The Orange Customer Analysis Platform

Raphaël Féraud, Marc Boullé, Fabrice Clérot,
Françoise Fessant, and Vincent Lemaire

Orange Labs,
2 avenue Pierre Marzin
22300 Lannion

Abstract. In itself, the continuous exponential increase of the data-warehouses size does not necessarily lead to a richer and finer-grained information since the processing capabilities do not increase at the same rate. Current state-of-the-art technologies require the user to strike a delicate balance between the processing cost and the information quality. We describe an industrial approach which leverages recent advances in treatment automatization and relevant data/instance selection and indexing so as to dramatically improve our capability to turn huge volumes of raw data into useful information.

1 Introduction

The rapid and robust detection of the most predictive variables is a key factor in a marketing application. An industrial customer targeting platform developed at Orange Labs, capable of building predictive models for datasets having a very large number of input variables (ten of thousands) and instances (tens of thousands), is currently in use by Orange marketing. A key requirement is the complete automation of the whole process. The system extracts a large number of variables from a relational database, selects a subset of informative variables and instances, and efficiently builds in a few hours an accurate classifier. When the models are deployed, the platform exploits sophisticated indexing structures and parallelization in order to compute the scores of millions of customers, using the best representation.

The challenge KDD Cup 2009 [1] was to beat the in-house system developed by Orange Labs on three standard marketing campaigns : the propensity of customers to switch provider (churn), buy new products or services (appetency), or buy upgrades or add-ons proposed to them to make the sale more profitable (up-selling). The results of KDD Cup show that automatic modeling on thousands of variables leads within few hours to results close to those obtained by top level researchers in a month. Knowing that a datamart containing tens of thousands of variables describing millions of instances is practically unfeasible, this interesting result raises two questions for industrial use:

1. How to build hundreds of models on tens of thousands of variables ?
2. How to deploy hundreds of models on millions of instances ?

These questions were at the origin of the Customer Analysis Platform.

This paper describes this industrial customer targeting platform developed at Orange Labs. The paper is organized as follow : Section 2 describes the process of the targeting of marketing campaign, Section 3 presents the processing architecture and the modeling step, Section 4 presents the Deployment Cycle and Section 5 several experiments.

2 The targeting of marketing campaign

Customer Relationship Management (CRM) is a key element of modern marketing strategies. The most practical way to build useful knowledge on customers in a CRM system is to produce scores to detect churn, propensity to subscribe to a new service... Hundreds of scores are produced by Orange marketing each month. These scores are then injected in the CRM tools to target incoming and outcoming marketing campaigns. The scoring process is an industrial process containing a lot of complex tasks :

1. Each month a customer datamart, called datafolder, is fed from the datawarehouse. As the datamart contains different domains of data such as customer, billing, uses, contacts..., it is ready to be used when the last domain of data is produced. Billing data are the last produced, at the middle of the current month. Few days after, all the scores have to be produced to feed CRM tools.
2. For each marketing campaign, a filter is applied on the datamart. The filter defines the population concerned by the marketing campaign. For example, for a churn purpose the filter selects the customers you want to retain.
3. Then the current model used to target customers of the marketing campaign is tested. To test the current model, the scores of the previous month are compared to the present. If the accuracy indicator such as AUC is not stable in comparison to previous values, a new model is learnt with recent data. The lifetime of a model is usually in the order of one year.
4. The model is deployed to produce scores of the current marketing campaign.

This description of the targeting process shows that the bottleneck is not the modeling task but the deployment task : most of the models are re-used each month, and the time constraint is strong on deployment since hundreds of scores have to be produced for millions of customers in only few days.

3 Platform Architecture

3.1 Introduction

This section gives an overview of the Orange Customer Analysis Platform. The block diagram of the Orange Customer Analysis Platform is presented Figure 1. The next sections of this paper enter more in depth to detail several parts of this platform.

The first step to obtain scores on customers is to build a datafolder: the input data from information system are structured, and stored in a simple relational database (see Section 3.2). Then the platform includes 2 mains cycle:

1. The modeling cycle which includes two different steps:
   - The modeling step: using the extraction language, with specification 'A', a modeling database is extracted from the datafolder (see section 3.3). This database contains $P1$ instances and $N1$ explanatory variables. $P1$ is a subset of the customers to be scored. Using this database the modeling step is performed; this step includes two main functions: the variable selection (see Section 3.4) and the construction of a classifier (see Section 3.5). At the end of the modeling step one has a classifier which uses a subset of the $N1$ explanatory variables: $N2$ ($N2 \ll N1$). Only this $N2$ variables will be used in the extraction language with specification 'B' and 'C'.
   - The indexing step: using extraction language, with specification 'B', a filtered database is extracted from the datafolder. This database contains $P2$ instances and $N2$ explanatory variables where $P2$ represents all the customers to be scored at the end of the complete process ($P2 \gg P1$). Then the instance selection step is performed to extract a paragon table (see Section 4.2) which contains $P3$ real customers ($P3 \ll P2$), each described by $N2$ explanatory variables. The application of a k nearest neighbor (knn) and Locality Sensitive Hashing (LSH) algorithms on this table allows the creation of an indexation table (see Section 4.3). This indexation table links any customer ($P2$) to a customer of the paragon table ($P3$).

   The complete output of the modeling cycle is therefore: a classifier, an indexation table, and the extraction specification 'C' corresponding to the $N2$ explanatory variables, and to the $P3$ paragons.

2. The deployment cycle: knowing the output of the modeling cycle the extraction query with specification 'C' can be written in the extraction language and applied on a new data folder. This produces the paragon table (PT table in the Figure 1) and the indentifier table (ID table in the Figure 1). Then the classifier is applied on the paragon table to obtain the scores of the paragons. Finally knowing the scores of the paragons and the indexation table a joint is realized and therefore all the customers are scored (see Section 4.1).

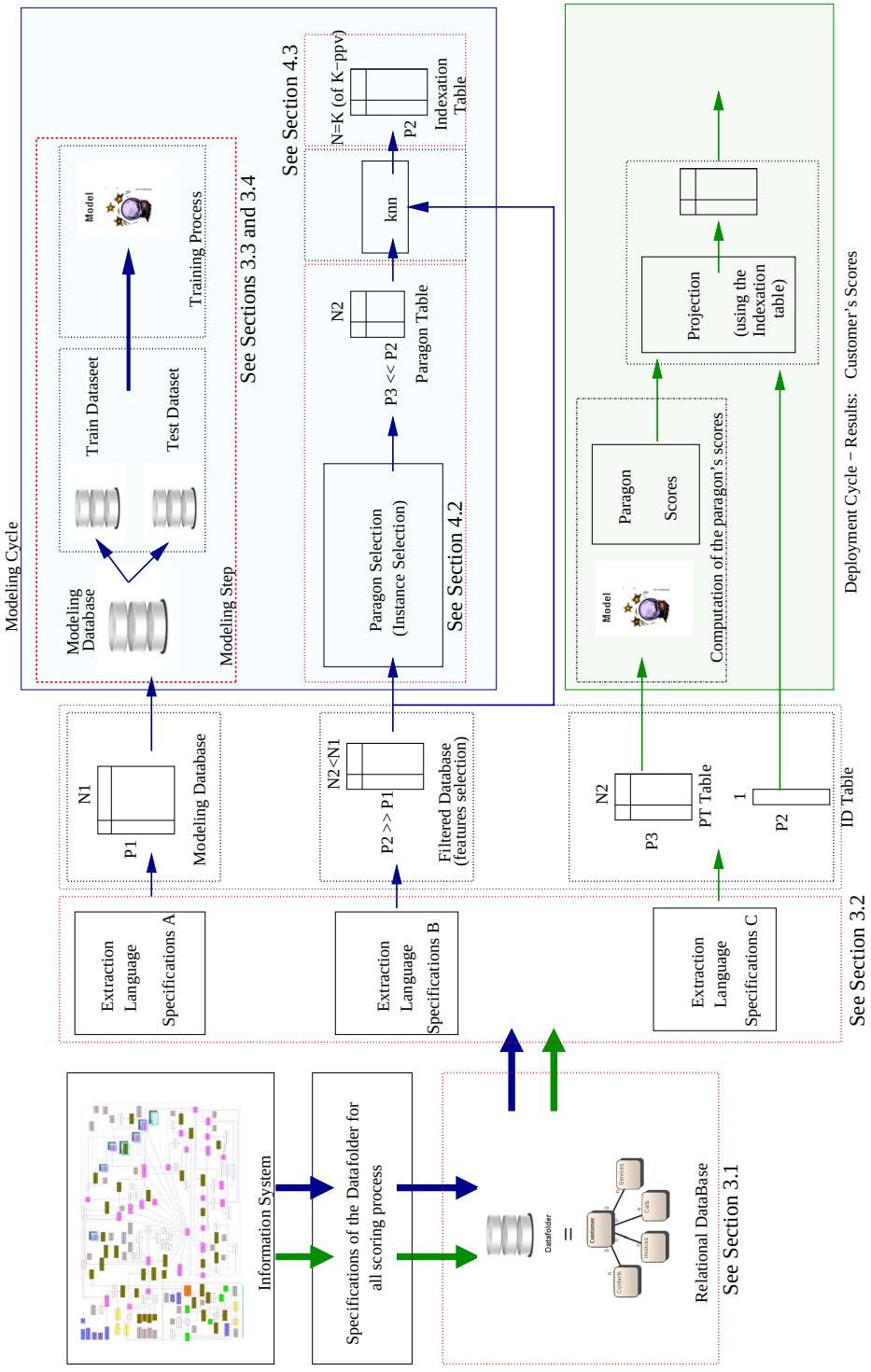

Fig. 1. The Orange Customer Analysis Platform. Blue arrows and background : modeling cycle, Green arrows and background: deployment cycle.

3.2 The data folder

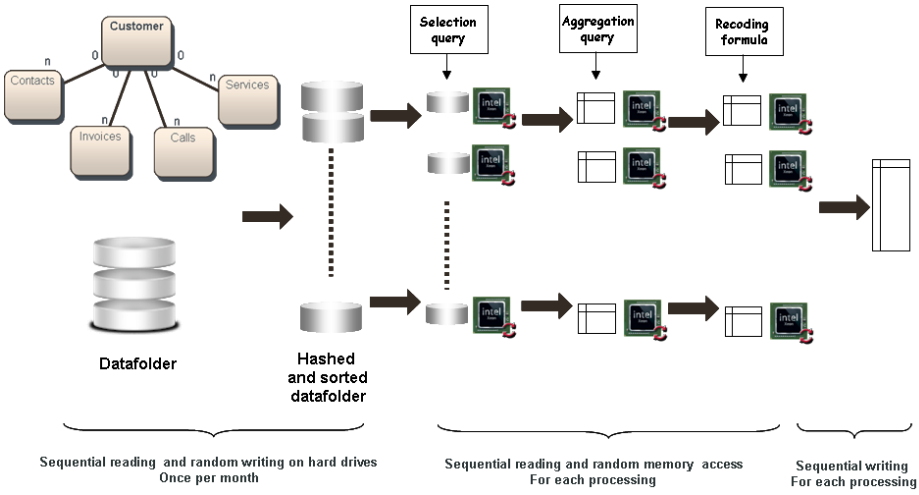

Fig. 2. Principle : data are normalized and hashed in a star schema database. Using extraction languages, learning algorithms drive data preparation, modeling and instance selection. A server executes in parallel most of the process. In this illustrative example, a flat table is extracted from the input datafolder

Unlike the current practice of data mining architecture, the explanatory variables are not a priori designed and computed in a datamart. In our platform architecture, the input data from information system are structured, and stored in a simple relational database : the data folder (Figure 2). The explanatory variables are built and selected automatically for each specific marketing project. In order to be computed in parallel and in memory, the datafolder is hashed in small datafolders of size 1 Go (Figure 2).

The data folder model provides a unique view of the available input data sources, normalized according a star schema:

- The primary table is related to the marketing domain. For customer data analysis, this table contains all the fields directly connected to the customer, such as his name or address.
- The secondary tables have a 1-N relationship with the primary table. Each instance of the primary table may be related to a variable number of instances of a secondary table. For telecommunication data for example, the secondary tables contains the list of services, of usages of theses services, the call details.

This type of data modeling has a large expressiveness, suitable for many data mining projects. It offers an efficient trade-of between single-table data mining

and full multi-relational data mining. The star schema allows to efficiently build many constructed variables, when the join key belongs to the primary table, whereas in a traditional data-warehouse, the construction of one single variable may involve multiple table joins. Finally, this star schema modeling allows the design of formatted data extraction languages, with the purpose of automation of the data mining process.

3.3 Data Extraction

The data extraction functionality of the platform is parametrized using three languages:

- a selection language to filter the instances,
- a construction language to build a flat instance x variables representation from the data folder,
- a preparation language to specify the recoding of the explanatory variables.

These languages are both simple enough to be automatically exploited by the process of variable selection and expressive enough to build a large variety of explanatory variables. Each language expression deals with at most two tables: the primary table plus eventually one secondary table. The join key always belongs to the primary table, and the selection and construction operands exploit the fields of any table, primary or secondary. For example, to build the number of usages of each service per weekday for all customers, one single language expression needs to be specified, with the use of the "Count" operator on the secondary table "Usage" with two operands "WeekDay(Date)" and "Label(ServiceId)". It is then possible to specify up to thousands of variables to construct, using one single expression of the construction language.

3.4 Variable Selection

The platform architecture allows to easily build flat data tables with up to tens of thousands of constructed variables. In order to select the best representation, that is the best subset of informative variables, a powerful variable selection method [2, 3] is required, both robust and efficient. In the context of decision trees [4–6], supervised discretization methods are employed at each node of the tree in order to select the next split variable, using filter criteria based on statistical tests [7], error rate or entropy [8]. When the number of intervals of the discretization is a free parameter, the trade-off between information and robustness is an issue. In the MODL (Minimum Optimized Description Length) approach, supervised discretization [9] (or value grouping [10]) is treated as a nonparametric model of conditional probability of the output variable given an input variable. The discretization is turned into a model selection problem and solved in a Bayesian way. The best discretization and value groupings are optimized using the bottom-up greedy heuristic described in [9]. One advantage of this filter approach is that non informative variables are discretized in one

single interval and can thus be reliably discarded. The algorithmic complexity of $O(n \log n)$ of this heuristic and the excellent reliability of this method allow to preprocess a very large number of variables, around 50000 in our experiments, and to select a small subset of informative variables, typically 10% of the input variables in the marketing domain.

3.5 Modeling

The naive Bayes classification approach [11–13] is based on the assumption that the variables are independent within each output label, and simply relies on the estimation of univariate conditional probabilities. In the Orange Customer Analysis Platform, this approach benefits from the high quality MODL preprocessing. The naive independence assumption can harm the performance when violated. In order to better deal with highly correlated variables, the selective naive Bayes approach [14] exploits a wrapper approach to select the subset of variables by optimizing the classification accuracy. In this seminal work, the search algorithm has a quadratic time complexity w.r.t the number of the variables, and the selection process which is prone to overfitting. In [15], the search algorithm is able to process large numbers of variables with super-linear time complexity, and the over-fitting problem is tackled using a Bayesian regularization approach. Finally, a model averaging approach is applied in order to achieve better accuracy and reliability. Using the naive Bayes assumption, weighting many models of variable selection reduces to one single naive Bayes classifier with weighted variables, allowing an efficient deployment of the ensemble of selective naive Bayes classifiers.

To summarize, in the platform, a selective naïve Bayes classifier [15] leverages the MODL preprocessing, variable selection regularization and model averaging in order to build effective scores fully automatically. This method is efficiently implemented into the Khiops scoring tool (available as shareware, see `www.khiops.com`).

4 Efficient Deployment

4.1 Principle

To produce scores, a model has to be applied for all instances on all explanatory variables. To speed up this process, a table of paragons containing representative individuals is extracted. The paragons are connected by an index to all the population. The scores of all instances are obtained by a simple join between the table of the paragons and the index. This method of deployment is particularly effective when the model is deployed several times. For example for monthly marketing campaigns, only the reduced table of the paragons is built each month to produce the scores of all instances. This approach makes it possible to increase dramatically the number of scores which can be produced on the same technical architecture.

4.2 Paragons Selection

The table of the paragons is crucial for the final performance of the system. A poorly representative paragon table leads to ineffective scores, on the other hand, a too large paragon table increases computational cost.

The table of paragons is drawn from the datafolder to be representative of the variables relevant for the model. To produce and maintain online a sample of size n, Reservoir Sampling algorithm [16]) can be used. An inclusion probability of $n/(t+1)$ is given for each tuple arrived at time t. An interesting property of this algorithm is that, when t tuples have been observed, all the t tuple have the same probability to be included in the reservoir: n/t. Biased versions of this algorithm may take into account recent data ([17]) or weighted data ([18–20]).

As the frequencies of discretized explanatory variables are known from the variable selection stage, a biased version of Reservoir Sampling can be used to draw the paragons. To control and speed up the convergence time, we use a deterministic version of Biaised Reservoir Sampling. A reservoir is filled until it reaches the desired size P without removing any instance :

1. The reservoir is initialized with the first K instances.
2. At each iteration an instance is chosen to optimize Khi^2 criterion between theoretical frequencies and frequencies observed in a windows of size M, with $M \ll P$.
3. Then the search window is shifted of L instances in order to fill the reservoir of size P in one pass on the table of size N : $L = (N - M)/P$.

The size of the search windows allows to tune the trade-off between computational time cost and accuracy of the algorithm : the more M is, the more the accuracy is and the less the computational time cost is.

4.3 Data Indexing

The problem to be solved is simple to state: being given an individual, to find his nearest neighbor in the table of paragons. The $L1$ norm between the explanatory variables is used to evaluate the distance between instances. This task has to be executed for all the instances of the datafolder. The search of nearest neighbors is an expensive operation. Its naive implementation implies an exhaustive research among the paragons, therefore a complexity in $\mathrm{O}(nmp)$, n being the number of instances, m the number of explanatory variables and p the number of paragons. In order to accelerate the research of nearest neighbors, a compromise between speed and accuracy can be done : to find a paragon close to the nearest using Locality Sensitive Hashing [21] allows. This algorithm is based on a technique of hashing to select good candidates among the paragons to be close to the nearest. Then an exhaustive search is done on good candidates to find the paragon. Our implementation of this technique makes it possible to bring back the complexity of the search close to $\mathrm{O}(nm\sqrt{p})$. It reduces the computational cost of a factor 300 per 100000 paragons, and leaves to the user the control of the compromise speed / performance.

5 Experiments

We compared the scores produced with our platform (including the Khiops scoring tool) and with the current model for several Orange marketing campaigns.

The current model is built with KXEN [22] on a datamart containing about 700 explanatory variables. To supply the platform, we have collected data on about one million of customers between January and June 2005. The information comes from decisional applications of Orange Company. The first four months have been used to build the customer profiles, the last two to compute the target variable. 20% of the customers are kept for the evaluation of the models.

The performance of a model is measured with the cumulative gain curve (Figure 3). It is a graphical representation of the advantage of using a predictive model to choose which customers to contact. The x-axis gives the proportion of the population with the best probability to correspond to the target, according to the model. The y-axis gives the percentage of the targeted population reached.

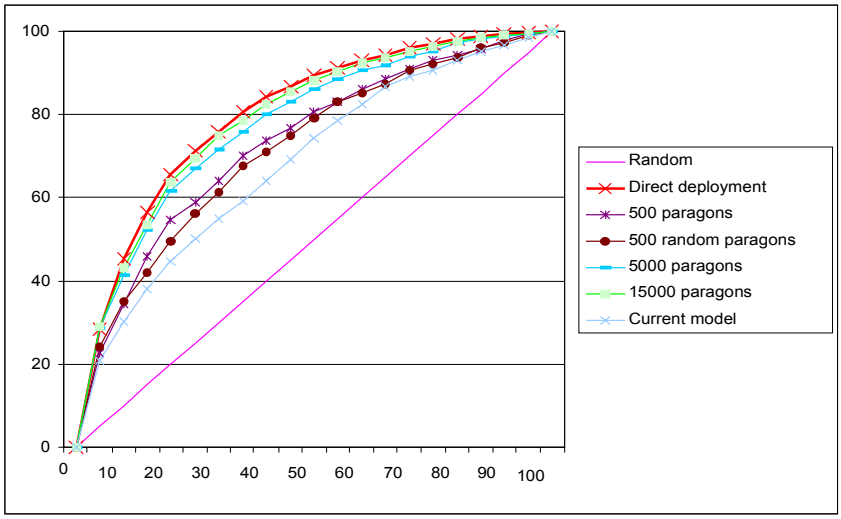

Fig. 3. Lift curve of predictive models for churner detection

The goal of the campaign presented below is to prevent a customer to switch ADSL provider.

We plotted the cumulative gain curves for several predictive models on Figure 3. The diagonal represents the performance of a random model. If we target 20% of the population with this random model, we are able to reach 20% of the customers who will churn in next two months. With the current model, when 20% of the population is targeted, 45% of the fragile customers are reached. Compared with a random targeting we have a gain (G_1) of 2.25 ($G_1 = \frac{45}{20} = 2.25$)

The automation of the search of representation has led us to select a model based on 191 explanatory variables chosen among a set of 50000 variables.

The model deployment is then achieved on all the instances with a variable number of paragons: 500, 5000, 15000 and also directly on the population. In the case of a direct deployment on all the instances, if we contact 20% of the population based on this new modelling, 65% of the fragile customers are targeted. Compared with the current technique, we have a gain (G_2) of 1.4 ($G_2 = \frac{65}{45} = 1.44$). This improvement remains true for the entire cumulative gain curve.

An in-depth analysis of the most relevant variables kept by the targeting model built with the platform can help us draw the portrait of a typical churner. His engagement ends in next 4 months, he lives in a dense area, he is young (between 14 and 27 years old) and his volume of traffic has changed a lot (decrease or strong increase) in last 3 months. 5 of the 10 most important variables are not present in the initial datamart, used for the current model. They have been constructed directly from the data folder and specifically for the churn campaign. This is the strength of our methodology: with our platform we are able to explore a large number of new variables on demand, according to a specific campaign and select the most relevant of them.

Let's turn now to score deployment with paragons. When such a technique is applied, there is a loss of reliability which depends on the number of paragons. The targeting comes close to the best when the number of paragons increases but it is also very costly. For example, when 5000 paragons are used to represent 1000000 customers, at a level of 20% of the targeted population, 60% of the fragile customers are reached (+40% of gain compared with a random targeting and +15% compared with the current technique). With 15 000 paragons, the performances are similar to those of the direct deployment. To evaluate the quality of the algorithm of paragon selection, we have compared the performances obtained when the paragons are randomly selected and when the paragons are using a biased reservoir sampling on the theoretical distribution of explanatory variables. With 500 paragons, at the level of 20% of population, 50% of the target is reached for the random selection and 55% with biased reservoir sampling (Figure 3).

The whole process of extraction of a paragon table from one million customers and a representation space of 50000 variables takes about 3 hours on a server with 16 processors and 32 Go of RAM. One third of processing time is for the selection of the representation and two thirds are for the search and indexation of paragons. Once the paragons are available, the score production from the paragon table takes less than one minute.

One processing hour is necessary in a direct deployment to generate a table of one million instances with 191 explanatory variables and apply the predictive model on this table. It is very efficient to use paragons for the deployment of a recurrent score like fragility scores or ADSL recruiting. For an opportunist score such as appetency to a specific offer, a direct deployment is better.

6 Conclusion

We have described a data-mining platform which allows to build predictive models using two orders of magnitude more explanatory variables than the current state-of-the-art, resulting in a dramatic improvement of performances. The Orange Customer Analysis Platform relies on a novel architecture which allows to leverage recent advances in treatment automatization and relevant data/instances selection and indexing. The processing time associated with data table flattening remains the main limitation to the exploration of an even larger data space. The conception of an explanatory technique guiding the flatening towards the most promising areas of such huge spaces is a direction for further research.

References

1. `http://www.kddcup-orange.com/`: last access on 28th december.
2. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of Machine Learning Research **3** (2003) 1157–1182
3. Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L., eds.: Feature Extraction: Foundations And Applications. Springer (2006)
4. Kass, G.: An exploratory technique for investigating large quantities of categorical data. Applied Statistics **29**(2) (1980) 119–127
5. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. California: Wadsworth International (1984)
6. Quinlan, J.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
7. Kerber, R.: Chimerge discretization of numeric attributes. In: Proceedings of the 10th International Conference on Artificial Intelligence, Cambridge, Massachusetts, MIT Press (1992) 123–128
8. Kohavi, R., Sahami, M.: Error-based and entropy-based discretization of continuous features. In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, AAAI Press/MIT Press (1996) 114–119
9. Boullé, M.: MODL: a Bayes optimal discretization method for continuous attributes. Machine Learning **65**(1) (2006) 131–165
10. Boullé, M.: A Bayes optimal approach for partitioning the values of categorical attributes. Journal of Machine Learning Research **6** (2005) 1431–1452
11. Langley, P., Iba, W., Thompson, K.: An analysis of Bayesian classifiers. In: 10th National Conference on Artificial Intelligence, AAAI Press (1992) 223–228
12. Domingos, P., Pazzani, M.: On the optimality of the simple bayesian classifier under zero-one loss. Machine Learning **29**(2-3) (1997) 103–130
13. Hand, D., Yu, K.: Idiot bayes ? not so stupid after all? International Statistical Review **69**(3) (2001) 385–399
14. Langley, P., Sage, S.: Induction of selective Bayesian classifiers. In: Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann (1994) 399–406
15. Boullé, M.: Compression-based averaging of selective naive Bayes classifiers. Journal of Machine Learning Research **8** (2007) 1659–1685
16. Vitter, J.: Random sampling with a reservoir. ACM Trans. Math. Software **11**(1) (1985) 37–57

17. Aggrawal, C.: On biased reservoir sampling in the presence of stream evolution. In: Proceedings of the VLDB conference. (2006)
18. Chaudhuri, S., Motwani, R.: On sampling and relational operators. In: IEEE on Data Engineering. (1999)
19. Kolonko, M., Wasch, D.: Sequential reservoir sampling with a non-uniform distribution. Technical report, University of Clausthal (2004)
20. Efraimidis, P.S., Spirakis, P.G.: Weighted random sampling. Technical report, Research Academic Computer Technology Institute (2004)
21. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: VLDB Conference. (1999)
22. `http://www.kxen.com/`: last access on 21th december.