Predicting the future, Part 3: Create a predictive solution

Alex Guazzelli July 03, 2012

This is the third article of a four-part series focusing on the most important aspects of predictive analytics. Part 1 offered a general overview of predictive analytics. Part 2 focused on predictive modeling techniques; the mathematical algorithms that make up the core of predictive analytics. This article describes how to use those techniques and create a predictive solution.

View more content in this series

Introduction

Predicting the future has come a long way since the advent of palm reading. It involves large amounts of historical data that need to be analyzed and pre-processed before being submitted to a predictive technique for training. A predictive model entails the coming together of data and clever mathematics to solve a specific problem. Given a well-defined problem and a model built to solve it, any prediction mistakes need to be thoroughly measured and evaluated. Model evaluation is therefore used to determine model accuracy. Evaluation results are then used for selecting the best model and to set ideal discrimination thresholds. When paired with business rules, predictive models have the potential to truly affect the bottom line of your business. A predictive solution is therefore the coming together of data, clever mathematics and business rules, which together deliver enhanced business decision capabilities.

Predictive analytics allows companies and individuals to build solutions that are capable of using historical data to predict the future. More often than not, even extraordinary predictive models are never used since the problem they try to address was insufficiently defined.

Everybody assumes that the building of a predictive solution starts with data analysis and pre-processing followed by model building and so on. In fact, it starts earlier than that. A predictive solution is the coming together of data and clever mathematics to solve a specific problem. This problem needs to be well defined if the solution is to succeed. If the problem is ill defined, the results obtained when evaluating the solution will be hard to measure and, consequently, it may never be operationally deployed. On the other hand, a well-defined problem allows for a clear assessment of how good the proposed solution is. That makes it easier for all parties involved to understand and trust its results.

Given a specific, well-defined problem, data scientists can look at the historical data to make sure that it supports the building of a solution. Even if not perfect, the available data may provide enough information for the building of a successful predictive solution. Whenever data analysis is completed, pre-processing ensues. This phase is followed by model building, which will result in a predictive model. After the model is evaluated (does it work?), it is then placed into a business context and finally used. This article focuses on all phases involved in the making of a predictive solution, from data pre-processing all the way to its operational deployment. Below, I will use IBM SPSS Statistics to illustrate many of these phases.

Data pre-processing: From raw data to features

Should a predictive model be built directly from the raw data? Or, should the data be pre-processed before model building? As usual, the answer is "it depends." Although certain data fields may be used as is, the vast majority requires some kind of massaging.

Historical data comes in all shapes and forms. For example, the data may contain structured and unstructured information about a particular customer. In this scenario, structured data encompasses fields such as customer age, gender, and number of purchases in the last month. This information is obtained from the customer account record and past transactions. Unstructured data may be represented by a comment the same customer provided as feedback for items or services purchased. In Parts 1 and 2 of this series, I used similar data fields to illustrate one application of predictive analytics: The prediction of customer churn or defection based on attrition. If you are reading the articles in sequence, you are probably familiar with this problem by now.

To build a predictive model to output churn risk, we may decide to bin values together for certain input fields. For example, all ages lower than 21 may be binned together into a student category. Similarly, all ages higher than 55 may be binned together into a retiree category. Category worker may then be assigned to everyone 21 to 55 years old. This is obviously a simplification of the original age field, but it may augment its predictive power when presented to a predictive technique for training. In fact, augmenting the predictive power of input fields is the ultimate goal of data pre-processing. By deriving features from raw data, we are in fact highlighting what matters. In so doing, we make it easier for a predictive technique to learn the important patterns hidden in the data.

Another goal of pre-processing is altering the data so that it is suitable for training. For example, depending on the technique used for model building, the three age-related categories above may need to be discretized. In this case, if customer A is 25 years old, her age would then be represented by three distinct fields: student, worker, and retiree, which would be mapped to 0, 1, and 0, respectively. For the same reason, any continuous fields may need to be normalized. In this case, number of purchases in the last month would be transformed into a number between 0 and 1. Note that the original field by itself is already a result of pre-processing since it represents an aggregation: The total number of purchases for all transactions that happened during a certain period.

In addition, we may decide to use text mining to identify attrition cues in comments and create an attrition measure that may also be represented by a value between 0 and 1. The idea is then to

represent each customer by her list of features and combine them into a record. If 100 features are selected and 100K customers exist, we will end up with a dataset containing 100K rows or records and 100 columns.

To make data massaging easier, in addition to allowing users to manipulate their data as described above, quite a few statistical packages allow users to select an option to pre-process the data automatically. IBM SPSS Statistics, for example, allows for the automatic building of features. Simply choose **Transform menu > Prepare Data for Modeling** and click on **Automatic**. You will then be asked about your objective. You can choose from a list of four options. They are: 1) Balance speed and accuracy; 2) Optimize for speed; 3) Optimize for accuracy; and 4) Customize analysis. This last option is recommended for experienced users only. IBM SPSS Statistics also allows for a myriad of transformations to be applied to the data. These can be easily accessed through the Transform menu.

After the data has been appropriately massaged, it is time for model training.

Model training: Learning patterns from data

During training, all data records are presented to a predictive technique that is responsible for learning patterns from the data. In case of customer churn, that will consist of patterns, which differentiate churners from non-churners. Note that the objective here is to create a mapping function between the input data (age, gender, number of items purchased in the last month, and so on) and the target or dependent variable (churn vs. non-churn). Predictive techniques come in different varieties. Some, such as neural networks (NNs) and support vector machines, are very powerful and exult the capacity to learn complex tasks. These techniques are also generic and can be applied to a variety of problems. Others, such as Decision Trees and Scorecards, are also capable of explaining the reasoning behind their predictions. Given that predictive techniques were covered in Part 2 of this series, I will focus in this article on NNs. Keep in mind that the same model building principles apply to all other techniques. I will also assume from now on that you will be the data scientist in charge of the modeling task.

To build a NN in IBM SPSS Statistics, choose **Analyze menu > Neural Networks** and click on **Multilayer Perceptron**. After you do that, a tab-separated window appears that will allow you to configure all the parameters necessary for building your predictive model.

In the "Variables" tab, select the target or dependent variable and all the input variables you want to present to the network for training. IBM SPSS Statistics requires you to split your input into factors and covariates. Factors represent categorical input fields such as age, which we transformed earlier to contain values: student, worker, and retiree. IBM SPSS Statistics will automatically discretize these before training begins. Covariates represent continuous variables. These will also be rescaled automatically to improve network training. Given that discretization of categorical variables and rescaling of continuous variables are applied by default, there is no need for you to do that before model training.

It is common practice to split the dataset containing the massaged data into two: one reserved for model training, and the other for testing. The first dataset usually consists of 70 percent of the total

data. The second dataset, which contains the remaining data, is then used for model validation (see below). That is the separation you get by default with IBM SPSS Statistics. You can change that by selecting the Partitions tab.

As with any predictive technique, NNs are accompanied by parameters that can be tweaked depending on the data and problem you are trying to solve. In the Architecture tab, you can choose to go ahead with an option for Automatic architecture selection or you can decide, depending on your level of expertise, to customize the network to have a specific number of layers and nodes per layer. For more information on the architecture of a NN and its importance for training, please refer to Part 2 of this article series. In the Training tab, you will encounter other parameters including the learning rate (how fast should the network learn?) and the optimization algorithm. Keep in mind that the wrong choice of parameters may prevent learning from happening. Note also that as with any kind of medicine (over the counter or not), choosing to set a parameter one way or another for a predictive technique always comes with side effects. For example, set the learning rate too low for a NN, and it will get stuck. Set it too high, and it will not converge. Keep in mind that the default parameters offered by IBM SPSS Statistics are always a good place to start.

IBM SPSS Statistics also offers other configuration tabs that will be discussed in the following section. If all tabs have already been configured appropriately, it is finally time to build the model. For that, just click on the **OK** button at the bottom of the window containing all the different configuration tabs.

Model validation: Judging accuracy

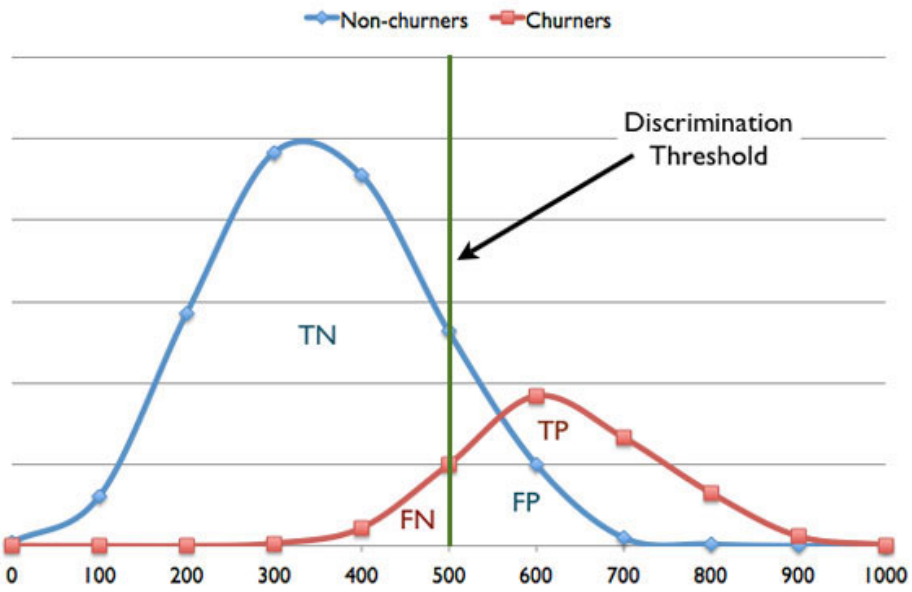
It is good practice to reserve up to 30 percent of the data for model validation. Using a data sample that was not involved in the training of the model allows for a non-biased assessment of its accuracy.

In the "Output" tab, select to output the "Classification results". When model training is completed, the "Classification table" gives you an instantaneous snapshot of the model's health. It lets you know quickly if it was able to learn the task at hand or not. It need not be 100 percent accurate. In fact, predictive models are known to make incorrect predictions. What we want to make sure is that they make as few mistakes as possible while making many correct predictions. Translating into predictive analytics jargon, you want the model to have a low rate of false-positives (FP) and false-negatives (FN), which implies a high rate of true-positives (TP) and true-negatives (TN). For the customer churn problem, if the model accurately assigns a high churn risk to someone that is about to defect, you get a TP. If it assigns a low churn risk to someone that will in fact continue to be a loyal customer, you get a TN. However, every time the model assigns a high churn risk to a satisfied customer, you get a FP. Also, if it wrongly assigns a low churn risk to someone about to jump ship, you get a FN.

Figure 1 shows a plot of the testing data after it has been scored. The plot assumes that the model outputs a risk score between 0 and 1000 and that the higher the score, the higher the risk. Since the testing data is labeled, we can separate it into two distinct curves: churners, represented by the red curve, and non-churners, represented by the blue curve. The line in green represents a

discrimination threshold, which we chose to be at 500. In this way, if a customer scores under this threshold, she is considered a non-churner. And, if her score is higher than 500, she is consider a churner. However, as can be seen in Figure 1, a threshold equal to 500 creates quite a few errors: FPs and FNs. By selecting a higher threshold, say 600, we minimize FPs, but are left with several FNs. By selecting a lower threshold, say 400, we substantially minimize FNs, but are left with lots of FPs. This give and take begs the question: Given all the potential choices of threshold values, which one is ideal? The answer depends on business goals and the cost associated with the different types of errors.

Figure 1. Scored testing data showing model scores from 0 to 1000 for churners and non-churners. Discrimination threshold = 500.

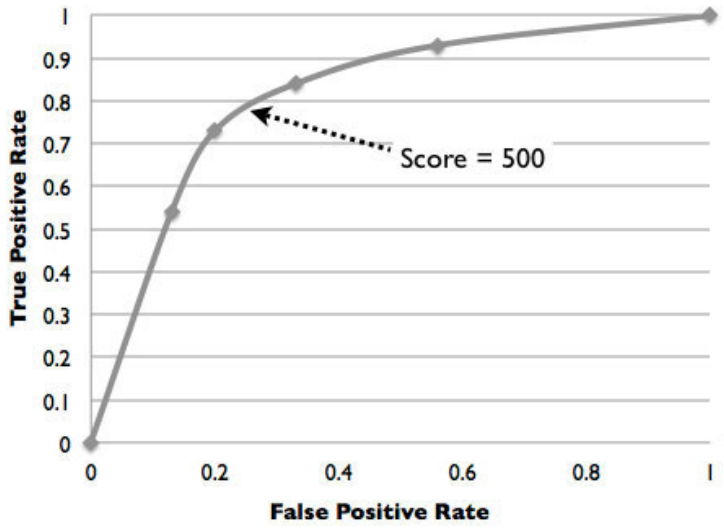


The cost associated with a FN is usually very different from the cost associated with a FP. For the customer churn example, by failing to identify a customer that is about to defect (FN), the company forfeits any future revenues that customer may have brought in. By offering freebies and retention packages to satisfied customers (FPs), it is throwing money down the chute. For this reason, depending on the cost associated with each type of error, you may decide for a predictive model that produces less FNs than FPs or vice-versa. This decision will determine the ideal discrimination threshold.

A great way to assess the accuracy of a model trained to classify input data into two classes relies on the inspection of its ROC (Receiver Operating Characteristic) curve, which can also be used for selecting the ideal discrimination threshold. You can output this curve in IBM SPSS Statistics by selecting the Output tab and clicking on ROC Curve. The ROC curve provides a graphical representation of the true positive rate (sensitivity) versus the false positive rate (one minus specificity) for a binary classifier as its discrimination threshold varies. Figure 2 shows the ROC curve obtained for a churn risk model in which the true positive rate is obtained at different discrimination thresholds by dividing the number of TPs by the total number of churners (represented by the red curve in Figure 1). Similarly, the false positive rate is obtained at different

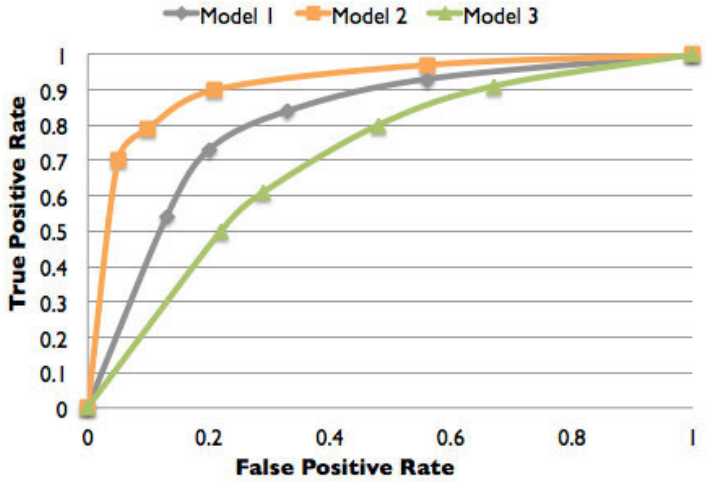
discrimination thresholds by dividing the number of FPs by the total number of non-churners (represented by the blue curve in Figure 1). As shown in Figure 2, a score of 500 identifies a tipping point in the curve where we still get a relatively low false positive rate and a high true positive rate.

Figure 2. ROC graph showing the true positive rate versus false positive rate as the discrimination threshold changes. Dots in line represent different discrimination thresholds. These are: 0, 200, 400, 600, 800, and 1000.



Furthermore, you can use the area under the ROC curve (output by IBM SPSS Statistics together with the ROC graph) to compare different models (in case you build several models with different parameters). The larger the area, the more accurate the model is believed to be. Figure 3 shows the ROC graph for three different models: 1, 2, and 3. We can readily deduce that model 2 is more accurate than models 1 and 3.

Figure 3. Three models represented by their ROC curves. Because model 2 has the largest area under the curve, it is believed to be more accurate than models 1 and 3.



Data post-processing: From scores to business decisions

The raw output of a predictive model is usually a value between 0 and 1 or -1 and 1. To be more palatable to humans though, this output is often times scaled to a value between 0 and 1000. The output of a model may also be scaled according to a given function, which makes sure a certain score indicates a desirable number of TPs and FPs.

Post-processing also implies embedding scores into the operational process. In this case, scores need to be translated into business decisions. For example, for the customer churn problem, your company may decide to offer a great retention package for customers with a high churn risk. In fact, different discrimination thresholds can be used to determine different business decisions depending on the score and what it represents in terms of TPs, FPs, and their cost.

Decision management solutions that traditionally incorporated only business rules, now embed predictive models. They do that by wrapping up models into rules. The combination of business rules and predictive analytics allows companies to benefit from two types of knowledge: expert and data-driven. In this case, the decisions based on different thresholds can be implemented right away and even expanded to contain other important deciding factors. For example, to determine how valuable a customer really is, her churn risk score may be paired with the amount of money spent in the past. If spending is high and risk of defeating is also high, the more important it is to make sure the customer does not churn.

By combining expert and data-driven knowledge, decision management solutions become smarter, since they are capable of offering enhanced decision capabilities.

Conclusion

The building of a predictive solution starts by a clear definition of the problem it is trying to solve. With a well-defined goal, data analytics scientists can then embark on the building of a predictive model that will be accurate, whose benefits will be readily explainable to all the parties involved.

After we set the problem we are trying to solve, we analyze and pre-process historical data in preparation for model training. The result of this process is a set of features that augment the predictive value of the raw data. Data pre-processing is necessary to make the original data suitable for model training. Given that a myriad of predictive techniques exist, a predictive solution may benefit from a specific technique such as NNs, or from an ensemble of techniques that work together to solve the problem at hand.

After a building a predictive model, we need to evaluate it. Because errors are part of the predictive analytics world, we can use different discrimination thresholds not only to minimize errors, but also to reduce their associated cost. Because different thresholds can work in tandem with business rules, a tiered system can be put in place to optimize the use of the scores produced by a predictive model. By combining business rules and predictive analytics, a true predictive solution is born.

Part 4, the last article in this series, will focus on deploying predictive analytics and putting predictive solutions to work.