The Architecture of a Churn Prediction System Based on Stream Mining

Borja Balle[a], Bernardino Casas[a], Alex Catarineu[a], Ricard Gavaldà[a], David Manzano-Macho[b]

[a] *Universitat Politècnica de Catalunya - BarcelonaTech. e-mail: {balle,bcasas,catarineu,gavalda}@lsi.upc.edu*

[b] *Ericsson Spain. e-mail: david.manzano.macho@ericsson.com*

Abstract. *Churning* is the movement of customers from a company to another. For any company, being able to predict with some time which of their customers will churn is essential to take actions in order to retain them, and for this reason most sectors invest substantial effort in techniques for (semi)automatically predicting churning, and data mining and machine learning are among the techniques successfully used to this effect. In this paper we describe a prototype for churn prediction using *stream mining* methods, which offer the additional promise of detecting new patterns of churn in real-time streams of high-speed data, and adapting quickly to a changing reality. The prototype is implemented on top of the MOA (Massive Online Analysis) framework for stream mining. The application implicit in the prototype is the telecommunication operator (mobile phone) sector.

A shorter version of this paper, omitting Section 5, was presented at CCIA'13 (`http://mon.uvic.cat/ccia2013/en/`).

Keywords. Data stream mining, churn prediction, Hoeffding trees, machine learning, MOA

1. Introduction

Customer churning is the movement of customers from a company to another. For any company, being able to predict which of their customers will churn before they actually leave is essential to take actions in order to retain them. For this reason most sectors invest substantial effort in techniques for (semi)automatically predicting churning as accurately and as early as possible.

Traditional data mining / machine learning techniques are often applied for building predictive models that assign each customer a probability that s/he churns in one or more prespecified periods of time. However, these techniques have a number of limitations. First, they tend to be batch-based: all the data to be used for the modeling must be available upfront for the model construction; many of them cannot work incrementally, i.e., incorporate into the model information arrived after it has been built; the model construction is computationally costly, and often needs human intervention for e.g. parameter optimization, data selection, or model evaluation; and, most importantly, methods assume that the source of data is stationary, that is, does not change in nature over time.

The latter assumption is patently false in many scenarios: customers change their behavior over time in reaction to market conditions, product or price changes, or sociological phenomena. Sometimes change is gradual, but abrupt, overnight change can also occur. The problem is particularly acute in sectors such as the telecommunications market (say, companies to which customers subscribe for mobile phone service), which are affected by high customer churn rate.

In this setting, models built sometimes in the past can quickly or gradually get out of sync with the current customer patterns, resulting in suboptimal churn prediction rate and therefore revenue loss. An obvious solution is to rebuild the models periodically, but this typically either requires human analyst time (which is slow and expensive) or automatizing decisions such as when and on which amount data to retrain new models, which may be hard to do in abstract.

Data stream mining techniques have emerged in the last decade or so to transport the benefits of data mining to such scenarios: they intend to provide e.g. model induction techniques that work on streams of data that may potentially never end, that can process thousands or millions of instances per second both for training and for prediction, and that can maintain these models updated as the statistical or logical nature of the data change over time.

In this paper we describe the architecture and a proof-of-concept implementation of a system for churn prediction based on stream mining, with mobile phone subscriber churning as the background scenario. The use of *stream mining* methods offer the promise of detecting new patterns of churn in real-time streams of high-speed data, and adapting quickly to changing realities.

The core stream mining platform used is MOA (Massive Online Analysis) [2], and in particular we use its implementation of a variant of decision trees (Hoeffding trees) that can be built and maintained efficiently and incrementally from streams. Unfortunately, as usual in this field for e.g. privacy reasons, we could not use real data for our experiments. We developed a synthetic data generator based on descriptions of real data available in the literature, which may be of interest in itself.

The main conclusion from our proof-of-concept is that the technology exists to perform real-time, high-throughput prediction on streams of items containing information similar to that contained in churn prediction literature. On further work we will concentrate on several issues concerning scalability to international scale data volumes.

The paper is structured as follows: Section 2 discusses in detail the problem of churning, particularly in the telecommunications market and related previous work. Section 3 describes the requirements for a churn prediction system in a high-speed, highly volatile scenario such as mobile phone customer behavior analysis. Section 4 describes the architecture of the proposed solution and its main elements, as well as some details of the implementation. Section 5 presents an example of the workflow of events processed by the system. Section 6 describes the generator of synthetic data used to test the system. Section 7 describes qualitatively the results of the prototype. Finally, Section 8 summarizes the lessons learned during the implementation and testing, the main challenges to be solved for a large-scale, fully scalable solution, and other future work.

2. Preliminaries

2.1. The Problem of Churning in the telecommunication market

Like in every business sector, operators are always looking to get a better understanding about their customers' preferences and their satisfaction to offer them better products and services. Despite of the maturity of this market and due to subscriber growth down and revenues flat, churn management has become into one of the most pressing problems faced by operators. There are many reasons that may affect a subscriber on deciding to churn. Some of them may be:

- In contrast to post-paid customers, prepaid customers are not bound by a contract. The central problem concerning prepaid customers is that the actual churn date in most cases is difficult to assess.
- Customer loyalty is directly related to the customer service and service experience. Lack of connection capabilities or quality in places where the customer requires service can cause customers to abandon their current service provider in favor of others with broader reach or a more robust network. Besides, a slow response to customer complaints or billing errors are sure paths to a customer relations disaster.
- Finally other factors such as the pricing, lack of features (customers may switch carriers for features not provided by their current providers) or coverage, new technology introduced by competitors (for example, high-speed data) or the fact that new competitors enter the market, are also reason that has their influence on the churn rate.

Churn prediction modeling techniques attempt to understand the precise customer behaviors and attributes which signal the risk of customer churn. The accuracy of the technique used is obviously critical to the success of any proactive retention efforts. In order to have successful loyalty programs operators need to analyze their customers based on several parameters such as spending and usage of each service and product as well as their behavior and traffic patterns. All these decisions should be made upon the analysis of the data related to customer. Time is critical, which means that online approaches are gaining more momentum to support operational decisions in almost real time. By means of online analysis, an operator can launch and execute quickly some actions with aim of retaining some customers, and measure the impact of the campaign in real time which allows adapting them instantly as required.

2.2. Previous Work

There is substantial published literature on churn prediction techniques, although probably most of the experiences have remained unpublished in the company. We mention only a few representative ones of those using data mining, machine learning, or, more in general, that build models from information gathered in the past; see [8] for a good survey. References [1,11,5,10] use methods such as decision trees, support vector machines, genetic algorithms, and multilayer neural networks for churn prediction in various contexts, from finantial-service providers to landline phones and mobile phones; [6] carries out an intensive and state-of-the art comparison of this and other methods. [9] discusses the problem of evaluating predictors in this context. [12] discusses the use of social network information (and customer social circle) to help in churn prediction.

As mentioned in the introduction, all these works and those that we are aware of use the batch paradigm: a model is built offline by the data analyst and then used online to make predictions. Some of the works do mention the possibility of retraining the model periodically. By contrast, we aim at techniques that allow building the models online too and keeping them accurate as the customers change their behavior.

We omit the many other papers discuss the features actually used in practice and the "feature engineering" process, as well as the business implications of churn detection and possible customer retention strategies once they have been identified as possible churners.

For background on stream mining, see for example the books [3,4].

3. Requirements

The telecommunication market is highly and unpredictably dynamic. The ability for immediately detecting such a trend on even a small segment of subscribers may be essential to retaining many more leaving in the near future. However, the reasons why subscribers churn may change suddenly. For many (and hard to define a priori) reasons, subscribers that were not considering leaving their company may suddenly consider another company very compelling and decide to leave overnight. A scenario like this demands the application of online analytics, to detect and react in timely manner to the changes in the expected behavior of operators customers. Other requirements are:

- Accuracy. High recall (all churners are flagged as such) and relatively high precision (not many non-churners are flagged) are both important.
- Performance. Time is critical. A right action taken out of its proper window time is useless. Detect and react to any relevant change detected through the analysis of the incoming data flows is a competitive advantage.
- Flexibility. Data sources may change and any churning system has to have the ability to be adapted into the new scenario.
- Scalability. Churn prediction technology needs to handle difficult contexts, in which there are big data flows related to customers activity, with real-time requirements and prone to changes.
- Ability to segment customers and incorporate analyst's existing knowledge. Customer profiling demonstrates the direct business benefit gained from analytics. Traditional parameters include customer type, spending, subscription type and preferred services.

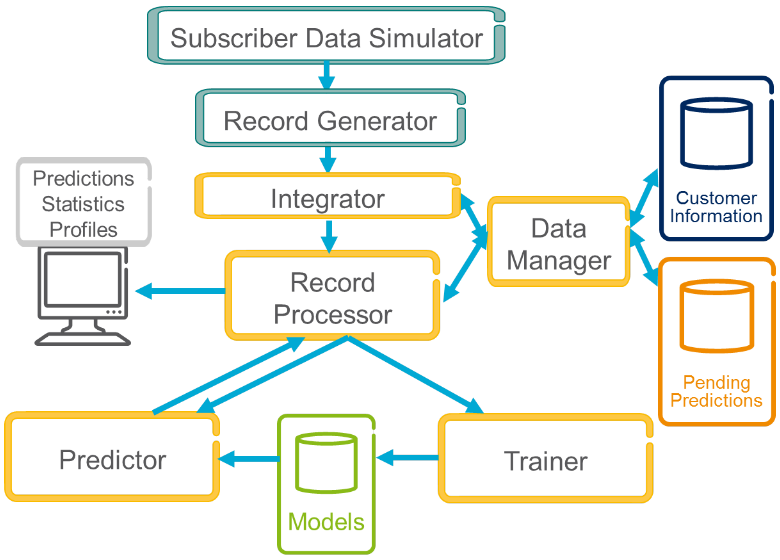

Figure 1. Architecture and Design of a Platform for Adaptive, Real-time Churn Prediction

4. Architecture

4.1. General Description

In this paper, we describe the architecture proposed for our prototype. It is depicted in Figure 1, and its main components are described next.

4.2. Integrator

The system processes as input a number of streams with diverse information. For example, a stream of call records, a stream with billing actions by the company and bill payments by the subscribers, contents from social networking services such as Twitter, etc. The Integrator module integrates all these streams, generating a logically unique stream of *events*. We distinguish several type of events, including at least a subscriber joining the company, calls and SMSes, complaints, bills emitted and bills paid, tweets by a subscriber, and "churn" events (e.g., a user explicitly has left the company, or for a prepaid user, it has been declared as a churner according to the company's criterion).

4.3. Data Manager

The Data Manager module manages the customer information database and the Pending Predictions queue.

The *customer information database* contains basic information about our subscribers (age, address, type of contract, etc.) as well as highly dynamic information (e.g. last numbers called, most frequently numbers called). The *Pending Predictions* queue contains all predictions that are awaiting for confirmation or refusal in the future (that is, whether the subscriber to which the prediction refers to has churned or not within a specified time frame).

These two structures, the customer information database and the Pending prediction queue, can easily become the two main bottlenecks in the system if not carefully implemented, both for time and for memory usage. If they do not fit in RAM, a write-optimized disk-resident database will be required.

4.4. Record Generator

The Record Generator receives the stream of Events generated by the integrator and uses it for two purposes. First, it updates the customer information database according to each Event. Second, it generates one or more Records from each Event using information from the customer information database. Thus, it creates a stream of Records passed downstream.

A Record is a vector of features, the first of which is a subscriber identifier, and the rest contains all the information about that subscriber state that is considered relevant for prediction. Many of them cannot be directly derived from the Event, but are aggregations of information about the customer precomputed in the database. One of the features (say, the last one) indicates if the subscriber has churned so far: it will be true when the Event originating the record was a churning one.

The prototype currently use this set of features for prediction, which figure among the most widely reported as useful in the literature:

- Age, sex, income range
- Contract type (mobile or landline, pre-paid or post-paid)
- Average call duration during last month
- Number of calls last week and last month
- Increase of decrease in number of calls in the last 2 weeks and the last 2 months
- % of calls by/to this subscriber where the caller/recipient belongs to another company
- # of complaints in the last 2 months, and % of these that were resolved satisfactorily
- Average bill value
- Increase or decrease of value last 2 bills

We made the following optimization for efficiency. Every event gives rise to at least one record, with the exception that every day only one call by or from a subscriber generates a record and a prediction. That is, if a subscriber gets or receives 20 calls in a day, all of them will be used to update his/her statistics in the customer database, but only the first one will generate a record and a prediction. This introduces a delay of (at most) one day in flagging this customer as churner, but reduces a lot of overhead.

4.5. Record Processor

The Record Processor is the heart of the system: it builds, maintains, and applies the predictive models. It therefore contains the data mining or machine learning algorithms that make prediction possible.

When a record not indicating churn arrives, it passes the record through the current model and makes a churn prediction for it. The record with its prediction is stored the Pending Predictions queue, waiting for future confirmation. When a record indicating churn arrives, records for that subscriber are searched in the Pending Predictions queue and, if found, passed to the model trainer as positive instances of churning. Expired records in the Pending Predictions queue (corresponding to subscribers that did not churn within a specified time) are passed to the model trainer as negative instances of churn. All records (describing current states of subscribers) are passed a clustering submodule to build subscriber profiles. We have used both 1) clustering methods available in MOA and 2) the split induced by the Hoeffding tree branches to define customer segments; they may give alternate segmentations of potential use for analysis.

Additionally, a background process periodically scans customers for which no Event has occurred and injects a special record indicating "no activity", so that 1) a prediction for the customer is generated from time to time (in case e.g. inactivity may indicate churning propension) and 2) the system is also trained to predict well on periods of customer inactivity.

4.6. Reporting and Interface to other Systems

The Record Processor module thus produces predictions, statistics and profiles of the predicted churners. The predicted churners id's and their current profiles are passed to the user interface or other parts of the customer management system so that adequate actions can be assessed and taken.

The subscriber profiles provide information for human analysts to build understandable portraits of churners and causes for their churning. Moreover it allows to focus the retention action efforts, such as calling with a promotion, to subsets of the subscribers with propensity to churning, even before they are flagged as churners by the system.

4.7. Implementation

We implemented a prototype of this architecture using the Java language. The prediction core of the system uses MOA (Massive Online Analysis), a platform related to the popular WEKA machine learning package, which includes a collection of machine learning algorithms (classification, regression, and clustering) and tools for algorithm evaluation, visualization, synthetic data generation, and stream management.

We used in particular MOA's Hoeffding Adaptive Tree classifier, which is a variation of the CVFDT method proposed in a seminal paper on stream mining by Hulten, Spencer, and Domingos [7]. A Hoeffding tree is an incremental, anytime decision tree induction algorithm that is capable of learning from massive data streams. The Adaptive version in [7] and implemented in MOA is able to adapt to changes of the data over time, updating and revising the structure and contents of the tree to keep it accurate.

5. Example Workflow

The following illustrates by a few examples the process workflow.

Suppose that Alice, currently a company customer, calls Bob, who is currently subscribed to another company. The (external) call management system passes the identifiers (e.g. numbers) of Alice and Bob together with the information deemed relevant (say, start time, duration, approximate location of Alice and Bob if available) to the Integrator module, which generates a Call Event and passes it to the Record generator. The Record generator queries its customer database, determines that Alice is a customer but Bob is not, so it 1) updates Alice's information in the database recording that she participated in this call and 2) generates a record with Alice as identifier, as described next. The record generated has Alice's identifier, and consists of a vector of "features" describing as accurately as possible the state of Alice at this point in time as recorded in the database. The record is passed to the Record processor. As it encodes a call, the Record processor will input it to the current prediction model, who will output a prediction about Alice's probability of churning in some period of reference such as a month; this is called the "prediction expiry date". The record, together with the prediction, is placed in the PendingPrediction queue, as it has to be verified in the future, and also passed to the Loyalty management module. The latter decides according to current rules if the customer merits some retention action, and which one.

The process is similar for most other types for events generated by Alice (SMS, bill payment, complaint, tweet, etc., but not those indicating customer churn): One record associated to Alice is generated, for which a prediction is generated. The pair (record,prediction) is stored in the PendingPrediction queue and used by the Loyalty Management module to possibly generate actions aimed at retaining Alice.

Concurrently with processing these incoming records the Record Processor monitors the PendingQueue for expired predictions. These records are labeled "noChurn", meaning that a customer did not churn within a prescribed period, and passed to the model updater, which will use to confirm, update, or revise the current model (in the usually called "training" model). For example, a pending prediction created on April 16th indicating that Alice may churn with probability 60% within one month, will be removed from the queue and processed as below by May 16th, because it is known at that time that Alice has indeed not churned in one month.

When customer Alice is determined to have churned (either by explicit action on her part or by e.g. lack of any activity in prepaid contracts), the system should receive a Churn event. The Record generator will generate a record with Alice's identifier and content "Churn" and the churning date and pass it to the Record processor. The Record processor retrieves any existing records associated to Alice in the PendingPrediction queue (which should be non-expired, as by the previous paragraph), and labels them with the "Churn" prediction, meaning that the right prediction at the time the record was produced should have been "Churn". This record is also passed to the model updater.

When either a prediction about Alice expires or a record indicating that Alice has Churned, the Record Processor notifies the Loyalty management module so that this information can be taken into account (by the module's internal rules) when actions have to be proposed to customers with a profile similar to Alice's.

As a hypothetical example of what "model update" means, suppose that at the time in which Alice's call arrives, one of the rules used by the system to predict churn is "customers who are female, aged 20-35, with two or more complaints about service in the last month, paying an average monthly bill > 200, and who have used roaming in the last 6 months have a probability of churning in the next month of 80%". The system additionally keeps track that it has recently seen 1000 such customers, of which 800 have indeed churned within one month (hence the 80% probability). If Alice fits into this profile, the system will predict that she will churn in the next month within 80%. After one month, if she has indeed churned, the system will update its statistics to 801/1001, so the rule will be updated to "...with 80.1% probability", and otherwise to 800/1001. More interestingly, the system may realize that for customers in this profile living in urban areas, the probability is in fact higher (say, 90%), and lower those living in rural areas is in fact lower (say, 70%). Then the rule above would be split into two rules, the first of which would be: "customers who are female, aged 20-35, with two or more complaints about service in the last month, paying an average monthly bill $>$ 200, who have used roaming in the last 6 months, and live in urban areas have a probability of churning in the next month of 90%". And the second would apply to customers living in rural areas and have a probability of 70%. This description, intended for comprehension, does not necessarily reflect any of the known stream mining techniques that could be used in an implementation of the invention.

6. The Synthetic Data Generator

In order to test the proposed system, we developed a simulator able to generate realistic synthetic data similar to what an operator gets from their customers. Basically, the simulation is a probabilistic dynamical system containing a predetermined number of customers. The system tries to reproduce the major types of interactions between customers and providers and their dynamics. For example, customers can place calls to other customers, either having the same provider or a different one. Customers are periodically billed for their calls, and big bills may make them angry. Angry customers sometimes complain to their provider and ask for cheaper rates, or maybe churn without further notice.

In particular, each user is modeled in the system using five parameters. Three of them are socio-economic factors which are known to providers: gender, age, and income. The other two are indices of how communicative (C) and impulsive (I) the user is; these are hidden parameters unknown to the provider. Furthermore, these indices evolve over time depending on the "mood" of the user, which is influenced by several factors described below. At start users are generated by sampling these parameters from a fixed probability distribution. In particular, gender, age and income follow a distribution trained real data. The other indices are sampled from a hand-crafted distribution, and in particular are not independent of the socio-economic factors. As a result, the simulation is populated with users that will behave differently from one another, and whose "profile" is loosely correlated with the features known a priori by the provider. This scheme is depicted in Figure 2.

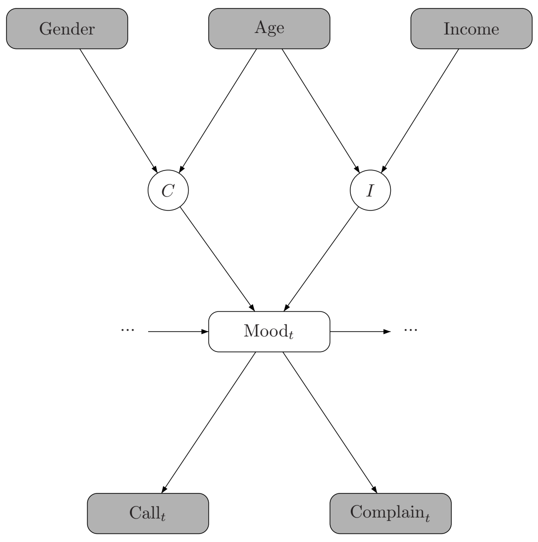

Figure 2. Factors affecting mood and behavior

The actions of each user are governed by a dynamic markovian model whose current state determines the user's "mood", which is in one of four states: {happy,neutral,angry,churn}. The dynamics of this model are as follows:

1. Time between state changes is larger for smaller values of I
2. The more time spent in "angry" state, the higher probability of churning
3. A high bill (w.r.t. the subscriber income) or an unresolved complaint makes you more angry (moves you one state towards "angry" or, if you are already in "angry", makes your churning probability higher), with a probability that depends on I
4. A complaint resolved ok makes you go back towards "happy", with a probability that depends on I

This internal mood state, which is unknown to the provider, affects the behavior of the user in multiple ways.

1. A user only complains if "Angry" (and the longer the time in "Angry", the more s/he complains.
2. A user only churns if "Angry" (and this becomes more likely the longer time s/he's been "Angry")
3. The longer time in "Angry", the less s/he calls.
4. The longer time in "Happy", the more s/he calls.
5. When s/he goes back to "neutral", the rate of calls per day goes back slowly towards the default value for the user.
6. Both duration and number of calls depend on the hidden parameter C.

This scheme is depicted in Figure 3.

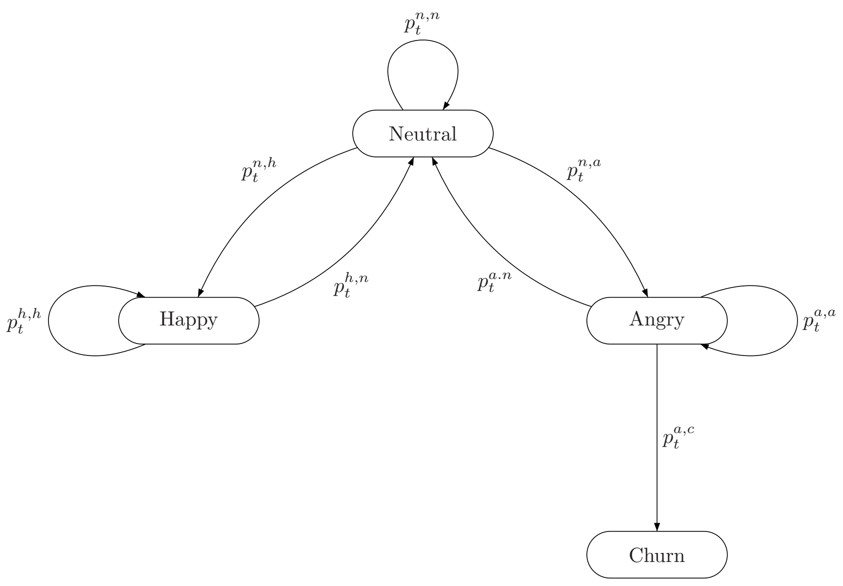

Figure 3. Factors affecting mood and behavior

7. Results and Scalability

We obtain good levels of recall and precision, roughly to the point that the randomness that we placed in the random generator allows. That is, we can predict which users will churn with an accuracy that is close to the probability with which (randomly) decide to churn or not given their internal states. In particular, if we happen to make the subscribers absolutely deterministic, we get results close to 100%. Of course, the absolute "goodness" of these figures does not mean much, other than how difficult or easy to predict we made our synthetic data. The point is that the system is able to correctly remember and put to use the information in the event stream for one particular purpose, that of churn prediction.

We also checked that Hoeffding trees are extremely good at adapting to changes. Via the prototype GUI we can vary during the execution parameters such as prices of our company and the competition, frequency of complaint calls and % of those resolved satisfactorily, average number of calls per subscriber, etc. which affect our subscribers' churn rate. We verified that after a change, prediction accuracy falls because the predictor gets out of sync, but after a few thousand calls, the new behaviors are captured by the tree and accuracy rises to almost optimal levels again.

On a commodity PC, the system processes about 10,000 records per second. Average memory consumption is about 40Mbytes for each 1,000 subscribers with more than realistic levels of average activities (40 calls day, 2% daily churn rate, etc.). Thus, there is ample room for upscaling using higher-end machines.

For deployment by large operators, with possibly many million subscribers, it is clear that scaling out by distributed processing would be necessary. Additionally, the customer base would be geographically distributed over the planet, so communication latencies among datacenters and traffic splitting and routing must be taken into account. Finally, the emergence of new technologies and services, as well as company culture, will undoubtedly put additional constraints on the processing. From a data mining point of view, techniques for distributed model building will have to be incorporated. In fact, building several models at geographically distinct location may be advantageous to capture different customer patterns at different zones. Since the models themselves are compact, they could possibly be exchanged among machines and sites and be used cooperatively (e.g., with ensemble methods) for better accuracy.

8. Conclusions and Extensions

We have hopefully shown that stream mining technology may help customer churn prediction on high-volume streams originating from customer activity. The main difference with exiting, batch-oriented, data mining approaches to the problem is the ability of these technologies for reacting and adapting fast to changes in customer behavior, without human intervention, which may have a direct impact on revenue and image for companies. Although the system is a prototype far from being deployable, we have shown that even on a single low-end machine we can deal with quite high data speeds and gracefully handle all the churn prediction process, including user segmentation and connecting with the customer relation management subsystem.

Further work and additional research includes testing the system with real subscriber data collected from live networks and combine additional data sources from outside operators boundaries.

Acknowledgements

This work was supported by a collaboration agreement between Ericsson Research and U. Politècnica de Catalunya. Research at UPC was also partially supported by the BASMATI MICINN project (TIN2011-27479-C04-03). We thank Albert Bifet and Germán Blanco, for their help at various stages.

References

[1] C. Archaux, H. Laanaya, A. Martin, and A. Khenchaf. An svm based churn detector in prepaid mobile telephony. In *Intl. Conf. on Information and Communication Technologies: from Theory to Applications (ICTTA)*, Damascus, Syria, 19-23 April 2004.

[2] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. MOA: Massive online analysis. *J. Mach. Learn. Res.*, 11:1601–1604, August 2010.

[3] J. Gama. *Knowledge Discovery from Data Streams.* Data Mining and Knowledge Discovery. Chapman & Hall/CRC, 2010.

[4] J. Gama and M.M. Gaber. *Learning from Data Streams: Processing Techniques in Sensor Networks.* New generation computing. Springer, 2007.

[5] B. Q. Huang, T. M. Kechadi, B. Buckley, G. Kiernan, E. Keogh, and T. Rashid. A new feature set with new window techniques for customer churn prediction in land-line telecommunications. *Expert Syst. Appl.*, 37(5):3657–3665, May 2010.

[6] Bing Quan Huang, Mohand Tahar Kechadi, and Brian Buckley. Customer churn prediction in telecommunications. *Expert Syst. Appl.*, 39(1):1414–1425, 2012.

[7] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proc. 2001 ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, pages 97–106, 2001.

[8] Sahand KhakAbi, Mohammad R. Gholamian, and Morteza Namvar. Data mining applications in customer churn management. In *Proceedings of the 2010 International Conference on Intelligent Systems, Modelling and Simulation*, ISMS '10, pages 220–225, Washington, DC, USA, 2010. IEEE Computer Society.

[9] Scott A. Neslin, Sunil Gupta, Wagner Kamakura, Junxiang Lu, and Charlotte H. Mason. Defection Detection: Measuring and Understanding the Predictive Accuracy of Customer Churn Models. *Journal of Marketing Research*, 43(2):204–211, May 2006.

[10] P.C. Pendharkar. Genetic algorithm based neural network approaches for predicting churn in cellular wireless network services. *Expert Syst. Appl.*, 36(3):6714–6720, April 2009.

[11] Anita Prinzie and Dirk Van den Poel. Incorporating sequential information into traditional classification models by using an element/position-sensitive SAM. *Decis. Support Syst.*, 42(2):508–526, November 2006.

[12] Yossi Richter, Elad Yom-Tov, and Noam Slonim. Predicting customer churn in mobile networks through analysis of social groups. In *SIAM Intl. Conf. on Data Mining (SDM)*, pages 732–741. SIAM, 2010.