root

Table of Contents

LinearRegression
LogisticRegression
ML_Core
PBblas

LinearRegression

Name	LinearRegression
Version	3.0.0
Description	Linear Regression Algorithm Bundle
License	http://www.apache.org/licenses/LICENSE-2.0
Copyright	Copyright (C) 2017 HPCC Systems
Authors	HPCCSystems
DependsOn	ML_Core, PBblas
Platform	6.2.0

Table of Contents

OLS

IMPORTS

ML_Core | ML_Core.Types | PBblas | PBblas.Types | PBblas.Converted | PBblas.MatUtils | ML_Core.Math |

DESCRIPTIONS

MODULE OLS

OLS
`(DATASET(NumericField) X=empty_data, DATASET(NumericField) Y=empty_data)`

Ordinary Least Squares (OLS) Linear Regression aka Ordinary Linear Regression Regression learns a function that maps a set of input data (independents) to one or more output variables (dependents). The resulting learned function is known as the model. That model can then be used repetitively to predict (i.e. estimate) the output value(s) based on new input data. Two major use cases are supported: 1) Learn and return a model 2) Use an existing (e.g. persisted) model to predict new values for Y Of course, both can be done in a single run. Alternatively, the model can be persisted and used indefinitely for prediction of Y values, as long as the record format has not changed, and the original training data remains representative of the population. OLS supports any number of independent variables (Multiple Regression) and multiple dependent variables (Multivariate Regression). In this way, multiple variables' values can be predicted from the same input (i.e. independent) data. Training data is presented as parameters to this module. When using a previously persisted model (use case 2 above), these parameters should be omitted. This module provides a rich set of analytics to assess the usefulness of the resulting linear regression model, and to determine the best subset of independent variables to include in the model. These include: For the whole model: - Analysis of Variance (ANOVA) - R-squared - Adjusted R-squared - F-Test - Akaike Information Criterion (AIC) For each coefficient: - Standard Error (SE) -

T-statistic - P-value - Confidence Interval

PARAMETER <u>Y</u> ||| TABLE (NumericField) — The dependent variable training data in DATASET(NumericField) format. Each observation (e.g. record) is identified by 'id', and each feature is identified by field number. Omit this parameter when predicting from a persisted model.

PARAMETER <u>X</u> ||| TABLE (NumericField) — The independent variable training data in DATASET(NumericField) format. Each observation (e.g. record) is identified by 'id', and each feature is identified by field number (i.e. 'number'). Omit this parameter when predicting from a persisted model.

PARENT **ML_Core.Interfaces.IRegression** <../ML_Core/Interfaces/IRegression.ecl.tex>

Children

1. GetModel : GetModel Returns the learned model that maps X's to Y's

2. Betas : Return raw Beta values as numeric fields Extracts Beta values from the model

3. Predict : Predict the dependent variable values (Y) for any set of independent variables (X)

4. makeRSQ : No Documentation Found

5. RSquared : RSquared Calculate the R-Squared Metric used to assess the fit of the regression line to the training data

6. AnovaRec : No Documentation Found

7. calcAnova : No Documentation Found

8. Anova : ANOVA (Analysis of Variance) report Analyzes the sources of variance

9. SE : Standard Error of the Regression Coefficients Describes the variability of the regression error for each coefficient

10. TStat : T-Statistic The T-statistic identifies the significance of the value of each regression coefficient

11. AdjRSquared : Adjusted R2 Calculate Adjusted R Squared which is a scaled version of R Squared that does not arbitrarily increase with the number of features

12. AICRec : No Documentation Found

13. AIC : Akaike Information Criterion (AIC) Information theory based criterion for assessing Goodness of Fit (GOF)

14. RangeVec : No Documentation Found

15. DistributionBase : No Documentation Found

16. TDistribution : No Documentation Found

17. FDistribution : No Documentation Found

18. NormalDistribution : No Documentation Found

19. pVal : P-Value Calculate the P-value for each coefficient, which is the probability that the coefficient is insignificant (i.e

20. ConfintRec : No Documentation Found

21. ConfInt : Confidence Interval The Confidence Interval determines the upper and lower bounds of each estimated coefficient given a confidence level (level) that is required

22. FTestRec : No Documentation Found

23. FTest : F-Test Calculate the P-value for the full regression, which is the probability that all of the coefficients are insignificant (i.e

---

ATTRIBUTE **GetModel**

OLS \

`DATASET(Layout_Model)`	**GetModel**

GetModel Returns the learned model that maps X's to Y's. In the case of OLS, the model represents a set of Betas which are the coefficients of the linear model: Beta0 * 1 + Beta1 * Field1 + Beta2 * Field2 ... The ID of each model record specifies to which Y variable the coefficient applies. The Field Number ('number') indicates to which field of X the beta is to be applied. Field number 1 provides the intercept portion of the linear model and is always multiplied by 1. Note that if multiple work-items are provided within X and Y, there will be multiple models returned. The models can be separated by their work item id (i.e. 'wi'). A single model can be extracted from a myriad model by using e.g., model(wi=myWI_id). GetModel should not be called when predicting using a previously persisted model (i.e. when training data was not passed to the module.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 value })** — Model in DATASET(Layout_Model) format

SEE ML_core/Types.Layout_Model

OVERRIDE

FUNCTION Betas

DATASET(NumericField)	**Betas**
(DATASET(Layout_Model) model=GetModel)	

Return raw Beta values as numeric fields Extracts Beta values from the model. Can be used during training and prediction phases. For use during training phase, the 'model' parameter can be omitted. GetModel will be called to retrieve the model based on the training data. For use during prediction phase, a previously persisted model should be provided. The 'number' field of the returned NumericField records specifies to which Y the coefficient applies. The 'id' field of the returned record indicates the position of the Beta value. ID = 1 provides the Beta for the constant term (i.e. the Y intercept) while subsequent values reflect the Beta for each correspondingly numbered X feature. Feature 1 corresponds to Beta with 'id' = 2 and so on. If 'model' contains multiple work-items, Separate sets of Betas will be returned for each of the 'myriad' models (distinguished by 'wi').

PARAMETER <u>**model**</u> ||| TABLE (Layout_Model) — Optional parameter provides a model that was previously retrieved using GetModel. If omitted, GetModel will be used as the model.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 value })** — DATASET(NumericField) containing the Beta values.

FUNCTION Predict

DATASET(NumericField)	**Predict**
(DATASET(NumericField) newX, DATASET(Layout_Model) model=GetModel)	

Predict the dependent variable values (Y) for any set of independent variables (X). Returns a predicted Y values for each observation (i.e. record) of X. This supports the 'myriad' style interface in that multiple independent work items may be present in 'newX', and multiple independent models may be provided in 'model'. The resulting predicted values will also be separable by work item (i.e. wi).

PARAMETER <u>**newX**</u> ||| TABLE (NumericField) — The set of observations of independent variables in DATASET(NumericField) format.

PARAMETER <u>model</u> ||| TABLE (Layout_Model) — Optional. A model that was previously returned from GetModel (above). Note that a model from a previous run will only be valid if the field numbers in X are the same as when the model was learned. If this parameter is omitted, the current model will be used.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 value })** — An estimation of the corresponding Y value for each observation of newX. Returned in DATASET(NumericField) format with field number (i.e. 'number') indicating the dependent variable that is predicted.

OVERRIDE

TRANSFORM **makeRSQ**

OLS \

R2Rec	**makeRSQ**
(CoCoRec coco)	

No Documentation Found

PARAMETER <u>coco</u> ||| ROW (CoCoRec) — No Doc

RETURN **R2Rec** —

ATTRIBUTE **RSquared**

OLS \

DATASET(R2Rec)	**RSquared**

RSquared Calculate the R-Squared Metric used to assess the fit of the regression line to the training data. Since the regression has chosen the best (i.e. least squared error) line matching the data, this can be thought of as a measurement of the linearity of the training data. R Squared generally varies between

0 and 1, with 1 indicating an exact linear fit, and 0 indicating that a linear fit will have no predictive power. Negative values are possible under certain conditions, and indicate that the mean(Y) will be more predictive than any linear fit. Moderate values of R squared (e.g. .5) may indicate that the relationship of X -> Y is non-linear, or that the measurement error is high relative to the linear correlation (e.g. many outliers). In the former case, increasing the dimensionality of X, such as by using polynomial variants of the features, may yield a better fit. R squared always increases when additional independent variables are added, so it should not be used to determine the optimal set of X variables to include. For that purpose, use Adjusted R Squared (below) which penalizes larger numbers of variables. Note that the result of this call is only meaningful during training phase (use case 1 above) as it is an analysis based on the training data which is not provided during a prediction-only phase.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 number , REAL8 RSquared }) —** DATASET(R2Rec) with one record per dependent variable, per work-item. The number field indicates the dependent variable and coresponds to the number field of the dependent (Y) variable to which it applies.

RECORD **AnovaRec**

OLS \

AnovaRec

No Documentation Found

FIELD **model_f** ||| REAL8 — No Doc

FIELD **error_ss** ||| REAL8 — No Doc

FIELD **total_ss** ||| REAL8 — No Doc

FIELD **wi** ||| UNSIGNED2 — No Doc

FIELD **error_ms** ||| REAL8 — No Doc

FIELD **error_df** ||| UNSIGNED8 — No Doc

FIELD **model_ms** ||| REAL8 — No Doc

FIELD **total_df** ||| UNSIGNED8 — No Doc

FIELD **model_ss** ||| REAL8 — No Doc

FIELD **model_df** ||| UNSIGNED8 — No Doc

FIELD **total_ms** ||| REAL8 — No Doc

FIELD <u>number</u> ||| UNSIGNED4 — No Doc

TRANSFORM calcAnova

OLS \

AnovaRec	**calcAnova**
(tmpRec le)	

No Documentation Found

PARAMETER <u>le</u> ||| ROW (tmpRec) — No Doc

RETURN **AnovaRec** —

ATTRIBUTE Anova

OLS \

	Anova

ANOVA (Analysis of Variance) report Analyzes the sources of variance. Basic ANOVA equality: Model + Error = Total Determines how much of the variance of Y is explained by the regression model, versus how much is due to the error term (i.e. unexplained variance). This attribute is only meaningful during the training phase. Provides one record per work-item. Each record provides the following statistics: - Total_SS – Total Sum of Squares (SS) variance of the dependent data - Model_SS – The SS variance represented within the model - Error_SS – The SS variance not reflected by the model (i.e. Total_SS - Error_SS) - Total_DF – The total degrees of freedom within the dependent data - Model_DF – Degrees of freedom of the model - Error_DF – Degrees of freedom of the error component - Total_MS – The Mean Square (MS) variance of the dependent data - Model_MS – The Mean Square (MS) variance represented within the model - Error_MS – The MS variance not reflected by the model - Model_F – The F-Test statistic: Model_MS / Error_MS

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 number , REAL8 Total_SS , REAL8 Model_SS , REAL8 Error_SS , UNSIGNED8 Total_DF , UNSIGNED8**

Model_DF , UNSIGNED8 Error_DF , REAL8 Total_MS , REAL8 Model_MS , REAL8 Error_MS , REAL8 Model_F }) — DATASET(AnovaRec), one per work-item per dependent (Y) variable The number field indicates the dependent variable to which the analysis applies.

`ATTRIBUTE` SE

OLS \

`DATASET(NumericField)`	**SE**

Standard Error of the Regression Coefficients Describes the variability of the regression error for each coefficient. Only meaningful during the training phase.

`RETURN` **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 value })** — DATASET(NumericField), one record per Beta coefficient per dependent variable per work-item. The 'id' field is the coefficient number, with 1 being the Y intercept, 2 being the coefficient for the first feature, etc. The 'number' field indicates the dependent variable to which the coefficient applies.

`ATTRIBUTE` TStat

OLS \

`DATASET(NumericField)`	**TStat**

T-Statistic The T-statistic identifies the significance of the value of each regression coefficient. Its calculation is simply the value of the coefficient divided by the Standard Error of the coefficient. A larger absolute value of the T-statistic indicates that the coefficient is more significant. Only meaningful during the training phase.

`RETURN` **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 value })** — DATSET(NumericField), one record per Beta coefficient per dependent variable per work-item. The 'id' field is the coefficient number, with 1 being the Y intercept, 2 being the coefficient for the first feature, etc. The number field indicates the dependent variable to which the coefficient applies.

ATTRIBUTE AdjRSquared

OLS \

DATASET(R2Rec)	AdjRSquared

Adjusted R2 Calculate Adjusted R Squared which is a scaled version of R Squared that does not arbitrarily increase with the number of features. Adjusted R2, rather than R2 should always be used when trying to determine the best set of features to include in a model. When adding features, R2 will always increase, whether or not it improves the predictive power of the model. Adjusted R2, however, will only increase with the predictive power of the model.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 number , REAL8 RSquared }) —** DATASET(R2Rec), one record per dependent variable per work-item. The number field indicates the dependent variable and corresponds to the number field of the dependent (Y) variable to which it applies.

RECORD AICRec

OLS \

	AICRec

No Documentation Found

FIELD <u>aic</u> ||| REAL8 — No Doc

FIELD <u>wi</u> ||| UNSIGNED2 — No Doc

FIELD <u>number</u> ||| UNSIGNED4 — No Doc

ATTRIBUTE AIC

OLS \

DATASET(AICRec)	AIC

Akaike Information Criterion (AIC) Information theory based criterion for assessing Goodness of Fit (GOF). Lower values mean better fit.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 number , REAL8 AIC })** —
DATASET(AICRec), one record per dependent variable per work-item. The number field indicates the dependent variable and corresponds to the number field of the dependent (Y) variable to which it applies.

RECORD RangeVec

OLS \

	RangeVec

No Documentation Found

FIELD **p** ||| REAL8 — No Doc

FIELD **rangehigh** ||| REAL8 — No Doc

FIELD **rangelow** ||| REAL8 — No Doc

FIELD **rangenumber** ||| UNSIGNED8 — No Doc

MODULE DistributionBase

OLS \

	DistributionBase
`(t_Count Nranges = 10000)`	

No Documentation Found

PARAMETER **nranges** ||| UNSIGNED8 — No Doc

Children

1. Low : No Documentation Found

2. High : No Documentation Found

3. Density : No Documentation Found

4. RangeWidth : No Documentation Found

5. DensityV : No Documentation Found

6. CumulativeV : No Documentation Found

7. Cumulative : No Documentation Found

8. NTile : No Documentation Found

9. InvDensity : No Documentation Found

10. Discrete : No Documentation Found

`ATTRIBUTE` Low

OLS \ DistributionBase \

Low

No Documentation Found

`RETURN` **INTEGER8** —

`ATTRIBUTE` High

OLS \ DistributionBase \

High

No Documentation Found

`RETURN` **INTEGER8** —

FUNCTION Density

`t_FieldReal`	**Density**
`(t_FieldReal t)`	

No Documentation Found

PARAMETER **t** ||| REAL8 — No Doc

RETURN **REAL8** —

ATTRIBUTE RangeWidth

	RangeWidth

No Documentation Found

RETURN **REAL8** —

FUNCTION DensityV

`DATASET(RangeVec)`	**DensityV**
`()`	

No Documentation Found

RETURN **TABLE ({ UNSIGNED8 RangeNumber , REAL8 RangeLow , REAL8 RangeHigh , REAL8 P })** —

FUNCTION CumulativeV

OLS \ DistributionBase \

	CumulativeV
()	

No Documentation Found

RETURN **TABLE ({ UNSIGNED8 RangeNumber , REAL8 RangeLow , REAL8 RangeHigh , REAL8 P }) —**

FUNCTION Cumulative

OLS \ DistributionBase \

t_FieldReal	Cumulative
(t_FieldReal t)	

No Documentation Found

PARAMETER **t** ||| REAL8 — No Doc

RETURN **REAL8 —**

FUNCTION NTile

OLS \ DistributionBase \

t_FieldReal	NTile
(t_FieldReal Pc)	

No Documentation Found

FUNCTION InvDensity

OLS \ DistributionBase \

InvDensity
(t_FieldReal delta)

No Documentation Found

ATTRIBUTE Discrete

OLS \ DistributionBase \

Discrete

No Documentation Found

MODULE TDistribution

OLS \

	TDistribution
`(t_Discrete v_in,t_Count NRanges = 10000)`	

No Documentation Found

PARAMETER **v__in** ||| INTEGER4 — No Doc

PARAMETER **nranges** ||| UNSIGNED8 — No Doc

PARENT **linearregression.ols.distributionbase** <OLS.ecl.tex>

Children

1. DensityV : No Documentation Found

2. NTile : No Documentation Found

3. Discrete : No Documentation Found

4. InvDensity : No Documentation Found

5. High : No Documentation Found

6. Low : No Documentation Found

7. RangeWidth : No Documentation Found

8. Density : No Documentation Found

9. CumulativeV : No Documentation Found

10. Cumulative : No Documentation Found

FUNCTION DensityV

OLS \ TDistribution \

`DATASET(RangeVec)`	**DensityV**
`()`	

No Documentation Found

RETURN **TABLE ({ UNSIGNED8 RangeNumber , REAL8 RangeLow , REAL8 RangeHigh , REAL8 P })** —

OVERRIDE

FUNCTION NTile

OLS \ TDistribution \

t_FieldReal	**NTile**
(t_FieldReal Pc)	

No Documentation Found

PARAMETER **pc** ||| REAL8 — No Doc

RETURN **REAL8** —

OVERRIDE

ATTRIBUTE Discrete

OLS \ TDistribution \

	Discrete

No Documentation Found

RETURN **BOOLEAN** —

INHERITED

FUNCTION InvDensity

OLS \ TDistribution \

InvDensity
(t_FieldReal delta)

No Documentation Found

PARAMETER <u>delta</u> ||| REAL8 — No Doc

RETURN REAL8 —

OVERRIDE

ATTRIBUTE High

OLS \ TDistribution \

High

No Documentation Found

RETURN INTEGER8 —

OVERRIDE

ATTRIBUTE Low

OLS \ TDistribution \

Low

No Documentation Found

ATTRIBUTE RangeWidth

OLS \ TDistribution \

RangeWidth

No Documentation Found

RETURN REAL8 —

OVERRIDE

FUNCTION Density

OLS \ TDistribution \

t_FieldReal	Density
(t_FieldReal t)	

No Documentation Found

PARAMETER <u>t</u> ||| REAL8 — No Doc

RETURN REAL8 —

OVERRIDE

FUNCTION CumulativeV

OLS \ TDistribution \

	CumulativeV
()	

No Documentation Found

RETURN TABLE ({ UNSIGNED8 RangeNumber , REAL8 RangeLow , REAL8 RangeHigh , REAL8 P }) —

OVERRIDE

FUNCTION Cumulative

OLS \ TDistribution \

t_FieldReal	Cumulative
(t_FieldReal t)	

No Documentation Found

PARAMETER **t** ||| REAL8 — No Doc

RETURN REAL8 —

OVERRIDE

MODULE FDistribution

OLS \

FDistribution
`(t_Discrete d1_in, t_Discrete d2_in, t_Count NRanges = 10000)`

No Documentation Found

PARAMETER **d1__in** ||| INTEGER4 — No Doc

PARAMETER **nranges** ||| UNSIGNED8 — No Doc

PARAMETER **d2__in** ||| INTEGER4 — No Doc

PARENT **linearregression.ols.distributionbase** <OLS.ecl.tex>

Children

1. DensityV : No Documentation Found

2. CumulativeV : No Documentation Found

3. Cumulative : No Documentation Found

4. NTile : No Documentation Found

5. InvDensity : No Documentation Found

6. Discrete : No Documentation Found

7. Low : No Documentation Found

8. High : No Documentation Found

9. RangeWidth : No Documentation Found

10. Density : No Documentation Found

FUNCTION **DensityV**

OLS \ FDistribution \

`DATASET(RangeVec)`	**DensityV**
`()`	

No Documentation Found

RETURN TABLE ({ UNSIGNED8 RangeNumber , REAL8 RangeLow , REAL8 RangeHigh , REAL8 P }) —

OVERRIDE

FUNCTION CumulativeV

OLS \ FDistribution \

	CumulativeV
()	

No Documentation Found

RETURN TABLE ({ UNSIGNED8 RangeNumber , REAL8 RangeLow , REAL8 RangeHigh , REAL8 P }) —

OVERRIDE

FUNCTION Cumulative

OLS \ FDistribution \

t_FieldReal	Cumulative
(t_FieldReal t)	

No Documentation Found

PARAMETER t ||| REAL8 — No Doc

RETURN REAL8 —

OVERRIDE

FUNCTION NTile

t_FieldReal	NTile
(t_FieldReal Pc)	

No Documentation Found

PARAMETER **pc** ||| REAL8 — No Doc

RETURN REAL8 —

OVERRIDE

FUNCTION InvDensity

	InvDensity
(t_FieldReal delta)	

No Documentation Found

PARAMETER **delta** ||| REAL8 — No Doc

RETURN REAL8 —

INHERITED

ATTRIBUTE Discrete

	Discrete

No Documentation Found

RETURN BOOLEAN —

INHERITED

ATTRIBUTE Low

	Low

No Documentation Found

RETURN INTEGER8 —

INHERITED

ATTRIBUTE High

	High

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE RangeWidth

OLS \ FDistribution \

	RangeWidth

No Documentation Found

RETURN **REAL8** —

FUNCTION Density

OLS \ FDistribution \

t_FieldReal	Density
(t_FieldReal t)	

No Documentation Found

PARAMETER **t** ||| REAL8 — No Doc

RETURN **REAL8** —

MODULE NormalDistribution

NormalDistribution
`(t_Count NRanges)`

No Documentation Found

PARAMETER **nranges** ||| UNSIGNED8 — No Doc

PARENT **linearregression.ols.distributionbase** <OLS.ecl.tex>

Children

1. Low : No Documentation Found

2. High : No Documentation Found

3. RangeWidth : No Documentation Found

4. DensityV : No Documentation Found

5. CumulativeV : No Documentation Found

6. Cumulative : No Documentation Found

7. NTile : No Documentation Found

8. InvDensity : No Documentation Found

9. Discrete : No Documentation Found

10. Density : No Documentation Found

ATTRIBUTE Low

Low

No Documentation Found

ATTRIBUTE High

OLS \ NormalDistribution \

	High

No Documentation Found

RETURN **INTEGER8** —

ATTRIBUTE RangeWidth

OLS \ NormalDistribution \

	RangeWidth

No Documentation Found

RETURN **REAL8** —

OVERRIDE

FUNCTION DensityV

OLS \ NormalDistribution \

DATASET(RangeVec)	DensityV
()	

No Documentation Found

RETURN **TABLE ({ UNSIGNED8 RangeNumber , REAL8 RangeLow , REAL8 RangeHigh , REAL8 P }) —**

OVERRIDE

FUNCTION CumulativeV

OLS \ NormalDistribution \

	CumulativeV
()	

No Documentation Found

RETURN **TABLE ({ UNSIGNED8 RangeNumber , REAL8 RangeLow , REAL8 RangeHigh , REAL8 P }) —**

OVERRIDE

FUNCTION Cumulative

OLS \ NormalDistribution \

t_FieldReal	Cumulative
(t_FieldReal t)	

No Documentation Found

RETURN **REAL8** —

OVERRIDE

FUNCTION NTile

OLS \ NormalDistribution \

t_FieldReal	**NTile**
(t_FieldReal Pc)	

No Documentation Found

RETURN **REAL8** —

OVERRIDE

FUNCTION InvDensity

OLS \ NormalDistribution \

	InvDensity
(t_FieldReal delta)	

No Documentation Found

PARAMETER <u>**delta**</u> ||| REAL8 — No Doc

RETURN **REAL8** —

INHERITED

ATTRIBUTE Discrete

OLS \ NormalDistribution \

	Discrete

No Documentation Found

RETURN **BOOLEAN** —

INHERITED

FUNCTION Density

OLS \ NormalDistribution \

t_FieldReal	Density
(t_FieldReal t)	

No Documentation Found

PARAMETER <u>**t**</u> ||| REAL8 — No Doc

RETURN **REAL8** —

OVERRIDE

ATTRIBUTE pVal

pVal

P-Value Calculate the P-value for each coefficient, which is the probability that the coefficient is insignificant (i.e. actually zero). A low P-value (e.g. .05) provides evidence that the coefficient is significant in the model. A high P-value indicates that the coefficient value should, in fact, be zero. P-value is related to the T-Statistic, and can be thought of as a normalized version of the T-Statistic. Only meaningful during the training phase.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 value })** — DATSET(NumericField), one record per Beta coefficient per dependent variable per work-item. The 'id' field is the coefficient number, with 1 being the Y intercept, 2 being the coefficient for the first feature, etc. The number field indicates the dependent variable and corresponds to the number field of the dependent (Y) variable to which it applies.

RECORD ConfintRec

ConfintRec

No Documentation Found

FIELD <u>lowerint</u> ||| REAL8 — No Doc

FIELD <u>upperint</u> ||| REAL8 — No Doc

FIELD <u>id</u> ||| UNSIGNED8 — No Doc

FIELD <u>wi</u> ||| UNSIGNED2 — No Doc

FIELD <u>number</u> ||| UNSIGNED4 — No Doc

FUNCTION ConfInt

OLS \

	ConfInt
(Types.t_fieldReal level)	

Confidence Interval The Confidence Interval determines the upper and lower bounds of each estimated coefficient given a confidence level (level) that is required. For example, one could say that there is a 95% probability (level) that the coefficient of the first independent variable is between 2.05 and 3.62. This allows error margins to be determined with the desired confidence level. If the confidence interval spans zero, it implies that the coefficient may not be significant at the specified confidence level.

PARAMETER level ||| REAL8 — The level of confidence required, expressed as a percentage from 0.0 to 100.0

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 LowerInt , REAL8 UpperInt })** — DATASET(ConfintRec) with one record per coefficient per dependent variable per work-item. The 'id' field is the coefficient number, with 1 being the Y intercept, 2 being the coefficient for the first feature, etc. The number field indicates the dependent variable and corresponds to the number field of the dependent (Y) variable to which it applies.

RECORD FTestRec

OLS \

	FTestRec

No Documentation Found

FIELD model_f ||| REAL8 — No Doc

FIELD pvalue ||| REAL8 — No Doc

FIELD wi ||| UNSIGNED2 — No Doc

FIELD number ||| UNSIGNED4 — No Doc

ATTRIBUTE **FTest**

OLS \

| DATASET(FTestRec) | **FTest** |

F-Test Calculate the P-value for the full regression, which is the probability that all of the coefficients are insignificant (i.e. actually zero). A low P-value (e.g. .05) provides evidence that at least one coefficient is significant. A high P-value indicates that all the coefficient values should in fact be zero, implying that the regression has no statistically significant predictive power. P-value is related to the ANOVA F-Statistic, and can be thought of as a standardized version of the ANOVA F-Statistic. The F-Test and T-Test are similar, except that the T-test is used to test the significance of each coefficient, while the F-Test is used to test the significance of the entire regression. For simple linear regression (i.e. only one independent variable, the T-Test and F-Test are equivalent.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 number , REAL8 Model_F , REAL8 pValue })** — DATASET(FTestRec), one record per dependent variable per work-item. The number field indicates the dependent variable and corresponds to the number field of the dependent (Y) variable to which it applies.

LogisticRegression

Name	LogisticRegression
Version	1.0.0
Description	Logistic Regression implementation
License	http://www.apache.org/licenses/LICENSE-2.0
Copyright	Copyright (C) 2017 HPCC Systems
Authors	HPCCSystems
DependsOn	ML_Core, PBblas
Platform	6.2.0

Table of Contents

Extract the beta values form the model dataset

ExtractBeta_CI.ecl
Extract the beta values form the model dataset
ExtractBeta_pval.ecl
Extract the beta values form the model dataset
ExtractReport.ecl
Extract Report records from model
LogitPredict.ecl
Predict the category values with the logit function and the the supplied beta coefficients
LogitScore.ecl
Calculate the score using the logit function and the the supplied beta coefficients
Model_Deviance.ecl
Model Deviance
Null_Deviance.ecl
Deviance for the null model, that is, a model with only an intercept
Types.ecl
IRLS
performance
Tests
validation

BinomialConfusion

[Go Up](#)

IMPORTS

ML_Core.Types | LogisticRegression | LogisticRegression.Types |

DESCRIPTIONS

FUNCTION BinomialConfusion

DATASET(Types.Binomial_Confusion_Summary)	**BinomialConfusion**
(DATASET(Core_Types.Confusion_Detail) d)	

Binomial confusion matrix. Work items with multinomial responses are ignored by this function. The higher value lexically is considered to be the positive indication.

PARAMETER **d** ||| TABLE (Confusion_Detail) — confusion detail for the work item and classifier

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 classifier , UNSIGNED8 true_positive , UNSIGNED8 true_negative , UNSIGNED8 false_positive , UNSIGNED8 false_negative , UNSIGNED8 cond_pos , UNSIGNED8 pred_pos , UNSIGNED8 cond_neg , UNSIGNED8 pred_neg , REAL8 prevalence , REAL8 accuracy , REAL8 true_pos_rate , REAL8 false_neg_rate , REAL8 false_pos_rate , REAL8 true_neg_rate , REAL8 pos_pred_val , REAL8 false_disc_rate , REAL8 false_omit_rate , REAL8 neg_pred_val })** — confusion matrix for a binomial classifier

BinomialLogisticRegression

IMPORTS

LogisticRegression | LogisticRegression.Constants | ML_Core.Interfaces | ML_Core.Types |

DESCRIPTIONS

MODULE **BinomialLogisticRegression**

BinomialLogisticRegression
`(UNSIGNED max_iter=200, REAL8 epsilon=Constants.default_epsilon, REAL8 ridge=Constants.default_ridge)`

Binomial logistic regression using iteratively re-weighted least squares.

PARAMETER **ridge** ||| REAL8 — a value to populate a diagonal matrix that is added to a matrix help assure that the matrix is invertible.

PARAMETER **max_iter** ||| UNSIGNED8 — maximum number of iterations to try

PARAMETER **epsilon** ||| REAL8 — the minimum change in the Beta value estimate to continue

PARENT **ML_Core.Interfaces.IClassify** <../ML_Core/Interfaces/IClassify.ecl.tex>

Children

1. GetModel : Calculate the model to fit the observation data to the observed classes

2. Classify : Classify the observations using a model

3. Report : Report the confusion matrix for the classifier and training data

---

FUNCTION **GetModel**

BinomialLogisticRegression \

`DATASET(Types.Layout_Model)`	**GetModel**
`(DATASET(Types.NumericField) observations,` `DATASET(Types.DiscreteField) classifications)`	

Calculate the model to fit the observation data to the observed classes.

PARAMETER **observations** ||| TABLE (NumericField) — the observed explanatory values

PARAMETER **classifications** ||| TABLE (DiscreteField) — the observed classification used to build the model

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 value })** — the encoded model

OVERRIDE

FUNCTION **Classify**

BinomialLogisticRegression \

`DATASET(Types.Classify_Result)`	**Classify**
`(DATASET(Types.Layout_Model) model,` `DATASET(Types.NumericField) new_observations)`	

Classify the observations using a model.

PARAMETER <u>**new_observations**</u> ||| TABLE (NumericField) — observations to be classified

PARAMETER <u>**model**</u> ||| TABLE (Layout_Model) — The model, which must be produced by a corresponding getModel function.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , INTEGER4 value , REAL8 conf })** — Classification with a confidence value

OVERRIDE

FUNCTION **Report**

BinomialLogisticRegression \

`DATASET(Types.Confusion_Detail)`	**Report**

`(DATASET(Types.Layout_Model) model,`
`DATASET(Types.NumericField) observations,`
`DATASET(Types.DiscreteField) classifications)`	

Report the confusion matrix for the classifier and training data.

PARAMETER <u>**classifications**</u> ||| TABLE (DiscreteField) — the classifications associated with the observations

PARAMETER <u>**observations**</u> ||| TABLE (NumericField) — the explanatory values.

PARAMETER <u>**model**</u> ||| TABLE (Layout_Model) — the encoded model

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 classifier , INTEGER4 actual_class , INTEGER4 predict_class , UNSIGNED4 occurs , BOOLEAN correct })** — the confusion matrix showing correct and incorrect results

OVERRIDE

Confusion

IMPORTS

ML_Core | ML_Core.Types | LogisticRegression | LogisticRegression.Types |

DESCRIPTIONS

FUNCTION Confusion

`DATASET(Confusion_Detail)`	**Confusion**
`(DATASET(DiscreteField) dependents, DATASET(DiscreteField) predicts)`	

Detail confusion records to compare actual versus predicted response variable values.

PARAMETER **predicts** ||| TABLE (DiscreteField) — the predicted responses

PARAMETER **dependents** ||| TABLE (DiscreteField) — the original response values

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 classifier , INTEGER4 actual_class , INTEGER4 predict_class , UNSIGNED4 occurs , BOOLEAN correct })** — confusion counts by predicted and actual response values.

Constants

DESCRIPTIONS

MODULE **Constants**

Constants

No Documentation Found

Children

1. limit_card : No Documentation Found

2. default_epsilon : No Documentation Found

3. default_ridge : No Documentation Found

4. local_cap : No Documentation Found

5. id_base : No Documentation Found

6. id_iters : No Documentation Found

7. id_delta : No Documentation Found

8. id_correct : No Documentation Found

9. id_incorrect : No Documentation Found

10. id_stat_set : No Documentation Found

11. id_betas : No Documentation Found

12. id_betas_coef : No Documentation Found

13. id_betas_SE : No Documentation Found

14. base_builder : No Documentation Found

15. [base_max_iter](#) : No Documentation Found

16. [base_epsilon](#) : No Documentation Found

17. [base_ind_vars](#) : No Documentation Found

18. [base_dep_vars](#) : No Documentation Found

19. [base_obs](#) : No Documentation Found

20. [builder_irls_local](#) : No Documentation Found

21. [builder_irls_global](#) : No Documentation Found

22. [builder_softmax](#) : No Documentation Found

ATTRIBUTE **limit_card**

[Constants](#) \

UNSIGNED2	**limit_card**

No Documentation Found

RETURN **UNSIGNED2** —

ATTRIBUTE **default_epsilon**

[Constants](#) \

REAL8	**default_epsilon**

No Documentation Found

RETURN **REAL8** —

ATTRIBUTE default_ridge

Constants \

REAL8	default_ridge

No Documentation Found

RETURN REAL8 —

ATTRIBUTE local_cap

Constants \

UNSIGNED4	local_cap

No Documentation Found

RETURN UNSIGNED4 —

ATTRIBUTE id_base

Constants \

	id_base

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE id_iters

	id__iters

No Documentation Found

RETURN **INTEGER8** —

ATTRIBUTE id_delta

	id__delta

No Documentation Found

RETURN **INTEGER8** —

ATTRIBUTE id_correct

	id__correct

No Documentation Found

RETURN **INTEGER8** —

ATTRIBUTE id_incorrect

Constants \

id__incorrect

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE id_stat_set

Constants \

id__stat__set

No Documentation Found

RETURN SET (INTEGER8) —

ATTRIBUTE id_betas

Constants \

id__betas

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE id_betas_coef

Constants \

id__betas__coef

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE id_betas_SE

Constants \

id__betas__SE

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE base_builder

Constants \

base__builder

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE base_max_iter

base__max__iter

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE base_epsilon

base__epsilon

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE base_ind_vars

base__ind__vars

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE base_dep_vars

Constants \

base__dep__vars

No Documentation Found

RETURN **INTEGER8** —

ATTRIBUTE base_obs

Constants \

base__obs

No Documentation Found

RETURN **INTEGER8** —

ATTRIBUTE builder_irls_local

Constants \

builder__irls__local

No Documentation Found

RETURN **INTEGER8** —

ATTRIBUTE builder_irls_global

builder_irls_global

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE builder_softmax

builder_softmax

No Documentation Found

RETURN INTEGER8 —

DataStats

IMPORTS

ML_Core.Types | LogisticRegression | LogisticRegression.Types | LogisticRegression.Constants |

DESCRIPTIONS

FUNCTION **DataStats**

DATASET(Types.Data_Info)	**DataStats**
(DATASET(Core_Types.NumericField) indep, DATASET(Core_Types.DiscreteField) dep, BOOLEAN field_details=FALSE)	

Information about the datasets. Without details the range for the x and y (independent and dependent) columns. Note that a column of all zero values cannot be distinguished from a missing column. When details are requested, the cardinality, minimum, and maximum values are returned. A zero cardinality is returned when the field cardinality exceeds the Constants.limit_card value.

PARAMETER **indep** ||| TABLE (NumericField) — data set of independent variables

PARAMETER **field_details** ||| BOOLEAN — Boolean directive to provide field level info

PARAMETER **dep** ||| TABLE (DiscreteField) — data set of dependent variables

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 dependent_fields , UNSIGNED4 dependent_records , UNSIGNED4 independent_fields , UNSIGNED4**

independent_records , UNSIGNED4 dependent_count , UNSIGNED4 independent_count , TABLE (Field_Desc) dependent_stats , TABLE (Field_Desc) independent_stats }) —

52

RETURNS a data set of information on each work item

Deviance_Analysis

IMPORTS

LogisticRegression | LogisticRegression.Types |

DESCRIPTIONS

FUNCTION Deviance_Analysis

`DATASET(Types.AOD_Record)`	**Deviance_Analysis**
`(DATASET(Types.Deviance_Record) proposed,` `DATASET(Types.Deviance_Record) base)`	

Compare deviance information for an analysis of deviance.

PARAMETER <u>base</u> ||| TABLE (Deviance_Record) — the base model for comparison

PARAMETER <u>proposed</u> ||| TABLE (Deviance_Record) — the proposed model

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 classifier , UNSIGNED8 residual_df , UNSIGNED8 df , REAL8 residual_dev , REAL8 deviance , REAL8 p_value })** — the comparison of the deviance between the models

Deviance_Detail

Go Up

IMPORTS

ML_Core | ML_Core.Types | LogisticRegression | LogisticRegression.Types |

DESCRIPTIONS

FUNCTION Deviance_Detail

`DATASET(Types.Observation_Deviance)`	**Deviance_Detail**
`(DATASET(Core_Types.DiscreteField) dependents,` `DATASET(Types.Raw_Prediction) predicts)`	

Detail deviance for each observation.

PARAMETER **predicts** ||| TABLE (Raw_Prediction) — the predicted values of the response variable

PARAMETER **dependents** ||| TABLE (DiscreteField) — original dependent records for the model

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 classifier , INTEGER4 actual , INTEGER4 predicted , REAL8 mod_ll , REAL8 mod_dev_component , REAL8 mod_dev_residual , REAL8 nil_ll , REAL8 nil_dev_component , REAL8 nil_dev_residual })** — the deviance information by observation and the log likelihood of the predicted result.

dimm

IMPORTS

std.blas | std.BLAS.Types |

DESCRIPTIONS

EMBED **dimm**

Types.matrix_t	dimm
(BOOLEAN transposeA, BOOLEAN transposeB, BOOLEAN diagonalA, BOOLEAN diagonalB, Types.dimension_t m, Types.dimension_t n, Types.dimension_t k, Types.value_t alpha, Types.matrix_t A, Types.matrix_t B, Types.value_t beta=0.0, Types.matrix_t C=[])	

Matrix multiply when either A or B is a diagonal and is passed as a vector. alpha*op(A) op(B) + beta*C where op() is transpose

PARAMETER **diagonalB** ||| BOOLEAN — true when B is the diagonal matrix

PARAMETER **transposeA** ||| BOOLEAN — true when transpose of A is used

PARAMETER **alpha** ||| REAL8 — scalar used on A

PARAMETER **n** ||| UNSIGNED4 — number of columns in product

PARAMETER **beta** ||| REAL8 — scalar for matrix C

PARAMETER **m** ||| UNSIGNED4 — number of rows in product

PARAMETER <u>**k**</u> ||| UNSIGNED4 — number of columns/rows for the multiplier/multiplicand

PARAMETER <u>**C**</u> ||| SET (REAL8) — matrix C or empty

PARAMETER <u>**A**</u> ||| SET (REAL8) — matrix A

PARAMETER <u>**transposeB**</u> ||| BOOLEAN — true when transpose of B is used

PARAMETER <u>**diagonalA**</u> ||| BOOLEAN — true when A is the diagonal matrix

PARAMETER <u>**B**</u> ||| SET (REAL8) — matrix B

RETURN **SET (REAL8)** —

Distributions

IMPORTS

ML_Core.Constants | ML_Core.Math |

DESCRIPTIONS

MODULE **Distributions**

	Distributions

No Documentation Found

Children

1. Normal_CDF : Cumulative Distribution of the standard normal distribution, the probability that a normal random variable will be smaller than x standard deviations above or below the mean

2. Normal_PPF : Normal Distribution Percentage Point Function

3. T_CDF : Students t distribution integral evaluated between negative infinity and x

4. T_PPF : Percentage point function for the T distribution

5. Chi2_CDF : The cumulative distribution function for the Chi Square distribution

6. Chi2_PPF : The Chi Squared PPF function

FUNCTION Normal_CDF

REAL8	Normal__CDF
(REAL8 x)	

Cumulative Distribution of the standard normal distribution, the probability that a normal random variable will be smaller than x standard deviations above or below the mean. Taken from C/C++ Mathematical Algorithms for Scientists and Engineers, n. Shammas, McGraw-Hill, 1995

PARAMETER **x** ||| REAL8 — the number of standard deviations

RETURN **REAL8** —

RETURNS probability of exceeding x.

FUNCTION Normal_PPF

REAL8	Normal__PPF
(REAL8 x)	

Normal Distribution Percentage Point Function. Translated from C/C++ Mathematical Algorithms for Scientists and Engineers, N. Shammas, McGraw-Hill, 1995

PARAMETER **x** ||| REAL8 — probability

RETURN **REAL8** —

RETURNS number of standard deviations from the mean

FUNCTION **T_CDF**

REAL8	**T_CDF**
(REAL8 x, REAL8 df)	

Students t distribution integral evaluated between negative infinity and x. Translated from NIST SEL DATAPAC Fortran TCDF.f source

PARAMETER **df** ||| REAL8 — degrees of freedom

PARAMETER **x** ||| REAL8 — value of the evaluation

RETURN **REAL8** —

RETURNS the probability that a value will be less than the specified value

FUNCTION **T_PPF**

REAL8	**T_PPF**
(REAL8 x, REAL8 df)	

Percentage point function for the T distribution. Translated from NIST SEL DATAPAC Fortran TPPF.f source

PARAMETER **df** ||| REAL8 — No Doc

PARAMETER **x** ||| REAL8 — No Doc

RETURN **REAL8** —

FUNCTION Chi2_CDF

REAL8	Chi2__CDF
(REAL8 x, REAL8 df)	

The cumulative distribution function for the Chi Square distribution. the CDF for the specfied degrees of freedom. Translated from the NIST SEL DATAPAC Fortran subroutine CHSCDF.

PARAMETER **df** ||| REAL8 — No Doc

PARAMETER **x** ||| REAL8 — No Doc

RETURN **REAL8** —

FUNCTION Chi2_PPF

REAL8	Chi2__PPF
(REAL8 x, REAL8 df)	

The Chi Squared PPF function. Translated from the NIST SEL DATAPAC Fortran subroutine CHSPPF.

PARAMETER **df** ||| REAL8 — No Doc

PARAMETER **x** ||| REAL8 — No Doc

RETURN **REAL8** —

ExtractBeta

IMPORTS

ML_Core.Types | LogisticRegression | LogisticRegression.Types |

DESCRIPTIONS

FUNCTION ExtractBeta

ExtractBeta
`(DATASET(Core_Types.Layout_Model) mod_ds)`

Extract the beta values form the model dataset.

PARAMETER **mod_ds** ||| TABLE (Layout_Model) — the model dataset

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 ind_col , UNSIGNED4 dep_nom , REAL8 w , REAL8 SE })** — a beta values as Model Coefficient records, zero as the constant term.

LogisticRegression/
ExtractBeta_CI

[Go Up](#)

IMPORTS

ML_Core.Types | LogisticRegression | LogisticRegression.Types |

DESCRIPTIONS

FUNCTION ExtractBeta_CI

`DATASET(Types.Confidence_Model_Coef)`	**ExtractBeta_CI**
`(DATASET(Core_Types.Layout_Model) mod_ds, REAL8 level)`	

Extract the beta values form the model dataset.

PARAMETER **mod_ds** ||| TABLE (Layout_Model) — the model dataset

PARAMETER **level** ||| REAL8 — the significance value for the intervals

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 ind_col , UNSIGNED4 dep_nom , REAL8 w , REAL8 SE , REAL8 upper , REAL8 lower })** — the beta values with confidence intervals term.

ExtractBeta__pval

IMPORTS

ML__Core.Types | LogisticRegression | LogisticRegression.Types |

DESCRIPTIONS

FUNCTION **ExtractBeta_pval**

`DATASET(Types.pval_Model_Coef)`	**ExtractBeta__pval**
`(DATASET(Core_Types.Layout_Model) mod_ds)`	

Extract the beta values form the model dataset.

PARAMETER **mod__ds** ||| TABLE (Layout_Model) — the model dataset

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 ind__col , UNSIGNED4 dep__nom , REAL8 w , REAL8 SE , REAL8 z , REAL8 p__value })** — the beta values with p-values as Model Coefficient records, zero as the constant term.

ExtractReport

IMPORTS

ML_Core.Types | LogisticRegression | LogisticRegression.Types | LogisticRegression.Constants |

DESCRIPTIONS

FUNCTION **ExtractReport**

`DATASET(Types.Model_Report)`	**ExtractReport**
`(DATASET(Core_Types.Layout_Model) mod_ds)`	

Extract Report records from model

PARAMETER **mod_ds** ||| TABLE (Layout_Model) — the model dataset

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 max_iterations , REAL8 epsilon , UNSIGNED4 dep_vars , UNSIGNED4 ind_vars , UNSIGNED8 obs , UNSIGNED2 builder , TABLE (Classifier_Stats) stats })** — the model report dataset

LogitPredict

IMPORTS

ML_Core.Types | LogisticRegression | LogisticRegression.Types |

DESCRIPTIONS

FUNCTION **LogitPredict**

DATASET(Classify_Result)	**LogitPredict**
(DATASET(Model_Coef) coef, DATASET(NumericField) independents)	

Predict the category values with the logit function and the the supplied beta coefficients.

PARAMETER **independents** ||| TABLE (NumericField) — the observations

PARAMETER **coef** ||| TABLE (Model_Coef) — the model beta coefficients

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , INTEGER4 value , REAL8 conf })** — the predicted category values and a confidence score

LogitScore

IMPORTS

ML_Core.Types | LogisticRegression | LogisticRegression.Types |

DESCRIPTIONS

FUNCTION **LogitScore**

`DATASET(Raw_Prediction)`	**LogitScore**
`(DATASET(Model_Coef) coef, DATASET(NumericField) independents)`	

Calculate the score using the logit function and the the supplied beta coefficients.

PARAMETER **independents** ||| TABLE (NumericField) — the observations

PARAMETER **coef** ||| TABLE (Model_Coef) — the model beta coefficients

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 raw })** — the raw prediction value

Model__Deviance

IMPORTS

LogisticRegression | LogisticRegression.Types |

DESCRIPTIONS

FUNCTION Model_Deviance

`DATASET(Types.Deviance_Record)`	**Model__Deviance**
`(DATASET(Types.Observation_Deviance) od,` `DATASET(Types.Model_Coef) mod)`	

Model Deviance.

PARAMETER **mod** ||| TABLE (Model_Coef) — model co-efficients

PARAMETER **od** ||| TABLE (Observation_Deviance) — observation deviance record

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 classifier , UNSIGNED8 df ,
REAL8 deviance , REAL8 AIC })** — model deviance

Null_Deviance

IMPORTS

LogisticRegression | LogisticRegression.Types |

DESCRIPTIONS

FUNCTION Null_Deviance

DATASET(Types.Deviance_Record)	**Null_Deviance**
(DATASET(Types.Observation_Deviance) od)	

Deviance for the null model, that is, a model with only an intercept.

PARAMETER **od** ||| TABLE (Observation_Deviance) — Observation Deviance record set.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 classifier , UNSIGNED8 df , REAL8 deviance , REAL8 AIC })** — a data set of the null model deviances for each work item and classifier.

Types

IMPORTS

ML_Core.Types |

DESCRIPTIONS

MODULE Types

	Types

No Documentation Found

Children

1. t_Universe : No Documentation Found

2. Field_Desc : No Documentation Found

3. Data_Info : No Documentation Found

4. NumericField_U : No Documentation Found

5. DiscreteField_U : No Documentation Found

6. Layout_Column_Map : No Documentation Found

7. Classifier_Stats : No Documentation Found

8. Model_Report : No Documentation Found

9. Binomial_Confusion_Summary : No Documentation Found

10. Model_Coef : No Documentation Found

11. Confidence_Model_Coef : No Documentation Found

12. pval_Model_Coef : No Documentation Found

13. Raw_Prediction : No Documentation Found

14. Observation_Deviance : No Documentation Found

15. Deviance_Record : No Documentation Found

16. AOD_Record : No Documentation Found

ATTRIBUTE t_Universe

Types \

t__Universe

No Documentation Found

RETURN UNSIGNED1 —

RECORD Field_Desc

Types \

Field__Desc

No Documentation Found

FIELD <u>cardinality</u> ||| UNSIGNED4 — No Doc

FIELD <u>min__value</u> ||| REAL8 — No Doc

FIELD <u>number</u> ||| UNSIGNED4 — No Doc

FIELD **max__value** ||| REAL8 — No Doc

RECORD Data_Info

Types \

	Data__Info

No Documentation Found

FIELD **dependent__count** ||| UNSIGNED4 — No Doc

FIELD **dependent__stats** ||| TABLE (Field_Desc) — No Doc

FIELD **wi** ||| UNSIGNED2 — No Doc

FIELD **independent__fields** ||| UNSIGNED4 — No Doc

FIELD **independent__count** ||| UNSIGNED4 — No Doc

FIELD **dependent__records** ||| UNSIGNED4 — No Doc

FIELD **independent__stats** ||| TABLE (Field_Desc) — No Doc

FIELD **independent__records** ||| UNSIGNED4 — No Doc

FIELD **dependent__fields** ||| UNSIGNED4 — No Doc

RECORD NumericField_U

Types \

	NumericField__U

No Documentation Found

FIELD **id** ||| UNSIGNED8 — No Doc

FIELD <u>value</u> ||| REAL8 — No Doc

FIELD <u>wi</u> ||| UNSIGNED2 — No Doc

FIELD <u>u</u> ||| UNSIGNED1 — No Doc

FIELD <u>number</u> ||| UNSIGNED4 — No Doc

RECORD DiscreteField_U

Types \

	DiscreteField_U

No Documentation Found

FIELD <u>id</u> ||| UNSIGNED8 — No Doc

FIELD <u>value</u> ||| INTEGER4 — No Doc

FIELD <u>wi</u> ||| UNSIGNED2 — No Doc

FIELD <u>u</u> ||| UNSIGNED1 — No Doc

FIELD <u>number</u> ||| UNSIGNED4 — No Doc

RECORD Layout_Column_Map

Types \

	Layout__Column__Map

No Documentation Found

FIELD <u>orig__number</u> ||| UNSIGNED4 — No Doc

FIELD <u>remap__number</u> ||| UNSIGNED4 — No Doc

FIELD <u>wi</u> ||| UNSIGNED2 — No Doc

RECORD Classifier_Stats

Classifier__Stats

No Documentation Found

FIELD <u>incorrect</u> ||| UNSIGNED4 — No Doc

FIELD <u>iterations</u> ||| UNSIGNED4 — No Doc

FIELD <u>max__delta</u> ||| REAL8 — No Doc

FIELD <u>correct</u> ||| UNSIGNED4 — No Doc

FIELD <u>column</u> ||| UNSIGNED4 — No Doc

RECORD Model_Report

Model__Report

No Documentation Found

FIELD <u>obs</u> ||| UNSIGNED8 — No Doc

FIELD <u>max__iterations</u> ||| UNSIGNED4 — No Doc

FIELD <u>wi</u> ||| UNSIGNED2 — No Doc

FIELD <u>dep__vars</u> ||| UNSIGNED4 — No Doc

FIELD <u>stats</u> ||| TABLE (Classifier_Stats) — No Doc

FIELD <u>ind__vars</u> ||| UNSIGNED4 — No Doc

FIELD <u>builder</u> ||| UNSIGNED2 — No Doc

FIELD <u>epsilon</u> ||| REAL8 — No Doc

RECORD Binomial_Confusion_Summary

Binomial_Confusion_Summary

No Documentation Found

FIELD **true_neg_rate** ||| REAL8 — No Doc

FIELD **neg_pred_val** ||| REAL8 — No Doc

FIELD **false_negative** ||| UNSIGNED8 — No Doc

FIELD **wi** ||| UNSIGNED2 — No Doc

FIELD **accuracy** ||| REAL8 — No Doc

FIELD **false_positive** ||| UNSIGNED8 — No Doc

FIELD **cond_pos** ||| UNSIGNED8 — No Doc

FIELD **true_negative** ||| UNSIGNED8 — No Doc

FIELD **pred_neg** ||| UNSIGNED8 — No Doc

FIELD **false_disc_rate** ||| REAL8 — No Doc

FIELD **true_positive** ||| UNSIGNED8 — No Doc

FIELD **false_pos_rate** ||| REAL8 — No Doc

FIELD **false_omit_rate** ||| REAL8 — No Doc

FIELD **false_neg_rate** ||| REAL8 — No Doc

FIELD **cond_neg** ||| UNSIGNED8 — No Doc

FIELD **pos_pred_val** ||| REAL8 — No Doc

FIELD **true_pos_rate** ||| REAL8 — No Doc

FIELD **prevalence** ||| REAL8 — No Doc

FIELD **classifier** ||| UNSIGNED4 — No Doc

FIELD **pred_pos** ||| UNSIGNED8 — No Doc

RECORD Model_Coef

Model_Coef

No Documentation Found

FIELD <u>se</u> ||| REAL8 — No Doc

FIELD <u>w</u> ||| REAL8 — No Doc

FIELD <u>wi</u> ||| UNSIGNED2 — No Doc

FIELD <u>ind_col</u> ||| UNSIGNED4 — No Doc

FIELD <u>dep_nom</u> ||| UNSIGNED4 — No Doc

RECORD Confidence_Model_Coef

Confidence_Model_Coef

No Documentation Found

FIELD <u>dep_nom</u> ||| UNSIGNED4 — No Doc

FIELD <u>wi</u> ||| UNSIGNED2 — No Doc

FIELD <u>ind_col</u> ||| UNSIGNED4 — No Doc

FIELD <u>lower</u> ||| REAL8 — No Doc

FIELD <u>se</u> ||| REAL8 — No Doc

FIELD <u>upper</u> ||| REAL8 — No Doc

FIELD <u>w</u> ||| REAL8 — No Doc

RECORD pval_Model_Coef

	pval__Model__Coef

No Documentation Found

FIELD <u>z</u> ||| REAL8 — No Doc

FIELD <u>p__value</u> ||| REAL8 — No Doc

FIELD <u>dep__nom</u> ||| UNSIGNED4 — No Doc

FIELD <u>wi</u> ||| UNSIGNED2 — No Doc

FIELD <u>ind__col</u> ||| UNSIGNED4 — No Doc

FIELD <u>se</u> ||| REAL8 — No Doc

FIELD <u>w</u> ||| REAL8 — No Doc

RECORD Raw_Prediction

	Raw__Prediction

No Documentation Found

FIELD <u>id</u> ||| UNSIGNED8 — No Doc

FIELD <u>raw</u> ||| REAL8 — No Doc

FIELD <u>wi</u> ||| UNSIGNED2 — No Doc

FIELD <u>number</u> ||| UNSIGNED4 — No Doc

RECORD Observation_Deviance

Observation_Deviance

No Documentation Found

FIELD <u>actual</u> ||| INTEGER4 — No Doc

FIELD <u>nil_dev_component</u> ||| REAL8 — No Doc

FIELD <u>id</u> ||| UNSIGNED8 — No Doc

FIELD <u>mod_ll</u> ||| REAL8 — No Doc

FIELD <u>wi</u> ||| UNSIGNED2 — No Doc

FIELD <u>predicted</u> ||| INTEGER4 — No Doc

FIELD <u>nil_dev_residual</u> ||| REAL8 — No Doc

FIELD <u>nil_ll</u> ||| REAL8 — No Doc

FIELD <u>mod_dev_component</u> ||| REAL8 — No Doc

FIELD <u>classifier</u> ||| UNSIGNED4 — No Doc

FIELD <u>mod_dev_residual</u> ||| REAL8 — No Doc

RECORD Deviance_Record

Deviance_Record

No Documentation Found

FIELD <u>deviance</u> ||| REAL8 — No Doc

FIELD <u>df</u> ||| UNSIGNED8 — No Doc

FIELD <u>wi</u> ||| UNSIGNED2 — No Doc

FIELD <u>classifier</u> ||| UNSIGNED4 — No Doc

FIELD <u>aic</u> ||| REAL8 — No Doc

RECORD **AOD_Record**

Types \

	AOD_Record

No Documentation Found

FIELD <u>residual_df</u> ||| UNSIGNED8 — No Doc

FIELD <u>p_value</u> ||| REAL8 — No Doc

FIELD <u>residual_dev</u> ||| REAL8 — No Doc

FIELD <u>wi</u> ||| UNSIGNED2 — No Doc

FIELD <u>deviance</u> ||| REAL8 — No Doc

FIELD <u>df</u> ||| UNSIGNED8 — No Doc

FIELD <u>classifier</u> ||| UNSIGNED4 — No Doc

IRLS

Table of Contents

GetModel

IMPORTS

ML_Core | ML_Core.Types | LogisticRegression | LogisticRegression.Constants | LogisticRegression.Types | logisticregression.irls |

DESCRIPTIONS

FUNCTION **GetModel**

`DATASET(Layout_Model)`	**GetModel**
`(DATASET(NumericField) independents, DATASET(DiscreteField) dependents, UNSIGNED max_iter=200, REAL8 epsilon=Constants.default_epsilon, REAL8 ridge=Constants.default_ridge)`	

Generate logistic regression model from training data. The size of the inputs is used to determin which work items are processed with purely local operations (the data is moved once as necessary) or with global operations supporting a work item to use multiple nodes.

PARAMETER **independents** ||| TABLE (NumericField) — the independent values

PARAMETER **ridge** ||| REAL8 — a value to pupulate a diagonal matrix that is added to a matrix help assure that the matrix is invertible.

PARAMETER **max_iter** ||| UNSIGNED8 — maximum number of iterations to try

PARAMETER **dependents** ||| TABLE (DiscreteField) — the dependent values.

PARAMETER <u>**epsilon**</u> ||| REAL8 — the minimum change in the Beta value estimate to continue

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 value })** — coefficient matrix plus model building stats

---

GetModel__global

IMPORTS

ML_Core | ML_Core.Types | PBblas | PBblas.Types | LogisticRegression | LogisticRegression.Constants | LogisticRegression.Types |

DESCRIPTIONS

FUNCTION **GetModel_global**

DATASET(Layout_Model)	**GetModel__global**
(DATASET(NumericField) independents, DATASET(DiscreteField) dependents, UNSIGNED max_iter=200, REAL8 epsilon=Constants.default_epsilon, REAL8 ridge=Constants.default_ridge)	

Internal function to determine values for the model coefficients and selected statistics from building the model.

PARAMETER **independents** ||| TABLE (NumericField) — the independent values

PARAMETER **ridge** ||| REAL8 — a value to pupulate a diagonal matrix that is added to a matrix help assure that the matrix is invertible.

PARAMETER **max__iter** ||| UNSIGNED8 — maximum number of iterations to try

PARAMETER **dependents** ||| TABLE (DiscreteField) — the dependent values

PARAMETER **epsilon** ||| REAL8 — the minimum change in the Beta value estimate to continue

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 value })** — coefficient matrix plus model building statistics

GetModel_local

IMPORTS

ML_Core | ML_Core.Types | LogisticRegression | LogisticRegression.Constants |
LogisticRegression.Types | LogisticRegression.IRLS | std | std.blas |

DESCRIPTIONS

FUNCTION **GetModel_local**

`DATASET(Layout_Model)`	**GetModel_local**
`(DATASET(NumericField) independents, DATASET(DiscreteField) dependents, UNSIGNED2 max_iter=200, REAL8 epsilon=Constants.default_epsilon, REAL8 ridge=Constants.default_ridge)`	

Internal function to determine values for the model co-efficients and selected stats from building the model.

PARAMETER **independents** ||| TABLE (NumericField) — the independent values

PARAMETER **ridge** ||| REAL8 — a value to populate a diagonal matrix that is added to a matrix help assure that the matrix is invertible.

PARAMETER **max_iter** ||| UNSIGNED2 — maximum number of iterations to try

PARAMETER **dependents** ||| TABLE (DiscreteField) — the dependent values.

PARAMETER **epsilon** ||| REAL8 — the minimum change in the Beta value estimate to continue

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 value })** — coefficient matrix plus model building stats

performance

Table of Contents

RunBinomial

[Go Up]

IMPORTS

logisticregression | ML_Core.Types |

DESCRIPTIONS

ATTRIBUTE **RunBinomial**

	RunBinomial

No Documentation Found

RETURN —

Tests

Table of Contents

Check_Dist

[Go Up](#)

IMPORTS

LogisticRegression.Distributions | ML_Core | python |

DESCRIPTIONS

ATTRIBUTE Check_Dist

Check_Dist	

No Documentation Found

RETURN —

validation

Table of Contents

BinomialRegression

[Go Up](#)

IMPORTS

LogisticRegression.validation | logisticregression | ML_Core.Types | ML_Core |
LogisticRegression.Types |

DESCRIPTIONS

ATTRIBUTE **BinomialRegression**

BinomialRegression

No Documentation Found

RETURN —

discrete__GermanDS

[Go Up](#)

IMPORTS

ML__Core.Types |

DESCRIPTIONS

MODULE discrete_GermanDS

discrete__GermanDS

No Documentation Found

Children

1. content : No Documentation Found

ATTRIBUTE content

discrete__GermanDS \

content

No Documentation Found

RETURN **TABLE (discrete_GermanRECORD) —**

IrisDS

[Go Up](#)

IMPORTS

ML_Core | ML_Core.Types |

DESCRIPTIONS

ATTRIBUTE irisDS

irisDS

No Documentation Found

RETURN **TABLE (dsRecord)** —

unit__test__dimm

IMPORTS

LogisticRegression | std.BLAS.Types |

DESCRIPTIONS

ATTRIBUTE **unit__test__dimm**

unit__test__dimm

No Documentation Found

RETURN —

ML_Core

Name	ML_Core
Version	3.1.0
Description	Common definitions for Machine Learning
License	SeeLICENSE.TXT
Copyright	Copyright (C) 2017 HPCC Systems
Authors	HPCCSystems
Platform	6.2.0

Table of Contents

AppendID.ecl
AppendSeqID.ecl
Config.ecl
Constants.ecl Useful constants
FieldAggregates.ecl
FromField.ecl
Generate.ecl
ToField.ecl
Types.ecl
Interfaces
Math
Tests
Utils

AppendID

[Go Up](#)

DESCRIPTIONS

MACRO **AppendID**

AppendID
`(dIn,idfield,dOut)`

No Documentation Found

PARAMETER <u>**dout**</u> ||| INTEGER8 — No Doc

PARAMETER <u>**idfield**</u> ||| INTEGER8 — No Doc

PARAMETER <u>**din**</u> ||| INTEGER8 — No Doc

RETURN —

AppendSeqID

[Go Up](#)

DESCRIPTIONS

MACRO **AppendSeqID**

AppendSeqID
`(dIn,idfield,dOut)`

No Documentation Found

PARAMETER <u>dout</u> ||| INTEGER8 — No Doc

PARAMETER <u>idfield</u> ||| INTEGER8 — No Doc

PARAMETER <u>din</u> ||| INTEGER8 — No Doc

RETURN —

Config

DESCRIPTIONS

MODULE Config

Config

No Documentation Found

Children

1. MaxLookup : No Documentation Found

2. Discrete : No Documentation Found

3. RoundingError : No Documentation Found

ATTRIBUTE MaxLookup

Config \

MaxLookup

No Documentation Found

RETURN **INTEGER8** —

ATTRIBUTE Discrete

Config \

	Discrete

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE RoundingError

Config \

	RoundingError

No Documentation Found

RETURN REAL8 —

Constants

[Go Up](#)

DESCRIPTIONS

MODULE Constants

	Constants

Useful constants

Children

1. Pi : Constant PI

2. Root_2 : Constant square root of 2

ATTRIBUTE Pi

Constants \

	Pi

Constant PI

RETURN **REAL8** —

ATTRIBUTE Root_2

Root_2	

Constant square root of 2

RETURN **REAL8** —

FieldAggregates

[Go Up](#)

IMPORTS

ML_Core | ML_Core.Types | ML_Core.Utils | std.system.ThorLib |

DESCRIPTIONS

MODULE **FieldAggregates**

FieldAggregates
`(DATASET(Types.NumericField) d)`

No Documentation Found

PARAMETER **d** ||| TABLE (NumericField) — No Doc

Children

1. [Simple](#) : No Documentation Found

2. [SimpleRanked](#) : No Documentation Found

3. [Medians](#) : No Documentation Found

4. [MinMedNext](#) : No Documentation Found

5. [Buckets](#) : No Documentation Found

6. [BucketRanges](#) : No Documentation Found

7. Modes : No Documentation Found

8. Cardinality : No Documentation Found

9. RankedInput : No Documentation Found

10. NTiles : No Documentation Found

11. NTileRanges : No Documentation Found

ATTRIBUTE **Simple**

FieldAggregates \

	Simple

No Documentation Found

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 number , REAL8 minval , REAL8 maxval , REAL8 sumval , REAL8 countval , REAL8 mean , REAL8 var , REAL8 sd }) —**

ATTRIBUTE **SimpleRanked**

FieldAggregates \

	SimpleRanked

No Documentation Found

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 value , UNSIGNED8 Pos }) —**

ATTRIBUTE Medians

FieldAggregates \

	Medians

No Documentation Found

RETURN TABLE ({ UNSIGNED2 wi , UNSIGNED4 number , REAL8 median }) —

ATTRIBUTE MinMedNext

FieldAggregates \

	MinMedNext

No Documentation Found

RETURN TABLE ({ UNSIGNED2 wi , UNSIGNED4 number , REAL8 median , REAL8 nextval , REAL8 minval , REAL8 maxval , REAL8 sumval , REAL8 countval , REAL8 mean , REAL8 var , REAL8 sd }) —

FUNCTION Buckets

FieldAggregates \

	Buckets
(Types.t_Discrete n)	

No Documentation Found

PARAMETER n ||| INTEGER4 — No Doc

RETURN TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 value , UNSIGNED8 Pos , INTEGER4 bucket }) —

FUNCTION BucketRanges

FieldAggregates \

	BucketRanges
	(Types.t_Discrete n)

No Documentation Found

PARAMETER <u>n</u> ||| INTEGER4 — No Doc

RETURN TABLE ({ UNSIGNED2 wi , UNSIGNED4 number , INTEGER4 bucket , REAL8 Min , REAL8 Max , UNSIGNED8 cnt }) —

ATTRIBUTE Modes

FieldAggregates \

	Modes

No Documentation Found

RETURN TABLE ({ UNSIGNED2 wi , UNSIGNED4 number , REAL8 mode , UNSIGNED8 cnt }) —

ATTRIBUTE Cardinality

FieldAggregates \

Cardinality

No Documentation Found

RETURN TABLE ({ UNSIGNED2 wi , UNSIGNED4 number , UNSIGNED8 cardinality }) —

ATTRIBUTE **RankedInput**

FieldAggregates \

RankedInput

No Documentation Found

RETURN TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 value , REAL8 Pos }) —

FUNCTION **NTiles**

FieldAggregates \

NTiles
(Types.t_Discrete n)

No Documentation Found

PARAMETER $\underline{n}$ ||| INTEGER4 — No Doc

RETURN TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 value , REAL8 Pos , INTEGER4 ntile }) —

FUNCTION **NTileRanges**

	NTileRanges
`(Types.t_Discrete n)`	

No Documentation Found

PARAMETER <u>n</u> ||| INTEGER4 — No Doc

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 number , INTEGER4 ntile ,
REAL8 Min , REAL8 Max , UNSIGNED8 cnt }) —**

FromField

[Go Up](#)

DESCRIPTIONS

MACRO FromField

	FromField
	`(dIn,lOut,dOut,dMap='')`

No Documentation Found

PARAMETER **dmap** ||| INTEGER8 — No Doc

PARAMETER **lout** ||| INTEGER8 — No Doc

PARAMETER **dout** ||| INTEGER8 — No Doc

PARAMETER **din** ||| INTEGER8 — No Doc

RETURN —

Generate

IMPORTS

ML_Core | ML_Core.Types |

DESCRIPTIONS

MODULE Generate

	Generate

No Documentation Found

Children

1. tp_Method : No Documentation Found

2. MethodName : No Documentation Found

3. ToPoly : No Documentation Found

ATTRIBUTE tp_Method

Generate \

tp_Method

No Documentation Found

RETURN **UNSIGNED1** —

FUNCTION MethodName

Generate \

MethodName
(tp_Method x)

No Documentation Found

PARAMETER <u>x</u> ||| UNSIGNED1 — No Doc

RETURN **STRING7** —

FUNCTION ToPoly

Generate \

ToPoly
(DATASET(Types.NumericField) seedCol, UNSIGNED maxN=6)

No Documentation Found

PARAMETER <u>seedcol</u> ||| TABLE (NumericField) — No Doc
PARAMETER <u>maxn</u> ||| UNSIGNED8 — No Doc

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 value })** —

ToField

DESCRIPTIONS

MACRO **ToField**

	ToField
	`(dIn,dOut,idfield=", wifield=", wivalue=",datafields=")`

No Documentation Found

PARAMETER **datafields** ||| INTEGER8 — No Doc

PARAMETER **wivalue** ||| INTEGER8 — No Doc

PARAMETER **idfield** ||| INTEGER8 — No Doc

PARAMETER **dout** ||| INTEGER8 — No Doc

PARAMETER **wifield** ||| INTEGER8 — No Doc

PARAMETER **din** ||| INTEGER8 — No Doc

RETURN —

Types

Go Up

DESCRIPTIONS

MODULE **Types**

	Types

No Documentation Found

Children

1. t_RecordID : No Documentation Found

2. t_FieldNumber : No Documentation Found

3. t_FieldReal : No Documentation Found

4. t_FieldSign : No Documentation Found

5. t_Discrete : No Documentation Found

6. t_Item : No Documentation Found

7. t_Count : No Documentation Found

8. t_Work_Item : No Documentation Found

9. AnyField : No Documentation Found

10. NumericField : No Documentation Found

11. DiscreteField : No Documentation Found

12. Layout_Model : No Documentation Found

13. Classify_Result : No Documentation Found

14. l_result : No Documentation Found

15. Confusion_Detail : No Documentation Found

16. ItemElement : No Documentation Found

17. t_node : No Documentation Found

18. t_level : No Documentation Found

19. NodeID : No Documentation Found

ATTRIBUTE t_RecordID

Types \

t_RecordID

No Documentation Found

RETURN **UNSIGNED8** —

ATTRIBUTE t_FieldNumber

Types \

t_FieldNumber

No Documentation Found

RETURN **UNSIGNED4** —

ATTRIBUTE t_FieldReal

Types \

t_FieldReal

No Documentation Found

RETURN REAL8 —

ATTRIBUTE t_FieldSign

Types \

t_FieldSign

No Documentation Found

RETURN INTEGER1 —

ATTRIBUTE t_Discrete

Types \

t_Discrete

No Documentation Found

RETURN INTEGER4 —

ATTRIBUTE t_Item

Types \

	t_Item

No Documentation Found

RETURN UNSIGNED4 —

ATTRIBUTE t_Count

Types \

	t_Count

No Documentation Found

RETURN UNSIGNED8 —

ATTRIBUTE t_Work_Item

Types \

	t_Work_Item

No Documentation Found

RETURN UNSIGNED2 —

RECORD AnyField

	AnyField

No Documentation Found

FIELD <u>id</u> ||| UNSIGNED8 — No Doc

FIELD <u>wi</u> ||| UNSIGNED2 — No Doc

FIELD <u>number</u> ||| UNSIGNED4 — No Doc

RECORD NumericField

	NumericField

No Documentation Found

FIELD <u>id</u> ||| UNSIGNED8 — No Doc

FIELD <u>value</u> ||| REAL8 — No Doc

FIELD <u>wi</u> ||| UNSIGNED2 — No Doc

FIELD <u>number</u> ||| UNSIGNED4 — No Doc

RECORD DiscreteField

	DiscreteField

No Documentation Found

FIELD <u>id</u> ||| UNSIGNED8 — No Doc

FIELD <u>value</u> ||| INTEGER4 — No Doc

FIELD <u>wi</u> ||| UNSIGNED2 — No Doc

FIELD <u>number</u> ||| UNSIGNED4 — No Doc

RECORD Layout_Model

Types \

	Layout_Model

No Documentation Found

FIELD <u>id</u> ||| UNSIGNED8 — No Doc

FIELD <u>value</u> ||| REAL8 — No Doc

FIELD <u>wi</u> ||| UNSIGNED2 — No Doc

FIELD <u>number</u> ||| UNSIGNED4 — No Doc

RECORD Classify_Result

Types \

	Classify_Result

No Documentation Found

FIELD <u>id</u> ||| UNSIGNED8 — No Doc

FIELD <u>value</u> ||| INTEGER4 — No Doc

FIELD <u>conf</u> ||| REAL8 — No Doc

FIELD <u>wi</u> ||| UNSIGNED2 — No Doc

FIELD <u>number</u> ||| UNSIGNED4 — No Doc

RECORD l_result

Types \

	l_result

No Documentation Found

FIELD <u>id</u> ||| UNSIGNED8 — No Doc

FIELD <u>value</u> ||| INTEGER4 — No Doc

FIELD <u>conf</u> ||| REAL8 — No Doc

FIELD <u>wi</u> ||| UNSIGNED2 — No Doc

FIELD <u>number</u> ||| UNSIGNED4 — No Doc

RECORD Confusion_Detail

Types \

	Confusion_Detail

No Documentation Found

FIELD <u>predict_class</u> ||| INTEGER4 — No Doc

FIELD <u>wi</u> ||| UNSIGNED2 — No Doc

FIELD <u>occurs</u> ||| UNSIGNED4 — No Doc

FIELD <u>actual_class</u> ||| INTEGER4 — No Doc

FIELD <u>classifier</u> ||| UNSIGNED4 — No Doc

FIELD <u>correct</u> ||| BOOLEAN — No Doc

RECORD ItemElement

ItemElement

No Documentation Found

FIELD <u>id</u> ||| UNSIGNED8 — No Doc
FIELD <u>value</u> ||| UNSIGNED4 — No Doc
FIELD <u>wi</u> ||| UNSIGNED2 — No Doc

ATTRIBUTE t_node

t_node

No Documentation Found

RETURN INTEGER4 —

ATTRIBUTE t_level

t_level

No Documentation Found

RETURN UNSIGNED2 —

RECORD NodeID

NodeID

No Documentation Found

FIELD **node_id** ||| INTEGER4 — No Doc

FIELD **level** ||| UNSIGNED2 — No Doc

FIELD **wi** ||| UNSIGNED2 — No Doc

Interfaces

Table of Contents

IClassify.ecl
Interface definition for Classification
IRegression.ecl
Interface Definition for Regression Modules Regression learns a function that maps a set of input data to one or more output variables

IClassify

IMPORTS

ML_Core | ML_Core.Types |

DESCRIPTIONS

MODULE IClassify

IClassify

Interface definition for Classification. Actual implementation modules will probably take parameters.

Children

1. GetModel : Calculate the model to fit the observation data to the observed classes

2. Classify : Classify the observations using a model

3. Report : Report the confusion matrix for the classifier and training data

FUNCTION GetModel

IClassify \

DATASET(Types.Layout_Model)	GetModel
(DATASET(Types.NumericField) observations, DATASET(Types.DiscreteField) classifications)	

Calculate the model to fit the observation data to the observed classes.

PARAMETER <u>**observations**</u> ||| TABLE (NumericField) — the observed explanatory values

PARAMETER <u>**classifications**</u> ||| TABLE (DiscreteField) — the observed classification used to build the model

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 value })** — the encoded model

---

FUNCTION Classify

IClassify \\

DATASET(Types.Classify_Result)	Classify
(DATASET(Types.Layout_Model) model, DATASET(Types.NumericField) new_observations)	

Classify the observations using a model.

PARAMETER <u>**new_observations**</u> ||| TABLE (NumericField) — observations to be classified

PARAMETER <u>**model**</u> ||| TABLE (Layout_Model) — The model, which must be produced by a corresponding getModel function.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , INTEGER4 value , REAL8 conf })** — Classification with a confidence value

---

FUNCTION Report

IClassify \\

`DATASET(Types.Confusion_Detail)`	**Report**
`(DATASET(Types.Layout_Model) model,` `DATASET(Types.NumericField) observations,` `DATASET(Types.DiscreteField) classifications)`	

Report the confusion matrix for the classifier and training data.

PARAMETER   **classifications** ||| TABLE (DiscreteField) — the classifications associated with the observations

PARAMETER   **observations** ||| TABLE (NumericField) — the explanatory values.

PARAMETER   **model** ||| TABLE (Layout_Model) — the encoded model

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 classifier , INTEGER4 actual_class , INTEGER4 predict_class , UNSIGNED4 occurs , BOOLEAN correct })** — the confusion matrix showing correct and incorrect results

IRegression

IMPORTS

ML_Core | ML_Core.Types |

DESCRIPTIONS

MODULE IRegression

IRegression
`(DATASET(NumericField) X=empty_data, DATASET(NumericField) Y=empty_data)`

Interface Definition for Regression Modules Regression learns a function that maps a set of input data to one or more output variables. The resulting learned function is known as the model. That model can then be used repetitively to predict (i.e. estimate) the output value(s) based on new input data.

PARAMETER **Y** ||| TABLE (NumericField) — The dependent variable(s) in DATASET(NumericField) format. Each statistical unit (e.g. record) is identified by 'id', and each feature is identified by field number (i.e. 'number').

PARAMETER **X** ||| TABLE (NumericField) — The independent data in DATASET(NumericField) format. Each statistical unit (e.g. record) is identified by 'id', and each feature is identified by field number (i.e. 'number').

Children

1. GetModel : Calculate and return the 'learned' model The model may be persisted and later used to make predictions using 'Predict' below

2. Predict : Predict the output variable(s) based on a previously learned model

---

ATTRIBUTE GetModel

IRegression \

`DATASET(Layout_Model)`	**GetModel**

Calculate and return the 'learned' model The model may be persisted and later used to make predictions using 'Predict' below.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 value })** — DATASET(LayoutModel) describing the learned model parameters

---

FUNCTION Predict

IRegression \

`DATASET(NumericField)`	**Predict**
`(DATASET(NumericField) newX, DATASET(Layout_Model) model)`	

Predict the output variable(s) based on a previously learned model

PARAMETER <u>**newX**</u> ||| TABLE (NumericField) — DATASET(NumericField) containing the X values to b predicted.

PARAMETER <u>**model**</u> ||| TABLE (Layout_Model) — No Doc

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 value })** — DATASET(NumericField) containing one entry per observation (i.e. id) in newX. This represents the predicted values for Y.

Math

Table of Contents

Beta.ecl
Return the beta value of two positive real numbers, x and y
Distributions.ecl
DoubleFac.ecl
The 'double' factorial is defined for ODD n and is the product of all the odd numbers up to and including that number
Fac.ecl
Factorial function
gamma.ecl
Return the value of gamma function of real number x A wrapper for the standard C tgamma function
log_gamma.ecl
Return the value of the log gamma function of the absolute value of X
lowerGamma.ecl
Return the lower incomplete gamma value of two real numbers,
NCK.ecl
Poly.ecl
Evaluate a polynomial from a set of co-effs
StirlingFormula.ecl
Stirling's formula
upperGamma.ecl
Return the upper incomplete gamma value of two real numbers, x and y

ML_Core/ Math/
Beta

[Go Up](#)

IMPORTS

ML_Core.Math |

DESCRIPTIONS

FUNCTION **Beta**

Beta
(REAL8 x, REAL8 y)

Return the beta value of two positive real numbers, x and y

PARAMETER <u>y</u> ||| REAL8 — the value of the second number

PARAMETER <u>x</u> ||| REAL8 — the value of the first number

RETURN **REAL8** — the beta value

Distributions

[Go Up](#)

IMPORTS

ML_Core.Constants | ML_Core.Math |

DESCRIPTIONS

MODULE **Distributions**

	Distributions

No Documentation Found

Children

1. Normal_CDF : Cumulative Distribution of the standard normal distribution, the probability that a normal random variable will be smaller than x standard deviations above or below the mean

2. Normal_PPF : Normal Distribution Percentage Point Function

3. T_CDF : Students t distribution integral evaluated between negative infinity and x

4. T_PPF : Percentage point function for the T distribution

5. Chi2_CDF : The cumulative distribution function for the Chi Square distribution

6. Chi2_PPF : The Chi Squared PPF function

FUNCTION Normal_CDF

REAL8	Normal__CDF
(REAL8 x)	

Cumulative Distribution of the standard normal distribution, the probability that a normal random variable will be smaller than x standard deviations above or below the mean. Taken from C/C++ Mathematical Algorithms for Scientists and Engineers, n. Shammas, McGraw-Hill, 1995

PARAMETER **x** ||| REAL8 — the number of standard deviations

RETURN **REAL8** —

RETURNS probability of exceeding x.

FUNCTION Normal_PPF

REAL8	Normal__PPF
(REAL8 x)	

Normal Distribution Percentage Point Function. Translated from C/C++ Mathematical Algorithms for Scientists and Engineers, N. Shammas, McGraw-Hill, 1995

PARAMETER **x** ||| REAL8 — probability

RETURN **REAL8** —

RETURNS number of standard deviations from the mean

FUNCTION T_CDF

REAL8	**T_CDF**
(REAL8 x, REAL8 df)	

Students t distribution integral evaluated between negative infinity and x. Translated from NIST SEL DATAPAC Fortran TCDF.f source

PARAMETER <u>df</u> ||| REAL8 — degrees of freedom

PARAMETER <u>x</u> ||| REAL8 — value of the evaluation

RETURN **REAL8** —

RETURNS the probability that a value will be less than the specified value

---

FUNCTION T_PPF

REAL8	**T_PPF**
(REAL8 x, REAL8 df)	

Percentage point function for the T distribution. Translated from NIST SEL DATAPAC Fortran TPPF.f source

PARAMETER <u>df</u> ||| REAL8 — No Doc

PARAMETER <u>x</u> ||| REAL8 — No Doc

RETURN **REAL8** —

FUNCTION Chi2_CDF

Distributions \

REAL8	**Chi2_CDF**
(REAL8 x, REAL8 df)	

The cumulative distribution function for the Chi Square distribution. the CDF for the specfied degrees of freedom. Translated from the NIST SEL DATAPAC Fortran subroutine CHSCDF.

PARAMETER **df** ||| REAL8 — No Doc

PARAMETER **x** ||| REAL8 — No Doc

RETURN **REAL8** —

FUNCTION Chi2_PPF

Distributions \

REAL8	**Chi2_PPF**
(REAL8 x, REAL8 df)	

The Chi Squared PPF function. Translated from the NIST SEL DATAPAC Fortran subroutine CHSPPF.

PARAMETER **df** ||| REAL8 — No Doc

PARAMETER **x** ||| REAL8 — No Doc

RETURN **REAL8** —

DoubleFac

DESCRIPTIONS

EMBED **DoubleFac**

REAL8	**DoubleFac**
(INTEGER2 i)	

The 'double' factorial is defined for ODD n and is the product of all the odd numbers up to and including that number. We are extending the meaning to even numbers to mean the product of the even numbers up to and including that number. Thus DoubleFac(8) = 8*6*4*2 We also defend against i < 2 (returning 1.0)

PARAMETER **i** ||| INTEGER2 — the value used in the calculation

RETURN **REAL8** — the factorial of the sequence, declining by 2

Fac

DESCRIPTIONS

EMBED **Fac**

REAL8	**Fac**
(UNSIGNED2 i)	

Factorial function

PARAMETER **i** ||| UNSIGNED2 — the value used, (i)(i-1)(i-2)...(2)

RETURN **REAL8** — the factorial i!

gamma

DESCRIPTIONS

EMBED **gamma**

`REAL8`	**gamma**
`(REAL8 x)`	

Return the value of gamma function of real number x A wrapper for the standard C tgamma function.

PARAMETER **x** ||| REAL8 — the input x

RETURN **REAL8** — the value of GAMMA evaluated at x

log_gamma

DESCRIPTIONS

EMBED **log_gamma**

REAL8	log_gamma
(REAL8 x)	

Return the value of the log gamma function of the absolute value of X. A wrapper for the standard C lgamma function. Avoids the race condition found on some platforms by taking the absolute value of the of the input argument.

PARAMETER **x** ||| REAL8 — the input x

RETURN **REAL8** — the value of the log of the GAMMA evaluated at ABS(x)

lowerGamma

DESCRIPTIONS

EMBED **lowerGamma**

`REAL8`	**lowerGamma**
`(REAL8 x, REAL8 y)`	

Return the lower incomplete gamma value of two real numbers, x and y

PARAMETER <u>y</u> ||| REAL8 — the value of the second number

PARAMETER <u>x</u> ||| REAL8 — the value of the first number

RETURN **REAL8** — the lower incomplete gamma value

NCK

IMPORTS

ML_Core.Math |

DESCRIPTIONS

FUNCTION NCK

REAL8	**NCK**
`(INTEGER2 N, INTEGER2 K)`	

No Documentation Found

PARAMETER <u>n</u> ||| INTEGER2 — No Doc

PARAMETER <u>k</u> ||| INTEGER2 — No Doc

RETURN **REAL8** —

Poly

DESCRIPTIONS

EMBED **Poly**

`REAL8`	**Poly**
`(REAL8 x, SET OF REAL8 Coeffs)`	

Evaluate a polynomial from a set of co-effs. Co-effs 1 is assumed to be the HIGH order of the equation. Thus for ax^2+bx+c - the set would need to be Coef := [a,b,c];

PARAMETER **Coeffs** ||| SET (REAL8) — a set of coefficients forthe polynomial. The ALL set is considered to be all zero values

PARAMETER **x** ||| REAL8 — the value of x in the polynomial

RETURN **REAL8** — value of the polynomial at x

StirlingFormula

Go Up

IMPORTS

ML_Core.Math | ML_Core.Constants |

DESCRIPTIONS

FUNCTION **StirlingFormula**

StirlingFormula
`(REAL x)`

Stirling's formula

PARAMETER **x** ||| REAL8 — the point of evaluation

RETURN **REAL8** — evaluation result

upperGamma

Go Up

DESCRIPTIONS

EMBED **upperGamma**

`REAL8`	**upperGamma**
`(REAL8 x, REAL8 y)`	

Return the upper incomplete gamma value of two real numbers, x and y.

PARAMETER <u>y</u> ||| REAL8 — the value of the second number

PARAMETER <u>x</u> ||| REAL8 — the value of the first number

RETURN **REAL8** — the upper incomplete gamma value

Tests

Table of Contents

Check_Dist.ecl
field_aggregates.ecl
generate.ecl
test_appends.ecl
test_discrete.ecl
to_from.ecl
Validate_Betas.ecl
Validate_Gammas.ecl

Check_Dist

IMPORTS

ML_Core.Math.Distributions | ML_Core | python |

DESCRIPTIONS

ATTRIBUTE **Check_Dist**

	Check_Dist

No Documentation Found

RETURN —

field_aggregates

IMPORTS

ML_Core | ML_Core.Types |

DESCRIPTIONS

ATTRIBUTE **field_aggregates**

field_aggregates

No Documentation Found

RETURN —

generate

IMPORTS

ML_Core |

DESCRIPTIONS

ATTRIBUTE **generate**

generate

No Documentation Found

RETURN —

test_appends

IMPORTS

ML_Core | std.system.thorlib |

DESCRIPTIONS

ATTRIBUTE **test_appends**

test_appends

No Documentation Found

RETURN —

test_discrete

IMPORTS

ML_Core | ML_Core.Types |

DESCRIPTIONS

ATTRIBUTE **test_discrete**

test_discrete

No Documentation Found

RETURN —

to_from

IMPORTS

ML_Core | ML_Core.Types |

DESCRIPTIONS

ATTRIBUTE **to_from**

to_from

No Documentation Found

RETURN —

Validate_Betas

[Go Up]

IMPORTS

ML_Core | ML_Core.Math | python |

DESCRIPTIONS

ATTRIBUTE **Validate_Betas**

	Validate_Betas

No Documentation Found

RETURN —

Validate_Gammas

[Go Up](#)

IMPORTS

ML_Core | ML_Core.Math | python |

DESCRIPTIONS

ATTRIBUTE **Validate_Gammas**

	Validate_Gammas

No Documentation Found

RETURN —

Utils

Table of Contents

Fat

IMPORTS

ML_Core.Types |

DESCRIPTIONS

FUNCTION **Fat**

`DATASET(Types.NumericField)`	**Fat**
`(DATASET(Types.NumericField) d0, Types.t_FieldReal v=0)`	

Will take a potentially sparse file d and fill in the missing with value v for Numeric Field datasets

PARAMETER **v** ||| REAL8 — The value to assign missing records

PARAMETER **d0** ||| TABLE (NumericField) — They myriad format Numeric Field dataset to be filled

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 value })** — A full Numeric Field dataset with every field populated

FatD

IMPORTS

ML_Core.Types |

DESCRIPTIONS

FUNCTION FatD

`DATASET(Types.DiscreteField)`	**FatD**
`(DATASET(Types.DiscreteField) d0, Types.t_Discrete v=0)`	

Will take a potentially sparse file d and fill in the missing with value v for Discrete Field datasets

PARAMETER **v** ||| INTEGER4 — The value to assign missing records

PARAMETER **d0** ||| TABLE (DiscreteField) — They myriad format Discrete Field dataset to be filled

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , INTEGER4 value })** — A full Discrete Field dataset with every field populated

Gini

DESCRIPTIONS

MACRO **Gini**

Gini
`(infile, pivot, target, wi_name='wi')`

Creates a file of pivot/target pairs with a Gini impurity value.

PARAMETER <u>**target**</u> ||| INTEGER8 — the name of the field used as the target

PARAMETER <u>**wi_name**</u> ||| INTEGER8 — the name of the work item field, default is "wi" return A table by Work Item and Pivot value giving count and Gini impurity value

PARAMETER <u>**infile**</u> ||| INTEGER8 — the input file, any type with a work item field

PARAMETER <u>**pivot**</u> ||| INTEGER8 — the name of the pivot field

RETURN **BOOLEAN** —

ML_Core/ Utils/

SequenceInField

DESCRIPTIONS

MACRO **SequenceInField**

SequenceInField
`(infile,infield,seq,wi_name='wi')`

Given a file which is sorted by the work item identifier and INFIELD (and possibly other values), add sequence numbers within the range of each infield. Slighly elaborate code is to avoid having to partition the data to one value of infield per node and to work with very large numbers of records where a global count project would be inappropriate. This is useful for assigning rank positions with the groupings.

PARAMETER **wi_name** ||| INTEGER8 — work item field name, default is wi

PARAMETER **infile** ||| INTEGER8 — the input file, any type

PARAMETER **seq** ||| INTEGER8 — name of the field to receive the sequence number

PARAMETER **infield** ||| INTEGER8 — field name of grouping field

RETURN **BOOLEAN** — a file of the same type with sequence numbers applied

PBblas

Name	PBblas
Version	3.0.1
Description	Parallel Block Basic Linear Algebra Subsystem
License	http://www.apache.org/licenses/LICENSE-2.0
Copyright	Copyright (C) 2016, 2017 HPCC Systems
Authors	HPCCSystems
DependsOn	ML_Core
Platform	6.2.0

Table of Contents

Apply2Elements.ecl
Apply a function to each element of the matrix Use PBblas.IElementFunc as the prototype function
asum.ecl
Absolute sum – the "Entrywise" 1-norm
axpy.ecl
Implements alpha*X + Y
Constants.ecl
Converted.ecl
Module to convert between ML_Core/Types Field layouts (i.e
ExtractTri.ecl
Extract the upper or lower triangle from the composite output from getrf (LU Factorization)
gemm.ecl
Extended Parallel Block Matrix Multiplication Module Implements: Result = alpha * op(A)op(B) + beta * C
getrf.ecl
LU Factorization Splits a matrix into Lower and Upper triangular factors Produces composite LU matrix for the diagonal blocks
HadamardProduct.ecl

Element-wise multiplication of X * Y
[IElementFunc.ecl](#)
Function prototype for a function to apply to each element of the
[MatUtils.ecl](#)
Provides various utility attributes for manipulating cell-based matrixes
[potrf.ecl](#)
Implements Cholesky factorization of A = U**T * U if Triangular.Upper requested or A = L * L**T if Triangualr.Lower is requested
[scal.ecl](#)
Scale a matrix by a constant Result is alpha * X This supports a "myriad" style interface in that X may be a set of independent matrices separated by different work-item ids
[tran.ecl](#)
Transpose a matrix and sum into base matrix
[trsm.ecl](#)
Partitioned block parallel triangular matrix solver
[Types.ecl](#)
Types for the Parallel Block Basic Linear Algebra Sub-programs support WARNING: attributes marked with WARNING can not be changed without making corresponding changes to the C++ attributes
[Vector2Diag.ecl](#)
Convert a vector into a diagonal matrix

Apply2Elements

IMPORTS

PBblas | PBblas.Types | std.blas |

DESCRIPTIONS

FUNCTION Apply2Elements

DATASET(Layout_Cell)	Apply2Elements
(DATASET(Layout_Cell) X, IElementFunc f)	

Apply a function to each element of the matrix Use PBblas.IElementFunc as the prototype function. Input and ouput may be a single matrix, or myriad matrixes with different work item ids.

PARAMETER **f** ||| FUNCTION [REAL8 , UNSIGNED4 , UNSIGNED4] (REAL8) — A function based on the IElementFunc prototype

PARAMETER **X** ||| TABLE (Layout_Cell) — A matrix (or multiple matrices) in Layout_Cell form

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }** **)** — A matrix (or multiple matrices) in Layout_Cell form

SEE PBblas/IElementFunc

SEE PBblas/Types.Layout_Cell

PBblas/

asum

IMPORTS

PBblas | PBblas.Types | PBblas.internal | PBblas.internal.Types |
PBblas.internal.MatDims | PBblas.internal.Converted | std.blas |

DESCRIPTIONS

FUNCTION asum

DATASET(Layout_Norm)	asum
(DATASET(Layout_Cell) X)	

Absolute sum – the "Entrywise" 1-norm Compute SUM(ABS(X))

PARAMETER **X** ||| TABLE (Layout_Cell) — Matrix or set of matrices in Layout_Cell format

RETURN **TABLE ({ UNSIGNED2 wi_id , REAL8 v })** — DATASET(Layout_Norm) with one record per work item

SEE PBblas/Types.Layout_Cell

axpy

[Go Up](#)

IMPORTS

PBblas | PBblas.Types |

DESCRIPTIONS

FUNCTION **axpy**

`DATASET(Layout_Cell)`	**axpy**
`(value_t alpha, DATASET(Layout_Cell) X, DATASET(Layout_Cell) Y)`	

Implements alpha*X + Y X and Y must have same shape

PARAMETER **Y** ||| TABLE (Layout_Cell) — Y matrix in DATASET(Layout_Call) form

PARAMETER **alpha** ||| REAL8 — Scalar multiplier for the X matrix

PARAMETER **X** ||| TABLE (Layout_Cell) — X matrix in DATASET(Layout_Cell) form

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }) — Matrix in DATASET(Layout_Cell) form**

SEE PBblas/Types.layout_cell

Constants

DESCRIPTIONS

MODULE **Constants**

	Constants

No Documentation Found

Children

1. Block_Minimum : No Documentation Found

2. Block_NoSplit : No Documentation Found

3. Block_Maximum : No Documentation Found

4. Block_Vec_Rows : No Documentation Found

5. Dimension_Incompat : No Documentation Found

6. Dimension_IncompatZ : No Documentation Found

7. Distribution_Error : No Documentation Found

8. Distribution_ErrorZ : No Documentation Found

9. Not_Square : No Documentation Found

10. Not_SquareZ : No Documentation Found

11. Not_PositiveDef : No Documentation Found

12. Not_PositiveDefZ : No Documentation Found

13. Not_Single_Block : No Documentation Found

14. Not_Single_BlockZ : No Documentation Found

ATTRIBUTE **Block_Minimum**

Constants \

Block_Minimum

No Documentation Found

RETURN **INTEGER8** —

ATTRIBUTE **Block_NoSplit**

Constants \

Block_NoSplit

No Documentation Found

RETURN **INTEGER8** —

ATTRIBUTE **Block_Maximum**

Constants \

Block_Maximum

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE Block_Vec_Rows

Constants \

Block_Vec_Rows

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE Dimension_Incompat

Constants \

Dimension_Incompat

No Documentation Found

RETURN STRING34 —

ATTRIBUTE Dimension_IncompatZ

Constants \

Dimension_IncompatZ

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE Distribution_Error

Constants \

Distribution_Error

No Documentation Found

RETURN STRING32 —

ATTRIBUTE Distribution_ErrorZ

Constants \

Distribution_ErrorZ

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE Not_Square

Constants \

Not_Square

No Documentation Found

RETURN STRING20 —

ATTRIBUTE Not_SquareZ

Constants \

	Not_SquareZ

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE Not_PositiveDef

Constants \

	Not_PositiveDef

No Documentation Found

RETURN STRING40 —

ATTRIBUTE Not_PositiveDefZ

Constants \

	Not_PositiveDefZ

No Documentation Found

ATTRIBUTE Not_Single_Block

Constants \

Not_Single_Block

No Documentation Found

RETURN STRING28 —

ATTRIBUTE Not_Single_BlockZ

Constants \

Not_Single_BlockZ

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE Not_Block_Vector

Constants \

Not_Block_Vector

No Documentation Found

RETURN STRING25 —

ATTRIBUTE Not_Block_VectorZ

Constants \

Not_Block_VectorZ

No Documentation Found

RETURN INTEGER8 —

PBblas/
Converted

IMPORTS

PBblas | PBblas.Types | ML_Core.Types |

DESCRIPTIONS

MODULE Converted

	Converted

Module to convert between ML_Core/Types Field layouts (i.e. NumericField and DiscreteField) and PBblas matrix layout (i.e. Layout_Cell)

Children

1. NFToMatrix : Convert NumericField dataset to Matrix

2. DFToMatrix : Convert DiscreteField dataset to Matrix

3. MatrixToNF : Convert Matrix to NumericField dataset

4. MatrixToDF : Convert Matrix to DiscreteField dataset

FUNCTION NFToMatrix

Converted \

DATASET(Layout_Cell)	**NFToMatrix**
(DATASET(NumericField) recs)	

Convert NumericField dataset to Matrix

PARAMETER <u>recs</u> ||| TABLE (NumericField) — Record Dataset in DATASET(NumericField) format

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }** **) —** Matrix in DATASET(Layout_Cell) format

SEE PBblas/Types.Layout_Cell

SEE ML_Core/Types.NumericField

FUNCTION DFToMatrix

Converted \

DATASET(Layout_Cell)	**DFToMatrix**
(DATASET(DiscreteField) recs)	

Convert DiscreteField dataset to Matrix

PARAMETER <u>recs</u> ||| TABLE (DiscreteField) — Record Dataset in DATASET(DiscreteField) format

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }** **) —** Matrix in DATASET(Layout_Cell) format

SEE PBblas/Types.Layout_Cell

SEE ML_Core/Types.DiscreteField

FUNCTION MatrixToNF

Converted \

DATASET(NumericField)	**MatrixToNF**
(DATASET(Layout_Cell) mat)	

Convert Matrix to NumericField dataset

PARAMETER <u>mat</u> ||| TABLE (Layout_Cell) — Matrix in DATASET(Layout_Cell) format

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 value })** — NumericField Dataset

SEE PBblas/Types.Layout_Cell

SEE ML_Core/Types.NumericField

FUNCTION MatrixToDF

Converted \

DATASET(DiscreteField)	**MatrixToDF**
(DATASET(Layout_Cell) mat)	

Convert Matrix to DiscreteField dataset

PARAMETER <u>mat</u> ||| TABLE (Layout_Cell) — Matrix in DATASET(Layout_Cell) format

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , INTEGER4 value })** — DiscreteField Dataset

SEE PBblas/Types.Layout_Cell

SEE ML_Core/Types.DiscreteField

PBblas/

ExtractTri

[Go Up](#)

IMPORTS

PBblas | std.blas | PBblas.Types | PBblas.internal | PBblas.internal.Types |
PBblas.internal.MatDims | PBblas.internal.Converted |

DESCRIPTIONS

FUNCTION **ExtractTri**

`DATASET(Layout_Cell)`	**ExtractTri**
`(Triangle tri, Diagonal dt, DATASET(Layout_Cell) A)`	

Extract the upper or lower triangle from the composite output from getrf (LU Factorization).

PARAMETER <u>**tri**</u> ||| UNSIGNED1 — Triangle type: Upper or Lower (see Types.Triangle)

PARAMETER <u>**dt**</u> ||| UNSIGNED1 — Diagonal type: Unit or non unit (see Types.Diagonal)

PARAMETER <u>**A**</u> ||| TABLE (Layout_Cell) — Matrix of cells. See Types.Layout_Cell

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }
)** — Matrix of cells in Layout_Cell format representing a triangular matrix (upper or lower)

SEE Std.PBblas.Types

gemm

IMPORTS

PBblas | PBblas.Types | PBblas.internal | PBblas.internal.Types | std.blas |
PBblas.internal.MatDims | std.system.Thorlib |

DESCRIPTIONS

FUNCTION gemm

DATASET(Layout_Cell)	gemm
(BOOLEAN transposeA, BOOLEAN transposeB, value_t alpha, DATASET(Layout_Cell) A_in, DATASET(Layout_Cell) B_in, DATASET(Layout_Cell) C_in=emptyC, value_t beta=0.0)	

Extended Parallel Block Matrix Multiplication Module Implements: Result = alpha * op(A)op(B) + beta
* C. op is No Transpose or Transpose. Multiplies two matrixes A and B, with an optional pre-multiply
transpose for each Optionally scales the product by the scalar "alpha". Then adds an optional C matrix
to the product after scaling C by the scalar "beta". A, B, and C are specified as DATASET(Layout_Cell),
as is the Resulting matrix. Layout_Cell describes a sparse matrix stored as a list of x, y, and value. This
interface also provides a "Myriad" capability allowing multiple similar operations to be performed on
independent sets of matrixes in parallel. This is done by use of the work-item id (wi_id) in each cell of
the matrixes. Cells with the same wi_id are considered part of the same matrix. In the myriad form,
each input matrix A, B, and (optionally) C can contain many independent matrixes. The wi_ids are
matched up such that each operation involves the A, B, and C with the same wi_id. A and B must
therefore contain the same set of wi_ids, while C is optional for any wi_id. The same parameters: alpha,
beta, transposeA, and transposeB are used for all work-items. The result will contain cells from all
provided work-items. Result has same shape as C if provided. Note that matrixes are not explicitly

dimensioned. The shape is determined by the highest value of x and y for each work-item.

PARAMETER **transposeA** ||| BOOLEAN — Boolean indicating whether matrix A should be transposed before multiplying

PARAMETER **A_in** ||| TABLE (Layout_Cell) — 'A' matrix (multiplier) in Layout_Cell format

PARAMETER **beta** ||| REAL8 — A scalar multiplier for beta * C, scales the C matrix before addition. May be omitted.

PARAMETER **C_in** ||| TABLE (Layout_Cell) — Same as above for the 'C' matrix (addend). May be omitted.

PARAMETER **transposeB** ||| BOOLEAN — Same as above but for matrix B

PARAMETER **B_in** ||| TABLE (Layout_Cell) — Same as above for the 'B' matrix (multiplicand)

PARAMETER **alpha** ||| REAL8 — Scalar multiplier for alpha * A * B

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }** **)** — Result matrix in Layout_Cell format.

SEE PBblas/Types.Layout_Cell

getrf

IMPORTS

PBblas | PBblas.Types | PBblas.internal | PBblas.internal.Types | std.blas | PBblas.internal.MatDims | std.system.Thorlib |

DESCRIPTIONS

FUNCTION getrf

DATASET(Layout_Cell)	getrf
(DATASET(Layout_Cell) A)	

LU Factorization Splits a matrix into Lower and Upper triangular factors Produces composite LU matrix for the diagonal blocks. Iterates through the matrix a row of blocks and column of blocks at a time. Partition A into M block rows and N block columns. The A11 cell is a single block. A12 is a single row of blocks with N-1 columns. A21 is a single column of blocks with M-1 rows. A22 is a sub-matrix of M-1 x N-1 blocks. | A11 A12 | | L11 0 | | U11 U12 | | A21 A22 | == | L21 L22 | * | 0 U22 | | L11*U11 L11*U12 | == | L21*U11 L21*U12 + L22*U22 | Based upon PB-BLAS: A set of parallel block basic linear algebra subprograms by Choi and Dongarra This module supports the "Myriad" style interface, allowing many independent problems to be worked on at once. The A matrix can contain multiple matrixes to be factored, indicated by different values for work-item id (wi_id). Note: The returned matrix includes both the upper and lower factors. This matrix can be used directly by trsm which will only use the part indicated by trsm's 'triangle' parameter (i.e. upper or lower). To extract the upper or lower triangle explicitly for other purposes, use the ExtractTri function. When passing the Lower matrix to the triangle solver (trsm), set the "Diagonal" parameter to "UnitTri". This is necessary because both triangular matrixes returned from this function are packed into a square matrix with only one diagonal. By convention, The Lower triangle is assumed to be a Unit Triangle (diagonal all ones), so the diagonal

contained in the returned matrix is for the Upper factor and must be ignored (i.e. assumed to be all ones) when referencing the Lower triangle.

PARAMETER <u>**A**</u> ||| TABLE (Layout_Cell) — The input matrix in Types.Layout_Cell format

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }** **)** — Resulting factored matrix in Layout_Cell format

SEE Types.Layout_Cell

SEE ExtractTri

HadamardProduct

IMPORTS

PBblas | PBblas.internal | PBblas.internal.MatDims | PBblas.Types |
PBblas.internal.Types | PBblas.internal.Converted | std.blas | std.system.Thorlib |

DESCRIPTIONS

FUNCTION **HadamardProduct**

`DATASET(Layout_Cell)`	**HadamardProduct**
`(DATASET(Layout_Cell) X, DATASET(Layout_Cell) Y)`	

Element-wise multiplication of X * Y. Supports the "myriad" style interface – X and Y may contain multiple separate matrixes. Each X will be multiplied by the Y with the same work-item id. Note: This performs element-wise multiplication. For dot-product matrix multiplication, use PBblas.gemm.

PARAMETER **Y** ||| TABLE (Layout_Cell) — A matrix (or multiple matrices) in Layout_Cell form

PARAMETER **X** ||| TABLE (Layout_Cell) — A matrix (or multiple matrices) in Layout_Cell form

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }** **)** — A matrix (or multiple matrices) in Layout_Cell form

SEE PBblas/Types.Layout_Cell

PBblas/

IElementFunc

[Go Up](#)

IMPORTS

PBblas |

DESCRIPTIONS

FUNCTION **IElementFunc**

`value_t`	**IElementFunc**
`(value_t v, dimension_t r, dimension_t c)`	

Function prototype for a function to apply to each element of the distributed matrix Base your function on this prototype:

PARAMETER <u>**v**</u> ||| REAL8 — Input value

PARAMETER <u>**c**</u> ||| UNSIGNED4 — Column number (1 based)

PARAMETER <u>**r**</u> ||| UNSIGNED4 — Row number (1 based)

RETURN **REAL8** — Output value

SEE PBblas/Apply2Elements

MatUtils

IMPORTS

PBblas | PBblas.Types | PBblas.internal | PBblas.internal.Types | PBblas.internal.MatDims |

DESCRIPTIONS

MODULE MatUtils

MatUtils

Provides various utility attributes for manipulating cell-based matrixes

SEE Std/PBblas/Types.Layout_Cell

Children

1. GetWorkItems : Get a list of work-item ids from a matrix containing one or more work items

2. InsertCols : Insert one or more columns of a fixed value into a matrix

3. Transpose : Transpose a matrix This attribute supports the myriad interface

FUNCTION GetWorkItems

`DATASET(Layout_WI_ID)`	**GetWorkItems**
`(DATASET(Layout_Cell) cells)`	

Get a list of work-item ids from a matrix containing one or more work items

PARAMETER <u>cells</u> ||| TABLE (Layout_Cell) — A matrix in Layout_Cell format

RETURN **TABLE ({ UNSIGNED2 wi_id })** — DATASET(Layout_WI_ID), one record per work-item

SEE PBblas/Types.Layout_Cell

SEE PBblas/Types.Layout_WI_ID

FUNCTION InsertCols

`DATASET(Layout_Cell)`	**InsertCols**
`(DATASET(Layout_Cell) M, UNSIGNED cols_to_insert=1, value_t insert_val=1)`	

Insert one or more columns of a fixed value into a matrix. Columns are inserted before the first original column. This attribute supports the myriad interface. Multiple independent matrixes can be represented by M.

PARAMETER <u>cols_to_insert</u> ||| UNSIGNED8 — the number of columns to insert, default 1

PARAMETER <u>insert_val</u> ||| REAL8 — the value for each cell of the new column(s), default 0

PARAMETER <u>M</u> ||| TABLE (Layout_Cell) — the input matrix

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v })** — matrix in Layout_Cell format with additional column(s)

FUNCTION Transpose

`DATASET(Layout_Cell)`	**Transpose**
`(DATASET(Layout_Cell) M)`	

Transpose a matrix This attribute supports the myriad interface. Multiple independent matrixes can be represented by M.

PARAMETER **M** ||| TABLE (Layout_Cell) — A matrix represented as DATASET(Layout_Cell)

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }** **)** — Transposed matrix in Layout_Cell format

SEE PBblas/Types.Layout_Cell

potrf

IMPORTS

PBblas | PBblas.Types | std.blas | PBblas.internal | PBblas.internal.Types |
PBblas.internal.MatDims | PBblas.internal.Converted | std.system.Thorlib |

DESCRIPTIONS

FUNCTION potrf

DATASET(Layout_Cell)	potrf
(Triangle tri, DATASET(Layout_Cell) A_in)	

Implements Cholesky factorization of A = U**T * U if Triangular.Upper requested or A = L * L**T if
Triangualr.Lower is requested. The matrix A must be symmetric positive definite.

```
| A11   A12 |       | L11    0  |     | L11**T      L21**T |
| A21   A22 | ==  | L21   L22 | *  | 0               L22  |

                       | L11*L11**T          L11*L21**T       |
                ==  | L21*L11**T   L21*L21**T + L22*L22**T |
```

So, use Cholesky on the first block to get L11. L21 = A21*L11**T**-1 which can be found by dtrsm on
each column block A22' is A22 - L21*L21**T

Based upon PB-BLAS: A set of parallel block basic linear algebra subprograms by Choi and Dongarra

This module supports the "Myriad" style interface, allowing many independent problems to be worked on at once. The A matrix can contain multiple matrixes to be factored, indicated by different values for work-item id (wi_id).

PARAMETER <u>**tri**</u> ||| UNSIGNED1 — Types.Triangle enumeration indicating whether we are looking for the Upper or the Lower factor

PARAMETER <u>**A__in**</u> ||| TABLE (Layout_Cell) — The matrix or matrixes to be factored in Types.Layout_Cell format

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }** **)** — Triangular matrix in Layout_Cell format

SEE Std.PBblas.Types.Layout_Cell

SEE Std.PBblas.Types.Triangle

scal

IMPORTS

PBblas | PBblas.Types |

DESCRIPTIONS

FUNCTION **scal**

`DATASET(Layout_Cell)`	**scal**
`(value_t alpha, DATASET(Layout_Cell) X)`	

Scale a matrix by a constant Result is alpha * X This supports a "myriad" style interface in that X may be a set of independent matrices separated by different work-item ids.

PARAMETER **alpha** ||| REAL8 — A scalar multiplier

PARAMETER **X** ||| TABLE (Layout_Cell) — The matrix(es) to be scaled in Layout_Cell format

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }** **)** — Matrix in Layout_Cell form, of the same shape as X

SEE PBblas/Types.Layout_Cell

tran

IMPORTS

PBblas | PBblas.Types | PBblas.internal | PBblas.internal.Types |

PBblas.internal.MatDims | PBblas.internal.Converted | std.blas | std.system.Thorlib |

DESCRIPTIONS

FUNCTION tran

DATASET(Layout_Cell)	tran
(value_t alpha, DATASET(Layout_Cell) A, value_t beta=0, DATASET(Layout_Cell) C=empty_c)	

Transpose a matrix and sum into base matrix result <== alpha * A**t + beta * C, A is n by m, C is m by n A**T (A Transpose) and C must have same shape

PARAMETER **beta** ||| REAL8 — Scalar multiplier for the C matrix

PARAMETER **alpha** ||| REAL8 — Scalar multiplier for the A**T matrix

PARAMETER **C** ||| TABLE (Layout_Cell) — C matrix in DATASET(Layout_Call) form

PARAMETER **A** ||| TABLE (Layout_Cell) — A matrix in DATASET(Layout_Cell) form

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }** **)** — Matrix in DATASET(Layout_Cell) form alpha * A**T + beta * C

trsm

IMPORTS

PBblas | PBblas.Types | std.blas | PBblas.internal | PBblas.internal.Types |
PBblas.internal.MatDims | PBblas.internal.Converted | std.system.Thorlib |

DESCRIPTIONS

FUNCTION **trsm**

DATASET(Layout_Cell)	trsm
(Side s, Triangle tri, BOOLEAN transposeA, Diagonal diag, value_t alpha, DATASET(Layout_Cell) A_in, DATASET(Layout_Cell) B_in)	

Partitioned block parallel triangular matrix solver. Solves for X using: AX = B or XA = B A is is a square triangular matrix, X and B have the same dimensions. A may be an upper triangular matrix (UX = B or XU = B), or a lower triangular matrix (LX = B or XL = B). Allows optional transposing and scaling of A. Partially based upon an approach discussed by MJ DAYDE, IS DUFF, AP CERFACS. A Parallel Block implementation of Level-3 BLAS for MIMD Vector Processors ACM Tran. Mathematical Software, Vol 20, No 2, June 1994 pp 178-193 and other papers about PB-BLAS by Choi and Dongarra This module supports the "Myriad" style interface, allowing many independent problems to be worked on at once. Corresponding A and B matrixes are related by a common work-item identifier (wi_id) within each cell of the matrix. The returned X matrix will contain cells for the same set of work-items as specified for the A and B matrices.

PARAMETER <u>s</u> ||| UNSIGNED1 — Types.Side enumeration indicating whether we are solving AX = B or XA = B

PARAMETER **A_in** ||| TABLE (Layout_Cell) — The A matrix in Layout_Cell format

PARAMETER **diag** ||| UNSIGNED1 — Types.Diagonal enumeration indicating whether A is a unit matrix or not. This is primarily used after factoring matrixes using getrf (LU factorization). That module produces a factored matrix stored within the same space as the original matrix. Since the diagonal is used by both factors, by convention, the Lower triangle has a unit matrix (diagonal all 1's) while the Upper triangle uses the diagonal cells. Setting this to UnitTri, causes the contents of the diagonal to be ignored, and assumed to be 1. NotUnitTri should be used for most other cases.

PARAMETER **transposeA** ||| BOOLEAN — Boolean indicating whether or not to transpose the A matrix before solving

PARAMETER **B_in** ||| TABLE (Layout_Cell) — The B matrix in Layout_Cell format

PARAMETER **alpha** ||| REAL8 — Multiplier to scale A

PARAMETER **tri** ||| UNSIGNED1 — Types.Triangle enumeration indicating whether we are solving an Upper or Lower triangle.

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }
)** — X solution matrix in Layout_Cell format

SEE Types.Layout_Cell

SEE Types.Triangle

SEE Types.Side

Types

IMPORTS

ML_Core | ML_Core.Types |

DESCRIPTIONS

`MODULE` Types

Types

Types for the Parallel Block Basic Linear Algebra Sub-programs support WARNING: attributes marked with WARNING can not be changed without making corresponding changes to the C++ attributes.

Children

1. dimension_t : Type for matrix dimensions

2. partition_t : Type for partition id – only supports up to 64K partitions

3. work_item_t : Type for work-item id – only supports up to 64K work items

4. value_t : Type for matrix cell values

5. m_label_t : Type for matrix label

6. Triangle : Enumeration for Triangle type

7. Diagonal : Enumeration for Diagonal type

8. Side : Enumeration for Side type

9. t_mu_no : Type for matrix universe number

10. Layout_Cell : Layout for Matrix Cell Main representation of Matrix cell at interface to all PBBlas functions

11. Layout_Norm : Layout for Norm results

ATTRIBUTE dimension_t

Types \

	dimension_t

Type for matrix dimensions. Uses UNSIGNED four as matrixes are not designed to support more than 4 B rows or columns.

RETURN UNSIGNED4 —

ATTRIBUTE partition_t

Types \

	partition_t

Type for partition id – only supports up to 64K partitions

RETURN UNSIGNED2 —

ATTRIBUTE work_item_t

Types \

work__item__t

Type for work-item id – only supports up to 64K work items

RETURN **UNSIGNED2** —

ATTRIBUTE value_t

Types \

value_t

Type for matrix cell values WARNING: type used in C++ attribute

RETURN **REAL8** —

ATTRIBUTE m_label_t

Types \

m__label__t

Type for matrix label. Used for Matrix dimensions (see Layout_Dims) and for partitions (see Layout_Part)

RETURN **STRING3** —

ATTRIBUTE Triangle

Types \

	Triangle

Enumeration for Triangle type WARNING: type used in C++ attribute

RETURN **UNSIGNED1** —

ATTRIBUTE **Diagonal**

Types \

	Diagonal

Enumeration for Diagonal type WARNING: type used in C++ attribute

RETURN **UNSIGNED1** —

ATTRIBUTE **Side**

Types \

	Side

Enumeration for Side type WARNING: type used in C++ attribute

RETURN **UNSIGNED1** —

ATTRIBUTE **t_mu_no**

Types \

t_mu_no

Type for matrix universe number Allow up to 64k matrices in one universe

RETURN **UNSIGNED2** —

RECORD **Layout_Cell**

Types \

Layout_Cell

Layout for Matrix Cell Main representation of Matrix cell at interface to all PBBlas functions. Matrixes are represented as DATASET(Layout_Cell), where each cell describes the row and column position of the cell as well as its value. Only the non-zero cells need to be contained in the dataset in order to describe the matrix since all unspecified cells are considered to have a value of zero. The cell also contains a work-item number that allows multiple separate matrixes to be carried in the same dataset. This supports the "myriad" style interface that allows the same operations to be performed on many different sets of data at once. Note that these matrixes do not have an explicit size. They are sized implicitly, based on the maximum row and column presented in the data. A matrix can be converted to an explicit dense form (see matrix_t) by using the utility module MakeR8Set. This module should only be used for known small matrixes (< 1M cells) or for partitions of a larger matrix. The Converted module provides utility functions to convert to and from a set of partitions (See Layout_parts).

FIELD **v** ||| REAL8 — Real value for the cell

FIELD **y** ||| UNSIGNED4 — 1-based column position within the matrix

FIELD **x** ||| UNSIGNED4 — 1-based row position within the matrix

FIELD **wi_id** ||| UNSIGNED2 — Work Item Number – An identifier from 1 to 64K-1 that separates and identifies individual matrixes

SEE matrix_t

SEE Std/PBblas/MakeR8Set.ecl

SEE Std/PBblas/Converted.ecl WARNING: Used as C++ attribute. Do not change without corresponding changes to MakeR8Set.

RECORD Layout_Norm

Layout_Norm

Layout for Norm results.

FIELD **v** ||| REAL8 — Real value for the norm

FIELD **wi_id** ||| UNSIGNED2 — Work Item Number – An identifier from 1 to 64K-1 that separates and identifies individual matrixes

Vector2Diag

IMPORTS

PBblas | PBblas.internal | PBblas.internal.MatDims | PBblas.Types | PBblas.internal.Types | PBblas.Constants |

DESCRIPTIONS

FUNCTION Vector2Diag

`DATASET(Layout_Cell)`	**Vector2Diag**
`(DATASET(Layout_Cell) X)`	

Convert a vector into a diagonal matrix. The typical notation is D = diag(V). The input X must be a 1 x N column vector or an N x 1 row vector. The resulting matrix, in either case will be N x N, with zero everywhere except the diagonal.

PARAMETER **X** ||| TABLE (Layout_Cell) — A row or column vector (i.e. N x 1 or 1 x N) in Layout_Cell format

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }** **)** — An N x N matrix in Layout_Cell format

SEE PBblas/Types.Layout_cell