Custom R Modules in Predictive Analysis

With the release of version 1.0.1 of Predictive Analysis in early June 2013, SAP added a feature allowing users to add new R algorithms to the Predictive Analysis algorithm library. This might be desirable for several reasons:

1.) An organization has an existing model or algorithm that they would like to make available within Predictive Analysis for either analytical or comparison purposes
2.) A modeler/analyst would like to build a model using an algorithm that is not currently supported in Predictive Analysis
3.) A modeler/analyst would like to implement an algorithm that exists within Predictive Analysis, but would like to perform additional processing, have different visualizations, or make different settings available.

In this blog post, we’ll walk through an example of how to create a custom R component for Logistic Regression.

Logistic Regression Background

Logistic regression is one of my favorite predictive algorithms because it is accurate, versatile, and granular. I’ve heard the explanation that logistic regression is the equivalent of linear regression for binary responses. Therefore, where a modeler might use linear regression to predict a continuous output (like what our profit will be next quarter based on economic factors), he would use logistic regression to predict the likelihood a customer will purchase a product, respond to an offer, or the likelihood that data is accurate, for example. Logistic regression by definition will take inputs and return a value between 0 and 1, which is the probability of the outcome. If we used linear regression for this, we could easily return values below 0 or above 1 if our inputs move to the extremes, which are not valid probabilities.

While there are lots of classification algorithms that predict categorical or binary values, what I really like about logistic regression is the granularity of the prediction; because logistic regression can take in continuous variables (potentially many continuous variables), the user can get back extremely granular results with probabilities anywhere between 0 and 1. This allows the user to self-select a cutoff point or points for handling of records and even just to prioritize records; for example, if we are trying to predict customers likely to respond to a marketing offer, we can use a logistic regression model in any of the following ways:

1.) Send marketing offer to customers predicted to respond (logistic regression predicts probability of response >0.5)
2.) If we have budget to send only 100,000 offers, we can send marketing offer to the 100,000 customers with the highest predicted response probability.
3.) Perhaps we determined that based on the model accuracy and response rates and marketing costs that the cost-benefit for the marketing offer is only break-even for customers that have a response probability of 68% or higher—therefore we can use 0.68 as our cutoff.

Other popular classification algorithms like decision trees give a simple 0/1 output, and tend to create “chunks” of customers with no differentiation within large groups, because they do not handle continuous data very well. They also struggle to handle continuous variables, compared to regression.

With version 1.0.11, SAP added the PAL-based logistic regression algorithm for HANA Online mode. The PAL logistic regression algorithm allows only numeric predictors and has minimal visualizations and fit information available in the automated output, so we will create a custom R logistic regression algorithm that can be used in Local/Offline mode and has our desired visualizations.

Further resources on Logistic Regression:
Example of Simple Logistic Regression: [http://www.vassarstats.net/logreg1.html](http://www.vassarstats.net/logreg1.html)

Bank Customer Churn Data
For this example, we will be testing our algorithm on a dataset with bank customers and some demographic and financial data, including the age, number of children, income, wealth, and bank account/loan amounts for each customer. We’re concerned with the attrition rate of our customers, so we’d like to be able to predict which customers are most likely to churn or defect, so we can target them for additional retention measures. The picture below shows a portion of the variables available in the churn dataset; we also have an integer (0, 1) column indicating whether the customer churned or defected from the organization (0 = no defection, 1 = defected).

This dataset is used as a test case for the logistic regression algorithm that we are creating today, but the logistic regression algorithm can be used on any dataset to predict a binary (0/1) outcome.

Create a Custom R Logistic Regression Component
The “Add a New Component” button was added at the top of the Predict pane, next to the Import/Export models buttons:
Clicking on the “Add New Component” allows the selection of “R Component” only. It then steps through a wizard-like interface to create the component, first setting the tab the new object should appear under and the name of the new component.

When the new component interface pops up, enter a name for the new component, as shown in the picture below, and then click “Next”.

On the R Script entry screen, paste the R code described below (without red annotations) in the Script Editor window and then set the configurations as shown in the screenshot of the Script screen.

This is a support function that creates one of the visualizations for the logistic regression model.

Source: http://www.itc.nl/~rossiter/teach/R/R_lcc.pdf

```r
logit.plot <- function(model, title="Success of logistic model") {
  # sort the fitted values
```
This is the function that builds the call for and runs the logistic regression algorithm. It also generates the visualizations for the model fit (a ROC curve, a histogram, an influence chart, and the logit plot).

This function requires the ROCR and car R libraries to be installed in the installation of R that is used. For details on installing R libraries, refer to the previous blog on SAP Predictive Analysis installation.

```r
LogReg <- function(InputDataFrame, IndependentColumns, DependentColumn) {
  formattedString <- paste(IndependentColumns, collapse = '+');
  finalString <- paste(paste(DependentColumn, '~'), formattedString);
  print(paste('Modeling function call:', finalString))
  lr_model <- glm(finalString, family=binomial(logit), data=InputDataFrame);
  result <- lr_model$fitted; #predict(lr_model, InputDataFrame);
  output <- cbind(InputDataFrame, result);
  par(mfrow=c(2,2))
  library("ROCR")
  dep <- InputDataFrame[IndependentColumn]
  perf <- prediction(result, dep)
  auc.tmp <- performance(perf, "tpr", "fpr")
  auc <- as.numeric(auc.tmp@y.values)
  plot(perf, colorize=T, main="ROC Curve")
  text(usr[2], usr[3],
       paste("AUC:" , round(auc, digits=3)), adj = c(1,0), col = "blue")
  hist(result)
  library(car)
  influencePlot(lr_model, main = "Influence Plot")
  logit.plot(lr_model)
}
```
This is the function that calls the scoring function of our fitted model in order to score new records through against our existing model. It is used when the “save model” option is chosen and the saved model transform is used in a predictive workflow.

LogRegScoring<-function (MInputDataFrame, MIndependentColumns, Model) {
  newdata<-MInputDataFrame[MIndependentColumns]
  Predicted<-round(predict(Model, newdata, type="response") , 5)
  return(list(modelresult=Predicted))
}

After clicking finish, we have our new algorithm available under the Custom R Component category of the Algorithms tab:
Next, go into the predictive workspace and create a predictive workflow for our bank customer dataset that uses the new Hillary-R Logistic Regression algorithm module.

In the configurations for the new Logistic Regression module, I select the independent columns Age, Income, and Nbr_of_Children and the dependent column ChurnFlagInt (the 0/1 field indicating whether the record churned). I’m also going to save the model so we can run the scoring algorithm later.

Click Run Analysis to execute the predictive workflow.

Once the workflow has completed successfully, we can review the actual churn (our dependent variable) and the predicted churn produced by the new model in the data output:
The result of the logistic regression model in the column “predicted churn” is the probability, or percentage chance, that the customer will churn or defect, based on prior experience of similar customers. We compare the “predicted churn” to the dependent variable “ChurnFlagInt”; for instance where the ChurnFlagInt=1 and the predicted churn is close to one—this is an accurate prediction.

On the Charts portion of the logistic regression output, we see the 4 visualizations in the Results→Charts output area, shown in the screenshot below. These charts show that the logistic regression model is very accurately predicting the customer churn. The ROC curve in the upper left shows the prediction is significantly more accurate with the model than random chance; the further the line is curved to the upper left from a diagonal straight line from bottom left to upper right, the better the model. In the bottom right of the ROC Curve chart is the AUC (Area Under the Curve) measure—in this case it is 0.992. The closer the AUC value is to 1, the more accurate at differentiating successes and failures (defections and non-defections). The histogram in the upper right shows a large concentration of records being assigned probabilities close to 0 and 1, indicating that the model is decisively identifying customers as either likely to churn or unlikely to churn; similarly, the chart in the bottom right shows a steep S-curve between records with low likelihood to churn, and those with high, with most of the “high likelihood” customer actually churning (green marks on top).
At this point, we may decide that we like our predictive model and want to use it again later; the saved model is available on the Saved Models tab of the library window, as shown in the picture below.

New data can be scored against this model by inserting this new saved model transform into a predictive workflow and selecting the same 3 predictors (must have the same names as the original modeling dataset). In this case, we’ll test our model scoring algorithm against the same input dataset.

The configuration of the model scoring module requires the user to select the input predictor columns:

The output of the model scoring module now produces the same “predicted churn” values as the original output of the modeling dataset, and the saved model object can be saved, exported, and shared with other Predictive Analysis users by right-clicking on the saved model module and clicking “Export Model” and saving the model as a *.SPAR document.
With this, we have created our own custom R algorithm including visualizations, however if this algorithm were created within an HANA Online document, the visualizations would not be viewable because they are not supported by the Rserve process employed by the HANA-R integration. The custom R algorithm we just created is now available in any Predictive Analysis document in the same more (HANA Online vs. Local/Offline) on the workstation it was created on. I expect with a near future release, SAP will make available the ability to feature to share these algorithms between users.

Hillary Bliss, Analytics Practice Lead
Decision First Technologies
Hillary.bliss@decisionfirst.com
twitter @HillaryBlissDFT

Hillary Bliss is the Analytics Practice Lead at Decision First Technologies, and specializes in data warehouse design, ETL development, statistical analysis, and predictive modeling. She works with clients and vendors to integrate business analysis and predictive modeling solutions into the organizational data warehouse and business intelligence environments based on their specific operational and strategic business needs. She has a master’s degree in statistics and an MBA from Georgia Tech.