



**COLLEGE OF ENGINEERING & TECHNOLOGY**

**LABORATORY MANUAL**

**DATA MINING AND BUSINESS INTELLIGENCE**

**SUBJECT CODE: 2170715**

**COMPUTER SCIENCE ENGINEERING  
DEPARTMENT**

**B.E. 7<sup>TH</sup> SEMESTER**

**NAME:** \_\_\_\_\_

**ENROLLMENT NO:** \_\_\_\_\_

**BATCH NO:** \_\_\_\_\_

**YEAR:** \_\_\_\_\_

**Amiraj College of Engineering and Technology**

Nr.Tata Nano Plant, Khoraj, Sanand, Ahmedabad.



**COLLEGE OF ENGINEERING & TECHNOLOGY**

**Amiraj College of Engineering and Technology**

Nr.Tata Nano Plant, Khoraj, Sanand, Ahmedabad.

**CERTIFICATE**

*This is to certify that Mr. / Ms. \_\_\_\_\_ Of  
class \_\_\_\_\_ Enrolment No \_\_\_\_\_ has  
Satisfactorily completed the course in \_\_\_\_\_ as by  
the Gujarat Technological University for \_\_\_\_ Year (B.E.) semester \_\_\_\_ of Computer  
Science Engineering in the Academic year \_\_\_\_\_.*

*Date of Submission:-*

**Faculty Name and Signature  
PROF.NENSI KANSAGARA**

**Head of Department  
(COMPUTER)**



**COMPUTER ENGINEERING DEPARTMENT**  
**B.E. 7<sup>TH</sup> SEMESTER**  
**SUBJECT: DATA MINING AND BUSINESS**  
**INTELLIGENCE**  
**SUBJECT CODE: 2170715**  
**List of Experiment**

<b>Sr. No.</b>	<b>Title</b>	<b>Date of Performance</b>	<b>Date of submission</b>	<b>Sign</b>	<b>Remark</b>
1.	Write a note on KDD process				
2	Write a program to implement apriori algorithm in c language				
3	Demonstration of Association rule process on dataset contactlenses.arff using apriori algorithm				
4	Demonstration of classification rule process on dataset student.arff using j48 algorithm				
5	Demonstration of Association rule process on dataset supermarket.arff using apriori algorithm				
6	Study about different tools: Weka, XLMiner, Hadoop				
7	Introduction to the classification of Mining techniques				
8	Introduction to Decision Tree and Neural Networks				

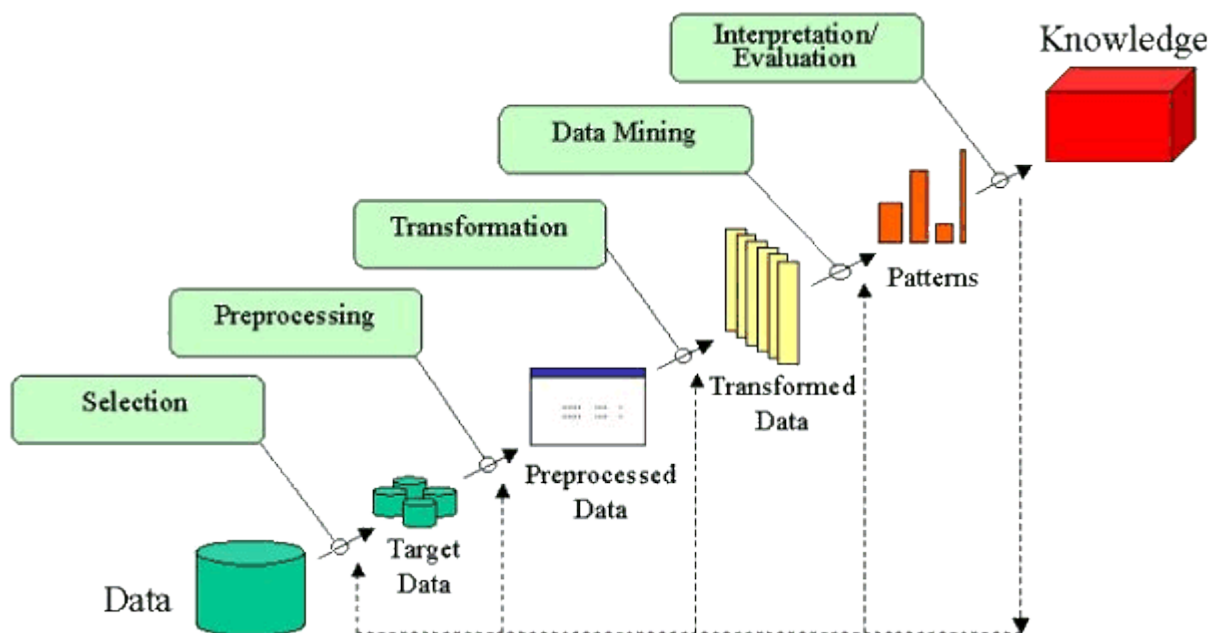
## PRACTICAL 1

### AIM: Write a note on KDD process

The term *Knowledge Discovery in Databases*, or KDD for short, refers to the broad process of finding knowledge in data, and emphasizes the "high-level" application of particular data mining methods. It is of interest to researchers in [machine learning](#), pattern recognition, databases, statistics, artificial intelligence, knowledge acquisition for expert systems, and data visualization. The unifying goal of the KDD process is to extract knowledge from data in the context of large databases.

It does this by using [data mining methods](#) (algorithms) to extract (identify) what is deemed knowledge, according to the specifications of measures and thresholds, using a database along with any required preprocessing, subsampling, and transformations of that database.

### An Outline of the Steps of the KDD Process



- Data Cleaning – In this step, the noise and inconsistent data is removed.
- Data Integration – In this step, multiple data sources are combined.
- Data Selection – In this step, data relevant to the analysis task are retrieved from the database.
- Data Transformation – In this step, data is transformed or consolidated into forms appropriate for mining by performing summary or aggregation operations.

- Data Mining – In this step, intelligent methods are applied in order to extract data patterns.
- Pattern Evaluation – In this step, data patterns are evaluated.
- Knowledge Presentation – In this step, knowledge is represented.

The overall process of finding and interpreting patterns from data involves the repeated application of the following steps:

1. Developing an understanding of
  - the application domain
  - the relevant prior knowledge
  - the goals of the end-user
2. Creating a target data set: selecting a data set, or focusing on a subset of variables, or data samples, on which discovery is to be performed.
3. Data cleaning and preprocessing.
  - Removal of noise or outliers.
  - Collecting necessary information to model or account for noise.
  - Strategies for handling missing data fields.
  - Accounting for time sequence information and known changes.
4. Data reduction and projection.
  - Finding useful features to represent the data depending on the goal of the task.
  - Using dimensionality reduction or transformation methods to reduce the effective number of variables under consideration or to find invariant representations for the data.
5. Choosing the [data mining task](#).
  - Deciding whether the goal of the KDD process is classification, regression, clustering, etc.
6. Choosing the [data mining algorithm\(s\)](#).
  - Selecting method(s) to be used for searching for patterns in the data.
  - Deciding which models and parameters may be appropriate.
  - Matching a particular data mining method with the overall criteria of the KDD process.
7. Data mining.
  - Searching for patterns of interest in a particular representational form or a set of such representations as classification rules or trees, regression, clustering, and so forth.
8. Interpreting mined patterns.
9. Consolidating discovered knowledge.

## PRACTICAL 2

**AIM: Write a program to implement apriori algorithm in c language**

```
#include<stdio.h>
#include<stdlib.h>
int main(int argc, char *argv[])
{
    FILE *fin;
    int i,cols,rows,*count;
    char val;
    if(argc!=2)
    {
        return 1;
    }
    fin = fopen(argv[1],"r");

//finding the number of cols
    cols=0;
    fscanf(fin,"%c",&val);
    while(1)
    {
        if(val=='\n')
            break;
        if(val!=' ')
            cols++;
        fscanf(fin,"%c",&val);
    }
    printf("\nNumber of columns = %d\n",cols);
    fclose(fin);
    fin = fopen(argv[1],"r");
//Generation of 1 item frequent items

    count = (int*)malloc(sizeof(int)*cols);

    for(i=0;i<cols;i++)
    {
        count[i] =0;
    }

    while(!feof(fin))
```

```
{
for(i=0;i<cols;i++)
{
fscanf(fin,"%c",&val);
if(val=='1')
count[i]++;
fscanf(fin,"%c",&val);
}
}
//Generation of 1-item frequent sets completed!!

printf("\n 1-item frequent item sets..\n");
for(i=0;i<cols;i++)
{
printf("\n%d -> %d",i+1,count[i]);
}
fclose(fin);
return 0;
}
```

## PRACTICAL 3

### AIM: Demonstration of Association rule process on dataset contactlenses.arff using apriori algorithm

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

Step1: 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. Step2: Clicking on the associate tab will bring up the interface for association rule algorithm.

Step3: We will use apriori algorithm. This is the default algorithm.

Step4: 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.

#### Dataset contactlenses.arff

The screenshot shows the WEKA Explorer interface. The 'Viewer' window displays the 'contactlenses' dataset with the following data:

No.	age	spectacle-prescrip	astigmatism	tear-prod-rate	contact-lenses
	Nominal	Nominal	Nominal	Nominal	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-pr...	myope	no	reduced	none
10	pre-pr...	myope	no	normal	soft
11	pre-pr...	myope	yes	reduced	none
12	pre-pr...	myope	yes	normal	hard
13	pre-pr...	hypermetrope	no	reduced	none
14	pre-pr...	hypermetrope	no	normal	soft
15	pre-pr...	hypermetrope	yes	reduced	none
16	pre-pr...	hypermetrope	yes	normal	none
17	presb...	myope	no	reduced	none
18	presb...	myope	no	normal	none
19	presb...	myope	yes	reduced	none
20	presb...	myope	yes	normal	hard
21	presb...	hypermetrope	no	reduced	none
22	presb...	hypermetrope	no	normal	soft
23	presb...	hypermetrope	yes	reduced	none
24	presb...	hypermetrope	yes	normal	none

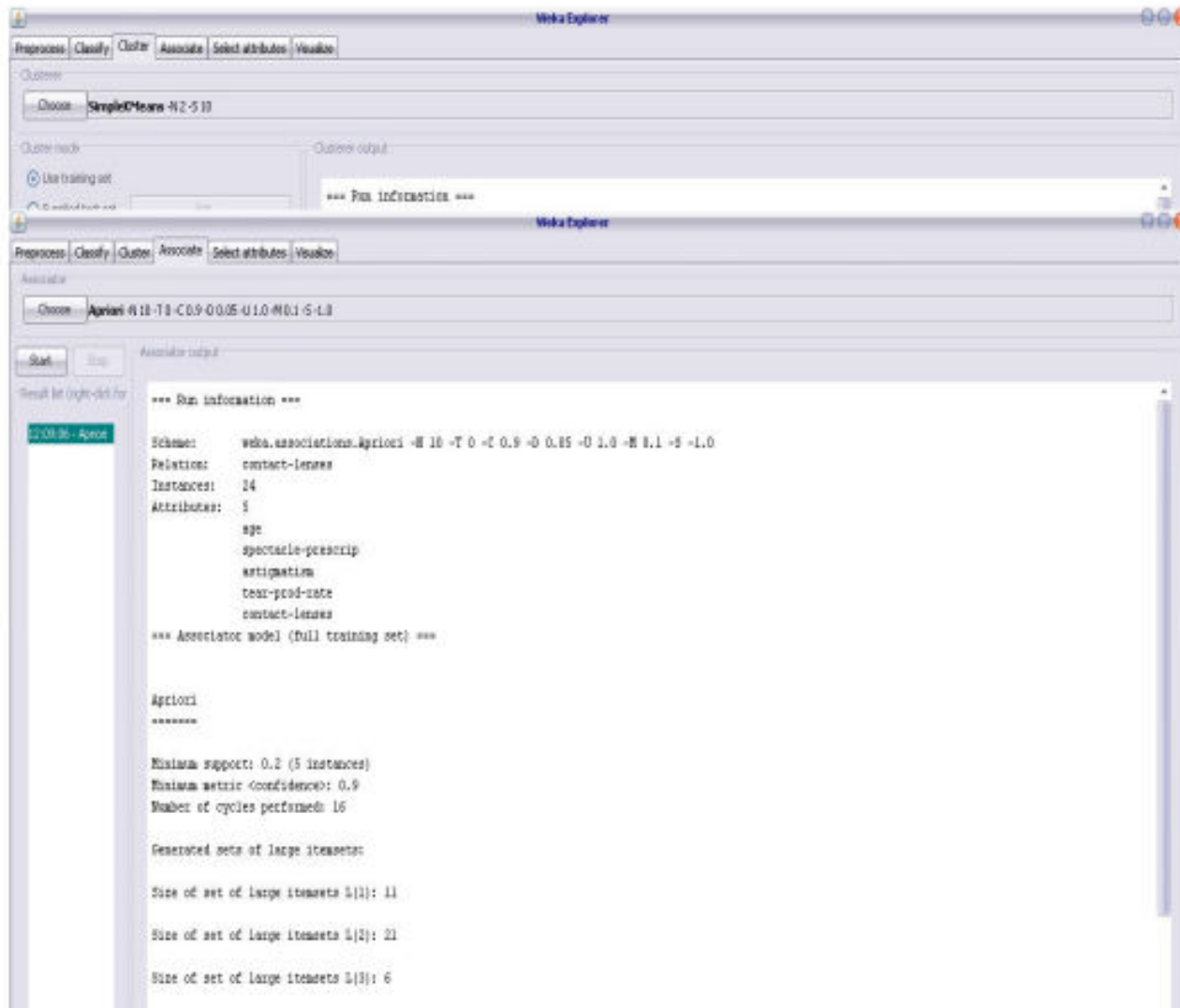
The 'Associate' window shows the 'age' attribute selected. The 'Selected attribute' section displays: Name: age, Missing: 0 (0%), Distinct: 3, Type: Nominal, Unique: 0 (0%). Below this is a table with 3 rows:

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

The 'Class: contact-lenses (Nom)' section shows three stacked bar charts representing the distribution of the 'age' attribute across the 'contact-lenses' classes. The bars are colored cyan, red, and blue from top to bottom.



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



Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Cluster

Choose SimpleKMeans -K2 -S10

Cluster mode

Use training set

Cluster output

\*\*\* Run information \*\*\*

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Associate

Choose Apriori -M10 -T0 -C0.9 -D0.05 -U1.0 -M0.1 -S-1.0

Start Stop

Associate output

Result list (right-click for)

12:09:06 - Apriori

```

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)
2. astigmatism=yes tear-prod-rate-reduced 6 ==> contact-lenses-none 6   conf:(1)
3. astigmatism=no tear-prod-rate-reduced 6 ==> contact-lenses-none 6   conf:(1)
4. spectacle-prescrip-hypermetrope tear-prod-rate-reduced 6 ==> contact-lenses-none 6   conf:(1)
5. spectacle-prescrip-myope tear-prod-rate-reduced 6 ==> contact-lenses-none 6   conf:(1)
6. contact-lenses-soft 5 ==> astigmatism=no tear-prod-rate-normal 5   conf:(1)
7. astigmatism=no contact-lenses-soft 5 ==> tear-prod-rate-normal 5   conf:(1)
8. tear-prod-rate-normal contact-lenses-soft 5 ==> astigmatism=no 5   conf:(1)

```

## PRACTICAL 4

### **AIM: Demonstration of classification rule process on dataset student.arff using j48 algorithm**

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 (student.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 69%.this indicates that we may find more work. (Either in preprocessing or in selecting current parameters for the classification)

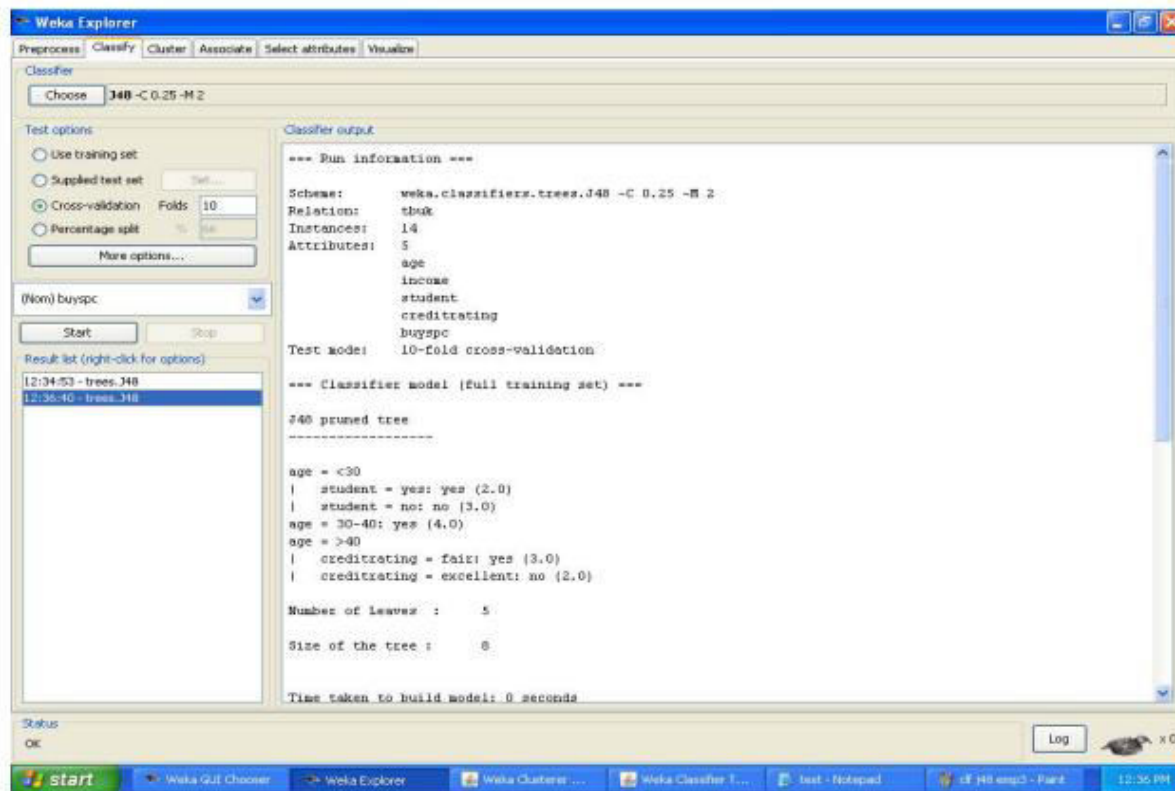
Step-7: Now weka also lets us a view a graphical version of the classification tree. This can be done by right clicking the last result set and selecting “visualize tree” from the pop-up menu.

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 will pop-up a window which will allow you to open the file containing test instances.

```
Dataset student .arff @relation student @attribute age {<30,30-40,>40} @attribute income {low,
medium, high} @attribute student {yes, no} @attribute credit-rating {fair, excellent} @attribute
buyspc {yes, no} @data % <30, high, no, fair, no <30, high, no, excellent, no 30-40, high, no,
fair, yes >40, medium, no, fair, yes >40, low, yes, fair, yes >40, low, yes, excellent, no 30-40,
low, yes, excellent, yes <30, medium, no, fair, no <30, low, yes, fair, no >40, medium, yes, fair,
yes <30, medium, yes, excellent, yes 30-40, medium, no, excellent, yes 30-40, high, yes, fair, yes
>40, medium, no, excellent, no %
```

The following screenshot shows the classification rules that were generated when j48 algorithm is applied on the given dataset.



Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose 348 -C 0.25 -M 2

Test options

Use training set

Supplied test set

Cross-validation Folds 10

Percentage split %

(Name) buyopc

Result list (right-click for options)

12:34:53 - trees\_348

12:36:40 - trees\_348

Classifier output

Size of the tree : 8

Time taken to build model: 0 seconds

--- Stratified cross-validation ---

--- Summary ---

Correctly Classified Instances	7	50	%
Incorrectly Classified Instances	7	50	%
Kappa statistic	-0.0426		
Mean absolute error	0.4167		
Root mean squared error	0.5984		
Relative absolute error	87.5 %		
Root relative squared error	121.2907 %		
Total Number of Instances	14		

--- Detailed Accuracy By Class ---

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.556	0.6	0.625	0.556	0.588	0.633	yes
	0.4	0.444	0.333	0.4	0.364	0.633	no
Weighted Avg.	0.5	0.544	0.521	0.5	0.508	0.633	


--- Confusion Matrix ---

```

a b <-- classified as
5 4 | a = yes
3 2 | b = no

```

Status

OK   x 0

Start Weka GUI Chooser Weka Explorer Weka Clusterer ... Weka Classifier T... test - Notepad C:\MSI\stull1 - Part 12:37 PM

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Classifier: Choose J48 - C 0.25 - M 2

Test options:
 

- Use training set
- Supplied test set
- Cross-validate
- Percentage split

(Non) buyspc: Start Stop

Result list:
 

- 12:34:53 - trees.J48
- 12:36:40 - trees.J48

Classifier output: Size of the tree : 8

Weka Classifier Tree Visualizer: 12:36:40 - trees.J48 (rbuk)

Tree View

```

    graph TD
      A((age)) -- "= < 30" --> B((student))
      A -- "= 30-40" --> C[yes (4.0)]
      A -- "= > 40" --> D((creditrating))
      B -- "= yes" --> E[yes (2.0)]
      B -- "= no" --> F[no (3.0)]
      D -- "= fair" --> G[yes (3.0)]
      D -- "= excellent" --> H[no (2.0)]
  
```

Class: yes no

Status: OK

Log

Windows taskbar: start, Weka GUI C..., Weka Explorer, Weka Cluste..., Weka Classifi..., Weka Classifi..., test - Notepad, if H8 stud2..., 12:37 PM

## PRACTICAL 5

**AIM: Demonstration of Association rule process on dataset supermarket.arff using apriori algorithm**

### Execution steps

**Step1:** 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.

**Step2:** Clicking on the associate tab will bring up the interface for association rule algorithm.

**Step3:** We will use apriori algorithm. This is the default algorithm.

**Step4:** 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.

### Dataset contactlenses.arff

The screenshot shows the Weka Explorer interface. On the left, a file explorer shows the 'contact-lenses.arff' file. The main window displays a table of the dataset with columns: 'age', 'spectacle-prescrip', 'astigmatism', 'tear-prod-rate', and 'contact-lenses'. The 'contact-lenses' column is selected. On the right, the 'Associate' interface is open, showing the 'Selected attribute' as 'age' with 3 distinct values: 'young', 'pre-presbyopic', and 'presbyopic'. Below this, there are three bar charts representing the distribution of the 'age' attribute across the 'contact-lenses' classes.

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

Dataset test.arff

@relation test

@attribute admissionyear {2005,2006,2007,2008,2009,2010}

@attribute course {cse,mech,it,ece}

@data

%

2005, cse

2005, it

2005, cse

2006, mech

2006, it

2006, ece

2007, it

2007, cse

2008, it

2008, cse

2009, it

2009, ece

%

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



Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Associator

Choose Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1

Start Stop

Result list (right-click for)

12:24:15 - Apriori

Associator output

```

=== 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
Relation:    test
Instances:   12
Attributes:  2
              admissionyear
              course

=== Associator model (full training set) ===

Apriori
*****

Minimum support: 0.1 (1 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 18

Generated sets of large itemsets:

Size of set of large itemsets L(1): 9

```

Status OK Log x 0

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Associator

Choose Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1

Start Stop

Result list (right-click for)

12:24:15 - Apriori

Associator output

```

              course
=== Associator model (full training set) ===

Apriori
*****

Minimum support: 0.1 (1 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 18

Generated sets of large itemsets:

Size of set of large itemsets L(1): 9

Size of set of large itemsets L(2): 11

Best rules found:

1. course=mech 1 ==> admissionyear=2006 1   conf: (1)

```

Status OK Log x 0

**Conclusion:**

The experiment displays Set of large itemsets, best rule found for the given support and the confidence values. We get the results faster using the toolkits.

## PRACTICAL 6

### AIM: Study about different tools: Weka, XLMiner, Hadoop

#### WEKA

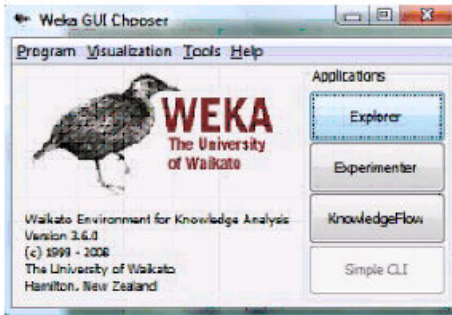
Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes.

Waikato Environment for Knowledge Analysis (Weka) is a suite of [machine learning](#) software written in [Java](#), developed at the [University of Waikato, New Zealand](#). It is [free software](#) licensed under the [GNU General Public License](#).

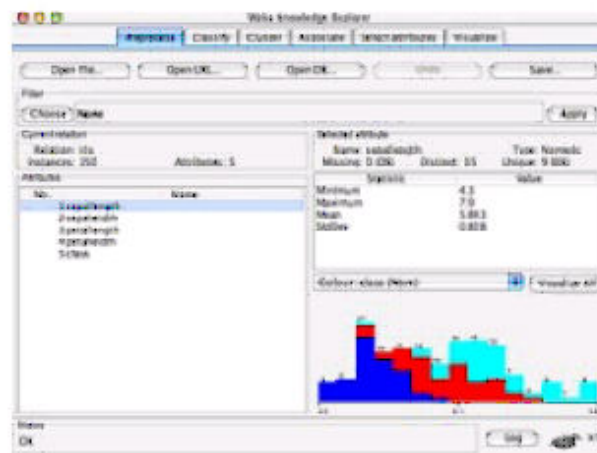
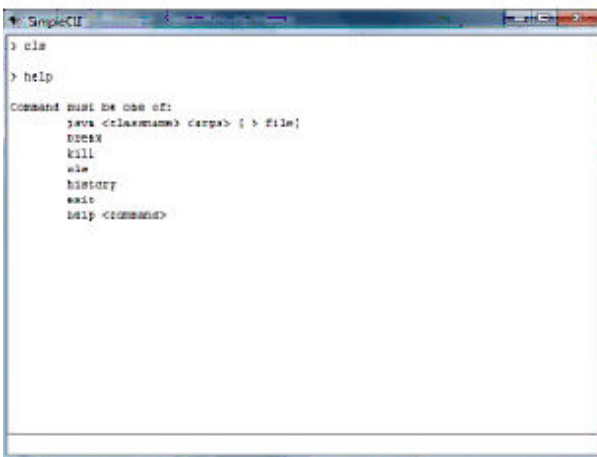
Weka contains a collection of visualization tools and algorithms for [data analysis](#) and [predictive modeling](#), together with graphical user interfaces for easy access to these functions.<sup>[1]</sup> The original non-Java version of Weka was a [Tcl/Tk](#) front-end to (mostly third-party) modeling algorithms implemented in other programming languages, plus [data preprocessing](#) utilities in [C](#), and a [Makefile](#)-based system for running machine learning experiments. This original version was primarily designed as a tool for analyzing data from agricultural domains,<sup>[2][3]</sup> but the more recent fully [Java](#)-based version (Weka 3), for which development started in 1997, is now used in many different application areas, in particular for educational purposes and research. Advantages of Weka include:

- Free availability under the [GNU General Public License](#).
- Portability, since it is fully implemented in the [Java programming language](#) and thus runs on almost any modern computing platform.
- A comprehensive collection of data preprocessing and modeling techniques.
- Ease of use due to its graphical user interfaces.

Weka supports several standard [data mining](#) tasks, more specifically, data preprocessing, [clustering](#), [classification](#), [regression](#), visualization, and [feature selection](#). All of Weka's techniques are predicated on the assumption that the data is available as one flat file or relation, where each data point is described by a fixed number of attributes (normally, numeric or nominal attributes, but some other attribute types are also supported). Weka provides access to [SQL databases](#) using [Java Database Connectivity](#) and can process the result returned by a database query. Weka provides access to [deep learning](#) with [Deeplearning4j](#).<sup>[4]</sup> It is not capable of multi-relational data mining, but there is separate software for converting a collection of linked database tables into a single table that is suitable for processing using Weka.<sup>[5]</sup> Another important area that is currently not covered by the algorithms included in the Weka distribution is sequence modeling.



## CLI Vs GUI



- Recommended for in-depth usage
- Offers some functionality not available via the GUI

- Explorer
- Experimenter
- Knowledge Flow

**WEKA only deals with “flat” files**  
 @relation heart-disease-simplified

```
@attribute age numeric
@attribute sex { female, male }
@attribute chest_pain_type { typ_angina, asympt, non_anginal, atyp_angina }
@attribute cholesterol numeric
@attribute exercise_induced_angina { no, yes }
@attribute class { present, not_present }
@data
63,male,typ_angina,233,no,not_present
67,male,asympt,286,yes,present
67,male,asympt,229,yes,present
38,female,non_anginal,?,no,not_present
...
```

**XLMINER:**

XLMiner™ is a comprehensive data mining add-in for Excel. Data mining is a discovery-driven data analysis technology used for identifying patterns and relationships in data sets. With overwhelming amounts of data now available from transaction systems and external data sources, organizations are presented with increasing opportunities to understand their data and gain insights into it. Data mining is still an emerging field, and is a convergence of fields like statistics, machine learning, and artificial intelligence.

XLMiner is a tool belt offering a variety of methods to analyze data. It has extensive coverage of statistical and machine-learning techniques for classification, prediction, affinity analysis, data exploration, and reduction.

On the XLMiner ribbon, click XLMiner Platform to view the XLMiner ribbon tabs.

The XLMiner ribbon is divided into five tabs: Data, Data Analysis, Time Series, Data Mining, and Applying Your Model.

Data - This tab includes the Get Data icon. Click Get Data to retrieve a sample from a worksheet or database. Select File Folder to import a collection of text documents located within a file folder. Select Big Data to use Apache Spark to bring big data into Excel.

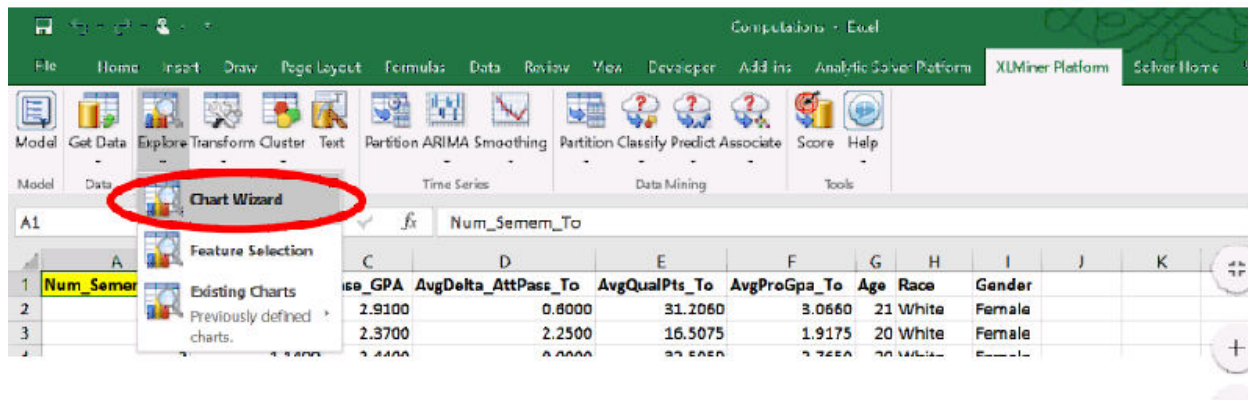
Data Analysis - This tab includes four icons: Explore, Transform, Cluster, and Text. Click Explore to use the Chart Wizard, Feature Selection, or view existing charts. Click Transform to transform data sets with missing data, perform binning, or to transform categorical data. Click Cluster to perform cluster analysis using k-Means or Hierarchical methods. Click Text to perform an analysis on a collection of text documents using the new Text Miner feature.

Time Series - This tab includes three icons: Partition, ARIMA, and Smoothing. These functions are used when analyzing a time series.

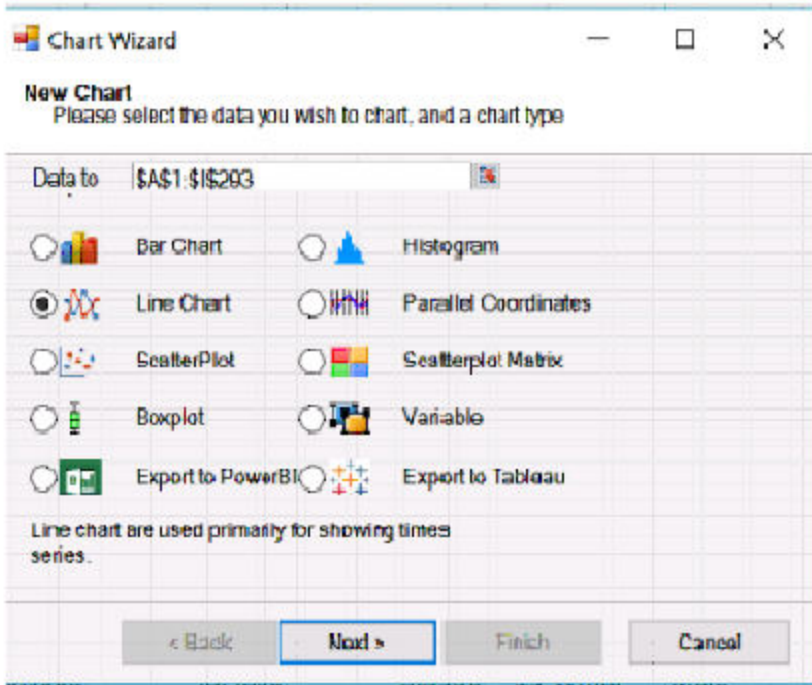
Data Mining - This tab includes four icons: Partition, Classify, Predict, and Associate. These functions are used to perform data mining activities.

Applying Your Model - This tab includes two icons: Score and Help. Click Score to score data in a database or worksheet using the classification or prediction algorithms. Click Help to change the product, check your license status, view data set examples, view the Help File, download the User Guide, check for updates, and view copyright information related to XLMiner.

1. Open your Computations file in Excel. Go to XL Miner tab. Then click Explore -> Chart Wizard



2. You will be prompted to choose your chart type. Choose Line Chart for this Dashboard. Click Next

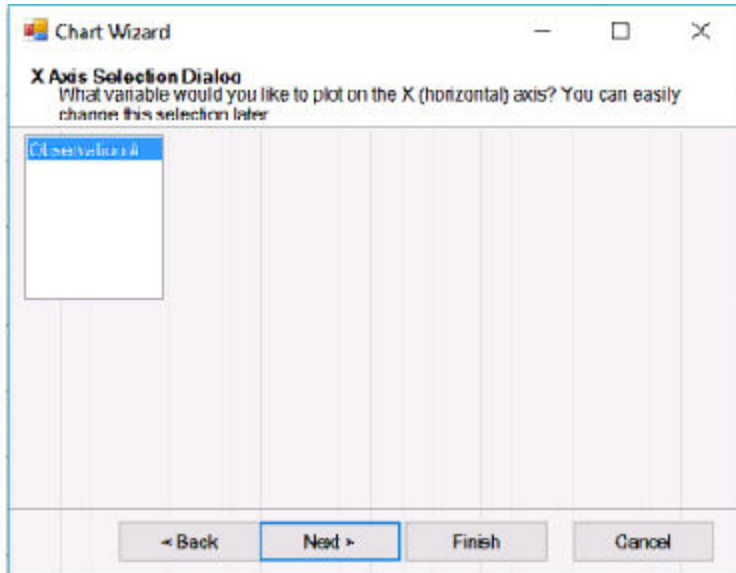


3. Now you need to select your variables. For this dashboard, choose the following variables and click Next.

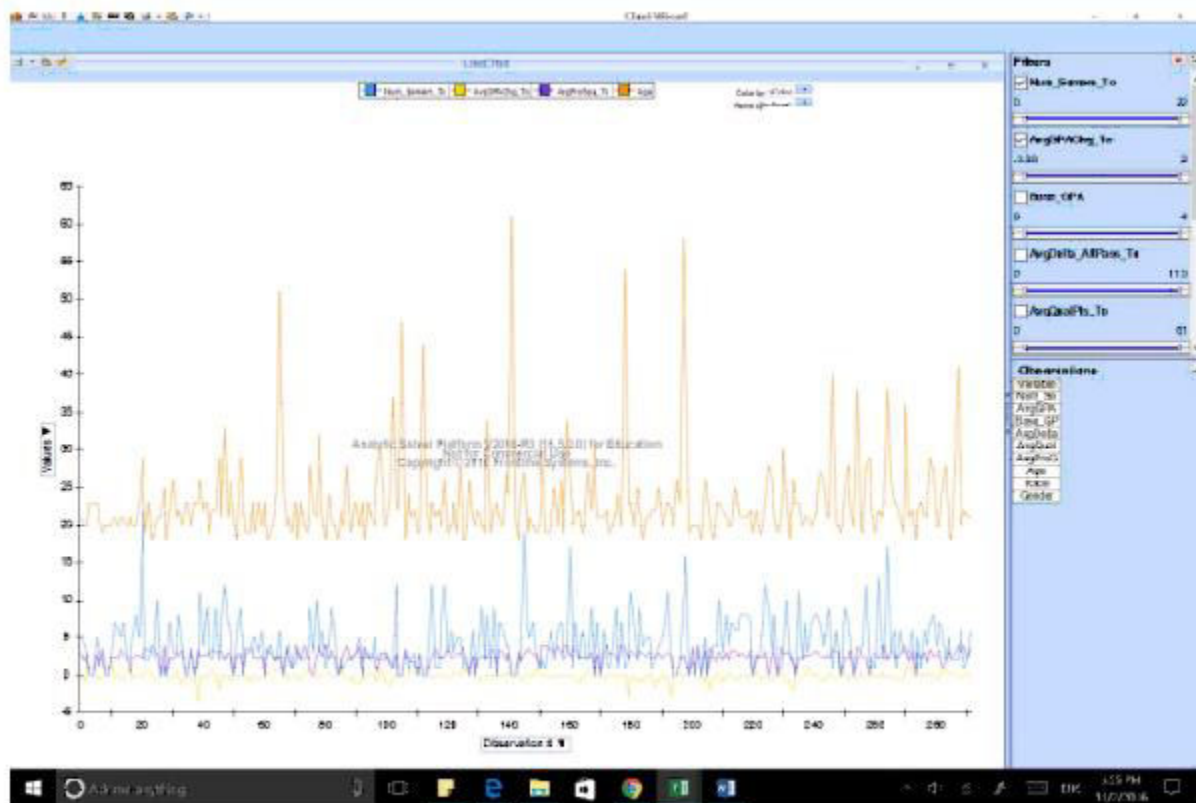


4. You will be prompted to choose X- axis. Choose Observations # and click Finish.





5. You will see the goal dashboard displayed



6. When you are done with this chart, you can close it by clicking X in the top right corner. You will be given an option to either save or discard the chart

The screenshot shows a 'Save Chart Window' dialog box. It contains a text input field labeled 'Name:' and four buttons: 'Delete', 'Save', 'Discard', and 'Cancel'. The dialog box also includes a close button (X) in the top right corner.

## HADOOP:

Hadoop is an open-source framework that allows to store and process big data in a distributed environment across clusters of computers using simple programming models. It is designed to

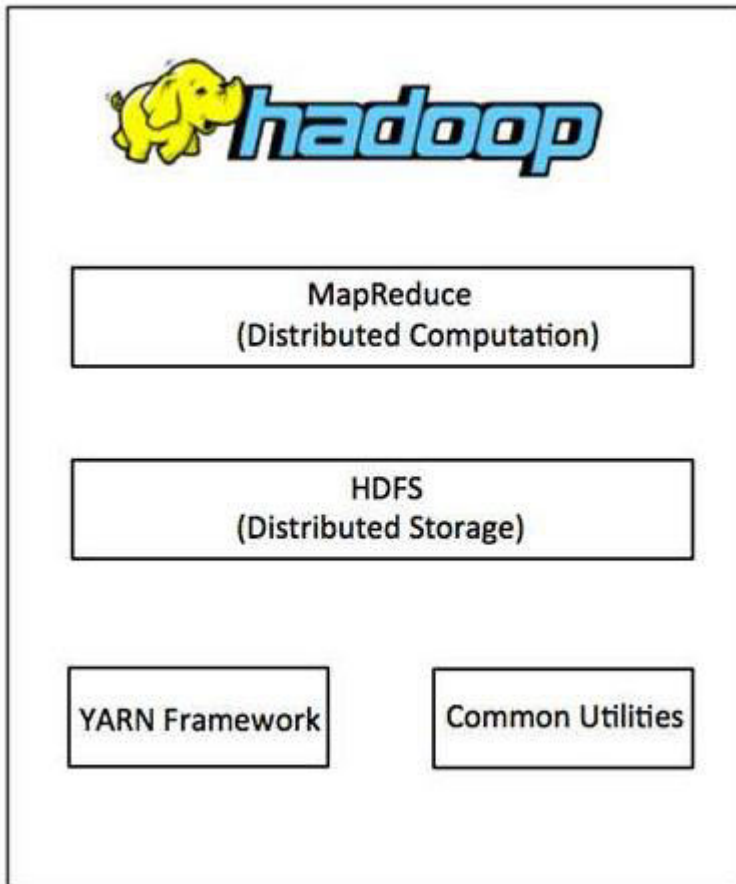
scale up from single servers to thousands of machines, each offering local computation and storage.

### **Hadoop Architecture**

Hadoop framework includes following four modules:

- **Hadoop Common:** These are Java libraries and utilities required by other Hadoop modules. These libraries provides filesystem and OS level abstractions and contains the necessary Java files and scripts required to start Hadoop.
- **Hadoop YARN:** This is a framework for job scheduling and cluster resource management.
- **Hadoop Distributed File System (HDFS™):** A distributed file system that provides high-throughput access to application data.
- **Hadoop MapReduce:** This is YARN-based system for parallel processing of large data sets.

We can use following diagram to depict these four components available in Hadoop framework.



## MapReduce

Hadoop MapReduce is a software framework for easily writing applications which process big amounts of data in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.

The term MapReduce actually refers to the following two different tasks that Hadoop programs perform:

- The Map Task: This is the first task, which takes input data and converts it into a set of data, where individual elements are broken down into tuples (key/value pairs).
- The Reduce Task: This task takes the output from a map task as input and combines those data tuples into a smaller set of tuples. The reduce task is always performed after the map task.

Typically both the input and the output are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks.

The MapReduce framework consists of a single master JobTracker and one slave TaskTracker per cluster-node. The master is responsible for resource management, tracking resource consumption/availability and scheduling the jobs component tasks on the slaves, monitoring them and re-executing the failed tasks. The slaves TaskTracker execute the tasks as directed by the master and provide task-status information to the master periodically.

The JobTracker is a single point of failure for the Hadoop MapReduce service which means if JobTracker goes down, all running jobs are halted.

### **Hadoop Distributed File System**

Hadoop can work directly with any mountable distributed file system such as Local FS, HFTP FS, S3 FS, and others, but the most common file system used by Hadoop is the Hadoop Distributed File System (HDFS).

The Hadoop Distributed File System (HDFS) is based on the Google File System (GFS) and provides a distributed file system that is designed to run on large clusters (thousands of computers) of small computer machines in a reliable, fault-tolerant manner.

HDFS uses a master/slave architecture where master consists of a single NameNode that manages the file system metadata and one or more slave DataNodes that store the actual data.

A file in an HDFS namespace is split into several blocks and those blocks are stored in a set of DataNodes. The NameNode determines the mapping of blocks to the DataNodes. The DataNodes takes care of read and write operation with the file system. They also take care of block creation, deletion and replication based on instruction given by NameNode.

HDFS provides a shell like any other file system and a list of commands are available to interact with the file system. These shell commands will be covered in a separate chapter along with appropriate examples.

### **How Does Hadoop Work?**

Stage 1A user/application can submit a job to the Hadoop (a hadoop job client) for required process by specifying the following items:

1. The location of the input and output files in the distributed file system.
2. The java classes in the form of jar file containing the implementation of map and reduce functions.
3. The job configuration by setting different parameters specific to the job.

## Stage 2

The Hadoop job client then submits the job (jar/executable etc) and configuration to the JobTracker which then assumes the responsibility of distributing the software/configuration to the slaves, scheduling tasks and monitoring them, providing status and diagnostic information to the job-client.

## Stage 3

The TaskTrackers on different nodes execute the task as per MapReduce implementation and output of the reduce function is stored into the output files on the file system.

### **Advantages of Hadoop**

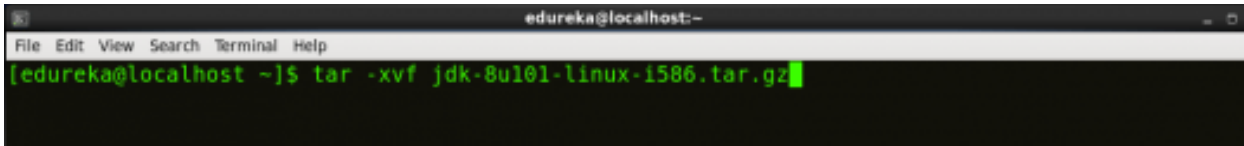
- Hadoop framework allows the user to quickly write and test distributed systems. It is efficient, and it automatically distributes the data and work across the machines and in turn, utilizes the underlying parallelism of the CPU cores.
- Hadoop does not rely on hardware to provide fault-tolerance and high availability (FTHA), rather Hadoop library itself has been designed to detect and handle failures at the application layer.
- Servers can be added or removed from the cluster dynamically and Hadoop continues to operate without interruption.
- Another big advantage of Hadoop is that apart from being open source, it is compatible on all the platforms since it is Java based.

## Hadoop Installation Step:

**Step 1:** [Click here](#) to download the Java 8 Package. Save this file in your home directory.

**Step 2:** Extract the Java Tar File.

**Command:** `tar -xvf jdk-8u101-linux-i586.tar.gz`

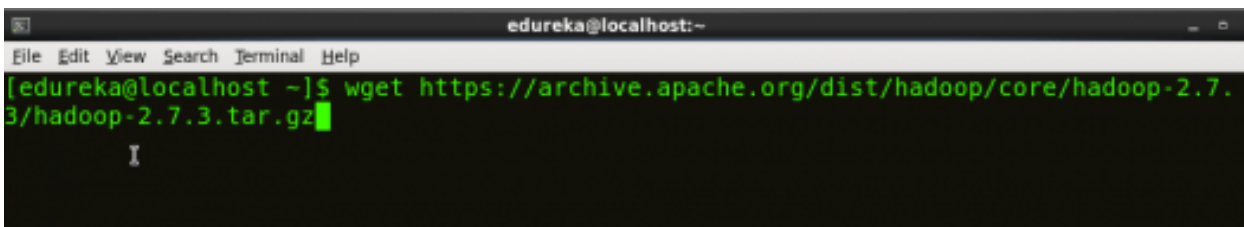


```
edureka@localhost:~$ tar -xvf jdk-8u101-linux-i586.tar.gz
```

*Fig: Hadoop Installation – Extracting Java Files*

**Step 3:** Download the Hadoop 2.7.3 Package.

**Command:** `wget https://archive.apache.org/dist/hadoop/core/hadoop-2.7.3/hadoop-2.7.3.tar.gz`

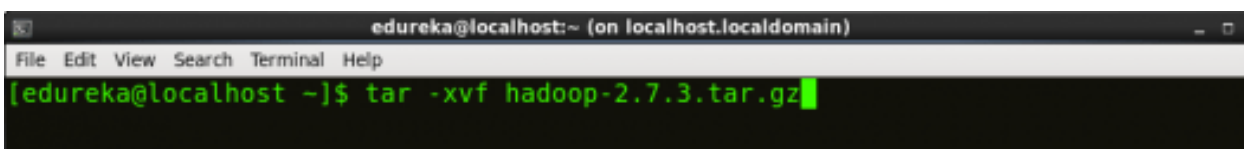


```
edureka@localhost:~$ wget https://archive.apache.org/dist/hadoop/core/hadoop-2.7.3/hadoop-2.7.3.tar.gz
```

*Fig: Hadoop Installation – Downloading Hadoop*

**Step 4:** Extract the Hadoop tar File.

**Command:** `tar -xvf hadoop-2.7.3.tar.gz`



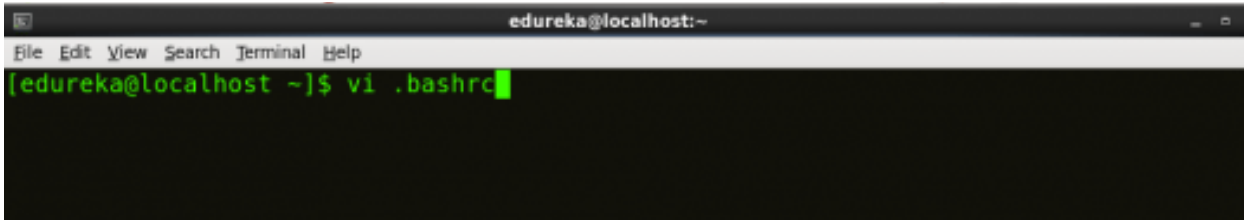
```
edureka@localhost:~ (on localhost.localdomain)$ tar -xvf hadoop-2.7.3.tar.gz
```

*Fig: Hadoop Installation – Extracting Hadoop Files*

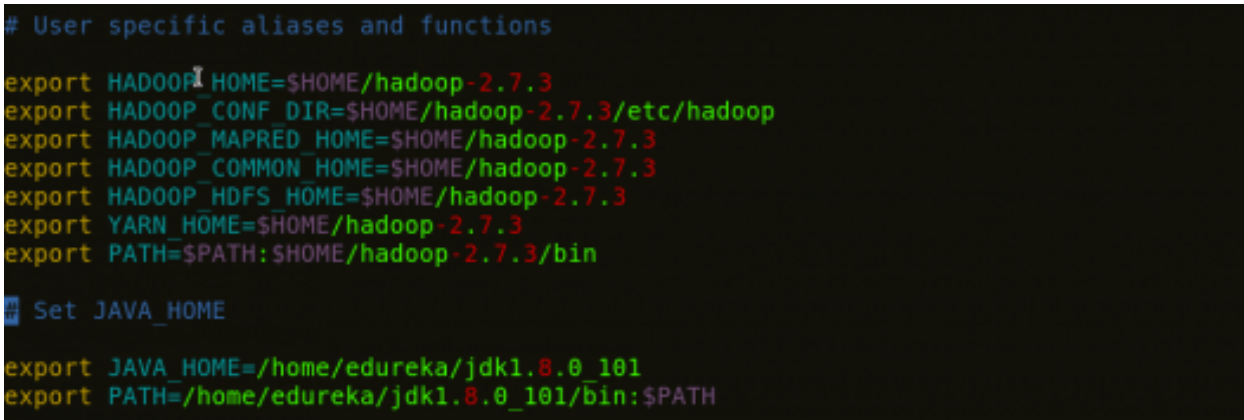
**Step 5:** Add the Hadoop and Java paths in the bash file (.bashrc).

Open. **bashrc** file. Now, add Hadoop and Java Path as shown below.

**Command:** `vi .bashrc`



```
edureka@localhost:~  
File Edit View Search Terminal Help  
[edureka@localhost ~]$ vi .bashrc
```



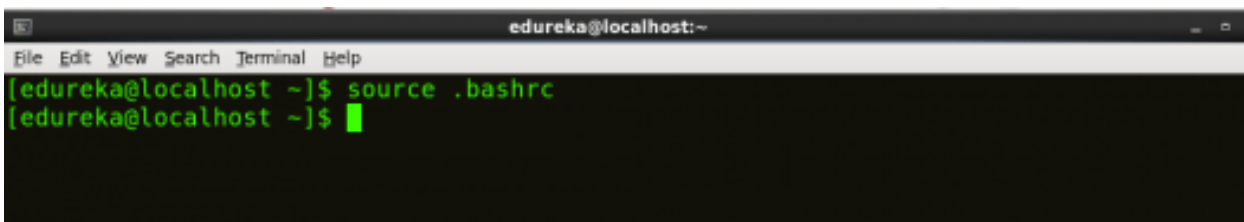
```
# User specific aliases and functions  
  
export HADOOP_HOME=$HOME/hadoop-2.7.3  
export HADOOP_CONF_DIR=$HOME/hadoop-2.7.3/etc/hadoop  
export HADOOP_MAPRED_HOME=$HOME/hadoop-2.7.3  
export HADOOP_COMMON_HOME=$HOME/hadoop-2.7.3  
export HADOOP_HDFS_HOME=$HOME/hadoop-2.7.3  
export YARN_HOME=$HOME/hadoop-2.7.3  
export PATH=$PATH:$HOME/hadoop-2.7.3/bin  
  
Set JAVA_HOME  
  
export JAVA_HOME=/home/edureka/jdk1.8.0_101  
export PATH=/home/edureka/jdk1.8.0_101/bin:$PATH
```

*Fig: Hadoop Installation – Setting Environment Variable*

Then, save the bash file and close it.

For applying all these changes to the current Terminal, execute the source command.

**Command:** source .bashrc



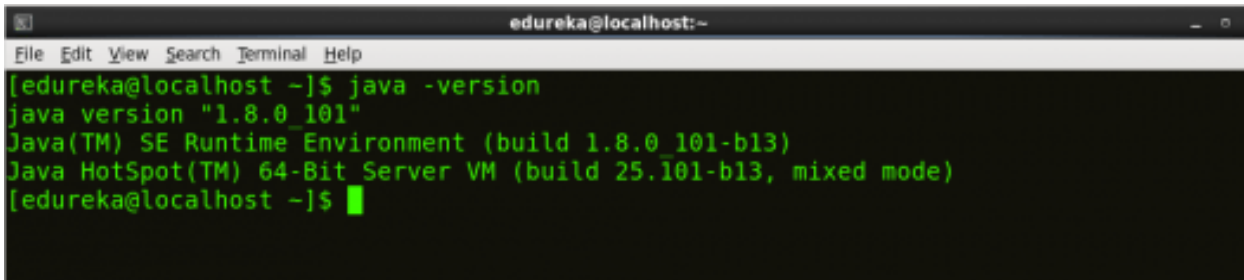
```
edureka@localhost:~  
File Edit View Search Terminal Help  
[edureka@localhost ~]$ source .bashrc  
[edureka@localhost ~]$
```

*Fig: Hadoop Installation – Refreshing environment variables*

To make sure that Java and Hadoop have been properly installed on your system and can be accessed through the Terminal, execute the java -version and hadoop version commands.



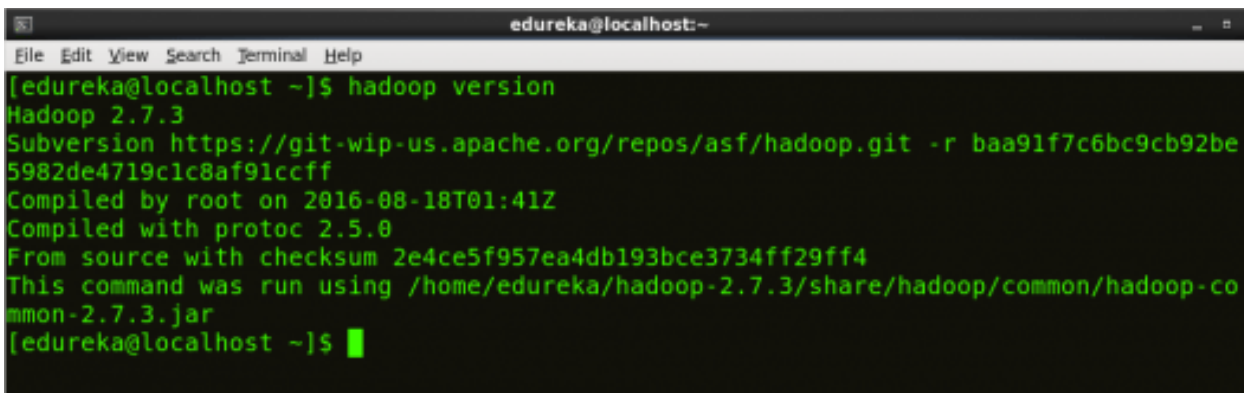
**Command:** java -version



```
edureka@localhost:~  
File Edit View Search Terminal Help  
[edureka@localhost ~]$ java -version  
java version "1.8.0_101"  
Java(TM) SE Runtime Environment (build 1.8.0_101-b13)  
Java HotSpot(TM) 64-Bit Server VM (build 25.101-b13, mixed mode)  
[edureka@localhost ~]$
```

*Fig: Hadoop Installation – Checking Java Version*

**Command:** hadoop version



```
edureka@localhost:~  
File Edit View Search Terminal Help  
[edureka@localhost ~]$ hadoop version  
Hadoop 2.7.3  
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r baa91f7c6bc9cb92be5982de4719c1c8af91ccff  
Compiled by root on 2016-08-18T01:41Z  
Compiled with protoc 2.5.0  
From source with checksum 2e4ce5f957ea4db193bce3734ff29ff4  
This command was run using /home/edureka/hadoop-2.7.3/share/hadoop/common/hadoop-common-2.7.3.jar  
[edureka@localhost ~]$
```

*Fig: Hadoop Installation – Checking Hadoop Version*

**Step 6:** Edit the [Hadoop Configuration files](#).

**Command:** cd hadoop-2.7.3/etc/hadoop/

**Command:** ls

All the Hadoop configuration files are located in **hadoop-2.7.3/etc/hadoop** directory as you can see in the snapshot below:

```
edureka@localhost:~/hadoop-2.7.3/etc/hadoop
File Edit View Search Terminal Help
[edureka@localhost ~]$ cd hadoop-2.7.3/etc/hadoop/
[edureka@localhost hadoop]$ ls
capacity-scheduler.xml      httpfs-env.sh              mapred-env.sh
configuration.xml          httpfs-log4j.properties   mapred-queues.xml.template
container-executor.cfg     httpfs-signature.secret   mapred-site.xml.template
core-site.xml              httpfs-site.xml           slaves
hadoop-env.cmd            kms-acls.xml              ssl-client.xml.example
hadoop-env.sh             kms-env.sh                ssl-server.xml.example
hadoop-metrics2.properties kms-log4j.properties     yarn-env.cmd
hadoop-metrics.properties kms-site.xml              yarn-env.sh
hadoop-policy.xml         log4j.properties         yarn-site.xml
hdfs-site.xml             mapred-env.cmd
```

Fig: Hadoop Installation – Hadoop Configuration Files

**Step 7:** Open *core-site.xml* and edit the property mentioned below inside configuration tag: *core-site.xml* informs Hadoop daemon where NameNode runs in the cluster. It contains configuration settings of Hadoop core such as I/O settings that are common to HDFS & MapReduce.

**Command:** vi core-site.xml

```
edureka@localhost:~/hadoop-2.7.3/etc/hadoop
File Edit View Search Terminal Help
[edureka@localhost hadoop]$ vi core-site.xml
```

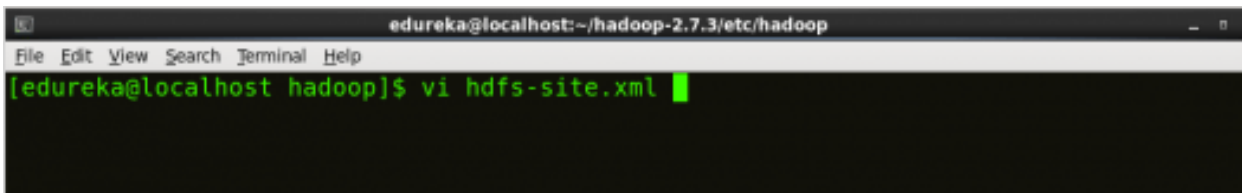
```
<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:9000</value>
</property>
</configuration>
```

Fig: Hadoop Installation – Configuring core-site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:9000</value>
</property>
</configuration>
```

**Step 8:** Edit *hdfs-site.xml* and edit the property mentioned below inside configuration tag:  
*hdfs-site.xml* contains configuration settings of HDFS daemons (i.e. NameNode, DataNode, Secondary NameNode). It also includes the replication factor and block size of HDFS.

**Command:** vi hdfs-site.xml



```
edureka@localhost:~/hadoop-2.7.3/etc/hadoop
File Edit View Search Terminal Help
[edureka@localhost hadoop]$ vi hdfs-site.xml
```



```
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.permission</name>
<value>>false</value>
</property>
```

*Fig: Hadoop Installation – Configuring hdfs-site.xml*

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
```

```
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.permission</name>
<value>>false</value>
</property>
</configuration>
```

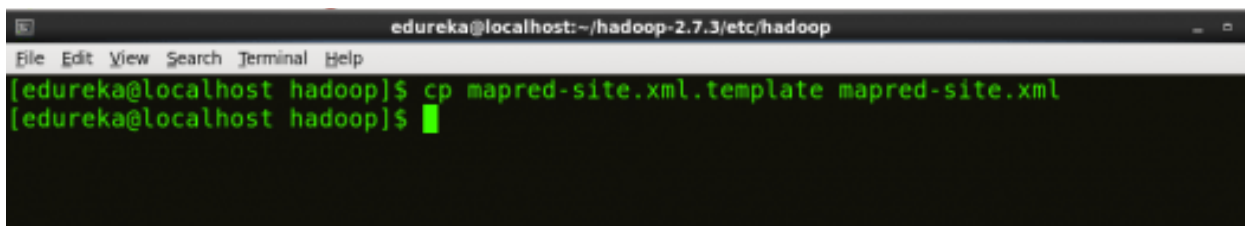
**Step 9:** Edit the *mapred-site.xml* file and edit the property mentioned below inside configuration tag:

*mapred-site.xml* contains configuration settings of MapReduce application like number of JVM that can run in parallel, the size of the mapper and the reducer process, CPU cores available for a process, etc.

In some cases, *mapred-site.xml* file is not available. So, we have to create the *mapred-site.xml* file using *mapred-site.xml* template.

**Command:** `cp mapred-site.xml.template mapred-site.xml`

**Command:** `vi mapred-site.xml`.



```
edureka@localhost:~/hadoop-2.7.3/etc/hadoop
File Edit View Search Terminal Help
[edureka@localhost hadoop]$ cp mapred-site.xml.template mapred-site.xml
[edureka@localhost hadoop]$ █
```

```
edureka@localhost:~/hadoop-2.7.3/etc/hadoop
File Edit View Search Terminal Help
[edureka@localhost hadoop]$ vi mapred-site.xml
```

```
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>
```

*Fig: Hadoop Installation – Configuring mapred-site.xml*

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>
```

**Step 10:** Edit *yarn-site.xml* and edit the property mentioned below inside configuration tag:  
*yarn-site.xml* contains configuration settings of ResourceManager and NodeManager like application memory management size, the operation needed on program & algorithm, etc.

**Command:** vi yarn-site.xml

```
edureka@localhost:~/hadoop-2.7.3/etc/hadoop
File Edit View Search Terminal Help
[edureka@localhost hadoop]$ vi yarn-site.xml
```

```
<configuration>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
</configuration>
```

*Fig: Hadoop Installation – Configuring yarn-site.xml*

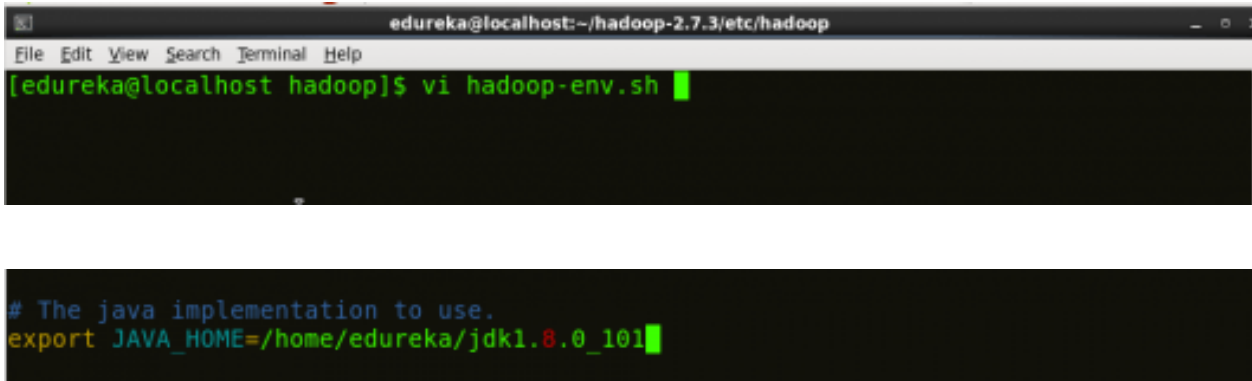
```
<?xml version="1.0">
<configuration>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
</configuration>
```

**Step 11:** Edit *hadoop-env.sh* and add the Java Path as mentioned below:

*hadoop-env.sh* contains the environment variables that are used in the script to run Hadoop like

Java home path, etc.

**Command:** vi hadoop-env.sh



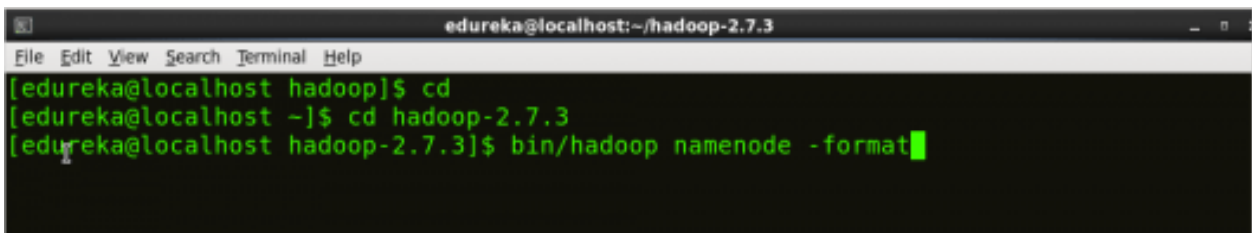
*Fig: Hadoop Installation – Configuring hadoop-env.sh*

**Step 12:** Go to Hadoop home directory and format the NameNode.

**Command:** cd

**Command:** cd hadoop-2.7.3

**Command:** bin/hadoop namenode -format



*Fig: Hadoop Installation – Formatting NameNode*

This formats the HDFS via NameNode. This command is only executed for the first time. Formatting the file system means initializing the directory specified by the dfs.name.dir variable. Never format, up and running Hadoop filesystem. You will lose all your data stored in the HDFS.

**Step 13:** Once the NameNode is formatted, go to `hadoop-2.7.3/sbin` directory and start all the daemons.

**Command:** `cd hadoop-2.7.3/sbin`

Either you can start all daemons with a single command or do it individually.

**Command:** `./start-all.sh`

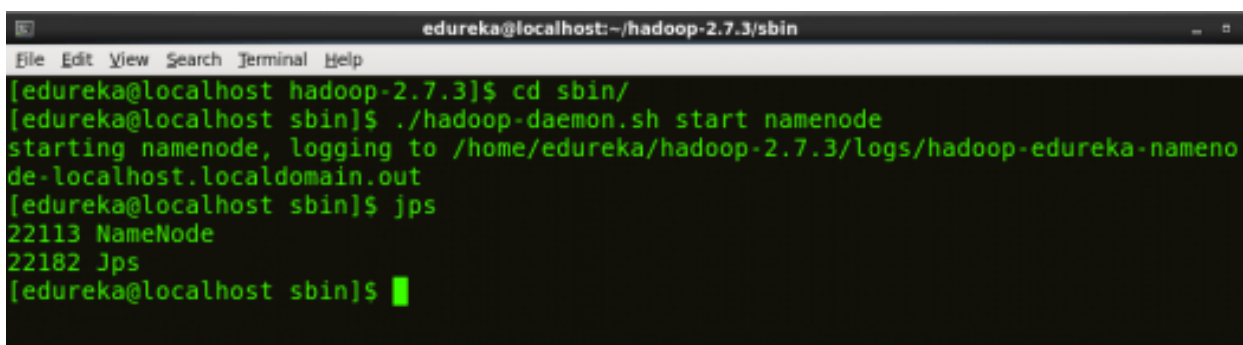
The above command is a combination of *start-dfs.sh*, *start-yarn.sh* & *mr-jobhistory-daemon.sh*

Or you can run all the services individually as below:

### Start NameNode:

The NameNode is the centerpiece of an HDFS file system. It keeps the directory tree of all files stored in the HDFS and tracks all the file stored across the cluster.

**Command:** `./hadoop-daemon.sh start namenode`



```
edureka@localhost:~/hadoop-2.7.3/sbin
File Edit View Search Terminal Help
[edureka@localhost hadoop-2.7.3]$ cd sbin/
[edureka@localhost sbin]$ ./hadoop-daemon.sh start namenode
starting namenode, logging to /home/edureka/hadoop-2.7.3/logs/hadoop-edureka-nameno
de-localhost.localdomain.out
[edureka@localhost sbin]$ jps
22113 NameNode
22182 Jps
[edureka@localhost sbin]$
```

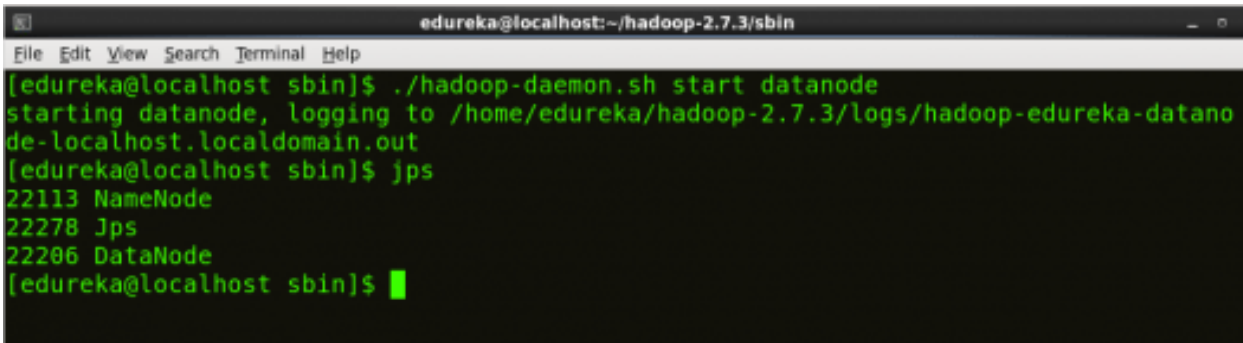
*Fig: Hadoop Installation – Starting NameNode*

### Start DataNode:

On startup, a DataNode connects to the Namenode and it responds to the requests from the Namenode for different operations.



**Command:** `./hadoop-daemon.sh start datanode`



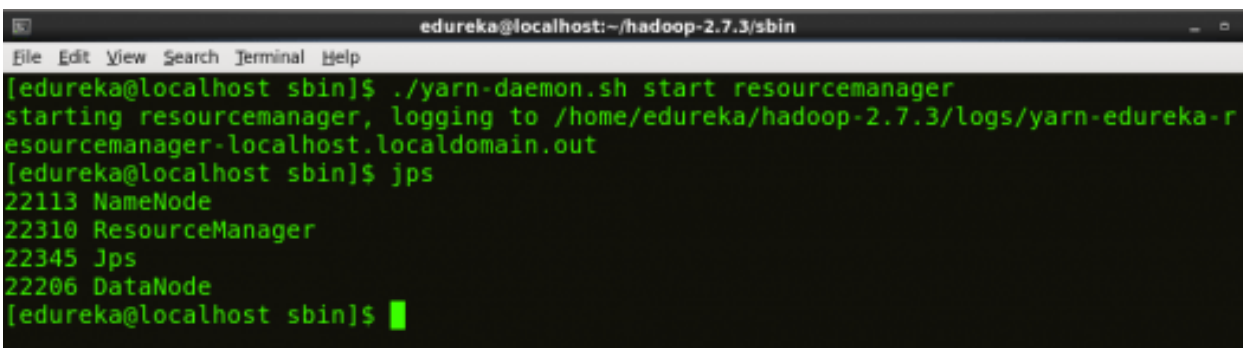
```
edureka@localhost:~/hadoop-2.7.3/sbin
File Edit View Search Terminal Help
[edureka@localhost sbin]$ ./hadoop-daemon.sh start datanode
starting datanode, logging to /home/edureka/hadoop-2.7.3/logs/hadoop-edureka-datano
de-localhost.localdomain.out
[edureka@localhost sbin]$ jps
22113 NameNode
22278 Jps
22206 DataNode
[edureka@localhost sbin]$ █
```

*Fig: Hadoop Installation – Starting DataNode*

### **Start ResourceManager:**

ResourceManager is the master that arbitrates all the available cluster resources and thus helps in managing the distributed applications running on the YARN system. Its work is to manage each NodeManagers and the each application's ApplicationMaster.

**Command:** `./yarn-daemon.sh start resourcemanager`



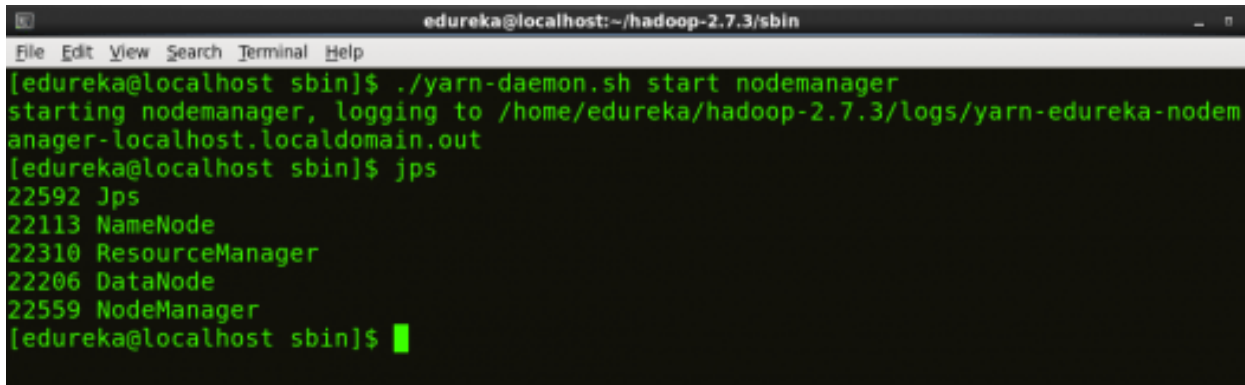
```
edureka@localhost:~/hadoop-2.7.3/sbin
File Edit View Search Terminal Help
[edureka@localhost sbin]$ ./yarn-daemon.sh start resourcemanager
starting resourcemanager, logging to /home/edureka/hadoop-2.7.3/logs/yarn-edureka-r
esourcemanager-localhost.localdomain.out
[edureka@localhost sbin]$ jps
22113 NameNode
22310 ResourceManager
22345 Jps
22206 DataNode
[edureka@localhost sbin]$ █
```

*Fig: Hadoop Installation – Starting ResourceManager*

### **Start NodeManager:**

The NodeManager in each machine framework is the agent which is responsible for managing containers, monitoring their resource usage and reporting the same to the ResourceManager.

**Command:** `./yarn-daemon.sh start nodemanager`

A terminal window titled 'edureka@localhost:~/hadoop-2.7.3/sbin' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the following commands and output:

```
[edureka@localhost sbin]$ ./yarn-daemon.sh start nodemanager
starting nodemanager, logging to /home/edureka/hadoop-2.7.3/logs/yarn-edureka-nodemanager-localhost.localdomain.out
[edureka@localhost sbin]$ jps
22592 Jps
22113 NameNode
22310 ResourceManager
22206 DataNode
22559 NodeManager
[edureka@localhost sbin]$
```

*Fig: Hadoop Installation – Starting NodeManager*

### **Start JobHistoryServer:**

JobHistoryServer is responsible for servicing all job history related requests from client.

**Command:** `./mr-jobhistory-daemon.sh start historyserver`

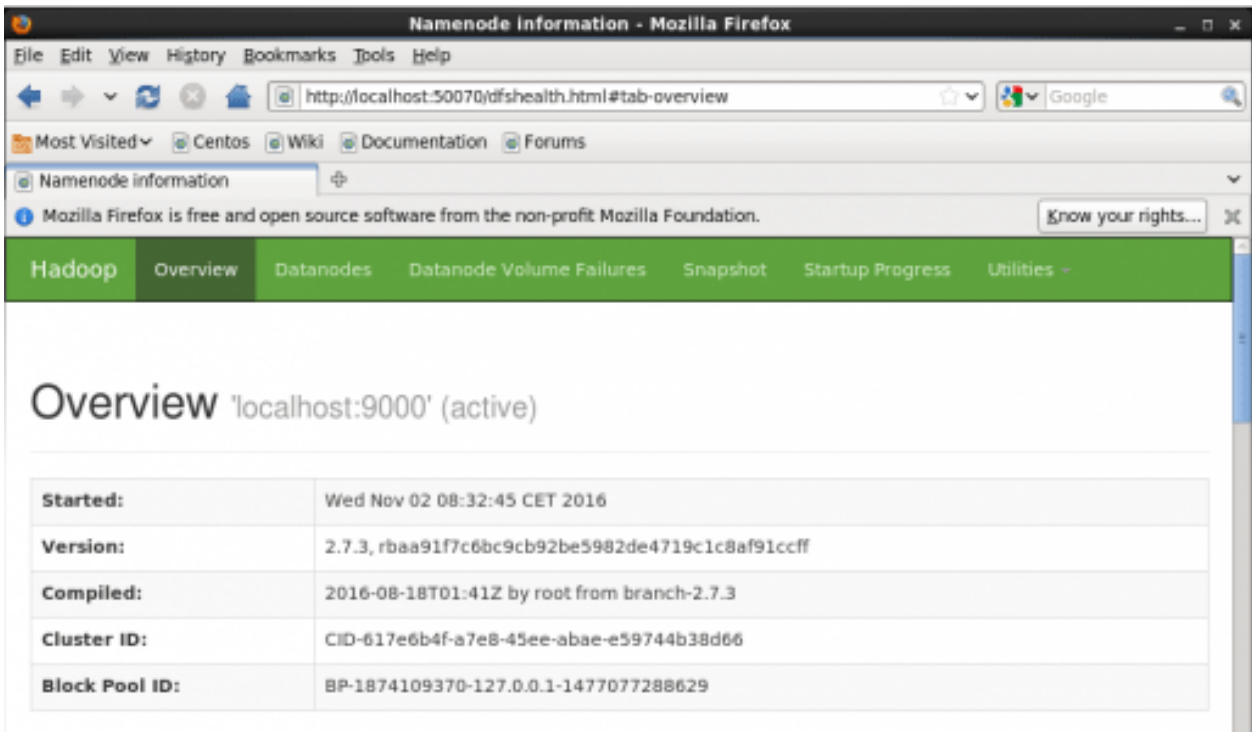
**Step 14:** To check that all the Hadoop services are up and running, run the below command.

**Command:** `jps`

```
edureka@localhost:~/hadoop-2.7.3/sbin
File Edit View Search Terminal Help
[edureka@localhost sbin]$ ./mr-jobhistory-daemon.sh start historyserver
starting historyserver, logging to /home/edureka/hadoop-2.7.3/logs/mapred-edureka-h
istoryserver-localhost.localdomain.out
[edureka@localhost sbin]$ jps
22113 NameNode
22310 ResourceManager
22694 JobHistoryServer
22727 Jps
22206 DataNode
22559 NodeManager
[edureka@localhost sbin]$ █
```

*Fig: Hadoop Installation – Checking Daemons*

**Step 15:** Now open the Mozilla browser and go to **localhost:50070/dfshealth.html** to check the NameNode interface.



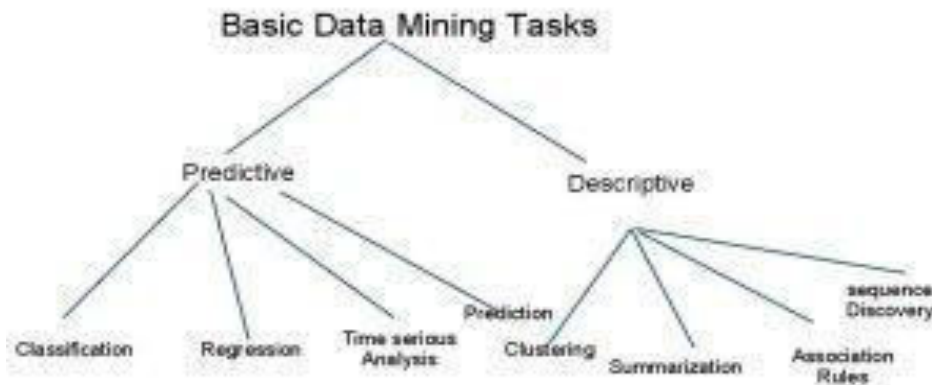
*Fig: Hadoop Installation – Starting WebUI*

## PRACTICAL 7

### AIM: Introduction to the classification of Mining techniques

Data mining techniques can be classified broadly as

- 1. Predictive:
  - a. Classification
  - b. Regression
  - c. Time series Analysis
  - d. Prediction
- 2. Descriptive:
  - a. clustering
  - b. Summarization
  - c. Association Rules
  - d. Sequence Discovery
- You can refer the below chart about that.



- 
- Data Mining Techniques Graph
- Now I am going to give brief intro about each types.
- **Classification:**
- It is often referred as “supervised learning”. It has a predefined set of groups or models based on that we predict values.
- (e.g) Airport security maintains a set of metrics and try to predict the terrorist
- **Regression:**
- The regression using known data formats like linear or logistic and assume the future data format will fall in to the data structure. It then try to predict the value by applying some mathematical algorithms on the data set.
- (e.g) Investing on Pension fund. Calculating your annual income and try to predict what you need after you retire. Then based on the present income and needed income makes investment decision. The Prediction done by simple regression formula to revise every year.
- **Time series Analysis:**
- With time series analysis, every attribute value determine by the different time interval.

- (e.g) Buying a company stock. Take X,Y,Z companies month by month performance and try to predict their next one year growth and based on the growth you buy stocks.
- **Prediction:**
- Prediction is relates with time series but not time bound. It is used to predict value based on past data and current data.
- (e.g) Water flow of a river will be calculated by various monitors at different levels and different time intervals. It then using those information to predict the water flow of future.
- **Clustering:**
- It is widely called as unsupervised learning. It is similar to classification except it won't have any predefined groups. instead the data itself define the group.
- (e.g) Consider a super market has buying details like age, job and purchase amount we can group by age against percentage as well job against percentage to make meaningful business decision to target the specific user group.
- **Summarization:**
- Summarization is associating the sample subset with small description or snippet.
- **Association Rules:**
- It is also called as linked analysis. It is all about under covering relationship among data.
- (e.g) Amazon "People bought this also bought this" model
- **Sequence Discovery:**
- Sequence discovery is about finding sequence of an activity.
- (e.g) In a shop people may often buy tothpaste after toothbrush. It is all about what sequence user buying the product and based on theo shop owner can arrange the items near by each others.

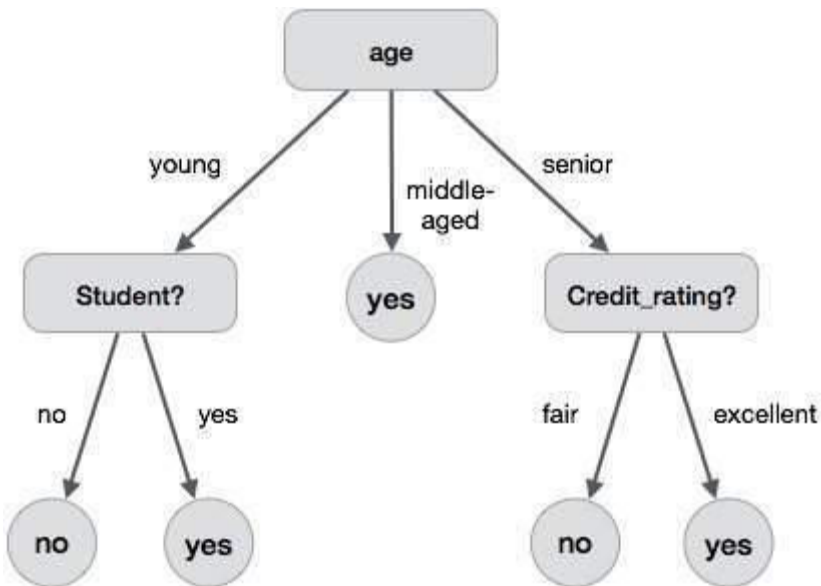
## PRACTICAL 8

### AIM: Introduction to Decision Tree and Neural Networks

#### Decision Tree

A decision tree is a structure that includes a root node, branches, and leaf nodes. Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label. The topmost node in the tree is the root node.

The following decision tree is for the concept buy\_computer that indicates whether a customer at a company is likely to buy a computer or not. Each internal node represents a test on an attribute. Each leaf node represents a class.



The benefits of having a decision tree are as follows –

- It does not require any domain knowledge.
- It is easy to comprehend.
- The learning and classification steps of a decision tree are simple and fast.

#### Decision Tree Induction Algorithm

A machine researcher named J. Ross Quinlan in 1980 developed a decision tree algorithm known as ID3 (Iterative Dichotomiser). Later, he presented C4.5, which was the successor of ID3. ID3 and C4.5 adopt a greedy approach. In this algorithm, there is no backtracking; the trees are constructed in a top-down recursive divide-and-conquer manner.

Generating a decision tree from training tuples of data partition D  
Algorithm : Generate\_decision\_tree

Input:

Data partition, D, which is a set of training tuples and their associated class labels.

attribute\_list, the set of candidate attributes.

Attribute selection method, a procedure to determine the splitting criterion that best partitions the data tuples into individual classes. This criterion includes a splitting\_attribute and either a splitting point or splitting subset.

Output:

A Decision Tree

Method

create a node N;

if tuples in D are all of the same class, C then

return N as leaf node labeled with class C;

if attribute\_list is empty then

return N as leaf node with labeled with majority class in D;|| majority voting

apply attribute\_selection\_method(D, attribute\_list)

to find the best splitting\_criterion;

label node N with splitting\_criterion;

if splitting\_attribute is discrete-valued and

multiway splits allowed then // no restricted to binary trees

attribute\_list = splitting\_attribute; // remove splitting attribute

for each outcome j of splitting\_criterion

// partition the tuples and grow subtrees for each partition

let D<sub>j</sub> be the set of data tuples in D satisfying outcome j; // a partition

if D<sub>j</sub> is empty then

attach a leaf labeled with the majority class in D to node N;

```
else
  attach the node returned by Generate
  decision tree(Dj, attribute list) to node N;
end for
return N;
```

### Tree Pruning

Tree pruning is performed in order to remove anomalies in the training data due to noise or outliers. The pruned trees are smaller and less complex.

#### Tree Pruning Approaches

There are two approaches to prune a tree –

- Pre-pruning – The tree is pruned by halting its construction early.
- Post-pruning - This approach removes a sub-tree from a fully grown tree.

### **Cost Complexity**

The cost complexity is measured by the following two parameters –

- Number of leaves in the tree, and
- Error rate of the tree.

## **Neural Network**

Neural network method is used for classification, clustering, feature mining, prediction and pattern recognition. It imitates the neurons structure of animals, bases on the M-P model and Hebb learning rule, so in essence it is a distributed matrix structure. Through training data mining, the neural network method gradually calculates (including repeated iteration or cumulative calculation) the weights the neural network connected. The neural network model can be broadly divided into the following three types: (a) Feed-forward networks: It regards the perception back-propagation model and the function network as representatives, and mainly used in the areas such as prediction and pattern recognition; (b) Feedback network: It regards Hopfield discrete model and continuous model as representatives, and mainly used for associative memory and optimization calculation; (c) Self-organization networks: it regards adaptive resonance theory (ART) model and Kohonen model as representatives, and mainly used for cluster analysis.



**NEURAL NETWORKS IN DATA MINING** In more practical terms neural networks are non-linear statistical data modeling tools. They can be used to model complex relationships between inputs and outputs or to find patterns in data. Using neural networks as a tool, data warehousing firms are harvesting information from datasets in the process known as data mining. The difference between these data warehouses and ordinary databases is that there is actual manipulation and cross-fertilization of the data helping users makes more informed decisions. Neural networks essentially comprise three pieces: the architecture or model; the learning algorithm; and the activation functions. Neural networks are programmed or “trained” to “. . . store, recognize, and associatively retrieve patterns or database entries; to solve combinatorial optimization problems; to filter noise from measurement data; to control ill-defined problems; in summary, to estimate sampled functions when we do not know the form of the functions.” It is precisely these two abilities (pattern recognition and function estimation) which make artificial neural networks (ANN) so prevalent a utility in data mining. As data sets grow to massive sizes, the need for automated processing becomes clear. With their “model-free” estimators and their dual nature, neural networks serve data mining in a myriad of ways.

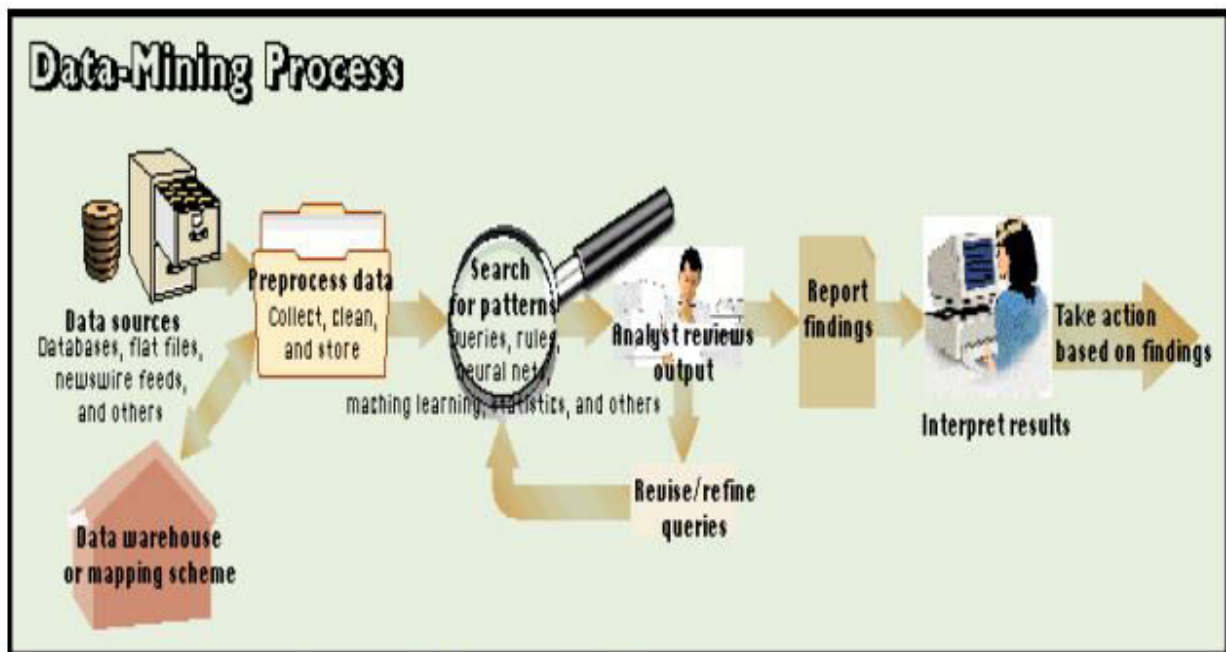


Figure 3. Image of data-mining process.

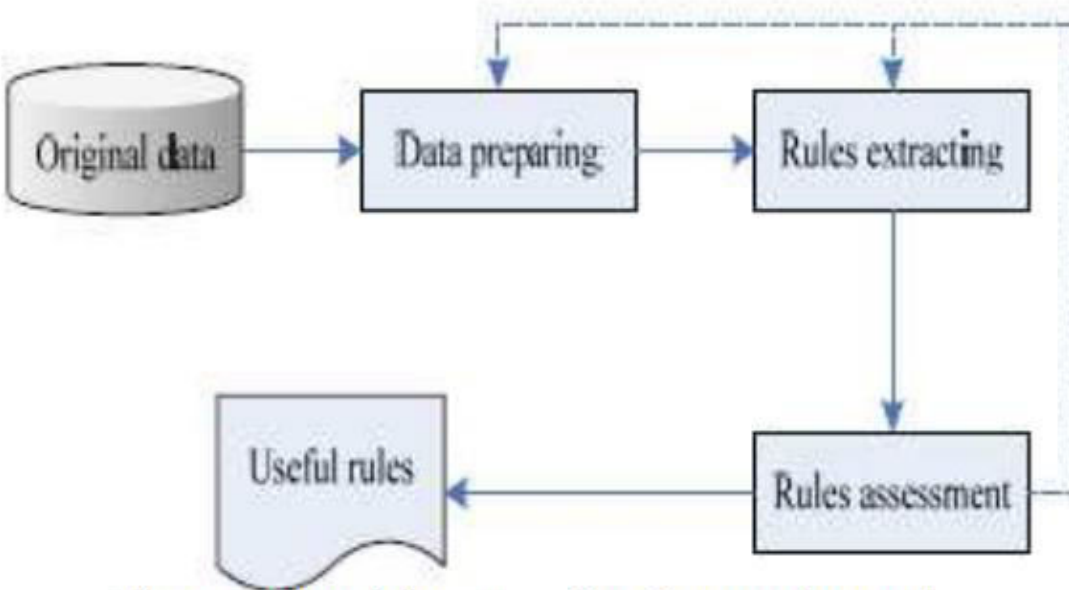


Figure 5. Data mining process based on neural network.

A. Data Preparation Data preparation is to define and process the mining data to make it fit specific data mining method. Data preparation is the first important step in the data mining and plays a decisive role in the entire data mining process. It mainly includes the following four processes: a) Data cleaning: Data cleansing is to fill the vacancy value of the data, eliminate the noise data and correct the inconsistencies data in the data. b) Data option: Data option is to select the data arrange and row used in this mining. c) Data preprocessing: Data preprocessing is to enhanced process the clean data which has been selected. d) Data expression Data expression is to transform the data after preprocessing into the form which can be accepted by the data mining algorithm based on neural network. The data mining based on neural network can only handle numerical data, so it is need to transform the sign data into numerical data. The simplest method is to establish a table with one-to-one correspondence between the sign data and the numerical data. The other more complex approach is to adopt appropriate Hash function to generate a unique numerical data according to given string. Although there are many data types in relational database, but they all basically can be simply come down to sign data, discrete numerical data and serial numerical data three logical data types. Fig. 6 gives the conversion of the three data types. The symbol “Apple” in the figure can be transformed into the corresponding discrete numerical data by using symbol table or Hash function. Then, the discrete numerical data can be quantified into continuous numerical data and can also be encoded into coding data.

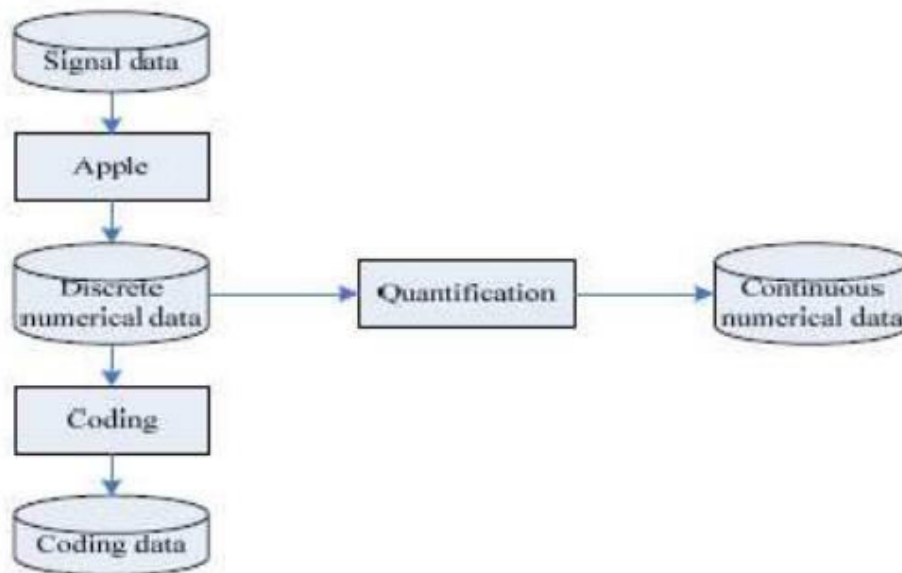


Fig. 6 Data expression and conversion in data mining based on neural network

B. Rules Extracting There are many methods to extract rules, in which the most commonly used methods are LRE method, black-box method, the method of extracting fuzzy rules, the method of extracting rules from recursive network, the algorithm of binary input and output rules extracting (BIO-RE), partial rules extracting algorithm (Partial-RE) and full rules extracting algorithm (Full-RE). C. Rules Assessment Although the objective of rules assessment depends on each specific application, but, in general terms, the rules can be assessed in accordance with the following objectives. 1) Find the optimal sequence of extracting rules, making it obtains the best results in the given data set; 2) Test the accuracy of the rules extracted; 3) Detect how much knowledge in the neural network has not been extracted; 4) Detect the inconsistency between the extracted rules and the trained neural network.