k-Nearest Neighbors

k-Nearest Neighbors (kNN) is a simple classification technique that is unique in the sense that no model is explicitly trained. In the kNN process, two datasets are read in: the training dataset in which the dependent variable is already known, and the test dataset in which the dependent variable is unknown. Classifications for the test set are made by determining the k most similar records in the training dataset (known as neighbors) and returning the majority vote amongst those neighbors.

For instance, a supermarket might use kNN to determine whether a prospective customer "Jim" is likely to respond to an offer. If four of the five most similar customers to Jim have responded to the offer, then Jim will be classified as likely to respond.

For more information on kNN, see the following wikibook. [3]

While kNN does not have explicit model training, this R Script has two functional modes governed by the value of the TrainIncluded parameter:

- When the training data is included in the dataset passed to R, then classifications will be made on the test set and the training data can be saved to an .Rdata file for later re-use.
- When the training data is not included in the dataset passed to R, then the data can be loaded in to memory from the .Rdata file it was saved to.

How to Deploy to MicroStrategy:

Prerequisite: Please follow the instructions in the R Integration Pack User Guide [1] for configuring your MicroStrategy environment with R and that the R Script functions have been installed in your MicroStrategy project(s).

1) Download the kNN.R file from the R Script Shelf [2].
2) From the R console, run the kNN.R script to verify the script runs correctly. For details, see the "**Running from the R Console**" section below.
3) Cut-and-paste the appropriate metric expression below in any MicroStrategy metric editor. Map the arguments to the appropriate MicroStrategy metrics. A description of each output is shown below.
4) Use the new metric in reports, dashboards and documents.

Metric Expressions:

1) **Class:** Returns the predicted class as a string:

   When the training data is included, use this metric expression:

```
RScript<_RScriptFile="KNN.R",_InputNames="ID,Target,Training,Vars",
StringParam9="KNN", BooleanParam9=TRUE, NumericParam1=1>(ID, Target, Training,
Vars)
```

   When the training data is not included and should be loaded from the .Rdata file, use this metric expression:

```
RScript<_RScriptFile="KNN.R",_InputNames="ID,Target,Training,Vars",
StringParam9="KNN", BooleanParam9=FALSE, NumericParam1=1>(ID, Target, Training,
Vars)
```

2) **ClassId:** Returns the predicted class as a number:

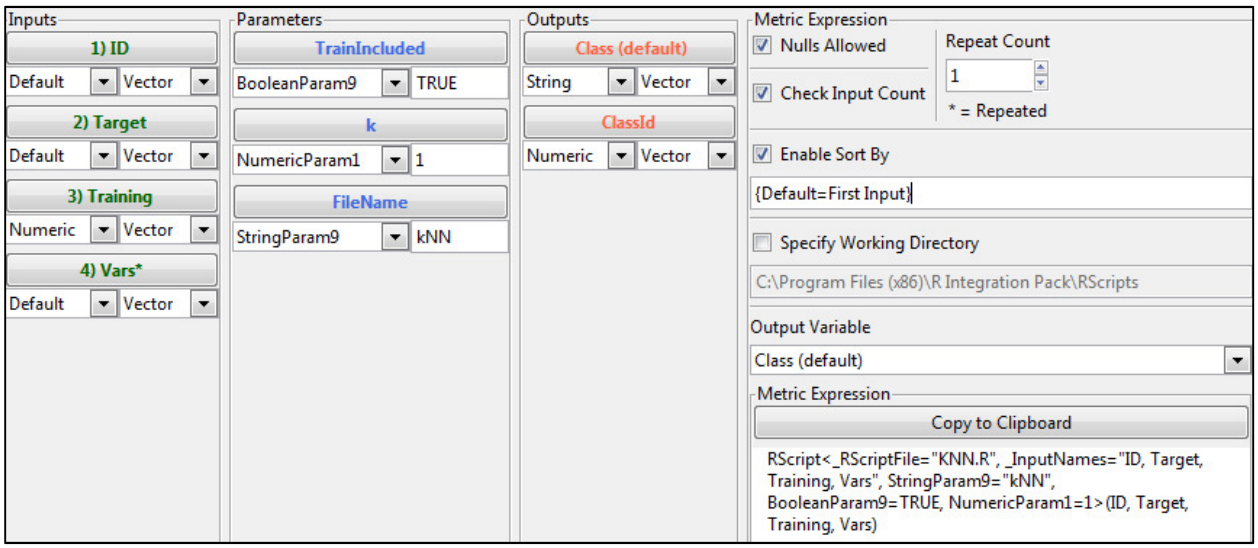
When the training data is included, use this metric expression:

```
RScript<_RScriptFile="KNN.R",_InputNames="ID,Target,Training,Vars",
[_OutputVar]="ClassId",StringParam9="KNN",BooleanParam9=TRUE,NumericParam1=1>(I
D, Target, Training, Vars)
```

When the training data is not included and should be loaded from the .Rdata file, use this metric expression:

```
RScript<_RScriptFile="KNN.R",_InputNames="ID,Target,Training,Vars",
[_OutputVar]="ClassId",StringParam9="KNN",BooleanParam9=FALSE,NumericParam1=1>(
ID, Target, Training, Vars)
```

Analytic Signature:



Inputs:

ID: The ID associated with each record. This input is used to ensure that the predictions are joined back to our dataset correctly.

Target: The observed class of each record. This variable is known for the training data but not the test data. When the training data is included, this variable must correspond to what we want to predict. When the training data is not included, this value is not known so the script does not utilize the metric passed in. Nevertheless, this input is required to maintain the structure of the expression so map target to any metric.

Training: A 0/1 vector that tells us whether this record is part of the training data or test data. If Training = 0, then the data is considered part of the test set. If Training = 1, then the data is considered part of the training set. This variable is only considered when the training data is included.

Vars: The independent variables that are used as the basis for defining what are the nearest neighbors. Since the Vars argument is a repeated input, it can be mapped to any number of MicroStrategy metrics. In this way, we enable kNN to consider any number of numeric variables when generating predictions.

Note: **Repeated inputs must all be the same type (i.e. all variables must be numeric or all variables must be strings)**

Parameters:

TrainIncluded: Uses BooleanParam9 with a default of TRUE. When set to TRUE, the data being passed in to R should include both the training set and test set. When set to FALSE, the data being passed in to R only includes the test set and the training set is loaded into memory from the .Rdata file named by the FileName parameter.

k: Uses NumericParam1 with a default of 1. This parameter controls the number of neighbors from the training set to consider when generating predictions. For instance, if this value is changed to 3, then for each record in the test set, the classification returned will be the majority vote of the three nearest neighbors in the training set.

FileName: Uses StringParam9 with a default of "kNN". This parameter specifies the file name where objects from the R environment are saved. This is most useful in saving the training data so that it does not have to be passed into R for subsequent executions. Please note the R Script automatically appends the ".Rdata" file extension to this file name. This parameter is only used when TrainIncluded=TRUE; when TrainIncluded=FALSE, no Rdata file is saved.

Outputs:

Class: A string value representing the predicted class for that record. Used when classes are represented by strings.

ClassID: A numeric value representing the predicted class for that record. Used when classes are represented by numbers.

Additional Results Generated by the R Script:

One file is stored in the working directory:

Rdata File: This file persists the state of several objects from the R environment for later inspection, analysis, and reuse, including train_df (a data frame containing the training data), test_df (a data frame containing the test data), model (the knn "model" which consists of just the predictions for the test set), Class (the predicted class of each record), Training (the 0/1 Training input) and Target (the input representing the variable we are trying to predict). This file is loaded when TrainIncluded is set to FALSE so that the training data does not have to be sent to R each time we want to generate predictions.

Running from the R Console:

In addition to processing data from MicroStrategy during execution of a report or dashboard, the R script is also configured to run from the R console. Running the script for the R Console verifies that the script is functioning as expected, a good practice when initially deploying this analytic to a new system (for more details, see "Configuring dual execution modes" in [1]).

When run from the R Console, if the script is executing properly, a "Success!" message will appear in the console. If a "Success!" message does not appear, then please note the error in order to take appropriate action. For common pitfalls, please consult the **Troubleshooting** section below.

Troubleshooting:

This section covers certain situations you might encounter but it's not intended as a comprehensive list of possible errors.

If an error occurs, the report may fail with an error message, or nulls returned as the output. In these cases, please refer to the RScriptErrors.log file generated for further guidance and the DSSErrors.log. Please consult the User Guide [1] and the R documentation for additional guidance.

The script will attempt to install the required R package. If the package is not successfully installed, you can install using the R console using the command:

```
install.pacakges("class", repos="http://cran.rstudio.com/")
```

- **class**: This R package contains functions that support k-Nearest Neighbors modeling .

If a mix of string and numeric variables is passed in to the Vars argument, the report will fail with an error message indicating that a variable with unexpected type was passed in. This can be remedied by using all strings or all numeric variables corresponding to the Vars argument.

If, when scoring, the error message "cannot open the connection" is returned, that means that the R script either cannot load the .Rdata file from the specified location or does not have access to the working directory to write the .Rdata file. Please make sure that the value passed in to StringParam9 is an existing file. For instance, if StringParam9 is set to kNN, then there must be a file kNN.Rdata in the working directory. Please consult the User Guide [1] for more information. Additionally, please make sure that the working directory is a writable location.

Example:

In order to combat an escalation in the number of customers who are churning, or leaving the company in the midst of their contracts, a large Telco organization wants to employ k- Nearest Neighbors modeling to predict which of their existing customers are most likely to churn. By proactively identifying at-risk customers, the organization can implement different measures aimed at assuaging these disgruntled customers.

Using historical information about the characteristics of customers and whether they've churned or not, we'll configure our k-Nearest Neighbors model. First, we start by creating our metric that represents whether the customer is part of the training or test set. For this example, we will consider any of the 4000 customers with an ID less than 4000 to be part of our training set and any of the 6000 customers with an ID greater than 4000 to be part of our test set:

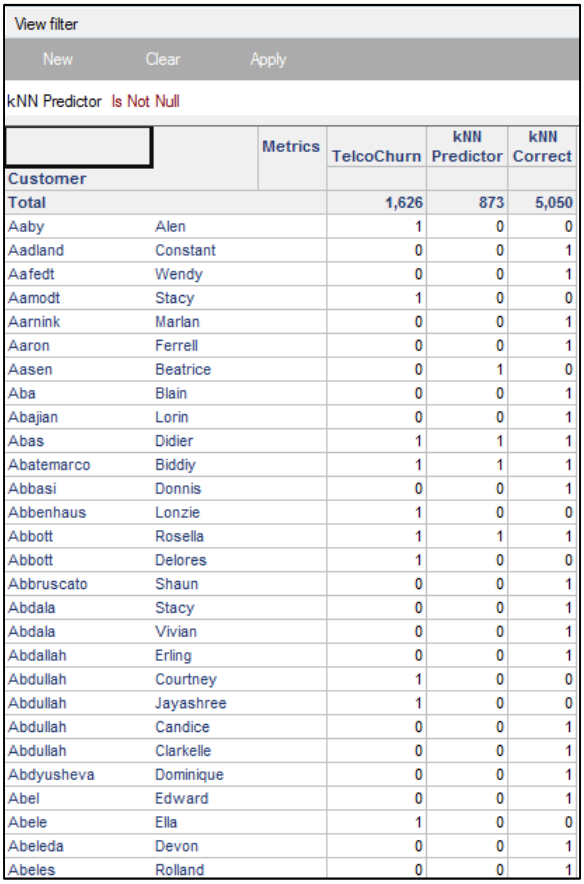
1. Create a metric named CustomerID with the expression `Max(Customer@ID){Customer}`.
2. Create a metric named Training with the expression `IF(CustomerID>4000,1,0)`.

We are now ready to create our k Nearest Neighbors predictor. In this case, we want to generate predictions for each of our 6000 customers by analyzing the outcome of the 5 most similar customers, based on the household count, income bracket, number of dropped calls, and number of helpdesk calls placed, from our test set. To do so, create a metric named kNN Predictor with the following expression:

```
RScript<[BooleanParam9]=True,[NumericParam1]="5",[StringParam9]="kNN",[_OutputV
ar]="ClassId",[_RScriptFile]="KNN.R",[_InputNames]="ID,Churn,Training,Household
Count,IncomeBracket,DroppedCalls,HelpdeskCalls">(CustomerID,TelcoChurn,Training
,[Household Count ID], [Income Bracket ID], DroppedCalls, HelpdeskCalls)
```

Lastly, to analyze the quality of our predictions, let's create one more metric named kNN Correct? with the expression `If(TelcoChurn=kNNPredictor,1,0)`.

Now, create a report with Customer on rows and TelcoChurn, kNN Predictor, and kNN Correct on columns. Apply the view filter where "kNN Predictor Is Not Null". The following report should result:


View filter

New Clear Apply

kNN Predictor Is Not Null

Customer		Metrics	TelcoChurn	kNN Predictor	kNN Correct
Total			**1,626**	**873**	**5,050**
Aaby	Alen		1	0	0
Aadland	Constant		0	0	1
Aafedt	Wendy		0	0	1
Aamodt	Stacy		1	0	0
Aarnink	Marlan		0	0	1
Aaron	Ferrell		0	0	1
Aasen	Beatrice		0	1	0
Aba	Blain		0	0	1
Abajian	Lorin		0	0	1
Abas	Didier		1	1	1
Abatemarco	Biddiy		1	1	1
Abbasi	Donnis		0	0	1
Abbenhaus	Lonzie		1	0	0
Abbott	Rosella		1	1	1
Abbott	Delores		1	0	0
Abbruscato	Shaun		0	0	1
Abdala	Stacy		0	0	1
Abdala	Vivian		0	0	1
Abdallah	Erling		0	0	1
Abdullah	Courtney		1	0	0
Abdullah	Jayashree		1	0	0
Abdullah	Candice		0	0	1
Abdullah	Clarkelle		0	0	1
Abdyusheva	Dominique		0	0	1
Abel	Edward		0	0	1
Abele	Ella		1	0	0
Abeleda	Devon		0	0	1
Abeles	Rolland		0	0	1

As you can see, our kNN Predictor is able to correctly classify 5050 out of 6000 customers from our test set, an accuracy of 84.2%.

References:

1) MicroStrategy R Integration Pack User Guide: https://rintegrationpack.codeplex.com/documentation
2) R Script Shelf: http://rintegrationpack.codeplex.com/wikipage?title=R%20Script%20%22Shelf%22&referringTitle=Home
3) http://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Classification/kNN

