Random Forests for Classification

Random Forest is an advanced classification technique wherein the training dataset is used to construct many independent decision trees. Every record is then passed into each individual decision tree for classification, and the class that is predicted by the majority of those decision trees is returned as the predicted class for that record.

Since the Random Forest algorithm relies on the creation of many individual trees, it is considered an ensemble model. By aggregating many individual decision trees into a single forest, we are typically able to return a better result than basic classifiers like decision trees and logistic regression.

For more detailed information on Random Forests, please consult this site [3].

Random Forests 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 RandomForest.R file from the R Script Shelf [2].
2) From the R console, run the RandomForest.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="RandomForest.r",_InputNames="Target,Vars",StringParam9="R
andomForest",BooleanParam9=TRUE,NumericParam1=750,NumericParam2=3,NumericParam3
=42>(Target, Vars)\
```

   For scoring, use this metric expression:

```
RScript<_RScriptFile="RandomForest.r",_InputNames="Target,Vars",StringParam9="R
andomForest",BooleanParam9=FALSE,NumericParam1=750,NumericParam2=3,NumericParam
3=42>(Target, Vars)
```

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

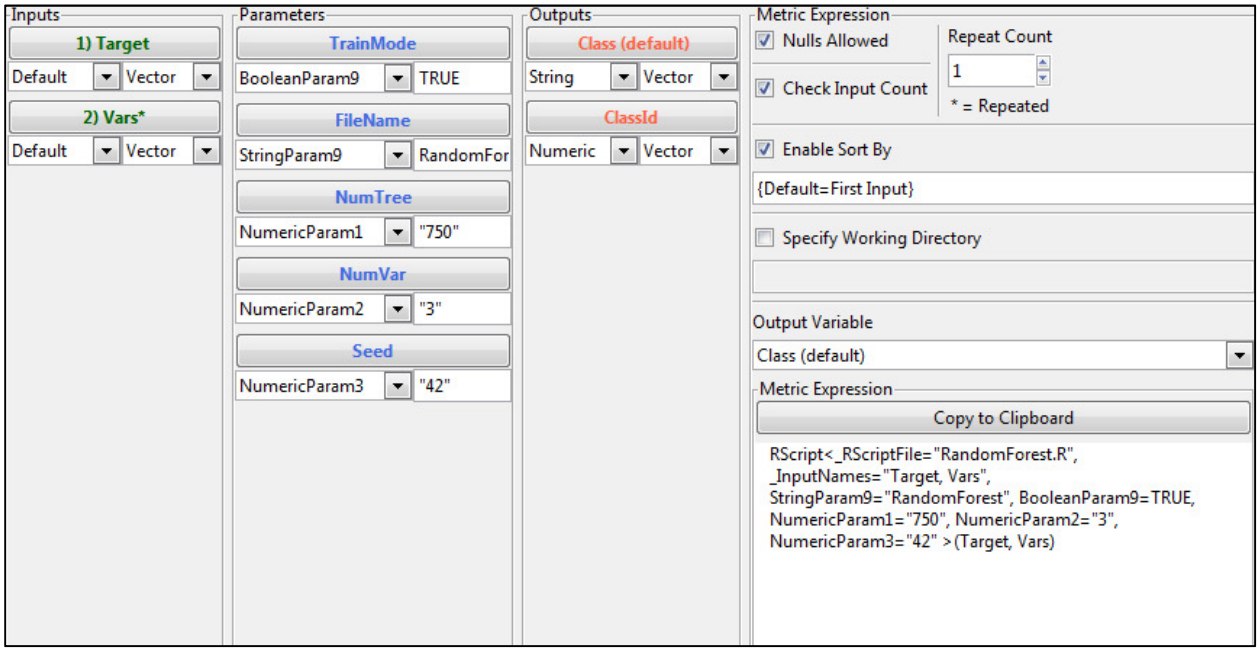
   For training, use this metric expression:


```
RScript<_RScriptFile="RandomForest.r",_InputNames="Target,Vars",[_OutputVar]="C
lassId",StringParam9="RandomForest",BooleanParam9=TRUE,NumericParam1=750,Numeri
cParam2=3,NumericParam3=42>(Target, Vars)
```

For scoring, use this metric expression:

```
RScript<_RScriptFile="RandomForest.r",_InputNames="Target,Vars",[_OutputVar]="C
lassId",StringParam9="RandomForest",BooleanParam9=FALSE,NumericParam1=750,Numer
icParam2=3,NumericParam3=42>(Target, Vars)
```

Analytic Signature:



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 Random Forests 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 RandomForest 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 "RandomForest". 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.

NumTree: Uses NumericParam1 with a default of 500. This parameter controls the number of trees that are constructed when training the model. In general, the more trees, the more accurate the results, but obviously more trees results in slower performance.

NumVars: Uses NumericParam2 with a default of 3. This parameter controls the number of variables that are randomly considered when constructing each tree**.**

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 Random Forest 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("RandomForest", repos="http://cran.rstudio.com/")
```

- **RandomForest**: This R package contains functions that support the creation of the Random Forest 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 RandomForest, then there must be a file RandomForest.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 Random Forest 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 Random Forest Trainer with the following definition:

```
RScript<[BooleanParam9]=True,[NumericParam1]="750",[NumericParam2]="3",[NumericParam3]
="42",[StringParam9]="randomforest",[_OutputVar]="ClassId",[_RScriptFile]="RandomFores
t.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 Random Forest Trainer metric to the report. By adding this metric to the report, we train the model and store it in a file called randomforest.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'	Random Forest Trainer	DT Correct?	RF Correct
126	2	60K-70K	0	0	1	0	0	No	No
133	6	50K-60K	2	2	1	1	1	Yes	Yes
140	3	60K-70K	0	1	0	0	0	Yes	Yes
147	5	40K-50K	0	1	1	1	1	Yes	Yes
154	3	80K-90K	1	1	0	0	0	Yes	Yes
161	2	70K-80K	0	1	0	0	0	Yes	Yes
168	1	40K-50K	0	0	0	0	0	Yes	Yes
175	5	40K-50K	1	2	1	1	1	Yes	Yes
182	2	50K-60K	0	0	0	0	0	Yes	Yes
189	2	40K-50K	0	1	0	0	0	Yes	Yes
196	1	50K-60K	0	0	0	0	0	Yes	Yes
203	5	20K and Under	0	0	0	0	0	Yes	Yes
210	3	50K-60K	0	0	1	0	0	No	No
217	1	20K and Under	0	0	0	0	0	Yes	Yes
224	2	70K-80K	0	0	0	0	0	Yes	Yes
231	1	50K-60K	0	1	1	0	0	No	No
238	1	60K-70K	0	0	0	0	0	Yes	Yes
245	4	60K-70K	0	0	0	0	0	Yes	Yes
252	3	20K-30K	0	0	0	0	0	Yes	Yes
259	3	40K-50K	0	0	0	0	0	Yes	Yes
266	3	90K-100K	4	2	0	0	0	Yes	Yes
273	4	30K-40K	1	1	0	0	0	Yes	Yes
280	3	50K-60K	0	0	0	0	0	Yes	Yes
287	2	60K-70K	0	0	0	0	0	Yes	Yes
294	1	80K-90K	0	0	0	0	0	Yes	Yes
301	3	40K-50K	0	0	0	0	0	Yes	Yes
308	3	40K-50K	2	0	0	0	0	Yes	Yes
315	3	20K-30K	0	0	0	0	0	Yes	Yes
322	4	Over 100K	3	0	1	0	1	No	Yes
329	5	40K-50K	2	1	1	1	1	Yes	Yes
336	3	20K and Under	0	0	0	0	0	Yes	Yes

Execution complete

:

As you can see, the Random Forest does a better job than the native decision tree of classifying customers in this dataset by correctly classifying 1,269 out of 1,428 customers. The native decision tree correctly classified 1,235 customers. Since this model appears 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 Random Forest Predictor:

```
RScript<[BooleanParam9]=False,[NumericParam1]="750",[NumericParam2]="3",[NumericParam3
]="42",[StringParam9]="randomforest",[_OutputVar]="ClassId",[_RScriptFile]="RandomFore
st.r">(Revenue, [Household Count ID], [Income Bracket ID], HelpdeskCalls,
DroppedCalls)
```

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

Customer	Metrics	Telco Churn Predictor (Tree)	Telco Churn Propensity (Tree)	RF Scoring
Total		**28**	**15%**	**197**
Count		**7350**	**7350**	**7350**
Aaronson	Maxwell	0	15%	0
Abarca	Hugh	0	15%	0
Abelson	Hazel	0	15%	0
Abern	Brooks	0	15%	0
Abram	Ross	0	15%	0
Addison	Don	0	15%	0
Adess	Merrell	0	15%	0
Adler	Keith	0	15%	0
Aguilar	Daniel	0	15%	0
Aguirre	Deborah	0	15%	0
Ahern	Dean	0	15%	0
Alcaraz	Dorthy	0	15%	0
Aldo	Sarah	0	15%	0
Alexander	Maxwell	0	15%	0
Alonso	Mario	0	15%	0
Alvarado	Calbert	0	15%	0
Amsov	Belinda	0	15%	0
Angelina	Kent	0	15%	0
Anovitz	Brent	0	15%	0
Anthony	Nicholas	0	15%	0
Applebaum	Brock	0	15%	0
Ardini	Laurell	0	15%	0
Arenburg	Saul	0	15%	0
Armstrong	Jean	0	15%	0
Arrezola	Veda	0	15%	0
Ascher	Mara	0	15%	0
Astorino	Gilbert	0	15%	0
Aya	Rebecca	0	15%	0
Azzi	Stacy	0	15%	0
Bagleman	Sherwood	0	15%	0

As can be seen, the Random Forest algorithm identifies 197 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/Random_forest

