Pannasoft Ingenuity User Guide

Revision 1.9

For Pannasoft Ingenuity Version 1.0.2.x

April 2005

TABLE OF CONTENTS

SECTION 1: INTRODUCTION

Artificial neural networks (ANNs) resembled the human nervous system, with algorithms consisting of weighted interconnecting processing units. ANNs are tried and tested artificial intelligence technology for superior pattern recognition. Pattern recognition offers benefits of solving practical problems in the business, medical and engineering world.

Proper data preprocessing, architecture selection and network training are required to reap the best from ANNs. The arduous task to optimized ANNs is alleviated with **Pannasoft Ingenuity**, the ANNs engine that is simplified for use.

Pannasoft Ingenuity's component engine is based on Autonomous Adaptive Algorithm™ (AAA), a hybrid artificial neural networks (ANNs) technology. AAA synergistically matches state-of-the-art machine learning algorithms such as neural computation, fuzzy computation, and evolutionary computation to form a hybrid algorithm. AAA combines the best features from all three methodologies to produce neural-fuzzy, fuzzy-evolutionary, and/or evolutionary-neural models.

Pannasoft Ingenuity comprised methods and algorithms specially configured and evaluated by Pannasoft's experts for optimized application on neural network-based solutions. **Pannasoft Ingenuity** provides the best-of-breed artificial neural networks technology to be infused into real world software products to achieve optimal solutions to specific problems, yielding numerous returns in terms of best results, enhanced value and competitive edge.

How This User Guide Is Organized

This user guide covers the steps required to get Pannasoft Ingenuity up and running inclusive of examples on how to complete the data normalization process, with description based on Microsoft SQL Server 2000 database. However, the user can use this system with other databases and change the database specific functions as specified in this document.

Section 1 briefly introduces Pannasoft Ingenuity.

Section 2 highlights the features of Pannasoft Ingenuity.

Section 3 enlists and compares features available in various editions of Pannasoft Ingenuity.

Section 4 describes the steps user should take to prepare and normalize the data before running Pannasoft Ingenuity.

Section 5 provides full details on installation and usage of the Training and Prediction modules. Configuration of the system is described in detail here.

SECTION 2: FEATURES

Automated/Manual Parameters Selection

Pannasoft Ingenuity gives users the flexibility to automatically or manually adjust the network parameters for analyzing data. In auto mode, the efficient learning algorithm will automatically adjust network parameters to find a suitable architecture. In manual mode, the user is able to manually fine-tune the parameters.

Rule Extraction and Pruning Strategy

Pannasoft Ingenuity provides rule explanation capability to gain a wider degree of user acceptance on the results and to let user understand the potential and ability of the trained engine in handling classification problems. Pruning strategy enables user to remove low confidence and unimportant prototypes (knowledge) from the system to reduce complexity and network size.

Retrieve Old Knowledge

Pannasoft Ingenuity enables user to retrieve old prototypes from the database and combine them with new input samples to form new prototypes. Thus, the time used to retrain is significantly reduced.

Multiple Classifier System

Multiple Classifier Systems enable Pannasoft Ingenuity to produce better results and higher accuracy in decision making using a voting system mechanism.

Multiple Jobs

Pannasoft Ingenuity can accommodate different category of data sets (e.g. data on credit card spending, health diagnosis, manufacturing operation and etc) in one database.

Ease of Use

Pannasoft Ingenuity is developed with the highest attention to usability. The communication between Pannasoft Ingenuity and datasets is based on the database as well as from the results produced by the system.

Advantages of Pannasoft Ingenuity:

1. Capable of self-organizing to an arbitrary sequence of sample patterns in stationary as well as non-stationary environments.
2. Ability to continuously learn new patterns without losing previously learned information.
3. Revelation of embedded rule set in the network can be accomplished in linguistic format that users can clearly comprehend.
4. Multiple Classifier Systems enable Pannasoft Ingenuity to produce better results for higher accuracy in decision making.

Pannasoft Ingenuity overcomes the limitation of Artificial Neural Networks

Neural networks have been empirically confirmed to be useful in many applications. Nonetheless, most neural network models refused to tell what they have learned due to their distributed knowledge representation. Neural networks are regarded as a "black box" technology because of the lack of comprehensibility. In general, comprehensibility is one of the required characteristics of reliable systems. Therefore many works have been carried out to address the issue of improving the comprehensibility of neural networks. Rule extraction is a technique to overcome this problem. The merits of including rule extraction technique as a supplement to neural networks are as follows:

1. Provide an explanation component to neural networks
2. Overcome the problem of knowledge acquisition
3. Explore data and induce scientific theories
4. Improve the generalization of neural networks solutions

Pannasoft Ingenuity with AAA is a hybrid neural networks based system, integrated with the rule extraction technique where it has revealed promising characteristic of building a genuinely autonomous system. The revelation of embedded rule set in Pannasoft Ingenuity can be accomplished in the IF-THEN rule, which is straightforward and much more comprehensible.

SECTION 3: EDITIONS

The table below lists the various features that are available for different editions of Pannasoft Ingenuity.

Feature	Pannasoft Ingenuity Edition					
	Demo	Evaluation	Express	Lite	Standard	Professional
Number of Features	Per demo data	Unlimited	15	25	35	Unlimited
Multiple Classifier	✓	✓	X	✓	✓	✓
Retrieve Old Knowledge	✓	✓	X	✓	✓	✓
Automated Parameters Selection	✓	✓	X	X	✓	✓
Rule Extraction	✓	✓	X	X	X	✓
IP Address dependent	No		Yes			
License Key	Pre-packaged	Email upon request	Email upon purchase			
Training data query	Pre-defined	Configurable				

Note:
The Evaluation Edition has all the features of the Professional Edition but valid for 45 days only, starting from the date when Pannasoft Ingenuity's License file is created. The user is required to purchase the release edition after the evaluation license has expired to continue using Pannasoft Ingenuity.

SECTION 4: DATA PREPARATION AND NORMALIZATION

Pannasoft Ingenuity combines fuzzy logic and neural network technologies to create learning and predicting abilities. Therefore, data that is provided to Pannasoft Ingenuity must be a set of fuzzy numbers (between 0 and 1). Here, normalization process is performed with the database and Pannasoft Ingenuity will retrieve the normalized data from the database. The normalization process is to convert all unprocessed data into fuzzy numbers which are later used by Pannasoft Ingenuity.

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 Residence	nvarchar smallint	Other debtors / guarantors 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 (value 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. For this demonstration, refer to the "German" table. Column "checking" and "duration" are selected as features while "class" is selected as class.

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
```

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

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.

Trigger for automating data manipulation

A database object called trigger is require to automate record insertion into the "GCDATA" table when there is a new record created in the table raw data (referring to "German" table in this demonstration). User requires creating this mentioned 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 in above. The same activities need to apply to updating and deletion process in table raw data. Detail flag uses to indicate the insertion or updating or deletion process occurring in table raw data and need to note Pannasoft Ingenuity's reference data are described by item "TRFLAG" in Appendix A.

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

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 following the steps below. For example, "features" as the column's 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's 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 the column "checking" in the "German" table. The DESC_NAME contains explanation for the particular string in DESC_VAL. Please refer to Appendix B for more details about this 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 specify as 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
```

Result of this view:

	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 shown below:

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 shown below:

	ID	DESC_VAL	DESC_NAME	NV
	3	A13	>= 200 DM	0.66666666666666
	2	A12	0 DM <= ... < 200	0.33333333333333
	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 shown below:

	MAXID	MINID
	72	4
▶		

ii) Create a view to dsplay the minimum, maximum and distinct data 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].[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 shown below:

	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 $(0 \le x \le 1)$ for the DESC_VAL that represents a particular number using a script as shown below:

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	0.0147058823529411 7
	6	0.0294117647058823 5
	7	0.0441176470588235 2
	8	0.0588235294117647
	9	0.0735294117647058 8
	10	0.0882352941176470 5
	11	0.1029411764705882 3
	12	0.1176470588235294 1
	13	0.1323529411764705 8
	14	0.1470588235294117 6
	15	0.1617647058823529 4
	16	0.1764705882352941 1
	18	0.2058823529411764 7
	20	0.2352941176470588 2
	21	0.25
	22	0.2647058823529411 7
	24	0.2941176470588235 2
	26	0.3235294117647058 8
	27	0.3382352941176470 5
	28	0.3529411764705882 3
	30	0.3823529411764705 8
	33	0.4264705882352941 1
	36	0.4705882352941176 4

3) Normalization for target/class value

As stated above, Pannasoft Ingenuity is a supervised learning system that requires a class. The normalization procedure remains 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 can be obtained from Appendix C.

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

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 fuzzy number data.

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

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

- Please verify whether the raw data required by Pannasoft Ingenuity has been inserted into <Job_Code>DATA Reference Table (example: GCDATA or WBCDATA) via suggested database trigger.
- Please 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 (example NormGCCheckingData).

The result of this view is:

	REC_ID	class	checking	duration
▶	1	0	0	0.02941176470588
	2	1	0.33333333333333	0.64705882352941
	3	0	1	0.11764705882352
	4	0	0	0.55882352941176
	5	1	0	0.29411764705882
	6	0	1	0.47058823529411
	7	0	1	0.29411764705882
	8	0	0.33333333333333	0.47058823529411
	9	0	1	0.11764705882352
	10	1	0.33333333333333	0.38235294117647
	11	1	0.33333333333333	0.11764705882352
	12	1	0	0.64705882352941
	13	0	0.33333333333333	0.11764705882352
	14	1	0	0.29411764705882
	15	0	0	0.16176470588235
	16	1	0	0.29411764705882
	17	0	1	0.29411764705882
	18	0	0	0.38235294117647
	19	1	0.33333333333333	0.29411764705882
	20	0	1	0.29411764705882
	21	0	1	0.07352941176470
	22	0	0	0.02941176470588
	23	0	0	0.08823529411764
	24	0	0.33333333333333	0.11764705882352
	25	0	1	0.08823529411764
	26	0	0	0.02941176470588
	27	0	1	0.02941176470588

All the steps, from 1 to 4 have created view names with Pannasoft's suggested naming convention. You can locate your own naming convention for ease of use and for your integration purposes.

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 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 work 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 method is applied in this demonstration.

i) Same as the previous method. Users can by-pass the description table due to the nature of the target raw data that is not string/character and proceed 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.

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 D.

The result of this View is shown below:

	MAXID	MINID
	10	1
▶		

ii) Create a view to display the minimum, maximum and distinct data 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].[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 the DESC_VAL that represents a particular number using a script as shown below:

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 [dbo].[NormWBCNucleiData]
As
SELECT  NormWBCNucleiFilter.DESC_VAL,
CASE WHEN ((SELECT count(ID) FROM WBCancer WHERE Nuclei IS NULL)>0)
     THEN 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
     ELSE
         ((CAST(NormWBCNucleiFilter.DESC_VAL as Decimal(10,6))-
CAST(NormWBCNucleiFilter.minid as Decimal(106)))/
          (CAST(NormWBCNucleiFilter.maxid as Decimal(10,6))-
CAST(NormWBCNucleiFilter.minid as Decimal(10,6))))
     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

SECTION 5: USING PANNASOFT INGENUITY

Pannasoft Ingenuity consists of 2 modules:

1. Training module
2. Prediction module

Pannasoft Ingenuity is available in Java Application Programming Interface (API). The target machine need to pre-installed Java Development Kit (JDK) / Java Runtime Engine (JRE) version 1.4.2 and above. Pannasoft Ingenuity requires a RDBMS that able to create database object View and Trigger for storing knowledge.

The following description explains and attaches some sample codes to invoke both modules mentioned earlier.

Pannasoft Ingenuity API consists of the following files:

1. com.pns.extract-x.x.x.x.jar
2. pannasoft.properties
3. pannasoft.lcf

There is 1 file for Apache Log4J implementation:

1. log4j-x.x.x.jar

There are 3 files for Microsoft SQL 2000 JDBC:

1. msbase.jar
2. mssqlserver.jar
3. msutil.jar

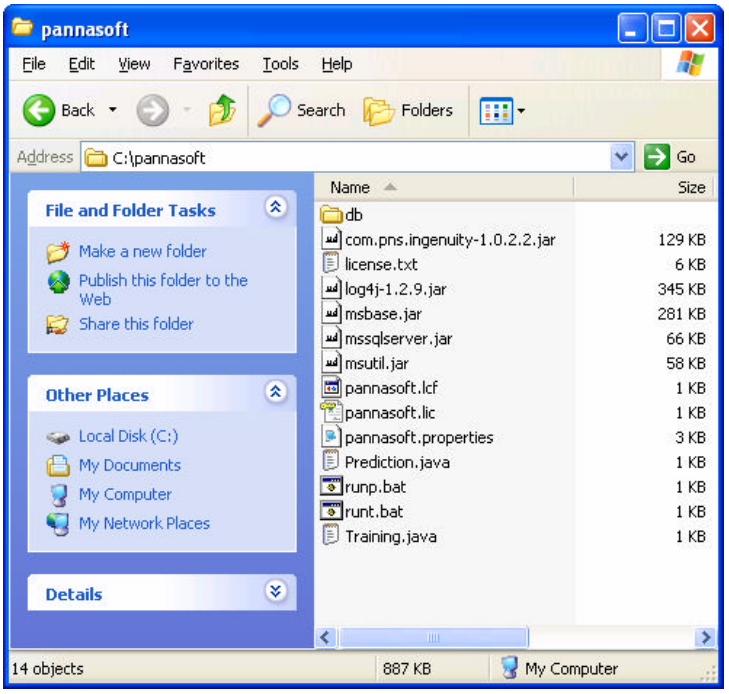
Sample Java application used to invoke Pannasoft Ingenuity:

1. To invoke Pannasoft Ingenuity's Training module: Training.java (binary file: Training.class)
2. To invoke Pannasoft Ingenuity's Prediction module: Prediction.java (binary file: Prediction.class)

Sample script used to compile and execute the sample Java application:

1. For Pannasoft Ingenuity's Training module: runt.bat
2. For Pannasoft Ingenuity's Prediction module: runp.bat

Extract all files into location C:\pannasoft as shown on the following page. To do a quick test, simply execute the batch files given to test both modules.



System Table

To create the required system tables, launch the given Pannasoft Ingenuity's database script and point ito your target database. If you are given database backup image[5], restore them according to your target database machine.

Training Module

After the normalization process, the user is ready to train Pannasoft Ingenuity to become an intelligent system by invoking the Training module. Pannasoft Ingenuity's Training module will utilize the normalized training data, i.e., View<Job_Code>NormTrainData, in the training process.

Pannasoft Ingenuity's Training module will check all TRFLAG in <Job_Code>DATA table (example table GCDATA) before performing any training process. For records that are needed for prediction, the user is required to set PRFLAG='Y' AND TRFLAG ='D' to avoid any training process on these records because prediction records contain no class (reference to column "class" in table "German") and Pannasoft Ingenuity's training module cannot process this type of records. Please refer to Appendix A for more details information about this "TRFLAG".

The user is able to configure network parameters to improve Pannasoft Ingenuity's accuracy in making prediction and decision support. The network parameters are discussed in the following section of this document.

[5] Refer to Appendix F for steps of creating and restoring Microsoft SQL 2000 database.

To ensure Pannasoft Ingenuity's Training module computes consistent knowledge, data manipulation (including the activities of inserting, updating and deleting processes) in raw data table (example "German" table) and Pannasoft Ingenuity's reference table (example "GCDATA" table) is prohibited during the execution of the training module. Changes in both tables' data will have a direct impact on the chances of all database View objects referred by Pannasoft Ingenuity's Training module. Data inconsistencies will occur and Pannasoft Ingenuity's knowledge will be directly affected as the reading on both tables is in progress.

Sample Code: Invoke Training module in Java

```
String szPropFile = "C:/pannasoft/pannasoft.properties";
com.pns.extract.Ingenuity myObj = com.pns.extract.Ingenuity.getInstance
(szPropFile);
myObj.doTraining ();
```

Note: Always remember to include the given jar file, i.e. file com.pns.extract-x.x.x.x.jar, into your classpath before compiling and executing Pannasoft Ingenuity modules.

In order to have trained record(s), Pannasoft Ingenuity's Training module will search all records with TRFLAG='A', 'U' or 'T' in Pannasoft Ingenuity's reference table (example "CRSDATA" table) for predicting activities. Pannasoft Ingenuity will always assume that these records contained data in column "class", table "German".

After the training activities have ended for all the record(s) mentioned, the TRFLAG will change to 'T' to indicate that they are trained by Pannasoft Ingenuity's Training module.

Pannasoft Ingenuity will always assume that those record(s) in column "TRFLAG" of Pannasoft Ingenuity's reference table (example table "GCDATA") with value 'D' or 'P' are not used for training activities. Details of this flag are explained in the Appendix A.

Prediction Module

The prediction module that will utilize the knowledge (prototypes) learned in the Training module. It is capable of predicting unknown patterns based on the features given. The number of features presented must be the same with the number of attributes in the knowledge. However, missing values are allowed for these attributes presented to Pannasoft Ingenuity's prediction modules.

All attributes need to be in fuzzy numbers. Refer to the previous normalization chapter to produce the required data before submitting to the Prediction module. Pannasoft Ingenuity's prediction module will utilize the normalized data, i.e., View<Job_Code>NormPredictData, in the predicting process.

Pannasoft Ingenuity's prediction module will check the PRFLAG in Pannasoft Ingenuity's reference table (example "CRSDATA" table) before making any prediction. User needs to set PRFLAG to 'Y' for those samples that are needed to make prediction and 'N' for those that are not ready for prediction. To perform the training process, Pannasoft Ingenuity's training module will select

those records that contain TRFLAG='A', TRFLAG='U' or TRFLAG='T'. Please refer to Appendix A for more details about this PRFLAG.

To retrieve the predicted results, the user can refer to column RESULT in Pannasoft Ingenuity's reference table (example "CRSDATA" table). After the prediction, PRFLAG is set to 'N' and prediction will not be redone in the future. The value in column RESULT is in normalized value. The database object view named Norm<Job_Code>ClassData (example view NormGCClassData) has the description value for the column RESULT.

For example, when raw data contains only 2 classes (good and bad customers) in the German Credit Data, the normalization process sets the good customers to 0 and bad customers to 1 (as defined in View named NormGCClassData). Then the column RESULT will show only 0 or 1 for the prediction. Column CONFIDEN indicates the confidence factor of the prediction. Value of 1.0 indicates full confidence while 0.0 indicates no confidence. This column will only contain value if the "ruleOn" flag in pannasoft.properties is set to "true", otherwise, value=-1000 will be given to indicate no confidence value. This predicted result can be obtained by using your primary reference key in column "ID" of "GCDATA" table (it should be the same as the data in column "ID" of "GCDATA" table) to retrieve Pannasoft Ingenuity's prediction result.

Sample Code: Invoke Prediction module in Java

```
String szPropFile = "c:/pannasoft/pannasoft.properties";
com.pns.extract.Ingenuity myObj = com.pns.extract.Ingenuity.getInstance
(szPropFile);
myObj.doPrediction ();
```

Note: Always remember to include the given jar file, i.e. file com.pns.extract-x.x.x.x.jar, into your classpath before compiling and executing Pannasoft Ingenuity modules.

In order to have predicted record(s), the records in Pannasoft Ingenuity's reference table (example "CRSDATA" table) need to have data PRFLAG='Y' and TRFLAG ='D'. Pannasoft Ingenuity's prediction module will search all records with PRFLAG='Y' in Pannasoft Ingenuity's reference table (example "CRSDATA" table) for predicting activities. Pannasoft Ingenuity will always assume record(s) in Pannasoft Ingenuity's reference table (example "CRSDATA" table) of column "PRFLAG" that has the value 'N' is not used for predicting activities.

Configuration in pannasoft.properties file

All configurations applied to both Pannasoft Ingenuity modules. The description for these network parameters is shown below.

The database configuration needed to be filled in order for Pannasoft Ingenuity to manipulate the target database data:

1. **db_authentication**: the database authentication approach. The possible values available are "server", "client".

2. **db_user_id**: the database authorized user accessing the target database.

3. **db_password**: the database authorized user password.

4. **db_url**: the JDBC driver dependent string allowing Pannasoft Ingenuity to access the target database, example jdbc:microsoft:sqlserver://192.168.1.1:1433;DatabaseName=ingenuity;SelectMethod=cursor

5. **db_driver**: the JDBC driver name in used. It is database vendor specific components, example com.microsoft.jdbc.sqlserver.SQLServerDriver.

6. **db_schema**: the database scheme for all target database objects.

The Apache Log4J configuration needed to be filled for logging purposes.

1. **log4j_lcf**: the Apache Log4J configuration (pannasoft.lcf) path

In Pannasoft Ingenuity, there are a few network parameters that the user might be interested in and request to modify the configuration file named pannasoft.properties.

1. **arta_beta:** this is a learning parameter that will decide at which rate the recoding of the pattern should occur in the learning process. Note: 0 (more resistance to noisy data) < arta_beta ≤ 1 (less resistance to noisy data). Default value is 1.0.

   Let's assume that we have a prototype (knowledge), **A**, in the system and a new input, **B**, that passed the constraint in the vigilance parameter. If the learning parameter is set to 1, 100% of the information/pattern from **B** is combined with the pattern in **A**. However, if the learning parameter is set to 0.1, only 10% of the information/pattern from **B** is combined with **A** and a new prototype, **C**, is formed in the system. If **B** contains noisy data* and the learning parameter is set to 0.1, then **B** won't have much influence on the existing prototype **A** (more resistance to noisy data from B) as 10% of the information/pattern from B is combined with 100% of the information/pattern from A.

   * noisy data = data that contains a lot of missing values in the features

2. **arta_rho/TestARTa_rho:** A vigilance parameter that determines the degree of granularity of the learned patterns. Low vigilance values (→ 0) lead to rough categories with broad generalization while high vigilance values (→ 1) lead to fine categories with narrow generalization. **Note:** 0 ≤ **Vigilance parameter** < 1. Default value is 0.0.

   For example, a user needs to gather a group of people according to their body size. Let's assume that there are 3 males and 2 females with big body, 2 males and 2 females with medium body and finally, 2 males and a female with small body. With vigilance parameter = 0.0, the system will probably gather all males in a group and all females in another group and form 2 patterns for 2 classes (male or female). Although both male and female body sizes are different, the system will probably form a pattern for one class (male or female) only. This is due to the vigilance parameter (constraint to pass the grouping) is set to 0. However, if the vigilance parameter is high, i.e., 0.9, the system will group together 3 males with big body in one group, 2 males with medium body in another group and 2 males with small body in the third group and this process will be repeated for grouping of the females. All the tests and calculations are based on the algorithm coded in the system.

3. **seed:** a number that controls the randomization for creating a training data and a testing data from raw data. Default value is 10.

   Let's assume that the raw data contains A1, A2, B1, A3, B2, B3 and the system needs to distribute a set of trained data and a set of test data. For example, when **seed** is set to 5, the system will take A1, A3, B2 and B3 as the trained set and the remaining as test set. If **seed** is set to 10, the system will take A1, B1, A2 and B3 as trained set while the remaining as test set. If the user feels that the previous trained set (A1, A3, B2 and B3) produced better results, then the user can set the **seed** back to 5 and the previous results will be produced. The 5 and 10 of the **seed** are various sets of randomized choice patterns.

   Raw Data - A1, A2, B1, A3, B2, B3

	randomized choice patterns	
Seed	**Trained Set**	**Test Set**
5	A1, A3, B2, B3	A2, B1
10	A1, B1, A2, B3	A3, B2
Etc		

4. **iNominator/iDenominator:** number of raw data that is distributed to become trained set. Default value = 2/3.

5. **noVoter:** number of base network in Pannasoft Ingenuity. Default value = 1.

6. **uthreshold:** A pruning threshold. Prototypes (knowledge) which are lower than this threshold will be deleted from the system to avoid complexity to form in the system. For example, when **uthreshold** = 0.5, the prototype with confidence level of < 0.5 will be deleted. The default value is 0.4.

7. **reload_weights:** true = enable network to retrieve old prototype from the database, false = network will re-train all the available data (TRFLAG = 'A' and TRFLAG = 'T').

8. **ruleOn:** true = enable rule extraction (convert the prototype from database to linguistic format); false = disable rule extraction.

9. **autoTune:** true = enable network to auto adjust suitable network parameters in (2) and (3), false = user needs to manually insert the network parameter of arta_beta and arta_rho.

10. **job:** multiple jobs in a database. Symbols comprising semi colon (;), comma (,) and plus (+) are reserved characters to differentiate between databases, tables/views and features.

Example:
german_data;GC DATA;ViewGCNormData;ViewGCNormTrainData;ViewGCNormPredictData;rec_id,class,checking,duration,history,purpose,credAmt,saving,employ,installment,status,debtors,residence,property,age,otherPlan,housing,existCred,job,liability,tel,foreignWorker+wbc_data;WBCDATA;ViewWBCNormData;ViewWBCNormTrainData;ViewWBCNormPredictData;rec_id,class,Clump,CellSize,CellShap,Adhesion,EpiCell,Nuclei,Chroma,Nucleoli,Mitoses

From the earlier example, german_data identifies the first job. GCDATA is a reference table for the first job. ViewGCNormData is a complete set of normalized data. ViewGCNormTrainData and ViewGCNormPredictData are the normalized data for training and prediction respectively, separated from ViewGCNormData by using the scripts below:

Script: ViewYyyNormTrainData View

(View<Job_Code>NormTrainData)

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

CREATE View [dbo].[ViewGCNormTrainData]
As
SELECT PatternData.*, GCDATA.TRFLAG
from ViewGCNormData PatternData inner join GCDATA on
GCDATA.REC_ID=PatternData.REC_ID
where GCDATA.TRFLAG='A' or GCDATA.TRFLAG='U' or GCDATA.TRFLAG='T'
GO
```

Script: ViewYyyNormPredictData View

(View<Job_Code>NormPredictData)

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

Create View [dbo].[ViewGCNormPredictData]
As
SELECT PatternData.*
from ViewGCNormData PatternData inner join GCDATA on
GCDATA.REC_ID=PatternData.REC_ID
where GCDATA.PRFLAG='Y'
GO
```

Configuration in pannasoft.lcf file

All configurations in pannasoft.lcf are originated from Apache[6]. To enable Pannasoft Ingenuity's logging, the following configuration is required to be added.

1. **log4j.appender.xxx.File**: the log file path for Pannasoft Ingenuity's logging.

[6] Details of Apache Log4J configuration documentation can be obtained at url http://logging.apache.org/log4j/docs/manual.html

Display Linguistic Rules

In this section, a demonstration on how to display a linguistic rule in Wisconsin Breast Cancer data is shown. Assuming the raw data description is as follows:

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 EpiCell	float float	Marginal Adhesion 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

After the training process is completed, Pannasoft Ingenuity will create a set of weights which can be translated into linguistic rules.

Use the script (View<Job_Code>Rule, example ViewWBCRule) as shown below to gather 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 "ViewWBCCF"). Finally, combine both views to produce a complete linguistic rule set with confidence factors as shown in View<Job_Code>RuleExtraction (example "VIewWBCRuleExtraction").

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].[ViewWBCRule]
GO

Create View [dbo].[ViewWBCRule]
As
select rule_id, att_id,
case when att_id=1 then 'Clump'
     when att_id=2 then 'CellSize'
     when att_id=3 then 'CellShape'
     when att_id=4 then 'Adhesion'
     when att_id=5 then 'EpiCell'
     when att_id=6 then 'Nuclei'
     when att_id=7 then 'Chroma'
     when att_id=8 then 'Nucleoli'
     when att_id=9 then 'Mitoses' end as features,
case when att_id=1 then (select CAST(desc_val as VARCHAR) from
NormWBCClumpData where nv=(select min(nv) from NormWBCClumpData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=2 then (select CAST(desc_val as VARCHAR) from
NormWBCCellSizeData where nv=(select min(nv) from NormWBCCellSizeData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=3 then (select CAST(desc_val as VARCHAR) from
NormWBCCellShapData where nv=(select min(nv) from NormWBCCellShapData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=4 then (select CAST(desc_val as VARCHAR) from
NormWBCAdhesionData where nv=(select min(nv) from NormWBCAdhesionData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=5 then (select CAST(desc_val as VARCHAR) from
NormWBCEpiCellData where nv=(select min(nv) from NormWBCEpiCellData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=6 then (select CAST(desc_val as VARCHAR) from
NormWBCNucleiData where nv=(select min(nv) from NormWBCNucleiData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=7 then (select CAST(desc_val as VARCHAR) from
NormWBCChromaData where nv=(select min(nv) from NormWBCChromaData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=8 then (select CAST(desc_val as VARCHAR) from
NormWBCNucleoliData where nv=(select min(nv) from NormWBCNucleoliData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=9 then (select CAST(desc_val as VARCHAR) from
NormWBCMitosesData where nv=(select min(nv) from NormWBCMitosesData where
nv>=LBOUND and nv<=UBOUND)) end as lbound,
case when att_id=1 then (select CAST(desc_val as VARCHAR) from
NormWBCClumpData where nv=(select max(nv) from NormWBCClumpData where
nv>=LBOUND and nv<=UBOUND))
```

```
     when att_id=2 then (select CAST(desc_val as VARCHAR) from
NormWBCCellSizeData where nv=(select max(nv) from NormWBCCellSizeData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=3 then (select CAST(desc_val as VARCHAR) from
NormWBCCellShapData where nv=(select max(nv) from NormWBCCellShapData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=4 then (select CAST(desc_val as VARCHAR) from
NormWBCAdhesionData where nv=(select max(nv) from NormWBCAdhesionData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=5 then (select CAST(desc_val as VARCHAR) from
NormWBCEpiCellData where nv=(select max(nv) from NormWBCEpiCellData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=6 then (select CAST(desc_val as VARCHAR) from
NormWBCNucleiData where nv=(select max(nv) from NormWBCNucleiData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=7 then (select CAST(desc_val as VARCHAR) from
NormWBCChromaData where nv=(select max(nv) from NormWBCChromaData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=8 then (select CAST(desc_val as VARCHAR) from
NormWBCNucleoliData where nv=(select max(nv) from NormWBCNucleoliData where
nv>=LBOUND and nv<=UBOUND))
     when att_id=9 then (select CAST(desc_val as VARCHAR) from
NormWBCMitosesData where nv=(select max(nv) from NormWBCMitosesData where
nv>=LBOUND and nv<=UBOUND)) end as ubound
from datarule where data_id='wbc_data' and voter_id=1
GO
```

Result of this view:

	rule_id	att_id	features	lbound	ubound
▶	1	1	Clump	1	7
	1	2	CellSize	1	3
	1	3	CellShape	1	4
	1	4	Adhesion	1	4
	1	5	EpiCell	2	7
	1	6	Nuclei	1	7
	1	7	Chroma	1	6
	1	8	Nucleoli	1	6
	1	9	Mitoses	1	5
	2	1	Clump	4	9
	2	2	CellSize	1	8
	2	3	CellShape	2	8
	2	4	Adhesion	4	9
	2	5	EpiCell	3	8
	2	6	Nuclei	7	10
	2	7	Chroma	5	8
	2	8	Nucleoli	4	9
	2	9	Mitoses	2	7
	3	1	Clump	6	9
	3	2	CellSize	2	7
	3	3	CellShape	4	8
	3	4	Adhesion	1	9
	3	5	EpiCell	3	7
	3	6	Nuclei	1	10
	3	7	Chroma	3	6
	3	8	Nucleoli	2	8
	3	9	Mitoses	1	3

Script: ViewYyyCF View

(View<Job_Code>CF)

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

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

Result of this view:

	CF_ID	CF_VAL	CLASS
▶	1	1	2
	2	0.82000005	4
	3	0.88	4
	4	0.83000004	4
	5	0.76000005	4
	6	0.70000005	4
	7	0.67	4
*			

Script: ViewYyyRuleExtraction View

(View<Job_Code>RuleExtraction)

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

CREATE View [dbo].[ViewWBCRuleExtraction]
As
select rule_id, att_id, features, lbound, ubound, cf_val, class
from ViewWBCRule inner join ViewWBCCF on ViewWBCRule.rule_id=VIewWBCCF.cf_id

GO
```

Result of this view:

rule_id	att_id	features	lbound	ubound	cf_val	class
1	1	Clump	1	7	1	2
1	2	CellSize	1	3	1	2
1	3	CellShape	1	4	1	2
1	4	Adhesion	1	4	1	2
1	5	EpiCell	2	7	1	2
1	6	Nuclei	1	7	1	2
1	7	Chroma	1	6	1	2
1	8	Nucleoli	1	6	1	2
1	9	Mitoses	1	5	1	2
2	1	Clump	4	9	0.82000005	4
2	2	CellSize	1	8	0.82000005	4
2	3	CellShape	2	8	0.82000005	4
2	4	Adhesion	4	9	0.82000005	4
2	5	EpiCell	3	8	0.82000005	4
2	6	Nuclei	7	10	0.82000005	4
2	7	Chroma	5	8	0.82000005	4
2	8	Nucleoli	4	9	0.82000005	4
2	9	Mitoses	2	7	0.82000005	4
3	1	Clump	6	9	0.88	4
3	2	CellSize	2	7	0.88	4
3	3	CellShape	4	8	0.88	4
3	4	Adhesion	1	9	0.88	4
3	5	EpiCell	3	7	0.88	4
3	6	Nuclei	1	10	0.88	4
3	7	Chroma	3	6	0.88	4
3	8	Nucleoli	2	8	0.88	4
3	9	Mitoses	1	3	0.88	4

Interpretation for the table above is as follows:

When rule_id = 1, 9 features are listed as below, with cf_val = 1 (value of confidence factor is 100%).

Feature Id	Feature Name	Lower bound value	Upper bound value	Class
1	Clump	1	7	2
2	CellSize	1	3	2
3	CellShape	1	4	2
4	Adhesion	1	4	2
5	EpiCell	5	7	2
6	Nuclei	1	7	2
7	Chroma	1	6	2
8	Nucleoli	1	6	2
9	Mitoses	1	5	2

Referring to the raw data description, the class = 2 represents the meaning of “Benign”.

Whereas the rule_id = 2, 9 features are listed as below, with cf_val = 0.82 (value of confidence factor is 82%).

Feature Id	Feature Name	Lower bound value	Upper bound value	Class
1	Clump	4	9	4
2	CellSize	1	8	4
3	CellShape	2	8	4
4	Adhesion	4	9	4
5	EpiCell	3	8	4
6	Nuclei	7	10	4
7	Chroma	5	8	4
8	Nucleoli	4	9	4
9	Mitoses	2	7	4

Refer to the raw data description, the class = 4 represents the meaning of "Malignant".

Browse through all the generated rule_id. The user can interpret the class value with raw data description. There are various confidence levels shown in View<Job_Code>RuleExtraction for different rule_id. From each rule (refer to the rule_id), the user is able to know the raw data pattern (via each feature's value, ranging between the lower and upper bound) of the class mentioned (for example, either "Benign" or "Malignant").

With this rule extraction flexibility, the user can utilize the interpreted rule from Pannasoft Ingenuity's prediction result and show the pattern of a particular class with specific value of the confidence level. It is an enhancement to Pannasoft Ingenuity's prediction process which explains the predicted result.

From the example, one can see that Pannasoft Ingenuity is able to determine the decision boundary of Benign and Malignant. Thus, it serves as a good tool in decision making and support system. From the table, it clearly shows that malignant cases contain a higher value for all features, as compared to benign cases. For instance, cell sharp and bare nuclei of benign cases contain value of 1-4 and 1-7 respectively while for malignant cases, they will give a high value of 2-8 for cell sharp and 7-10 for bare nuclei. From the extracted rules, clinicians can easily determine the benign and malignant cases in the future using these rule sets as reference.

APPENDIX A: DETAILS OF THE REFERENCE TABLE

Also known as <Job_Code>DATA (example GCDATA)

No	Column Name	Column Type	Nullable	Column Description	Sample Data and Prerequisite
1.	APPNO	varchar	No. No 0 to 999 values are allowed for this field.	Application / Agreement number; such as Applicant filled in form number, existing system generated running number or etc. This number is the table primary key or indexed data.	
2.	CRDATE	timestamp	No	Record creation timestamp.	Current timestamp (Default Value)
3.	TRFLAG	varchar	No	Training flag, it is the record status used for Pannasoft Ingenuity training.	**A**=Activate new record for training or re-training. **Example record**: historical/existing data in old system; transaction record completed (refer to all installment paid by customer). **U**= Activate existing record for training or re-training. **Example record**: existing record information is updated, and this flag will trigger Pannasoft Ingenuity to update Pannasoft Ingenuity repository. **D**=Deactivate record for training. **P**=Purged record, upon customer request, to avoid taking in this record for Pannasoft Ingenuity training.

					Pannasoft possible used flag: **T**= Pannasoft Ingenuity trained record. Pannasoft will turn this into T after the end of the training process.
4.	PRFLAG	varchar	No	Prediction flag, it is the record status used for Pannasoft Ingenuity prediction. Every new record in this table will trigger Pannasoft Ingenuity prediction in operation under batch mode. **Note**: Initial data (historical data from existing system) should use default value (N) in this column.	**Y**=Prediction yes flag; It is waiting for Pannasoft Ingenuity prediction result flag. This flag instructs Pannasoft Ingenuity to perform prediction. Pannasoft Ingenuity will only retrieve this kind of record for performing prediction. Pannasoft possible used flag: **N**=Prediction no flag; It is Pannasoft Ingenuity complete prediction result flag. This flag shows Pannasoft Ingenuity has finished performing prediction. (Default value)
5.	RESULT	float	No	Pannasoft Ingenuity prediction result. **Note**: Initial data (historical data from existing system) should use default value (N) in this column.	1000=Unknown/ no prediction has been made (Default value)
6.	MDDATE	timestamp	Yes	Record modification timestamp. Pannasoft Ingenuity will update this timestamp to trace the system response time.	NULL=(Default Value)

7.	CONFIDEN	double	Yes	Pannasoft Ingenuity's prediction confidence level. Pannasoft will update this value when the result is generated in column "result".	**Sample data:** 0.99, 0.8, 0.75 NULL=Column is not implemented yet (Default value)
8.	REC_ID	int	No	An auto increased reference number for Pannasoft Ingenuity to use. This auto number will auto increased when a new key ID (original ID from raw data) is inserted (from trigger) into column APPNO.	

Note:
This table needs to be created as the specified name for Pannasoft Ingenuity to work.

APPENDIX B: 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

Attibute 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 C: 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 D: EXTRAORDINARY MISSING DATA HANDLING

Missing data (a missing or null data value in a field) indicates the absence of any meaningful or invalid 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 figure below.

	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 figure above, 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 script below and insert an integer to represent a particular string in the 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 the missing or invalid data. Two types of "where" clause can be used to exclude the missing data by using the script below 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 a script as shown below:

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 produce the 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
```

Result of this view:

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 has been replaced by 0.

APPENDIX E: 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 below.

Use the script (View<Job_Code>Rule, example ViewGCRule) as shown below to gather 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
```

Result of this view:

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
```

Result of this view:

	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)

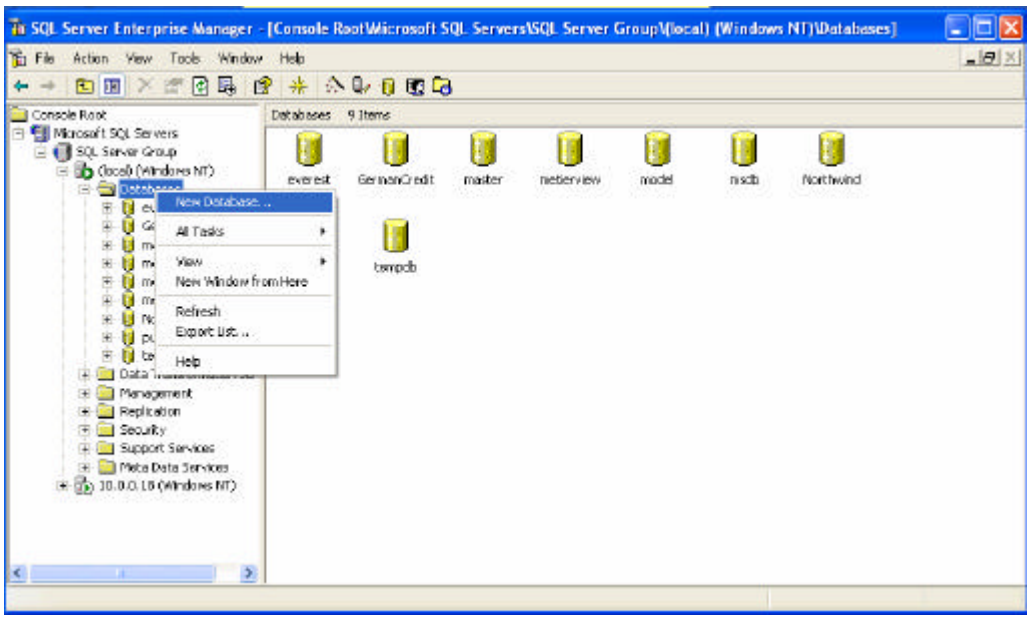
```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[ViewGCRuleExtraction]') and OBJECTPROPERTY(id, N'IsView')
= 1)
drop view [dbo].[ViewGCRuleExtraction]
GO

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
```

Result of this view:

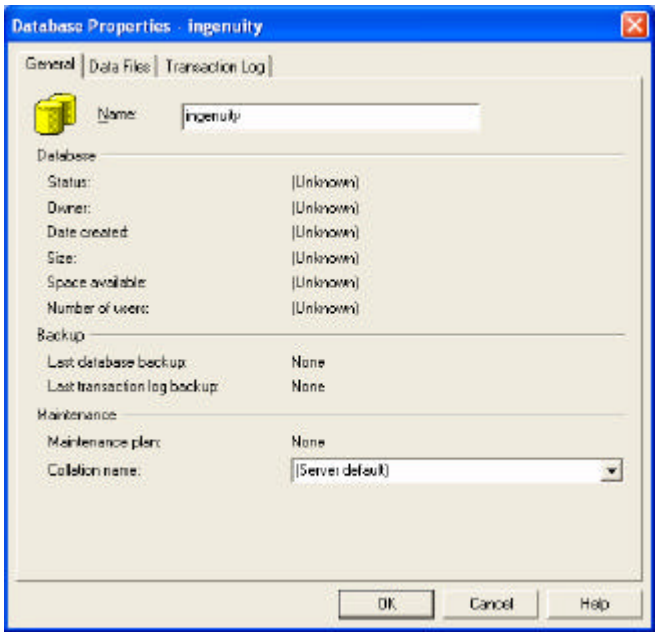
rule_id	att_id	features	lbound	ubound	cf_val	class
1	1	checking	A12	A14	0.8829268	1
1	2	duration	7	33	0.8829268	1
1	3	history	A32	A34	0.8829268	1
1	4	purpose	A40	A46	0.8829268	1
1	5	credAmt	484	5117	0.8829268	1
1	6	saving	A62	A64	0.8829268	1
1	7	employ	A72	A74	0.8829268	1
1	8	installment	2	4	0.8829268	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

APPENDIX F: CREATE & RESTORE MICROSOFT SQL 2000 DATABASE

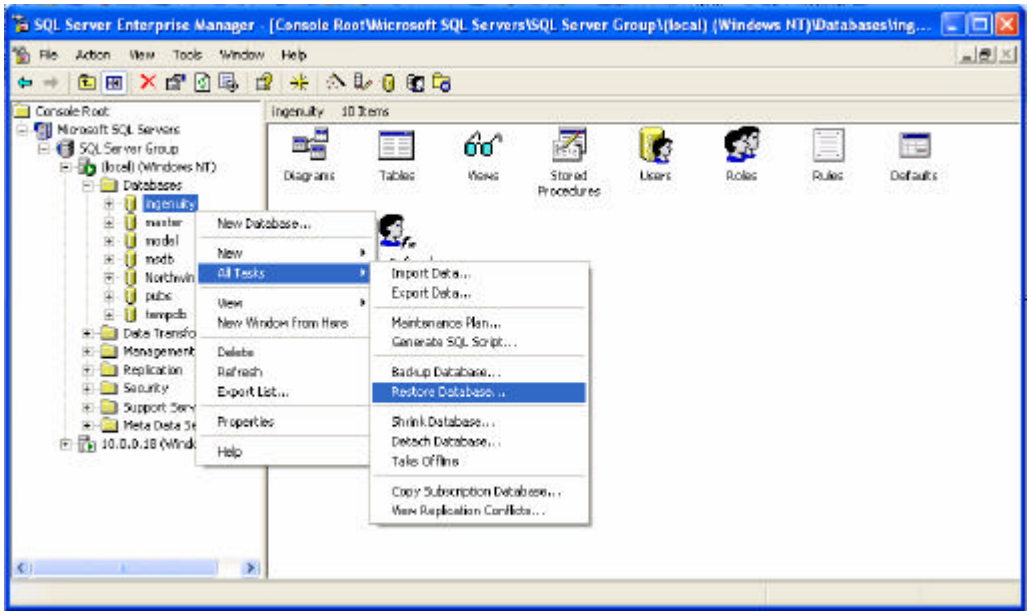
User can use SQL Server Enterprise manager to perform the mentioned steps.



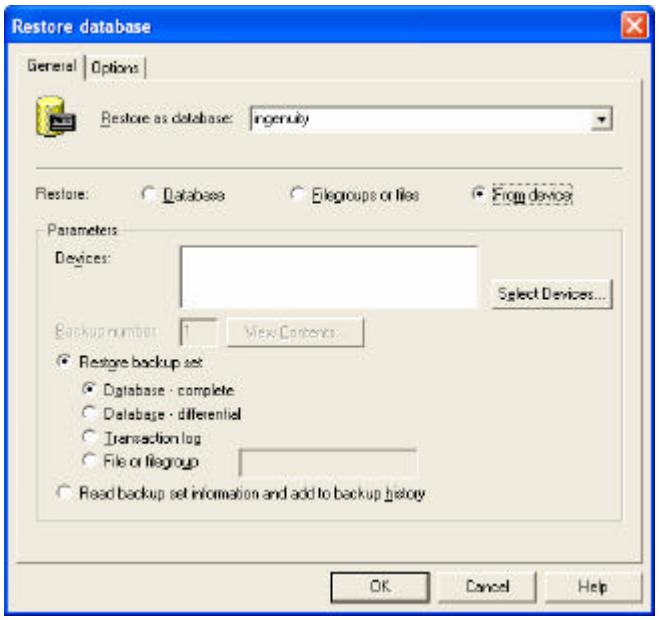
Right click on “Databases” and choose “New Database...” to create a new database.



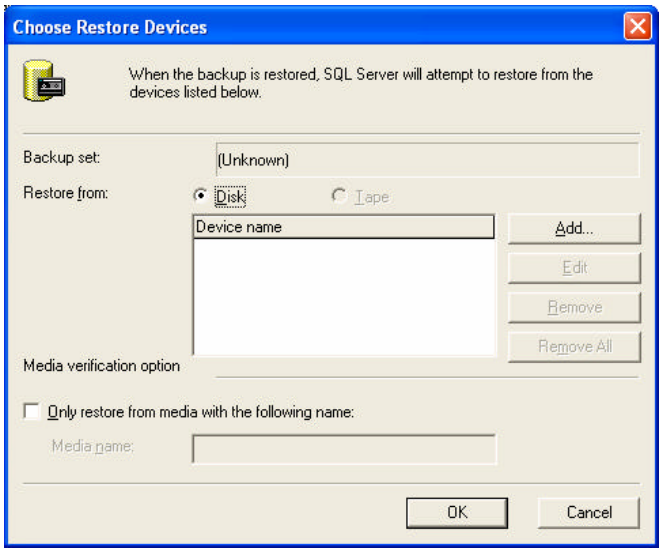
Name the new database as “ingenuity” as shown above.



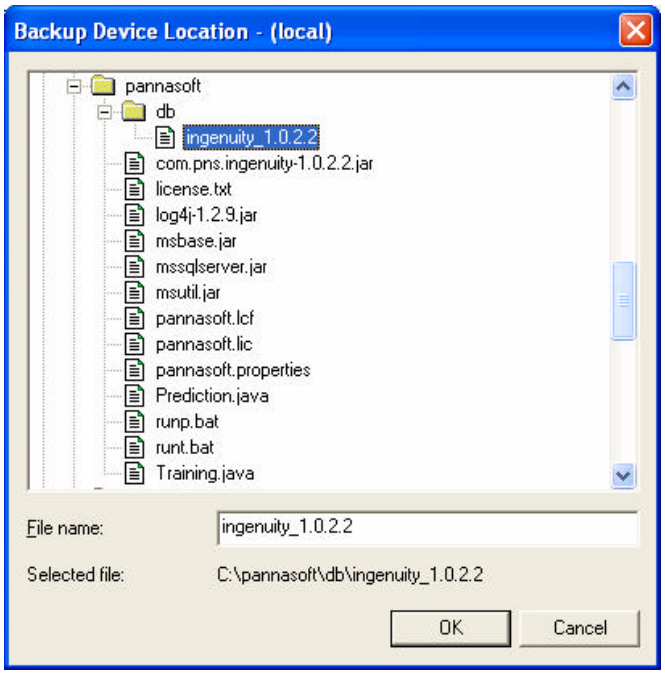
Right click on “ingenuity” to perform the backup step by choosing “Restore Database…”.



Choose the option “From device”, and click on the “Select devices…” button.

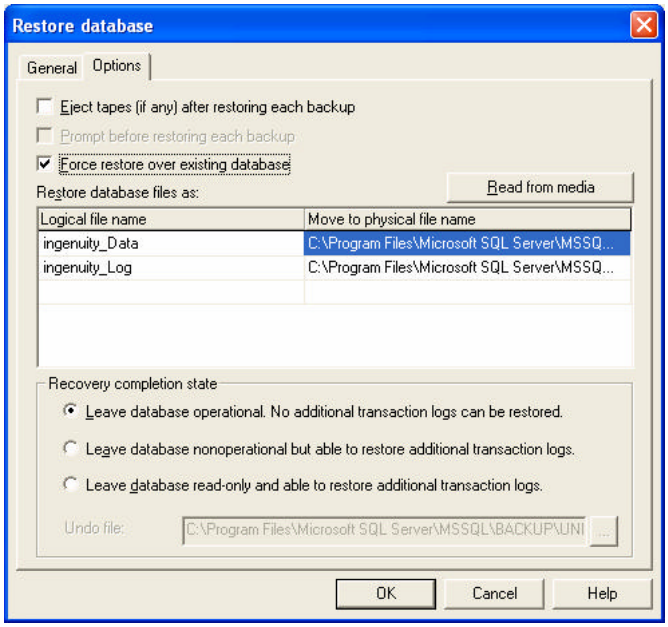


Click on the "Add..." button as shown on the screen above.

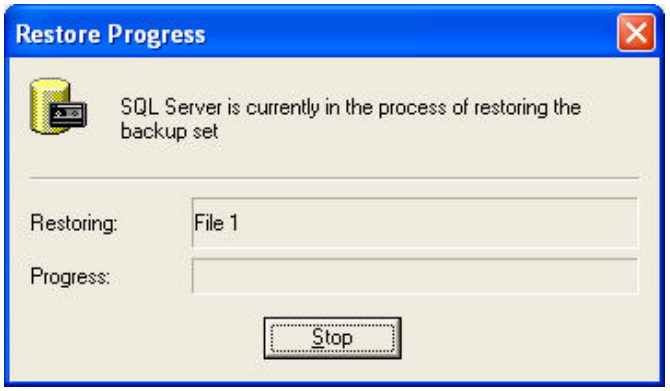


Browse to the location where the Microsoft SQL 2000 database backup file is residing and click on the "OK" button when done.

Click the "OK" button to proceed with the step.



Choose the option "Force restore over existing database" to force the backup process into the target database server.



The screen above shows the restoration of the “ingenuity” database is in progress.

When the “ingenuity” database is completely restored, a notification is displayed as shown on the screen above.

GLOSSARY

Data mart: It is a repository of data gathered from operational data and other sources. It is designed to serve a particular community of knowledge workers

Features or attributes: The characteristics of the data

Classes or targets: Categories of a data, e.g., commercial airplanes have large body, loading is high (number of people it carries) but fly slower while jet has smaller body, loading is low but fly faster. Here, class is referred to commercial airplane and jet while size, loading and speed represent features of a particular class.

Normalization: A process to standardize the data

Fuzzy numbers: Existing numbers between 0 and 1, e.g., 0.125, 0.0026

Supervised learning system A system that has a learning process on data which is labeled by output targets

Weight: It is also called prototype or Pannasoft Ingenuity's knowledge.

Linguistic rule: Statements that are understood by human, i.e., IF A happens, THEN B will fail where IF-THEN is a linguistic rule.

Confidence factor: Confidence level of a rule provided by Pannasoft Ingenuity. The higher the value of the confidence factor, the higher the confidence of this rule in decision making

Missing data: Missing or null data value in a field. It indicates the absence of any meaningful or invalid information in the features.

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*