Neural Networks for Classification

Neural Network is an advanced machine learning classification technique wherein a model is constructed that aims to simulate the thought process performed by the human brain. A model consists of "neurons" that are interconnected by an activation function. Every record is then passed through the network from the appropriate input neuron to the proper output neuron through a series of weights and transformations defined by the activation function.

Neural Networks have been observed to perform particularly well for many non-linear problems. In such cases, Neural Networks will often return a much better result than basic classifiers like decision trees and logistic regression.

For more detailed information on Neural Networks, please consult this site [3].

Neural Networks can easily be extended to forecast a number rather than a class. Such an application is known as Regression.

This R Script has two functional modes:

- Training creates a model which is persisted as an .Rdata file and returns predictions.
- Scoring uses the model created during training to return predictions..

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 NeuralNetwork.R file from the R Script Shelf [2].
2) From the R console, run the NeuralNetwork.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:

   For training, use this metric expression:
   ```
   RScript<_RScriptFile="NeuralNetwork.r",_InputNames="Target,Vars",_NullsAllowed=
   False,StringParam9="NeuralNetwork",BooleanParam9=TRUE,NumericParam1=3,
   NumericParam2=42>(Target,Vars)
   ```

   For scoring, use this metric expression:
   ```
   RScript<_RScriptFile="NeuralNetwork.r",_InputNames="Target,Vars",_NullsAllowed=
   False,StringParam9="NeuralNetwork",BooleanParam9=FALSE,NumericParam1=3,
   NumericParam2=42>(Target,Vars)
   ```

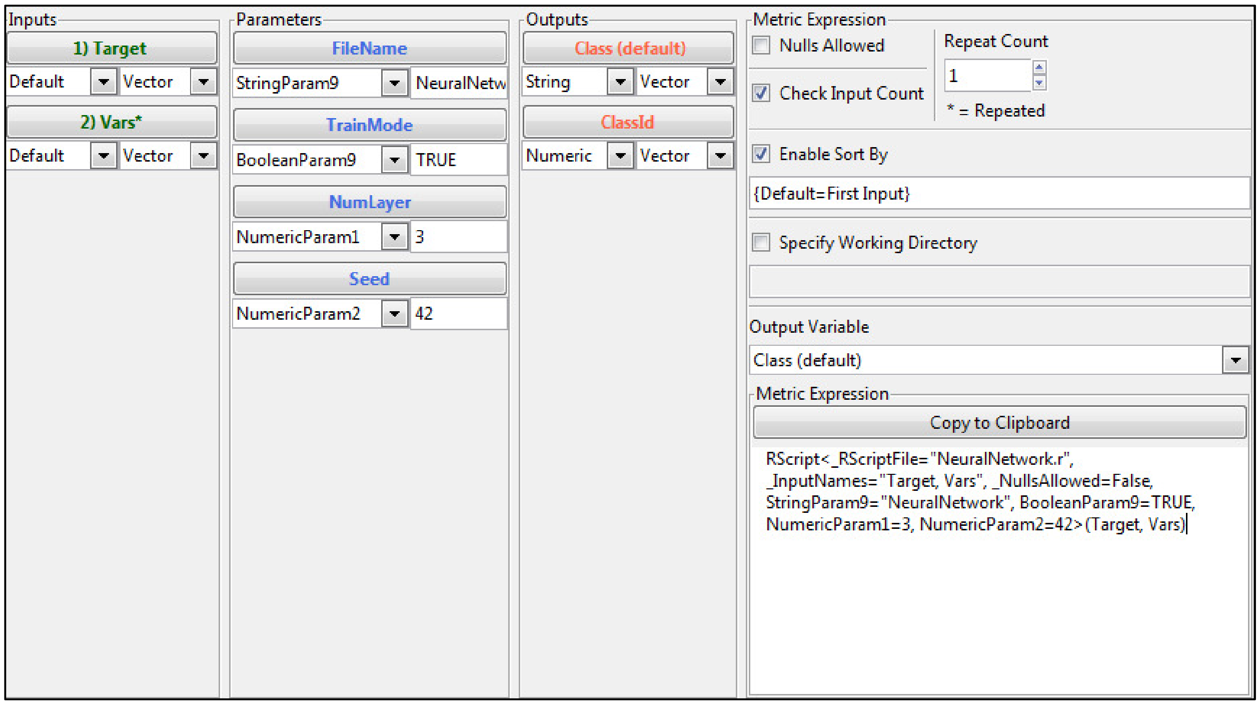
2) **ClassID:** Returns the predicted class as a number:

   For training, use this metric expression:

RScript<_RScriptFile="NeuralNetwork.r",_InputNames="Target,Vars",_NullsAllowed=False,[_OutputVar]="ClassId",StringParam9="NeuralNetwork",BooleanParam9=TRUE,NumericParam1=3,NumericParam2=42>(Target,Vars)

For scoring, use this metric expression:
RScript<_RScriptFile="NeuralNetwork.r",_InputNames="Target,Vars",_NullsAllowed=False,[_OutputVar]="ClassId",StringParam9="NeuralNetwork",BooleanParam9=FALSE,NumericParam1=3,NumericParam2=42>(Target,Vars)

Analytic Signature:



Inputs
- 1) Target: Default, Vector
- 2) Vars*: Default, Vector

Parameters
- FileName: StringParam9, NeuralNetw
- TrainMode: BooleanParam9, TRUE
- NumLayer: NumericParam1, 3
- Seed: NumericParam2, 42

Outputs
- Class (default): String, Vector
- ClassId: Numeric, Vector

Metric Expression
- [ ] Nulls Allowed
- [✓] Check Input Count
- Repeat Count: 1
- * = Repeated
- [✓] Enable Sort By: {Default=First Input}
- [ ] Specify Working Directory
- Output Variable: Class (default)

Metric Expression

Copy to Clipboard

```
RScript<_RScriptFile="NeuralNetwork.r",
_InputNames="Target, Vars", _NullsAllowed=False,
StringParam9="NeuralNetwork", BooleanParam9=TRUE,
NumericParam1=3, NumericParam2=42>(Target, Vars)
```

Inputs:

Target: The observed class of each record. When training, this input represents the value we are trying to predict. When scoring, this value is generally not known so the script does not rely on the metric passed in. Nevertheless, this input is required to maintain the structure of the expression so map target to any metric with non-null values (i.e. revenue) when scoring.

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

TrainMode: Uses BooleanParam9 with a default of TRUE. When set to TRUE, the R script will train (i.e. create) and save a Neural Network model using the dataset provided. When set to FALSE, the R script will score the records on the report against a pre-existing model. In practice, one first trains a model and once that model has been validated, scores unknown records against that model. For more information on training/scoring, please see this document.

FileName: Uses StringParam9 with a default of "Neural Network". This parameter specifies the file name where the trained model is stored in an Rdata file . Please note the R Script automatically appends the ".Rdata" file extension to this file name. This parameter is only used when TrainMode=TRUE; when TrainMode=FALSE, no Rdata file is saved.

NumLayer: Uses NumericParam1 with a default of 3. This parameter controls the number of hidden layers in the Neural Network model.

Seed: Uses NumericParam3 with a default of 42. This parameter controls the seed for random number generation to provide a mechanism for ensuring consistent results from execution to execution. By changing this number, different random results could be generated.

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 df (a data frame containing the data read in from MicroStrategy), model (the Neural Network model object), and Class (the predicted class of each record). This file is loaded when scoring new records to reference the model that was trained based on past history.

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("nnet", repos="http://cran.rstudio.com/")
```

- **nnet**: This R package contains functions that support the creation of the Neural Network algorithm used to build the classification model.

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 cannot load the Rdata file from the specified location. Please make sure that the value passed in to StringParam9 is an existing file. For instance, if StringParam9 is set to NeuralNetwork, then there must be a file NeuralNetwork.Rdata in the working directory. Please consult the User Guide [1] for more information.

Example (using MicroStrategy Tutorial Project):

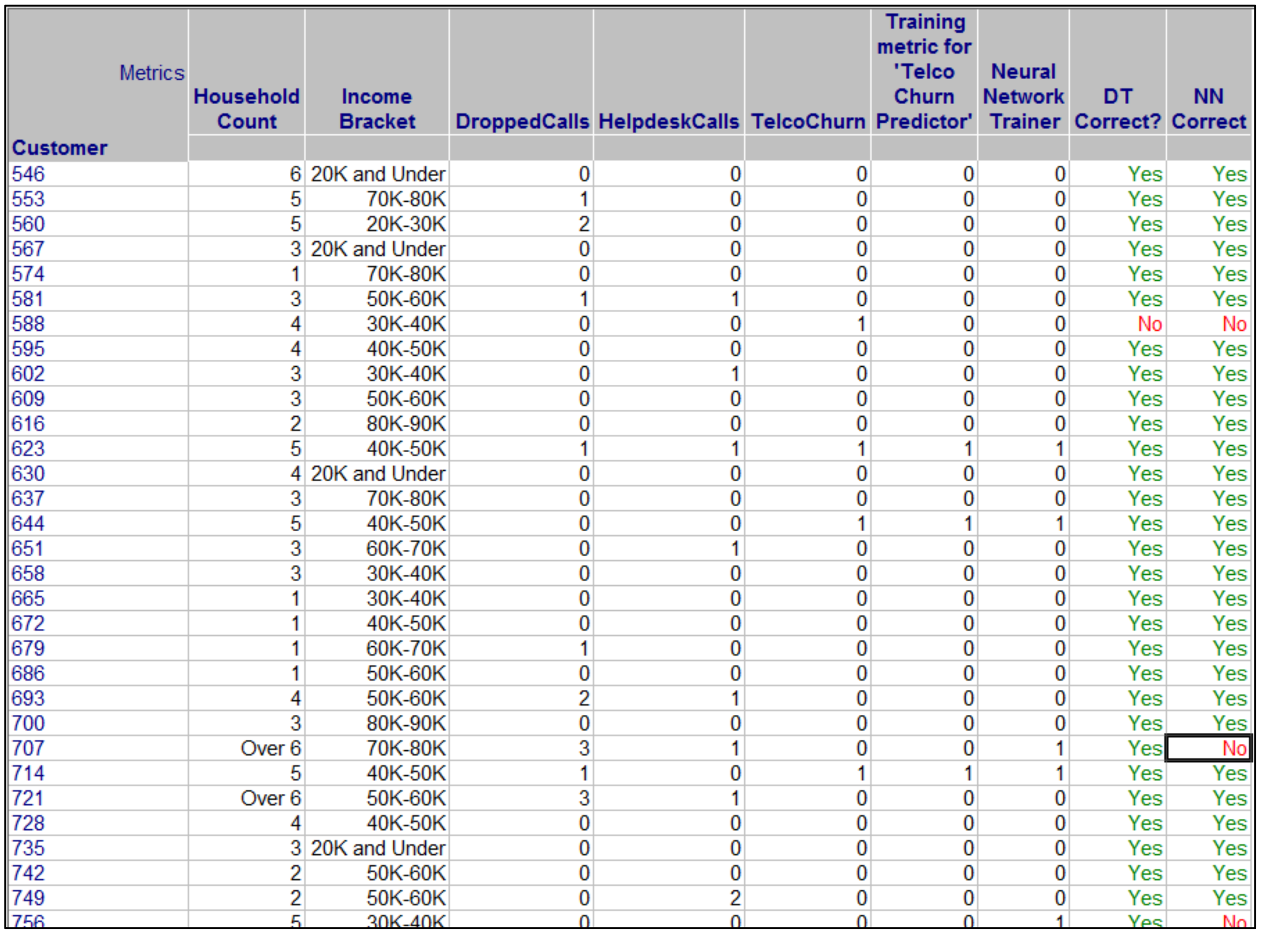
In order to combat an escalation in the number of customers who are churning, or leaving the company in the midst of their contracts, a large Telco organization wants to create and deploy a Neural Network model that predicts which customers are most likely to churn. By proactively identifying at-risk 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 create a metric Neural Network Trainer with the following definition:

```
RScript<[BooleanParam9]=True,[NumericParam1]="3",[NumericParam2]="42",
[StringParam9]="NeuralNetwork",[_OutputVar]="ClassId",[_RScriptFile]="NeuralNetwork.r"
>(TelcoChurn, [Household Count ID], [Income Bracket ID], HelpdeskCalls, DroppedCalls)
```

Where Target is mapped to TelcoChurn and Vars consists of the metrics Household_Count_ID, Income_Bracket_ID, HelpdeskCalls, and DroppedCalls.

Then, open up report "1—Training Report for Telco Churn Predictor (Tree)" and add the Neural Network Trainer metric to the report. By adding this metric to the report, we train the model and store it in a file called NeuralNetwork.Rdata. Here is how the report should look after all metrics not included in our model are removed from the grid:



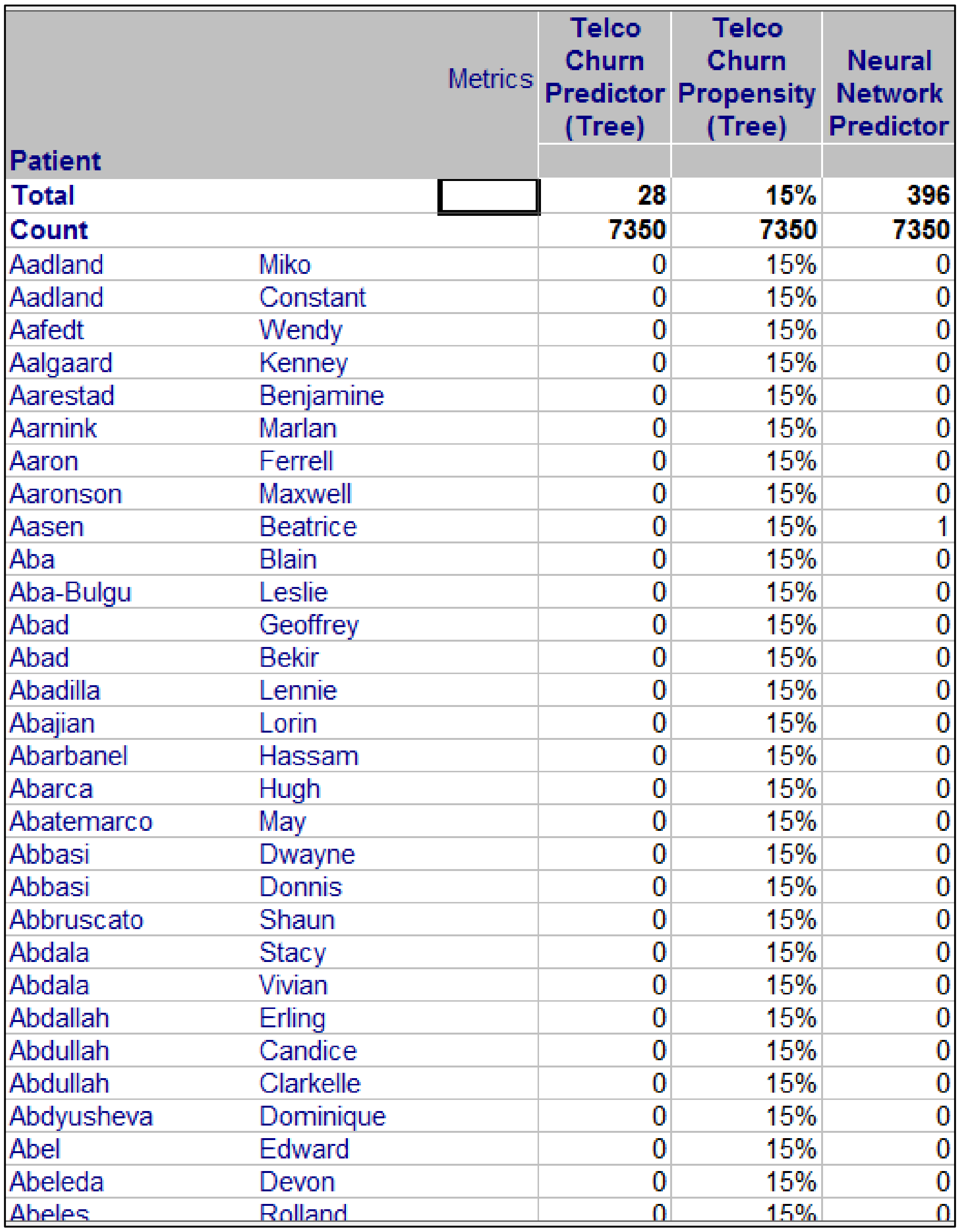
Metrics / Customer	Household Count	Income Bracket	DroppedCalls	HelpdeskCalls	TelcoChurn	Training metric for 'Telco Churn Predictor'	Neural Network Trainer	DT Correct?	NN Correct
546	6	20K and Under	0	0	0	0	0	Yes	Yes
553	5	70K-80K	1	0	0	0	0	Yes	Yes
560	5	20K-30K	2	0	0	0	0	Yes	Yes
567	3	20K and Under	0	0	0	0	0	Yes	Yes
574	1	70K-80K	0	0	0	0	0	Yes	Yes
581	3	50K-60K	1	1	0	0	0	Yes	Yes
588	4	30K-40K	0	0	1	0	0	No	No
595	4	40K-50K	0	0	0	0	0	Yes	Yes
602	3	30K-40K	0	1	0	0	0	Yes	Yes
609	3	50K-60K	0	0	0	0	0	Yes	Yes
616	2	80K-90K	0	0	0	0	0	Yes	Yes
623	5	40K-50K	1	1	1	1	1	Yes	Yes
630	4	20K and Under	0	0	0	0	0	Yes	Yes
637	3	70K-80K	0	0	0	0	0	Yes	Yes
644	5	40K-50K	0	0	1	1	1	Yes	Yes
651	3	60K-70K	0	1	0	0	0	Yes	Yes
658	3	30K-40K	0	0	0	0	0	Yes	Yes
665	1	30K-40K	0	0	0	0	0	Yes	Yes
672	1	40K-50K	0	0	0	0	0	Yes	Yes
679	1	60K-70K	1	0	0	0	0	Yes	Yes
686	1	50K-60K	0	0	0	0	0	Yes	Yes
693	4	50K-60K	2	1	0	0	0	Yes	Yes
700	3	80K-90K	0	0	0	0	0	Yes	Yes
707	Over 6	70K-80K	3	1	0	0	1	Yes	No
714	5	40K-50K	1	0	1	1	1	Yes	Yes
721	Over 6	50K-60K	3	1	0	0	0	Yes	Yes
728	4	40K-50K	0	0	0	0	0	Yes	Yes
735	3	20K and Under	0	0	0	0	0	Yes	Yes
742	2	50K-60K	0	0	0	0	0	Yes	Yes
749	2	50K-60K	0	2	0	0	0	Yes	Yes
756	5	30K-40K	0	0	0	0	1	Yes	No

:

As you can see, the Neural Network model does a poorer job than the native decision tree of classifying customers in this dataset by correctly classifying only 1,192 out of 1,428 customers. The native decision tree correctly classified 1,235 customers. However, since this model still appears relatively strong, we can score the model against the active customers that we have. In this case, we utilize the metric expression for scoring to create a metric called Neural Network Predictor:

```
RScript<[BooleanParam9]=FALSE,[NumericParam1]="3",[NumericParam2]="42",
[StringParam9]="NeuralNetwork",[_OutputVar]="ClassId",[_RScriptFile]="NeuralNetwork.r"
>(TelcoChurn, [Household Count ID], [Income Bracket ID], HelpdeskCalls, DroppedCalls)
```

We then place the Neural Network Predictor on the report "4—Telco Churn Predictor (Tree)". Here is the scoring report:



Patient	Metrics	Telco Churn Predictor (Tree)	Telco Churn Propensity (Tree)	Neural Network Predictor
Total		**28**	**15%**	**396**
Count		**7350**	**7350**	**7350**
Aadland	Miko	0	15%	0
Aadland	Constant	0	15%	0
Aafedt	Wendy	0	15%	0
Aalgaard	Kenney	0	15%	0
Aarestad	Benjamine	0	15%	0
Aarnink	Marlan	0	15%	0
Aaron	Ferrell	0	15%	0
Aaronson	Maxwell	0	15%	0
Aasen	Beatrice	0	15%	1
Aba	Blain	0	15%	0
Aba-Bulgu	Leslie	0	15%	0
Abad	Geoffrey	0	15%	0
Abad	Bekir	0	15%	0
Abadilla	Lennie	0	15%	0
Abajian	Lorin	0	15%	0
Abarbanel	Hassam	0	15%	0
Abarca	Hugh	0	15%	0
Abatemarco	May	0	15%	0
Abbasi	Dwayne	0	15%	0
Abbasi	Donnis	0	15%	0
Abbruscato	Shaun	0	15%	0
Abdala	Stacy	0	15%	0
Abdala	Vivian	0	15%	0
Abdallah	Erling	0	15%	0
Abdullah	Candice	0	15%	0
Abdullah	Clarkelle	0	15%	0
Abdyusheva	Dominique	0	15%	0
Abel	Edward	0	15%	0
Abeleda	Devon	0	15%	0
Abeles	Rolland	0	15%	0

As can be seen, the Neural Network algorithm identifies 396 customers who are likely to churn, as opposed to the decision tree which only identifies 28. With knowledge of the customers who are at-risk, the Telco company can now intelligently target and assuage the correct customers with promotions, combating the escalation in customer churn that is posing a threat to the organization.

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.wikipedia.org/wiki/Neural_network