing special personalized offers and services to those subscribers who are predicted with higher likelihood to churn.

- Another future scope can be a module intended for the future analysis of Customer Data. Here the data would be stored in HDFS and analysis of data can be done over a large period of time say two months, a year or more. This system would use Google Protocol Buffer to develop a language independent object/message model. Further before the permanent storage of data, it can be compressed using LZO compression algorithm. The data then would be permanently stored in HDFS. The data from the local file system would be moved to HDFS using a shell script.
Bibliography


[8] Vladislav Lazarov and Marius Capota : Churn Prediction