EXPERIMENT - 1

Aim: To Study of ETL process and its applications.

Theory:

ETL is a process that extracts the data from different source systems, then transforms the data (like applying calculations, concatenations, etc.) and finally loads the data into the Data Warehouse system. Full form of ETL is Extract, Transform and Load. Extract, Transform, Load each denotes a process in the movement of data from its source to a data storage system, often referred to as a data warehouse.

1) Extraction

In this step, data is extracted from the source system into the staging area. Transformations if any are done in staging area so that performance of source system in not degraded. Also, if corrupted data is copied directly from the source into Data warehouse database, rollback will be a challenge. Staging area gives an opportunity to validate extracted data before it moves into the Data warehouse.

Three Data Extraction methods:

1. Full Extraction
2. Partial Extraction- without update notification.
3. Partial Extraction- with update notification

2) Transformation

Data extracted from source server is raw and not usable in its original form. Therefore, it needs to be cleansed, mapped and transformed. In fact, this is the key step where ETL process adds value and changes data such that insightful BI reports can be generated.

3) Loading

Loading data into the target data warehouse database is the last step of the ETL process. In a typical Data warehouse, huge volume of data needs to be loaded in a relatively short period (nights). Hence, load process should be optimized for performance. In case of load failure, recover mechanisms should be configured to restart from the point of failure without data integrity loss. Data Warehouse admins need to monitor, resume, cancel loads as per prevailing server performance.

Types of Loading:

1. **Initial Load** — populating all the Data Warehouse tables
2. **Incremental Load** — applying ongoing changes as when needed periodically.

3. **Full Refresh** —erasing the contents of one or more tables and reloading with fresh data.

Benefits of ETL:

The main benefit of ETL is that it is much easier and faster to use than traditional methods that move data by manually writing codes. ETL tools contain graphical interfaces which speed up the process of mapping tables and columns between the source and target storages.
Below are some key advantages of ETL tools:

1. **Ease of Use:** The first and foremost advantage of using an ETL tool is the ease of use. The tool itself specifies data sources and the rules for extracting and processing data, and then, it implements the process and loads that data. This eliminates the need for coding in a traditional programming sense, where you have to write the procedures and code.
2. **Visual Flow:** ETL tools are based on Graphical User Interface (GUI) and offer a visual flow of the system's logic.
3. **Operational Resilience:** ETL tools possess built-in error-handling functionality which helps data engineers to build on the features of an ETL tool to develop a resilient and well-instrumented ETL system.
4. **Good for Complex Data Management Situations:** ETL tools offer better utility for moving large volumes of data and transferring them in batches.
5. **Advanced Data Profiling and Cleansing:** ETL tools bring in a richer set of cleansing functions as compared to the ones available in SQL.
6. **Enhanced Business Intelligence:** ETL tools improve the access to data as it simplifies the process of extracting, transforming and loading. Improved access to information directly impacts the strategic and operational decisions that are based on data-driven facts.
7. **Performance:** The structure of an ETL platform simplifies the process of building a high-quality data warehousing system.

ETL Tools List:

Several commercial and open source ETL tools are listed below:

- IBM InfoSphere DataStage
- Microsoft SQL Server Integration Services
- Oracle Warehouse Builder / Data Integrator
- SAP Data Services

RESULT: Studied the ETL process and its applications.

EXPERIMENT - 2

Aim: Implementation of Data Warehouse Cleansing, to remove inconsistency and format the data.

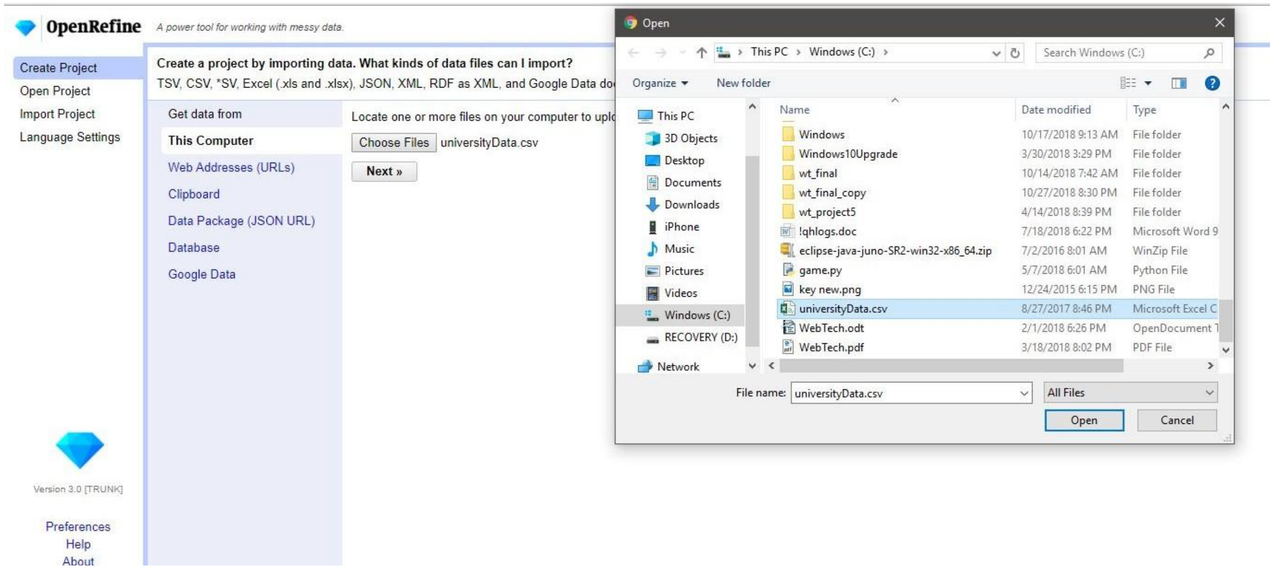
Tool Used: Open Refine

- OpenRefine, formerly called Google Refine and before that Freebase Gridworks, is a standalone open source desktop application for data cleanup and transformation to other formats, the activity known as data wrangling. It is similar to spreadsheet applications (and can work with spreadsheet file formats); however, it behaves more like a database.
- OpenRefine (formerly Google Refine) is a powerful tool for working with messy data: cleaning it; transforming it from one format into another; and extending it with web services and external data.
- It operates on *rows* of data which have cells under *columns,* which is very similar to relational database tables. An OpenRefine project consists of one table. The user can filter the rows to display using *facets* that define filtering criteria (for example, showing rows where a given column is not empty). Unlike spreadsheets, most operations in OpenRefine are done on all visible rows: transformation of all cells in all rows under one column, creation of a new column based on existing column data, etc. All actions that were done on a dataset are stored in a project and can be replayed on another dataset.
- Unlike spreadsheets, no formulas are stored in the cells, but formulas are used to transform the data, and transformation is done only once. Transformation expressions can be written in General Refine Expression Language (GREL), Jython (i.e. Python) and Clojure.
- The program has a web user interface. However, it is not hosted on the web (SAAS), but is available for download and use on the local machine. When starting OpenRefine, it starts a web server and starts a browser to open the web UI powered by this web server.

Step 1: Load the data set and create project

Click on the choose file tab and browse data and choose the file. Click on next and then click on create project.

The data can be taken either from an online portal or downloaded to the system and then used.



OpenRefine — A power tool for working with messy data.

Create a project by importing data. What kinds of data files can I import?
TSV, CSV, *SV, Excel (.xls and .xlsx), JSON, XML, RDF as XML, and Google Data do

Get data from: This Computer

Locate one or more files on your computer to uplo

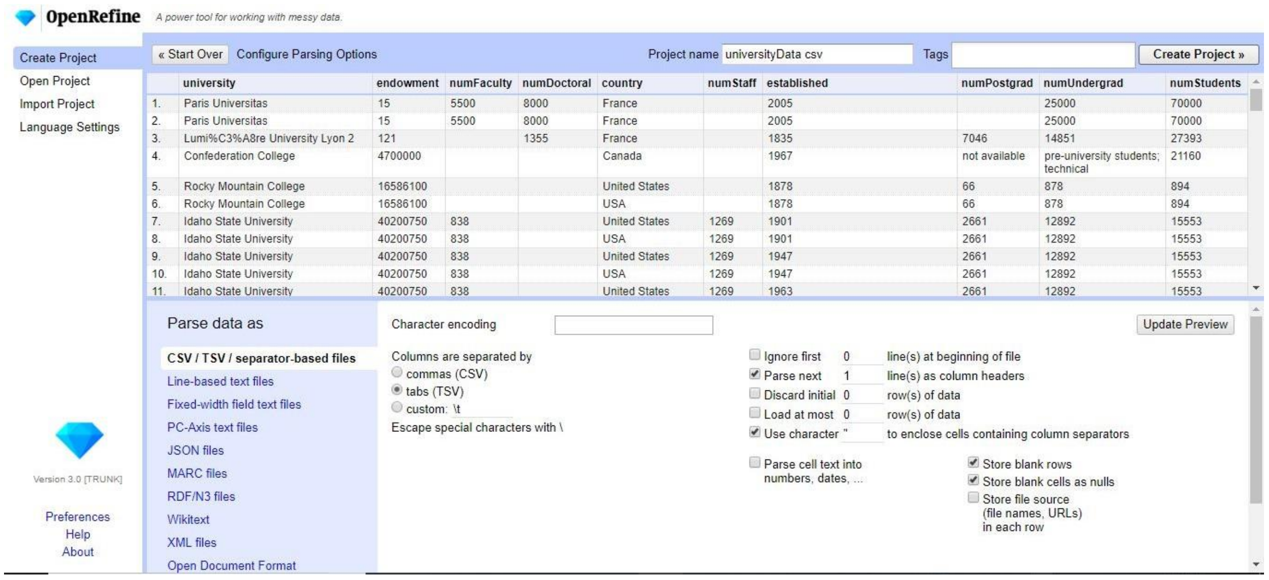
Choose Files: universityData.csv

Open — This PC › Windows (C:)

Name	Date modified	Type
Windows	10/17/2018 9:13 AM	File folder
Windows10Upgrade	3/30/2018 3:29 PM	File folder
wt_final	10/14/2018 7:42 AM	File folder
wt_final_copy	10/27/2018 8:30 PM	File folder
wt_project5	4/14/2018 8:39 PM	File folder
!qhlogs.doc	7/18/2018 6:22 PM	Microsoft Word 9
eclipse-java-juno-SR2-win32-x86_64.zip	7/2/2016 8:01 AM	WinZip File
game.py	5/7/2018 6:01 AM	Python File
key new.png	12/24/2015 6:15 PM	PNG File
universityData.csv	8/27/2017 8:46 PM	Microsoft Excel C
WebTech.odt	2/1/2018 6:26 PM	OpenDocument T
WebTech.pdf	3/18/2018 8:02 PM	PDF File

File name: universityData.csv — All Files

Fig 1. Choosing the data set



Project name: universityData csv

	university	endowment	numFaculty	numDoctoral	country	numStaff	established	numPostgrad	numUndergrad	numStudents
1.	Paris Universitas	15	5500	8000	France		2005		25000	70000
2.	Paris Universitas	15	5500	8000	France		2005		25000	70000
3.	Lumi%C3%A8re University Lyon 2	121		1355	France		1835	7046	14851	27393
4.	Confederation College	4700000			Canada		1967	not available	pre-university students; technical	21160
5.	Rocky Mountain College	16586100			United States		1878	66	878	894
6.	Rocky Mountain College	16586100			USA		1878	66	878	894
7.	Idaho State University	40200750	838		United States	1269	1901	2661	12892	15553
8.	Idaho State University	40200750	838		USA	1269	1901	2661	12892	15553
9.	Idaho State University	40200750	838		United States	1269	1947	2661	12892	15553
10.	Idaho State University	40200750	838		USA	1269	1947	2661	12892	15553
11.	Idaho State University	40200750	838		United States	1269	1963	2661	12892	15553

Parse data as: CSV / TSV / separator-based files

Character encoding:

Columns are separated by: commas (CSV) [✓ tabs (TSV)] custom: \t

Escape special characters with \

Ignore first 0 line(s) at beginning of file

[✓] Parse next 1 line(s) as column headers

Discard initial 0 row(s) of data

Load at most 0 row(s) of data

[✓] Use character " to enclose cells containing column separators

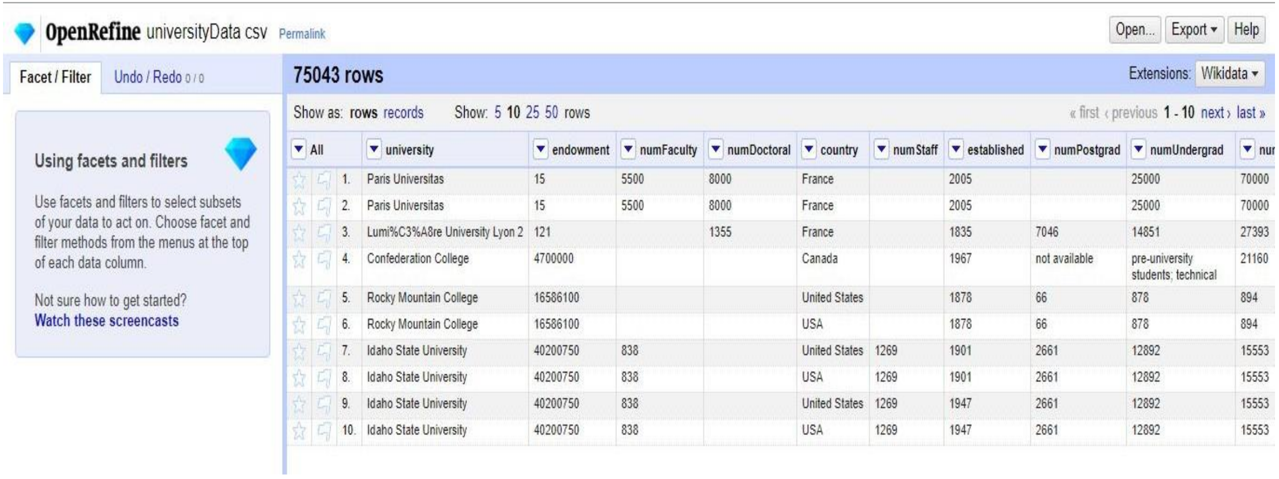
Parse cell text into numbers, dates, ...

[✓] Store blank rows

[✓] Store blank cells as nulls

Store file source (file names, URLs) in each row

Fig 2. Creating project



Using facets and filters

Use facets and filters to select subsets of your data to act on. Choose facet and filter methods from the menus at the top of each data column.

Not sure how to get started?
Watch these screencasts

75043 rows

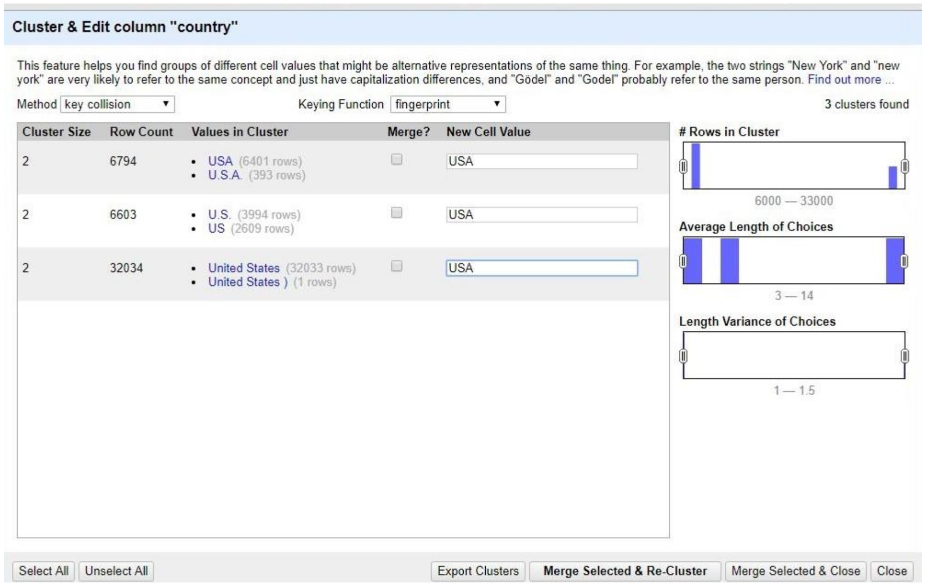
Show as: rows records   Show: 5 10 25 50 rows   « first ‹ previous 1 - 10 next › last »

All	university	endowment	numFaculty	numDoctoral	country	numStaff	established	numPostgrad	numUndergrad	nur
1.	Paris Universitas	15	5500	8000	France		2005		25000	70000
2.	Paris Universitas	15	5500	8000	France		2005		25000	70000
3.	Lumi%C3%A8re University Lyon 2	121		1355	France		1835	7046	14851	27393
4.	Confederation College	4700000			Canada		1967	not available	pre-university students; technical	21160
5.	Rocky Mountain College	16586100			United States		1878	66	878	894
6.	Rocky Mountain College	16586100			USA		1878	66	878	894
7.	Idaho State University	40200750	838		United States	1269	1901	2661	12892	15553
8.	Idaho State University	40200750	838		USA	1269	1901	2661	12892	15553
9.	Idaho State University	40200750	838		United States	1269	1947	2661	12892	15553
10.	Idaho State University	40200750	838		USA	1269	1947	2661	12892	15553

Fig 3. Data set loaded

Step 2: Edit column to remove inconsistency

As we can see above there are some inconsistencies in the data set, the country name for United States of America is inconsistently written as U.S.A., USA, U.S., US, United States, etc. To remove the inconsistencies all together from this huge data set we select the column and then choose the **Edit cells** option and then click on the **cluster and edit** option. We then observe the following window that opens up.



Cluster & Edit column "country"

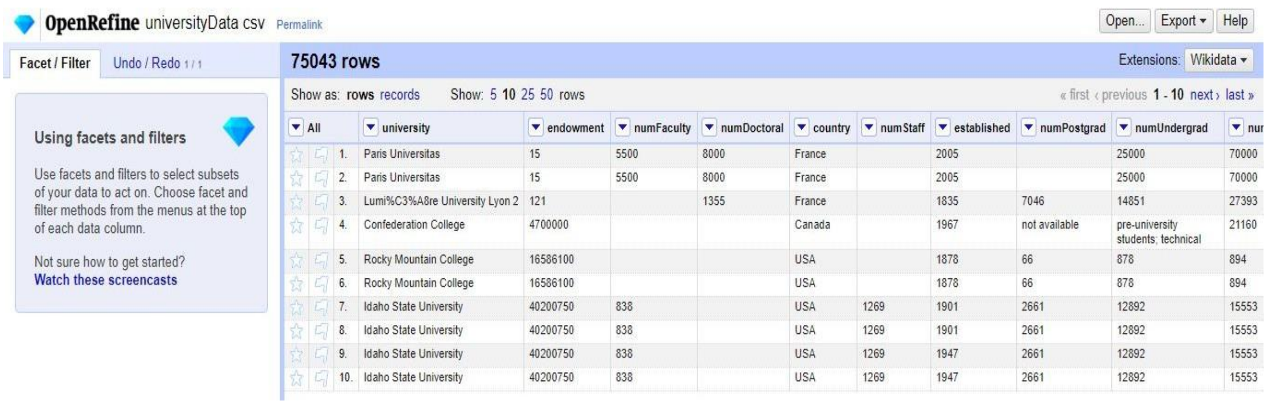
This feature helps you find groups of different cell values that might be alternative representations of the same thing. For example, the two strings "New York" and "new york" are very likely to refer to the same concept and just have capitalization differences, and "Gödel" and "Godel" probably refer to the same person. Find out more ...

Method key collision   Keying Function fingerprint   3 clusters found

Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value
2	6794	• USA (6401 rows) • U.S.A. (393 rows)	☐	USA
2	6603	• U.S. (3994 rows) • US (2609 rows)	☐	USA
2	32034	• United States (32033 rows) • United States) (1 rows)	☐	USA

Fig 4. Removing inconsistencies

Finally, click on the select all tab and then **Merge Selected and Re-Cluster** option is selected.



OpenRefine universityData csv Permalink

Open... Export ▾ Help

Facet / Filter | Undo / Redo 1 / 1

Using facets and filters

Use facets and filters to select subsets of your data to act on. Choose facet and filter methods from the menus at the top of each data column.

Not sure how to get started?
Watch these screencasts

75043 rows

Extensions: Wikidata ▾

Show as: rows records   Show: 5 10 25 50 rows   « first ‹ previous 1 - 10 next › last »

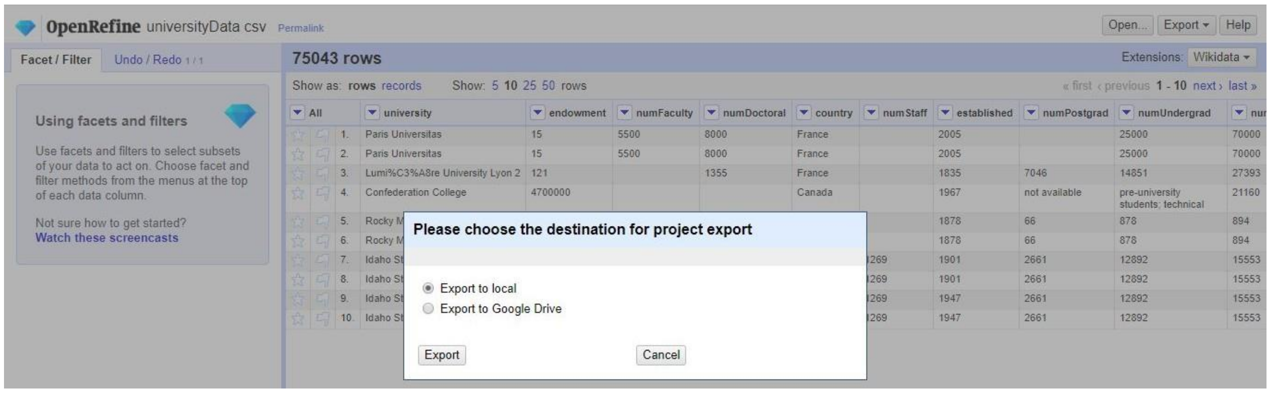
All	university	endowment	numFaculty	numDoctoral	country	numStaff	established	numPostgrad	numUndergrad	nur
1.	Paris Universitas	15	5500	8000	France		2005		25000	70000
2.	Paris Universitas	15	5500	8000	France		2005		25000	70000
3.	Lumi%C3%A8re University Lyon 2	121		1355	France		1835	7046	14851	27393
4.	Confederation College	4700000			Canada		1967	not available	pre-university students; technical	21160
5.	Rocky Mountain College	16586100			USA		1878	66	878	894
6.	Rocky Mountain College	16586100			USA		1878	66	878	894
7.	Idaho State University	40200750	838		USA	1269	1901	2661	12892	15553
8.	Idaho State University	40200750	838		USA	1269	1901	2661	12892	15553
9.	Idaho State University	40200750	838		USA	1269	1947	2661	12892	15553
10.	Idaho State University	40200750	838		USA	1269	1947	2661	12892	15553

Fig 5. Data set after removing inconsistencies

After, removing the inconsistencies, finally the data set is formatted for further use and analytics purpose.

Step 3: Export project

After clicking on the Export tab, the formatted data can be exported and downloaded.



OpenRefine universityData csv Permalink

Open... Export ▾ Help

Facet / Filter | Undo / Redo 1 / 1

Using facets and filters

Use facets and filters to select subsets of your data to act on. Choose facet and filter methods from the menus at the top of each data column.

Not sure how to get started?
Watch these screencasts

75043 rows

Extensions: Wikidata ▾

Show as: rows records   Show: 5 10 25 50 rows   « first ‹ previous 1 - 10 next › last »

All	university	endowment	numFaculty	numDoctoral	country	numStaff	established	numPostgrad	numUndergrad	nur
1.	Paris Universitas	15	5500	8000	France		2005		25000	70000
2.	Paris Universitas	15	5500	8000	France		2005		25000	70000
3.	Lumi%C3%A8re University Lyon 2	121		1355	France		1835	7046	14851	27393
4.	Confederation College	4700000			Canada		1967	not available	pre-university students; technical	21160
5.	Rocky M						1878	66	878	894
6.	Rocky M						1878	66	878	894
7.	Idaho St					1269	1901	2661	12892	15553
8.	Idaho St					1269	1901	2661	12892	15553
9.	Idaho St					1269	1947	2661	12892	15553
10.	Idaho St					1269	1947	2661	12892	15553

Please choose the destination for project export

(●) Export to local
( ) Export to Google Drive

Export   Cancel

Fig 6. Exporting the formatted data set

EXPERIMENT - 3

Aim: Implementation of Data Warehouse Cleansing to remove redundancy in data.

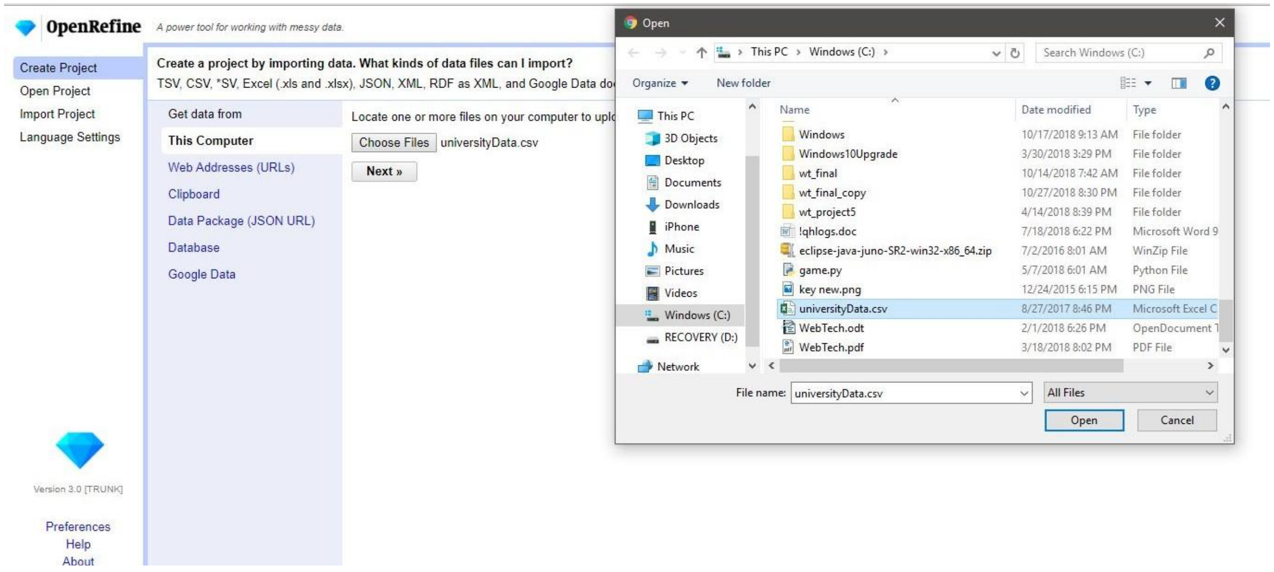
Tool Used: Open Refine

- OpenRefine, formerly called Google Refine and before that Freebase Gridworks, is a standalone open source desktop application for data cleanup and transformation to other formats, the activity known as data wrangling. It is similar to spreadsheet applications (and can work with spreadsheet file formats); however, it behaves more like a database.
- OpenRefine (formerly Google Refine) is a powerful tool for working with messy data: cleaning it; transforming it from one format into another; and extending it with web services and external data.
- It operates on *rows* of data which have cells under *columns,* which is very similar to relational database tables. An OpenRefine project consists of one table. The user can filter the rows to display using *facets* that define filtering criteria (for example, showing rows where a given column is not empty). Unlike spreadsheets, most operations in OpenRefine are done on all visible rows: transformation of all cells in all rows under one column, creation of a new column based on existing column data, etc. All actions that were done on a dataset are stored in a project and can be replayed on another dataset.
- Unlike spreadsheets, no formulas are stored in the cells, but formulas are used to transform the data, and transformation is done only once. Transformation expressions can be written in General Refine Expression Language (GREL), Jython (i.e. Python) and Clojure.
- The program has a web user interface. However, it is not hosted on the web (SAAS), but is available for download and use on the local machine. When starting OpenRefine, it starts a web server and starts a browser to open the web UI powered by this web server.

Step 1: Load the data set and create project

Click on the choose file tab and browse data and choose the file. Click on next and then click on create project.

The data can be taken either from an online portal or downloaded to the system and then used.



Create a project by importing data. What kinds of data files can I import?
TSV, CSV, *SV, Excel (.xls and .xlsx), JSON, XML, RDF as XML, and Google Data do

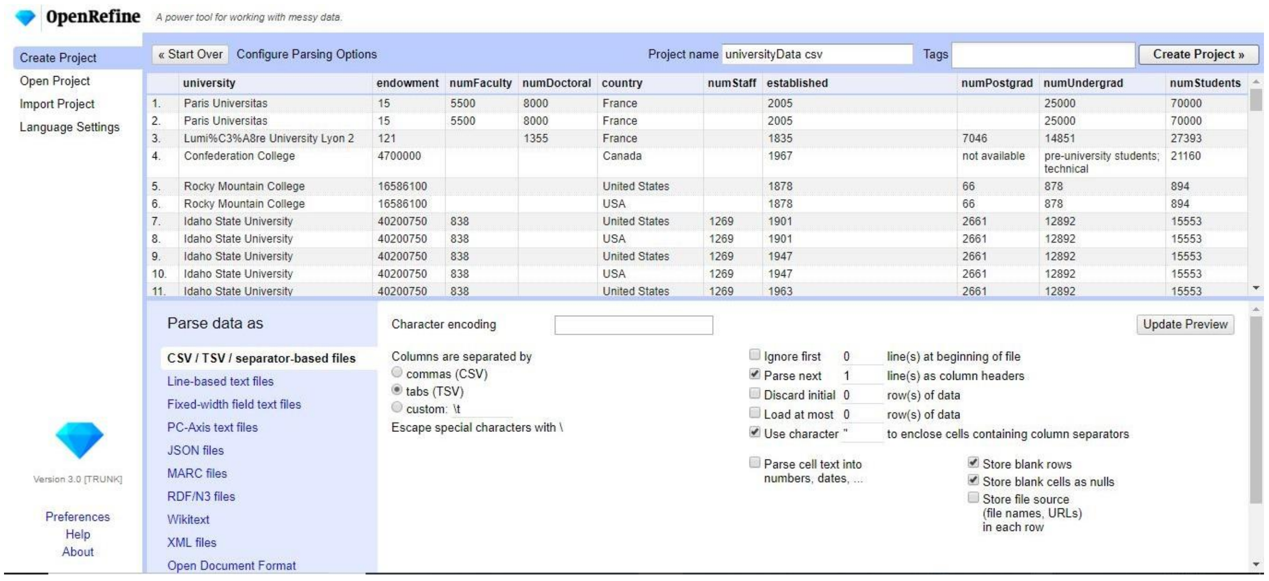
Locate one or more files on your computer to uplo

universityData.csv

Name	Date modified	Type
Windows	10/17/2018 9:13 AM	File folder
Windows10Upgrade	3/30/2018 3:29 PM	File folder
wt_final	10/14/2018 7:42 AM	File folder
wt_final_copy	10/27/2018 8:30 PM	File folder
wt_project5	4/14/2018 8:39 PM	File folder
!qhlogs.doc	7/18/2018 6:22 PM	Microsoft Word 9
eclipse-java-juno-SR2-win32-x86_64.zip	7/2/2016 8:01 AM	WinZip File
game.py	5/7/2018 6:01 AM	Python File
key new.png	12/24/2015 6:15 PM	PNG File
universityData.csv	8/27/2017 8:46 PM	Microsoft Excel C
WebTech.odt	2/1/2018 6:26 PM	OpenDocument T
WebTech.pdf	3/18/2018 8:02 PM	PDF File

File name: universityData.csv

Fig 1. Choosing the data set



Configure Parsing Options

Project name universityData csv

Tags

	university	endowment	numFaculty	numDoctoral	country	numStaff	established	numPostgrad	numUndergrad	numStudents
1.	Paris Universitas	15	5500	8000	France		2005		25000	70000
2.	Paris Universitas	15	5500	8000	France		2005		25000	70000
3.	Lumi%C3%A8re University Lyon 2	121		1355	France		1835	7046	14851	27393
4.	Confederation College	4700000			Canada		1967	not available	pre-university students; technical	21160
5.	Rocky Mountain College	16586100			United States		1878	66	878	894
6.	Rocky Mountain College	16586100			USA		1878	66	878	894
7.	Idaho State University	40200750	838		United States	1269	1901	2661	12892	15553
8.	Idaho State University	40200750	838		USA	1269	1901	2661	12892	15553
9.	Idaho State University	40200750	838		United States	1269	1947	2661	12892	15553
10.	Idaho State University	40200750	838		USA	1269	1947	2661	12892	15553
11.	Idaho State University	40200750	838		United States	1269	1963	2661	12892	15553

Character encoding

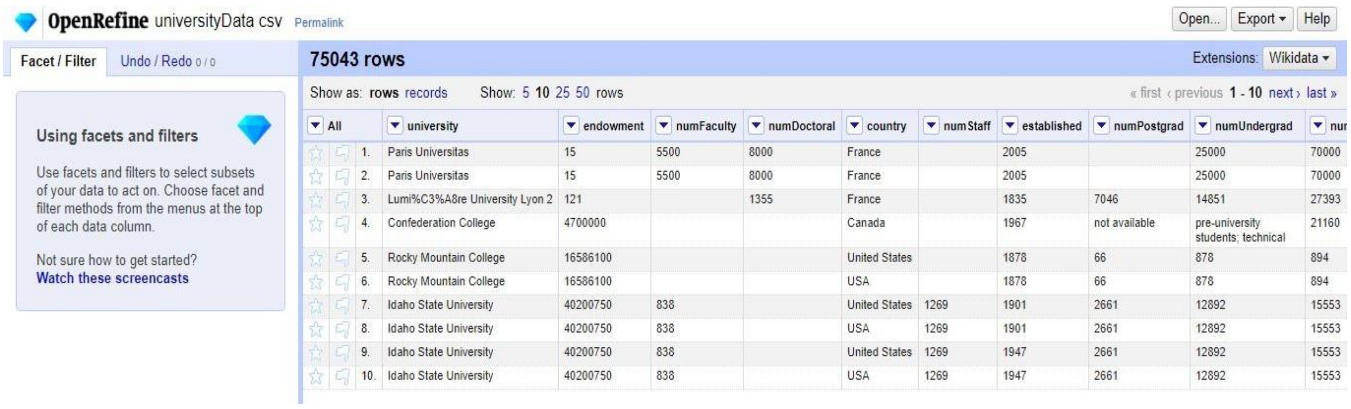
Columns are separated by
commas (CSV) [✓ tabs (TSV)] custom: \t
Escape special characters with \

- [ ] Ignore first 0 line(s) at beginning of file
- [x] Parse next 1 line(s) as column headers
- [ ] Discard initial 0 row(s) of data
- [ ] Load at most 0 row(s) of data
- [x] Use character " to enclose cells containing column separators
- [ ] Parse cell text into numbers, dates, ...
- [x] Store blank rows
- [x] Store blank cells as nulls
- [ ] Store file source (file names, URLs) in each row

Fig 2. Creating project

Step 2: Edit column to remove redundancy

As we can see, the data set below contains the redundant names of universities. To remove the redundancy all together from this huge data set we select the column and then choose the **Edit cells** option and then click on the **blank down** option.


OpenRefine universityData csv Permalink

Open... Export Help

Facet / Filter Undo / Redo 0 / 0

Using facets and filters

Use facets and filters to select subsets of your data to act on. Choose facet and filter methods from the menus at the top of each data column.

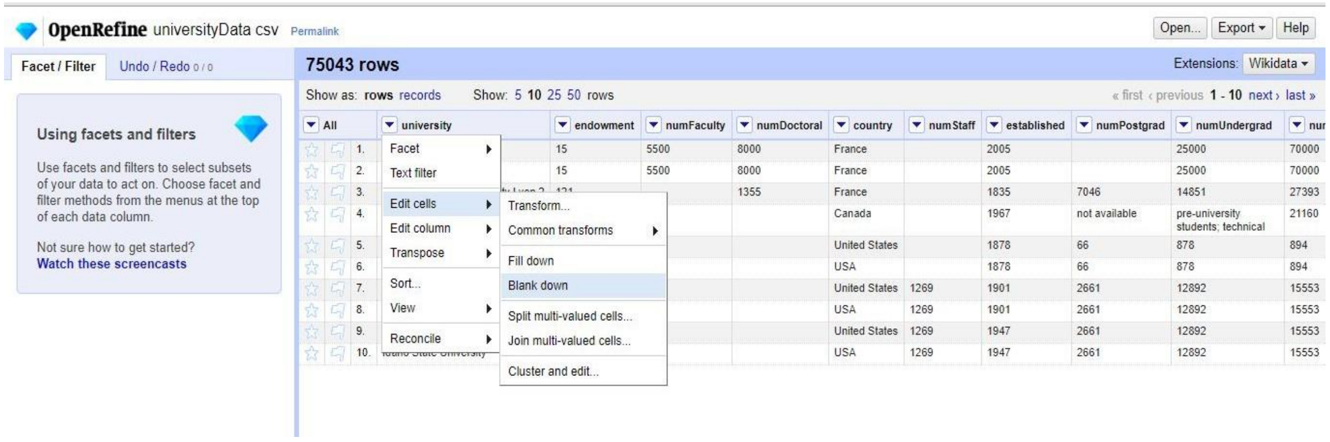
Not sure how to get started? Watch these screencasts

75043 rows Extensions: Wikidata

Show as: rows records Show: 5 10 25 50 rows « first ‹ previous 1 - 10 next › last »

All	university	endowment	numFaculty	numDoctoral	country	numStaff	established	numPostgrad	numUndergrad	nur
1.	Paris Universitas	15	5500	8000	France		2005		25000	70000
2.	Paris Universitas	15	5500	8000	France		2005		25000	70000
3.	Lumi%C3%A8re University Lyon 2	121		1355	France		1835	7046	14851	27393
4.	Confederation College	4700000			Canada		1967	not available	pre-university students; technical	21160
5.	Rocky Mountain College	16586100			United States		1878	66	878	894
6.	Rocky Mountain College	16586100			USA		1878	66	878	894
7.	Idaho State University	40200750	838		United States	1269	1901	2661	12892	15553
8.	Idaho State University	40200750	838		USA	1269	1901	2661	12892	15553
9.	Idaho State University	40200750	838		United States	1269	1947	2661	12892	15553
10.	Idaho State University	40200750	838		USA	1269	1947	2661	12892	15553

Fig 3. Data set with redundancy



OpenRefine universityData csv Permalink

Open... Export Help

Facet / Filter Undo / Redo 0 / 0

Using facets and filters

Use facets and filters to select subsets of your data to act on. Choose facet and filter methods from the menus at the top of each data column.

Not sure how to get started? Watch these screencasts

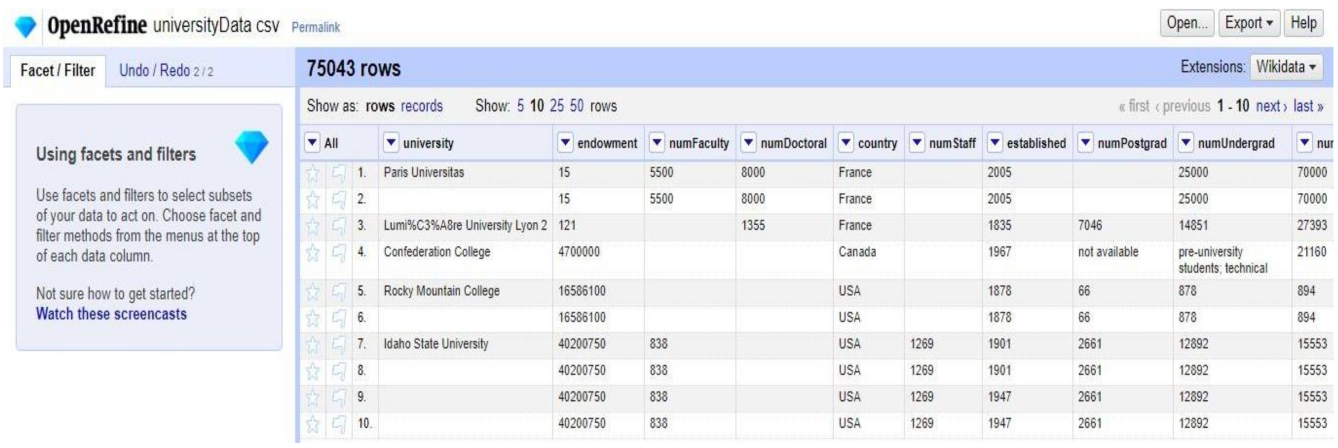
75043 rows Extensions: Wikidata

Show as: rows records Show: 5 10 25 50 rows « first ‹ previous 1 - 10 next › last »

All	university	endowment	numFaculty	numDoctoral	country	numStaff	established	numPostgrad	numUndergrad	nur
1.		15	5500	8000	France		2005		25000	70000
2.		15	5500	8000	France		2005		25000	70000
3.				1355	France		1835	7046	14851	27393
4.					Canada		1967	not available	pre-university students; technical	21160
5.					United States		1878	66	878	894
6.					USA		1878	66	878	894
7.					United States	1269	1901	2661	12892	15553
8.					USA	1269	1901	2661	12892	15553
9.					United States	1269	1947	2661	12892	15553
10.					USA	1269	1947	2661	12892	15553

Fig 4. Removing redundancy

After selecting the blank down option, redundant names of universities from the data set is removed. And we get the below data set which does not contain any redundancy.


OpenRefine universityData csv Permalink

Open... Export Help

Facet / Filter Undo / Redo 2 / 2

Using facets and filters

Use facets and filters to select subsets of your data to act on. Choose facet and filter methods from the menus at the top of each data column.

Not sure how to get started? Watch these screencasts

75043 rows Extensions: Wikidata

Show as: rows records Show: 5 10 25 50 rows « first ‹ previous 1 - 10 next › last »

All	university	endowment	numFaculty	numDoctoral	country	numStaff	established	numPostgrad	numUndergrad	nur
1.	Paris Universitas	15	5500	8000	France		2005		25000	70000
2.		15	5500	8000	France		2005		25000	70000
3.	Lumi%C3%A8re University Lyon 2	121		1355	France		1835	7046	14851	27393
4.	Confederation College	4700000			Canada		1967	not available	pre-university students; technical	21160
5.	Rocky Mountain College	16586100			USA		1878	66	878	894
6.		16586100			USA		1878	66	878	894
7.	Idaho State University	40200750	838		USA	1269	1901	2661	12892	15553
8.		40200750	838		USA	1269	1901	2661	12892	15553
9.		40200750	838		USA	1269	1947	2661	12892	15553
10.		40200750	838		USA	1269	1947	2661	12892	15553

Fig 5. Data set without redundancy

EXPERIMENT - 4

Aim:- Implementation of Classification technique on ARFF files using WEKA.

Theory:
This experiment illustrates the use of j-48 classifier in Weka. The sample data set used in this experiment is “student” data available at .arff format. This document assumes that appropriate data pre-processing has been performed.

Steps involved in this experiment:

Step-1: We begin the experiment by loading the data (employee1.arff)into Weka.

Step2: Next we select the “classify” tab and click “choose” button to select the “j48”classifier.

Step3: Now we specify the various parameters. These can be specified by clicking in the text box to the right of the chose button. In this example, we accept the default values. The default version does perform some pruning but does not perform error pruning.

Step4: Under the “text” options in the main panel. We select the 10-fold cross validation as our evaluation approach. Since we don’t have separate evaluation data set, this is necessary to get a reasonable idea of accuracy of generated model.

Step-5: We now click “start” to generate the model. The ASCII version of the tree as well as evaluation statistic will appear in the right panel when the model construction is complete.

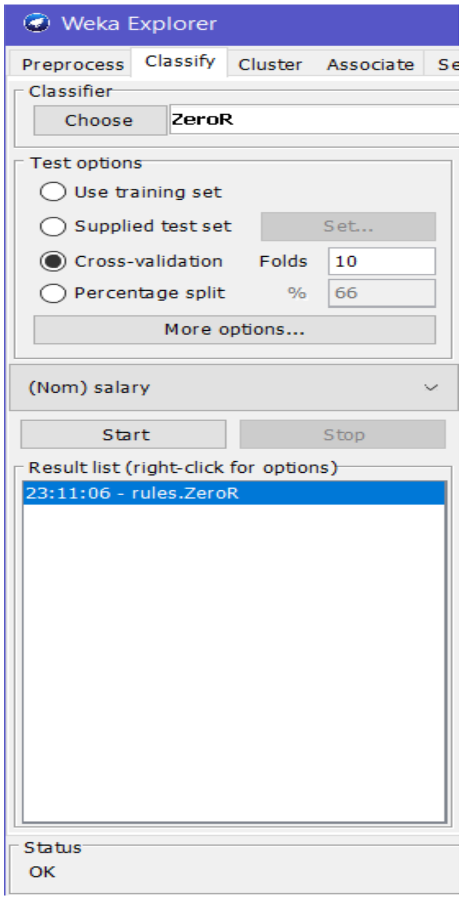
Step-6: Note that the classification accuracy of model is about 78.57%. This indicates that we may find more work. (Either in preprocessing or in selecting current parameters for the classification)

Step-7: Now Weka also lets us a view a graphical version of the classification tree. This can be done by right clicking the last result set and selecting “visualize tree” from the pop-up menu.

Step-8: We will use our model to classify the new instances.

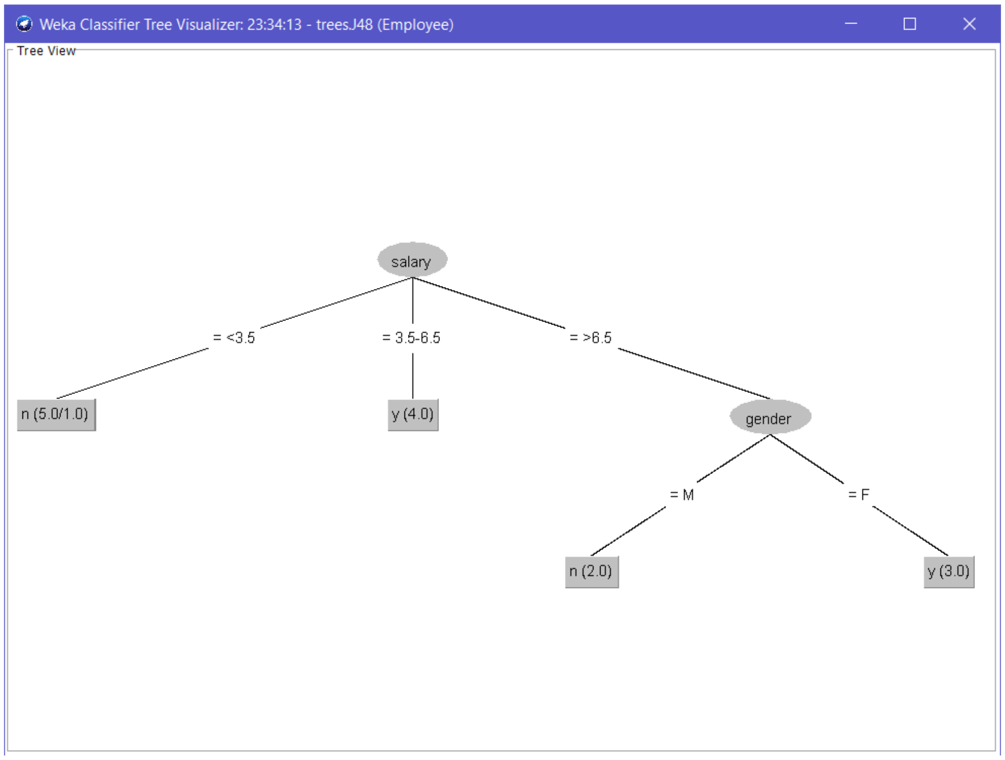
Step-9: In the main panel under “text” options click the “supplied test set” radio button and then click the “set” button. This wills pop-up a window which will allow you to open the file containing test instances.

Classification Window:



```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     Employee
Instances:    14
Attributes:   5
              salary
              income
              work_ex
              gender
              pentual
Test mode:    10-fold cross-validation
=== Classifier model (full training set) ===
J48 pruned tree
------------------

salary = <3.5: n (5.0/1.0)
salary = 3.5-6.5: y (4.0)
salary = >6.5
|   gender = M: n (2.0)
|   gender = F: y (3.0)

Number of Leaves  :     4

Size of the tree :      6

Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          11               78.5714 %
Incorrectly Classified Instances         3               21.4286 %
Kappa statistic                          0.5532
Mean absolute error                      0.25
Root mean squared error                  0.4058
Relative absolute error                 49.5283 %
Root relative squared error             79.6745 %
Total Number of Instances               14

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.875    0.333    0.778      0.875    0.824      0.559    0.854     0.919     y
                 0.667    0.125    0.800      0.667    0.727      0.559    0.854     0.727     n
Weighted Avg.    0.786    0.244    0.787      0.786    0.782      0.559    0.854     0.837

=== Confusion Matrix ===

 a b   <-- classified as
 7 1 | a = y
 2 4 | b = n
```

Classification Tree:



EXPERIMENT - 5

Aim: Implementation of Clustering technique on ARFF files using WEKA.

Theory:
This experiment illustrates the use of simple k-mean clustering with Weka explorer. The sample data set used for this example is based on the iris data available in ARFF format. This document assumes that appropriate preprocessing has been performed. This iris dataset includes 150 instances.

Steps involved in this Experiment

Step 1: Run the Weka explorer and load the data file iris.arff in preprocessing interface.

Step 2: In order to perform clustering, select the 'cluster' tab in the explorer and click on the choose button. This step results in a dropdown list of available clustering algorithms.

Step 3: In this case we select 'Simple K-Means'.

Step 4: Next click in text button to the right of the choose button to get popup window shown in the screenshots. In this window we enter six on the number of clusters and we leave the value of the seed on as it is. The seed value is used in generating a random number which is used for making the internal assignments of instances of clusters.

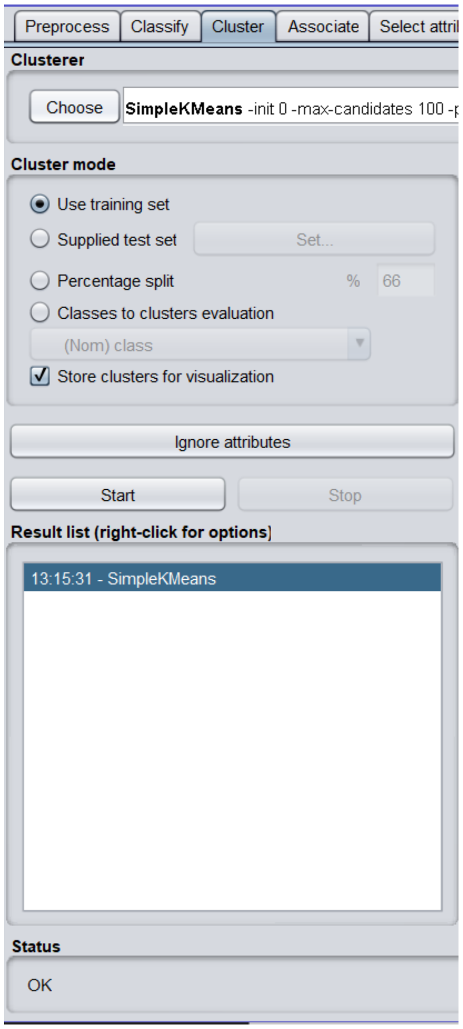
Step 5: Once of the option have been specified. We run the clustering algorithm there we must make sure that they are in the 'cluster mode' panel. The use of training set option is selected and then we click 'start' button. This process and resulting window are shown in the following screenshots.

Step 6 : The result window shows the centroid of each cluster as well as statistics on the number and the percent of instances assigned to different clusters. Here clusters centroid are means vectors for each clusters. This clusters can be used to characterized the cluster. For eg, the centroid of cluster1 shows the class iris.versicolor mean value of the sepal length is 5.4706, sepal width 2.4765, petal width 1.1294, petal length 3.7941.

Step 7: Another way of understanding characteristic of each cluster through visualization, we can do this, try right clicking the result set on the result. List panel and selecting the visualize cluster assignments.

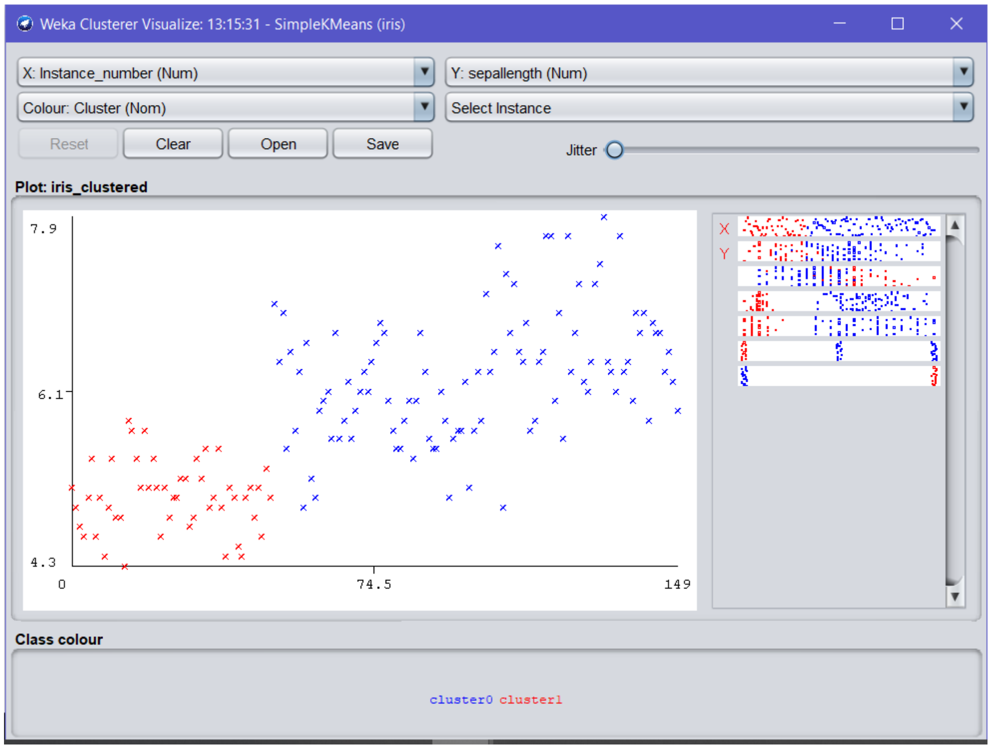
Step 8: From the above visualization, we can understand the distribution of sepal length and petal length in each cluster. For instance, for each cluster is dominated by petal length. In this case by changing the color dimension to other attributes we can see their distribution with in each of the cluster. We can assure that resulting dataset which included each instance along with its assign cluster. To do so we click the save button in the visualization window and save the result iris k-mean. The top portion of this file is shown in the following figure.

The following screenshot shows the clustering rules that were generated when Simple-K-Means algorithm is applied on the given dataset.



```
=== Run information ===

Scheme:       weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning
Relation:     iris
Instances:    150
Attributes:   5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:    evaluate on training data
=== Clustering model (full training set) ===
kMeans
======
Number of iterations: 7
Within cluster sum of squared errors: 62.1436882815797

Initial starting points (random):

Cluster 0: 6.1,2.9,4.7,1.4,Iris-versicolor
Cluster 1: 6.2,2.9,4.3,1.3,Iris-versicolor

Missing values globally replaced with mean/mode

Final cluster centroids:
                                      Cluster#
Attribute                Full Data           0              1
                           (150.0)     (100.0)         (50.0)
==================================================================
sepallength                 5.8433       6.262          5.006
sepalwidth                   3.054       2.872          3.418
petallength                 3.7587       4.906          1.464
petalwidth                  1.1987       1.676          0.244
class                  Iris-setosa Iris-versicolor  Iris-setosa



Time taken to build model (full training data) : 0.02 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       100 ( 67%)
1        50 ( 33%)
```

Visualization:



EXPERIMENT - 6

Aim: Implementation of Association Rule technique on ARFF files using WEKA.

Theory:
This experiment illustrates some of the basic elements of association rule mining using WEKA. The sample dataset used for this example is contactlenses.arff.

Step 1: Open the data file in Weka Explorer. It is presumed that the required data fields have been discretized. In this example it is age attribute.

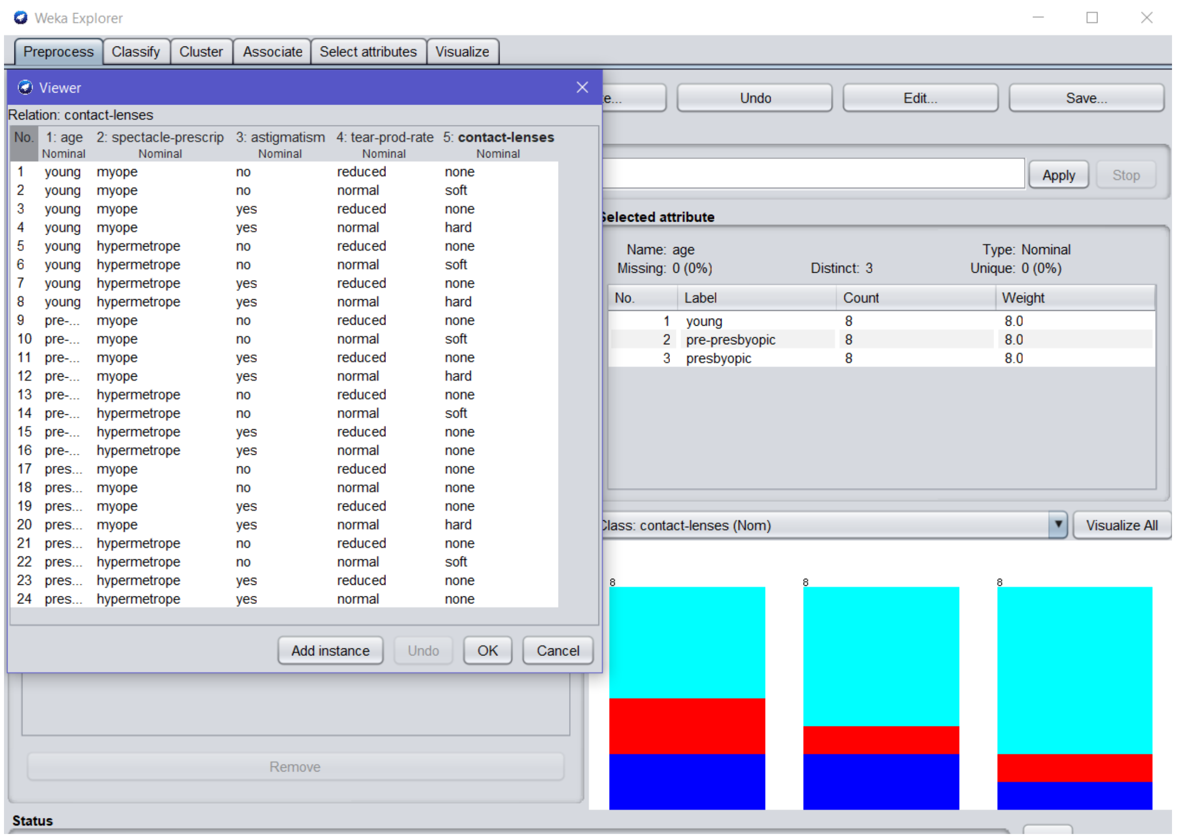
Step 2: Clicking on the associate tab will bring up the interface for association rule algorithm.

Step 3: We will use apriori algorithm. This is the default algorithm.

Step 4: In order to change the parameters for the run (example support, confidence etc.) we click on the text box immediately to the right of the choose button.

The following screenshot shows the association rules that were generated when apriori algorithm is applied on the given dataset.

Dataset contactlenses.arff



Relation: contact-lenses

No.	1: age Nominal	2: spectacle-prescrip Nominal	3: astigmatism Nominal	4: tear-prod-rate Nominal	5: contact-lenses Nominal
1	young	myope	no	reduced	none
2	young	myope	no	normal	soft
3	young	myope	yes	reduced	none
4	young	myope	yes	normal	hard
5	young	hypermetrope	no	reduced	none
6	young	hypermetrope	no	normal	soft
7	young	hypermetrope	yes	reduced	none
8	young	hypermetrope	yes	normal	hard
9	pre-...	myope	no	reduced	none
10	pre-...	myope	no	normal	soft
11	pre-...	myope	yes	reduced	none
12	pre-...	myope	yes	normal	hard
13	pre-...	hypermetrope	no	reduced	none
14	pre-...	hypermetrope	no	normal	soft
15	pre-...	hypermetrope	yes	reduced	none
16	pre-...	hypermetrope	yes	normal	none
17	pres...	myope	no	reduced	none
18	pres...	myope	no	normal	none
19	pres...	myope	yes	reduced	none
20	pres...	myope	yes	normal	hard
21	pres...	hypermetrope	no	reduced	none
22	pres...	hypermetrope	no	normal	soft
23	pres...	hypermetrope	yes	reduced	none
24	pres...	hypermetrope	yes	normal	none

Selected attribute

Name: age Type: Nominal
Missing: 0 (0%) Distinct: 3 Unique: 0 (0%)

No.	Label	Count	Weight
1	young	8	8.0
2	pre-presbyopic	8	8.0
3	presbyopic	8	8.0

Class: contact-lenses (Nom)

```
=== Run information ===
Scheme:       weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Relation:     contact-lenses
Instances:    24
Attributes:   5
              age
              spectacle-prescrip
              astigmatism
              tear-prod-rate
              contact-lenses
=== Associator model (full training set) ===
Apriori
=======
Minimum support: 0.2 (5 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 16

Generated sets of large itemsets:

Size of set of large itemsets L(1): 11

Size of set of large itemsets L(2): 21

Size of set of large itemsets L(3): 6

Best rules found:

 1. tear-prod-rate=reduced 12 ==> contact-lenses=none 12    <conf:(1)> lift:(1.6) lev:(0.19) [4] conv:(4.5)
 2. spectacle-prescrip=myope tear-prod-rate=reduced 6 ==> contact-lenses=none 6    <conf:(1)> lift:(1.6) lev:(0.09) [2] conv:(2.25)
 3. spectacle-prescrip=hypermetrope tear-prod-rate=reduced 6 ==> contact-lenses=none 6    <conf:(1)> lift:(1.6) lev:(0.09) [2] conv:(2.25)
 4. astigmatism=no tear-prod-rate=reduced 6 ==> contact-lenses=none 6    <conf:(1)> lift:(1.6) lev:(0.09) [2] conv:(2.25)
 5. astigmatism=yes tear-prod-rate=reduced 6 ==> contact-lenses=none 6    <conf:(1)> lift:(1.6) lev:(0.09) [2] conv:(2.25)
 6. contact-lenses=soft 5 ==> astigmatism=no 5    <conf:(1)> lift:(2) lev:(0.1) [2] conv:(2.5)
 7. contact-lenses=soft 5 ==> tear-prod-rate=normal 5    <conf:(1)> lift:(2) lev:(0.1) [2] conv:(2.5)
 8. tear-prod-rate=normal contact-lenses=soft 5 ==> astigmatism=no 5    <conf:(1)> lift:(2) lev:(0.1) [2] conv:(2.5)
 9. astigmatism=no contact-lenses=soft 5 ==> tear-prod-rate=normal 5    <conf:(1)> lift:(2) lev:(0.1) [2] conv:(2.5)
10. contact-lenses=soft 5 ==> astigmatism=no tear-prod-rate=normal 5    <conf:(1)> lift:(4) lev:(0.16) [3] conv:(3.75)
```

EXPERIMENT – 7

Aim: Implementation of Visualization technique on ARFF files using WEKA.

Data Visualization: Data Visualization is the process of extracting and visualizing the data in a very clear and understandable way without any form of reading or writing by displaying the results in the form of pie charts, bar graphs, statistical representation and through graphical forms as well. It enables decision makers to see analytics presented visually, so they can grasp difficult concepts or identify new patterns. With interactive visualization, you can take the concept a step further by using technology to drill down into charts, graphs for more detail, interactively changing what data you see, and how it is processed.

Weka: Weka makes learning applied machine learning easy, efficient, and fun. It is a GUI tool that allows you to load datasets, run algorithms and design and run experiments with results statistically robust enough to publish.

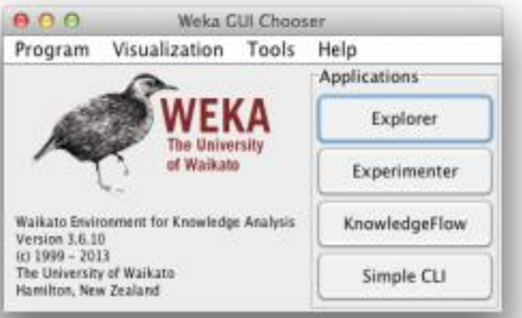
Data Visualization in Weka: When attributes are numeric, we can create a scatter plot of one attribute against another. This is useful as it can highlight any patterns in the relationship between the attributes, such as positive or negative correlations.

- We can create scatter plots for all pairs of input attributes. This is called a scatter plot matrix and reviewing it before modeling your data can shed more light on further preprocessing techniques that you could investigate.
- Weka provides a scatter plot matrix for review by default in the **"Visualize"** tab. All combinations of attributes are plotted in a systematic way.

2. Start Weka

Start Weka. This may involve finding it in program launcher or double clicking on the weka.jar file. This will start the Weka GUI Chooser.

The Weka GUI Chooser lets you choose one of the Explorer, Experimenter, KnowledgeFlow and the Simple CLI (command line interface).



Waikato Environment for Knowledge Analysis
Version 3.6.10
(c) 1999 - 2013
The University of Waikato
Hamilton, New Zealand

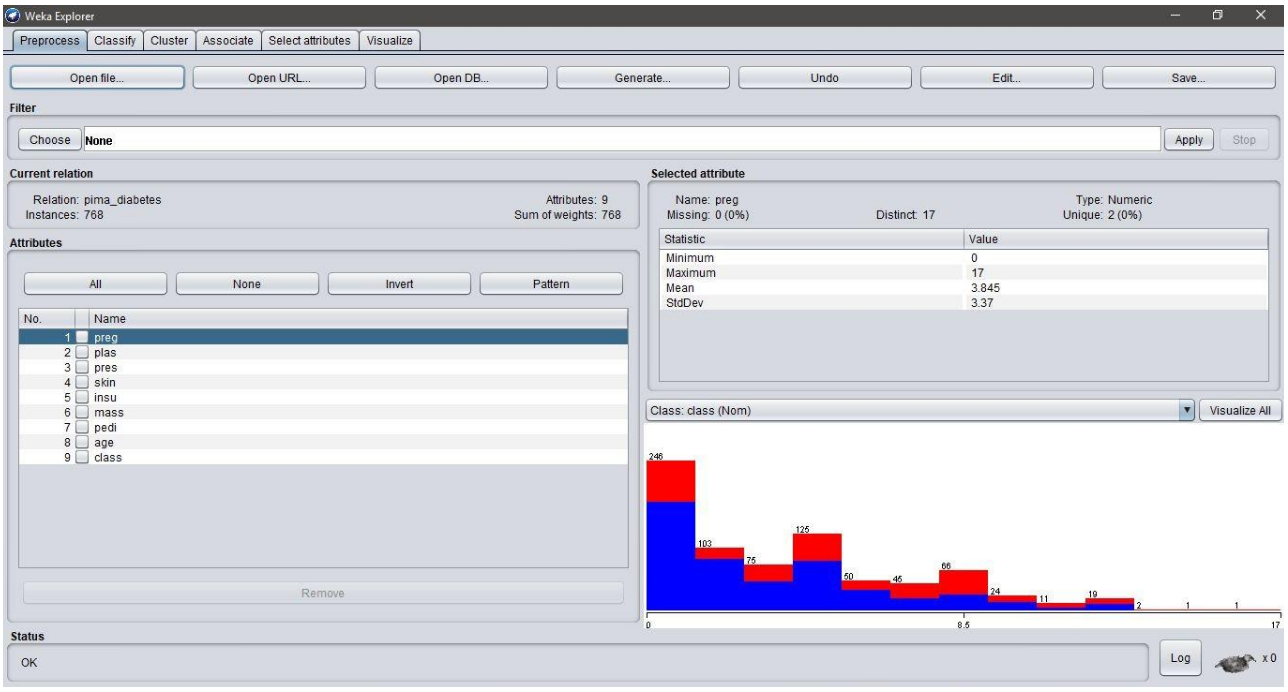
Fig.1. Weka GUI Chooser

Click the "***Explorer***" button to launch the Weka Explorer.

3. Open the data/ diabetes.arff Dataset

Click the "***Open file…***" button to open a data set and double click on the "***data***" directory. Weka provides a number of small common machine learning datasets that you can use to practice on.

Select the "***diabetes.arff***" file to load the dataset.



Filter: None

Current relation
Relation: pima_diabetes
Instances: 768
Attributes: 9
Sum of weights: 768

No.	Name
1	preg
2	plas
3	pres
4	skin
5	insu
6	mass
7	pedi
8	age
9	class

Selected attribute
Name: preg
Missing: 0 (0%)
Distinct: 17
Type: Numeric
Unique: 2 (0%)

Statistic	Value
Minimum	0
Maximum	17
Mean	3.845
StdDev	3.37

Class: class (Nom)

Status
OK

Fig.2. Weka Explorer Interface with the diabetes.arff dataset loaded

The button **"Visualize All"** will bring up a screen showing all distributions at once as in the picture below.

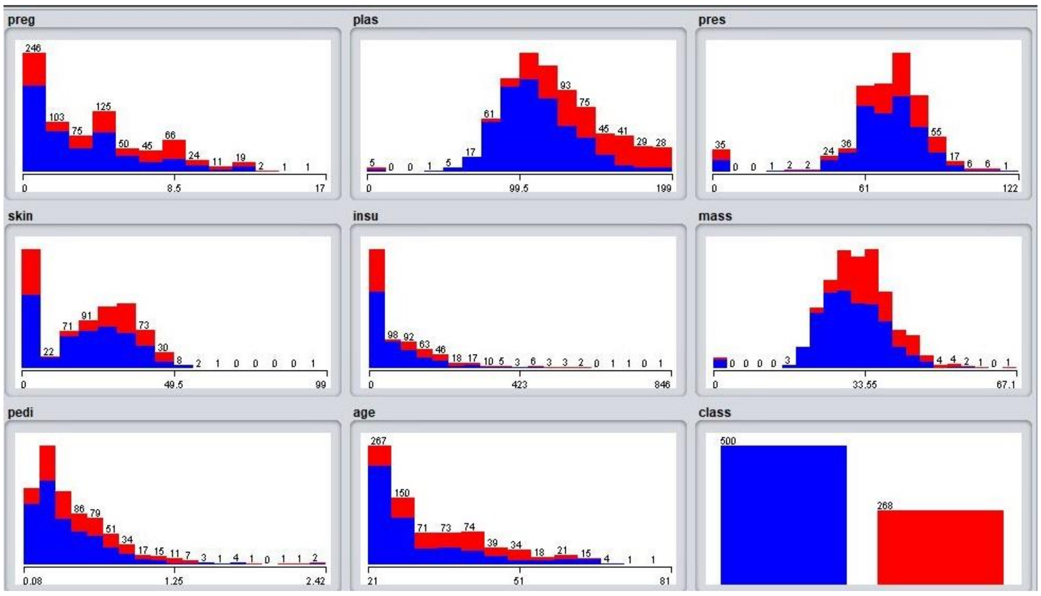
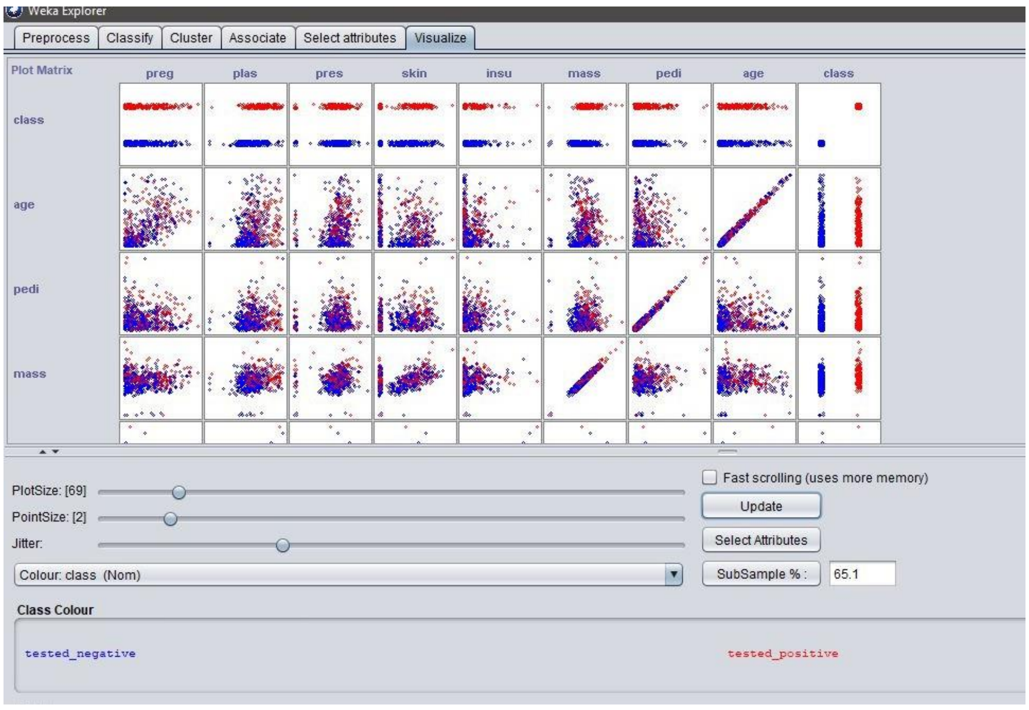


Fig.3. Weka Explorer Interface with the diabetes.arff dataset loaded and showing all the distributions at once

4. Select the "Visualize" Tab

Continuing on from the previous section with the Pima Indians dataset loaded, click the "Visualize" tab, and make the window large enough to review all of the individual scatter plots.



PlotSize: [69]
PointSize: [2]
Jitter:
Colour: class (Nom)
SubSample % : 65.1

Fig.4. Visualizing the data set

- The dots in the scatter plots are colored by their class value. It is good to look for trends or patterns in the dots, such as clear separation of the colors.
- Clicking on a plot provides a new window with the plot that you can further play with.

- The controls at the bottom of the screen helps to increase the size of the plots, increase the size of the dots and add jitter.
- This last point about jitter is useful when you have a lot of dots overlaying each other and it is hard to see what is going on. Jitter will add some random noise to the data in the plots, spread out the points a bit and help you see what is going on.
- After making changes to these controls, click the "Update" button to apply the changes.

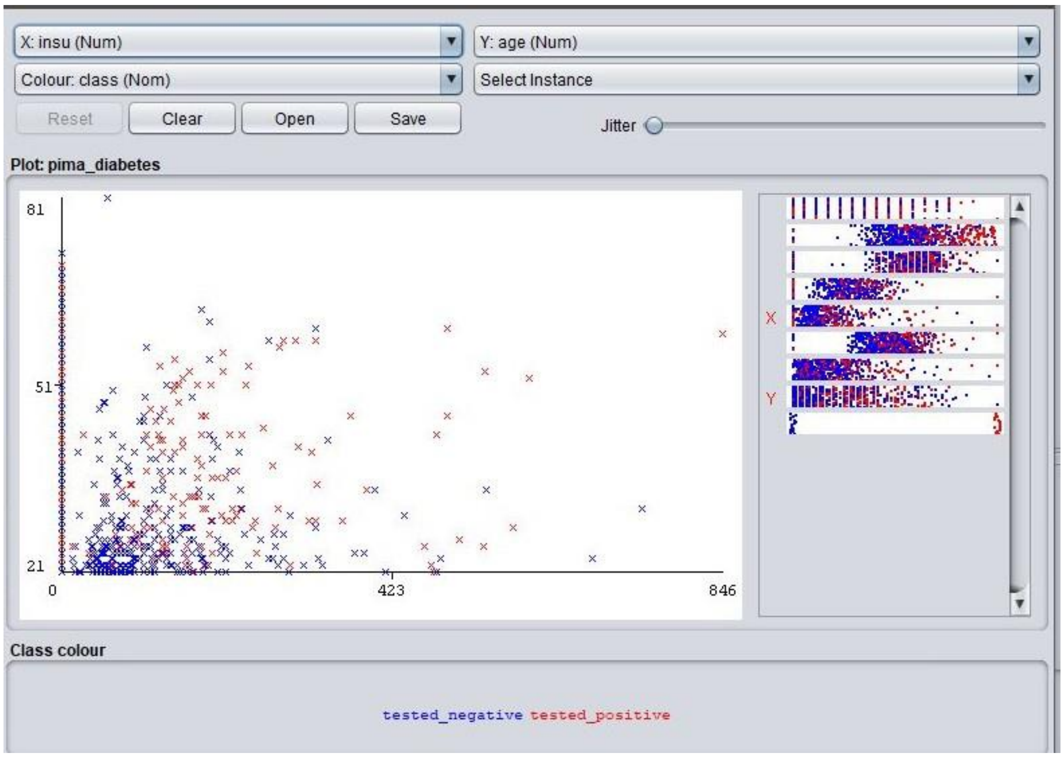

X: insu (Num)
Y: age (Num)
Colour: class (Nom)
Select Instance
Jitter
Plot: pima_diabetes
Class colour
tested_negative tested_positive

Fig.5. Individual Visualization of plotted graph (insu v/s age)

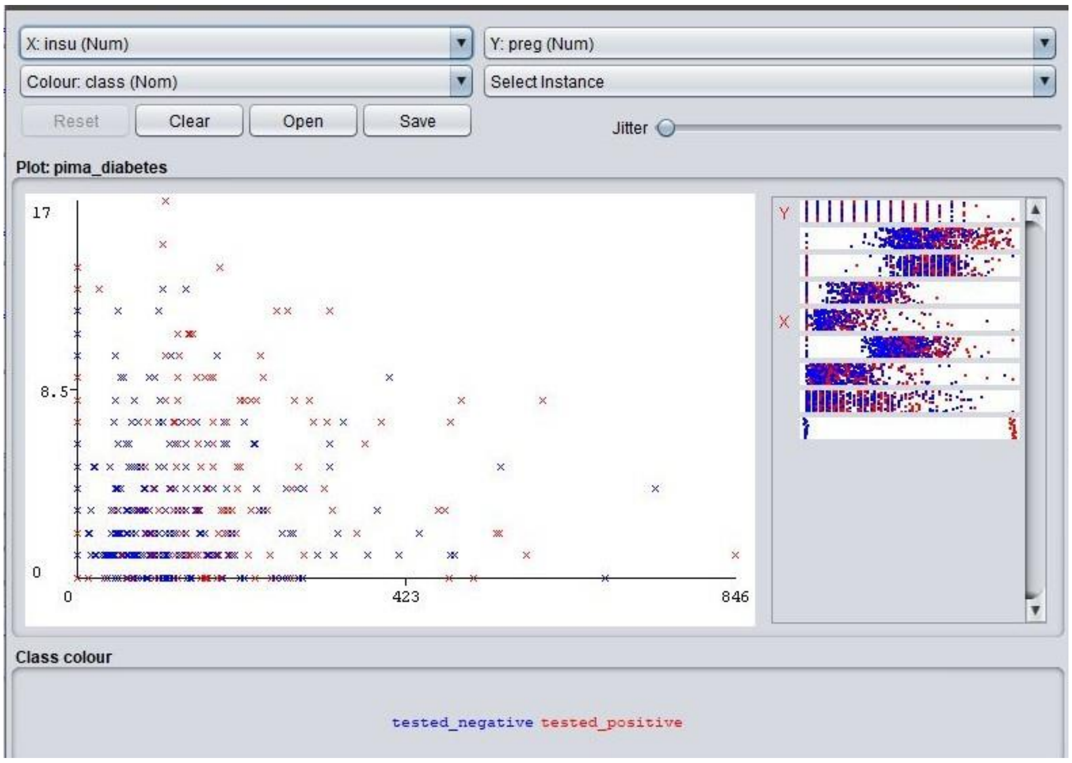

X: insu (Num)
Y: preg (Num)
Colour: class (Nom)
Select Instance
Jitter
Plot: pima_diabetes
Class colour
tested_negative tested_positive

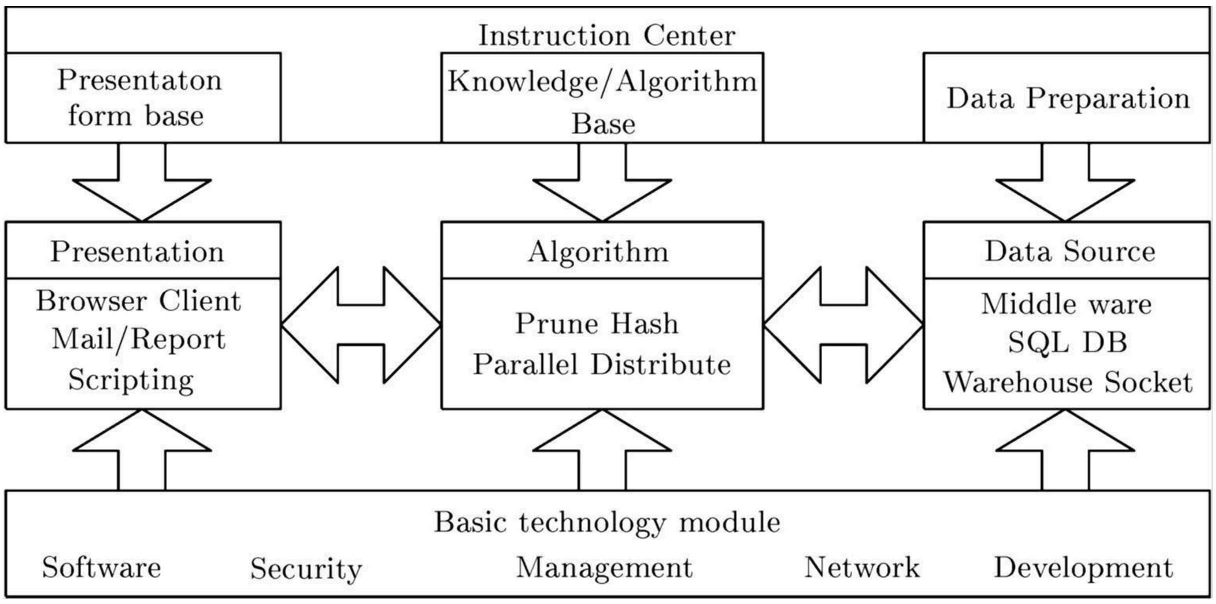
Fig.6. Individual Visualization of plotted graph (Insu v/s preg)

EXPERIMENT – 8

Aim: Study of ARMINER tool.

Theory:
ARMiner is a client-server data mining application specialized in finding association rules. The name ARMiner comes from Association Rules Miner. ARMiner has been written in Java and it is distributed under the GNU General Public License. ARMiner has been developed at UMass/Boston as a Software Engineering project in Spring 2000. It is useful to people interested in making decisions based on data analysis, to students interested in data mining, and to researchers that want to benchmark and test new algorithms.

Architecture:



Features:

i. Mining of databases for AR using a variety of parameters to adjust and ease the process of mining
ii. Comparison of the performance of different algorithms according to two criteria: time and number of database scans
iii. Dynamic addition of databases
iv. Generation of synthetic data for tests
v. Dynamic addition of algorithms
vi. Provides a comprehensive user management system in order to restrict or grant user access to algorithms

There are four important objects in the system:
Users:
- name, password, permissions
- Permissions: add algorithms, add databases, add groups

Groups:
- are used to grant access to an algorithm or database to a group of users
- special group admin indicates administrators

Algorithms and Databases:

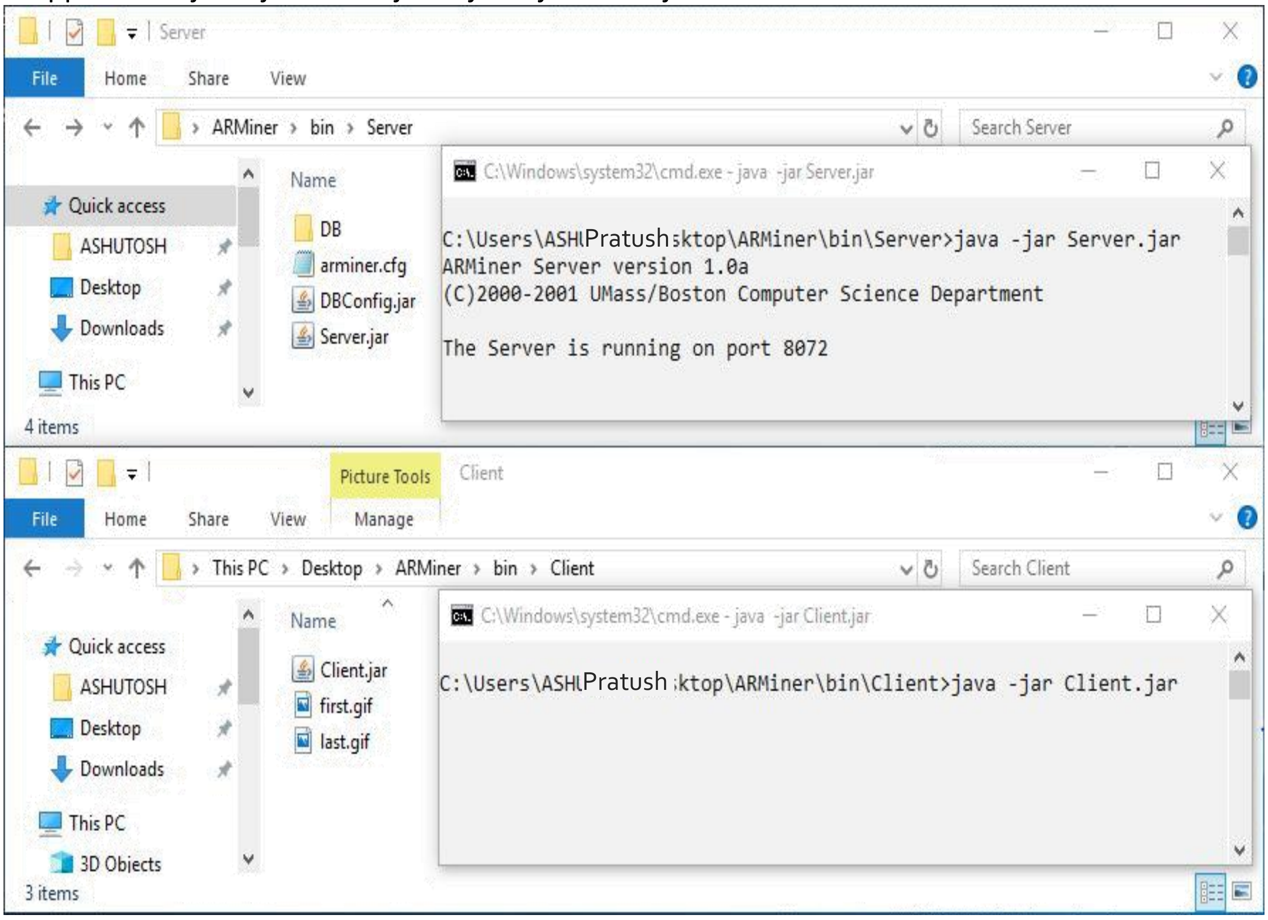
- owner, group
- “all” is a special group to which all users belong;
- “admin” is a special member of admin, a superadmin in fact; anonymous is a member of “all”. All of these groups and users cannot be deleted.
- Administrators have no restrictions.

Security:

Since passwords, databases, and communication between server and client are unencrypted, we depend on the security of a socket communication and on that of the OS on which the server runs.
Otherwise the server has been designed and implemented such that a malicious client could not gain access or corrupt its information.

<u>WORKING:</u>

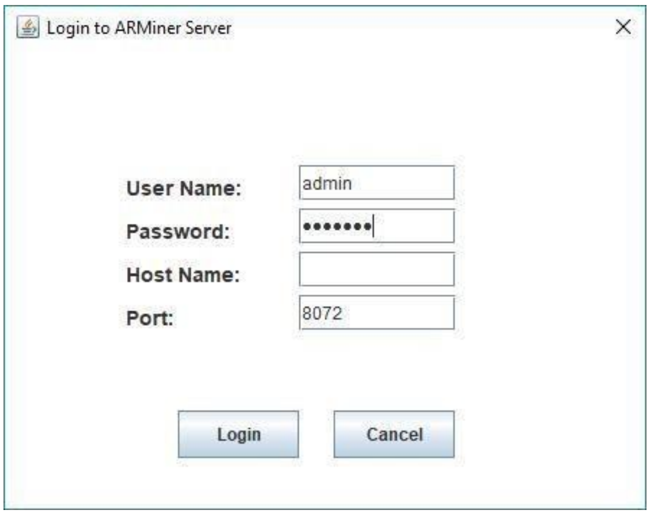
Step 1: First launch the server, and then, launch the client
application: java -jar Server.jar -> java -jar Client.jar



```
C:\Users\ASHIPratush sktop\ARMiner\bin\Server>java -jar Server.jar
ARMiner Server version 1.0a
(C)2000-2001 UMass/Boston Computer Science Department

The Server is running on port 8072
```

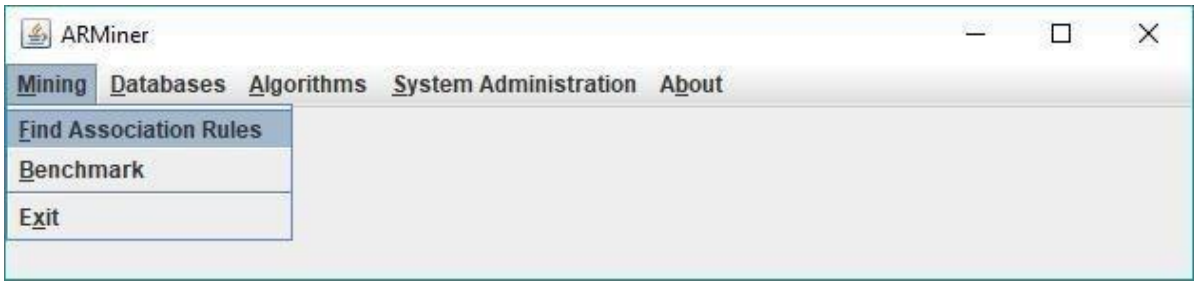
```
C:\Users\ASHIPratush ktop\ARMiner\bin\Client>java -jar Client.jar
```

Step 2: After launching client application, login dialog box appears. Login as *admin* with password *renimra*.

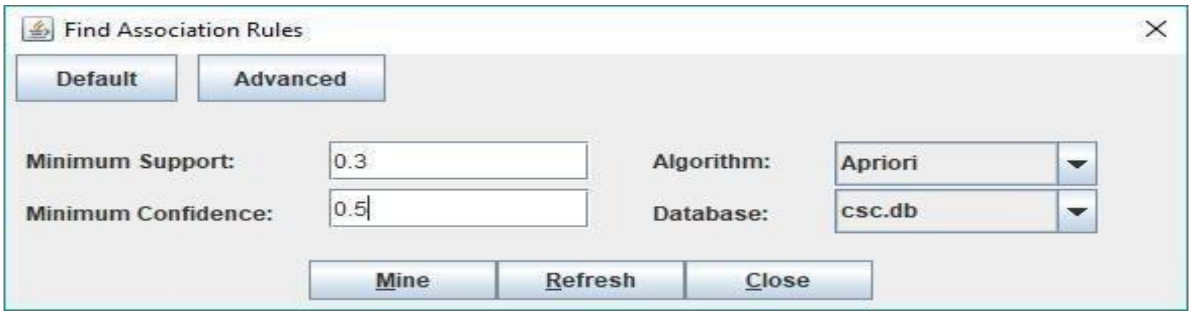


User Name: admin
Password: •••••••
Host Name:
Port: 8072

Step 3: Now, the ARMiner main interface is shown. Select "Mining" tab and click on "Find Association Rules"

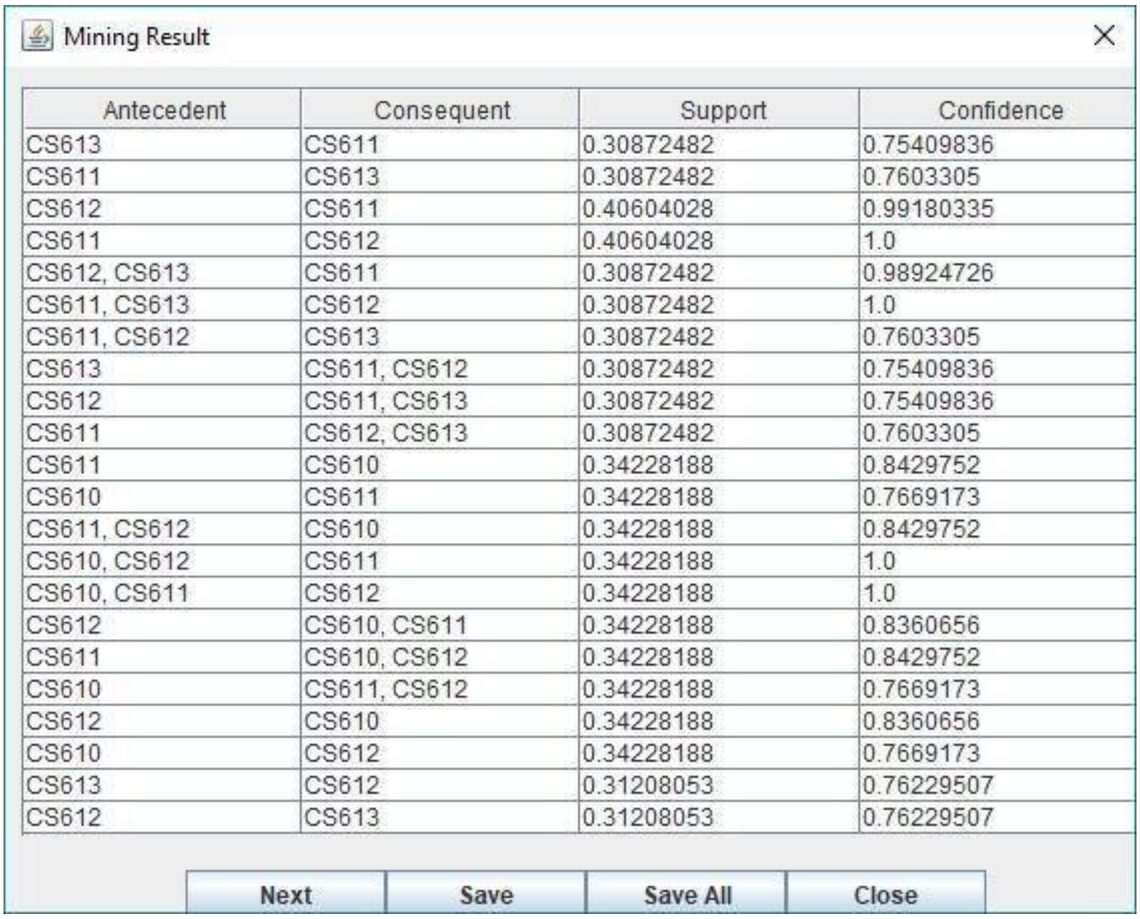


Step 4: Now, a new window titled "Find Association Rules" will appear. Enter the values for Minimum Support and Minimum Confidence. Here I have entered their values 0.3 and 0.5 respectively. Also we can choose the algorithm and select the database from their respective dropdowns. Then, click on "Mine" button.



Minimum Support: 0.3
Minimum Confidence: 0.5
Algorithm: Apriori
Database: csc.db

Step 5: On clicking "Mine" button in previous step, the Mining Result window will be shows as follows displaying the mining result.



Antecedent	Consequent	Support	Confidence
CS613	CS611	0.30872482	0.75409836
CS611	CS613	0.30872482	0.7603305
CS612	CS611	0.40604028	0.99180335
CS611	CS612	0.40604028	1.0
CS612, CS613	CS611	0.30872482	0.98924726
CS611, CS613	CS612	0.30872482	1.0
CS611, CS612	CS613	0.30872482	0.7603305
CS613	CS611, CS612	0.30872482	0.75409836
CS612	CS611, CS613	0.30872482	0.75409836
CS611	CS612, CS613	0.30872482	0.7603305
CS611	CS610	0.34228188	0.8429752
CS610	CS611	0.34228188	0.7669173
CS611, CS612	CS610	0.34228188	0.8429752
CS610, CS612	CS611	0.34228188	1.0
CS610, CS611	CS612	0.34228188	1.0
CS612	CS610, CS611	0.34228188	0.8360656
CS611	CS610, CS612	0.34228188	0.8429752
CS610	CS611, CS612	0.34228188	0.7669173
CS612	CS610	0.34228188	0.8360656
CS610	CS612	0.34228188	0.7669173
CS613	CS612	0.31208053	0.76229507
CS612	CS613	0.31208053	0.76229507