

Predicting Customers Churn in a Relational Database

Catalin CIMPOERU, Anca ANDREESCU

Bucharest University of Economic Studies, Bucharest, Romania

The Faculty of Cybernetics, Statistics and Economic Informatics

catalincimpoeru@yahoo.com, anca.andreescu@ie.ase.ro

This paper explores how two main classical classification models work and generate predictions through a commercial solution of relational database management system (Microsoft SQL Server 2012). The aim of the paper is to accurately predict churn among a set of customers defined by various discrete and continuous variables, derived from three main data sources: the commercial transactions history; the users' behavior or events happening on their computers; the specific identity information provided by the customers themselves. On a theoretical side, the paper presents the main concepts and ideas underlying the Decision Tree and Naïve Bayes classifiers and exemplifies some of them with actual hand-made calculations of the data being modeled by the software. On an analytical and practical side, the paper analyzes the graphs and tables generated by the classifying models and also reveal the main data insights. In the end, the classifiers' accuracy is evaluated based on the test data method. The most accurate one is chosen for generating predictions on the customers' data where the values of the response variable are not known.

Keywords: Data Mining, Predictive Analytics, Classification, Decision Tree, Naïve Bayes, Churn Analysis, Microsoft SQL Server

1 Introduction

Nowadays, predictive analytics is one of the common *buzz words*. Its methods and concepts are scientific, based on computer programming, mathematics and statistics, yet the interest for the topic went well beyond academics and research, to business/corporate area and, more recently, to the general public. In 2012, Nate Silver, editor in chief of FiveThirtyEight blog and author of the book *The Signal and The Noise*, became famous while correctly predicting the winner of the presidential elections in The United States, in all 50 states and the District of Colombia. In the same year, Harvard Business Review published a frequently mentioned article which described the *data scientist* as “the sexiest job of the 21st century” [1]. In 2014, Goldman Sachs, one of the most prestigious financial institutions in the world published a consistent paper, showing their predictions for the Football World Cup that took place in Brazil [2]. The same Nate Silver and his team at FiveThirtyEight also took their chance and offered alternative predictions for the main football event of the year. In the same year,

Warren Buffet, the well-known financial investor and one of the richest people on Earth, pushed the prediction challenge further and announced his willingness to offer one billion US dollars to anyone who correctly predicts the outcome of all 63 games in this year's NCAA men's college basketball tournament [3].

In most of the cases, the predictive analysis are oriented to solving more “serious” problems like identifying customer with a propensity to churn, detecting fraud or online spam, assessing the risk of an investment, sales forecast, predicting a medical diagnostic etc.

Lots of data science libraries are continuously created around open source programming languages like R and Python. They are free and quickly integrate new algorithms. At the same time, the main commercial database systems (SQL Server, Oracle etc.) introduced their own data mining modules, offering easiness of use for analysts and the possibility to build and integrate data models with the relational database systems. Starting from such data, available in a relational database, the *objective* of the

current paper is to show and explain a few models of predicting customers having a high propensity to churn after expiring the availability of their products. Identifying these cases is highly important for every business built on the subscription model (telecom, antivirus, cable-television etc.), as the reduction of the churn rate among customers can positively impact the retention rate and the profitability of any business. According to *The Chartered Institute of Marketing* [4], in some cases, preventing a customer from churning, involves costs from 5 to 20 times lower than acquiring a new customer.

2 The Data in the Relational Database

The starting point of this project is a relational database, made by 12 tables, designed after a snowflake model and built using Microsoft SQL Server 2012 technology. We deal with the software industry, where a company sells its products online, using the above mentioned subscription model. The data refer to online transactions, products, customers' details, offers, licenses, customer care activities involving employees and customers, online incidents, regions etc. For all the customers, based on the existing data, we are able to find out if they are still customers, or not. However, there are still a few thousand customers (both new and renewals), whose behavior we would like to predict, so that the company can apply a commercial "treatment" for the ones with a high probability to churn.

In order to build a complete profile of all the customers (with as many relevant variables as possible), we created two SQL *views* so that we can bring together:

- the data describing the "customers", basically including attributes like gender, marital status, income etc. These data were taken from the Customers table;
- the data referring to the acquisition behavior (how many times the customers renewed the license, weekly activity) and some details about their latest transaction (promo or regular price, automatic or

manual renewal). These data were based on the transactions table (OnlineSales);

- the data referring to the activity of the customers in relation to the company (number of calls to the customer care) and existing events on the customer's computers since their latest acquisition (software incidents).

The two SQL *views* are moderately complex, using a few *common table expressions* which are joined with the main Customers table, in order to enrich the number of available descriptive features. They are identical in terms of attributes, the only difference is that one of them represents the set of customers where we know the binary values (0, 1) of the target attribute (IsChurn); the other one, represents the customers where the values of the target attribute needs to be predicted, as those customers still own an available subscription that has not expired yet.

Given the specific of cases that we deal with (instances with known and unknown target variable), and considering the *discrete* or *categorical* type of the variable needed to be predicted, we can consider the process of identifying the future churn customers as a classification problem.

This paper describes a project that implemented two classifiers in order to solve the prediction objective stated earlier: *Decision Tree* and *Naïve Bayes*. Each of them offers its own solution to the problem of predicting which customers will NOT renew their software license and, thus, will become churn customers. After assessing the accuracy of the solutions of each classifier, we choose the one offering the best answer.

Each of the two models are trained on a subset (*training* data) of instances of the SQL *view* including the real values of the response variable (IsChurn: 1 or 0). Afterwards, predictions are made for the remaining instances of the same SQL *view* (*test* data), for which we initially pretend not knowing the values of the target variable (attribute IsChurn). The predictive accuracy of each model is then assessed comparing the real values of the attribute IsChurn with the values predicted by each of the three

classifying models. In the end, the model offering the best predictions is chosen to predict the values of the IsChurn target variable of the data in the second SQL view, which refers to the customers whose license is about to expire and, consequently, we do not know yet their behavior in terms of renewing or churning.

The rest of the paper follows the classical path outlined above. We will first explore each of the two classifying models in terms of underlying concepts, methodology and outputs resulting after modeling the *training* data.

3 The Decision Tree Classification Model

The essential idea that is the basis of the decision tree techniques is that there are some “hidden” rules in the data which determines the categorical values of the response variable. These rules are discovered by the algorithm by dividing the initial set of data into multiple smaller subsets, using as filters the values of the descriptive variables which are decided to be used for the split. In order to choose the descriptive variable responsible for making the split, each attribute is evaluated for determining the purity of the response variable that would result in case the division is made based on one value or another of the descriptive variable [5]. The above description can be represented by a hierarchical structure, made by nodes. Besides the *model node*, which represents the very base, there are 3 other types of nodes in a decision tree [5]:

- the *root node*, which is the starting point of the ramification;
- the *intermediate node*, which is further connected with other intermediate nodes, or with a leaf node;
- the *leaf node*, representing the values of the response variable.

In terms of the “hidden” rules mentioned earlier, they are revealed by following the path of data filtering starting from the root node to the leaf. In our case, the decision tree, generated by the algorithm implemented in Microsoft SQL Server Data Tools, identified 34 rules.

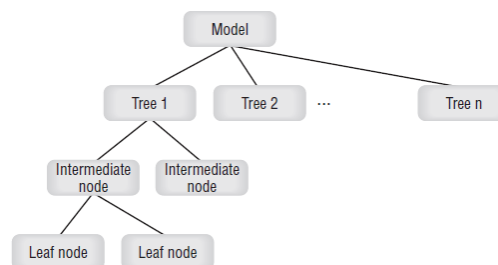


Fig. 1. The structure of a decision tree [5]

Theoretically, there is a high number of possible ways to build a decision tree, which increase as the number of variables and their specific states increase. However, there are some algorithms created with the purpose of eliminating the not optimal possibilities of making data divisions and, thus, speeding up the process. While building the tree, such algorithms, Hunt, ID3, C4.5 or CART, scan all the descriptive variables for identifying the most adequate one to make the data split. According to [6], there are two essential problems that the decision tree algorithms deal with:

1. a criteria is needed to choose between the variables, so that a proper split of data will follow;
2. another criteria is needed to decide when the partitioning of data is best to be stopped. That is because the “natural” stop, happening when all cases are “purely” classified, is likely to be inadequate, due to the poor ability to generalize the resulting rules, which might be too complex and, thus, also representing the “noise” in the data. This problem is known as *overfitting*.

Regarding the first problem, the typical approach of the Decision Tree algorithms is to choose the split variable based on the “least impurity” (or higher purity) of the response variable resulting after the data split would be made: e.g. the scenario when most of the IsChurn values is 1. The implementations are usually made using one of the existing mathematical criteria of measuring the impurity: *Entropy* or *Gini Index*.

$$Entropy = \sum p \times \log_2 \frac{1}{p}$$

p = frequency of one of the values of the variables which is considered for partitioning

Either Entropy or Gini Index being chosen, what we really want is finding the descriptive variable whose impurity is the lowest or so the *information gain* is the highest. Both the entropy and Gini Index have the highest values when the distribution of the values describing a categorical variable is uniform. Concretely, in order to determine the variable to be used for the data division, the algorithm compares the impurity state of the *parent node* with the ones of the *children nodes*, resulting after the division.

However, as [6] shows, the impurity measures are more likely to favour the attributes having a higher number of distinct states. That is why some decision tree algorithms (e.g. CART) actually proceed each time to a binary split (one value vs. all the rest), while others use a relative version of the impurity measure, by also considering the number of unique states of each variable that candidates to partitioning.

As mentioned earlier, another inherent problem of the decision trees algorithm is related to the identification of criteria for when to stop building the tree, so that the *overfitting* situation is avoided. For example, a built tree is likely to be overtrained when the node leaves are perfectly pure (100% made by one specific state of the response variable) and, at the same time, there are lots of intermediate nodes between the root node and the leaves and there is a small number of cases standing behind each existing branch. If this scenario happens, the model is likely to be too specific, closely capturing the noise of the data and, consequently, difficult to generalize for a new set of data, in order to predict the states of the target variable. This kind of fully developed decision tree do produce accurate predictions for the *training* data, but most likely, will be highly inefficient with lots of predictive errors, when applied on the *test* data.

The *overfitting* situation can be avoided by actually limiting the length of the decision

tree. This is usually achieved using either forward pruning, or backward pruning.

Forward pruning, which is also the method implemented by the Microsoft Decision Trees algorithm in SQL Server Data Tools, consists in prematurely stopping the generation of the decision tree in situations when:

- 1) there is not enough information gain at the intermediate node level to justify another split of the data. In cases when the impurity level that would result after a new division would increase above a certain level, the algorithm would abort the division and the potentially intermediate node becomes leaf node. In SQL Server Data Tools, the inhibition of the decision tree growth is controlled by the “complexity_penalty” parameter;
- 2) the potentially new node would be supported by just a few instances (e.g. 10-20 cases) and, consequently, it would not be relevant. This cutoff approach is controlled in SQL Server Data Tools by the parameter “minimum_support”, whose values can be controlled by the user;
- 3) the branch of the tree would simply have too many levels (e.g. 5-7) and, consequently, the derived rule for the leaf node would become too complex. Controlling this situation, known as “maximum depth cutoff”, is also made through the values set for the “complexity_penalty” parameter.

Backward pruning, which is considered by [7] to facilitate better results, allows the decision to be generated completely, but in the end eliminates the branches which are considered not relevant. As [6] mentions, the branches cutoff can be made using one of these two methods: a) the dismissed branch can be replaced by a leaf representing the majority class in the respective sub-tree; b) the dismissed branch can be replaced by the most used branch of the respective sub-tree.

As [8] notes, “a smaller tree with fewer splits might lead to lower variance and a better interpretation at the cost of a little bias”.

4 Exploring Customers data with Decision Tree

The training data represents a random selection of 11,851 instances, approximately 70% of the observations available in the SQL view that show data about the customers that either renewed their software license (IsChurn = 0), or – on the

contrary – became churn customers (IsChurn = 1). In SQL Server Data Tools, the training data, modeled by the classifiers can be explored by accessing the Mining Model Viewer tab. In the bellow figure, more intense blue refers to a higher concentration of customers who churn.

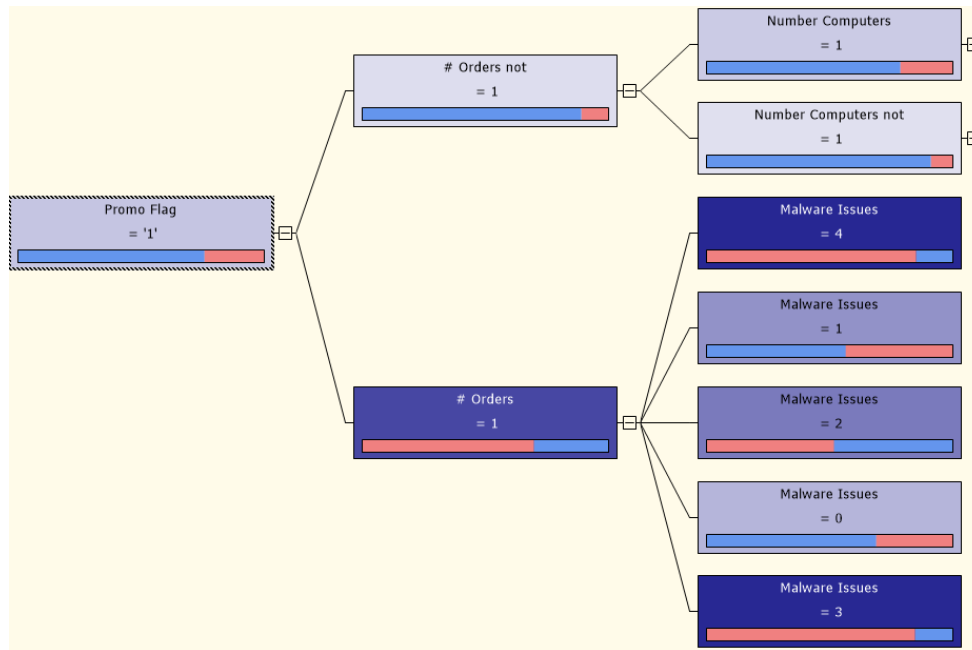


Fig. 2. The partial ramification of the root node PromoFlag = 1

Starting from a similar frequency of churn (1) and non-churn (0) customers, the Decision Tree model built two root nodes, starting from the binary values of the variable PromoFlag, which generated the least impurity level among from resulting types of customers. The “PromoFlag” is the most important attribute for a customer’s decision to churn. This variable splits the initial set of

data between customers whose latest acquisition was made for a promotional price (value 1), or for a regular price (value 0). As per the below table, the probability that a customer would become churn decreases dramatically from 51.8% initially, to 24.6% in case of the customers that had acquired their latest license for a promotional price (PromoFlag = 1).

Value	Cases	Probability	Histogram
<input checked="" type="checkbox"/> 0	2825	75.36%	
<input checked="" type="checkbox"/> 1	922	24.64%	
<input checked="" type="checkbox"/> Missing	0	0.00%	

Fig. 3. Variable IsChurn distribution given that PromoFlag = 1

In the complementary situation when the customers had not bought the product for a promotional price, the percent of churn customers increases significantly from 51.8%

to 64.4%. However, as one can notice, this increase of customer churn ($\Delta = 64.4\% - 51.8\% = 12.6\%$) is not as much as there is the decrease of churn customers in case their

acquisition price was promotional ($\Delta = 51.8\% - 24.6\% = 27.2\%$). This means that while PromoFlag variable is an initial good

predictor for the customer behavior, this works better for predicting non churn rather than churn customers.


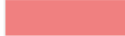
Value	Cases	Probability	Histogram
<input checked="" type="checkbox"/> 0	2882	35.57%	
<input checked="" type="checkbox"/> 1	5222	64.43%	
<input checked="" type="checkbox"/> Missing	0	0.00%	

Fig. 4. Variable IsChurn distribution given that PromoFlag = 0

The 1st root node: PromoFlag = 1

The intermediate nodes that start from the first root node binary divide the data into two type of cases: (new) customers with only one purchase (#Orders = 1) and (long term) customers with many purchases. As the figure shows, despite the fact that they bought the product for a promotional price (and, thus, have a propensity to renew), the first time purchasers are likely to be “offer hunters”, as they more frequently churn compared to the older customers, with many purchases. Going further on the branch, we immediately reach five leaf nodes, defined by the values of Computer Issues variable. Basically, the probability to churn further increases if the number of the technical problems on the computer is high. Thus, if the latest product has been bought on a promotional price and the customer placed his first order and the number of computer problem he/has had is high, the instance is classified as “churn” (IsChurn = 1); otherwise, given the above conditions, but the number of computer issues being low (zero or one computer issue), the instance is classified as “renewal” (IsChurn = 0).

The decision tree can be further explored in this manner. Given the parameters that we initially set-up, the model Microsoft Decision Trees generated 34 leaves nodes (equivalent to 34 rules), on 8 levels (one for the model node, one for the root nodes, five for the intermediate nodes and one for the leaf nodes).

5 The Naïve Bayes Classification Model

This classification model is based on the classical notions of conditional and initial

probabilities discovered and defined mathematically by Reverend Thomas Bayes in 1763, in one of his papers.

The *initial probability* refers to the (“overall”) probability of an event, the probability that comes “before any other information is available”.

Example 1:

$$P(IsChurn) = 1$$

Without any other data besides the ones related to the response variable (“IsChurn”), we can say that the (initial) probability of a customer to become churn is 51.8%, which is the relative frequency of the customers with the value 1 (8,778 customers out of a total of 16,930 included in the SQL view with the instances having a known response variable). The *conditional probability* refers to the (“filtered”) probability of an event happening, given that we already know the value of a certain descriptive variable. That’s why the conditional probability is also known as the posterior probability.

Example 2:

$$P(IsChurn = 1|Promo = 1)$$

The above notation refers to the conditional probability of customer to become churn, given that he/she acquired the latest product on a promotional price. Basically, this refers to the relative frequency of the customers not renewing their license (IsChurn = 1) out of the total customers who acquired their latest product on a promotional price. The total number of customers with Promo = 1 is 5,370 and, out of these, 1,334 have the value IsChurn = 1, which leads to a relative frequency and a probability of 24.8%.

The conditional/ posterior probability can be expressed even more complex, as in the

bellow example.

Example 3:

$$P(IsChurn = 1 | Promo = 1 \& Orders = 1 \& Computers = 2 \& CustServCalls = 3 \& AutoRenewal = 1 \& Region = APAC)$$

The above notation refers to the conditional probability that a customer is churn, GIVEN THAT the acquisition of his/her latest product was made on a promotional price AND the customer placed one order AND the number of computers is two AND the number of calls to customer care department is three AND the customer is “married” AND the region of residence is APAC.

According to [6] “estimating the posterior probabilities accurately for every possible

combination of class label and attribute value is a difficult problem because it requires a very large training set, even for a moderate number of attributes”.

The complexity of example 3 can be solved by decomposing the problem, so that we calculate the conditional probability of IsChurn = 1 not at once, at the same time, based on the all values together, but based on each value, in turn, one by one. After this step, according to Bayes formula, we can combine the resulting posterior probabilities, so that we get the same result as we would have got considering the more complex approach.

The bellow table shows the described approach, based on Bayes rule:

		Promo = 1	Orders = 1	Computers = 2	Cust Serv Calls = 3	Auto Renewal = 1	Region = APAC	Churn
CHURN	1	1,334	2,150	2,442	960	4,407	2,033	8,778
	0	4,036	734	2,202	540	4,479	1,308	8,152
% in Sub-total	1	25%	75%	53%	64%	50%	61%	52%
	0	75%	25%	47%	36%	50%	39%	48%
		Result	Calculation					
Multiplication	1	1.0%	meaning: 25% * 75% * 53% * 64% * 50% * 61% * 52%					
	0	0.3%	meaning: 75% * 25% * 47% * 36% * 50% * 39% * 48%					
	sum	1.3%						
Probability	1	76%						
	0	24%						

Fig. 5. Calculation of the conditional probability

In the above example, we calculated the conditional probability of the IsChurn values, given the mentioned conditions. Due to the fact that P(IsChurn=1) = 24% is lower than P(IsChurn=0) = 76%, the respective customer can be classified as not being churn.

The above example, illustrating *Bayes rule*, simplifies things as it “allows us to express the posterior probability in terms of the prior probability P(Y), the class-conditional probability P(X|Y) and the evidence, P(X)” [6].

$$P(Y|X) = \frac{P(X|Y) \times P(Y)}{P(X)}$$

[5] explains the above formula like this: the probability of an event Y given the evidence X is equal to the probability of the evidence X given the event Y multiplied by the probability of the event Y, and then normalized.

Multiplying the two probabilities in the numerator, we get the *common probability*, which is the probability that both events happen in the same time: e.g. probability of a

customer being churn $P(Y)$ and, in the same time, having bought the latest product on a promotional price $P(X|Y)$. The division operation performs a *normalization* of the result in the numerator, so that we get not an absolute value, but the relative weight within the overall probability of evidence $P(X)$.

The Naïve Bayes model uses the Bayes rule as a classification instrument, calculating the posterior probability of all the discrete values defining the response variable. However, there are certain, well known limitations of this algorithm. There is not enough space here to get into details, but one of the limitations that worth to be mentioned is reflected by the word “naïve” in the name of the model. The algorithm works with the assumption of independence between the descriptive attributes. This means that Naïve Bayes model considers that “the effect of the

value of one attribute on the probability of a given classification is independent of the values of the other attributes” [7].

6 Exploring Customers Data with Naïve Bayes

Exploring the results generated by the Naïve Bayes model when applied on the training data is available through the reports in the *Mining Model Viewer* tab in SQL Server Data Tools (SSDT). The *Attribute Profiles* report shows the distribution of the values defining the descriptive variables: on one hand, independent of the target variable (the column “Population All”) and, on the other hand, filtered by each of the two states (1 and 0) of the IsChurn variable. Here is a snapshot:



Fig. 6. Attribute Profiles for Naïve Bayes model

One can easily notice that the value 0 of the variable PromoFlag is correlated in a high degree with the value 1 of the target variable, IsChurn. This means that among the

customers who did not renewed their licenses, there is a high number of customers who had bought their latest product on a standard price (PromoFlag = 0). Among the

customers who did renew their licenses, there is a higher balance between the two types of customers (promo and non-promo). Another observation is related to the fact that the US-CA customers are more frequent in the category of those who renewed their license, compared to their amount in the complement category, of those who abandoned the product. On the other hand, the AutoRenewalFlag variable has similar distributions of its values in cases when the data are filtered for IsChurn = 1 and IsChurn = 0. This variable, by itself, is not significant in influencing the state of the IsChurn attribute and, that's why its predictive power is poor.

Without going here to further details, we can conclude by saying that the Naïve Bayes model reveals similar insights as the Decision Tree model.

7 Evaluating the Performance of the Classification Models

Usually, there are three main methods used to generate *test data*, in order to evaluate the performance of the classifiers: randomly

sampling the data into training and test subsets based on percentage or absolute number of cases, K-fold cross-validation and N-fold cross-validation. We choose the first method (30% of the overall cases as test data), also due to the fact that the last two methods are more suitable in situations with less data, while the SQL *view* that we are working with has almost 17,000 instances.

We evaluated the decision tree and Naïve Bayes models based on the predictive accuracy and based on different indicators calculated using the classification matrix: precision, true positive rate, false positive rate, F1 scores etc.

The *predictive accuracy* refers to the weighted amount of test instances that the model manages to classify correctly. The formula is straightforward: number of response values predicted correctly divided by the total number of cases existing in the test data set. The resulting figure is actually an estimate and a confidence interval is further calculated based on the value of the standard error.

Churn DecisionTree		95% confidence interval			
	Instances	Predictive accuracy	Standard error	Lower limit	Upper limit
True	3,845	75.7%	0.6%	74.5%	75.9%
False	1,234				
TOTAL	5,079				

Fig. 7. Estimation of the predictive accuracy for the Microsoft Decision Tree model

Churn NaïveBayes		95% confidence interval			
	Instances	Predictive accuracy	Standard error	Lower limit	Upper limit
True	3,681	72.5%	0.6%	71.2%	73.7%
False	1,398				
TOTAL	5,079				

Fig. 8. Estimation of the predictive accuracy for the Naïve Bayes model

The above figures show that, in terms of predictive accuracy, the best performing classifier was the Microsoft Decision Tree, being able to correctly predict between 74.5% and 75.9% of the cases. However, the predictive accuracy measure

mixes the predictions of the two values of the response variable into a single figure. What we really wanted from the beginning was to accurately identify the churn customers (IsChurn = 1). More than that, as [7] shows, the predictive accuracy can be a misleading

indicator if there is a high discrepancy between the distribution of the values of the binary response variable (e.g. one value in 1% of the cases and the other value in 99% of the cases).

The *classification matrix* [9] breaks the predictive accuracy in more figures, so that the performance of the classifiers can be evaluated separately, based on each of the classes defining the response variable.

		Predicted class		Total cases
		+	-	
True class	+	TP	FN	P
	-	FP	TN	N

Fig. 9. Classification matrix, reproduced after [7]

In cases like this, when we deal with a binary response variable, there are two types of possible errors, FP (false positive) and FN (false negative), corresponding to the two types of statistical errors. Starting from these two types of predictive errors, the data mining literature mentions two popular indicators for evaluating the classifiers: *Precision* and *Recall* [10].

Precision shows what percent of the instances classified as positive (TP + FP) are

really positive (TP). *Recall* shows what percent of the instances really positive (TP + FN) were correctly classified as positive.

$$Precision = \frac{TP}{TP + FP} \text{ and } Recall = \frac{TP}{P}$$

The data mining service of SSDT does not pre-calculate these indicators, but we did ourselves. Here are the results that we got for each of the two classifiers:

	<i>Precision</i>	<i>Recall</i>	<i>F1 Score</i>
Churn Decision Trees	71.9%	87.2%	78.8%
Churn Naïve Bayes	72.5%	75.6%	74.0%

Fig. 10. The values of Precision, Recall and F1 Score

Since the *Recall* (TP rate) measure is the equivalent of the predictive accuracy (but filtered to the positive cases), we can compare the two values. This leads to the observation that the Decision Tree model was more successful in correctly predicting the positive cases (IsChurn = 1) than the negative cases (87.2% vs. 75.7%); also the Naïve Bayes classifier did a slightly better job at predicting the positive value (75.6% vs. 72.5%).

We conclude observing that no matter the evaluation criteria that we may choose, the Microsoft Decision Tree model was the one performing better and we can further use this model for making predictions on the data where we really do not know if the respective customers are to renew their software licenses, or are to become churn customers.

8 Conclusions

This paper wanted to show, in both theory and practice, how decent predictions can be made starting from relevant data structured through a commercial relational database system, like Microsoft SQL Server 2012.

Our approach was to deepen the understanding of the base concepts through direct examples on the source data that are modelled by the classification algorithms behind the curtain. Beside this, the data themselves have been explored, using comments and short analysis on the models' output. This facilitated the understanding of the information carried out by the data, which is often at least as useful as the ability to come up with good data predictions.

The development of the relational database management systems, such as Microsoft SQL

Server or Oracle, led not only to the addition of the data mining technology to the software package, but also to the simplification and automation of processes like setting-up predictive models. However, the data mining facilities provided by the software should be joined, on the user side, by things like: a proper understanding of the raw data and the context they are collected (eg. the business conditions) and the relevancy of the predictive activities in relation with the objectives and the business issues.

The predictions presented in the current paper were mainly orientative, as the main purpose was to explain and show how the classification models work using a relational database management system. The quality of the predictions is likely to be improved using other approaches like:

- additional calibration of the parameters used of modeling the training data set;
- training new, recent and more powerful classification algorithms like Support Vector Machine, Random Forrest etc;
- considering the information related to the cost of the *treatment* applied to the group of customers with a high churn risk, so that the most optimal and cost efficient solution is chosen. Considering this aspect, if the financial situation requires, a higher level of churn or a higher level of the false positive rate can be preferred, if this has a positive effect on the profitability of the company.

Recent approaches in managing churn are not only focused on maximizing the correct classification of churn or not churn customers. “Profit from targeting a customer depends on not only a customer’s propensity to churn, but also on her spend or value, her probability of responding to retention offers, as well as the cost of these offers” [11]. This opens further perspectives, which are even more rooted in the overall conditions of running a successful business.

References

- [1] T. H. Davenport and D.J. Patil, “Data Scientist: The Sexiest Job of the 21st Century”, 2012: <http://hbr.org/2012/10/data-scientist-the-sexiest-job-of-the-21st-century/>
- [2] D. Wilson and J. Hatzius, G. Sachs, “The World Cup and Economics 2014”, 2014: <http://www.goldmansachs.com/our-thinking/outlook/world-cup-and-economics-2014-folder/world-cup-economics-report.pdf>
- [3] K. P. Erb, “Warren Buffet Offers \$1 Billion For Perfect March Madness Bracket”, 2014: <http://www.forbes.com/sites/kellyphillips/erb/2014/01/21/warren-buffett-offers-1-billion-for-perfect-march-madness-bracket/>
- [4] The Chartered Institute of Marketing, “Cost of customer acquisition vs customer retention”, 2010: <http://www.camfoundation.com/PDF/Cost-of-customer-acquisition-vs-customer-retention.pdf>
- [5] J. MacLennan, Z.H. Tang, B. Crivăț, “Data Mining with Microsoft SQL Server 2008”, Wiley Publishing
- [6] Tan, Steinbach, Kumar, “Introduction to Data Mining”, Pearson New International Edition, 2014
- [7] M. Bramer, “Principles of Data Mining”, second edition, London 2013
- [8] G. James, D. Witten, Trevor Hastie, Robert Tibshirani, “An Introduction to Statistical Learning with Applications in R”, Springer Science-Business Media, New York 2013
- [9] Classification Matrix (Analysis Services – Data Mining): [http://msdn.microsoft.com/en-us/library/ms174811\(v=sql.110\).aspx](http://msdn.microsoft.com/en-us/library/ms174811(v=sql.110).aspx)
- [10] Precision and Recall: http://en.wikipedia.org/wiki/Precision_and_recall
- [11] A. Lemmens, S. Gupta, “Managing Churn to Maximize Profits”, Harvard Business School, 2013: http://www.hbs.edu/faculty/Publication%20Files/14-020_3553a2f4-8c7b-44e6-

- 9711-f75dd56f624e.pdf
- [12] Gh. Ruxanda, "Data Mining" printed course for the "Business support databases" master degree, Bucharest, 2010
- [13] SQL Server Data Mining Tutorial: <http://technet.microsoft.com/en-us/library/ms167167.aspx>



Catalin I.V. CIMPOERU obtained his Master degree from The Faculty of Cybernetics, Statistics and Economic Informatics, at the Bucharest University of Economic Studies. He also holds a Bachelor degree in Letters and Communication. Currently, he works as a data scientist. Domains of interest: Data Mining, Business Intelligence, Python programming, Web Analytics.



Anca Ioana ANDREESCU is senior lecturer in Economic Informatics Department, Academy of Economic Studies of Bucharest. She published over 20 articles in journals and magazines in computer science, informatics and business management fields, over 20 papers presented at national and international conferences, symposiums and workshops and she was member in over twelve research projects. In January 2009, she finished the doctoral stage, the title of her PhD thesis being: The Development of Software Systems for iBusiness Management. She is the author of one book and she is coauthor of four books. Her interest domains related to computer science are: business rules approaches, requirements engineering and software development methodologies.