Business insight

TIMi suite

integrated data mining solutions

An automated predictive datamining tool

Target Bias and Other Modeling Errors v1.02

December 2014

TIMi suite
integrated data mining solutions

This document contains different examples of projects where the predictive models is unusable.

Example 1: First Time Modeling

A basic assumption, when you create predictive models for cross-selling purposes, is: "*When you create an appetency model for a product X, the people flagged as targets are interested by the product X*". This assumption might be violated for several reasons. One of the most frequent reason is "*Stubborn marketing campaigns*". Let's assume that the probability to buy the product X is:

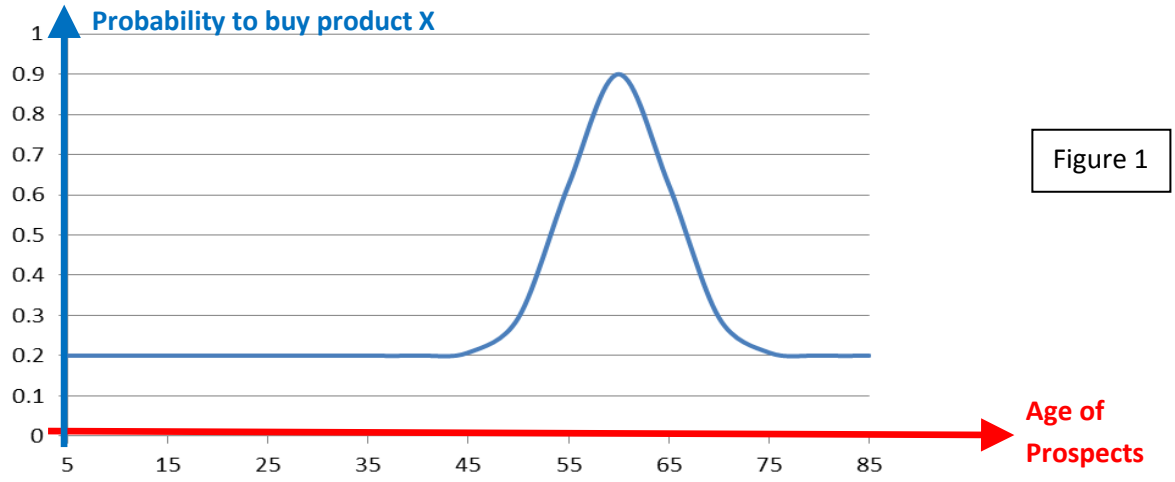


Figure 1

Let's assume that we are in one of the following situations:

- You ran several **random** marketing campaigns for product X.
- You did not run any marketing campaigns for product X at all: All the sales are due to spontaneous purchases from customers.

In such situations, you'll see in your customer database:

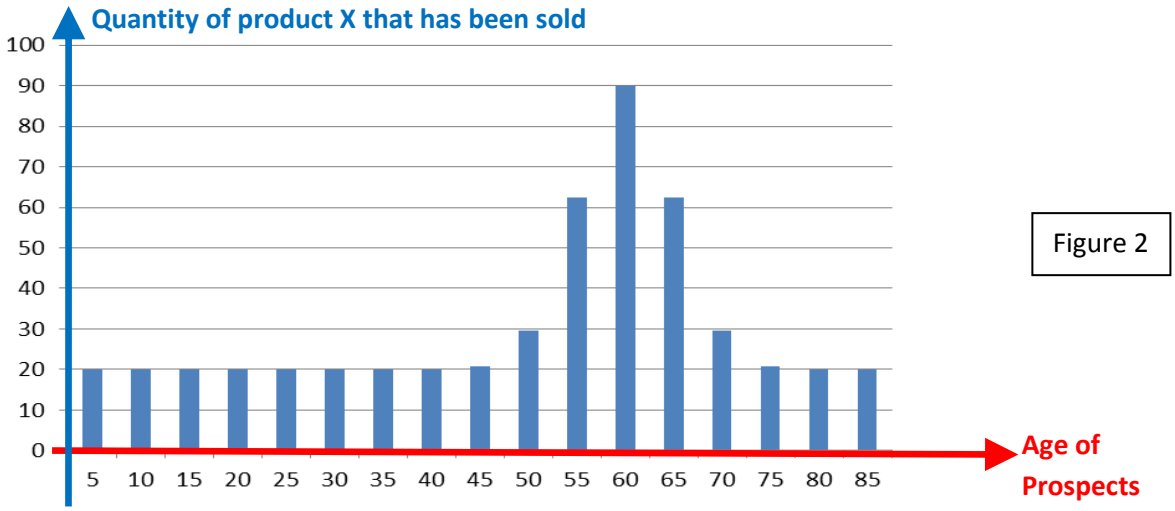


Figure 2

Let's now assume that your marketing team decided to run **10 times** more marketing campaigns for the product X on the prospect in the age-range 20 to 30. The prospects in this age-range will be 10 times more solicited. We'll get:

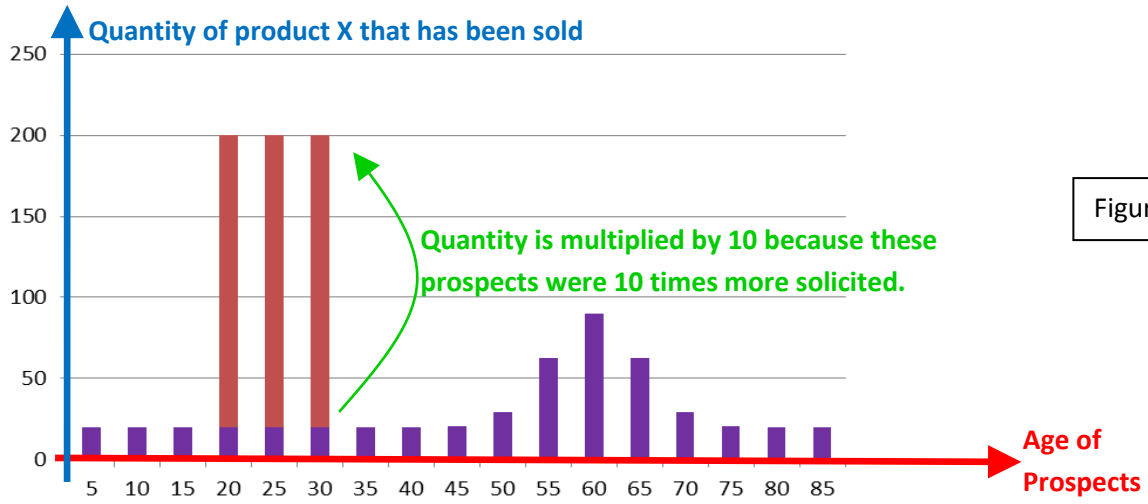


Figure 3

Now, if you try to build a predictive model based on the above "Skewed" Target (the exact term is "Biased Target"), you'll get a model that will tell you to run your marketing campaigns on the prospects inside the "*20 to 30 age-range*" (because there are a lot more Target in this age-range).

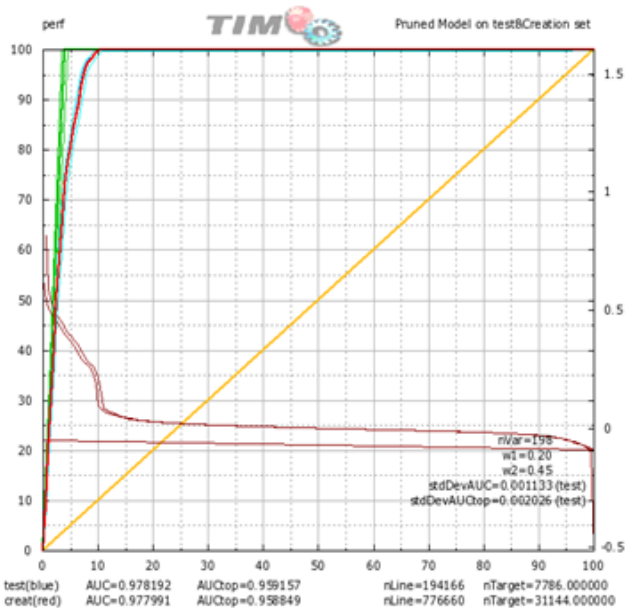
These types of "stubborn marketing campaigns" that systematically focus on the same population-segment are very common in companies running marketing campaigns based on "*common-sense business-rules*". Unfortunately, such behaviour very often creates strongly "Biased Target" that prevents the direct & easy usage of predictive modelling. Hopefully, there exist some solutions that are able to overcome these initial difficulties (see below).

If you act blindly (i.e. without detecting the "Biased Target" issue), you'll acquire even more Targets inside the "*20 to 30 age-range*" (because **all** your campaigns will focus on this age-range), still increasing the Target bias and degrading even further your models. This is obviously very bad.

Another way to see this problem can be named the "self-fulfilling prophecy". More precisely: "*If I am selling icecream only to children that are between 10 to 12 years old, I can predict that only children between 10 to 12 years old will buy icecream in the future*".

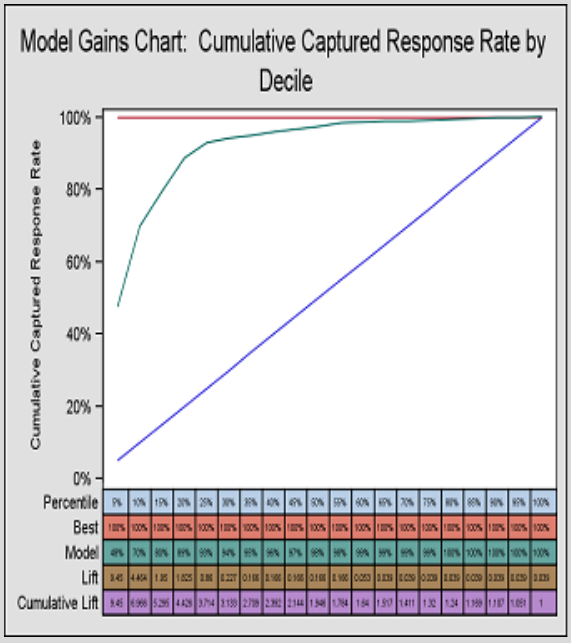
The "target bias" problem is usually quite easy to detect when using TIMi. There is a high probability of "target bias" problem when:

1. The lift-curve is abnormally high: For example, this is very suspicious when you get such lift-curve:



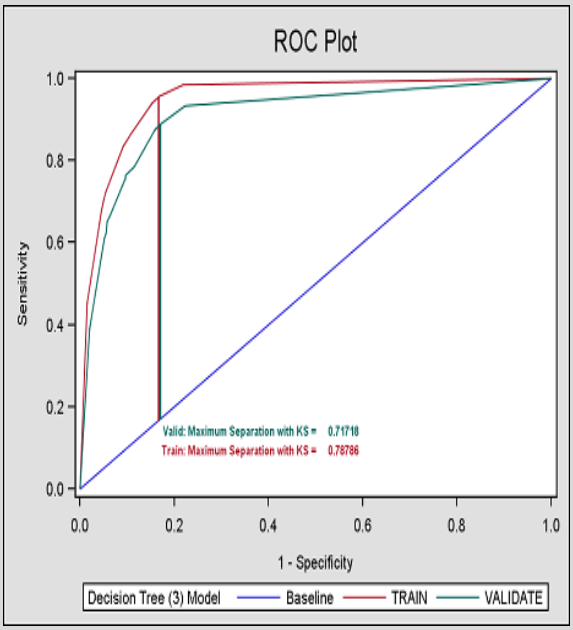
The above TIMi-lift-curve illustrates a case where there is a strong Target Bias: This is clearly visible. Please note that, strangely, on this case, the lift curves obtained with SAS or SPSS still "*looks ok*": They won't allow you to detect the "target bias" problem.

The lift curve obtained with SAS is:



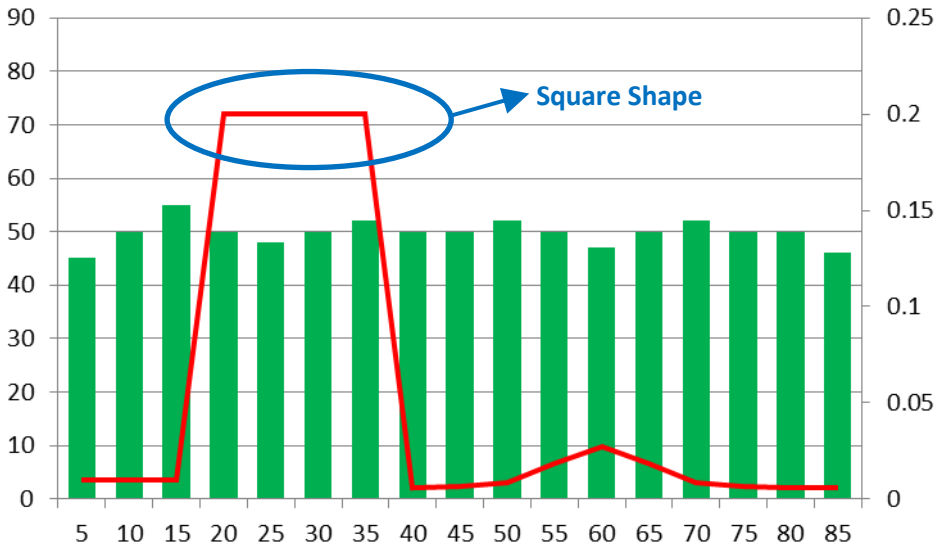
SAS

The lift curve obtained with IBM MODELER is:



IBM MODELER

2. A "Target Bias" problem is very likely when you see inside the Analyst report generated by TIMi a "square shape" like this:



The above "Square Shape" originates from the fact that many marketing campaigns were deployed on the population segment composed by the individuals inside the 20-35 age-range.

Junior data miners are very often using decision trees to erroneously run several successive marketing campaigns while having a very strong "Target Bias".

When this happens, you'll see the above "Square Shape" for all the variables included inside the decision tree. The "Square Shape" originates from the "*strong on/off cuts*" made by the decision tree.

There are several solutions to this "Biased Target" problem:

Solution 1.

Let's assume that the individuals included inside the previous campaigns have been selected using some rule based on the variables X1, X2, X3 (e.g. X1, X2, X3 were included inside the decision tree used inside the previous campaigns).

We will create a predictive model based on a learning dataset where we removed the variables X1, X2, X3 (because if you let these variables in, the modeling tool will directly try to use them to reproduce the past selections and we want to avoid that).

Unfortunately, removing the variables X1, X2, X3 from the learning dataset also very often means that you removed the best variable from your model, thus obtaining a very poor lift. If you obtain a very inaccurate model (i.e. a poor lift), you should try the "solution 2".

Removing the variables X1, X2, X3 does not guarantee you that you won't have any selection bias anymore (there can still be inside the learning dataset other variables strongly correlated with X1, X2 or X3). Thus, this first solution is not without danger and you might better use the "solution 2", for more security.

Solution 2.

We want to use as Targets the individuals who responded positively to our campaign. To obtain this target, you can run a small random campaign (on a small part of your population). You will then create a model using:

- As learning dataset: the people included in our small random campaign
- As target: the people that purchased the product X during our small random campaign.

Random campaigns are costly because of their low conversion rates (i.e. try to make the random campaign as small as possible). So, instead of running a completely random campaign, you can run a campaign that is slightly more focused on a population with a higher probability of purchase. To obtain this "slightly better" population, follow this procedure:
* Let's assume that the individuals included inside the previous campaigns have been selected using some rule based on the variables X1, X2, X3.
* Remove from the learning dataset the variables X1, X2, X3 and create a model on the remaining variables.

* Contact one-out-of-10 individuals inside the top 40% of the selection obtained using the model created at the previous step: We don't want to focus "too much" on the results of this first model because it could bias the results.

The people that bought the product X during our random campaign fall in two categories:

- **Category 1:** The people that responded positively to the campaign
- **Category 2:** The people that would anyway buy the product X, even if they would not have been solicited.

A really good target is only composed of the people inside the category 1 (and not from the people inside the category 2), but unfortunately we only have a "target group" that is a mix of the 2 categories. To be able to focus on the targets inside the Category 1 only, use "up-lift" modelling.

TIMi will soon include a very accurate up-lift modeling procedure.

In the interval, use the following procedure to do "up-lift":
* create a model "***M_spontaneous***" that computes the probability of spontaneous conversion. To create this model, use as learning dataset all the individuals **outside** the selection used in the previous campaigns.
* create a model "***M_solicited+spontaneous***" that computes the probability of purchase. To create this model, use as learning dataset all the individuals **inside** the selection used in the previous campaigns.
* For all individuals in your population, compute the increase "***Inc***" of purchase probability that originates from the solicitations. We have:

Inc = M_solicited+spontaneous - M_spontaneous

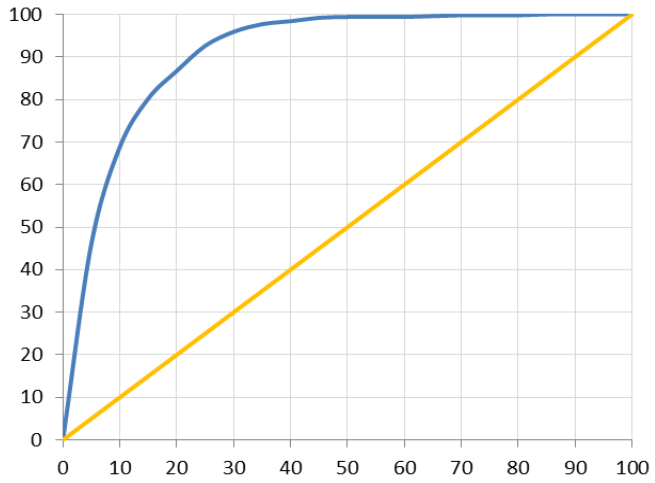
* Sort all the individuals from the highest "***Inc***" to the lowest "***Inc***" and contact the top 10%.

In general, you don't want to use the model "***M_spontaneous***" (for more information on how to create this model, see the note above) to run a direct marketing campaign because you'll lose money by contacting the individuals that will, anyway, spontaneously purchase your product (furthermore, there is the very real danger of having a **negative** "up-lift" if you use this model). You can still use the model "***M_spontaneous***" in some specific situations:
* When a client arrives at the desk in one of your selling point or agency: You can talk about the product X, just to be sure the client won't forget to purchase it (i.e. we are here talking about a "passive" solicitation outside a direct marketing campaign).
* When our product X is in direct competition with a product from a competitor, you want to be sure that the client buy your product (and not the one from the competitor).

Example 2: Dataset built from Questionnaire

We constructed a predictive model for the nestle company and we noticed a strange thing inside the "analysis report": Here are all the variables used by the model:

```
Variables Ranking
-----------------
  1 MEDIA.MAGAZINE SUBSCRIPTIONS.MAGAZINE  SCIENTIFIQUE                    100.00
  2 TRAVEL.TYPE OF HOLIDAY.WEEK-END                                        39.70
  3 MEDIA.NEWSPAPER SUBSCRIPTIONS.LE MATIN                                 34.91
  4 MEDIA.NEWSPAPER SUBSCRIPTIONS.HET LAATSTE NIEUWS                       34.00
  5 SHOPPING.MAIL ORDER FREQUENCY.NEVER                                    32.47
  6 SHOPPING.STORE.GAMMA / GB BRICO                                        29.19
  7 DRINKS.ALCOHOL.COGNAC & WHISKEY                                        27.90
  8 MEDIA.NEWSPAPER SUBSCRIPTIONS.LE SOIR                                  27.36
  9 NOMINAL.Discount                                                       26.27
 10 GENERAL.Age of Oldest Female                                           25.99
 11 MONEY.BANK.CITIBANK                                                    23.52
 12 TRAVEL.DESTINATION.ASIA                                                23.35
 13 SHOPPING.DISTANCE SELLING.COMPUTER & TELCO                             17.69
 14 FOOD & HEALTH.HEALTH.SMOKER                                            17.16
 15 MONEY.INFORMATION.PENSION PLAN                                         15.28
 16 MEDIA.MAGAZINES.AUTRES MAGAZINES FEMININS                              14.52
 17 TRAVEL.YEARLY BUDGET PER PERSON.> 75.000 BEF                           14.46
 18 MEDIA.MAGAZINES.DAG ALLEMAAL                                           14.11
 19 HOUSE.OWNERSHIP.HOME OWNER                                             13.99
```




```
Stats for Variables in Model
----------------------------
```

	Count Univers	Count Client	% Univers	% Client	Index	Z-Score
MEDIA.MAGAZINE SUBSCRIPTIONS.MAGAZINE SCIENTIFIQUE						
FALSE	262903	5673	75.1	98.0	130.48	114.28
TRUE	3530	90	1.0	1.6	154.17	0.42
UNKNOWN	83567	25	23.9	0.4	1.81	-18.06

Note: the *"index"* column is computed this way: $index = \frac{\text{density of target inside modality}}{\text{density of target inside full set}}$

Description of the problem:

The content of the learning dataset is collected through successive questionnaires. The population that will receive a question or be part of a questionnaire is selected by the companies that paid to have "their" questions in it. Hence there is a strong selection bias. The "*MEDIA.MAGAZINE*

SUBSCRIPTIONS.MAGAZINE SCIENTIFIQUE" variable says: if you never received the questionnaire before (UNKNOWN) then you have fewer chances to be in the target. However, if you received it, reading scientific magazine or not does not change significantly the chance of being in the target.

Now the question is: is this variable telling us something interesting? The answer is "*it depends*". **This variable shows that the past selection has something to say about the chance of being a target**. We have two possible relationships:

a) either the past selection is good with respect to the actual target and hence, the variable is just telling us "select using the same bias", which might be a very valuable information
b) either the target density just reflects the fact that more actions have been done on past selection and in that case, maybe we do not want to make the hypothesis that past selection was indeed good and is still the best (This is the most likely hypothesis).

Obviously, the semantic of this variable (scientific magazine) is not at all related to its usage in the predictive model. It is simply the best variable that gives some information on past selection.

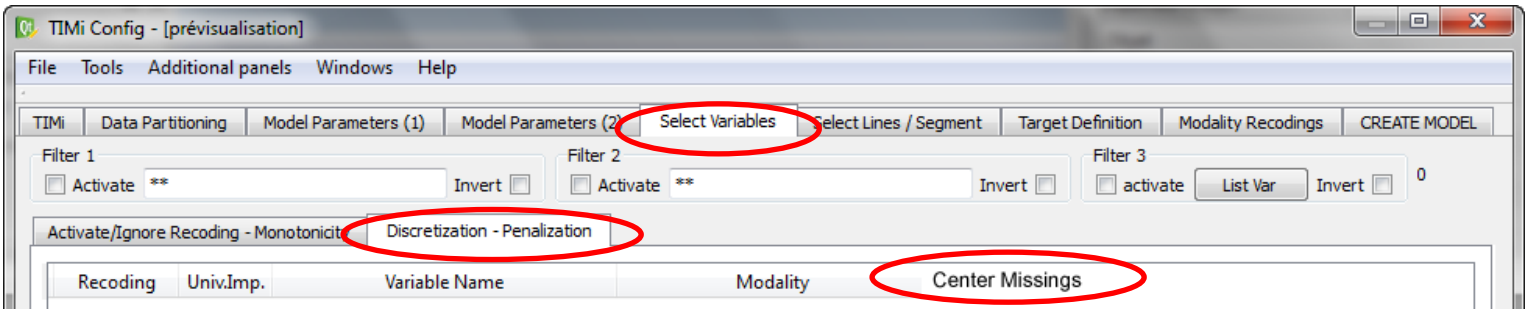
This behavior is quite frequent when we try to use the information contained inside the "missing" modality. (i.e. when the "*Center Missing*" option is set to "false"). Using the information contained inside the "missing" modality is sometime a good idea, sometime a bad idea: It is:

- A good idea:
  When "*missing*" means "*Not applicable*", then, in most cases, we want to use the information contained inside the missing modality (i.e. we want to set "*Center Missing*"="False"): it is the best that we can do. For example when "*number-of-cash-deposit*" = "*missing*" because the client has no saving account.
- A bad idea:
  When the missing means "*we don't know because we did not have a chance to learn it*", then the "*missing*" modality will typically say something on the way that we collected the data, and won't say anything about the preferences of the current profile or customer. In that later case, missing treated as a real missing is probably best (i.e. you need to set "*Center Missing*"="True"; that is not the default value)

This is why in the case of the current dataset (that has been collected by questionnaires) the "*Center Missing*"="False" is causing troubles.

So the final answer is: If we have good business reason to suspect that past selections are reliable for this campaign, then yes, we could leave the model as it is (this is unlikely). If we do not trust this assumption, we need to recode missing as missing (i.e. set "*Center Missing*"="True") and rebuilt the model.

The "*Center Missing*" option is available inside the TIMi Interface here:



Example 3: Too Generic

Let's assume that you are working for the post. You are building some predictive models that predicts if a given company will purchase a specific high-end from the post product (i.e. a large stamping machine). These models are typically used for a "next-to-buy" application. Here is one of the models that you obtained:

TIMi - The Intelligent Mining Machine Analyst report v12.11

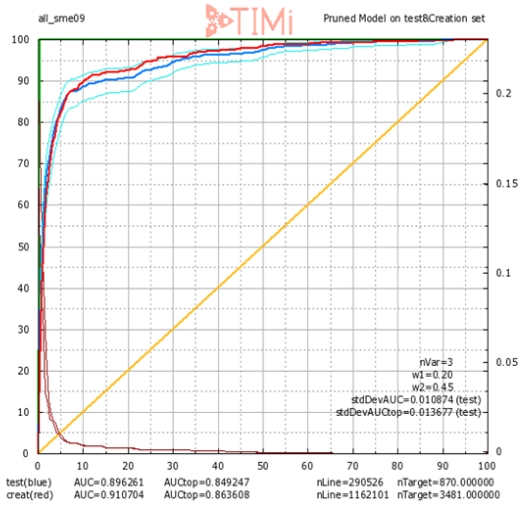
Target: Column: Segment

Discriminative Variables ranking

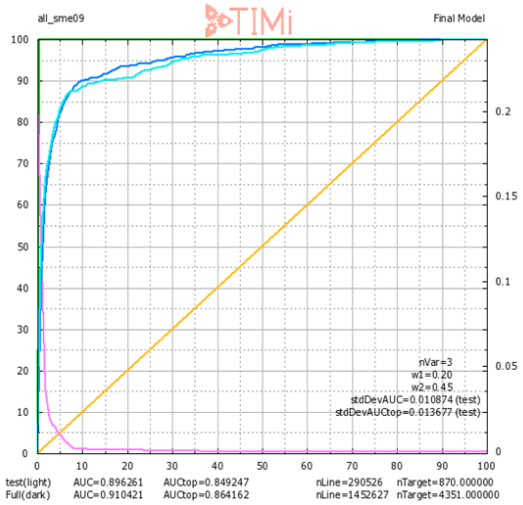
		Importance [%]	Univ.Import. [%]	correlation with Target[%]	Weight In Model[%]	Min	INDEX [%] Max	Highest Positive Discrim.
1	Total Number of Employee	24.5	87.5	3.3	75.2	8.1	3930.8	22<v<=9999
2	Already has an Automated Stamping Machine	5.9	45.5	25.6	100.0	53.9	6098.4	true
3	Code Activity Sector	4.3	62.3	3.3	9.7	6.1	552.4	Juridic Activities..

Lifts

Pruned Model on Test Set



Final Model



The 3 best variables of the above model are:

1. Total number of employee
2. Already has an Automated Stamping Machine
3. Code Activity Sector (best modality = Juridic Activities).

These 3 variables are highly discriminant (i.e. the lift curve is very high!). So, it seems that you already did your job pretty well and you don't need to put any more effort into this precise model.

The problem is the following: "*All the predictive models that you created (whatever the product or the service that you used as target) are using the same three variables*". In other words, all your models "*look the same*". What does this means? It means that TIMi is actually not creating a model to "*compute the probability to buy a specific product P1 from the Post*" (which is a difficult task) but it's rather "*computing the probability to buy a product from the Post (whatever the product P1,P2,P3,etc.)*" (which is a really simpler task). The TIMi model

still does what it was asked for because "*it actually still detects the companies interested in the product P1*". Unfortunately, the "*TIMi model also detects the companies interested in the product P2, P3, etc*". This phenomenon happens quite often when:

1. it's a lot easier to detect the companies buying any product P1,P2,P3,etc. from the post rather than a specific product P1 (because all the products P1,P2,P3,etc. are very similar with respect to the variables included inside the learning dataset).
2. The dataset does not contain some variables that can be used to easily discriminate between the different products.
3. The "pattern" of a company buying a specific product "Pi" is not really clear.

There are several solutions to this problem:

1. In the above example, we used a "one-vs-all" approach (this is the most common approach for a "next-to-buy" application). We could have used instead a "one-vs-one" approach: See section 6 inside the "TIMiAdvancedGuide.pdf".

2. To solve the above problem, the best solution might be the following:

   Let P(A), the probability of acquiring the product P1 (i.e. a stamping machine) from the post.
   Let P(B), the probability of being a customer form the post.

   When I create my model on the whole population, TIMi does not give me a model that calculates P(A) but it gives me a model that computes P(B): i.e. the probability of having at least one product from the post but not specifically the product P1 ("stamping machine").

   So, to still deliver a predictive model that is specific to the product P1 ("stamping machine"), we can create a TIMi model that calculates:
   P(A|B): the probability of acquiring P1 (of acquiring a stamping machine) knowing That we are already a customer from the Post.
   To create P(A|B), I used as "learning dataset" only customers from the post (rather than the entire population). The model P(A|B) is specific to the product P1, "stamping machine". More precisely: TIMi no longer does "fall back" on a easier concept to find (i.e. it no longer falls back on the concept "*being a customer of the post*") because every single person in my "learning dataset" is a customer from the post.

   Now, what interest us is to compute P(A) and, currently, I just have P(A|B).
   To remind you, the Bayes theorem is: P(A|B) = P(B|A) * P(A) / P(B).
   Thus, we obtain: P(A) = P(A|B) * P(B) / P(B|A).
   ...but in our case, we have: P(B|A)=1.
   …therefore: P(A) = P(A|B) * P(B)

To summarize, I must create another predictive model that properly calculates P(B) and then I score my whole population using P(A) = P(A|B) * P(B) and I take the top 5%.

At one point, one could think of scoring the entire population simply using directly the model P(A|B) only. In practice, I have observed a negative lift (i.e. a lift below the random) when I was working, on a similar problem, when using P(A|B). Therefore I think it is quite risky to score your population using P(A|B): it's better to score your population using "P(A) = P (A|B) * P(B)".

Example 4: Two steps models

When we build a "propensity to buy" model, most of the time, the product that we are selling has a fixed, constant price: e.g. it's the "subscription cost" for the service.

On some rare occasions, it can happen that the promoted product generate different revenues, depending on the customer that buys it: For example:

- We are asking for donations for a charity organisation. Each "customer" (i.e. each donator) will give a different amount (such that each customer generates a different revenue).
- We are selling shares of some mutual funds. Each "customer" can buy any amount of shares (such that each customer generates a different revenue).

When the above situation occurs, we need to give a "higher priority" for contacting the people that generate the more revenue. There are typically two different ways to do that:

- **Two-steps models.**
  A two-steps approach is the following:
  1. Create a continuous predictive model M1 that predicts the generated revenue. This is a model with a "continuous target" (and the target is the "generated revenue"). The creation dataset is small: It's only composed of the customers that bought the product (because these are the only customers for which we have the target: i.e. the generated revenue).
  2. Create a binary predictive model M2 that predicts if a prospect will buy the product or not. The creation dataset is larger since it contains the whole population.
  3. For each prospect, compute the Expected Return (ER):
     ER= "probability to buy the product, as estimated by M2" multiplied by "generated revenue, as estimated by M1"
  4. Sort all the prospects based on their ER, contact the top 5% of the prospects with the highest ER.

- **Weighted binary prediction.**
  This approach involves the following steps:
  1. Let aside a test dataset (the split between the test dataset and the creation dataset is stratified on the target).

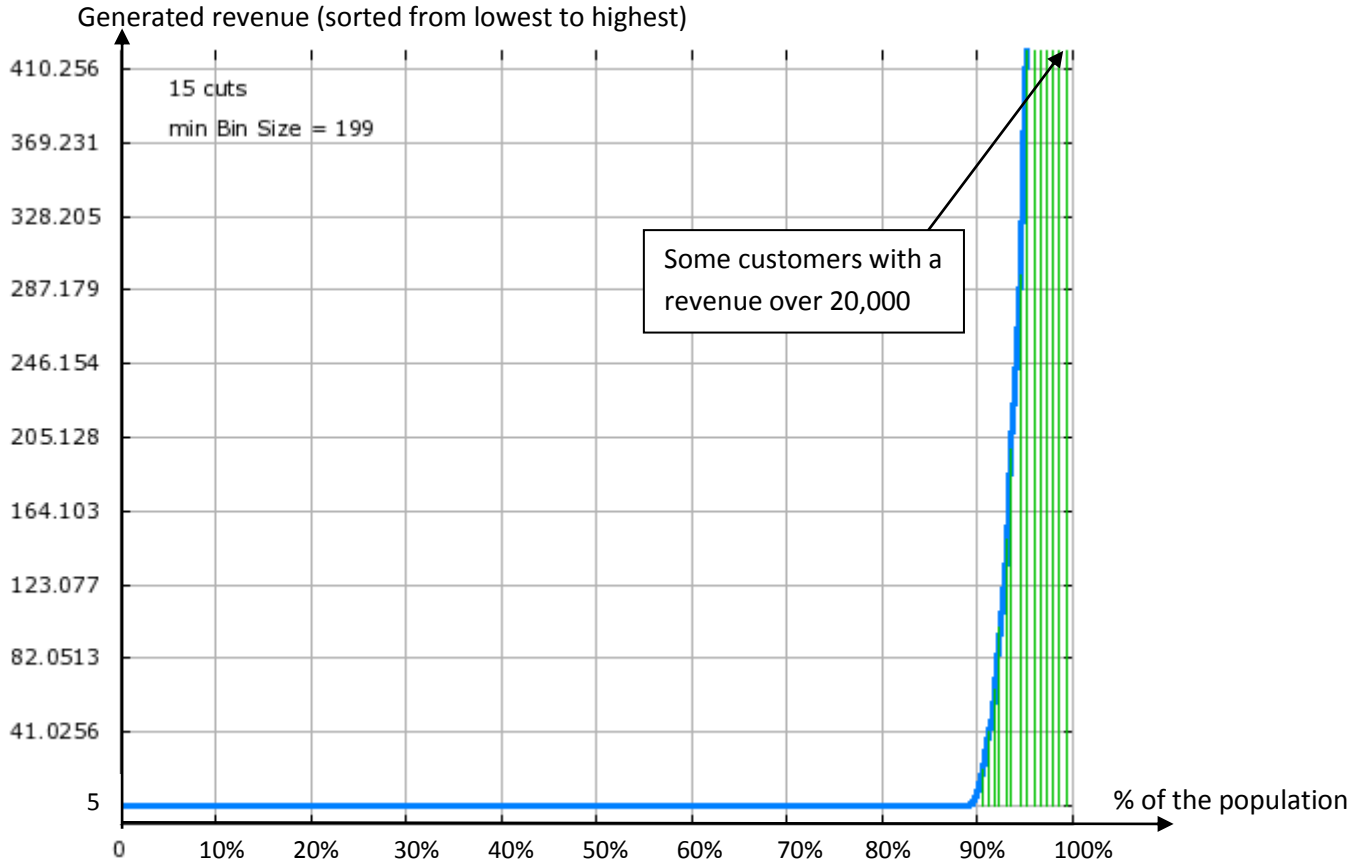
2. Create a new column inside the dataset named “weight”. The “weight” column is computed this way:
    a. For the customers that already bought the product, the “weight” column is: *“weight= min (ceiling, revenue generated by the customer)”*
    b. For the customers that did not buy the product (yet), the “weight” column is: *“weight= defaultWeight”*

    The parameters “*ceiling*” and “*defaultWeight*” are user-defined parameters that you need to set manually.
3. Create a binary prediction model, using:
    a. …as creation dataset, the whole population (minus the test dataset created during the step 1 herabove)
    b. …as target, the people that bought the product.
    c. …as row weight, the “weight” column, as computed on step 2 hereabove.
4. Apply your predictive model on the test dataset and compute the generated revenue (i.e. the revenue that you get when contacting the top 5% of the ranking).
5. Optimize the parameters “*ceiling*” and “*defaultWeight*” to get the highest revenue on the test dataset: This means that you will iterate the steps 2,3&4 many times, trying different values for the parameters “*ceiling*” and “*defaultWeight*” until you get the values for which the revenue is the highest on the test dataset.

The second approach (named “*weighted binary prediction*”) usually works a lot better than the first approach (named “*Two-steps model*”). This works better because:

- It does not involves creating a continuous model. Typically, continuous models for such kind of tasks are extremely difficult to create. It’s very common to get a R^2 of maximum 10% for such difficult tasks. When this happens (i.e. all the time), the continuous models gives so poor predictions that it’s better not to use it.
- The parameters “*ceiling*” and “*defaultWeight*” are the result of an optimisation process that maximize the revenue. Thus, the generated binary predictive model is also optimized to maximize the revenue. On the contrary, the models used inside the “*Two-steps model*” approach are not optimized to maximize the revenue and are thus less efficient for our ultimate goal (that is getting the highest revenue! ;-)).

To compute the “weight” column, we used the parameter “*ceiling*”. This parameter is required because of the nature of the variable “*revenue generated by the customer*”. As all financial variables, the variable “*Generated Revenue*” is typically strongly asymmetric: it’s quite common to see such kind of distribution:



We can see the following:

- 90% of the population has a small "Generate revenue" (i.e. the "Generate revenue" is only 5).
- There are a few customers (less than 1% of the population) that have an extremely high "Generate revenue" (i.e. the "Generate revenue" is above 20,000).

This means that, without a "ceiling" function on the revenue to compute the "weights":

- We'll give nearly "all the weight" to the very few customers with a high "Generated revenue", completely ignoring the 90% of the population (and nearly ignoring 99% of the population).
- We'll produce a binary predictive model that practically ignores the profiles of 99% of the population (and this is not good).

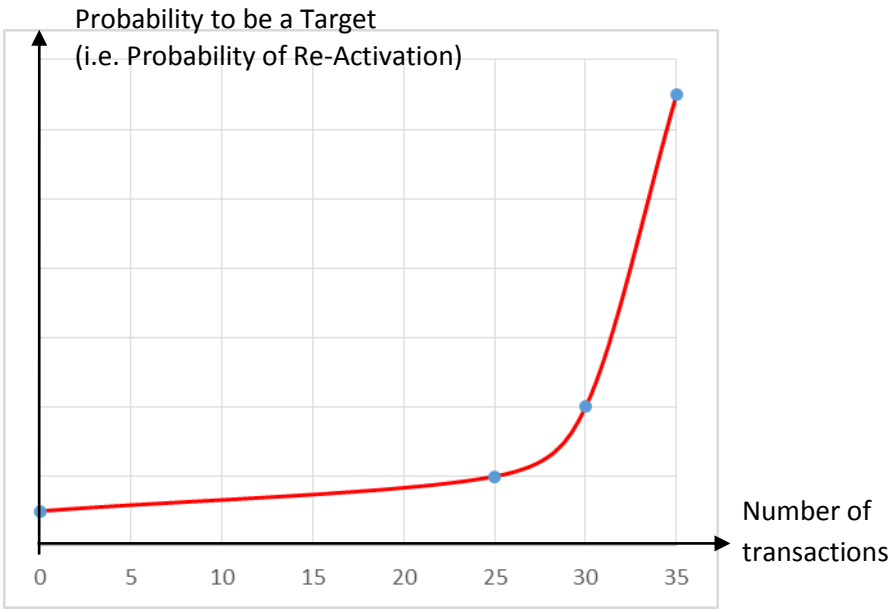
The second approach (named "*weighted binary prediction*") is not as common as the first approach (named "*Two-steps model*") because it involves creating many predictive models (i.e. we need to create a new predictive model at each iteration of the loop described in step 5 here above). Creating many models using SAS or IBM is a long and tedious task. This explains why, despite the largely superior performance of the "*weighted binary prediction*" approach, SAS and IBM only offer the "*Two-steps model*" approach.

Example 5: Obvious Model

Let's assume that you are working for a bank. Some of your customers are "dormant": i.e. they did not make any transactions for the last 6 months and they might even close their bank-account very soon (this is no good!). You want to contact some of these "dormant" customers, to invite them to use their account again. To select which customer to contact, you need to create a predictive model for "re-activation". You'll use:

- …as learning dataset: All the dormant customer: More precisely: All the customers that made less the 35 transactions during the 6 months before the observation date.
- …as target: The customers that made more than 35 transactions during the 6 months after the observation date.

Without building any model, I can already tell you the following: "*The customers that are close to the 35 transactions threshold are more likely to become target*". Graphically, we have something like:



…then it comes without surprise that the 6 most important variables inside my predictive model are all variations around the same concept (which is the "*Number of Transactions during the last 6 months before the observation date*"). Furthermore, the lift of my model is quite high (AUC>90%).

To remind you, the goal of the predictive model is to re-activate the dormant customers. To re-activate them, we need to send them a communication that stimulate them to use their account again. What to include inside this communication? Which argument(s) to invoke? To know the right arguments, we'll look at the variables selected by our reactivation model.

Inside our model, we have the variable "*Number of Transactions during the last 6 months before the observation date*". The argument linked to this variable is: "*You are close to the threshold of 35 transactions, why don't you do a little bit more transactions?*". This is obviously a very wrong argument to use inside a communication. What we are searching for

is some key behaviour that are the root cause of the reactivation. The variable "*Number of Transactions*" does not help us to find these causes and it must thus be discarded: We must create another model that does not use the variable "*Number of Transactions".*

To summarize: To learn something less obvious, to find the real cause of reactivation (and to target the right individuals), it's important to remove from the learning dataset all the variables related to the concept "*Number of Transactions"*.