



An automatic predictive datamining tool

Data Preparation – Propensity to Buy v1.05

Januray 2011



Data preparation - Introduction

If you are using “*The Intelligent Mining Machine*” (TIMi) inside your datamining projects, the data preparation step is the most time-consuming step in the whole project.

In opposition to other classical datamining softwares, **TIMi** has no limitations to the number of columns or rows inside the learning/creation dataset that it can process:

- **TIMi** is able to process dataset containing millions of rows in a few minutes. Sampling (and thus losing information) is (nearly) no longer required with **TIMi**.
- **TIMi** is able to easily process datasets with more than fifty thousands columns. Classical datamining softwares reach their limits around 300 columns (even if they don’t “crash” when reaching this limit, the accuracy of the predictive models produced by other softwares is degrading when you increase the number of columns above 300 inside your dataset). This also means that you **do not have** to eliminate arbitrarily (based on some “heuristic”) some variables of your dataset in order to stay below the column limitation of the other datamining softwares.

When using **TIMi**, it is strongly suggested to accumulate the highest number of information (rows and columns) about the process to predict: don’t reduce arbitraly the number of rows or columns: always keep the “full dataset” (even if the target size is less than one percent). **TIMi** will use this extra information (that is not available to other “limited” datamining softwares) to produce great predictive models that outperform any predictive model constructed with any other datamining software.

Of course, you should not forget to “let aside” a TEST set that will be used to really assert the quality of the delivered predictive models (to be able to compare in an objective way the models constructed with different predictive datamining tools). See this web page that explains the importance of the TEST set:

http://www.business-insight.com/html/intelligence/bi_test_dataset.html

To create the TEST set, it is best to use stratification techniques (It is suggested to stratify on the variable to predict). You can use the double-stratification algorithm available inside the “Convert, extract & sample” **TIMi** module to produce the TEST set.

This document describes very briefly the data preparation steps required for a preliminary analysis of your data. This preliminary analysis is just to assess if your databases contains enough useful, exploitable, information about your customers that are actionable from a business perspective.

This document is also useful when you want to setup very rapidly a benchmark to compare the performances of different datamining tools. In this case, the dataset given to TIMi should follow the recommendation and guidelines contained within the “Creation dataset file format” section.



Set up model design

To build a predictive model you need a database. This database must be presented to TIMi as a large table (the optimal format is as a compressed .csv flat file). The objective of this document is to describe how to build this table.

Let's assume the following situation: You have a database of all your customers. You want to predict for each customer the probability that he will buy the product X (this is a classical "propensity-to-buy" problem, but the same principles applies to any predictive modelling exercise: Cross-Selling, Up-Selling, Churn prevention, etc. For the simplicity of the exposition, we will now assume that we are in a classical "propensity-to-buy" setting). You will create a table (typically: a flat file). Each line of the table represents one customer. The columns of the table are information about the customer: its age, gender, financial income,... One column of the table has a special meaning: it's the **target**: It's the column that you want to predict. The **target** column contains two values (no missing values are allowed for the **target** column):

- value '0': the customer did NOT buy the product X.
- value '1': the customer bought the product X.

Let's assume that you have:

<u>Database SnapShot at the end of January</u>						
id	name	age	income	gender	date of purchase	target
1	frank	32	4500	M		0
2	sabrina	25	4000	F		0
3	max	33	2000	M		0

<u>Current Database at the end of April</u>						
id	name	age	income	gender	date of purchase	target
1	frank	32	2000	M		0
2	sabrina	25	4000	F	2-14-07	1
3	max	33	2000	M		0

Please note that Sabrina decided to purchase the product X at some point in time after the "End-of-January".



The table that you should provide to TIMi to build a predictive model is the following (this table is called the **model creation/learning dataset**):

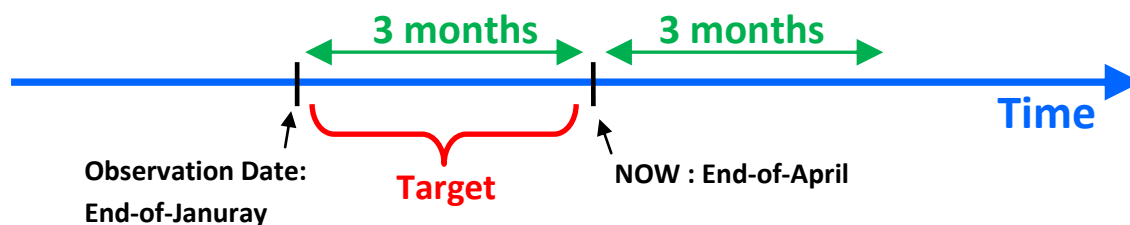
The target is based on what happened after January: It is simply the target column extracted from the most recent update of your database

Snapshot from end-of-January

id	name	age	income	gender	date of purchase	target
1	frank	32	4500	M		0
2	sabrina	25	4000	F		1
3	max	33	2000	M		0

The table (i.e. the **creation/learning dataset**) above is the best option: TIMi will construct a new predictive model m that will use the customer profiles as they appeared at the end-of-January to predict the **target** in {February, March, April}. The date “End-of-January” is called in technical term the “**Observation date**”: it’s the date where we “observed” the profile of the customers to construct the **learning dataset**.

Graphically, this can be illustrated in this way:



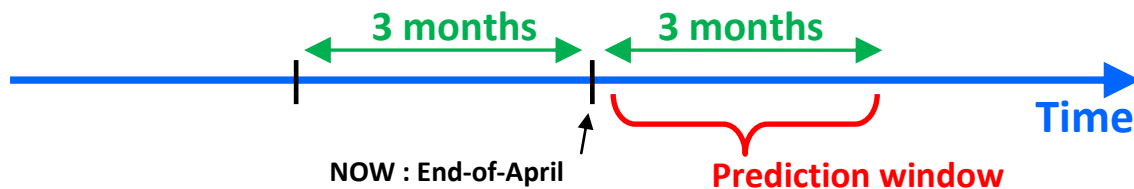
In the above example, the target is defined on a period of 3 months (i.e. it’s defined based on the period that extends between “now” and the “observation date”): in technical terms, the “**Prediction Window**” is 3 months.

More formally, we can also write the predictive model in the following way:

$$m(\text{CustomerProfile}_{\text{Time}=\text{T}_{\text{observation}}}) = \text{TargetPrediction}_{\text{Time} \in [\text{T}_{\text{observation}} \text{ T}_{\text{observation}+3\text{months}}]}$$



Once you have constructed your predictive model m , you can apply it on the most up-to-date version of your customer database (from End-Of-April) to predict who are the customers that will buy the product X in {March, June, July}. Graphically, this can be illustrated in this way:



To build such a **creation/learning dataset**, you need a database structure that supports time logging (or you need to create several “snapshots” of your customer database at different point in time) because you are mixing columns from End-Of-January (containing the profile of your customers) with one column from End-Of-March (containing the target column). Such complex database structure is not always available. As an alternative (but less accurate) solution, you can also use the following **creation/learning dataset**:

Most Up-ToDate database from End-of-March

id	name	age	income	gender	date of purchase	target
1	frank	32	2000	M		0
2	sabrina	25	4000	F	2-14-07	1
3	max	33	2000	M		0

You should try to avoid this second approach because it has many flaws. Unfortunately, inside the industry, this approach is used 99% of the time because it does not require time logging (nor snapshots).

The first major drawback of this second approach is: when you use TIMi (or any other predictive analytic tool) to construct a predictive model on this **creation dataset**: you will obtain the following predictive model:

If (“date of purchase” is missing) then target=0 else target=1

The above predictive model has not identified the **cause** of the purchase of the product X but the **consequence**. The column “date of purchase” does not contain any information that could be used to predict if a customer will purchase the product X (because this column is initialized **after** a purchase). When you use TIMi to construct a predictive model, you must tell to TIMi to ignore all the “consequences” columns. The visual interface to TIMi allows you to ignore a column very easily with only one mouse-click. 99% of the modelization time is usually spent finding these bad “consequences” columns. Usually, you don’t know them “in advance”, before starting the



modelization process. TIMi allows you to find these “consequences” columns very easily (because of the very concise and intuitive reports auto-generated by TIMi). At the end, your **creation dataset** will be:

Current database from beginning of March
minus all “consequence” columns.

id	name	age	income	gender	target
1	frank	32	2000	M	0
2	sabrina	25	4000	F	1
3	max	33	2000	M	0

To summarize, the **creation/learning dataset** to prepare before using TIMi can either be:

- **Approach 1 (this is the best approach):** A mix of different time period:

Snapshot from end of January

the **target** column (extracted from the most up-to-date customer database)

id	name	age	income	gender	date of purchase	target
1	frank	32	4500	M		0
2	sabrina	25	4000	F		1
3	max	33	2000	M		0

- **Approach 2 :** The database in its current state **and** a list of “consequence” columns to ignore

Most Up-To-Date customer database

id	name	age	income	gender	date of purchase	target
1	frank	32	2000	M		0
2	sabrina	25	4000	F	2-14-07	1
3	max	33	2000	M		0

The list of “consequence” columns to ignore is: *date of purchase*



NOTE 1: Approach 1 systematically delivers higher accuracy models (higher ROI) than Approach 2.

If you analyze closely the *creation/learning dataset* used in approach 2, you will arrive after some thoughts, to either one of these two predictive models:

If ("age" < 30) then target=1 else target=0
 or
If ("income" > 3000) then target=1 else target=0

If you look at the *creation/learning dataset* used in approach 1, you will notice that only the first model (out of the above 2 model) (the one based on the "age") is correct.

When your customer database is evolving rapidly, you cannot make the assumption that the *approximate* customer profiles available for "approach 2" are similar to the exact customer profiles available in "approach 1". When this assumption breaks down, the predictive model generated using the "approach 2" will have poor accuracy (because these are based on wrong profiling information).

If you are able to use "approach 1" because you possess several "snapshots" of your customer database at different point in time, then you also have another great opportunity to still increase the accuracy of your predictive models (and remember: For marketing campaigns, "**Accuracy=ROI**"!).

How so? Based on the different customer database snapshots, you can create additional columns/variables to include inside your *creation/learning dataset* that are very interesting from a prediction point-of-view (these might actually be the most interesting columns of all) (i.e. these columns will increase the accuracy of the predictive models).

These variables are typically: the increase/decrease of activity of your customers from one snapshot to the other ("activity" in terms of number/amount of transactions in a specific domain, frequency of usage of a specific service, etc.). The Anatella ETL includes special data-transformation operators that help you create easily such new variables. We usually call such variable "**delta variables**".

Using such a technique, it's very common to start from a simple customer database containing around 200 columns and end-up with a *creation/learning dataset* with more than 20.000 columns (that are "time-derivated").

TIMi is the only predictive datamining tool that is able to analyze datasets with more than a few thousands columns. Anatella is the only ETL tool that is able to manipulate datasets containing more than a few thousands columns.

NOTE 2: Prediction window and Target size.

In the above example, the "*prediction window*" is 3 months (it's a common value found in the industry). What happens if nobody decided to buy the product X during these 3 months? In such situation, the "target" will be "zero" for all the rows of your dataset and it won't be possible to create any predictive model. You will be forced to define a longer "*prediction window*" (to avoid the "null" target problem).



Let's assume that your "prediction window" is one year. This also means that your "*observation date*" is one year in the past and that your **creation/learning dataset** is based on some data that is already *one year old*. In one year, the whole economic situation of the market might have changed and your predictive model won't be adapted to this new situation. This is not good. Obviously, you want your "prediction window" to be as short as possible (to be able to capture the latest trend of your market) but still long enough to have a "reasonable target size" (to avoid the "null" or the "nearly-null" target problem).

This is exactly where TIMi has an important added value compared to other datamining tool: TIMi is able to work with extremely small target size. With TIMi, the target size can be as small as half a percent of your database. Typically, other datamining tools are not working properly when the target size is below 5% of the whole database.



Creation/Learning dataset file format

TIMi can read datasets from many data sources: simple "txt" or ".csv" flat files, SAS files, ACCESS files, ODBC & OleDb links to any databases (Teradata, Oracle, SQLServer, etc.). **But** for a first "quick benchmark", it's suggest to store the **creation dataset** inside a simple "txt" or ".csv" flat file (in order to prevent any inter-operability errors).

The "txt" or ".csv" flat file should follow the following format:

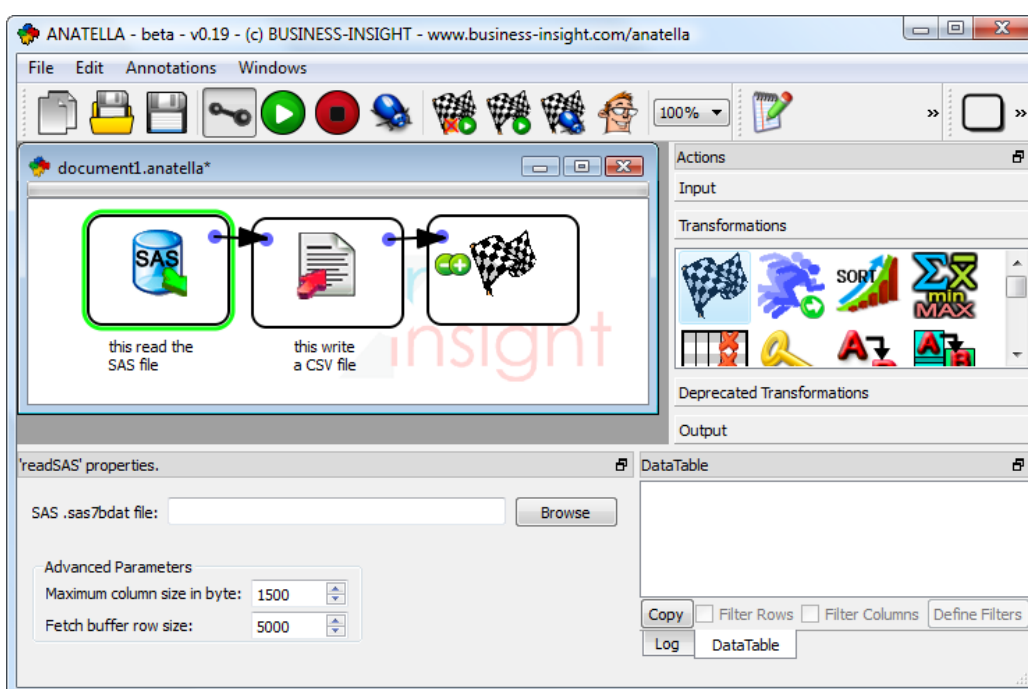
- The **creation dataset** is a "txt" or ".csv" file where the separator is a dot-comma '.;'. The first line of the file must contain the column names.

WARNING: "txt" files exported from SAS have a size limitation: one line cannot exceed 65535 characters. If you encounter this bug in SAS, there are several solutions:

- Use the SAS plug-in for **TIMi** that allows you to read directly natively .sas7bdat SAS files (this is however very slow because the SAS driver is very slow).

To be able to read SAS files on any PC (even PC without any SAS installation), you should first install these OleDb drivers (that are originating from the SAS website): <http://www.business-insight.com/downloads/sasoledb.zip>

- Convert the .sas7bdat SAS file to a simple "txt" file using Anatella:
Use the following Anatella-data-transformation-script:




To be able to read SAS files on any PC (even PC without any SAS installation), you should first install these Oledb drivers (that are originating from the SAS website): <http://www.business-insight.com/downloads/sasoledb.zip>

- Column names must be unique.
WARNING: TIMI is case IN-SENSITIVE (as is SQL)
- Column names are NOT within quotes.
- The data in the columns are NOT within quotes (never).
- The field separator character (here ‘;’) is not allowed (neither in the data, neither in the column names).
- The **creation dataset** contains one unique primary key.
- The decimal character is a dot and not a comma (Standard English notation or Scientific notation for numbers).
- If The **target** column (the column to predict) is:
 - Binary: then it must contains only ‘0’ and ‘1’ values (and the “one’s” are the value to predict and must be the **minority case**).
 - Continuous: then it should not contain any “missing value”.
- Missing values must always be encoded as empty values (“”).
- OPTIONAL: The **creation dataset** should not contain any “consequence columns”. If the dataset nevertheless contains some “consequence columns”, it’s good to know their name in advance. However, you can always use **TIMi** to find all the “consequence columns” easily.
- OPTIONAL: the flat file can be compressed in RAR (.rar), GZip(.gz), Winzip(.zip)
- OPTIONAL: all the columns that represent a “True/False” information may contain only two different value: ‘0’ (for false) or ‘1’ (for true) or are empty (“”) if the value is missing.




- **OPTIONAL:** all the columns that represent either:
 - a number
 - an information that can be ordered... should be encoded as pure number. For example:

number of cats	
missing	
no cat	
one cat	
2 cats	
3 or more	



number of cats	
	0
	1
	2
	3

Social class	
missing	
poor	
middle	
rich	



social class	
	0
	1
	2

