WEKA User Manual

Contents

WEKA
Introduction…………………………………………………………………………… 3
- Background information………………………………………………………………3

Installation……………………………………………………………………………… 3
- Where to get WEKA…………………………………………………………………... 3
- Downloading Information……………………………………………………………...3

Opening the program………………………………………………………………….. 4
- Chooser Menu…………………………………………………………………………. 4-6

Preprocessing………………………………………………………………………….. 6-7
Classify………………………………………………………………………………….. 7-8
Cluster…………………………………………………………………………………… 8
Associate………………………………………………………………………………… 9
Select Attributes………………………………………………………………………... 9-10
Visualization……………………………………………………………………………. 11-13

Microsoft SQL Database
Relational Database………………………………………………………………….. 15
Procedures to access and use Database………………………………………….. 15
Loading flat files into relational database………………………………………….. 15-16
- DTS…………………………………………………………………………………... 16-18
- Table Relationships…………………………………………………………………. 18-19
- Creating Views………………………………………………………………………. 19-20

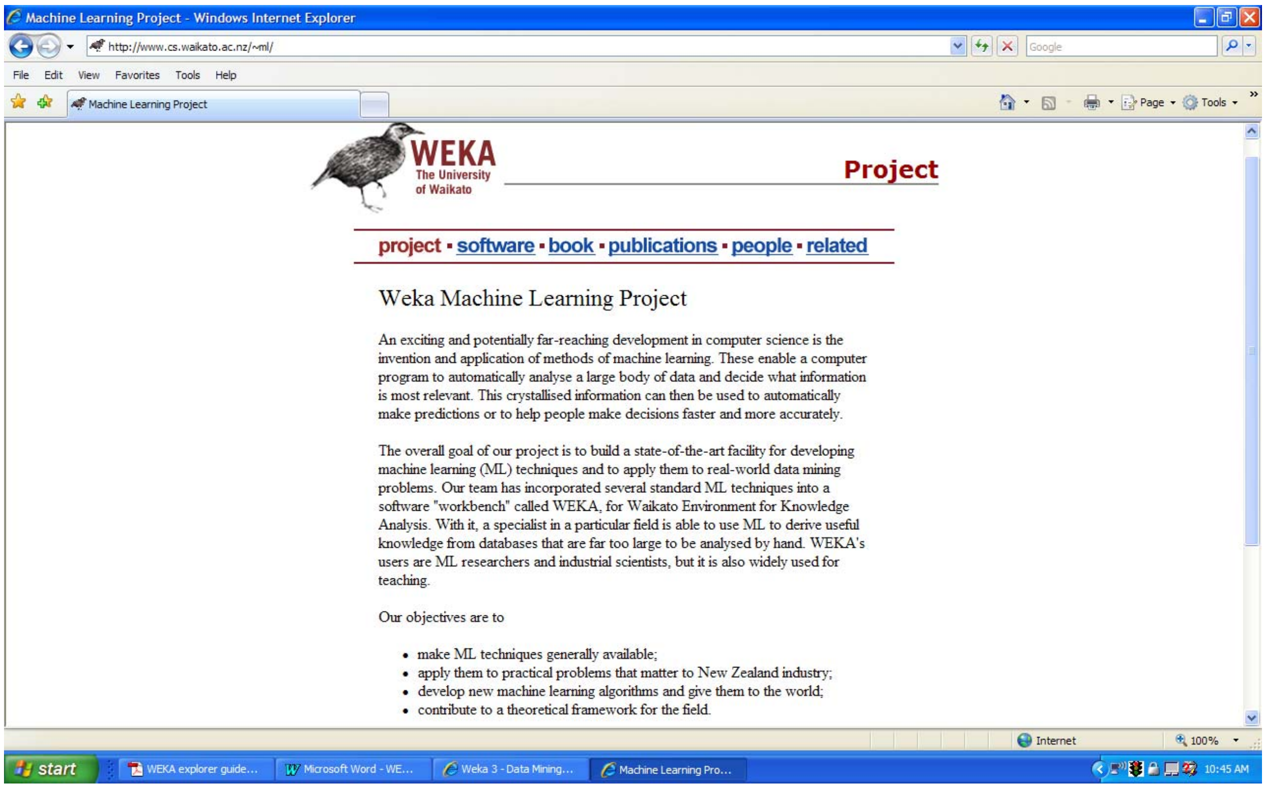
Queries to produce results……………………………………………………………. 20-21
- Temp Table example………………………………………………………………… 21-22
- Volume Data…………………………………………………………………………. 22

Introduction

WEKA, formally called Waikato Environment for Knowledge Learning, is a computer program that was developed at the University of Waikato in New Zealand for the purpose of identifying information from raw data gathered from agricultural domains. WEKA supports many different standard data mining tasks such as data preprocessing, classification, clustering, regression, visualization and feature selection. The basic premise of the application is to utilize a computer application that can be trained to perform machine learning capabilities and derive useful information in the form of trends and patterns. WEKA is an open source application that is freely available under the GNU general public license agreement. Originally written in C the WEKA application has been completely rewritten in Java and is compatible with almost every computing platform. It is user friendly with a graphical interface that allows for quick set up and operation. WEKA operates on the predication that the user data is available as a flat file or relation, this means that each data object is described by a fixed number of attributes that usually are of a specific type, normal alpha-numeric or numeric values. The WEKA application allows novice users a tool to identify hidden information from database and file systems with simple to use options and visual interfaces.

Installation

The program information can be found by conducting a search on the Web for WEKA Data Mining or going directly to the site at www.cs.waikato.ac.nz/~ml/WEKA . The site has a very large amount of useful information on the program's benefits and background. New users might find some benefit from investigating the user manual for the program. The main WEKA site has links to this information as well as past experiments for new users to refine the potential uses that might be of particular interest to them. When prepared to download the software it is best to select the latest application from the selection offered on the site. The format for downloading the application is offered in a self installation package and is a simple procedure that provides the complete program on the end users machine that is ready to use when extracted.


Machine Learning Project - Windows Internet Explorer
http://www.cs.waikato.ac.nz/~ml/
File Edit View Favorites Tools Help
Machine Learning Project
WEKA
The University of Waikato
Project
project • software • book • publications • people • related
Internet
100%
start
WEKA explorer guide...
Microsoft Word - WE...
Weka 3 - Data Mining...
Machine Learning Pro...
10:45 AM

Weka Machine Learning Project

An exciting and potentially far-reaching development in computer science is the invention and application of methods of machine learning. These enable a computer program to automatically analyse a large body of data and decide what information is most relevant. This crystallised information can then be used to automatically make predictions or to help people make decisions faster and more accurately.

The overall goal of our project is to build a state-of-the-art facility for developing machine learning (ML) techniques and to apply them to real-world data mining problems. Our team has incorporated several standard ML techniques into a software "workbench" called WEKA, for Waikato Environment for Knowledge Analysis. With it, a specialist in a particular field is able to use ML to derive useful knowledge from databases that are far too large to be analysed by hand. WEKA's users are ML researchers and industrial scientists, but it is also widely used for teaching.

Our objectives are to

- make ML techniques generally available;
- apply them to practical problems that matter to New Zealand industry;
- develop new machine learning algorithms and give them to the world;
- contribute to a theoretical framework for the field.

Opening the program

Once the program has been loaded on the user's machine it is opened by navigating to the programs start option and that will depend on the user's operating system. Figure 1 is an example of the initial opening screen on a computer with Windows XP.

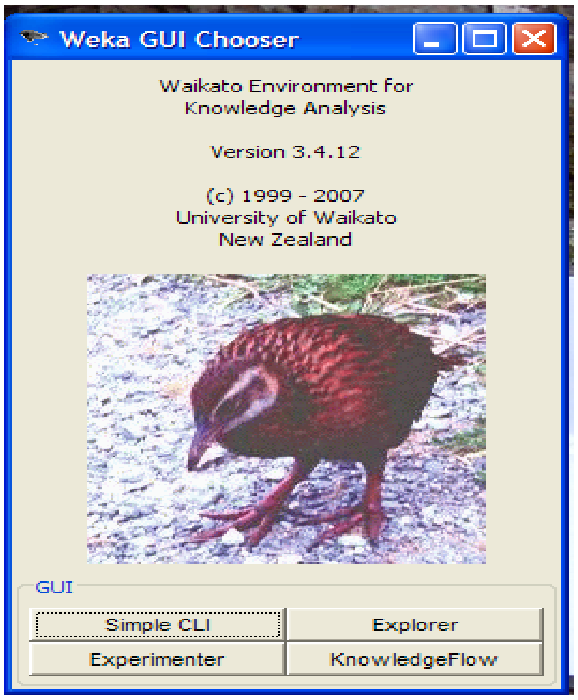


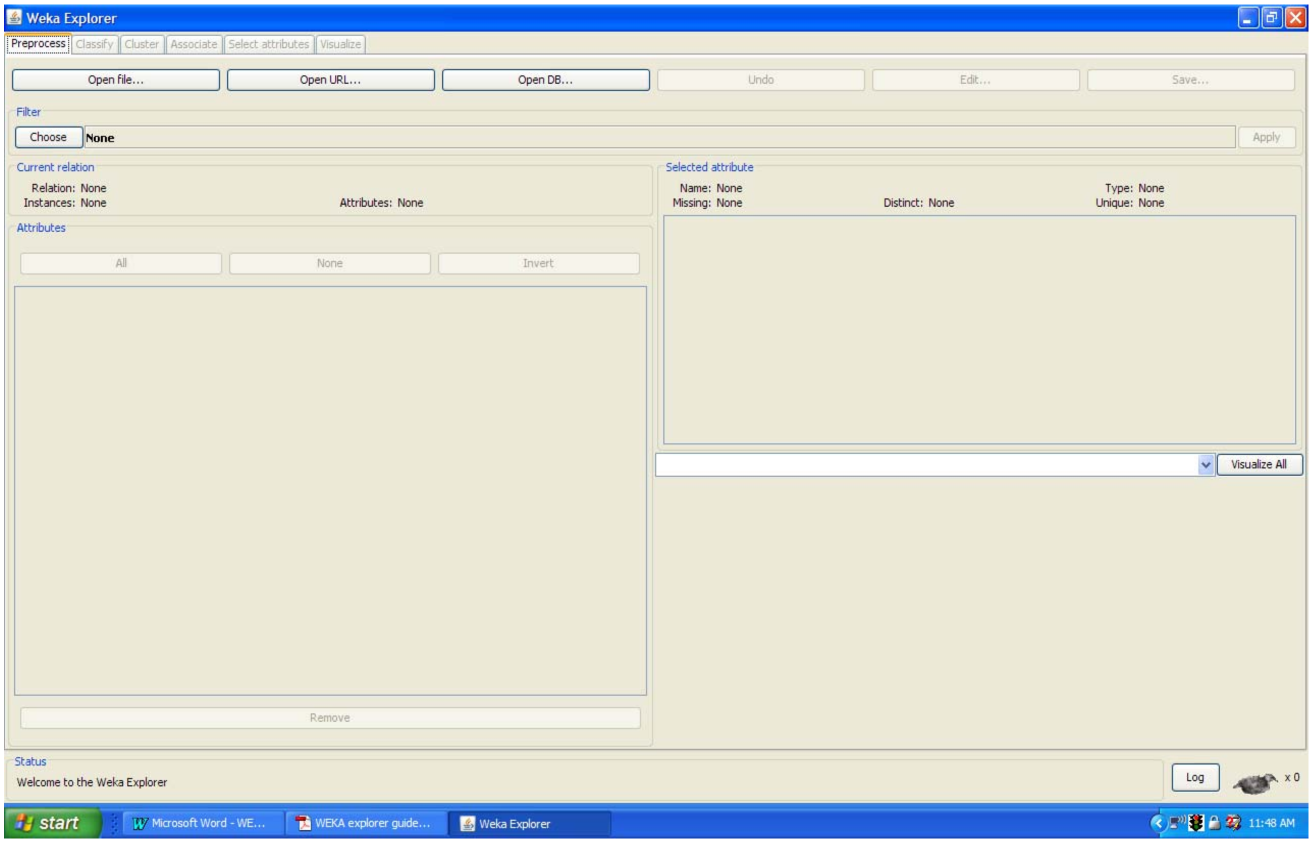
Figure 1 Chooser screen

There are four options available on this initial screen.

- Simple CLI- provides users without a graphic interface option the ability to execute commands from a terminal window.
- Explorer- the graphical interface used to conduct experimentation on raw data
- Experimenter- this option allows users to conduct different experimental variations on data sets and perform statistical manipulation
- Knowledge Flow-basically the same functionality as Explorer with drag and drop functionality. The advantage of this option is that it supports incremental learning from previous results

While the options available can be useful for different applications the remaining focus of the user manual will be on the Experimenter option through the rest of the user guide.

After selecting the Experimenter option the program starts and provides the user with a separate graphical interface.



Filter: None

Relation: None
Instances: None
Attributes: None

Name: None
Missing: None
Distinct: None
Type: None
Unique: None

Welcome to the Weka Explorer

Figure 2

Figure 2 shows the opening screen with the available options. At first there is only the option to select the Preprocess tab in the top left corner. This is due to the necessity to present the data set to the application so it can be manipulated. After the data has been preprocessed the other tabs become active for use.

There are six tabs:

1. Preprocess- used to choose the data file to be used by the application

2. Classify- used to test and train different learning schemes on the preprocessed data file under experimentation

3. Cluster- used to apply different tools that identify clusters within the data file

4. Association- used to apply different rules to the data file that identify association within the data

5. Select attributes-used to apply different rules to reveal changes based on selected attributes inclusion or exclusion from the experiment

6. Visualize- used to see what the various manipulation produced on the data set in a 2D format, in scatter plot and bar graph output

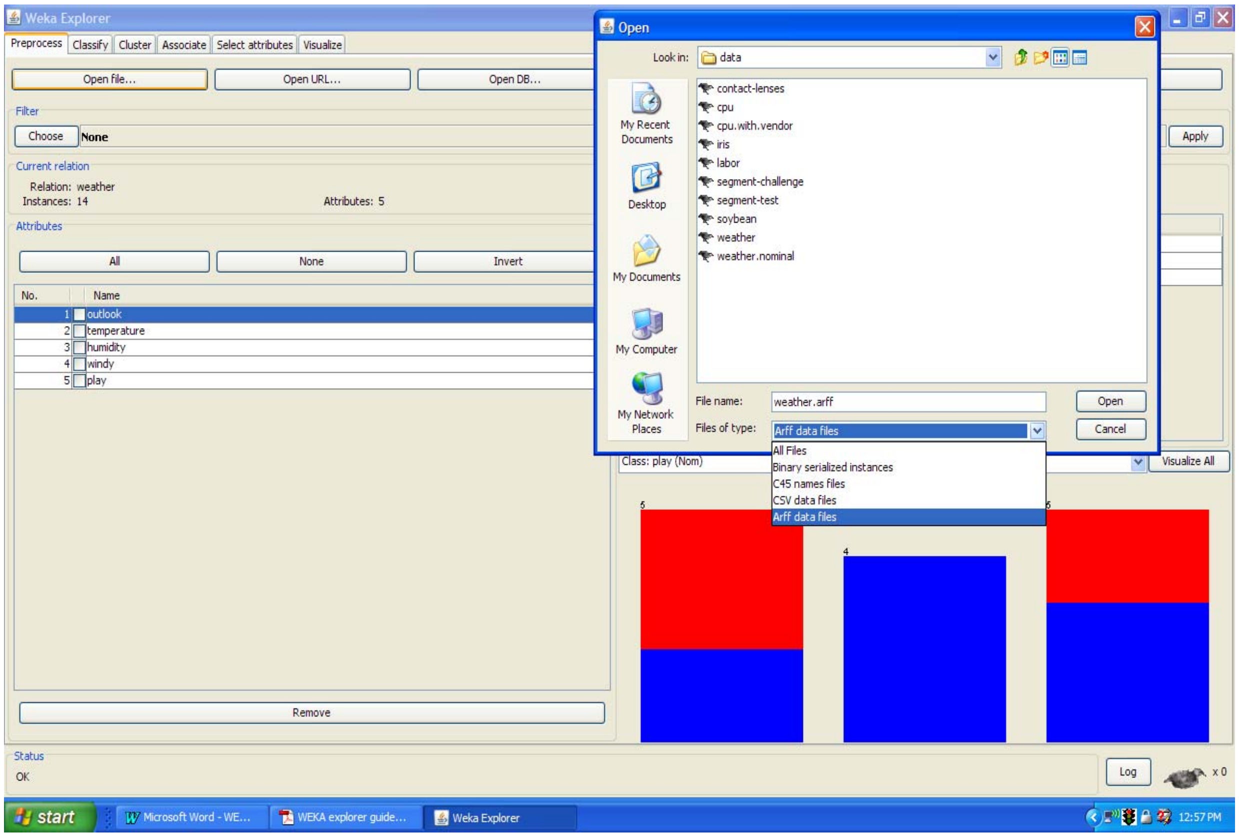
Once the initial preprocessing of the data set has been completed the user can move between the tab options to perform changes to the experiment and view the results in real time. This provides the benefit of having the ability to move from one option to the next so that when a condition becomes exposed it can be placed in a different environment to be visually changed instantaneously.

Preprocessing

In order to experiment with the application the data set needs to be presented to WEKA in a format that the program understands. There are rules for the type of data that WEKA will accept. There are three options for presenting data into the program.

- Open File- allows for the user to select files residing on the local machine or recorded medium
- Open URL- provides a mechanism to locate a file or data source from a different location specified by the user
- Open Database- allows the user to retrieve files or data from a database source provided by the user

There are restrictions on the type of data that can be accepted into the program. Originally the software was designed to import only ARFF files, newer versions allow different file types such as CSV, C4.5 and serialized instance formats. The extensions for these files include .csv, .arff, .names, .bsi and .data. Figure 3 shows an example of selection of the file weather.arff.



Relation: weather
Instances: 14
Attributes: 5

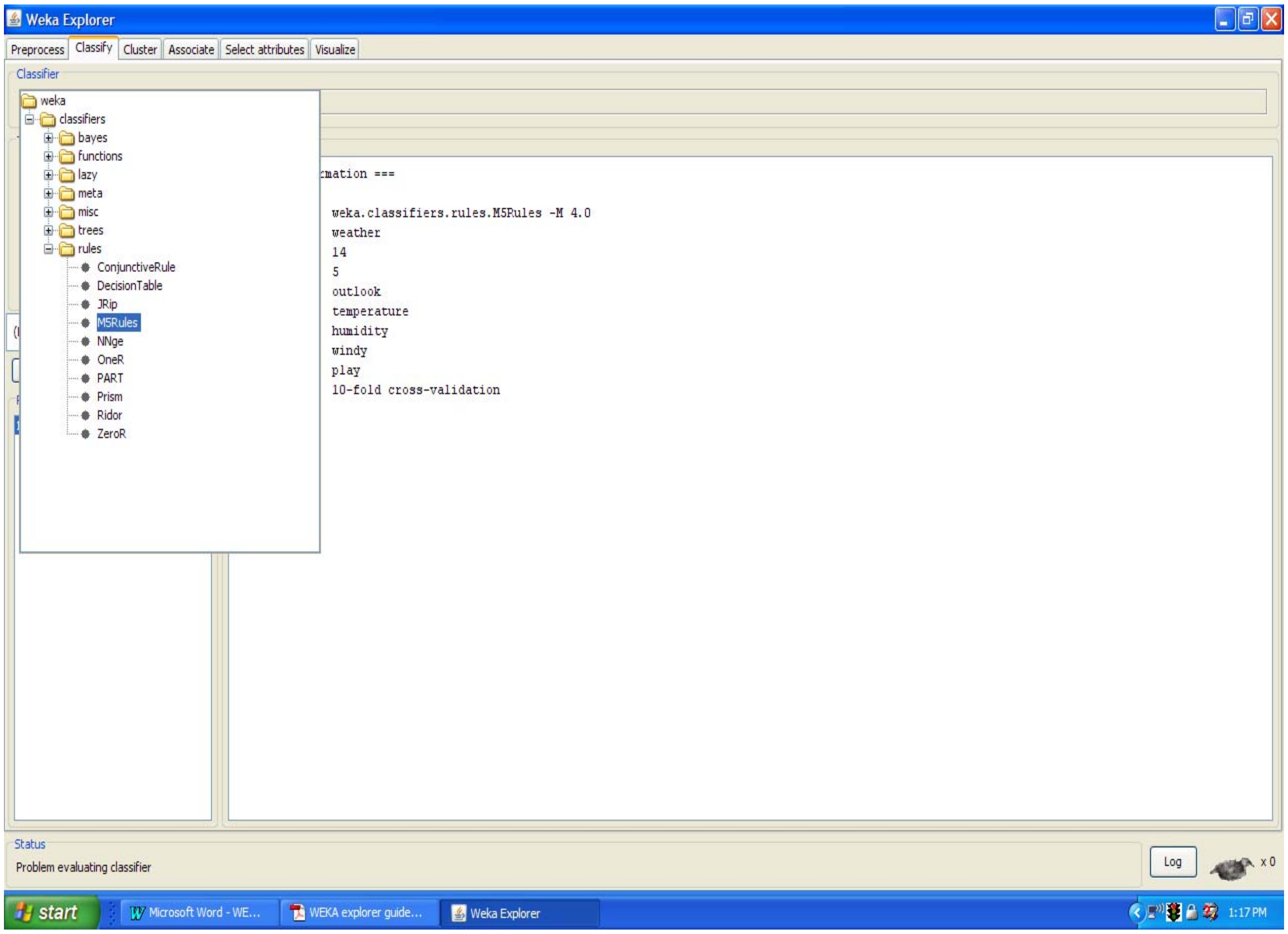
File name: weather.arff
Files of type: Arff data files

Figure 3

Once the initial data has been selected and loaded the user can select options for refining the experimental data. The options in the preprocess window include selection of optional filters to apply and the user can select or remove different attributes of the data set as necessary to identify specific information. The ability to pick from the available attributes allows users to separate different parts of the data set for clarity in the experimentation. The user can modify the attribute selection and change the relationship among the different attributes by deselecting different choices from the original data set. There are many different filtering options available within the preprocessing window and the user can select the different options based on need and type of data present.

Classify

The user has the option of applying many different algorithms to the data set that would in theory produce a representation of the information used to make observation easier. It is difficult to identify which of the options would provide the best output for the experiment. The best approach is to independently apply a mixture of the available choices and see what yields something close to the desired results. The Classify tab is where the user selects the classifier choices. Figure 4 shows some of the categories.



```
rmation ===

 weka.classifiers.rules.M5Rules -M 4.0
 weather
 14
 5
 outlook
 temperature
 humidity
 windy
 play
 10-fold cross-validation
```

Problem evaluating classifier

Figure 4

Again there are several options to be selected inside of the classify tab. Test option gives the user the choice of using four different test mode scenarios on the data set:

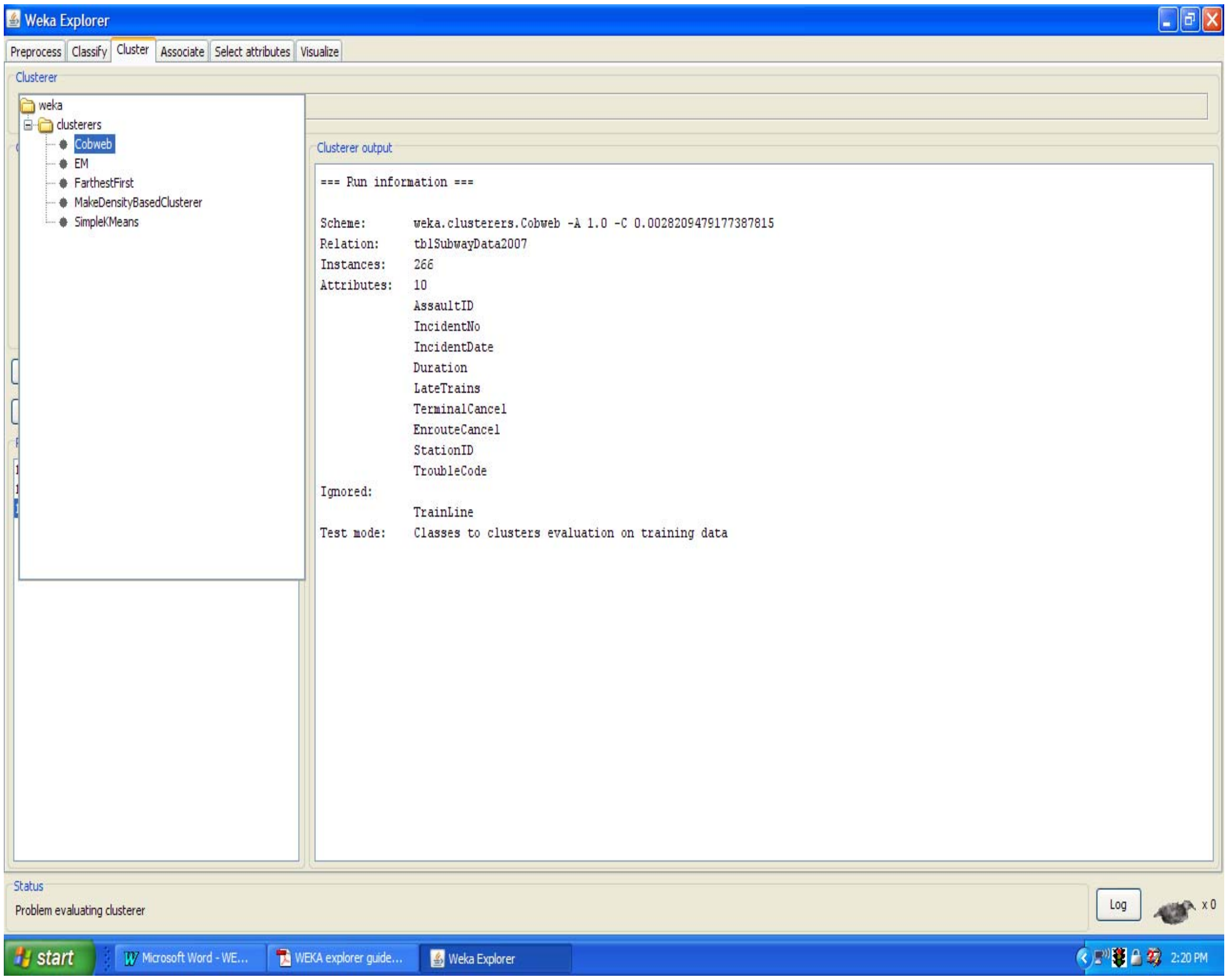
1. Use training set
2. Supplied training set
3. Cross validation
4. Split percentage

There is the option of applying any or all of the modes to produce results that can be compared by the user. Additionally inside the test options toolbox there is a dropdown menu so the user can select various items to apply that depending on the choice can provide output options such as saving the results to file or specifying the random seed value to be applied for the classification.

The classifiers in WEKA have been developed to train the data set to produce output that has been classified based on the characteristics of the last attribute in the data set. For a specific attribute to be used the option must be selected by the user in the options menu before testing is performed. Finally the results have been calculated and they are shown in the text box on the lower right. They can be saved in a file and later retrieved for comparison at a later time or viewed within the window after changes and different results have been derived.

Cluster

The Cluster tab opens the process that is used to identify commonalties or clusters of occurrences within the data set and produce information for the user to analyze. There are a few options within the cluster window that are similar to those described in the classifier tab. They are use training set, supplied test set, percentage split. The fourth option is classes to cluster evaluation, which compares how well the data compares with a pre-assigned class within the data. While in cluster mode users have the option of ignoring some of the attributes from the data set. This can be useful if there are specific attributes causing the results to be out of range or for large data sets. Figure 5 shows the Cluster window and some of its options.



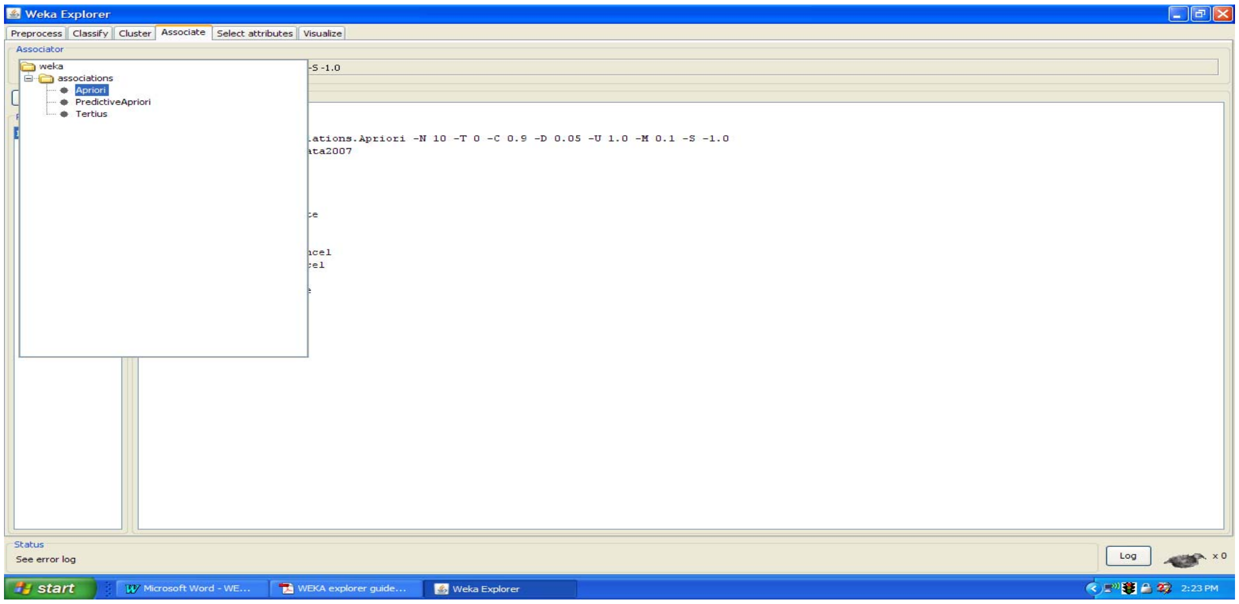
```
=== Run information ===

Scheme:       weka.clusterers.Cobweb -A 1.0 -C 0.0028209479177387815
Relation:     tblSubwayData2007
Instances:    266
Attributes:   10
              AssaultID
              IncidentNo
              IncidentDate
              Duration
              LateTrains
              TerminalCancel
              EnrouteCancel
              StationID
              TroubleCode
Ignored:
              TrainLine
Test mode:    Classes to clusters evaluation on training data
```

Problem evaluating clusterer

Figure 5

Associate

The associate tab opens a window to select the options for associations within the data set. The user selects one of the choices and presses start to yield the results. There are few options for this window and they are shown in Figure 6 below.

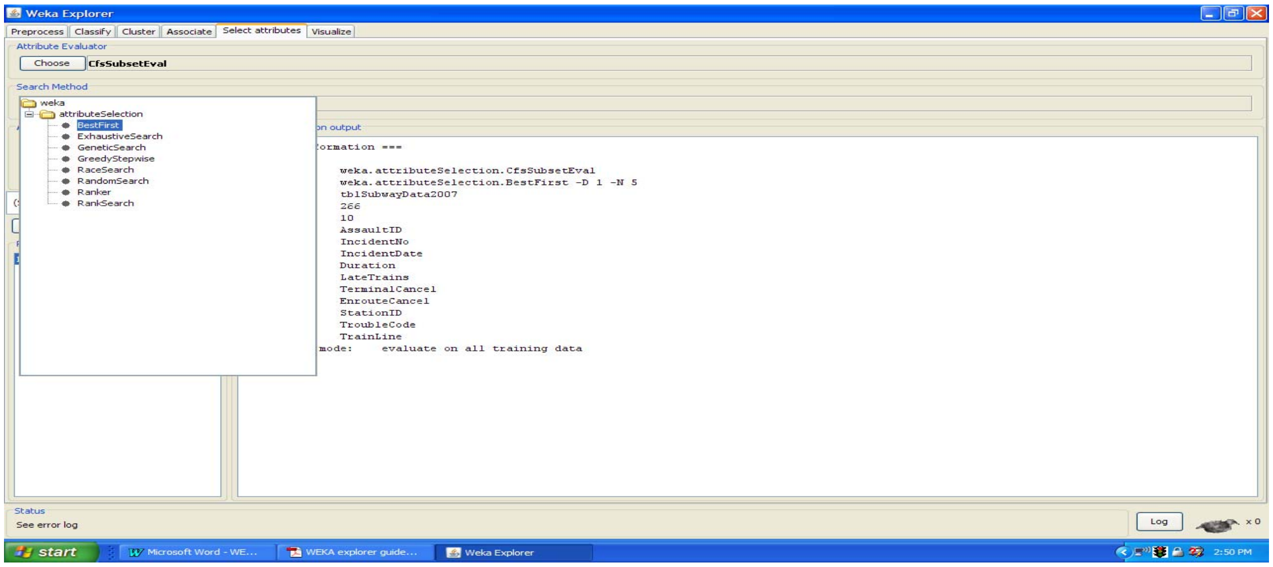


```
-S -1.0

.ations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0
.ta2007
```

Figure 6

Select Attributes

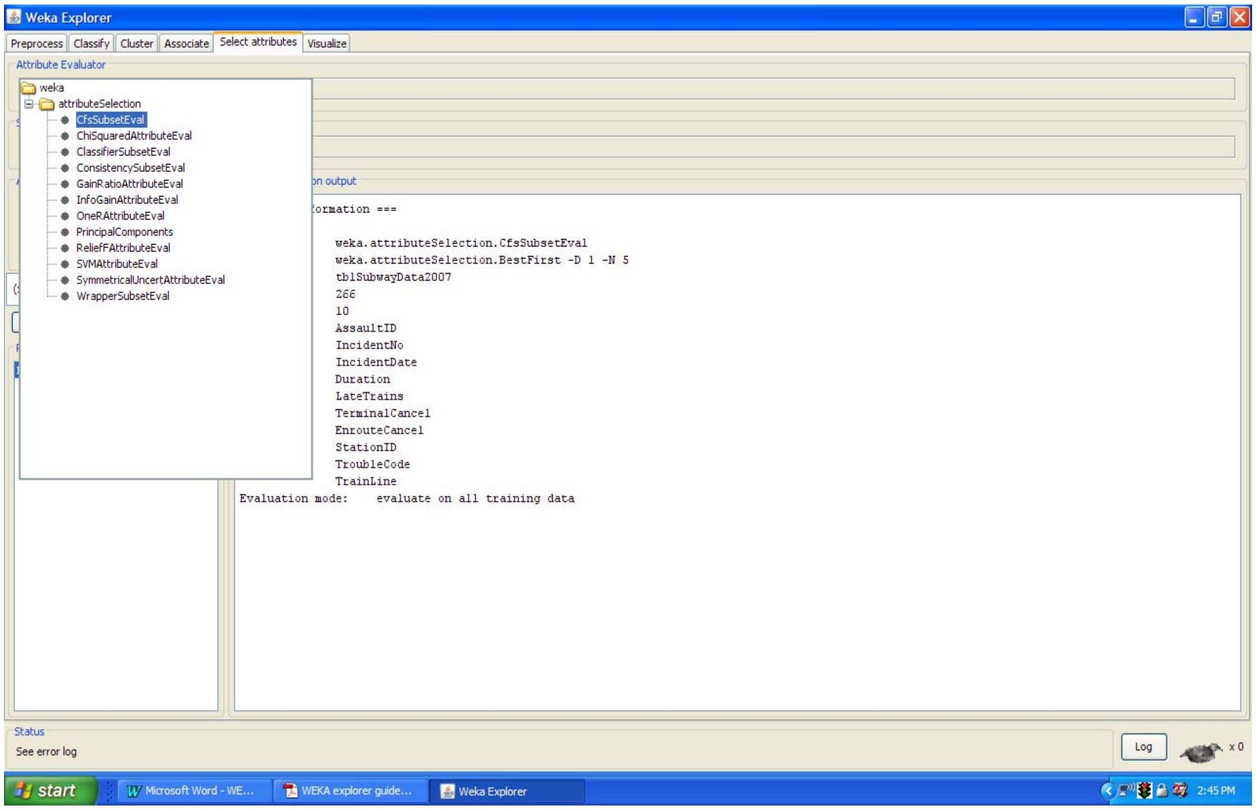
The next tab is used to select the specific attributes used for the calculation process. By default all of the available attributes are used in the evaluation of the data set. If the use wanted to exclude certain categories of the data they would deselect those specific choices from the list in the cluster window. This is useful if some of the attributes are of a different form such as alphanumeric data that could alter the results. The software searches through the selected attributes to decide which of them will best fit the desired calculation. To perform this, the user has to select two options, an attribute evaluator and a search method. Once this is done the program evaluates the data based on the sub set of the attributes then performs the necessary search for commonality with the date. Figure 7 shows the opinions of attribute evaluation. Figure 8 shows the options for the search method.



Attribute Evaluator: CfsSubsetEval

```
on output
formation ===

    weka.attributeSelection.CfsSubsetEval
    weka.attributeSelection.BestFirst -D 1 -N 5
    tblSubwayData2007
    266
    10
    AssaultID
    IncidentNo
    IncidentDate
    Duration
    LateTrains
    TerminalCancel
    EnrouteCancel
    StationID
    TroubleCode
    TrainLine
mode:    evaluate on all training data
```

Figure 7

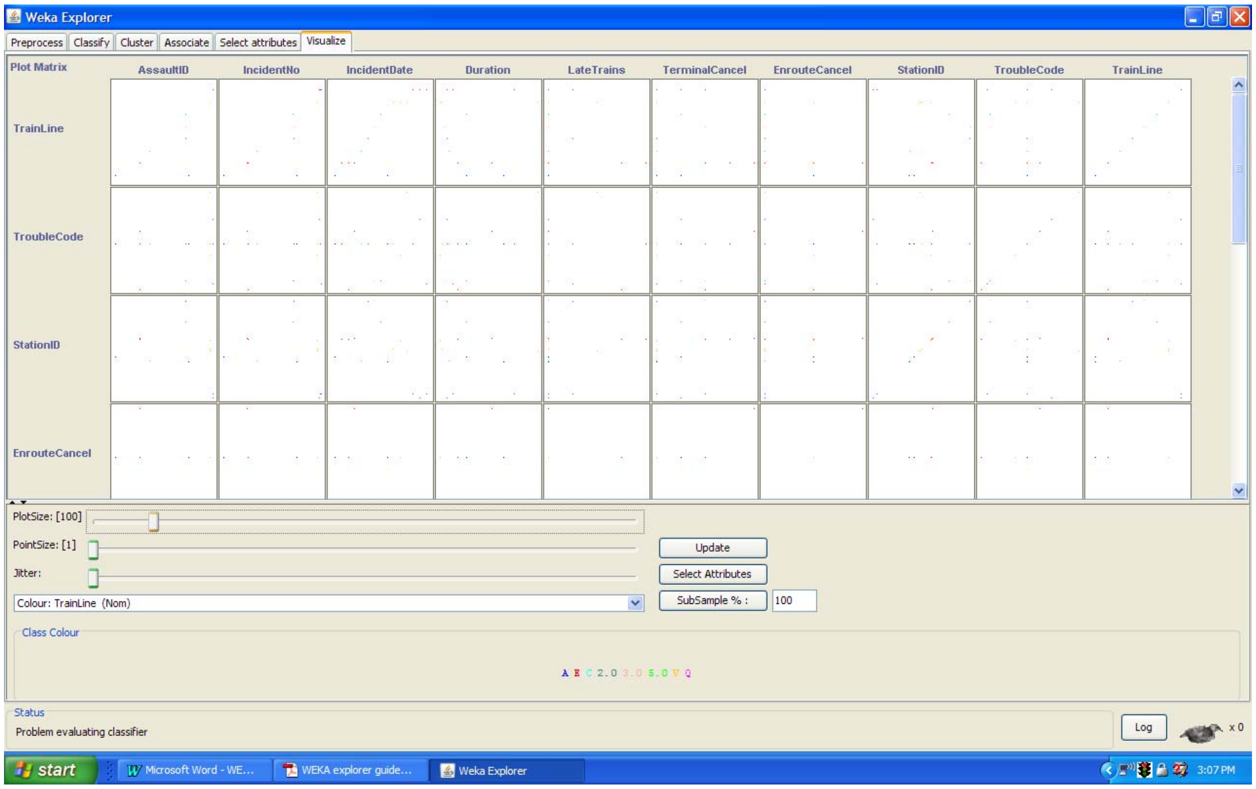


```
ormation ===

          weka.attributeSelection.CfsSubsetEval
          weka.attributeSelection.BestFirst -D 1 -N 5
          tblSubwayData2007
          266
          10
          AssaultID
          IncidentNo
          IncidentDate
          Duration
          LateTrains
          TerminalCancel
          EnrouteCancel
          StationID
          TroubleCode
          TrainLine
Evaluation mode:    evaluate on all training data
```

Figure 8

Visualization

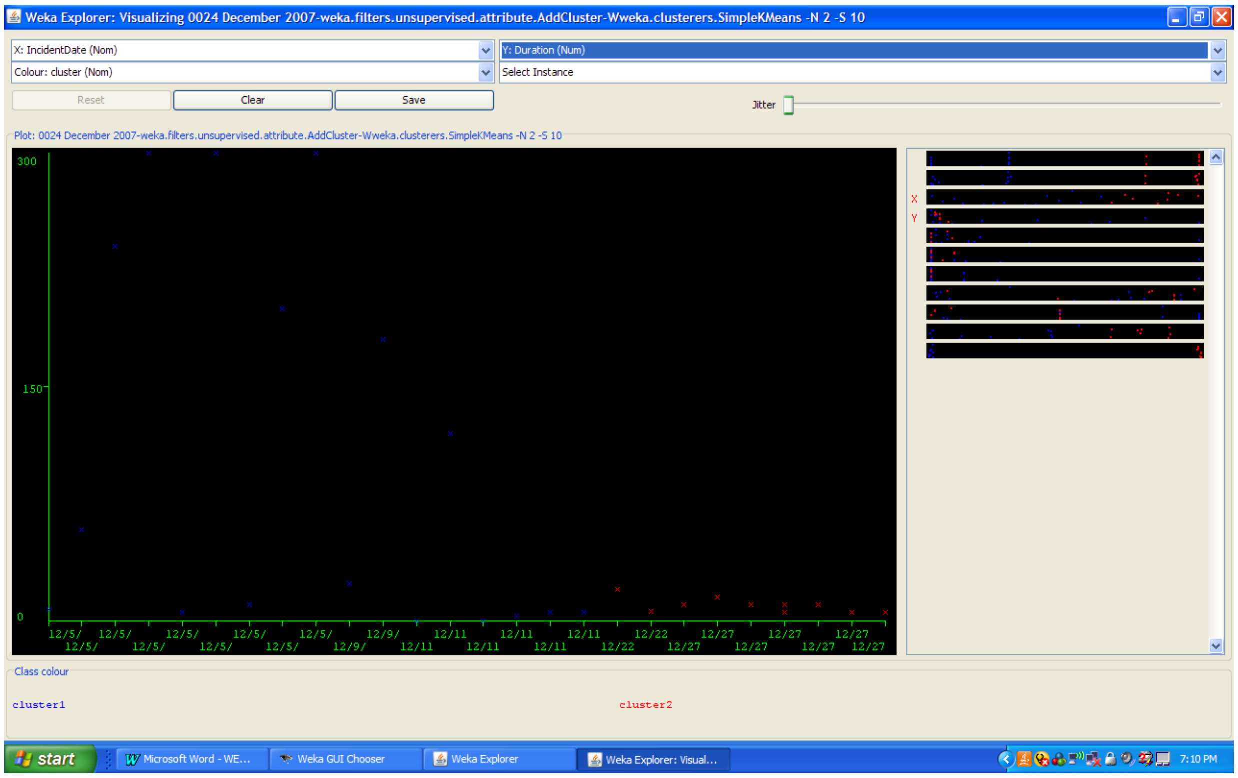
The last tab in the window is the visualization tab. Within the program calculations and comparisons have occurred on the data set. Selections of attributes and methods of manipulation have been chosen. The final piece of the puzzle is looking at the information that has been derived throughout the process. The user can now actually see the fruit of their efforts in a two dimensional representation of the information. The first screen that the user sees when they select the visualization option is a matrix of plots representing the different attributes within the data set plotted against the other attributes. If necessary there is a scroll bar to view all of the produced plots. The user can select a specific plot from the matrix to view its contents for analyzation. A grid pattern of the plots allows the user to select the attribute positioning to their liking and for better understanding. Once a specific plot has been selected the user can change the attributes from one view to another providing flexibility. Figure 9 shows the plot matrix view.



Problem evaluating classifier

Figure 9

The scatter plot matrix gives the user a visual representation of the manipulated data sets for selection and analysis. The choices are the attributes across the top and the same from top to bottom giving the user easy access to pick the area of interest. Clicking on a plot brings up a separate window of the selected scatter plot. The user can then look at a visualization of the data of the attributes selected and select areas of the scatter plot with a selection window or by clicking on the points within the plot to identify the point's specific information. Figure 10 shows the scatter plot for two attributes and the points derived from the data set. There are a few options to view the plot that could be helpful to the user. It is formatted similar to an X/Y graph yet it can show any of the attribute classes that appear on the main scatter plot matrix. This is handy when the scale of the attribute is unable to be ascertained in one axis over the other. Within the plot the points can be adjusted by utilizing a feature called jitter. This option moves the individual points so that in the event of close data points users can reveal hidden multiple occurrences within the initial plot. Figure 11 shows an example of this point selection and the results the user sees.

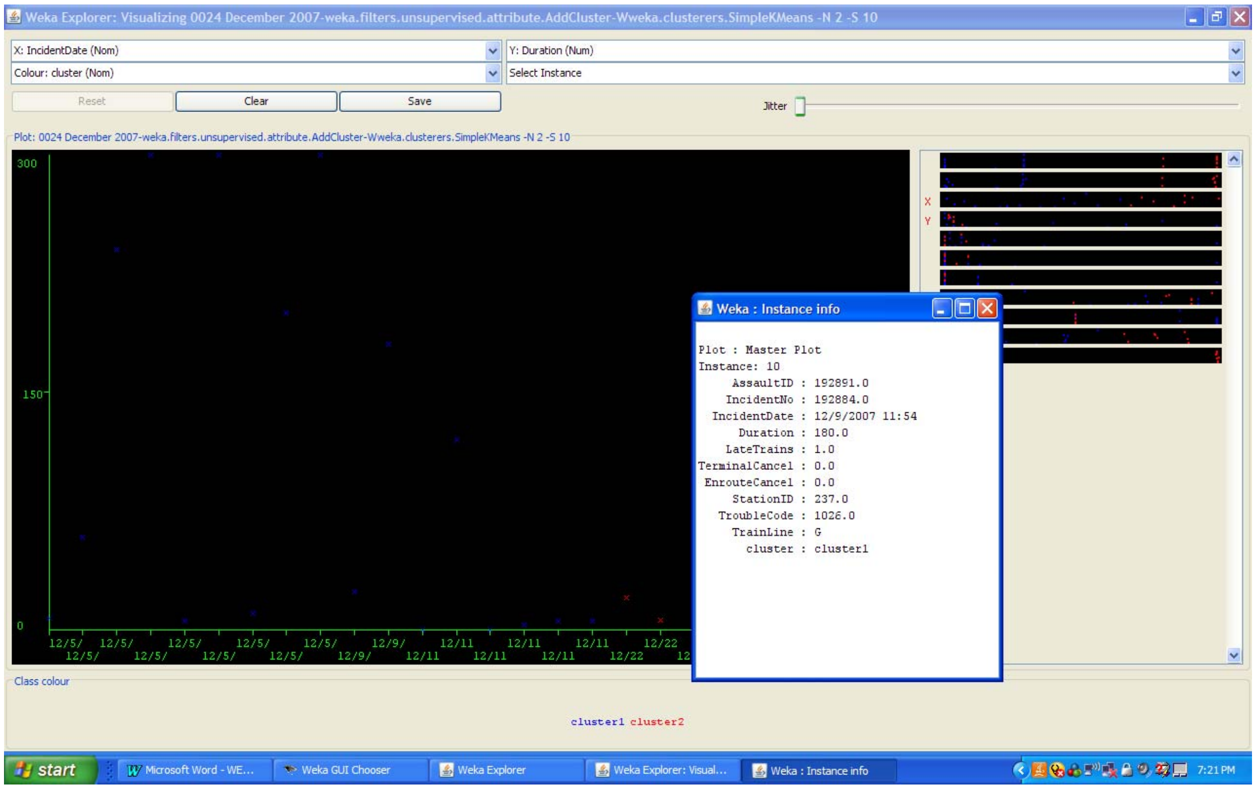


Weka Explorer: Visualizing 0024 December 2007-weka.filters.unsupervised.attribute.AddCluster-Wweka.clusterers.SimpleKMeans -N 2 -S 10

X: IncidentDate (Nom)
Y: Duration (Num)
Colour: cluster (Nom)
Select Instance

Plot: 0024 December 2007-weka.filters.unsupervised.attribute.AddCluster-Wweka.clusterers.SimpleKMeans -N 2 -S 10

Figure 10



Weka Explorer: Visualizing 0024 December 2007-weka.filters.unsupervised.attribute.AddCluster-Wweka.clusterers.SimpleKMeans -N 2 -S 10

X: IncidentDate (Nom)
Y: Duration (Num)
Colour: cluster (Nom)
Select Instance

Plot: 0024 December 2007-weka.filters.unsupervised.attribute.AddCluster-Wweka.clusterers.SimpleKMeans -N 2 -S 10

Weka : Instance info

```
Plot : Master Plot
Instance: 10
     AssaultID : 192891.0
    IncidentNo : 192884.0
  IncidentDate : 12/9/2007 11:54
      Duration : 180.0
    LateTrains : 1.0
TerminalCancel : 0.0
 EnrouteCancel : 0.0
     StationID : 237.0
   TroubleCode : 1026.0
     TrainLine : G
       cluster : cluster1
```

Figure 11

There are a few options to manipulate the view for the identification of subsets or to separate the data points on the plot.

- ♦ Polyline---can be used to segment different values for additional visualization clarity on the plot. This is useful when there are many data points represented on the graph.

- Rectangle--this tool is helpful to select instances within the graph for copying or clarification.
- Polygon—Users can connect points to segregate information and isolate points for reference.

This user guide is meant to assist users in their efforts to become familiar with some of the features within the Explorer portion of the WEKA data mining software application and is used for informational purposes only. It is a summary of the user information found on the programs main web site. For a more comprehensive and in depth version users can visit the main site http://www.cs.waikato.ac.nz/~ml/WEKA for examples and FAQ’s about the program.

Microsoft SQL Server User Manual

Relational Database:

Our team decided to load the provided data into a Microsoft SQL Server database. Mr. Washington gave us five text files, four of which are tab delimited files of New York Transit Authority data.

List of Files:
Data field descriptions – A list of 11 column headers for 2006 and 2007 data
2006 incident data – 11 column list of incident data for year 2006
2007 incident data – 11 column list of incident data for year 2007
Station list – two column list of station ID and station descriptions
Trouble list – two column list of trouble ID and trouble descriptions

Later in the project, two more files were provided. These files had rider volume by day for years 2006 and 2007.

List of Files:
Volume2006.xls
Volume 2007.xls

Procedures to access and use Database:

The database server is an internal server, which means it can only be directly access within Pace University grounds. For access off Pace campuses, the team would need to use the university's VPN dialer, which would give us network access as if we were inside the university. Once the dialer is downloaded and installed, we can connect using our Pace University issued user accounts (email address).

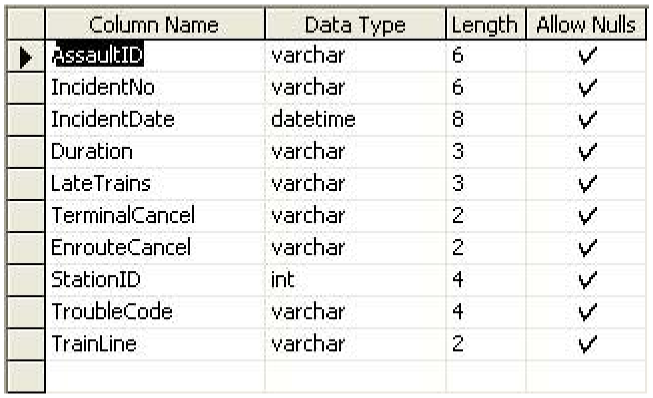
Now that we are virtually connected to the Pace University network, we can access the database server. There are a number of tools that can be used with Microsoft SQL Server. We narrowed our preferences down to two, Quest Toad for SQL Server (http://www.toadsoft.com/toadsqlserver/toad_sqlserver.htm) or SQL Server 2005 Express Edition (http://msdn2.microsoft.com/en-us/express/bb410792.aspx). Both software applications are free to use.

Once the software is downloaded and installed, we can connect to the database server and to our specific database. Here is the connection information:
Server Name: csis-rose (172.20.138.37)
Authentication: SQL Server Authentication
Database Name: SubwayIncidents
User Name: SubwayIncidents
PW: <ask for password>

Loading flat files into relational database:

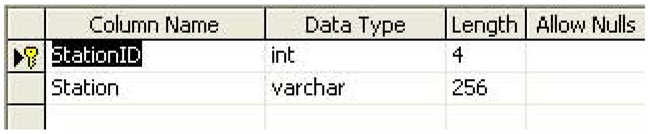
Before loading data into your relational database system, we first had to consider some preliminary questions: Table structure, data types, and relational integrity.
We decided to take the six data files and import them into their own tables. Deciding on data types was more complex. Because the two incident data files (2006 and 2007) contained information about stations and trouble codes (from station list and trouble code list), we wanted create relationships between those tables.

Here are the table structures:



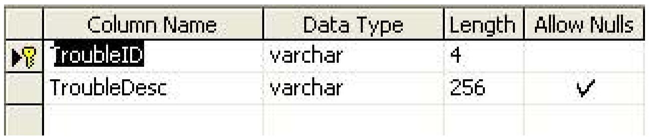
	Column Name	Data Type	Length	Allow Nulls
▶	AssaultID	varchar	6	✓
	IncidentNo	varchar	6	✓
	IncidentDate	datetime	8	✓
	Duration	varchar	3	✓
	LateTrains	varchar	3	✓
	TerminalCancel	varchar	2	✓
	EnrouteCancel	varchar	2	✓
	StationID	int	4	✓
	TroubleCode	varchar	4	✓
	TrainLine	varchar	2	✓

Incident Table (tblSubwayData2007 and tblSubwayData2006)



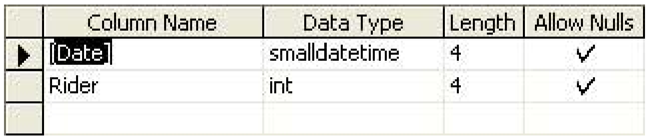
	Column Name	Data Type	Length	Allow Nulls
▶	StationID	int	4	
	Station	varchar	256	

Train Station Table (tblStationList)



	Column Name	Data Type	Length	Allow Nulls
▶	TroubleID	varchar	4	
	TroubleDesc	varchar	256	✓

Trouble List Table (tblTroubleList)



	Column Name	Data Type	Length	Allow Nulls
▶	[Date]	smalldatetime	4	✓
	Rider	int	4	✓

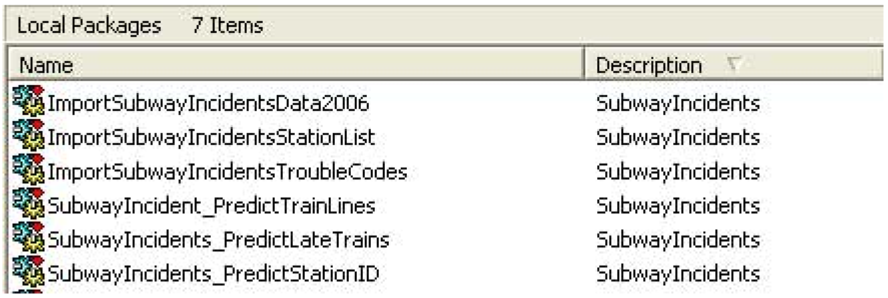
Volume Table (tblVolume2006 and tblVolume2007)

These six were created by executing SQL code. Here is an example of how the Incident table was created:

```
CREATE TABLE [SubwayIncidents].[pcronin].[tblSubwayData2006] (
[AssaultID] varchar (6) NULL,
[IncidentNo] varchar (6) NULL,
[IncidentDate] datetime NULL,
[Duration] varchar (3) NULL,
[LateTrains] varchar (3) NULL,
[TerminalCancel] varchar (2) NULL,
[EnrouteCancel] varchar (2) NULL,
[StationID] int NULL,
[TroubleCode] varchar (4) NULL,
[TrainLine] varchar (2) NULL
)
```

DTS:

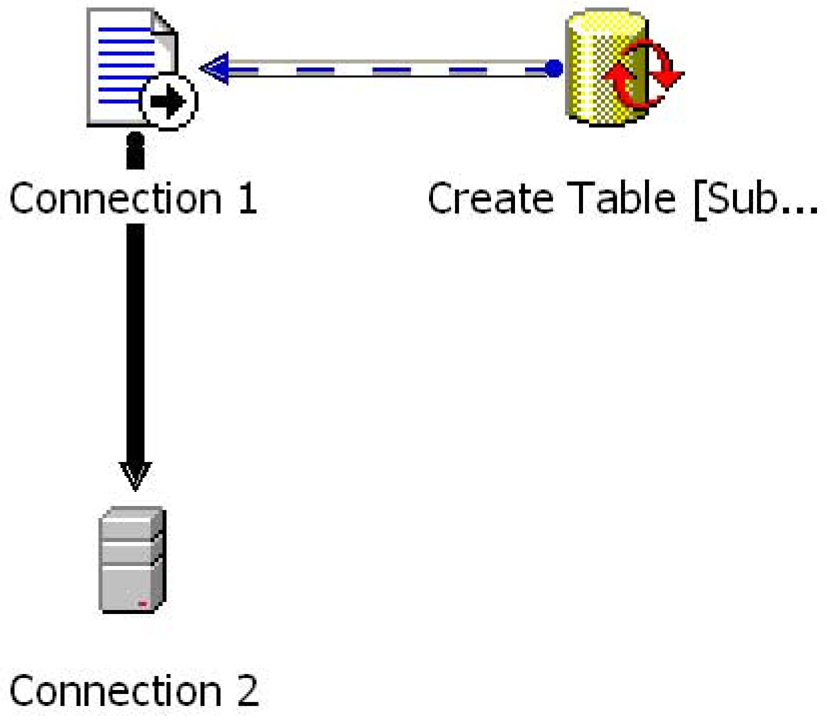
After the tables were created with desired data types and structure, we had to import the data from the text files to the relational tables inside SQL Server. We used DTS (Data Transformation Services) in Microsoft SQL Server to move the data from the text files into our database tables. Six packages were created, one for each file to table transfer.



Local Packages 7 Items

Name	Description
ImportSubwayIncidentsData2006	SubwayIncidents
ImportSubwayIncidentsStationList	SubwayIncidents
ImportSubwayIncidentsTroubleCodes	SubwayIncidents
SubwayIncident_PredictTrainLines	SubwayIncidents
SubwayIncidents_PredictLateTrains	SubwayIncidents
SubwayIncidents_PredictStationID	SubwayIncidents

These packages read the input file data and copy the data into the appropriate table columns.
Here is a graphical design of one of the DTS packages:



This package starts with the “Create Table” icon, which runs the code (shown above) to create the table. “Connection 1” is the text file with tab delimited data in it. “Connection 2” is the destination table in our relational database. The black line between the two represents the SQL code that copies the data.

The data copy is graphically represented here:

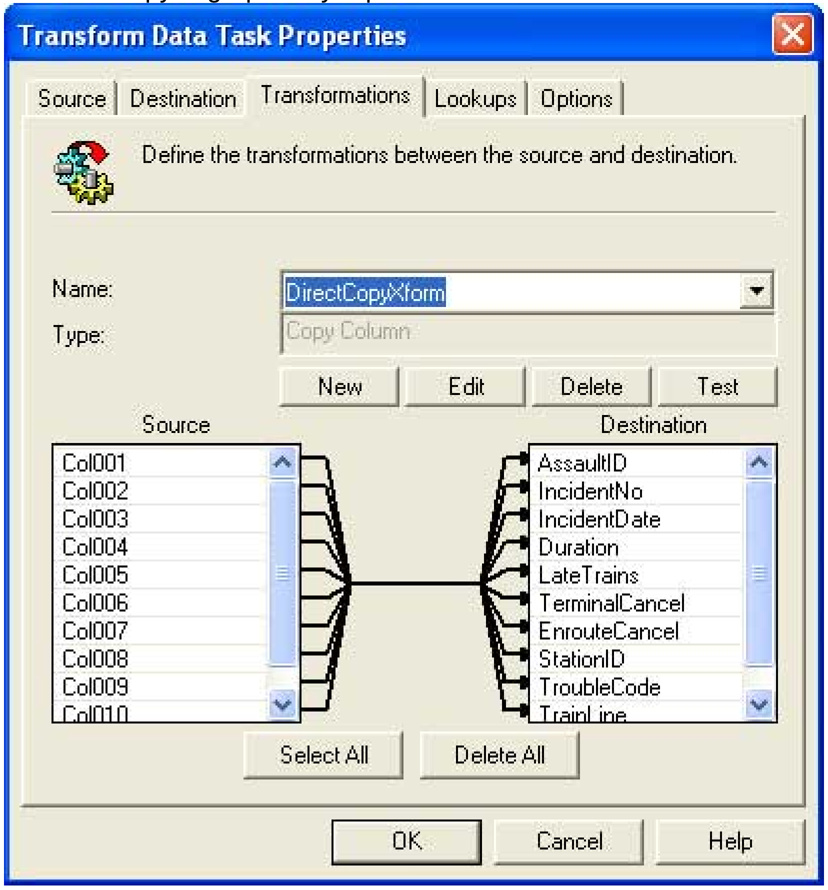
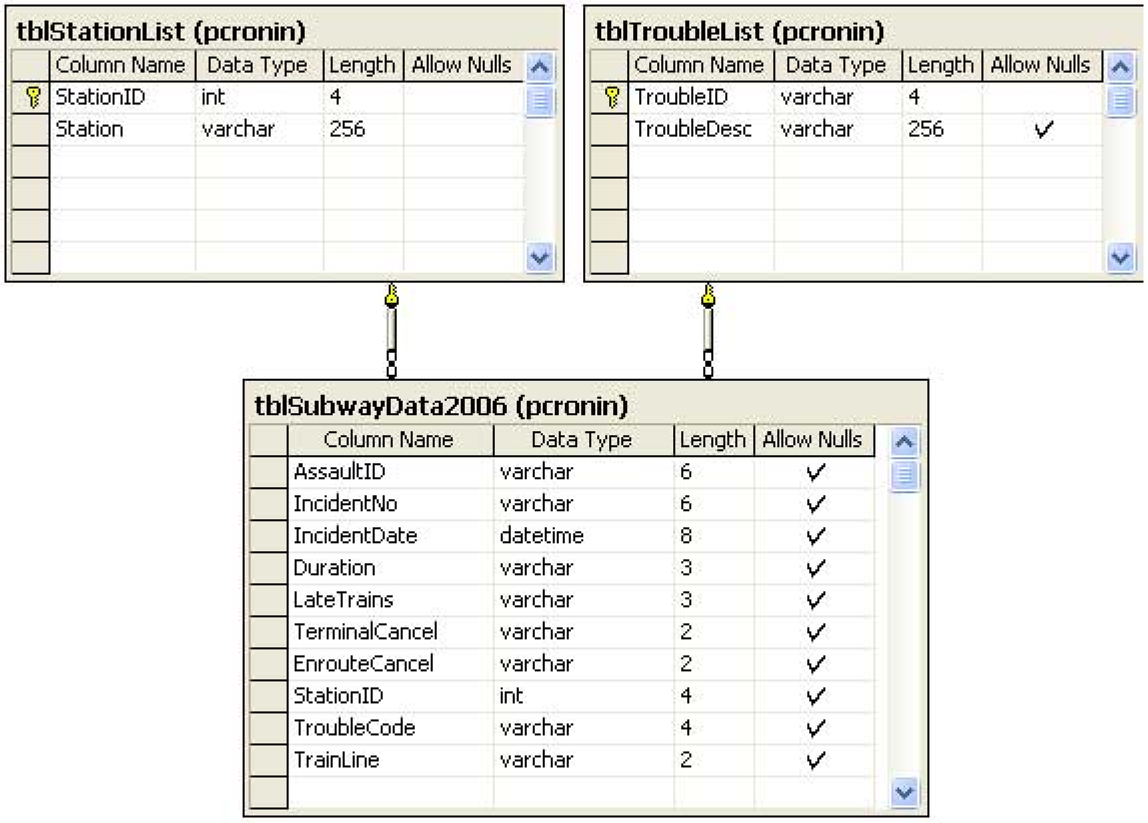


Table Relationships:

Now we have our six tables that are populated with the data that Mr. Washington provided us. The last step is to create relationships between them to protect data integrity. The main incident data are contained in two tables (tblSubwayData2007 and tblSubwayData2006) and the tblStationList and tblTroubleList are lookup or validation tables. These tables provide full text descriptions for numeric data in the incident tables.

Here are the graphical relationships:



tblStationList (pcronin)

	Column Name	Data Type	Length	Allow Nulls
🔑	StationID	int	4	
	Station	varchar	256	

tblTroubleList (pcronin)

	Column Name	Data Type	Length	Allow Nulls
🔑	TroubleID	varchar	4	
	TroubleDesc	varchar	256	✓

tblSubwayData2006 (pcronin)

Column Name	Data Type	Length	Allow Nulls
AssaultID	varchar	6	✓
IncidentNo	varchar	6	✓
IncidentDate	datetime	8	✓
Duration	varchar	3	✓
LateTrains	varchar	3	✓
TerminalCancel	varchar	2	✓
EnrouteCancel	varchar	2	✓
StationID	int	4	✓
TroubleCode	varchar	4	✓
TrainLine	varchar	2	✓

The relationships are bound by the tables Primary Key (PK) and Foreign Key (FK) fields.
The relationships:

1. tblStationList.StationID = tblSubwayData2006.StationID
2. tblTroubleList.TroubleID = tblSubwayData2006.TroubleCode

Creating Views:

To simplify comparing data between the two years, we thought it would be a good idea to create a view that would combine the data. The code below shows how we union the data together from the two tables.

```
CREATE view vwSubwayData as
        Select '2006' as [Year], AssaultID, IncidentNo, IncidentDate, Duration, LateTrains,
        TerminalCancel, EnrouteCancel, StationID, TroubleCode, TrainLine
        From tblSubwayData2006
Union all
        Select '2007' as [Year], AssaultID, IncidentNo, IncidentDate, Duration, LateTrains,
        TerminalCancel, EnrouteCancel, StationID, TroubleCode, TrainLine
        From tblSubwayData2007
```

Most of the queries that we constructed below are based on this SQL Server view.

Instead of creating relationships for the volume data, we created views that joined the two volume data tables with the two incident tables by year. The volume data contains a date field that represents the day of the year with total volume ridership. The views join the volume data with the incident data by the date. Here is the view creation of the 2006 data:

```
Create view vwSubwayData2006Volume AS
select s.*, v.Rider
```

```
	from tblSubwayData2006 s Inner Join tblVolume2006 v
		ON
		cast(datepart(month, IncidentDate) as varchar(2))+'/'+
		cast(datepart(day, IncidentDate) as varchar(2))+'/'+
		cast(datepart(year, IncidentDate) as varchar(4))
		=
		cast(datepart(month, Date) as varchar(2))+'/'+
		cast(datepart(day, Date) as varchar(2))+'/'+
		cast(datepart(year, Date) as varchar(4))
```

Here is the view creation of the 2006 data:

```
	Create view vwSubwayData2007Volume AS
	select s.*, v.Rider
	from tblSubwayData2007 s Inner Join tblVolume2007 v
		ON
		cast(datepart(month, IncidentDate) as varchar(2))+'/'+
		cast(datepart(day, IncidentDate) as varchar(2))+'/'+
		cast(datepart(year, IncidentDate) as varchar(4))
		=
		cast(datepart(month, Date) as varchar(2))+'/'+
		cast(datepart(day, Date) as varchar(2))+'/'+
		cast(datepart(year, Date) as varchar(4))
```

Queries used to produce results:
Using one of the two software tools mentioned above (TOAD or MS SQL Server Express), a user can execute these queries to produce results from the data table tables and views created above.

Trouble Code – Lists all trouble codes with occurrence count and average duration of train delays for year 2007 and having more than one occurrence.

```
	Select [Year], sd.TroubleCode, TroubleDesc,
		Count(sd.TroubleCode) as Incidents,
		Avg((CAST(Duration AS int))) as AvgDuration
	From vwSubwayData sd
		inner join tblTroubleList tl on sd.TroubleCode = tl.TroubleID
	Where [Year] = '2007'
	Group By [Year], sd.TroubleCode, TroubleDesc
	Having Count(sd.TroubleCode) > 2
	Order By Incidents desc, AvgDuration desc
```

Train Line – Lists all train lines with occurrence count and average duration of train delays for the trouble code of “Person Holding Doors” and year 2007.

```
	Select TrainLine, Count(TrainLine) as Incidents,
		Avg((CAST(Duration AS int))) as AvgDuration
	From vwSubwayData
	Where [Year] = '2007' and TroubleCode = '0741'
	Group by TrainLine
	Order by Incidents desc
```

Stations – Lists all stations with incident occurrence count for trouble code of “Employee Assaulted By Cust.” and year 2007 and having more than one occurrence.

```
	Select Station, Count(sd.StationID) as Incidents
	From vwSubwayData sd
		inner join tblStationList sl on sd.StationID = sl.StationID
```

```
Where [Year] = '2007' and TroubleCode = '4011'
Group by Station
Having Count(sd.StationID) > 2
Order by Incidents desc
```

"Person Holding Doors" incidents that lead to an assault on an employee – This is a Cursor. It queries all assaults and individually loops over the record set to find associated "Person Holding Doors" incidents. The cursor first selects all assaults (trouble code 4011) for year 2007. It loops over this record set to find associated trouble code 0741 "Persons Hold Doors".

```
Drop table #TempIncidents
Create Table #TempIncidents
(AssaultID Varchar(6), Duration Varchar(3), StationID int, TrainLine Varchar(2))

DECLARE @assault varchar(6)
DECLARE Assault_cursor CURSOR FOR
        Select AssaultID
        From vwSubwayData sd
        Where sd.Year = '2007' and TroubleCode =  '4011'

OPEN Assault_cursor
FETCH NEXT FROM Assault_cursor
INTO @assault

WHILE @@FETCH_STATUS = 0
BEGIN
        Insert Into #TempIncidents
        Select   AssaultID, Duration, StationID, TrainLine
        From     vwSubwayData sd
        Where    TroubleCode = '0741' and AssaultID = @assault

        FETCH NEXT FROM Assault_cursor
        INTO @assault
END

CLOSE Assault_cursor
DEALLOCATE Assault_cursor

Select * From #TempIncidents
```

Queries based on the temporary table "#TempIncidents":
Station – Lists the Station and the train line with how many occurrences and average duration of the delay during a door holding incident.

```
Select Station, TrainLine, Count(sd.StationID) as Incidents,
        Avg((CAST(Duration AS int))) as AvgDuration
From #TempIncidents sd
        inner join tblStationList sl on sd.StationID = sl.StationID
Group by Station, TrainLine
Having Count(sd.StationID) > 2
Order by Incidents desc
```

Train Line – Lists the train line with how many occurrences and average duration of the delay during a door holding incident.

```
Select TrainLine, Count(TrainLine) as Incidents,
        Avg((CAST(Duration AS int))) as AvgDuration
From #TempIncidents
Group by TrainLine
Order by Incidents desc
```

Queries based on Volume data

The following queries are based on the volume data using the volume views created earlier.

Average number of riders by month

```
select datepart(month, Date) as month, avg(Rider)
from tblVolume2007
group by datepart(month, Date)
order by avg(Rider) desc
```

Total number of assults by month

```
select datepart(month, IncidentDate) as month, count(IncidentNo) as Assault
from vwSubwayData2007Volume
where TroubleCode = '4011'
group by datepart(month, IncidentDate)
order by count(IncidentNo) desc
```

Total number of "DELAYED BY TRACK/WORK GANGS" incidents by month

```
select datepart(month, IncidentDate) as month, count(IncidentNo) as Incidents
from vwSubwayData2007Volume
where TroubleCode = '8204'
group by datepart(month, IncidentDate)
order by count(IncidentNo) desc
```

Total number of non assault incidents by month

```
select datepart(month, IncidentDate) as month, count(IncidentNo) as Incidents
from vwSubwayData2007Volume
where TroubleCode <> '4011'
group by datepart(month, IncidentDate)
order by count(IncidentNo) desc
```