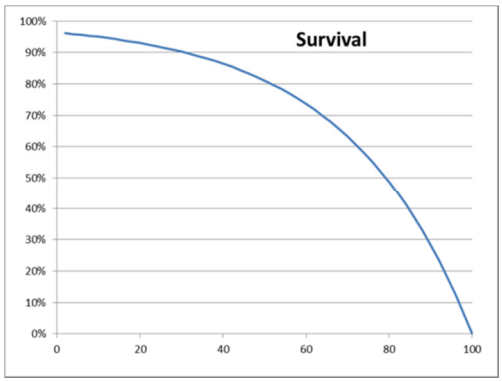
Survival Analysis using Cox Regression

"*On a long enough timeline, the survival rate for everything drops to zero*" –Tyler Durden

In the domain of Survival Analysis, the Cox Proportional Hazards model is a commonly used technique that calculates the relative risk of an event occurring as a function of any number of variables. It's called Survival Analysis because the "event" typically represents the end of something, such as a component failure, a customer being lost, or any other type of "end of life." The Cox Regression model quantifies the effect that each independent variable has on the Survival Ratio, or the likelihood that an event, assuming it has not occurred yet, will occur at any point in time.



Survival Ratio:
Greater than One → Sooner than Average
Equal to One → Average
Less than One → Later than Average

$$HR_{i-vs-Avg} = e^{(\beta_1 x_{i1}+\ldots+\beta_k x_{ik})-Avg_{j=1}^{n}\left[\beta_1 x_{j1}+\ldots+\beta_k x_{jk}\right]}$$

The Cox Regression model is considered a semi-parametric model, as it does not assume an underlying distribution. For more detailed information on Cox Regression, please consult section 3 of the paper entitled Cox Proportional-Hazards Regression for Survival Data" [3].

This R Script has two functional modes:

- Training creates a model and persists it in an .Rdata file while returning it's predictions.
- Scoring uses the model created during training to make predictions on a new dataset.

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 SurvivalAnalysis.R file from the R Script Shelf [2].
2) From the R console, run the SurvivalAnalysis.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) **Risk:** For each record, returns the risk of an event occurring relative to the average. For instance, a value of 120% means that an event is 20% more likely to occur to this record than a record who had the average values for each independent variable.

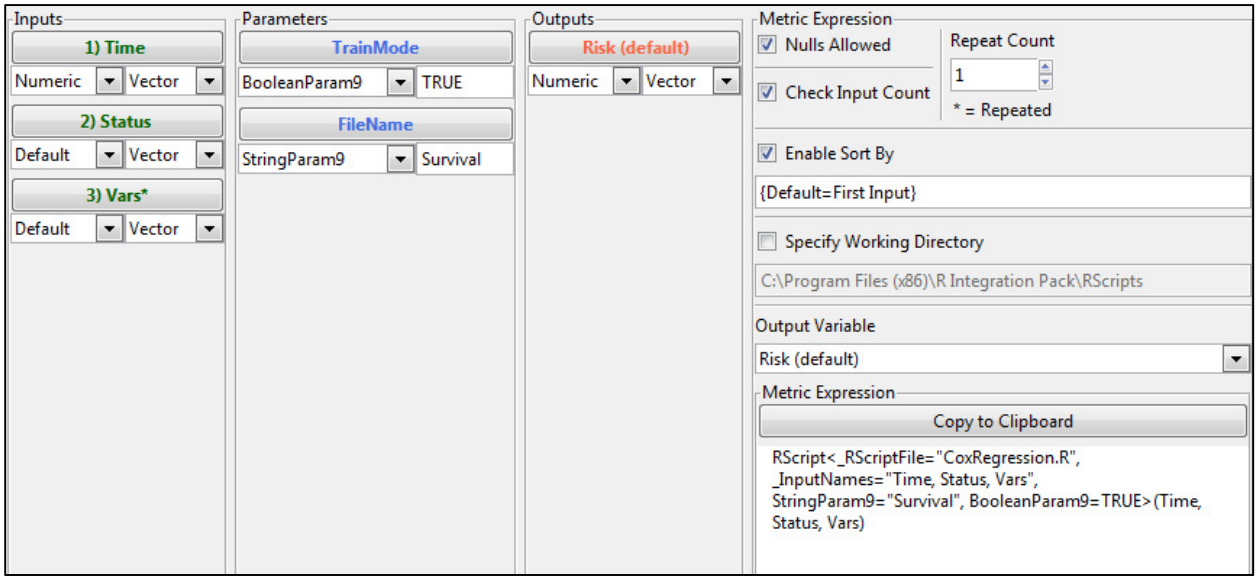
   For training, use this metric expression:

```
RScript<_RScriptFile="SurvivalAnalysis.R",_InputNames="Time,Status,Vars",
StringParam9="Survival", BooleanParam9=TRUE>(Time, Status, Vars)
```

   For scoring, use this metric expression:

```
RScript<_RScriptFile="SurvivalAnalysis.R",_InputNames="Time,Status,Vars",
StringParam9="Survival", BooleanParam9=FALSE>(Time, Status, Vars)
```

Analytic Signature:



Inputs:

Time: The amount of time that it took for the event to occur. If Status=0, then the event has not occurred and the time should be either 0 or null.

Status: A variable that represents whether or not the event occurred. Usually, this metric should be 0/1, meaning that the value is 0 if the event did not occur and is 1 if the event did occur.

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

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 Cox Regression 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 "Survival". 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.

Outputs:

Risk: For each record, returns the risk of an event occurring relative to the average. For instance, a value of 120% means that an event is 20% more likely to occur to this record than a record which had the average values for each independent variable.

Additional Results Generated by the R Script:

Two files are 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 dfTrain (a data frame containing the data read in from MicroStrategy used to train the model), model (the Cox Regression model object), and Risk (the relative risk of each record). This file is loaded when scoring new records to reference the model that was trained based on past history.

PMML Model: An XML representation of the exponential portion of the Cox Regression model, a regression equation represented using the PMML standard. This model can be imported into a predictive metric using MicroStrategy Developer. This predictive metric should be placed inside another metric for deployment to other reports and documents. For example, if the predictive metric created by importing the PMML is called "**Cox Prediction (Imported)**", then this metric expression will return the predictions:

```
Exp(([Cox Prediction (Imported)] - Avg([Cox Prediction (Imported)]) {} ))
```

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

- **survival**: This R package contains functions that support Survival Analysis, including the creation of the Cox Regression model used to calculate the relative risk of an event occurring.

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 Survival, then there must be a file Survival.Rdata in the working directory. Please consult the R Integration Pack User Guide [1] for more information. Additionally, please make sure that the working directory is a writable location.

Example (using MicroStrategy Tutorial Project):

In this example, a Telecommunications company builds a Survival model in an effort reduct the number of customers who are leaving, also known as Churn. By proactively identifying the most at-risk customers, the organization can implement different measures aimed at assuaging these disgruntled customers.

The first step in building our model is creating a metric that captures the time it took for each of our customers to churn. This metric's value should be 0 or null for all customers who have yet to churn. The following metric expression meets both of the requirements:

```
IF((TelcoChurn = 1),(((Month(CancelDate)+(12 * Year(CancelDate)))- Month(StartDate)) -
(12 * Year(StartDate))), 0)
```

A metric with this definition named Months Before Cancellation can be found in Tutorial -> Public Objects -> Reports-> MicroStrategy Platform Capabilities -> Imported PMML -> Telco Churn -> Cox Regression.

Using historical information about the characteristics of customers, whether they've churned or not, and how long it took them to churn, we'll create a metric Cox Regression Trainer with the following definition:

```
RScript<[BooleanParam9]=True,[StringParam9]="Survival",[_RScriptFile]="SurvivalAnalysi
s.R",[_InputNames]="Months,Churn,HHCount,IB,Dropped,Helpdesk">([Months Before
Cancellation],TelcoChurn, [Household Count ID], [Income Bracket ID], DroppedCalls,
HelpdeskCalls)
```

Where Time is mapped to Months Before Cancellation, Status is mapped to TelcoChurn and Vars consists of the metrics Household_Count_ID, Income_Bracket_ID, HelpdeskCalls, and DroppedCalls.

Then, create a report with Customer on rows, TelcoChurn on the columns and add the filter where Customer ID is less than 5000. Then, add the Cox Regression Trainer metric to the report. By adding this metric to the report, we train the model and store it in a file called Survival.Rdata. Here is how the report should look after formatting for our Cox Regression Trainer metric is set to Percentage and we sort by our Cox Regression Trainer metric:

Report details

Report Filter:
Customer (ID) < 5000

		Metrics	TelcoChurn	Cox Regression Trainer
Customer				
Laracuente	Tish		0	107.90%
Aasen	Beatrice		0	107.75%
Aberle	Dargie		0	107.66%
Antic	Jennett		0	107.55%
Lorenzana-Romero	Wayne		1	107.54%
Swift	Gerry		1	107.47%
Hicks	Tralene		1	107.35%
Africano	Benedict		0	107.30%
Haring	Dorita		0	107.21%
Slosson	Kiyohiko		0	107.19%
Acuna	Andre		0	107.08%
Phrasathane	Steph		0	106.98%
Bergan	Fannie		0	106.89%
Kern	Buck		1	106.83%
Fox	Leonid		0	106.80%
Rainbolt	Nadine		1	106.69%
Leech	Caesar		0	106.68%
Phillips	Don		0	106.67%
Stone	Lena		0	106.67%
Hegewald	Julianne		0	106.66%
Purinton	Gini		1	106.61%
Jalbert	Hannah		0	106.57%
Marl	Margret		0	106.57%
Newburg	Ola		0	106.57%
Behar	Terrell		0	106.57%
Caldwell	Wes		1	106.56%
Reed	Berry		1	106.56%
Teitelbaum	Greg		0	106.55%
Mcmichael	Melanie		0	106.49%
Haines	Marian		1	106.46%
Brookshire	Eron		0	106.46%
Wanvig	Zanna		1	106.38%

Now that we have trained our model, 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 Cox Regression:

```
RScript<[BooleanParam9]=False,[StringParam9]="Survival",
[_RScriptFile]="CoxRegressionPredictor.R",[_InputNames]="M
onths,Churn,HHCount,IB,Dropped,Helpdesk">([Months Before
Cancellation],TelcoChurn, [Household Count ID], [Income
Bracket ID], DroppedCalls, HelpdeskCalls)
```

We then create a report with Customer on Rows and a filter where Customer ID is greater than or equal to 5000 and place the Cox Regression Predictor on this report. This is what the report looks like after we sort on Cox Regression Predictor descending and change its formatting to percentage:

Report details

Report Filter:
Customer (ID) >= 5000

Customer		Metrics	Cox Regression Predictor
Murphy	Humberto		107.80%
Robare	Henry		107.70%
Holcombe	Ike		107.65%
Tews	Lilly		107.55%
Adjei	Griffith		107.53%
Stevens	Oriana		107.44%
Kallies	Antonio		107.39%
Terry	Jojeana		106.84%
Balchen	Donnie		106.72%
Vaness	Dorris		106.72%
Abrom	Octavia		106.67%
Camarena	Skeets		106.67%
Vogt	Kayce		106.67%
Walsh	Kenzo		106.67%
Pettitt	Kourosh		106.62%
Zalonis	Florence		106.62%
Frazier	Jackson		106.61%
Dusenbury	Audrey		106.60%
George	Richard		106.60%
Bryce	Valinda		106.58%
Kinney	Gladys		106.58%
Koopat	Michael		106.58%
Millan	Acasio		106.58%
Maras	Angela		106.57%
Holston	Aimee		106.55%

By sorting on the customers with the highest Relative Risk as identified by our Cox Regression model, the organization can focus its efforts on assuaging the most at-risk customers rather than focusing on customers who are likely to stay regardless if they are reached out to.

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) Cox Proportional-Hazards Regression for Survival Data
   http://cran.r-project.org/doc/contrib/Fox-Companion/appendix-cox-regression.pdf