Pannasoft Ingenuity Normalization Guide

Revision 1.00

For Pannasoft Ingenuity Version 1.0.2.x

September 2005

TABLE OF CONTENTS

Pannasoft Ingenuity combines fuzzy logic and neural network technologies to create learning and predicting capabilities. Therefore, data provided to Pannasoft Ingenuity must be a set of numbers (between 0 and 1). Normalization is needed for Pannasoft Ingenuity in order to let fuzzy operation in Pannasoft Ingenuity work properly. Here, the normalization process is performed with the database and Pannasoft Ingenuity will retrieve the normalized data from the database. The normalization process aims to convert all unprocessed data into fuzzy numbers which are later used by Pannasoft Ingenuity.

Section 1: Raw Data

In the following paragraphs, an example based on German Credit Data (a Statlog Project Database[1] obtained from UCI machine learning repository), is used to illustrate the data preparation and normalization process. The raw data is derived from a table called "German". The table contains the following fields:

Field	**Field type**	**Description**
ID	int	Unique Identity
checking	nvarchar	Status of existing checking account
Duration	float	Duration in month
history	nvarchar	Credit history
purpose	nvarchar	Purpose of loan
Credit amount	float	
Saving	nvarchar	Savings account/bonds
Employ	nvarchar	Present employment since
Installment	float	Installment rate in percentage of disposable income
Status	nvarchar	Personal status and sex
Debtors	nvarchar	Other debtors / guarantors
Residence	smallint	Present residence since
property	nvarchar	
Age	smallint	Age in years
Other Plan	nvarchar	Other installment plans
Housing	nvarchar	
Existing Credit	float	Number of existing credits at this bank
Job	nvarchar	
Liability	smallint	Number of dependents
Tel	nvarchar	
Foreign Worker	Nvarchar	
class	smallint	

Pannasoft Ingenuity does not use raw data or unprocessed data in the learning process. Raw data needs to be processed and converted into numerical format (between 0 to 1) before Pannasoft Ingenuity can use it for the learning process. For example, the "German" data table is

---

[1] Download from ftp://ftp.ics.uci.edu/pub/machine-learning-databases/statlog

stored with various types of data. Data preparation is required to convert those stored data into numerical format via the above mentioned data normalization process.

In the "German" table, each raw data record needs to be represented by a unique ID and it is the primary reference key for Pannasoft Ingenuity to interface with your existing system. To boost up the overall performance of Pannasoft Ingenuity, the ID mentioned earlier needs to be indexed in your database object. Example of this demonstration, the unique ID in "German" table is a column named "ID".

For other existing raw data, it can be sourced from other database object such as View. This flexibility enables Pannasoft Ingenuity to easily integrate with the existing data mart, providing a way to look at the existing database data in one or more tables (or other views) and deliver it as an input for Pannasoft Ingenuity.

Pannasoft Ingenuity is a supervised learning system that requires a set of normalized data with record id, features and class. With reference to the "German" table, column "checking" and "duration" are selected as features while "class" is selected as class.

Section 2: Pannasoft Ingenuity's Reference Table

Pair table that needs to co-exist with the raw data is called Pannasoft Ingenuity's reference table[2], i.e., "GCDATA". The details of "GCDATA" table and its columns are described in Appendix A. Use the script below to create the reference table, i.e., "GCDATA".

Script: Pannasoft Ingenuity's reference table (GCDATA)

```
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GCDATA]')
and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[GCDATA]
GO

CREATE TABLE [dbo].[GCDATA] (
       [ID] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
       [CRDATE] [datetime] NOT NULL,
       [TRFLAG] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
       [PRFLAG] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
       [RESULTS] [float] NOT NULL,
       [MDDATE] [datetime] NULL,
       [CONFIDEN] [float] NULL,
       [REC_ID] [int] IDENTITY (1, 1) NOT NULL
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[GCDATA] WITH NOCHECK ADD
       CONSTRAINT [PK_CRSDATA] PRIMARY KEY CLUSTERED
       (
              [REC_ID]
       )  ON [PRIMARY]
GO

CREATE UNIQUE INDEX [INX_CRSDATA] ON [dbo].[GCDATA]([ID]) ON [PRIMARY]
GO
```

Pannasoft Ingenuity's reference table is the main reference table with the raw data during execution of the modules. It is also the prerequisite table for the normalization process.

The creation of the unique index ("ID" in "German" table) is to boost up record searches via raw data primary reference key. The index is required in the reference table.

To execute the learning and prediction modules, all columns in the reference table have a role to play in determining the behavior of the modules, especially the "TRFLAG" and "PRFLAG", for processing raw data. The predicted result will be stored in batch mode in this table at the end of the execution of the prediction module. Details of each column in Pannasoft Ingenuity's reference table are described in Appendix A in Pannasoft Ingenuity User Guide.

---

[2] Naming convention for Pannasoft Ingenuity's reference table in this document is created in <Job_Code>DATA format

Section 3: Trigger for automating data manipulation

A database object called trigger is required to automate record insertion into the “GCDATA” table when there is a new record created in the raw data (referring to “German” table in this demonstration). The user is required to create this trigger to enable the process. No sample script is provided in this documentation.

This insertion requires the same primary reference key (which is “ID”) from table “German” to be put into column “ID” in “GCDATA” table as mentioned above. The same activities are applicable to the updating and deletion process in raw data. (Detailed flag used to indicate the insertion, updating or deletion process occurs in table raw data and take note of Pannasoft Ingenuity’s reference data are described by item “TRFLAG” in Appendix A in Pannasoft Ingenuity User Guide).

Pannasoft Ingenuity only performs selection on the raw data (which is “German” table), and manipulates data in Pannasoft Ingenuity’s reference table (which is “GCDATA” table). Database tuning should be concentrated on these two main reference tables and its related database object in the future.

Section 4: Preparation steps for normalization

After the GCDATA table and database object trigger are created, the user can start the normalization process for the raw data by using the following steps. For example, “features” as the column name in the “German” table used in this demonstration (based on the German Credit Data) is “Checking” and “Duration”, whereas the class, as the column name in the “German” table, is “Class”.

1) For column that contains string or characters (such as column “checking”).

i) Create a description table for string or characters data that represents feature for Pannasoft Ingenuity, i.e., NormGCChecking (Norm= Normalization; GC=German Credit; Checking=the column name) using a script as shown below:

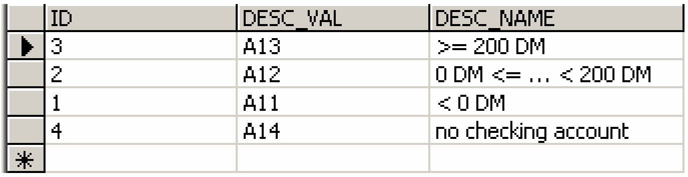
Script: NormYyyXxx Table

(Norm<Job_Code><Column_Name>)

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[NormGCChecking]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[NormGCChecking]
GO

CREATE TABLE [dbo].[NormGCChecking] (
   [ID] [int] NOT NULL,
   [DESC_VAL] [varchar] (20) COLLATE SQL_Latin1_General_CP1_CI_AS,
   [DESC_NAME] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS,
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[NormGCChecking] WITH NOCHECK ADD
   CONSTRAINT [PK_NormGCChecking] PRIMARY KEY CLUSTERED
   (
          [ID]
   )  ON [PRIMARY]
GO

CREATE UNIQUE INDEX [INX_NormGCChecking] ON
[dbo].[NormGCChecking]([DESC_VAL]) ON [PRIMARY]
GO
```

Column “ID” provided by the user, contains integer to represent a particular string in DESC_VAL, i.e., ID = 1 for DESC_VAL = A11. Column “DESC_VAL” contains all the various values in Column “checking” in the “German” table. The DESC_NAME contains explanation for the particular string in DESC_VAL. Please refer to Appendix A for more details on the data.

ii) Insert a numeric number into the column called “ID” in the table created in (i) to represent an existing string or character in “checking”. For example:



	ID	DESC_VAL	DESC_NAME
▶	3	A13	>= 200 DM
	2	A12	0 DM <= ... < 200 DM
	1	A11	< 0 DM
	4	A14	no checking account
✱			

Note: The purpose of creating this table is to allow the user to enter a representative integer for a particular string/character. The strings in column “DESC_VAL” are restricted data from “checking” in “German” table. ID = 0 is prohibited as it is a reserved value for any missing value/unknown data in “checking”. All the data shown above is required for normalization, which includes useful data as well as invalid/missing data. Those invalid/missing strings in the table above will be normalized to 0 to signify missing data.

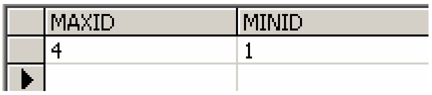
iii) Create a View to find the maximum and the minimum value of ID, excluding the ID that represents missing value or unknown data, i.e., NULL, using a script as shown below:

Script: NormYyyXxxLimit View

(Norm<Job_Code><Column_Name>Limit)

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[NormGCCheckingLimit]') and OBJECTPROPERTY(id,
N'IsView') = 1)
drop view [dbo].[NormGCCheckingLimit]
GO

CREATE VIEW [dbo].[NormGCCheckingLimit]
AS
SELECT MAX(ID) AS MAXID, MIN(ID) AS MINID
FROM NormGCChecking
WHERE DESC_VAL IS NOT NULL
GO
```

The result of this view is:



	MAXID	MINID
	4	1
▶		

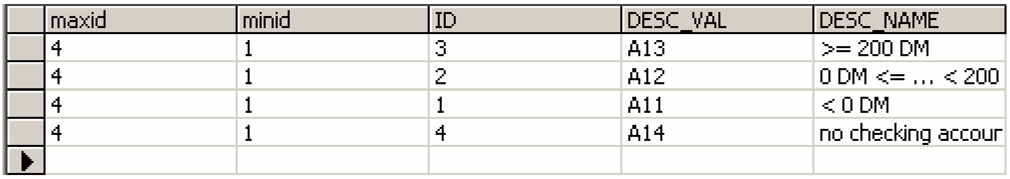
iv) Create another View to combine both (ii) and (iii) using a script as shown below:

Script: NormYyyXxxFilter View

(Norm<Job_Code><Column_Name>Filter)

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[NormGCCheckingFilter]') and OBJECTPROPERTY(id,
N'IsView') = 1)
drop view [dbo].[NormGCCheckingFilter]
GO

CREATE VIEW [dbo].[NormGCCheckingFilter]
AS
SELECT HL.maxid, HL.minid, DATA.*
FROM NormGCChecking DATA, NormGCCheckingLimit HL
GO
```

The result of this view is:



maxid	minid	ID	DESC_VAL	DESC_NAME
4	1	3	A13	>= 200 DM
4	1	2	A12	0 DM <= ... < 200
4	1	1	A11	< 0 DM
4	1	4	A14	no checking accour

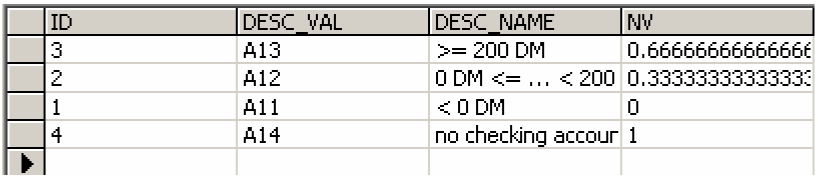
v) Create a View to calculate the normalization value, x ($0 \leq x \leq 1$) for the ID that represents a particular string/character using a script as follows:

Script: NormYyyXxxData View

(Norm<Job_Code><Column_Name>Data)

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[NormGCCheckingData]') and OBJECTPROPERTY(id,
N'IsView') = 1)
drop view [dbo].[NormGCCheckingData]
GO

Create View NormGCCheckingData
As
SELECT  NormGCCheckingFilter.ID,  NormGCCheckingFilter.DESC_VAL,
NormGCCheckingFilter.DESC_NAME,
CASE WHEN ((SELECT count(*) FROM German WHERE checking IS NULL)>0)
     THEN CASE WHEN (DESC_VAL is NOT NULL)
               THEN ((((CAST(NormGCCheckingFilter.ID as
Decimal(10,6))-CAST(NormGCCheckingFilter.minid as Decimal(10,6)))/
                       (CAST(NormGCCheckingFilter.maxid as
Decimal(10,6))-CAST(NormGCCheckingFilter.minid as
Decimal(10,6))))*0.75)+0.25)

               WHEN (DESC_VAL is NULL)
               THEN 0
          END
     ELSE
         ((CAST(NormGCCheckingFilter.ID as Decimal(10,6))-
CAST(NormGCCheckingFilter.minid as Decimal(10,6)))/
          (CAST(NormGCCheckingFilter.maxid as Decimal(10,6))-
CAST(NormGCCheckingFilter.minid as Decimal(10,6))))
     END AS NV
FROM  NormGCCheckingFilter
GO
```

The result of this View is:



ID	DESC_VAL	DESC_NAME	NV
3	A13	>= 200 DM	0.6666666666666
2	A12	0 DM <= ... < 200	0.3333333333333
1	A11	< 0 DM	0
4	A14	no checking accour	1

2) For column containing numeric value

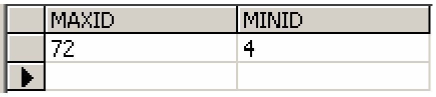
i) For column containing numerical numbers, the user can by-pass the description table, and step directly into creating a View to find the maximum and minimum value of the column from the raw data, excluding any missing value or unknown data, i.e., NULL, using a script as shown below. The example here is “duration” in “German” table.

Script: NormYyyXxxLimit View

(Norm<Job_Code><Column_Name>Limit)

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[NormGCDurationLimit]') and OBJECTPROPERTY(id,
N'IsView') = 1)
drop view [dbo].[NormGCDurationLimit]
GO

CREATE VIEW [dbo].[NormGCDurationLimit]
AS
select max(duration) as MAXID, min(duration) as MINID
from german where duration is not null
GO
```

The result of this View is:



MAXID	MINID
72	4

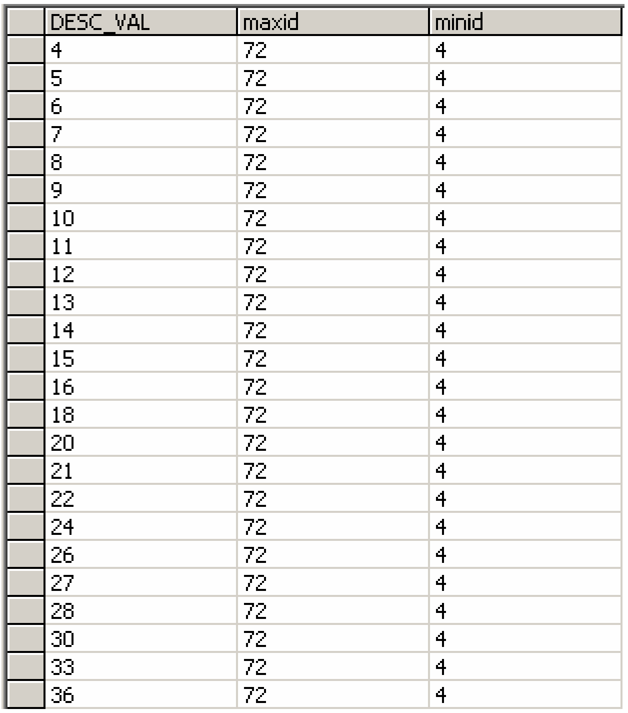
ii) Create a view to display the minimum, maximum and distinct data using a script as follows:

Script: NormYyyXxxFilter View

(Norm<Job_Code><Column_Name>Filter)

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[NormGCDurationFilter]') and OBJECTPROPERTY(id,
N'IsView') = 1)
drop view [dbo].[NormGCDurationFilter]
GO

CREATE VIEW [dbo].[NormGCDurationFilter]
AS
SELECT distinct(DATA.duration) as DESC_VAL, HL.maxid, HL.minid
FROM german DATA, NormGCDurationLimit HL
GO
```

The result of this View is:



DESC_VAL	maxid	minid
4	72	4
5	72	4
6	72	4
7	72	4
8	72	4
9	72	4
10	72	4
11	72	4
12	72	4
13	72	4
14	72	4
15	72	4
16	72	4
18	72	4
20	72	4
21	72	4
22	72	4
24	72	4
26	72	4
27	72	4
28	72	4
30	72	4
33	72	4
36	72	4

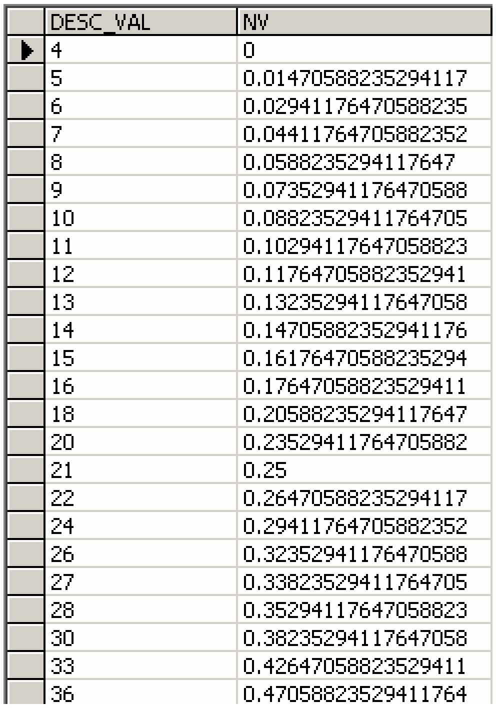
iii) The process continues with a view to calculate the normalization value, x ($0 \leq x \leq 1$) for DESC_VAL that represents a particular number using a script as follows:

Script: NormYyyXxxData View

(Norm<Job_Code><Column_Name>Data)

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[NormGCDurationData]') and OBJECTPROPERTY(id,
N'IsView') = 1)
drop view [dbo].[NormGCDurationData]
GO

CREATE View [dbo].[NormGCDurationData]
As
SELECT  NormGCDurationFilter.DESC_VAL,
CASE WHEN ((SELECT count(*) FROM german WHERE duration IS NULL)>0)
     THEN CASE WHEN (DESC_VAL is NOT NULL)
               THEN ((((CAST(NormGCDurationFilter.DESC_VAL as
Decimal(10,6))-CAST(NormGCDurationFilter.minid as Decimal(10,6)))/
                       (CAST(NormGCDurationFilter.maxid as
Decimal(10,6))-CAST(NormGCDurationFilter.minid as
Decimal(10,6))))*0.75)+0.25)

               WHEN (DESC_VAL is NULL)
               THEN 0
          END
     ELSE
         ((CAST(NormGCDurationFilter.DESC_VAL as Decimal(10,6))-
CAST(NormGCDurationFilter.minid as Decimal(10,6)))/
          (CAST(NormGCDurationFilter.maxid as Decimal(10,6))-
CAST(NormGCDurationFilter.minid as Decimal(10,6))))
     END AS NV
FROM  NormGCDurationFilter
GO
```

The result of this view is:



DESC_VAL	NV
4	0
5	0.01470588235294117
6	0.02941176470588235
7	0.04411764705882352
8	0.0588235294117647
9	0.07352941176470588
10	0.08823529411764705
11	0.10294117647058823
12	0.11764705882352941
13	0.13235294117647058
14	0.14705882352941176
15	0.16176470588235294
16	0.17647058823529411
18	0.20588235294117647
20	0.23529411764705882
21	0.25
22	0.26470588235294117
24	0.29411764705882352
26	0.32352941176470588
27	0.33823529411764705
28	0.35294117647058823
30	0.38235294117647058
33	0.42647058823529411
36	0.47058823529411764

3) Normalization for target/class value

As stated above, Pannasoft Ingenuity is a supervised learning system that requires a class. The normalization procedure is the same as normalization of the column containing strings or characters. First, create a Table, i.e., NormGCClass, and insert all the classes that the user needs into this table. Then, find the maximum and the minimum value of the class using a view, i.e., NormGCClassLimit. Next, create a view, i.e., NormGCClassFilter to combine both NormGCClass and NormGCClassLimit. Finally, calculate the normalization for the class using a view, i.e., NormGCClassData.

Various GCClass sample scripts are presented in Appendix B.

4) **After normalization has been made for each feature and class, combine all the normalized data using a view as follows:**

Script: ViewYyyNormData View

(View<Job_Code>NormData)

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[ViewGCNormData]') and OBJECTPROPERTY(id,
N'IsView') = 1)
drop view [dbo].[ViewGCNormData]
GO

CREATE View [dbo].[ViewGCNormData]
AS
SELECT GCDATA.REC_ID, NormGCClassData.nv as class,
NormGCCheckingData.nv as checking, NormGCDurationData.nv as duration
FROM (german
inner join GCDATA on GCDATA.rec_id = german.ID
inner join NormGCCheckingData on NormGCCheckingData.desc_val =
german.checking
OR (NormGCCheckingData.desc_val is null and german.checking is null)
inner join NormGCDurationData on NormGCDurationData.desc_val =
german.duration
OR (NormGCDurationData.desc_val is null and german.duration is null)
inner join NormGCClassData on NormGCClassData.desc_val = german.class
OR (NormGCClassData.desc_val is null and german.class is null))
GO
```

Note:

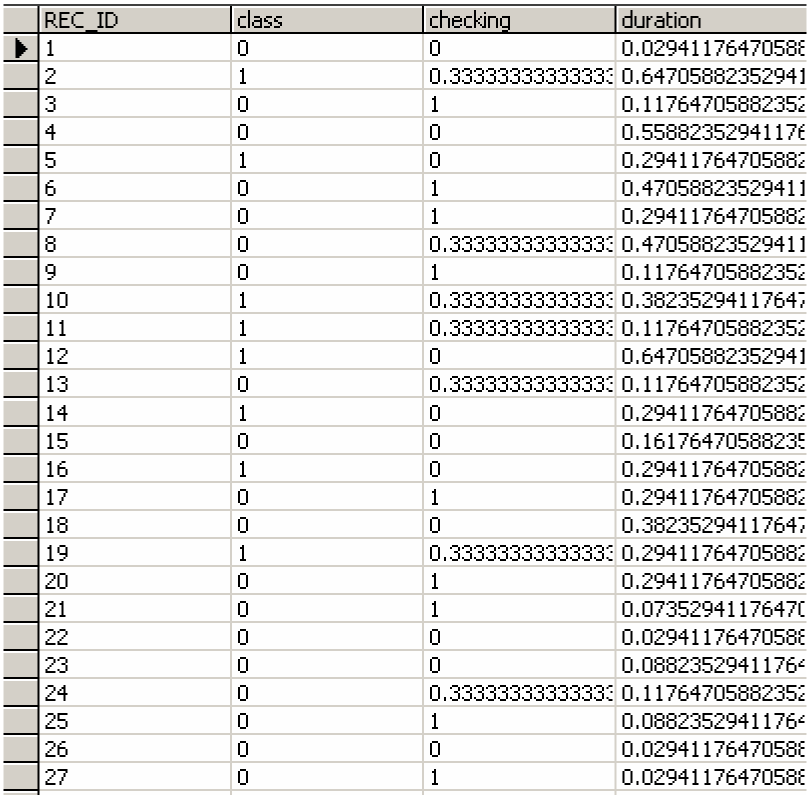
This view name needs to be specified in the configuration file *pannasoft.properties*. From here, Pannasoft Ingenuity will know the destination database object that keeps the processed data.

The result of the script above gives a complete normalized data set that is needed by Pannasoft Ingenuity with the first column indicating the unique key for the particular patterns. The second column shows the class/target for the particular pattern and the rest of the columns are features represented by the class/target.

If the user does not find any normalized data appearing in the View<Job_Code>NormData after going through the 4 steps above, try the following solution:

- Verify whether the raw data required by Pannasoft Ingenuity has been inserted into <Job_Code>DATA Reference Table (example: GCDATA or WBCDATA) via the suggested database trigger.
- Check for existing data in its dependences view(s). For example, no data exists in View<Job_Code>NormData (example ViewGCNormData), verification of data existence should go to Norm<Job_Code><Column_Name>Data (for example NormGCCheckingData).

The result of this view is:



REC_ID	class	checking	duration
1	0	0	0.0294117647058
2	1	0.33333333333333	0.64705882352941
3	0	1	0.1176470588235
4	0	0	0.5588235294117
5	1	0	0.2941176470588
6	0	1	0.47058823529411
7	0	1	0.2941176470588
8	0	0.33333333333333	0.47058823529411
9	0	1	0.1176470588235
10	1	0.33333333333333	0.3823529411764
11	1	0.33333333333333	0.1176470588235
12	1	0	0.64705882352941
13	0	0.33333333333333	0.1176470588235
14	1	0	0.2941176470588
15	0	0	0.1617647058823
16	1	0	0.2941176470588
17	0	1	0.2941176470588
18	0	0	0.3823529411764
19	1	0.33333333333333	0.2941176470588
20	0	1	0.2941176470588
21	0	1	0.0735294117647
22	0	0	0.0294117647058
23	0	0	0.0882352941176
24	0	0.33333333333333	0.1176470588235
25	0	1	0.0882352941176
26	0	0	0.0294117647058
27	0	1	0.0294117647058

Steps 1 to 4 above create view names with Pannasoft's suggested naming convention. You can locate your own naming convention for ease of use and for your integration purposes.

Section 5: Handling missing data in the submitted raw data

In this section, an example based on Wisconsin Breast Cancer[3] (obtained from UCI machine learning repository) is used to illustrate the normalization process on column that contains missing data[4]. The raw data set is derived from a table called “WBCancer”. The table contains the following fields:

Field	Field type	Description
ID	int	Unique ID
Clump	float	Clump Thickness
CellSize	float	Uniformity of Cell Size
CellShap	float	Uniformity of Cell Shape
Adhesion	float	Marginal Adhesion
EpiCell	float	Single Epithelial Cell Size
Nuclei	float	Bare Nuclei
Chroma	float	Bland Chromatin
Nucleoli	float	Normal Nucleoli
Mitoses	float	
class	float	2 for benign, 4 for malignant

In this data, feature stored in column “Nuclei” is used to illustrate how the normalization works on column that contains 16 missing values, which are identified as NULL. All the fields in WBCancer are numerical values. Therefore, normalization on numerical value is applied in this demonstration.

i) The same step as in the previous method applies. Users can by-pass the description table due to the nature of the target raw data that are not string/character and proceed directly to creating a View to find the maximum and minimum value of the column from the raw data, excluding any missing value or unknown data, i.e., NULL, using a script as shown below.

Script: NormYyyXxxLimit View

(Norm<Job_Code><Column_Name>Limit)

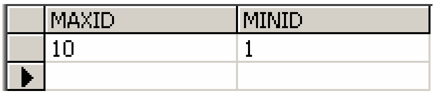
```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[NormWBCNucleiLimit]') and OBJECTPROPERTY(id,
N'IsView') = 1)
drop view [dbo].[NormWBCNucleiLimit]
GO

CREATE VIEW [dbo].[NormWBCNucleiLimit]
AS
select max(Nuclei) as MAXID, min(Nuclei) as MINID
from WBCancer where Nuclei is not null
GO
```

---

[3] Download from ftp://ftp.ics.uci.edu/pub/machine-learning-databases/breast-cancer-wisconsin

[4] Handling extraordinary description of Missing Data is described in Appendix C.

The result of this View is:



MAXID	MINID
10	1

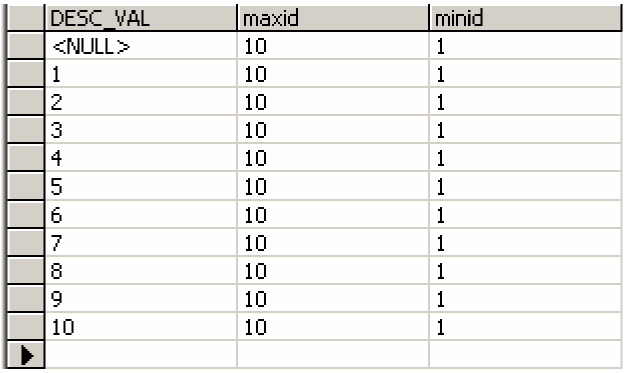
ii) Create a view to display the minimum, maximum, and distinct data using the following script:

Script: NormYyyXxxFilter View

(Norm<Job_Code><Column_Name>Filter)

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[NormWBCNucleiFilter]') and OBJECTPROPERTY(id,
N'IsView') = 1)
drop view [dbo].[NormWBCNucleiFilter]
GO

CREATE VIEW [dbo].[NormWBCNucleiFilter]
AS
SELECT distinct(DATA.Nuclei) as DESC_VAL, HL.maxid, HL.minid
FROM WBCancer DATA, NormWBCNucleiLimit HL
GO
```

The result of this View is shown below:



DESC_VAL	maxid	minid
<NULL>	10	1
1	10	1
2	10	1
3	10	1
4	10	1
5	10	1
6	10	1
7	10	1
8	10	1
9	10	1
10	10	1

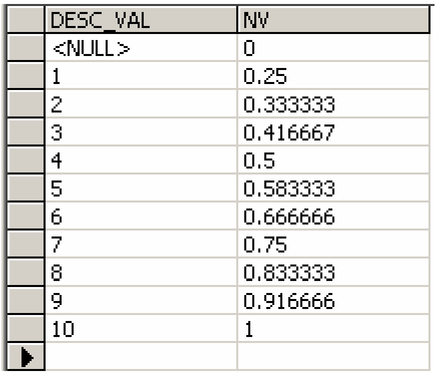
iii) The process continues with a view to calculate the normalization value, x ($0 \le x \le 1$) for DESC_VAL that represents a particular number using the following script:

Script: NormYyyXxxData View

(Norm<Job_Code><Column_Name>Filter)

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[NormWBCNucleiData]') and OBJECTPROPERTY(id,
N'IsView') = 1)
drop view [dbo].[NormWBCNucleiData]
GO

CREATE View NormWBCNucleiData
As
SELECT  NormWBCNucleiFilter.DESC_VAL,
CASE  WHEN (DESC_VAL is NOT NULL)
            THEN ((((CAST(NormWBCNucleiFilter.DESC_VAL as
Decimal(106))-CAST(NormWBCNucleiFilter.minid as Decimal(10,6)))/
                       (CAST(NormWBCNucleiFilter.maxid as
Decimal(10,6))-CAST(NormWBCNucleiFilter.minid as
Decimal(10,6))))*0.75)+0.25)

             WHEN (DESC_VAL is NULL)
             THEN 0  END AS NV
FROM  NormWBCNucleiFilter
GO
```

The result of this view is:



DESC_VAL	NV
<NULL>	0
1	0.25
2	0.333333
3	0.416667
4	0.5
5	0.583333
6	0.666666
7	0.75
8	0.833333
9	0.916666
10	1

Note: for DESC_VAL = NULL (here NULL = missing data), the normalized value (NV) becomes 0

APPENDIX A: ATTRIBUTES DESCRIPTION FOR GERMAN CREDIT DATA

```
Attribute 1:  (qualitative) - Status of existing checking account
             A11 :      ... <    0 DM
             A12 : 0 <= ... <  200 DM
             A13 :      ... >= 200 DM/salary assignments for at least 1
                              year
             A14 : no checking account

Attribute 2:  (numerical) - Duration in month

Attribute 3:  (qualitative) - Credit history
            A30 : no credits taken/all credits paid back duly
            A31 : all credits at this bank paid back duly
            A32 : existing credits paid back duly till now
            A33 : delay in paying off in the past
            A34 : critical account/other credits existing (not at this
                    bank)

Attribute 4:  (qualitative) - Purpose
            A40 : car (new)
            A41 : car (used)
            A42 : furniture/equipment
            A43 : radio/television
            A44 : domestic appliances
            A45 : repairs
            A46 : education
            A47 : (vacation - does not exist?)
            A48 : retraining
            A49 : business
            A410 : others

Attribute 5:  (numerical) - Credit amount

Attribute 6:  (qualitative) - Savings account/bonds
            A61 :          ... <  100 DM
            A62 :   100 <= ... <  500 DM
            A63 :   500 <= ... < 1000 DM
            A64 :          .. >= 1000 DM
            A65 :   unknown/ no savings account

Attribute 7:  (qualitative) - Present employment since
            A71 : unemployed
            A72 :       ... < 1 year
            A73 : 1  <= ... < 4 years
            A74 : 4  <= ... < 7 years
            A75 :       .. >= 7 years

Attribute 8:  (numerical - Installment rate in percentage of disposable
income



Attribute 9:  (qualitative) - Personal status and sex
```

```
            A91 : male   : divorced/separated
            A92 : female : divorced/separated/married
            A93 : male   : single
            A94 : male   : married/widowed
            A95 : female : single

Attribute 10: (qualitative) - Other debtors / guarantors
            A101 : none
            A102 : co-applicant
            A103 : guarantor

Attribute 11: (numerical) - Present residence since

Attribute 12: (qualitative) - Property
            A121 : real estate
            A122 : if not
            A121 : building society savings agreement/life insurance
            A123 : if not A121/A122 : car or other, not in attribute 6
            A124 : unknown / no property

Attribute 13: (numerical) - Age in years

Attribute 14: (qualitative) - Other installment plans
            A141 : bank
            A142 : stores
            A143 : none

Attribute 15: (qualitative) - Housing
            A151 : rent
            A152 : own
            A153 : for free

Attribute 16: (numerical) - Number of existing credits at this bank

Attribute 17: (qualitative) - Job
            A171 : unemployed/ unskilled  - non-resident
            A172 : unskilled - resident
            A173 : skilled employee / official
            A174 : management/ self-employed/highly qualified employee/
                     officer

Attribute 18: (numerical) - Number of people being liable to provide
maintenance for

Attribute 19: (qualitative) - Telephone
            A191 : none
            A192 : yes, registered under the customers name

Attribute 20: (qualitative) - foreign worker
            A201 : yes
            A202 : no
```

APPENDIX B: VARIOUS SAMPLE GCCLASS SCRIPTS

Script: NormGCClass Table

(Norm<Job_Code><Column_Name>)

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[NormGCClass]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[NormGCClass]
GO

CREATE TABLE [dbo].[NormGCClass] (
       [ID] [int] NOT NULL,
       [DESC_VAL] [varchar] (20) COLLATE SQL_Latin1_General_CP1_CI_AS,
       [DESC_NAME] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS,
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[NormGCClass] WITH NOCHECK ADD
       CONSTRAINT [PK_NormGCClass] PRIMARY KEY CLUSTERED
       (
              [ID]
       )  ON [PRIMARY]
GO

CREATE UNIQUE INDEX [INX_NormGCClass] ON [dbo].[NormGCClass]([DESC_VAL]) ON
[PRIMARY]
GO
```

Script: NormGCClassLimit Table

(Norm<Job_Code><Column_Name>Limit)

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[NormGCClassLimit]') and OBJECTPROPERTY(id, N'IsView') = 1)
drop view [dbo].[NormGCClassLimit]
GO

CREATE VIEW [dbo].[NormGCClassLimit]
AS
SELECT MAX(ID) AS MAXID, MIN(ID) AS MINID
FROM NormGCClass
WHERE DESC_VAL IS NOT NULL
GO
```

Script: NormGCClassFilter Table

(Norm<Job_Code><Column_Name>Filter)

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[NormGCClassFilter]') and OBJECTPROPERTY(id, N'IsView') = 1)
drop view [dbo].[NormGCClassFilter]
GO

CREATE VIEW [dbo].[NormGCClassFilter]
AS
SELECT HL.maxid, HL.minid, DATA.*
FROM NormGCClass DATA, NormGCClassLimit HL
GO
```

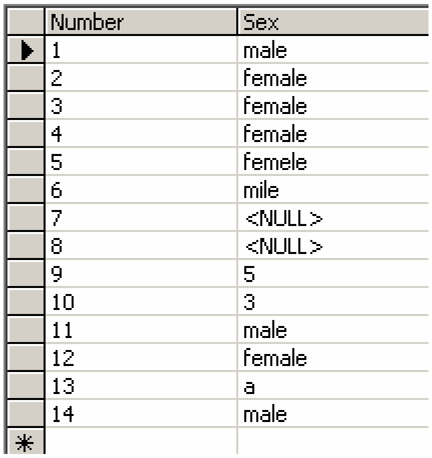
Script: NormGCClassData Table

(Norm<Job_Code><Column_Name>Data)

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[NormGCClassData]') and OBJECTPROPERTY(id, N'IsView') = 1)
drop view [dbo].[NormGCClassData]
GO

Create View [dbo].[NormGCClassData]
As
SELECT  NormGCClassFilter.ID,  NormGCClassFilter.DESC_VAL,
NormGCClassFilter.DESC_NAME,
CASE WHEN ((SELECT count(*) FROM German WHERE class IS NULL)>0)
     THEN CASE WHEN (DESC_VAL is NOT NULL)
               THEN ((((CAST(NormGCClassFilter.ID as Decimal(10,6))-
CAST(NormGCClassFilter.minid as Decimal(10,6)))/
                       (CAST(NormGCClassFilter.maxid as Decimal(10,6))-
CAST(NormGCClassFilter.minid as Decimal(10,6))))*0.75)+0.25)

               WHEN (DESC_VAL is NULL)
               THEN 0
          END
     ELSE
         ((CAST(NormGCClassFilter.ID as Decimal(10,6))-
CAST(NormGCClassFilter.minid as Decimal(10,6)))/
          (CAST(NormGCClassFilter.maxid as Decimal(10,6))-
CAST(NormGCClassFilter.minid as Decimal(10,6))))
     END AS NV
FROM  NormGCClassFilter
GO
```

APPENDIX C: EXTRAORDINARY MISSING DATA HANDLING

Missing data (a missing or null data value in a field) indicates the absence of any meaningful or valid information in features. Pannasoft Ingenuity does not allow any missing or incomplete data in the target column. To illustrate this statement, an example is given based on the following figure.

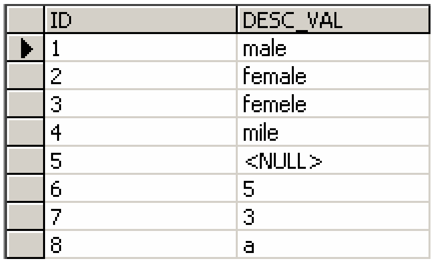


Number	Sex
1	male
2	female
3	female
4	female
5	femele
6	mile
7	<NULL>
8	<NULL>
9	5
10	3
11	male
12	female
13	a
14	male

From the above figure, logical acceptance for “sex” is either male or female. Other values are indicated as invalid or missing data. To continue the normalization process on this column, create a description table with the following script and insert an integer to represent a particular string in column “sex” as shown in the figure. In this table, only the strings encountered are allowed to include missing or invalid data.

Table: Norm<Job_Code><Column_Name> (NormGCSex)

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[NormGcSex]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[NormGcSex]
GO

CREATE TABLE [dbo].[NormGcSex] (
       [ID] [int] NULL ,
       [DESC_VAL] [varchar] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[NormGcSex] WITH NOCHECK ADD
       CONSTRAINT [PK_NormGcSex] PRIMARY KEY CLUSTERED
       (
              [ID]
       )  ON [PRIMARY]
GO

CREATE UNIQUE INDEX [INX_NormGcSex] ON [dbo].[NormGcSex]([DESC_VAL]) ON
[PRIMARY]
GO
```



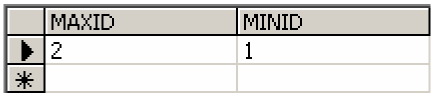
ID	DESC_VAL
1	male
2	female
3	femele
4	mile
5	<NULL>
6	5
7	3
8	a

While creating the database object called View Norm<Job_Code><Column_Name>Limit to find the maximum and the minimum value of ID, the user should try to exclude missing or invalid data. Two types of “where” clause can be used to exclude missing data by using the following script and both will produce the same result.

1) WHERE DESC_VAL='male' or DESC_VAL='female', or
2) WHERE DESC_VAL IS NOT NULL AND DESC_VAL<>'femele' AND DESC_VAL<>'mile' AND DESC_VAL<>'a' AND DESC_VAL <>'5' AND DESC_VAL <>'3'

View: Norm<Job_Code><Column_Name>Limit (NormGCSexLimit)

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[NormGCSexLimit]') and OBJECTPROPERTY(id, N'IsView') = 1)
drop view [dbo].[NormGCSexLimit]
GO

CREATE VIEW [dbo].[NormGCSexLimit]
AS
SELECT MAX(ID) AS MAXID, MIN(ID) AS MINID
FROM NormGCSex
WHERE DESC_VAL='male' or DESC_VAL='female'
GO
```

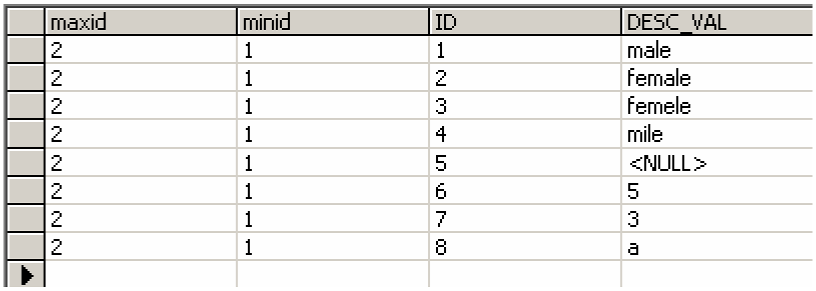


MAXID	MINID
2	1

Create another View to combine both view and table using the following script:

View: Norm<Job_Code><Column_Name>Filter (Example: NormGCSexFilter)

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[NormGCSexFilter]') and OBJECTPROPERTY(id, N'IsView') = 1)
drop view [dbo].[NormGCSexFilter]
GO

CREATE VIEW NormGCSexFilter
AS
SELECT HL.maxid, HL.minid, DATA.*
FROM NormGCSex DATA, NormGCSexLimit HL
GO
```



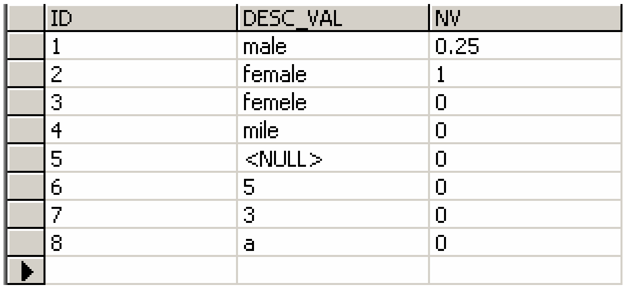
maxid	minid	ID	DESC_VAL
2	1	1	male
2	1	2	female
2	1	3	femele
2	1	4	mile
2	1	5	<NULL>
2	1	6	5
2	1	7	3
2	1	8	a

Finally, create a view to perform normalization on column “sex”

View: Norm<Job_Code><Column_Name>Data (Example: NormGCSexData)

```
if     exists     (select     *     from     dbo.sysobjects     where     id     =
object_id(N'[dbo].[NormGCSexData]') and OBJECTPROPERTY(id, N'IsView') = 1)
drop view [dbo].[NormGCSexData]
GO

Create View NormGCSexData
As
SELECT  NormGCSexFilter.ID,  NormGCSexFilter.DESC_VAL,
CASE  WHEN  ((SELECT  count(*)  FROM  NormGCSex  WHERE  DESC_VAL<>'male'  AND
DESC_VAL<>'female')>0)
     THEN CASE WHEN (DESC_VAL='male' OR DESC_VAL='female')
               THEN     ((((CAST(NormGCSexFilter.ID     as     Decimal(10,6))-
CAST(NormGCSexFilter.minid as Decimal(10,6)))/
                       (CAST(NormGCSexFilter.maxid     as     Decimal(10,6))-
CAST(NormGCSexFilter.minid as Decimal(10,6))))*0.75)+0.25)

               ELSE
               0
          END
     ELSE
         ((CAST(NormGCSexFilter.ID             as             Decimal(10,6))-
CAST(NormGCSexFilter.minid as Decimal(10,6)))/
          (CAST(NormGCSexFilter.maxid            as            Decimal(10,6))-
CAST(NormGCSexFilter.minid as Decimal(10,6))))
     END AS NV
FROM  NormGCSexFilter
GO
```

The result of this view is:



ID	DESC_VAL	NV
1	male	0.25
2	female	1
3	femele	0
4	mile	0
5	<NULL>	0
6	5	0
7	3	0
8	a	0

Take note of the SQL statement in blue on the previous script. It is important for the user to be careful when applying it into different columns which contain different data. From the result, one can see that all invalid/missing data have been replaced by 0.

APPENDIX D: LINGUISTIC RULE SCRIPTS FOR GERMAN CREDIT DATA

Assuming the raw data description for German Credit Data is as follows:

Field	Field type	Description
ID	int	Unique identity
checking	nvarchar	Status of existing checking account
Duration	float	Duration in month
history	nvarchar	Credit history
purpose	nvarchar	Purpose of loan
Credit amount	float	
Saving	nvarchar	Savings account/bonds
Employ	nvarchar	Present employment since
Installment	float	Installment rate in percentage of disposable income
Status	nvarchar	Personal status and sex
Debtors	nvarchar	Other debtors / guarantors
Residence	smallint	Present residence since
class	smallint	

The following script is used to describe only the 11 features mentioned above. Other features as stated in prior chapters are not included for the demonstration.

Use the script (View<Job_Code>Rule, example ViewGCRule) as follows to obtain a set of linguistic rules. Then, create a view to display the confidence factor of each rule as shown in View<Job_Code>CF (example “ViewGCCF”). Finally, combine both views to produce a complete linguistic rule set with confidence factors as shown in View<Job_Code>RuleExtraction (example “VIewGCRuleExtraction”).

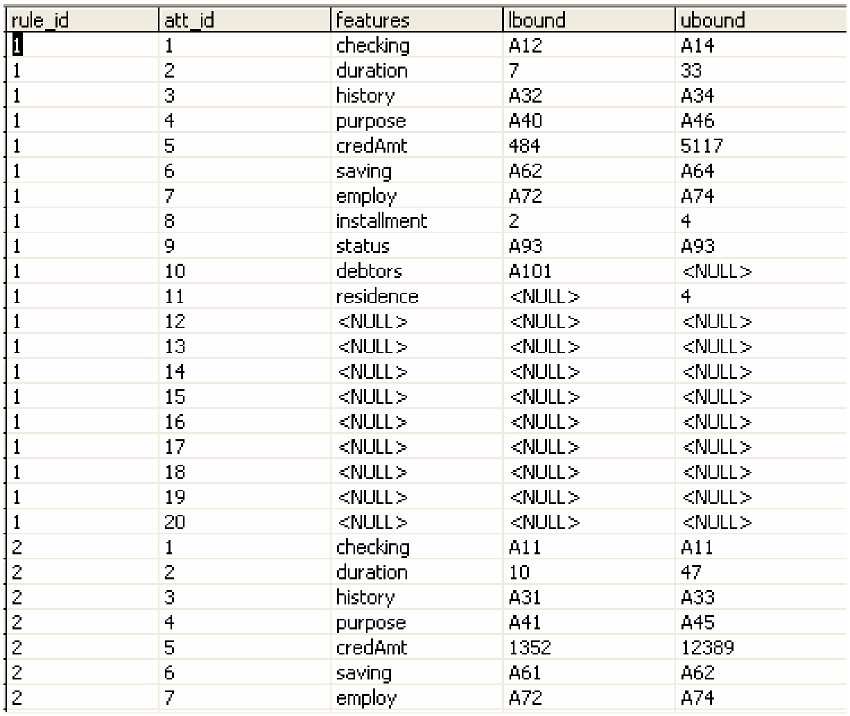
Script: ViewYyyRule View

(View<Job_Code>Rule)

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[ViewWBCRule]') and OBJECTPROPERTY(id, N'IsView') = 1)
drop view [dbo].[ViewGCRule]
GO

Create View [dbo].[ViewGCRule]
As
select rule_id, att_id,
case when att_id=1 then 'checking'
     when att_id=2 then 'duration'
     when att_id=3 then 'history'
     when att_id=4 then 'purpose'
     when att_id=5 then 'credAmt'
     when att_id=6 then 'saving'
     when att_id=7 then 'employ'
     when att_id=8 then 'installment'
     when att_id=9 then 'status'
     when att_id=10 then 'debtors'
     when att_id=11 then 'residence'
       end as features,
case when att_id=1 then (select CAST(desc_val as VARCHAR) from
NormGCCheckingData where nv=(select min(nv) from NormGCCheckingData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=2 then (select CAST(desc_val as VARCHAR) from
NormGCDurationData where nv=(select min(nv) from NormGCDurationData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=3 then (select CAST(desc_val as VARCHAR) from
NormGCHistoryData where nv=(select min(nv) from NormGCHistoryData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=4 then (select CAST(desc_val as VARCHAR) from
NormGCPurposeData where nv=(select min(nv) from NormGCPurposeData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=5 then (select CAST(desc_val as VARCHAR) from
NormGCCreditAmountData where nv=(select min(nv) from
NormGCCreditAmountData where nv>=LBOUND and nv<=UBOUND))
     when att_id=6 then (select CAST(desc_val as VARCHAR) from
NormGCSavingData where nv=(select min(nv) from NormGCSavingData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=7 then (select CAST(desc_val as VARCHAR) from
NormGCEmployData where nv=(select min(nv) from NormGCEmployData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=8 then (select CAST(desc_val as VARCHAR) from
NormGCInstallmentData where nv=(select min(nv) from NormGCInstallmentData
where nv>=LBOUND and nv<=UBOUND))
     when att_id=9 then (select CAST(desc_val as VARCHAR) from
NormGCStatusData where nv=(select min(nv) from NormGCStatusData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=10 then (select CAST(desc_val as VARCHAR) from
NormGCDebtorsData where nv=(select min(nv) from NormGCCheckingData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=11 then (select CAST(desc_val as VARCHAR) from
NormGCResidenceData where nv=(select min(nv) from NormGCDurationData
where nv>=LBOUND and nv<=UBOUND))
       end as lbound,
case when att_id=1 then (select CAST(desc_val as VARCHAR) from
NormGCCheckingData where nv=(select max(nv) from NormGCCheckingData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=2 then (select CAST(desc_val as VARCHAR) from
NormGCDurationData where nv=(select max(nv) from NormGCDurationData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=3 then (select CAST(desc_val as VARCHAR) from
NormGCHistoryData where nv=(select max(nv) from NormGCHistoryData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=4 then (select CAST(desc_val as VARCHAR) from
NormGCPurposeData where nv=(select max(nv) from NormGCPurposeData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=5 then (select CAST(desc_val as VARCHAR) from
NormGCCreditAmountData where nv=(select max(nv) from
NormGCCreditAmountData where nv>=LBOUND and nv<=UBOUND))
     when att_id=6 then (select CAST(desc_val as VARCHAR) from
NormGCSavingData where nv=(select max(nv) from NormGCSavingData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=7 then (select CAST(desc_val as VARCHAR) from
NormGCEmployData where nv=(select max(nv) from NormGCEmployData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=8 then (select CAST(desc_val as VARCHAR) from
NormGCInstallmentData where nv=(select max(nv) from NormGCInstallmentData
where nv>=LBOUND and nv<=UBOUND))
     when att_id=9 then (select CAST(desc_val as VARCHAR) from
NormGCStatusData where nv=(select max(nv) from NormGCStatusData where
nv>=LBOUND and nv<=UBOUND))
```

```
     when att_id=10 then (select CAST(desc_val as VARCHAR) from
NormGCDebtorsData where nv=(select max(nv) from NormGCStatusData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=11 then (select CAST(desc_val as VARCHAR) from
NormGCResidenceData where nv=(select max(nv) from NormGCCheckingData
where nv>=LBOUND and nv<=UBOUND))
       end as ubound
from datarule where data_id='german_data' and voter_id=1
GO
```

The result of this view is:



rule_id	att_id	features	lbound	ubound
1	1	checking	A12	A14
1	2	duration	7	33
1	3	history	A32	A34
1	4	purpose	A40	A46
1	5	credAmt	484	5117
1	6	saving	A62	A64
1	7	employ	A72	A74
1	8	installment	2	4
1	9	status	A93	A93
1	10	debtors	A101	<NULL>
1	11	residence	<NULL>	4
1	12	<NULL>	<NULL>	<NULL>
1	13	<NULL>	<NULL>	<NULL>
1	14	<NULL>	<NULL>	<NULL>
1	15	<NULL>	<NULL>	<NULL>
1	16	<NULL>	<NULL>	<NULL>
1	17	<NULL>	<NULL>	<NULL>
1	18	<NULL>	<NULL>	<NULL>
1	19	<NULL>	<NULL>	<NULL>
1	20	<NULL>	<NULL>	<NULL>
2	1	checking	A11	A11
2	2	duration	10	47
2	3	history	A31	A33
2	4	purpose	A41	A45
2	5	credAmt	1352	12389
2	6	saving	A61	A62
2	7	employ	A72	A74

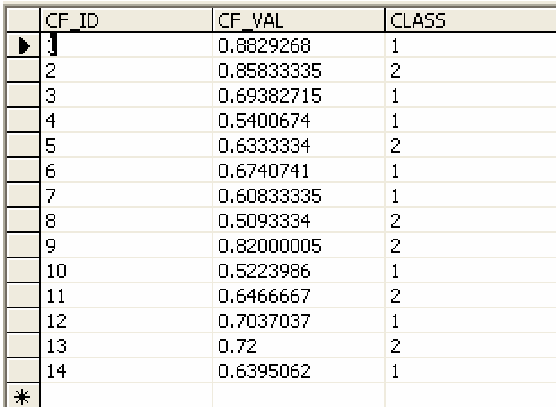
Note:
Features not implemented will be filled with <NULL> in column “features” as shown above.

Script: ViewYyyCF View

(View<Job_Code>CF)

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[ViewGCCF]') and OBJECTPROPERTY(id, N'IsView') = 1)
drop view [dbo].[ViewGCCF]
GO

CREATE View [dbo].[ViewGCCF]
AS
select CF_ID, CF_VAL,
(select desc_val from NormGCClassData where nv=class) as CLASS
from datacf
where data_id='german_data' and voter_id=1
GO
```

The result of this view is:

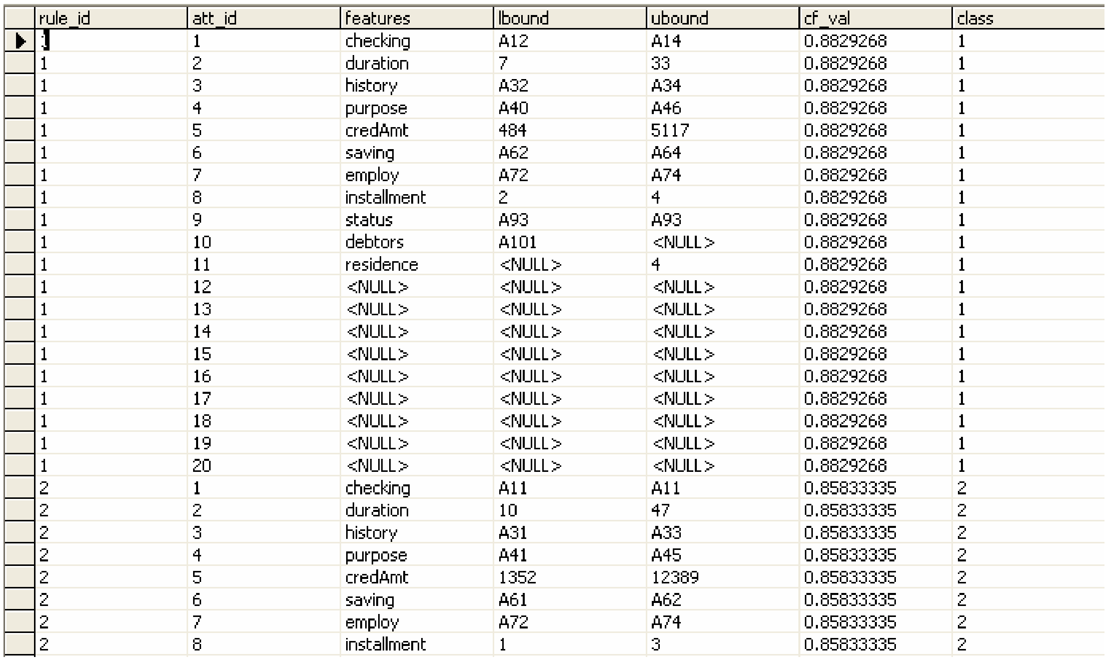


CF_ID	CF_VAL	CLASS
1	0.8829268	1
2	0.85833335	2
3	0.69382715	1
4	0.5400674	1
5	0.6333334	2
6	0.6740741	1
7	0.60833335	1
8	0.5093334	2
9	0.82000005	2
10	0.5223986	1
11	0.6466667	2
12	0.7037037	1
13	0.72	2
14	0.6395062	1

Script: ViewYyyRuleExtraction View (View<Job_Code>RuleExtraction)

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[ViewGCRuleExtraction]') and OBJECTPROPERTY(id, N'IsView')
= 1)
drop view [dbo].[ViewGCRuleExtraction]
GO

CREATE View [dbo].[ViewGCRuleExtraction]
As
select rule_id, att_id, features, lbound, ubound, cf_val, class
from ViewGCRule inner join ViewGCCF on ViewGCRule.rule_id=VIewGCCF.cf_id

GO
```

The result of this view is:



rule_id	att_id	features	lbound	ubound	cf_val	class
1	1	checking	A12	A14	0.8829268	1
1	2	duration	7	33	0.8829268	1
1	3	history	A32	A34	0.8829268	1
1	4	purpose	A40	A46	0.8829268	1
1	5	credAmt	484	5117	0.8829268	1
1	6	saving	A62	A64	0.8829268	1
1	7	employ	A72	A74	0.8829268	1
1	8	installment	2	4	0.8829268	1
1	9	status	A93	A93	0.8829268	1
1	10	debtors	A101	<NULL>	0.8829268	1
1	11	residence	<NULL>	4	0.8829268	1
1	12	<NULL>	<NULL>	<NULL>	0.8829268	1
1	13	<NULL>	<NULL>	<NULL>	0.8829268	1
1	14	<NULL>	<NULL>	<NULL>	0.8829268	1
1	15	<NULL>	<NULL>	<NULL>	0.8829268	1
1	16	<NULL>	<NULL>	<NULL>	0.8829268	1
1	17	<NULL>	<NULL>	<NULL>	0.8829268	1
1	18	<NULL>	<NULL>	<NULL>	0.8829268	1
1	19	<NULL>	<NULL>	<NULL>	0.8829268	1
1	20	<NULL>	<NULL>	<NULL>	0.8829268	1
2	1	checking	A11	A11	0.85833335	2
2	2	duration	10	47	0.85833335	2
2	3	history	A31	A33	0.85833335	2
2	4	purpose	A41	A45	0.85833335	2
2	5	credAmt	1352	12389	0.85833335	2
2	6	saving	A61	A62	0.85833335	2
2	7	employ	A72	A74	0.85833335	2
2	8	installment	1	3	0.85833335	2

About Pannasoft Technologies

Pannasoft Technologies was founded in 2003 with seed funding from a prominent venture capitalist. The company is a MSC (Multimedia Super Corridor) status company and an advanced partner for IBM Partnerworld. In 2004, Pannasoft Technologies was awarded the MGS (MSC Grant Scheme) to conduct R&D in data mining technology.

Pannasoft Technologies specializes in Soft Computing, a state-of-the-art technology that the company uses extensively to develop intelligent software solutions for a wide variety of industries including manufacturing, engineering, trading, education, healthcare and financial services. Our solutions enable monitoring and optimization of business and engineering processes to address their operational difficulties and to achieve optimal performance while minimizing costs and maximizing profits.

1-02-19, Jalan Mayang Pasir 1,
Mayang Mall,
11900 Bayan Lepas,
Penang,
Malaysia.

Tel : +604-645 5218
Fax : +604-645 2398

enquiry@pannasoft.com *email*
www.pannasoft.com *website*