Evaluating performance of regression machine learning models using multiple error metrics in Azure Machine Learning Studio

Alexei Botchkarev

Abstract

Data driven companies effectively use regression machine learning methods for making predictions in many sectors. Cloud-based Azure Machine Learning Studio (MLS) has a potential of expediting machine learning experiments by offering a convenient and powerful integrated development environment. The process of evaluating machine learning models in Azure MLS has certain limitations, e.g. small number of performance metrics and lack of functionality to evaluate custom built regression models with R language. This paper reports the results of an effort to build an Enhanced Evaluate Model (EEM) module which facilitates and accelerates Azure experiments development and evaluation. The EEM combines multiple performance metrics allowing for multisided evaluation of the regression models. EEM offers 4 times more metrics than built-in Azure Evaluate Model module. EEM metrics include: CoD, GMRAE, MAE, MAPE, MASE, MdAE, MdAPE, MdRAE, ME, MPE, MRAE, MSE, NRMSE_mm, NRMSE_sd, RAE, RMdSPE, RMSE, RMSPE, RSE, sMAPE, SMdAPE, SSE. Also, EEM supports evaluation of the R language based regression models. The operational Enhanced Evaluate Model module has been published to the web and openly available for experiments and extensions.

Keywords: machine learning, regression, performance metrics, accuracy, error, error measure, multiple types, models, prediction, evaluation, forecast, Azure Machine Learning Studio, R, R package.

JEL Classification: C1, C4, C52, C53, C6

Introduction

Regression machine learning algorithms have been proven effective for making predictions in many sectors. Several large data driven companies developed cloud based platforms offering Machine Learning as a Service (MLaaS): e.g. Amazon Web Services (AWS), Microsoft Azure, Google Cloud.

Microsoft has rolled out an Azure Machine Learning Studio (https://studio.azureml.net) which has a potential of expediting machine learning experiments from weeks and months to hours and days. Azure MLS offers a convenient integrated development environment with certain benefits, including: cloud-based machine learning offered as a service; web-based solution – user needs browser only to work with the system; easy to use drag and drop canvas interface; several ready to use built-in regression modules; flexibility of using R and Python languages to code experiments; ability to integrate functions from R packages; capability to re-use published experiments or their components.

One of the key phases in machine learning experiments is evaluation of the model. The purpose of the evaluation is to compare the trained model predictions with the actual (observed) data from the testing data set. Azure MLS has a designated module – Evaluate Model – to perform comparisons. This module shares convenience of drag and drop functionality with other Azure

modules. However, it has certain limitations. For example, it has a limited number of the metrics implemented to evaluate prediction errors (Evaluate Model 2018):

- Mean absolute error (MAE).
- Root mean squared error (RMSE).
- Relative absolute error (RAE.
- Relative squared error (RSE).
- Coefficient of determination (CoD).

It should be noted that recently, in spring of 2018, Microsoft added another performance metric to evaluate regressions: Mean Zero One Error (MZOE) which indicates whether the prediction was correct or not, i.e. ZeroOneLoss(x, y) = 1 when x != y; 0 otherwise. This metric is intended for use with ordinal regressions (e.g. Yu, Yu, Tresp & Kriegel 2006; Wang & King 2010) and is not considered in this paper.

Another Evaluate Model module limitation is that it can be used only with Azure MLS built-in regression models and cannot be used, if the experiment includes models developed using R language.

The purpose of this paper was to develop an Enhanced Evaluate Model module, based on the R language that would alleviate aforementioned problems.

Objective

The purpose of this experiment was to develop an Enhanced Evaluate Model module, based on the R language that would improve performance of the built-in Azure MLS Evaluate Model module. Specifically:

- Develop Enhanced Evaluate Model module that would enable evaluation of regression models with multiple performance metrics (22 compared to five in the built-in Azure module).
- Develop Enhanced Evaluate Model module that would perform evaluations of models implemented with R language using Azure “Create R Model” (function not available now in Azure) as well as combining them with evaluations of the Azure built-in regression models.

Overview of the experiment

The focus of this experiment is on the 'Enhanced Evaluate Model' module. However, for easier understanding of the module operation and potential reuse, it is shown in a typical Azure regression experiment workflow, i.e. input data, initialize model, train model, score model, evaluate model. The experiment workflow is shown in Fig 1. The experiment has been published to the web and openly available for experiments and extensions (Botchkarev, 2018a).

A component of the previously published experiment is used as a sample (Botchkarev, 2018b). Details of this experiment are described by Botchkarev (2018c).

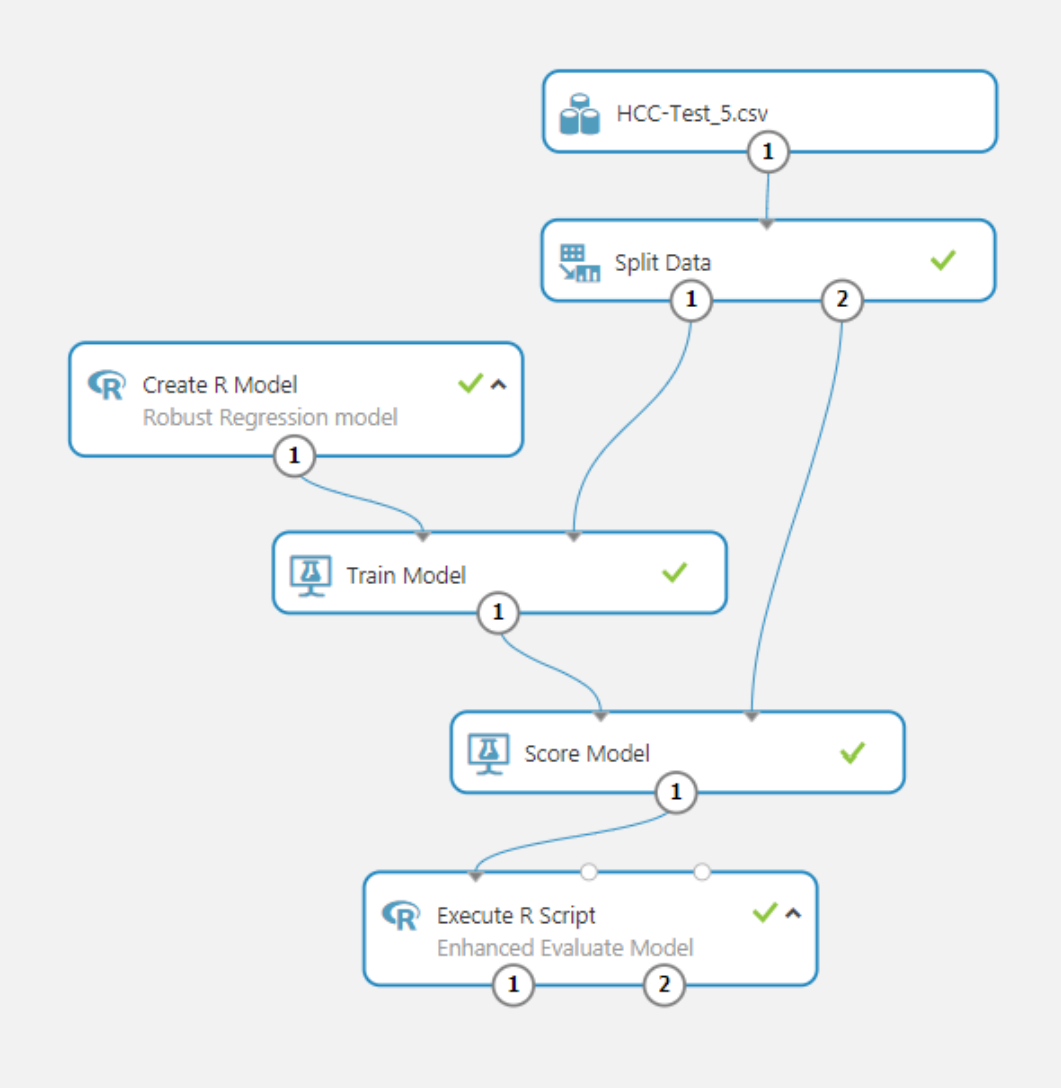

Fig 1 Experiment Workflow

To adopt the model in your experiment you need to use input from the 'Score Model' (dataset1) to create vectors (one-column data frames) for actual and predicted values. Check column names by visualizing output of the 'Score Model'. Column names for predicted variable may differ depending on which regression model is used, e.g. 'Scored Label Mean' for Decision Forest Regression, or 'Scored Labels' for Linear regression. Rename the column in the input data frame with target/label variable values ('Cost' in the example) to have a new name - 'Actual'. Similarly, rename the column in the input data frame with predicted variable values ('Predicted Cost' in the example) to have a new name - 'Predicted'. See hints in the R script of the 'Enhanced Evaluate Model'. After that, copy 'Enhanced Evaluate Model', paste it into your experiment canvas, and connect input of this module to the output of the 'Score Model'.

To see the output of the experiment, right click on the left output port (1) of the Enhanced Evaluate Model module. Select Visualize option from a drop-down menu. The output is presented in a table with three columns: metric abbreviation, metric full name and numerical estimate. Another option is to attach a Convert to CSV module to the Enhanced Evaluate Model

module and download output file. Note that results are rounded to four digits after the decimal point. A sample of the output table is presented in Appendix 1.

Performance metrics

Twenty two commonly used performance metrics were selected for the experiment. They are listed in Table 1 in alphabetical order of metric abbreviation.

Table 1 List of Performance Metrics implemented in the Enhanced Evaluate Model

Metric	Metric Name
CoD	Coefficient of Determination
GMRAE	Geometric Mean Relative Absolute Error
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MASE	Mean Absolute Scaled Error
MdAE	Median Absolute Error
MdAPE	Median Absolute Percentage Error
MdRAE	Median Relative Absolute Error
ME	Mean Error
MPE	Mean Percentage Error
MRAE	Mean Relative Absolute Error
MSE	Mean Squared Error
NRMSE_mm	Normalized Root Mean Squared Error (normalized to the difference between maximum and minimum actual data)
NRMSE_sd	Normalized Root Mean Squared Error (normalized to the standard deviation of the actual data)
RAE	Relative Absolute Error
RMdSPE	Root Median Square Percentage Error
RMSE	Root Mean Squared Error
RMSPE	Root Mean Square Percentage Error
RSE	Relative Squared Error

sMAPE	Symmetric Mean Absolute Percentage Error
SMdAPE	Symmetric Median Absolute Percentage Error
SSE	Sum of Squared Error

Mathematical definitions of the metrics implemented in the Enhanced Evaluate Model module are provided in Appendix 2. The R script of the developed module is presented in Appendix 3. Descriptions of the metrics can be found in recent papers, e.g. (Alencar, 2017; Chen, Twycross & Garibaldi, 2017; Davydenko & Fildes, 2016; Hyndman & Koehler, 2006; Kim & Kim, 2016; Shcherbakov et al, 2013).

Concluding Remarks

An Enhanced Evaluation Model (EEM) module has been developed which facilitates and accelerates Azure experiments development and evaluation.

EEM combines multiple performance metrics allowing for multisided evaluation of the regression models. EEM offers 4 times more metrics than built-in Azure Evaluate Model module. EEM metrics include: CoD, GMRAE, MAE, MAPE, MASE, MdAE, MdAPE, MdRAE, ME, MPE, MRAE, MSE, NRMSE_mm, NRMSE_sd, RAE, RMdSPE, RMSE, RMSPE, RSE, sMAPE, SMdAPE, SSE.

EEM allows users evaluate R language based regression models developed with Create R Model module (a function which is not supported with the Azure built-in Evaluate Model module).

Evaluate Model module has been published to the web and openly available for experiments and extensions (Botchkarev, 2018a).

Note that the operation of the module assumes that all data preparation has been done: there are no NA (missing) data elements, all data are numeric.

Note that some metrics perform division operation and may return error messages, if denominator is equal or close to zero.

Note that the module is publicly shared for information and training purposes only. All efforts were taken to make this module error-free. However, we do not guarantee the correctness, reliability and completeness of the material and the module is provided "as is", without warranty of any kind, express or implied. Any user is acting entirely at their own risk.

Note that the views, opinions and conclusions expressed in this document are those of the author alone and do not necessarily represent the views of the author’s current or former employers.

References

Alencar, D., Carvalho, D., Koenders, E., Mourao, F., & Rocha, L. (2017). Devising a computational model based on data mining techniques to predict concrete compressive strength. Procedia Computer Science, 108, 455-464.

Botchkarev, A. (2018a). Enhanced model evaluation with multiple performance metrics for regression analysis. Experiment in Microsoft Azure Machine Learning Studio. Azure AI Gallery. *https://gallery.azure.ai/Experiment/Enhanced-model-evaluation-with-multiple-performance-metrics-for-regression-analysis*

Botchkarev, A. (2018b). Revision 2. Integrated tool for rapid assessment of multi-type regression machine learning models. Experiment in Microsoft Azure Machine Learning Studio. Azure AI Gallery. *https://gallery.azure.ai/Experiment/Revision-2-Integrated-tool-for-rapid-assessment-of-multi-type-regression-machine-learning-models*

Botchkarev, A. (2018c). Evaluating Hospital Case Cost Prediction Models Using Azure Machine Learning Studio. arXiv:1804.01825 [cs.LG].

Chen, C., Twycross, J., & Garibaldi, J. M. (2017). A new accuracy measure based on bounded relative error for time series forecasting. PloS one, 12(3), e0174202. *https://doi.org/10.1371/journal.pone.0174202*

Davydenko, A., & Fildes, R. (2016). Forecast error measures: critical review and practical recommendations. *Business Forecasting: Practical Problems and Solutions. Wiley*, *34*.

Evaluate Model. (2018). Azure Machine Learning Studio. *https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/evaluate-model#bkmk_regression*

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International journal of forecasting*, *22*(4), 679-688.

Kim, S., & Kim, H. (2016). A new metric of absolute percentage error for intermittent demand forecasts. *International Journal of Forecasting*, *32*(3), 669-679.

Shcherbakov, M. V., Brebels, A., Shcherbakova, N. L., Tyukov, A. P., Janovsky, T. A., & Kamaev, V. A. E. (2013). A survey of forecast error measures. *World Applied Sciences Journal*, *24*, 171-176.

Wang, D., & King, I. (2010, November). An Enhanced Semi-supervised Recommendation Model Based on Green's Function. In *International Conference on Neural Information Processing* (pp. 397-404). Springer, Berlin, Heidelberg.

Yu, S., Yu, K., Tresp, V., & Kriegel, H. P. (2006, June). Collaborative ordinal regression. In *Proceedings of the 23rd international conference on Machine learning* (pp. 1089-1096). ACM.

Appendix 1 Experiment Output Sample Table
Enhanced Evaluate Model – Final > Execute R Script > Result Dataset
Rows 22 columns 3

metric_name	full_name	estimate
CoD	Coefficient of Determination	0.7189
GMRAE	Geometric Mean Relative Absolute Error	0.0067
MAE	Mean Absolute Error	254.5004
MAPE	Mean Absolute Percentage Error	3.0518
MASE	Mean Absolute Scaled Error	0.1011
MdAE	Median Absolute Error	7.335
MdAPE	Median Absolute Percentage Error	0.4794
MdRAE	Median Relative Absolute Error	0.0065
ME	Mean Error	233.4515
MPE	Mean Percentage Error	1.0185
MRAE	Mean Relative Absolute Error	0.0518
MSE	Mean Squared Error	4463364.6386
NRMSE_max_min	Normalized Root Mean Squared Error (norm. to max - min)	0.0313
NRMSE_sd	Normalized Root Mean Squared Error (norm. to SD)	0.5301
RAE	Relative Absolute Error	0.1369
RMdSPE	Root Median Square Percentage Error	0.048
RMSE	Root Mean Squared Error	2112.6677
RMSPE	Root Mean Square Percentage Error	1.677
RSE	Relative Squared Error	0.2811
SMAPE	Symmetric Mean Absolute Percentage Error	3.4404
SMdAPE	Symmetric Median Absolute Percentage Error	0.4799
SSE	Sum of Squared Error	15621776234.9768

Appendix 2 Mathematical definitions of the performance metrics

Metric Abbreviation	Metric Name	Metric Formula
		Legend: A_j – actual values $\bar{A}$ – the mean of the actual values P_j – predicted values $e_j = A_j - P_j$ – error n – size of the data set
ME	Mean Error	$$ME = \frac{1}{n}\sum_{j=1}^{n} e_j$$
MAE	Mean Absolute Error	$$MAE = \frac{1}{n}\sum_{j=1}^{n} \mid e_j \mid$$
MdAE	Median Absolute Error	$$MdAE = Md(\mid e_j \mid)$$
SSE	Sum of Squared Error	$$SSE = \sum_{j=1}^{n} e_j^2$$
MSE	Mean Squared Error	$$MSE = \frac{1}{n}\sum_{j=1}^{n} e_j^2$$
RMSE	Root Mean Squared Error	$$RMSE = \sqrt{\frac{\sum_{j=1}^{n} e_j^2}{n}}$$ or $$RMSE = \sqrt{MSE}$$
NRMSE_sd	Normalized Root Mean	

	Squared Error (normalized to the standard deviation of the actual data)	$NRMSE = \frac{RMSE}{sd}$ $sd = \sqrt{\frac{\sum_{j=1}^{n}(A_j - \bar{A})^2}{n-1}}$
NRMSE_mm	Normalized Root Mean Squared Error (to the difference between maximum and minimum actual data)	$NRMSE = \frac{RMSE}{maxA - minA}$
RAE	Relative Absolute Error	$RAE = \frac{\sum_{j=1}^{n} \mid e_j \mid}{\sum_{j=1}^{n} \mid A_j - \bar{A} \mid}$
MRAE	Mean Relative Absolute Error	$MRAE = \frac{1}{n}\sum_{j=1}^{n}\frac{\mid e_j \mid}{\mid A_j - \bar{A} \mid}$
GMRAE	Geometric Mean Relative Absolute Error	$GMRAE = exp\left(\frac{1}{n}\sum_{j=1}^{n}\ln\left(\frac{\mid e_j \mid}{\mid A_j - \bar{A} \mid}\right)\right)$ Or $\sqrt[n]{\prod_{j=1}^{n}\left(\frac{\mid e_j \mid}{\mid A_j - \bar{A} \mid}\right)}$
MdRAE	Median Relative Absolute Error	$MdRAE = \underset{j=1,n}{Md}\left(\frac{\mid e_j \mid}{\mid A_j - \bar{A} \mid}\right)$
RSE	Relative Squared Error	

		$RSE = \sqrt{\frac{\sum_{j=1}^{n} e_j^2}{\sum_{j=1}^{n} \lvert A_j - \bar{A} \rvert}}$
CoD	Coefficient of Determination	$CoD = 1 - \frac{\sum_{j=1}^{n} (P_j - A_j)^2}{\sum_{j=1}^{n} (A_j - \bar{A})^2}$
MPE	Mean Percentage Error	$MPE = \frac{100}{n} \sum_{j}^{n} \frac{e_j}{A_j}$
MAPE	Mean Absolute Percentage Error	$MAPE = \frac{100}{n} \sum_{j}^{n} \frac{\lvert e_j \rvert}{\lvert A_j \rvert}$
MdAPE	Median Absolute Percentage Error	$MdAPE = 100 * \underset{j=1,n}{Md} \left(\frac{\lvert e_j \rvert}{\lvert A_j \rvert} \right)$
sMAPE	Symmetric Mean Absolute Percentage Error	$sMAPE = \frac{100}{n} \sum_{j}^{n} \frac{2 * \lvert e_j \rvert}{\lvert A_j \rvert + \lvert P_j \rvert}$
SMdAPE	Symmetric Median Absolute Percentage Error	$SMdAPE = 100 * \underset{j=1,n}{Md} \left(\frac{2 * \lvert e_j \rvert}{\lvert A_j \rvert + \lvert P_j \rvert} \right)$
RMSPE	Root Mean Square Percentage Error	$RMSPE = \sqrt{\frac{100}{n} \sum_{j}^{n} \left(\frac{\lvert e_j \rvert}{\lvert A_j \rvert} \right)^2}$
RMdSPE	Root Median Square Percentage Error	

		$RMdSPE = \sqrt{100 * \underset{j=1,n}{Md}\left(\frac{\lvert e_j \rvert}{\lvert A_j \rvert}\right)^2}$
MASE	Mean Absolute Scaled Error	$MASE = MAE / MAE_{in-sample,naïve}$ $MASE = MAE / Q,$ *where* $Q = \frac{1}{n-1}\sum_{j=2}^{n} \lvert A_j - A_{j-1} \rvert$

Appendix 3 R script of the Enhanced Evaluate Model module

```
### PERFORMANCE METRICS MODULE -- Enhanced Evaluate Model ###

# Results of Scoring as input to module
dataset1 <- maml.mapInputPort(1)

## CHANGE ## Use input from the 'Score Model' (dataset1) to create vectors (one-column data frames) for
actual and predicted values.
#### Check column names by visualizing output of the 'Score Model'.
#### Column names for predicted variable may differ depending on which regression model is used,
#### e.g. 'Scored Label Mean' for Decision Forest Regression, or 'Scored Labels' for Linear regression.
#### Rename the column in the input data frame with target/label variable values ('Cost' in the example) to
have a new name - 'Actual'.
Actual <- dataset1$Cost

## CHANGE ## Similarly, rename the column in the input data frame with predicted variable values
#### ('Predicted Cost' in the example) to have a new name - 'Predicted'.
Predicted <- dataset1["Predicted Cost"]

#Frequently used components
Error <- Actual - Predicted
squaredError <- Error^2
absError <- abs(Error)

### Calculate MAE
n <- NROW(Actual)
estMAE <- sum(abs(Error))
MAE <- estMAE/n
MAE <- data.frame(MAE)

### Calculate ME
ME <- sum(Error)/n
ME <- data.frame(ME)

### Calculate SSE
SSE <- sum(squaredError)
SSE <- data.frame(SSE)

### Calculate MSE
MSE <- sum(squaredError) * (1/n)
MSE <- data.frame(MSE)

### Calculate MdAE
absError <- as.numeric(absError[[1]])
MdAE <- median(absError, na.rm = FALSE)
MdAE <- data.frame(MdAE)

### Calculate RMSE
###sum of squared errors - used in other metrics
estMRSEnom <- sum((Error)^2)
RMSE <- sqrt(estMRSEnom/n)
RMSE <- data.frame(RMSE)
```

```
#### OPTIONAL correct Calculate NRMSE_sd
#normalizerSD <- mean((Actual-mean(Actual))^2)
#NRMSE_sd <-  sqrt(mean((Error)^2) / normalizerSD)
#NRMSE_sd <- data.frame(NRMSE_sd)

### Calculate NRMSE_sd
normalizerST_DEV <- sd(Actual)
a <- sqrt(mean((Error)^2)) ###Correct RMSE
NRMSE_sd <- a / normalizerST_DEV
NRMSE_sd <- data.frame(NRMSE_sd)

### Calculate NRMSE_mm
normalizerMAX_MIN <- max(Actual) - min(Actual)
#a <- sqrt(mean((Error)^2)) ###Correct RMSE
NRMSE_mm <- a / normalizerMAX_MIN
NRMSE_mm <- data.frame(NRMSE_mm)

### Calculate RAE
# Calulate RAE nominator: use numerator from MAE --estMAE
# Calculate RAE denominator
M <- mean(Actual)
## Create vector all members of which equal to mean of Actuals
AverageAct <- data.frame(matrix(M, ncol=1, nrow = n))
## Create dataframe of Actuals minus average of Acutals - also used in other metrics
ACTminusAverACT <- Actual - AverageAct

estRAEdenom <- sum(abs(ACTminusAverACT))
#Use numerator from MAE ---eatMAE <- sum(abs(error)))
RAE <- estMAE/estRAEdenom
RAE <- data.frame(RAE)

### Calculate MRAE
# Numerator - abs(Error) - absError
# Denominator
MRAEdenom <- abs(ACTminusAverACT)
# Calcualte Numerator / Denominator
MRAEerror <- absError / MRAEdenom
# Calculate mean of error
MRAE <- sum(MRAEerror) * (1/n)
MRAE <- data.frame(MRAE)

### Calculate MdRAE
MRAEerror <- as.numeric(MRAEerror[[1]])
### Calculate MdRAE - median
MdRAE <- median(MRAEerror, na.rm=FALSE)
MdRAE <- data.frame(MdRAE)

### Calculate GMRAE
# Use MRAE eroor >>> abs(Error) / abs(Act - AveAct) >>>  absError/MRAEdenom >>>  MRAEerror
# Calculate logar (ln) of the MRAE error
LN_MRAEerror <- log(MRAEerror)
# Calculate sum of ln(MRAE)
sumLN_MRAEerror <- sum(LN_MRAEerror)
GMRAE <- exp(sumLN_MRAEerror / n)
GMRAE <- data.frame(GMRAE)
```

```
### Calculate RSE
# Calulate RSE numerator -
### sum of squared errors -estMRSEnom  used in other metrics

#Calculate RSE denominator
estRSEdenom <- sum((ACTminusAverACT)^2)
RSE <- estMRSEnom/estRSEdenom
RSE <- data.frame(RSE)

### Calculate CoD
### calculate nominator
PredMinusAct <- Predicted - Actual
PredMinusActsquared <- PredMinusAct^2
CoDnominator <- sum(PredMinusActsquared)
### Calculate denominator
#### Use (Actual - AveActual) from RAE ACTminusAverACT
CoDdenominator <- sum((ACTminusAverACT^2))
CoD <- 1-(CoDnominator/CoDdenominator)

### Calculate MPE
MPE <- sum(Error/Actual) * (100 / n)
MPE <- data.frame(MPE)

### Calculate MAPE
#### Use absError from MAE
#### Calculate absActual
absActual <- abs(Actual)
#### Calculate absError/absActual
ErrAct <- absError/absActual
#### Calculate MAPE: sum(ErrAct) *(100/n)
MAPE <- sum(ErrAct) * (100/n)
MAPE <- data.frame(MAPE)

### Calculate MdAPE (median APE)
MdAPE <- median(ErrAct) * 100
MdAPE <- data.frame(MdAPE)

### Calculate SMAPE
SMAPEnumer <- 2 * absError
SMAPEdenom <- absActual + abs(Predicted)
SMAPE <- sum(SMAPEnumer / SMAPEdenom) * (100 / n)
SMAPE <- data.frame(SMAPE)

### Calculate SMdAPE

SMdAPEratio <- (SMAPEnumer / SMAPEdenom)
SMdAPEratio <- as.numeric(SMdAPEratio[[1]])

SMdAPE <- median(SMdAPEratio, na.rm = FALSE) * (100)
SMdAPE <- data.frame(SMdAPE)

### Calculate RMSPE
# Use ErrAct <- absError/absActual from MAPE
# Calculate ErrAct^2
```

```
ErrActP2 <- ErrAct ^ 2
# Calculate sum{ErractP2}
SumErrActP2 <- sum(ErrActP2)
# Multiply SumErrActP2 by (100/n)
SumErrActP2mean <- SumErrActP2 * (100/n)
RMSPE <- sqrt(SumErrActP2mean)
RMSPE <- data.frame(RMSPE)

### Calculate RMdSPE
RatioRMdSPE <- (absError / Predicted) ^2
RatioRMdSPE < as.numeric(RatioRMdSPE[[1]])
medRMdSPE <- median(RatioRMdSPE, na.rm=FALSE)
RMdSPE <- sqrt(medRMdSPE *100)
RMdSPE <- data.frame(RMdSPE)


### Calculate MASE
################### ADOPTED from
#  CRAN / tsensembler / R/metrics.r
# R/metrics.r In tsensembler: Dynamic Ensembles for Time Series Forecasting
# https://rdrr.io/cran/tsensembler/src/R/metrics.r
# Author            Vitor Cerqueira [aut, cre], Luis Torgo [ctb], Carlos Soares [ctb]
#' Computing the mean absolute scaled error
#'#' Mean absolute scaled error using the previous value as
#' baseline (i.e. a random walk)
#' Computing the absolute error
#'#' Element-wise computation of the absolute error loss function.
#'#' @inheritParams se
#'#' @family error/performance functions
#'#' @keywords internal
#'#' @export
ae <- function(Actual, Predicted, ...) {
  stopifnot(length(Actual) == length(Predicted),
          is.numeric(Actual),
          is.numeric(Predicted))

  abs(Actual - Predicted)
}
  len <- length(Actual)
  baseline <- c(NA_real_, Actual[-length(Actual)])
  den <- (len / (len-1)) * sum(ae(Actual, baseline), na.rm = TRUE)
    MASE <- sum(abs(Actual - Predicted)) / den
MASE <- data.frame(MASE)


estimate <- cbind(CoD, GMRAE, MAE, MAPE, MASE, MdAE, MdAPE, MdRAE, ME, MPE, MRAE, MSE,
NRMSE_mm, NRMSE_sd, RAE, RMdSPE, RMSE, RMSPE, RSE, SMAPE, SMdAPE, SSE)

 # ME, MAE, RMSE, NRMSE_sd, NRMSE_mm, RAE, MRAE, MdRAE, GMRAE, RSE, CoD, SSE, MSE, MdAE, MPE,
MAPE, MdAPE, SMAPE, SMdAPE, RMSPE, RMdSPE, MASE)

estimate <- t(estimate)
estimate <- data.frame(estimate)
estimate <- round(estimate, digits = 4)
```

```
metric_name <- data.frame(metric_name=
  c("CoD",            "GMRAE",            "MAE",   "MAPE","MASE","MdAE","MdAPE",
"MdRAE","ME","MPE","MRAE","MSE","NRMSE_max_min","NRMSE_sd","RAE","RMdSPE","RMSE","RMSPE","RS
E","SMAPE","SMdAPE","SSE"))

full_name <- data.frame(full_name=
  c("Coefficient of Determination","Geometric Mean Relative Absolute Error",              "Mean Absolute Error",
  "Mean Absolute Percentage Error",           "Mean Absolute Scaled Error",       "Median Absolute Error","Median
Absolute Percentage Error",           "Median Relative Absolute Error ",
  "Mean Error",    "Mean Percentage Error",              "Mean Relative Absolute Error",    "Mean Squared Error",
  "Normalized Root Mean Squared Error (norm. to max - min)",          "Normalized Root Mean Squared Error
(norm. to SD)",
  "Relative Absolute Error",           "Root Median Square Percentage Error",    "Root Mean Squared Error",
            "Root Mean Square Percentage Error",
  "Relative Squared Error",           "Symmetric Mean Absolute Percentage Error",        "Symmetric Median
Absolute Percentage Error",           "Sum of Squared Error"))

metric_name <- cbind(metric_name, full_name, estimate)

maml.mapOutputPort("metric_name");
```