root

Go Up

Name	ML_Core
Version	3.2.0
Description	Common definitions for Machine Learning
License	SeeLICENSE.TXT
Copyright	Copyright (C) 2017 HPCC SystemsÂő
Authors	HPCCSystems
Platform	6.2.0

Table of Contents

ModelOps2.ecl This module provides a set of operations to provide manipulation of machine learning models (version 2) in the Types.Layout_Model2 format
ToField.ecl Convert a record-oriented dataset to a cell-oriented NumericField dataset for use with Machine Learning mechanisms
Types.ecl This module provides the major data type definitions for use with the various
Interfaces
Math
Tests
Utils

Analysis

Go Up

IMPORTS

Types |

DESCRIPTIONS

MODULE Analysis

	Analysis

Analyze and assess the effectiveness of a Machine Learning model.

Sub-modules provide support for both Classification and Regression.

Each of the functions in this module support multi-work-item (i.e. Myriad interface) data, as well as multi-variate data (supported by some ML bundles). The number field, which is usually = 1 for uni-variate data is used to distinguish multiple regressors in the case of multi- variate models.

Children

1. Classification : This sub-module provides functions for analyzing and assessing the effectiveness of an ML Classification model
2. Regression : This sub-module provides functions for analyzing and assessing the effectiveness of an ML Regression model

MODULE Classification

Analysis \

	Classification

This sub-module provides functions for analyzing and assessing the effectiveness of an ML Classification model. It can be used with any ML Bundle that supports classification.

Children

1. ClassStats : Given a set of expected dependent values, assess the number and percentage of records that were of each class
2. ConfusionMatrix : Returns the Confusion Matrix, counting the number of cases for each combination of predicted Class and actual Class
3. Accuracy : Assess the overall accuracy of the classification predictions
4. AccuracyByClass : Provides per class accuracy / relevance statistics (e.g

---

FUNCTION ClassStats

Analysis \ Classification \

DATASET(Class_Stats)	**ClassStats**
(DATASET(DiscreteField) actual)	

Given a set of expected dependent values, assess the number and percentage of records that were of each class.

PARAMETER **actual** ||| TABLE (DiscreteField) — The set of training-data or test-data dependent values in DATASET(DiscreteField) format.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 classifier , INTEGER4 class , INTEGER4 classCount , REAL8 classPct })** — DATASET(Class_Stats), one record per work-item, per classifier (i.e. number field) per class.

SEE ML_Core.Types.Class_Stats

FUNCTION ConfusionMatrix

Analysis \ Classification \

DATASET(Confusion_Detail)	**ConfusionMatrix**
(DATASET(DiscreteField) predicted, DATASET(DiscreteField) actual)	

Returns the Confusion Matrix, counting the number of cases for each combination of predicted Class and actual Class.

PARAMETER **actual** ||| TABLE (DiscreteField) — The actual (i.e. expected) values for each id in DATASET(DiscreteField) format.

PARAMETER **predicted** ||| TABLE (DiscreteField) — The predicted values for each id in DATASET(DiscreteField) format.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 classifier , INTEGER4 actual_class , INTEGER4 predict_class , UNSIGNED4 occurs , BOOLEAN correct , REAL8 pctActual , REAL8 pctPred })** — DATASET(Confusion_Detail). One record for each combination of work-item, number (i.e. classifier), predicted class, and actual class.

SEE ML_Core.Types.Confusion_Detail

FUNCTION Accuracy

Analysis \ Classification \

DATASET(Classification_Accuracy)	**Accuracy**
(DATASET(DiscreteField) predicted, DATASET(DiscreteField) actual)	

Assess the overall accuracy of the classification predictions.

ML_Core.Types.Classification_Accuracy provides a detailed description of the return values.

PARAMETER **actual** ||| TABLE (DiscreteField) — The actual (i.e. expected) values for each id in DATASET(DiscreteField) format.

PARAMETER **predicted** ||| TABLE (DiscreteField) — The predicted values for each id in DATASET(DiscreteField) format.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 classifier , UNSIGNED8 recCnt , UNSIGNED8 errCnt , REAL8 Raw_Accuracy , REAL8 PoD , REAL8 PoDE })** — DATASET(Classification_Accuracy). One record for each combination of work-item, and number (i.e. classifier).

SEE ML_Core.Types.Classification_Accuracy

FUNCTION AccuracyByClass

Analysis \ Classification \

DATASET(Class_Accuracy)	**AccuracyByClass**
(DATASET(DiscreteField) predicted, DATASET(DiscreteField) actual)	

Provides per class accuracy / relevance statistics (e.g. Precision / Recall, False-positive Rate).

ML_Core.Types.Class_Accuracy provides a detailed description of the return values.

PARAMETER **actual** ||| TABLE (DiscreteField) — The actual (i.e. expected) values for each id in DATASET(DiscreteField) format.

PARAMETER **predicted** ||| TABLE (DiscreteField) — The predicted values for each id in DATASET(DiscreteField) format.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 classifier , INTEGER4 class , REAL8 precision , REAL8 recall , REAL8 FPR })** — DATASET(Class_Accuracy). One record for each combination of work-item, number (i.e. classifier), and class.

SEE ML_Core.Types.Class_Accuracy

MODULE Regression

Analysis \

	Regression

This sub-module provides functions for analyzing and assessing the effectiveness of an ML Regression model. It can be used with any ML Bundle that supports regression.

Children

1. Accuracy : Assess the overall accuracy of the regression predictions

---

FUNCTION Accuracy

Analysis \ Regression \

DATASET(Regression_Accuracy)	**Accuracy**
(DATASET(NumericField) predicted, DATASET(NumericField) actual)	

Assess the overall accuracy of the regression predictions.

PARAMETER **actual** ||| TABLE (NumericField) — The actual (i.e. expected) values for each id in DATASET(DiscreteField) format.

PARAMETER **predicted** ||| TABLE (NumericField) — The predicted values for each id in DATASET(DiscreteField) format.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 regressor , REAL8 R2 , REAL8 MSE , REAL8 RMSE })** — DATASET(Regression_Accuracy). One record for each combination of work-item, and number (i.e. regressor).

SEE ML_Core.Types.Regression_Accuracy

AppendID

Go Up

DESCRIPTIONS

MACRO AppendID

	AppendID
(dIn,idfield,dOut)	

Macro takes any structured dataset, and appends a unique 1-based record ID column to it. Values will not be sequential and values will not be dense because of data skew. Gaps will appear when data ends on each node. If dense and sequential values are required, use AppendSeqID.

Note that, as a macro, nothing is returned, but attribute named in dOut will be defined to contain the resulting dataset.

Example:

```
ML\_Core.AppendID(dOrig, recID, dOrigWithId);
```

PARAMETER **dIn** ||| INTEGER8 — The name of the input dataset.

PARAMETER **dOut** ||| INTEGER8 — The name of the resulting dataset.

PARAMETER **idfield** ||| INTEGER8 — The name of the field to be appended containing the id for each row.

RETURN —

AppendSeqID

Go Up

DESCRIPTIONS

MACRO AppendSeqID

	AppendSeqID
(dIn,idfield,dOut)	

Macro takes any structured dataset, and appends a unique 1-based record ID column to it. Values will be in data sequence. Note: implemented as a count project, each node processes the data in series instead of parallel. For better cluster performance, use AppendID as long as dense, sequential ids are not needed.

Note that, as a macro, nothing is returned, but attribute named in dOut will be defined to contain the resulting dataset.

Example:

```
ML\_Core.AppendSeqID(dOrig, recID, dOrigWithId);
```

PARAMETER **dIn** ||| INTEGER8 — The name of the input dataset.

PARAMETER **dOut** ||| INTEGER8 — The name of the resulting dataset.

PARAMETER **idfield** ||| INTEGER8 — The name of the field to be appended containing the id for each row.

RETURN —

Config

Go Up

DESCRIPTIONS

MODULE Config

	Config

Global configuration constants that can be modified if needed.

Children

1. MaxLookup : The maximum amount of data to use in a LOOKUP JOIN
2. Discrete : The default number of groups to use when discretizing data
3. RoundingError : The tolerance for rounding error

ATTRIBUTE MaxLookup

Config \

	MaxLookup

The maximum amount of data to use in a LOOKUP JOIN.

RETURN INTEGER8 —

ATTRIBUTE Discrete

Config \

	Discrete

The default number of groups to use when discretizing data.

RETURN **INTEGER8** —

ATTRIBUTE RoundingError

Config \

	RoundingError

The tolerance for rounding error.

RETURN **REAL8** —

Constants

Go Up

DESCRIPTIONS

MODULE Constants

	Constants

Useful constants used in ML.

Children

1. Pi : Constant PI
2. Root_2 : Constant square root of 2

ATTRIBUTE Pi

Constants \

	Pi

Constant PI

RETURN **REAL8** —

ATTRIBUTE Root_2

Constants \

	Root_2

Constant square root of 2

RETURN **REAL8** —

Discretize

Go Up

IMPORTS

Types |

DESCRIPTIONS

MODULE Discretize

	Discretize

This module is used to turn a dataset of NumericFields into a dataset of DiscreteFields. This is not quite as trivial as it seems as there are a number of different ways to make the underlying data discrete; and even within one method there may be different parameters. Further - it is quite probable that different methods are going to be desired for each field.

There are two methods of interfacing:

- Call a discretization method directly to apply to all fields.
- Build a set of instructions on how to discretize each field and then call 'Do'.

The record format 'r_Method is used to build the set of instructions in the latter case.

For each discretization method (e.g. ByRounding), there is a corresponding attribute preceded by 'i_' that is used to build the r_Method instruction for using that method (e.g. i_ByRounding).

Three methods are currently provided:

- ByRounding – Numerically round the number to the nearest integer.

- ByBucketing – Split the range of each variable into a number of evenly spaced buckets.
- ByTiling – Splits the datapoints into an ordered set of equal-sized groups.

Children

1. c_Method : Enumerate the available discretization methods
2. r_Method : This format is used to construct an 'instruction stream' to allow a dataset to be discretized according to a set of instructions which are in (meta)data
3. i_ByRounding : Construct an instruction (rMethod) that will cause certain fields to be discretized by rounding
4. ByRounding : Round the values passed in to create a discrete element Scale is applied (by multiplication) first and can be used to bring the data into a desired range (rParam1), Delta is applied (by addition) second and can be used to re-base a range OR to cause truncation or roundup as required (rParam2)
5. i_ByBucketing : Construct an instruction (rMethod) that will cause certain fields to be discretized by bucketing
6. ByBucketing : Allocates a continuous variable into one of N buckets based upon an equal division of the RANGE of the variable
7. i_ByTiling : Construct an instruction (rMethod) that will cause certain fields to be discretized by tiling
8. ByTiling : Allocate a continuous variable into one of N groups such that each group (tile) contains roughly the same number of entries and that all of the elements of group 2 have a higher value than group 1, etc
9. Do : Execute a set of discretization instructions in order to discretize all of the fields of the dataset using the appropriate methods

---

ATTRIBUTE c_Method

Discretize \

	c_Method

Enumerate the available discretization methods.

RETURN UNSIGNED4 —

VALUE Rounding = 1

VALUE Bucketing = 2

VALUE Tiling = 3

RECORD r_Method

Discretize \

	r_Method

This format is used to construct an 'instruction stream' to allow a dataset to be discretized according to a set of instructions which are in (meta)data. It can be created directly, though the preferred method is to call i_ByRounding(...), i_ByBucketing(...), or i_ByTiling(...) to create each record.

FIELD **rParam2** ||| REAL8 — The second real parameter.

FIELD **iParam1** ||| INTEGER8 — The first integer parameter to the discretization method.

FIELD **method** ||| UNSIGNED4 — Indicator of the method to use (see c_method).

FIELD **rParam1** ||| REAL8 — The first real parameter.

FIELD **fields** ||| SET (UNSIGNED4) — No Doc

FUNCTION i_ByRounding

Discretize \

	i_ByRounding
`(SET OF Types.t_FieldNumber f, REAL Scale=1.0,REAL Delta=0.0)`	

Construct an instruction (rMethod) that will cause certain fields to be discretized by rounding. See ByRounding below.

PARAMETER **Scale** ||| REAL8 — (Optional) A number by which to multiply each field before rounding.

PARAMETER **f** ||| SET (UNSIGNED4) — A set of field numbers to which to apply this method.

PARAMETER **Delta** ||| REAL8 — (Optional) An offset that is applied after scaling but before rounding.

RETURN **TABLE (r_Method)** — DATASET(r_Method) containing one record.

FUNCTION ByRounding

Discretize \

	ByRounding
(DATASET(Types.NumericField) d,REAL Scale=1.0, REAL Delta=0.0)	

Round the values passed in to create a discrete element Scale is applied (by multiplication) first and can be used to bring the data into a desired range (rParam1), Delta is applied (by addition) second and can be used to re-base a range OR to cause truncation or roundup as required (rParam2).

PARAMETER **d** ||| TABLE (NumericField) — The NumericField dataset to be discretized.

PARAMETER **Scale** ||| REAL8 — (Optional) A number by which to multiply each field before rounding.

PARAMETER **Delta** ||| REAL8 — (Optional) An offset that is applied after scaling but before rounding.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , INTEGER4 value })** — DATASET(DiscreteField) containing the discretized dataset.

FUNCTION i_ByBucketing

Discretize \

	i_ByBucketing
(SET OF Types.t_FieldNumber f, Types.t_Discrete N=ML_Core.Config.Discrete)	

Construct an instruction (rMethod) that will cause certain fields to be discretized by bucketing. See ByBucketing below.

PARAMETER **N** ||| INTEGER4 — (Optional) The number of buckets into which to split the range. The default is to use the ML_Core. Config.Discrete configuration parameter.

PARAMETER **f** ||| SET (UNSIGNED4) — A set of field numbers to which to apply this method.

RETURN **TABLE (r_Method)** — DATASET(r_Method) containing one record.

FUNCTION ByBucketing

Discretize \

	ByBucketing
	`(DATASET(Types.NumericField) d, Types.t_Discrete N=ML_Core.Config.Discrete)`

Allocates a continuous variable into one of N buckets based upon an equal division of the RANGE of the variable.

The buckets will NOT have an even number of elements unless the underlying distribution of the variable is uniform.

PARAMETER **d** ||| TABLE (NumericField) — The NumericField dataset to be discretized.

PARAMETER **N** ||| INTEGER4 — (Optional) The number of buckets into which to split the range. The default is to use the ML_Core. Config.Discrete configuration parameter.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , INTEGER4 value })** — DATASET(DiscreteField) containing the discretized dataset.

FUNCTION i_ByTiling

Discretize \

	i_ByTiling
	`(SET OF Types.t_FieldNumber f, Types.t_Discrete N=ML_Core.Config.Discrete)`

Construct an instruction (rMethod) that will cause certain fields to be discretized by tiling. See ByTiling below.

PARAMETER **N** ||| INTEGER4 — (Optional) The number of tiles into which to split the data. The default is to use the ML_Core. Config.Discrete configuration parameter.

PARAMETER **f** ||| SET (UNSIGNED4) — A set of field numbers to which to apply this method.

RETURN **TABLE (r_Method)** — DATASET(r_Method) containing one record.

FUNCTION ByTiling

Discretize \

	ByTiling
	`(DATASET(Types.NumericField) d, Types.t_Discrete N=ML_Core.Config.Discrete)`

Allocate a continuous variable into one of N groups such that each group (tile) contains roughly the same number of entries and that all of the elements of group 2 have a higher value than group 1, etc.

PARAMETER **d** ||| TABLE (NumericField) — The NumericField dataset to be discretized.

PARAMETER **N** ||| INTEGER4 — (Optional) The number of tiles to create. The default is to use the ML_Core. Config.Discrete configuration parameter.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , INTEGER4 value })** — DATASET(DiscreteField) containing the discretized dataset.

FUNCTION Do

Discretize \

	Do
	`(DATASET(Types.NumericField) d, DATASET(r_Method) to_do)`

Execute a set of discretization instructions in order to discretize all of the fields of the dataset using the appropriate methods.

Note that the file d is read once for each instruction - so it is much better to combine the instructions for multiple fields into one (provided the parameters and method are the same).

PARAMETER **d** ||| TABLE (NumericField) — The NumericField dataset to be dicretized.

PARAMETER **to_do** ||| TABLE (r_Method) — The DATASET(r_Method) that contains the discretization instructions.

RETURN **TABLE (DiscreteField)** — DATASET(DiscreteField) containing the discretized dataset.

FieldAggregates

Go Up

IMPORTS

Types | Utils | std.system.ThorLib |

DESCRIPTIONS

MODULE FieldAggregates

	FieldAggregates
(DATASET(Types.NumericField) d)	

Calculate various statistical aggregations of the fields in a NumericField dataset.

PARAMETER **d** ||| TABLE (NumericField) — The dataset to be aggregated.

Children

1. Simple : Calculate basic statistics about each field
2. SimpleRanked : Calculate the rank (order) of each cell for each field
3. Medians : Calculate the median value of each field
4. MinMedNext : No Documentation Found
5. Buckets : Bucketize the datapoints into N buckets for each field
6. BucketRanges : Return the ranges associated with each of N buckets as computed by 'Buckets' above

7. Modes : Calculate the mode (i.e
8. Cardinality : Returns the cardinality of each field
9. RankedInput : No Documentation Found
10. NTiles : Calculate the N-tile of each datapoint within its field
11. NTileRanges : Return the ranges associated with each of N-tiles as computed by 'Ntiles' above

---

ATTRIBUTE Simple

FieldAggregates \

	Simple

Calculate basic statistics about each field.

Calculates: min, max, sum, count, mean, variance, and standard deviation for each field.

There are no parameters.

Example:

```
myAggs := FieldAggregates(myDS).simple;
```

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

ATTRIBUTE SimpleRanked

FieldAggregates \

	SimpleRanked

Calculate the rank (order) of each cell for each field.

The returned data adds a 'Pos' field to each cell, indicating its rank within it's field number.

There are no parameters.

Example:

```
myRankedDS := FieldAggregates(myDS).SimpleRanked;
```

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

ATTRIBUTE Medians

FieldAggregates \

	Medians

Calculate the median value of each field.

There are no parameters.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 number , REAL8 median }) —** DATASET({wi, number, median}), one record per work-item and field number. Example: <pre>myFieldMedians := FieldAggregates(myDS).Medians;</pre>

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)

Bucketize the datapoints into N buckets for each field.

Bucketization splits the range of the data into N equal size range buckets. The data will not normally be evenly split among buckets unless it is uniformly distributed. Contrast this with N-tile, where the data is split nearly evenly.

PARAMETER **n** ||| INTEGER4 — The number of buckets to use.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 value , UNSIGNED8 Pos , INTEGER4 bucket })** — DATASET OF {wi, id, number, value, pos, bucket}, where pos is the rank within each field, and bucket is the bucket number.

FUNCTION BucketRanges

FieldAggregates \

	BucketRanges
	(Types.t_Discrete n)

Return the ranges associated with each of N buckets as computed by 'Buckets' above.

PARAMETER **n** ||| INTEGER4 — The number of buckets.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 number , INTEGER4 bucket , REAL8 Min , REAL8 Max , UNSIGNED8 cnt })** — DATASET OF {wi, number, bucket, Min, and Max}, one for each bucket for each field.

ATTRIBUTE Modes

FieldAggregates \

	Modes

Calculate the mode (i.e. the most common value) for each field

There are no parameters.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 number , REAL8 mode , UNSIGNED8 cnt })** — DATASET OF {wi, number, mode, cnt}, one per field. 'cnt' is the number of times the mode value occurred.

ATTRIBUTE Cardinality

FieldAggregates \

	Cardinality

Returns the cardinality of each field. That is the number of different values occurring in each field.

There are no parameters.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 number , UNSIGNED8 cardinality })** — DATASET OF {wi, number, cardinality}, one per field.

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)

Calculate the N-tile of each datapoint within its field. For example, if N is 100, we calculate percentiles.

PARAMETER **n** ||| INTEGER4 — The number of groups into which to balance the data

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 value , REAL8 Pos , INTEGER4 ntile })** — DATASET OF {wi, id, number, value, pos, ntile}, where pos is the rank within each field.

FUNCTION NTileRanges

FieldAggregates \

	NTileRanges
	(Types.t_Discrete n)

Return the ranges associated with each of N-tiles as computed by 'Ntiles' above.

PARAMETER **n** ||| INTEGER4 — The number of N-tile groups.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 number , INTEGER4 ntile , REAL8 Min , REAL8 Max , UNSIGNED8 cnt })** — DATASET OF {wi, number, bucket, Min, and Max}, one for each N-tile group for each field.

FromField

Go Up

DESCRIPTIONS

MACRO FromField

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

Macro to convert a NumericField formatted, cell-based dataset to a Record formatted dataset. Typically used to return converted NumericField data back to its original layout.

Note that as a Macro, nothing is returned, but new attributes are created in-line for use in subsequent definitions.

In the simplest case, the assumption is that the field order of the resulting table is in line with the field number in the input dataset, with the ID field as the first field.

For example:

```
myRec := RECORD
  UNSIGNED recordId;
  REAL height;
  REAL weight;
END;
Value of NumericField records with field number = 1 would go to height.
Value of NumericField records with field number = 2 would go to weight.
The id field of the NumericField record would be mapped to the recordId
field of the result.
```

If the field orders have been changed (e.g. by customizing the ToField process, a field-mapping should be specified (See dMap below). Usage Examples:

```
ML.FromField(myNFData, myRecordLayout, myRecordData);
// Datamap to reorder the weight and height fields in the example above
dataMap := DATASET([\{'weight', '1'\},
                    \{'height', '2'\}], Types.Field\_Mapping);
ML.FromField(nyNFData, myRecordLayout, myRecordData, dataMap);
```

PARAMETER **dIn** ||| INTEGER8 — The name of the input dataset in NumericField format.

PARAMETER **lOut** ||| INTEGER8 — The name of the layout record defining the records of the result dataset.

PARAMETER **dOut** ||| INTEGER8 — The name of the result dataset.

PARAMETER **dMap** ||| INTEGER8 — [OPTIONAL] A Field_Mapping dataset as produced by ToField that describes the mapping between field name and field number. The format of this map is defined by Types.Field_Mapping.

RETURN — Nothing. The MACRO creates new attributes in-line as described above.

SEE Types.NumericField

SEE Types.Field_Mapping

SEE ToField

Generate

Go Up

IMPORTS

Types |

DESCRIPTIONS

MODULE Generate

	Generate

Increase dimensionality by adding polynomial transforms of the data to create new feature columns. This can be useful, for example, when building a linear model against data that may not have linear relationships.

Children

1. tp_Method : Enumeration of polynomial methods
2. MethodName : Convert a column number into a descriptive label
3. ToPoly : Generate up to seven, successively higher order, features from a single given feature

---

ATTRIBUTE tp_Method

Generate \

	tp_Method

Enumeration of polynomial methods.

RETURN **UNSIGNED1** —

VALUE LogX = 1

VALUE X = 2

VALUE XLogX = 3

VALUE XX = 4 – X squared

VALUE XXLogX = 5

VALUE XXX = 6 – X cubed

VALUE XXXLogX = 7

FUNCTION MethodName

Generate \

	MethodName
`(tp_Method x)`	

Convert a column number into a descriptive label.

PARAMETER **x** ||| UNSIGNED1 — The column number to describe.

RETURN **STRING7** — The descriptive label.

FUNCTION ToPoly

Generate \

	ToPoly
	`(DATASET(Types.NumericField) seedCol, UNSIGNED maxN=7)`

Generate up to seven, successively higher order, features from a single given feature.

The generated features are:

1. LogX (logs are base 10)
2. X
3. XLogX
4. X^2
5. X^2LogX
6. X^3
7. X^3LogX

Note that the returned fields will be numbered 1-7, as above.

PARAMETER **seedCol** ||| TABLE (NumericField) — A single column of NumericField data. The number field is ignored.

PARAMETER **maxN** ||| UNSIGNED8 — (Optional) The number of new columns to generate. For example: If 1, then one feature, LogX is generated. If 3, then LogX, X, and X^2 features are generated. The default is 7, in which case, all features are generated.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 value })** — DATASET(NumericField) with numOriginalRecs * maxN records.

SEE Types.NumericField

ModelOps2

Go Up

IMPORTS

Types |

DESCRIPTIONS

MODULE ModelOps2

	ModelOps2

This module provides a set of operations to provide manipulation of machine learning models (version 2) in the Types.Layout_Model2 format.

Layout_Model2 defines a flexible structure that allows storage of model information for any Machine Learning algorithm.

The model is based on a ”Naming Tree” paradigm.

The naming tree is a data structure that allows a hierarchical name (e.g. object-id) to be attached to each data-cell. Examples of naming-trees are OID trees such as those used in various network identifiers such as MIBs.

This structure is used within ML to store model information. It is a useful format for several reasons:

- It has the flexibility to store complex sets of data in a generic way.
- It easily stores scalar as well as matrix oriented data.
- It allows a model to contain data elements within scopes that are defined at different level. For example, part of the model may be defined globally, another may be common for a bundle, while another section is specific to a given module.

- It readily allows composite models to be created by encapsulating entire complex models (or sets of models) within branches of another model. The individual models can then be extracted from the composite model, and passed to the modules that created them.

Theory of Operation

The naming tree (NT) is conceptually simple. Each cell is identified by a hierarchical numbering scheme of arbitrary depth. Take, for example, the following NT:

```
1
  1.1
    1.1.1
    1.1.2
  1.2
    1.2.1
    1.2.2
2
```

This tree defines the following leaf (scalar) elements: 1.1.1, 1.1.2, 1.2.1, 1.2.2, 2.

Note that the deepest node on any branch is considered a leaf, and branches can be of variable depth. Note also that there is no explicit creation of branch nodes. The branches are implicitly defined by the ids of the leafs.

In this example, node 1.1 can be thought as representing an array, thought it could also be thought of as a structure of two distinct scalars, depending on whether the user expects a variable length list under 1.1 (i.e. 1.1.1 - 1.1.N) or a fixed set of cells.

Likewise node 1 can be thought of as a matrix (1.r.c, where r is the row index and c is the column index), in cases where r and c are of variable size.

This naming tree also supports the myriad interface, allowing multiple independent work-items to be represented, each of which may duplicate the same structure.

The id is represented by an ECL SET of Unsigned identifiers (e.g. [1,2,1] represents the OID 1.2.1).

Each cell is defined by three fields: wi (work-item-id), value (the cell contents) and indexes (the id).

A naming tree can be constructed as an inline dataset. For example, the following creates the tree in the example above:

```
DATASET([\{1, 3.2, [1,1,1]\},
         \{1, .0297, [1,1,2]\},
         \{1, 2.0, [1,2,1]\},
         \{1, 1550, [1,2,2]\},
```

```
\{1, 8.1, [2]\}], Layout\_Model2);
```

There are attributes in this module to assist with manipulation of naming trees:

- Creating a NT from a NumericField matrix.
- Extracting a NumericField matrix from an NT branch.
- Inserting an NT onto a branch of another NT.
- Extracting an NT from a branch of an NT.

SEE Types.Layout_Model2

Children

1. Extract : Extract an inner sub-tree from an existing model
2. ExtendIndices : Extend the indices of a model to fit within a deeper model
3. Insert : Insert a model into a sub-tree of an existing model
4. ToNumericField : Convert a two-level model or model sub-tree into a NumericField dataset
5. FromNumericField : Convert a NumericField dataset to a 2 level model (or model subtree)
6. GetItem : Get a single record (cell) from a model by index
7. SetItem : Add a single record (cell) to an model at a given set of coordinates

---

FUNCTION Extract

ModelOps2 \

DATASET(Layout_Model2)	**Extract**
(DATASET(Layout_Model2) mod, t_indexes fromIndx, t_work_item fromWi=0)	

Extract an inner sub-tree from an existing model.

Work-item = 0 (default) will extract all work-items

This is the opposite of Insert. For example:

```
If I have a tree:
 1
 2
 3
  3.1
  3.2
```

and I extract from index 3, it will return the Naming Tree:

```
 1
 2
```

containing the two sub-cells of the original index 3

PARAMETER **fromWi** ||| UNSIGNED2 — The work-item to extract or 0 to extract the same sub-tree from all work-items.

PARAMETER **fromIndx** ||| SET (UNSIGNED4) — The index from which to extract the subtree.

PARAMETER **mod** ||| TABLE (Layout_Model2) — The model from which to extract the sub-tree.

RETURN **TABLE ({ UNSIGNED2 wi , REAL8 value , SET (UNSIGNED4) indexes })** — A model containing all of the sub-cells below fromIndx with the indexes adjusted to the top of the tree.

FUNCTION ExtendIndices

ModelOps2 \

DATASET(Layout_Model2)	**ExtendIndices**
(DATASET(Layout_Model2) mod, t_indexes atIndex)	

Extend the indices of a model to fit within a deeper model.

For example, a cell with index [1,2] could be moved to index [1,2,3,1,2] by using atIndex := [1,2,3].

PARAMETER **atIndex** ||| SET (UNSIGNED4) — The prefix indexes to be prepended to the indexes of each cell in mod.

PARAMETER **mod** ||| TABLE (Layout_Model2) — The model whose indexes are to be extended.

RETURN **TABLE ({ UNSIGNED2 wi , REAL8 value , SET (UNSIGNED4) indexes })** — A model with extended indexes.

FUNCTION Insert

ModelOps2 \

DATASET(Layout_Model2)	**Insert**
(DATASET(Layout_Model2) mod1, DATASET(Layout_Model2) mod2, t_indexes atIndx)	

Insert a model into a sub-tree of an existing model.

Extends the indexes of the provided model to fit onto a branch of another model, and concatenates the two models. This is the opposite of extract. For example:

```
If I have a model:
1
2
and a second model:
1
2
3
That I would like to insert into the first tree at index 3, I would
end up with the tree:
1
2
3
 3.1
 3.2
 3.3
```

Example code:

```
mod3 := Insert(mod1, mod2, [3]);
```

PARAMETER **atIndx** ||| SET (UNSIGNED4) — The index prefix (in mod1) that will contain the cells from mod2.

PARAMETER **mod2** ||| TABLE (Layout_Model2) — The sub-model that is to be inserted into mod1.

PARAMETER **mod1** ||| TABLE (Layout_Model2) — The first (base) model.

RETURN **TABLE (Layout_Model2)** — a new model containing the cells from both models.

FUNCTION ToNumericField

ModelOps2 \

DATASET(NumericField)	**ToNumericField**
(DATASET(Layout_Model2) mod, t_indexes fromIndx = [])	

Convert a two-level model or model sub-tree into a NumericField dataset.

The last two indexes of the model subtree are used as the indexes for the NumericField matrix. The second to last index corresponds to the NF's id field and the last index corresponds to the NF's number field.

PARAMETER **fromIndx** ||| SET (UNSIGNED4) — The index from which to extract the matrix. Example: [3,1,5]. The default is from the top of the tree i.e. [].

PARAMETER **mod** ||| TABLE (Layout_Model2) — The model from which to extract the NumericField matrix.

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

FUNCTION FromNumericField

ModelOps2 \

DATASET(Layout_Model2)	**FromNumericField**
(DATASET(NumericField) nf, t_indexes atIndex=[])	

Convert a NumericField dataset to a 2 level model (or model subtree).

A two level model is created and appended to atIndex.

The first new index will contain the value of the NumericField's id field, and the second will contain the value of the NumericField's number field.

Example: If I have a NumericField with id=1 and number=3, and I use atIndex = [3,1,5], it will create a Naming Tree cell with indexes: [3,1,5,1,3].

PARAMETER **atIndex** ||| SET (UNSIGNED4) — The index at which to place the new subtree e.g., [3,1,5].

PARAMETER **nf** ||| TABLE (NumericField) — A NumericField dataset to be converted.

RETURN **TABLE ({ UNSIGNED2 wi , REAL8 value , SET (UNSIGNED4) indexes })** — DATASET(ntNumeric) Naming Tree.

FUNCTION GetItem

ModelOps2 \

Layout_Model2	**GetItem**
(DATASET(Layout_Model2) mod, t_indexes indxs, wi_num=1)	

Get a single record (cell) from a model by index.

PARAMETER **indxs** ||| SET (UNSIGNED4) — The id of the cell to extract (e.g. [3,1,5]).

PARAMETER **wi_num** ||| INTEGER8 — The work-item number to extract the cell from, default = 1.

PARAMETER **mod** ||| TABLE (Layout_Model2) — The model (DATASET(layout_model2)) from which to extract the cell.

RETURN **ROW (Layout_Model2)** — The model cell (Layout_Model2) or an empty cell (wi=0) if not found.

FUNCTION SetItem

ModelOps2 \

DATASET(Layout_Model2)	**SetItem**
(DATASET(Layout_Model2) mod, t_work_item wi, t_indexes indexes, t_fieldReal value)	

Add a single record (cell) to an model at a given set of coordinates.

PARAMETER **value** ||| REAL8 — The value of the cell.

PARAMETER **wi** ||| UNSIGNED2 — The work-item associated with the cell.

PARAMETER **indexes** ||| SET (UNSIGNED4) — The indices for the cell.

PARAMETER **mod** ||| TABLE (Layout_Model2) — The model to which to add a cell.

RETURN **TABLE (Layout_Model2)** — Model with the added cell.

ToField

Go Up

DESCRIPTIONS

MACRO ToField

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

Convert a record-oriented dataset to a cell-oriented NumericField dataset for use with Machine Learning mechanisms.

ToField Macro takes a record-oriented dataset, with each row containing an ID and one or more numeric fields, and expands it into the NumericField format used by ML.

Note that as a Macro, nothing is returned, but new attributes are created in-line for use in subsequent definitions.

Along with creating the NumericField table, this macro produces two simple functions to assist the user in mapping the field names to their corresponding numbers. These are "STRING dOut_ToName(UNSIGNED)" and "UNSIGNED dOut_ToNumber(STRING)", where the "dOut" portion of the function name is the name passed into that parameter of the macro.

The macro also produces a mapping table named "dOut_Map", again where "dOut" refers to the parameter, that contains a table of the field mappings. See Types.Field_Mapping for the layout of this mapping dataset. Examples:

```
ML.ToField(dOrig,dMatrix);
ML.ToField(dOrig,dMatrix,myid,'field5,field7,field10');
dMatrix\_ToName(2);    // returns 'field7'
dMatrix\_ToNumber('field10'); // returns 3
dMatrix\_Map;  // returns the mapping table of field name to number see
              // Types.Field\_Mapping
```

PARAMETER **dIn** ||| INTEGER8 — The name of the input dataset.

PARAMETER **dOut** ||| INTEGER8 — The name of the resulting dataset.

PARAMETER **wifield** ||| INTEGER8 — [OPTIONAL] The name of the field that contains the work item value. A constant is used if the field name is not supplied (as provided by wivalue below).

PARAMETER **datafields** ||| INTEGER8 — [OPTIONAL] A STRING containing a comma-delimited list of the fields to be treated as axes. If omitted, all numeric fields that are not the idfield or wifield will be treated as axes. NOTE: idfield defaults to the first field in the table, so if that field is specified as an axis field, then the user should be sure to specify a value in the idfield param.

PARAMETER **idfield** ||| INTEGER8 — [OPTIONAL] The name of the field that contains the UID for each row. If omitted, it is assumed to be the first field.

PARAMETER **wivalue** ||| INTEGER8 — [OPTIONAL} The constant value to use for work item. The value 1 is used if not supplied.

RETURN — Nothing. The MACRO creates new attributes in-line as described above.

SEE Types.NumericField

SEE Types.Field_Mapping

Types

Go Up

DESCRIPTIONS

MODULE Types

	Types

This module provides the major data type definitions for use with the various ML Bundles

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. t_index : No Documentation Found
10. t_indexes : No Documentation Found
11. AnyField : No Documentation Found
12. NumericField : The NumericField layout defines a matrix of Real valued data-points
13. DiscreteField : The Discrete Field layout defines a matrix of Integer valued data-points
14. Layout_Model2 : Layout for Model dataset (version 2) Generic Layout describing the model 'learned' by a Machine Learning algorithm

15. Layout_Model : No Documentation Found
16. Classify_Result : No Documentation Found
17. l_result : No Documentation Found
18. Class_Stats : Class_Stats
19. Confusion_Detail : Confusion_Detail
20. Classification_Accuracy : Classification_Accuracy
21. Class_Accuracy : Class_Accuracy Results layout for Analysis.Classification.AccuracyByClass See https://en.wikipedia.org/wiki/Precision_and_recall for a more detailed explanation
22. Regression_Accuracy : Regression_Accuracy
23. Data_Diagnostic : No Documentation Found
24. Field_Mapping : Field_Mapping is the format produced by ToField for field-name mapping
25. LUCI_Rec : LUCI Record – A dataset of lines each containing a string This is the DATASET format in which ML algorithm export LUCI files

---

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** —

ATTRIBUTE t_index

Types \

	t_index

No Documentation Found

RETURN **UNSIGNED4** —

ATTRIBUTE t_indexes

Types \

	t_indexes

No Documentation Found

RETURN **SET (UNSIGNED4)** —

RECORD AnyField

Types \

	AnyField

No Documentation Found

FIELD **id** ||| UNSIGNED8 — No Doc

FIELD **number** ||| UNSIGNED4 — No Doc

FIELD **wi** ||| UNSIGNED2 — No Doc

RECORD NumericField

Types \

	NumericField

The NumericField layout defines a matrix of Real valued data-points. It acts as the primary Dataset layout for interacting with most ML Functions. Each record represents a single cell in a matrix. It is most often used to represent a set of data-samples or observations, with the 'id' field representing the data-sample or observation, and the 'number' field representing the various fields within the observation.

FIELD **id** ||| UNSIGNED8 — This field represents the row-number of this cell of the matrix. It is also considered the record-id for observations / data-samples.

FIELD **number** ||| UNSIGNED4 — This field represents the matrix column number for this cell. It is also considered the field number of the observation

FIELD **value** ||| REAL8 — The value of this cell in the matrix.

FIELD **wi** ||| UNSIGNED2 — The work-item id, supporting the Myriad style interface. This allows multiple independent matrixes to be contained within a single dataset, supporting independent ML activities to be processed in parallel.

RECORD DiscreteField

Types \

	DiscreteField

The Discrete Field layout defines a matrix of Integer valued data-points. It is similar to the NumericField layout above, except for only containing discrete (integer) values. It is typically used to convey the class-labels for classification algorithms.

FIELD **id** ||| UNSIGNED8 — This field represents the row-number of this cell of the matrix. It is also considered the record-id for observations / data-samples.

FIELD **number** ||| UNSIGNED4 — This field represents the matrix column number for this cell. It is also considered the field number of the observation

FIELD **value** ||| INTEGER4 — The value of this cell in the matrix.

FIELD **wi** ||| UNSIGNED2 — The work-item id, supporting the Myriad style interface. This allows multiple independent matrixes to be contained within a single dataset, supporting independent ML activities to be processed in parallel.

RECORD Layout_Model2

Types \

	Layout_Model2

Layout for Model dataset (version 2) Generic Layout describing the model 'learned' by a Machine Learning algorithm. Models for all new ML bundles are stored in this format. Some older bundles may still use the Layout_Model (version 1) layout. Models are thought of as opaque data structures. They are not designed to be understandable except to the bundle that produced them. Most bundles contain mechanisms to extract useful information from the model. This version of the model is based on a Naming-Tree paradigm. This provides a flexible generic mechanism for storage and manipulation of models. For bundle developers (or the curious), the file modelOps2 provides a detailed description of the theory and usage of this model layout as well as a set of functions to manipulate models for use by bundle developers.

FIELD **indexes** ||| SET (UNSIGNED4) — The identifier for the cell – a set of unsigned integers e.g., [1,2,1,3]

FIELD **value** ||| REAL8 — The value of the cell

FIELD **wi** ||| UNSIGNED2 — The work-item-id

RECORD Layout_Model

Types \

	Layout_Model

No Documentation Found

FIELD **id** ||| UNSIGNED8 — No Doc

FIELD **number** ||| UNSIGNED4 — No Doc

FIELD **value** ||| REAL8 — No Doc

FIELD **wi** ||| UNSIGNED2 — No Doc

RECORD Classify_Result

Types \

	Classify_Result

No Documentation Found

FIELD **conf** ||| REAL8 — No Doc

FIELD **id** ||| UNSIGNED8 — No Doc

FIELD **number** ||| UNSIGNED4 — No Doc

FIELD **value** ||| INTEGER4 — No Doc

FIELD **wi** ||| UNSIGNED2 — No Doc

RECORD l_result

Types \

	l_result

No Documentation Found

FIELD **conf** ||| REAL8 — No Doc

FIELD **id** ||| UNSIGNED8 — No Doc

FIELD **number** ||| UNSIGNED4 — No Doc

FIELD **value** ||| INTEGER4 — No Doc

FIELD **wi** ||| UNSIGNED2 — No Doc

RECORD Class_Stats

Types \

	Class_Stats

Class_Stats Layout for data returned from Analysis.Regression.ClassStats

FIELD **classCount** ||| INTEGER4 — The number of times the class was seen in the data

FIELD **class** ||| INTEGER4 — The class label associated with this record

FIELD **classifier** ||| UNSIGNED4 — The field number associated with this dependent variable, for multi-variate classification. Otherwise 1.

FIELD **classPct** ||| REAL8 — The percent of records with this class.

FIELD **wi** ||| UNSIGNED2 — Work-item identifier

RECORD Confusion_Detail

Types \

	Confusion_Detail

Confusion_Detail Layout for storage of the confusion matrix for ML Classifiers Each row represents a pairing of a predicted class and an actual class

FIELD **pctPred** ||| REAL8 — Indicates the percent of items that were predicted as <predict_class> that were actually of <actual_class>.</actual_class></predict_class>

FIELD **predict_class** ||| INTEGER4 — The class number predicted by the ML algorithm

FIELD **occurs** ||| UNSIGNED4 — The number of times this pairing of (actual / predicted) classes occurred

FIELD **correct** ||| BOOLEAN — Boolean indicating if this represents a correct prediction (i.e. predicted = actual)

FIELD **classifier** ||| UNSIGNED4 — The field number associated with this dependent variable, for multi-variate. Otherwise 1.

FIELD **actual_class** ||| INTEGER4 — The target class number – the expected result.

FIELD **pctActual** ||| REAL8 — The percent of items that were actually of <actual_class> that were predicted as <predict_class>.</predict_class></actual_class>

FIELD **wi** ||| UNSIGNED2 — Work item identifier

RECORD Classification_Accuracy

Types \

	Classification_Accuracy

Classification_Accuracy Results layout for Analysis.Classification/Accuracy

FIELD **PoD** ||| REAL8 — Power of Discrimination. Indicates how this classification performed relative to a random guess of class. Zero or negative indicates that the classification was no better than a random guess. 1.0 indicates a perfect classification. For example if there are two equi-probable classes, then a random guess would be right about 50% of the time. If this classification had a Raw Accuracy of 75%, then its PoD would be .5 (half way between a random guess and perfection).

FIELD **classifier** ||| UNSIGNED4 — The field number associated with this dependent variable, for multi-variate. Otherwise 1.

FIELD **PoDE** ||| REAL8 — Power of Discrimination Extended. Indicates how this classification performed relative to guessing the most frequent class (i.e. the trivial solution). Zero or negative indicates that this classification is no better than the trivial solution. 1.0 indicates perfect classification. For example, if 95% of the samples were of class 1, then the trivial solution would be right 95% of the time. If this classification had a raw accuracy of 97.5%, its PoDE would be .5 (i.e. half way between trivial solution and perfection).

FIELD **recCnt** ||| UNSIGNED8 — The total number or records in the test set

FIELD **wi** ||| UNSIGNED2 — Work item identifier

FIELD **errCnt** ||| UNSIGNED8 — The number of errors (i.e. predicted <> actual)

FIELD **Raw_Accuracy** ||| REAL8 — The percentage of samples properly classified (0.0 - 1.0)

RECORD Class_Accuracy

Types \

	Class_Accuracy

Class_Accuracy Results layout for Analysis.Classification.AccuracyByClass See https://en.wikipedia.org/wiki/Precision_and_recall for a more detailed explanation.

FIELD **FPR** ||| REAL8 — The false positive rate for this class (i.e. False Positives / (False Positives + True Negatives)) What percentage of the items not in this class did we falsely predict as this class?

FIELD **class** ||| INTEGER4 — The class to which the analytics apply

FIELD **classifier** ||| UNSIGNED4 — The field number associated with this dependent variable, for multi-variate. Otherwise 1.

FIELD **recall** ||| REAL8 — The completeness of recall for this class (i.e. True Positives / (True Positives + False Negatives)) What percentage of the items that are actually in this class did we correctly predict as this class?

FIELD **precision** ||| REAL8 — The precision of the classification for this class (i.e. True Positives / (True Positives + FalsePositives)). What percentage of the items that we predicted as being in this class are actually of this class?

FIELD **wi** ||| UNSIGNED2 — Work item identifier

RECORD Regression_Accuracy

Types \

	Regression_Accuracy

Regression_Accuracy Results layout for Analysis.Regression.Accuracy

FIELD **RMSE** ||| REAL8 — Root Mean Squared Error = MSE^.5 (Square root of MSE)

FIELD **R2** ||| REAL8 — The R-Squared value (Coefficient of Determination) for the regression. R-squared of zero or negative indicates that the regression has no predictive value. R2 of 1 would indicate a perfect regression.

FIELD **MSE** ||| REAL8 — Mean Squared Error = SUM((predicted - actual)^2) / N (number of datapoints)

FIELD **regressor** ||| UNSIGNED4 — The field number associated with this dependent variable, for multi-variate. Otherwise 1.

FIELD **wi** ||| UNSIGNED2 — Work item identifier

RECORD Data_Diagnostic

Types \

	Data_Diagnostic

No Documentation Found

FIELD **valid** ||| BOOLEAN — No Doc

FIELD **message_text** ||| SET (VARSTRING) — No Doc

FIELD **wi** ||| UNSIGNED2 — No Doc

RECORD Field_Mapping

Types \

	Field_Mapping

Field_Mapping is the format produced by ToField for field-name mapping.

FIELD **assigned_name** ||| STRING — The integer field number used in the ML algorithm stored as a STRING

FIELD **orig_name** ||| STRING — The name of the field in the original layout

---

RECORD LUCI_Rec

Types \

	LUCI_Rec

LUCI Record – A dataset of lines each containing a string This is the DATASET format in which ML algorithm export LUCI files.

FIELD **line** ||| STRING — A single line in the LUCI csv file

---

Interfaces

Go Up

Table of Contents

Interfaces/

IClassify

Go Up

IMPORTS

Types |

DESCRIPTIONS

MODULE IClassify

	IClassify

DEPRECATED Interface Definition for Classification Modules (version 1). This interface is being deprecated and should not be used for new bundles or bundles undergoing substantial revision. Please use IClassify2 going forward. 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 **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

---

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 **new_observations** ||| TABLE (NumericField) — observations to be classified.

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

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

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

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 classifier , INTEGER4 actual_class , INTEGER4 predict_class , UNSIGNED4 occurs , BOOLEAN correct , REAL8 pctActual , REAL8 pctPred })** — the confusion matrix showing correct and incorrect results.

Interfaces/

IClassify2

Go Up

IMPORTS

Types |

DESCRIPTIONS

MODULE IClassify2

	IClassify2

Interface definition for Classification (Version 2). Classification learns a function that maps a set of input data to one or more output class-label (i.e. Discrete) variables. The resulting learned function is known as the model. That model can then be used repetitively to predict the class(es) for each sample when presented with new input data. Actual implementation modules will probably take configuration parameters to control the classification process. The Classification modules also expose attributes for assessing the effectiveness of the classification.

Children

1. GetModel : Calculate the model to fit the independent data to the observed classes (i.e
2. Classify : Classify the observations using a model
3. Accuracy : Return accuracy metrics for the given set of test data
   This is equivalent to calling Predict followed by Analysis.Classification.Accuracy(...)
4. AccuracyByClass : Return class-level accuracy by class metrics for the given set of test data
5. ConfusionMatrix : Return the confusion matrix for a set of test data

FUNCTION GetModel

IClassify2 \

DATASET(Layout_Model2)	GetModel
(DATASET(NumericField) independents, DATASET(DiscreteField) dependents)	

Calculate the model to fit the independent data to the observed classes (i.e. dependent data).

PARAMETER **dependents** ||| TABLE (DiscreteField) — The observed dependent(class label) values.

PARAMETER **indepenedents** ||| — The observed independent (explanatory) values.

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

RETURN **TABLE ({ UNSIGNED2 wi , REAL8 value , SET (UNSIGNED4) indexes })** — The encoded model.

SEE Types.Layout_Model2

SEE Types.NumericField

SEE Types.DiscreteField

FUNCTION Classify

IClassify2 \

DATASET(DiscreteField)	Classify
(DATASET(Layout_Model2) model, DATASET(NumericField) observations)	

Classify the observations using a model.

PARAMETER **observations** ||| TABLE (NumericField) — New observations (independent data) to be classified.

PARAMETER **model** ||| TABLE (Layout_Model2) — The model, which must be produced by a corresponding getModel function.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , INTEGER4 value })** — Predicted class values.

FUNCTION Accuracy

IClassify2 \

DATASET(Classification_Accuracy)	**Accuracy**
(DATASET(Layout_Model2) model, DATASET(DiscreteField) actuals, DATASET(NumericField) observations)	

Return accuracy metrics for the given set of test data

This is equivalent to calling Predict followed by Analysis.Classification.Accuracy(. . .).

Provides accuracy statistics as follows:

- errCount – The number of misclassified samples.
- errPct – The percentage of samples that were misclasified (0.0 - 1.0).
- RawAccuracy – The percentage of samples properly classified (0.0 - 1.0).
- PoD – Power of Discrimination. Indicates how this classification performed relative to a random guess of class. Zero or negative indicates that the classification was no better than a random guess. 1.0 indicates a perfect classification. For example if there are two equiprobable classes, then a random guess would be right about 50% of the time. If this classification had a Raw Accuracy of 75%, then its PoD would be .5 (half way between a random guess and perfection).
- PoDE – Power of Discrimination Extended. Indicates how this classification performed relative to guessing the most frequent class (i.e. the trivial solution). Zero or negative indicates that this classification is no better than the trivial solution. 1.0 indicates perfect classification. For example, if 95% of the samples were of class 1, then the trivial solution would be right 95% of the time. If this classification had a raw accuracy of 97.5%, its PoDE would be .5 (i.e. half way between trivial solution and perfection).

  Normally, this should be called using data samples that were not included in the training set. In that case, these statistics are considered Out-of-Sample error statistics. If it is called with the X and Y from the training set, it provides In-Sample error statistics, which should never be used to rate the classification model.

PARAMETER **observations** ||| TABLE (NumericField) — The independent (explanatory) values on which to base the test.

PARAMETER **model** ||| TABLE (Layout_Model2) — The encoded model as returned from GetModel.

PARAMETER **actuals** ||| TABLE (DiscreteField) — The actual class values associated with the observations.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 classifier , UNSIGNED8 recCnt , UNSIGNED8 errCnt , REAL8 Raw_Accuracy , REAL8 PoD , REAL8 PoDE })** — DATSET(Classification_Accuracy), one record per work-item.

SEE Types.Classification_Accuracy

FUNCTION AccuracyByClass

IClassify2 \

DATASET(Class_Accuracy)	**AccuracyByClass**
(DATASET(Layout_Model2) model, DATASET(DiscreteField) actuals, DATASET(NumericField) observations)	

Return class-level accuracy by class metrics for the given set of test data.

This is equivalent to calling Predict followed by Analysis.Classification.AccuracyByClass(...).

PARAMETER **observations** ||| TABLE (NumericField) — The independent (explanatory) values on which to base the test

PARAMETER **model** ||| TABLE (Layout_Model2) — The encoded model as returned from GetModel.

PARAMETER **actuals** ||| TABLE (DiscreteField) — The actual class values associated with the observations.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 classifier , INTEGER4 class , REAL8 precision , REAL8 recall , REAL8 FPR })** — DATASET(Class_Accuracy), one record per work-item per class.

SEE Types.Class_Accuracy.

FUNCTION ConfusionMatrix

IClassify2 \

DATASET(Confusion_Detail)	ConfusionMatrix
(DATASET(Layout_Model2) model, DATASET(DiscreteField) actuals, DATASET(NumericField) observations)	

Return the confusion matrix for a set of test data. This is equivalent to calling Predict follwed by Analysis.Classification.ConfusionMatrix(...).

The confusion matrix indicates the number of datapoints that were classified correctly or incorrectly for each class label.

The matrix is provided as a matrix of size numClasses x numClasses with fields as follows:

- 'wi' – The work item id
- 'pred' – the predicted class label (from Classify).
- 'actual' – the actual (target) class label.
- 'samples' – the count of samples that were predicted as 'pred', but should have been 'actual'.
- 'totSamples' – the total number of samples that were predicted as 'pred'.
- 'pctSamples' – the percentage of all samples that were predicted as 'pred', that should have been 'actual' (i.e. samples / totSamples)

This is a useful tool for understanding how the algorithm achieved the overall accuracy. For example: were the common classes mostly correct, while less common classes often misclassified? Which classes were most often confused? This should be called with test data that is independent of the training data in order to understand the out-of-sample (i.e. generalization) performance.

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

PARAMETER **model** ||| TABLE (Layout_Model2) — The encoded model as returned from GetModel.

PARAMETER **actuals** ||| TABLE (DiscreteField) — The actual class values.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 classifier , INTEGER4 actual_class , INTEGER4 predict_class , UNSIGNED4 occurs , BOOLEAN correct , REAL8 pctActual , REAL8 pctPred })** — DATASET(Confusion_Detail), one record per cell of the confusion matrix.

SEE Types.Confusion_Detail.

Interfaces/

IRegression

Go Up

IMPORTS

Types |

DESCRIPTIONS

MODULE IRegression

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

DEPRECATED Interface Definition for Regression Modules (version 1). This interface is being deprecated and should not be used for new bundles or bundles undergoing substantial revision. Please use IRegression2 going forward. 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 **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').

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').

Children

1. GetModel : Calculate and return the 'learned' model
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 **newX** ||| TABLE (NumericField) — DATASET(NumericField) containing the X values to b predicted.

PARAMETER **model** ||| 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.

Interfaces/

IRegression2

Go Up

IMPORTS

Types |

DESCRIPTIONS

MODULE IRegression2

	IRegression2

Interface Definition for Regression Modules (Version 2). Regression learns a function that maps a set of input data to one or more continuous 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. Actual implementation modules will probably take configuration parameters to control the regression process. The regression modules also expose attributes for assessing the effectiveness of the regression.

Children

1. GetModel : Calculate and return the 'learned' model
2. Predict : Predict the output variable(s) based on a previously learned model
3. Accuracy : Assess the accuracy of a set of predictions

FUNCTION GetModel

IRegression2 \

DATASET(Layout_Model2)	GetModel
(DATASET(NumericField) independents, DATASET(NumericField) dependents)	

Calculate and return the 'learned' model.

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

PARAMETER **independents** ||| 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').

PARAMETER **dependents** ||| 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').

RETURN **TABLE ({ UNSIGNED2 wi , REAL8 value , SET (UNSIGNED4) indexes })** — The encoded model.

SEE Types.NumericField

SEE Types.Layout_Model2

FUNCTION Predict

IRegression2 \

DATASET(NumericField)	Predict
(DATASET(Layout_Model2) model, DATASET(NumericField) observations)	

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

PARAMETER **independents** ||| — the observations upon which to predict.

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

PARAMETER **model** ||| TABLE (Layout_Model2) — No Doc

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 value })** — one entry per observation (i.e. id) in observations. This represents the predicted values for the dependent variable(s).

FUNCTION Accuracy

IRegression2 \

DATASET(Regression_Accuracy)	**Accuracy**
(DATASET(Layout_Model2) model, DATASET(NumericField) actuals, DATASET(NumericField) observations)	

Assess the accuracy of a set of predictions. This is equivalent to calling predict and then Analysis.Regression.Accuracy.

PARAMETER **observations** ||| TABLE (NumericField) — The independent data upon which the accuracy assessment is to be based.

PARAMETER **model** ||| TABLE (Layout_Model2) — The model as returned from GetModel

PARAMETER **actuals** ||| TABLE (NumericField) — The actual values of the dependent variable to compare with the predictions.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED4 regressor , REAL8 R2 , REAL8 MSE , REAL8 RMSE })** — Accuracy statistics (see Types.Regression_Accuracy for details)

Math

Go Up

Table of Contents

Beta

Go Up

IMPORTS

Math |

DESCRIPTIONS

FUNCTION Beta

	Beta
(REAL8 x, REAL8 y)	

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

PARAMETER **y** ||| REAL8 — the value of the second number

PARAMETER **x** ||| REAL8 — the value of the first number

RETURN **REAL8** — the beta value

Math/

Distributions

Go Up

IMPORTS

Constants | Math |

DESCRIPTIONS

MODULE Distributions

	Distributions

Compute PDF, CDF, and PPF values for various Probability Distributions.

The Probability Density Function(PDF(x)) of a distribution is the relative likelihood of a sample drawn from that distribution being of value x.

The Cumulative Distribution Function (CDF(x)) of a distribution is the probability of a sample drawn from that distribution to be less than or equal to x.

The Percentage Point Function (PPF(x)) of a distribution is the inverse of the CDF. Given a probability, it returns the value at which the probability of occurrence is less than or equal to the given probability.

Children

1. Normal_CDF : Cumulative Distribution Function (CDF) of the standard normal distribution
2. Normal_PPF : Percentage Point Function (PPF) for the Normal Distribution
3. T_CDF : Cumulative Distribution Function (CDF) for Students t distribution

4. T_PPF : Percentage point function (PPF) for the T distribution
5. Chi2_CDF : The Cumulative Distribution Function (CDF) for the Chi Square distribution for the specified degrees of freedom
6. Chi2_PPF : Probability Point Function (PPF) for the Chi Squared distribution

---

FUNCTION Normal_CDF

Distributions \

REAL8	**Normal_CDF**
(REAL8 x)	

Cumulative Distribution Function (CDF) of the standard normal distribution. The probability that a normal random variable will be smaller than or equal to 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** — probability of exceeding x.

FUNCTION Normal_PPF

Distributions \

REAL8	**Normal_PPF**
(REAL8 x)	

Percentage Point Function (PPF) for the Normal Distribution.

Translated from C/C++ Mathematical Algorithms for Scientists and Engineers, N. Shammas, McGraw-Hill, 1995.

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

RETURN **REAL8** — number of standard deviations from the mean.

FUNCTION T_CDF

Distributions \

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

Cumulative Distribution Function (CDF) for Students t distribution.

The 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** — the probability that a value will be less than or equal to the specified value.

FUNCTION T_PPF

Distributions \

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

Percentage point function (PPF) for the T distribution.

Translated from NIST SEL DATAPAC Fortran TPPF.f source.

PARAMETER **df** ||| REAL8 — degrees of freedom of the distribution.

PARAMETER **x** ||| REAL8 — the probability.

RETURN **REAL8** — the value with that probability.

FUNCTION Chi2_CDF

Distributions \

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

The Cumulative Distribution Function (CDF) for the Chi Square distribution for the specified degrees of freedom.

Translated from the NIST SEL DATAPAC Fortran subroutine CHSCDF.

PARAMETER **df** ||| REAL8 — the degrees of freedom of the distribution.

PARAMETER **x** ||| REAL8 — the value at which to compute.

RETURN **REAL8** — the cumulative probability.

FUNCTION Chi2_PPF

Distributions \

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

Probability Point Function (PPF) for the Chi Squared distribution.

Translated from the NIST SEL DATAPAC Fortran subroutine CHSPPF.

PARAMETER **df** ||| REAL8 — the degrees of freedom of the distribution.

PARAMETER **x** ||| REAL8 — the probability value.

RETURN **REAL8** — the value with that probability.

Math/

DoubleFac

Go Up

DESCRIPTIONS

EMBED DoubleFac

REAL8	**DoubleFac**
(INTEGER2 i)	

Compute the double factorial. The double factorial is defined for odd n as the product of all the odd numbers up to and including that number.

For even numbers it is the product of the even numbers up to and including that number.

Thus DoubleFac(8) = 8*6*4*2.

IF i < 2, the value 1 is returned.

PARAMETER **i** ||| INTEGER2 — the input value.

RETURN **REAL8** — the numeric result.

Math/

Fac

Go Up

DESCRIPTIONS

EMBED Fac

REAL8	**Fac**
(UNSIGNED2 i)	

Factorial function, (i)(i-1)(i-2)...(2)

PARAMETER **i** ||| UNSIGNED2 — the input value.

RETURN **REAL8** — the factorial i!.

---

gamma

Go Up

DESCRIPTIONS

EMBED gamma

REAL8	**gamma**
(REAL8 x)	

Compute the value of gamma function of real number x.

This is a wrapper for the standard C tgamma function.

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

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

Math/

log_gamma

Go Up

DESCRIPTIONS

EMBED log_gamma

REAL8	**log_gamma**
(REAL8 x)	

Compute the value of the log gamma function of the absolute value of X.

This is wrapper for the standard C lgamma function. Avoids the race condition found on some platforms by taking the absolute value of the input argument.

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

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

Math/

lowerGamma

Go Up

DESCRIPTIONS

EMBED lowerGamma

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

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

PARAMETER **y** ||| REAL8 — the value of the second number.

PARAMETER **x** ||| REAL8 — the value of the first number.

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

Math/

NCK

Go Up

IMPORTS

Math |

DESCRIPTIONS

FUNCTION NCK

REAL8	NCK
(INTEGER2 N, INTEGER2 K)	

N Choose K – finds the number of combinations of K elements out of a possible N.

PARAMETER **N** ||| INTEGER2 — the number of items in the population.

PARAMETER **K** ||| INTEGER2 — the number of items to choose.

RETURN **REAL8** — the number of combinations.

Math/

Poly

Go Up

DESCRIPTIONS

EMBED Poly

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

Evaluate a polynomial from a set of coefficients.

Coeffs 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 for the 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.

Math/

StirlingFormula

Go Up

IMPORTS

Math | Constants |

DESCRIPTIONS

FUNCTION StirlingFormula

	StirlingFormula
(REAL x)	

Stirling's formula.

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

RETURN **REAL8** — evaluation result.

Math/

upperGamma

Go Up

DESCRIPTIONS

EMBED upperGamma

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

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

PARAMETER **y** ||| REAL8 — the value of the second number.

PARAMETER **x** ||| REAL8 — the value of the first number.

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

Tests

Go Up

Table of Contents

Tests/

Check_Dist

Go Up

IMPORTS

Math.Distributions | python |

DESCRIPTIONS

ATTRIBUTE Check_Dist

	Check_Dist

No Documentation Found

RETURN —

Tests/

field_aggregates

Go Up

IMPORTS

Types |

DESCRIPTIONS

ATTRIBUTE field_aggregates

	field_aggregates

No Documentation Found

RETURN —

Tests/

generate

Go Up

IMPORTS

DESCRIPTIONS

ATTRIBUTE generate

	generate

No Documentation Found

RETURN —

Tests/

test_appends

Go Up

IMPORTS

std.system.thorlib |

DESCRIPTIONS

ATTRIBUTE test_appends

	test_appends

No Documentation Found

RETURN —

Tests/

test_discrete

Go Up

IMPORTS

Types |

DESCRIPTIONS

ATTRIBUTE test_discrete

	test_discrete

No Documentation Found

RETURN —

Tests/

to_from

Go Up

IMPORTS

Types |

DESCRIPTIONS

ATTRIBUTE to_from

	to_from

No Documentation Found

RETURN —

Tests/

Validate_Betas

Go Up

IMPORTS

Math | python |

DESCRIPTIONS

ATTRIBUTE Validate_Betas

	Validate_Betas

No Documentation Found

RETURN —

Tests/

Validate_Gammas

Go Up

IMPORTS

Math | python |

DESCRIPTIONS

ATTRIBUTE Validate_Gammas

	Validate_Gammas

No Documentation Found

RETURN —

Utils

Go Up

Table of Contents

Fat

Go Up

IMPORTS

Types |

DESCRIPTIONS

FUNCTION Fat

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

Make a sparse NumericField dataset dense by filling in missing values. All empty cells are set to the designated value.

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

PARAMETER **d0** ||| TABLE (NumericField) — They NumericField dataset to be filled.

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

FatD

Go Up

IMPORTS

Types |

DESCRIPTIONS

FUNCTION FatD

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

Make a sparse DiscreteField dataset dense by filling in missing values. All empty cells are set to the designated value.

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

PARAMETER **d0** ||| TABLE (DiscreteField) — The DiscreteField dataset to be filled.

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

Utils/

Gini

Go Up

DESCRIPTIONS

MACRO Gini

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

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

PARAMETER **target** ||| INTEGER8 — the name of the field used as the target.

PARAMETER **wi_name** ||| INTEGER8 — the name of the work item field, default is "wi".

PARAMETER **infile** ||| INTEGER8 — the input file, any type with a work item field.

PARAMETER **pivot** ||| INTEGER8 — the name of the pivot field.

RETURN **BOOLEAN** — A table by Work Item and Pivot value giving count and Gini impurity value.

Utils/

SequenceInField

Go Up

DESCRIPTIONS

MACRO SequenceInField

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

Assign sequence numbers within groups for a dataset. Given a file (dataset) 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 **seq** ||| INTEGER8 — name of the field to receive the sequence number.

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

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

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