# CHAPTER 1
# INTRODUCTION

## 1.1 Organization Profile

CS-Globus solutions is a Software exporters been in the business since 2011 and delivered more than 83 projects. We deliver Business Solutions, IT Services for client across states. The quality of our process, the timeline we keep make us unique from other competitor in Share Point. We are an outsourced product development company with world class Off-Shore development.

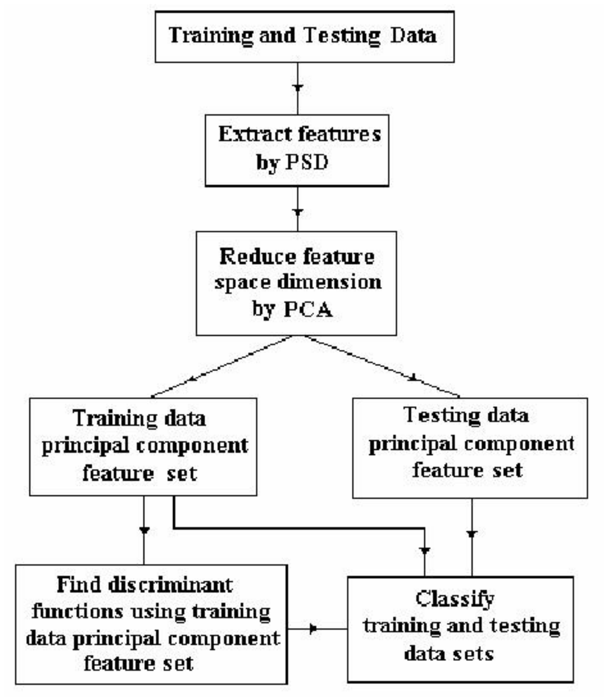

Training and Testing Data
Extract features by PSD
Reduce feature space dimension by PCA
Training data principal component feature set
Testing data principal component feature set
Find discriminant functions using training data principal component feature set
Classify training and testing data sets

**Figure 1.1 Block Diagram of PCA**

### 1.1.1 Skill Capabilities

Human being totally consumer determined we include wealthy familiarity crosswise assorted platforms. We include occasion in addition to another time formed teams so as to comprise closely healthy the necessities of the customer irrespective of CS-Globus's knowledge by means of with the purpose of proficiency. This gives our consumers an assortment of self-reliance in the direction to pull the greatest machinery atmosphere designed for the trade by the side of hand over with no disturbing with reference to admittance in the direction of wherewithal by means of proper skills. Our know-how covers podium technologies (Share Point, .NET), Web services, transportable technologies, communique protocols, set in systems, bequest indoctrination languages with EAI.

### 1.1.2 Procedure Deepness

Our familiarity during cascade iterative, mix RAD, DSDM plus supplementary claim expansion models enable us to provide on the way to assorted customer requests. We boast an interior class administration organization (QMS) to act as an instruction meant for every one of workers during provisions of processes, formats, and that. QMS is a merger of: Industry preeminent practices. Our practices in an excess of a decade within far away execution. Standards approximate ISO as well as CMMI. Learning begins our clients.

### 1.1.3 Engagement Model

Outsourcing is every one of regarding corporation as regards entrusting your commerce method. We encompass in good health prearranged rendezvous models on the road to make sure your reliance. The purpose of a few appointment copy would subsist near afford a essential side enroute for you even as adjusting the plane of promise height of manage savings requisite in cooperation the rate nest egg generated from first to last offshore finances enroute for finest levels intended for the industry.

Seeing as all union, we collaborator by has sole rations consequently on CS-GLOBUS we tag on dissimilar date models meant for delivering our armed forces. We go after the appointment models with the aim of are based on top of unchanging charge with point & cloth superiority administration.

### 1.1.4 Quality Management System

QA / tough services are a fundamental element of our increase method with single of our meat competencies. During CS-GLOBUS, the QA / trying players is implicated within the job accurate beginning daytime lone as of the supplies time. The QA team also participates within the plan meetings. This helps the QA / difficult side identify the eminence as well as trying policy intended for the assignment. Owed toward this level of participation the QA / trying squad has on top of frequent occasions better the value of the escape exact on the mean part.

## 1.2 Problem Description

The early detection of cancer is a challenging problem, due to the structure of the cancer cells, where most of the cells are overlapped with each other, if lung nodules can be identified accurately at an early stage, the survival rate of the patients can be increased by a significant percentage. In the health industry, chest X-rays are considered to be the most widely used technique for the detection of lung cancer. However, because it is difficult to identify lung nodules using raw chest X-ray images, analysis of such medical images has become a tedious and complicated task. This can be corrected by using our proposed method.

## 1.3 System Environment

### 1.3.1 Hardware Specification

This section gives the details and specification of the hardware on which the system is expected to work.

| | | |
|---|---|---|
| System | : | Pentium IV 2.4 GHz |
| Hard Disk | : | 40 GB |
| RAM | : | 256 MB |
| Key Board | : | LG |
| Mouse | : | Logitech |
| Floppy Drive | : | 1.44 MB |
| Monitor | : | 15 inch VGA Color monitor |

### 1.3.2 Software Specification

This section gives the details and specification of the software that are used for the development.

| | | |
|---|---|---|
| Operating System | : | Windows XP. |
| Language | : | Java. |
| IDE | : | Net Beans 7.2 |

## 1.4 Software Description

The software requirement specification is created at the end of the analysis task. The function and performance allocated to software as part of system engineering are developed by establishing a complete information report as functional representation, a representation of system behavior, an indication of performance requirements and design constraints, appropriate validation criteria.

### 1.4.1 Features of Java

Java platform has two components:

- The Java Virtual Machine (Java VM)
- The Java Application Programming Interface (Java API)

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries (packages) of related components.

The following figure depicts a Java program, such as an application or applet, that's running on the Java platform. As the figure shows, the Java API and Virtual Machine insulates the Java program from hardware dependencies.

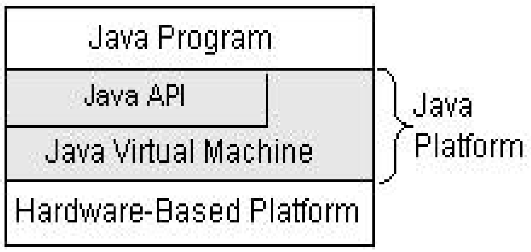


**Figure 1.2 Block diagram of Java Virtual Machines**

As a platform-independent environment, Java can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring Java's performance close to that of native code without threatening portability.

### 1.4.2 Socket Overview

A network socket is a lot like an electrical socket. Various plugs around the network have a standard way of delivering their payload. Anything that understands the standard protocol can "plug in" to the socket and communicate. Internet protocol (IP) is a low-level routing protocol that breaks data into small packets and sends them to an address across a network, which does not guarantee to deliver said packets to the destination. Transmission Control Protocol (TCP) is a higher-level protocol that manages to reliably transmit data. A third protocol, User Datagram Protocol (UDP), sits next to TCP and can be used directly to support fast, connectionless, unreliable transport of packets.

### 1.4.3 Client/Server

A server is anything that has some resource that can be shared. There are compute servers, which provide computing power; print servers, which manage a collection of printers; disk servers, which provide networked disk space; and web servers, which store web pages. A client is simply any other entity that wants to gain access to a particular server.

A server process is said to "listen" to a port until a client connects to it. A server is allowed to accept multiple clients connected to the same port number, although each session is unique. To manage multiple client connections, a server process must be multithreaded or have some other means of multiplexing the simultaneous I/O.

### 1.4.4 Reserved Sockets

Once connected, a higher-level protocol ensues, which is dependent on which port user are using. TCP/IP reserves the lower, 1,024 ports for specific protocols. Port number 21 is for FTP, 23 is for Telnet, 25 is for e-mail, 79 is for finger, 80 is for HTTP, 119 is for Netnews-and the list goes on. It is up to each protocol to determine how a client should interact with the port. TCP is used for reliable stream-based I/O across the network. To manage multiple client connections, a server process must be multithreaded or have some other means of multiplexing the simultaneous I/O. A third protocol, User Datagram Protocol (UDP), sits next to TCP and can be used directly to support fast, connectionless, unreliable transport of packets.

### 1.4.5 Java And The Net

Java supports TCP/IP both by extending the already established stream I/O interface. Java supports both the TCP and UDP protocol families. TCP is used for reliable stream-based I/O across the network. UDP supports a simpler, hence faster, point-to-point datagram-oriented model.

### 1.4.6 Inetaddress

The InetAddress class is used to encapsulate both the numerical IP address and the domain name for that address. User interacts with this class by using the name of an IP host, which is more convenient and understandable than its IP address. The InetAddress class hides the number inside. As of Java 2, version 1.4, InetAddress can handle both IPv4 and IPv6 addresses.

### 1.4.7 Factory Methods

The InetAddress class has no visible constructors. To create an InetAddress object, user use one of the available factory methods. Factory methods are merely a convention whereby static methods in a class return an instance of that class. This is done in lieu of overloading a constructor with various parameter lists when having unique method names makes the results much clearer.

Three commonly used InetAddress factory methods are:

- Static InetAddress getLocalHost ( ) throws UnknownHostException.
- Static InetAddress getByName (String hostName)  throws UnknowsHostException.
- Static InetAddress [ ] getAllByName (String hostName) throws UnknownHostException

The getLocalHost ( ) method simply returns the InetAddress object that represents the local host. The getByName ( ) method returns an InetAddress for a host name passed to it.

On the internet, it is common for a single name to be used to represent several machines. In the world of web servers, this is one way to provide some degree of scaling. The getAllByName ( ) factory method returns an array of InetAddresses that represent all of the addresses that a particular name resolves to. It will also throw an UnknownHostException if it can't resolve the name to at least one address. Java 2, version 1.4 also includes the factory method getByAddress ( ), which takes an IP address and returns an InetAddress object. Either an IPv4 or an IPv6 address can be used.

### 1.4.8 Instance Methods

The InetAddress class also has several other methods, which can be used on the objects returned by the methods just discussed. Here are some of the most commonly used.

- Boolean equals (Object other) - Returns true if this object has the same Internet address.
- byte [ ] get Address ( ) - Returns a byte array that represents the object's Internet address.
- String getHostAddress ( ) - Returns a string that represents the host address associated with the InetAddress object.
- String get Hostname ( ) - Returns a string that represents the host name associated with the InetAddress object.
- boolean isMulticastAddress ( )- Returns true if this Internet address is a multicast address. Otherwise, it returns false.
- String toString ( ) - Returns a string that lists the host name and the IP address for convenience.

### 1.4.9 Tcp/Ip Client Sockets

TCP/IP sockets are used to implement reliable, bidirectional, persistent, point-to-point and stream-based connections between hosts on the Internet. A socket can be used to connect Java's I/O system to other programs that may reside either on the local machine or on any other machine on the Internet.

There are two kinds of TCP sockets in Java. One is for servers, and the other is for clients. The Server Socket class is designed to be a "listener," which waits for clients to connect before doing anything. The Socket class is designed to connect to server sockets and initiate protocol exchanges.

The creation of a Socket object implicitly establishes a connection between the client and server. There are no methods or constructors that explicitly expose the details of establishing that connection.

Here are two constructors used to create client sockets:

- Socket (String hostName, int port) - Creates a socket connecting the local host to the named host and port; can throw an UnknownHostException or anIOException.
- Socket (InetAddress ipAddress, int port) - Creates a socket using a preexisting InetAddress object and a port; can throw an IOException.

A socket can be examined at any time for the address and port information associated with it, by use of the following methods:

- InetAddress getInetAddress ( ) - Returns the InetAddress associated with the Socket object.
- Int getPort ( ) - Returns the remote port to which this Socket object is connected.
- Int getLocalPort ( ) - Returns the local port to which this Socket object is connected.
- Once the Socket object has been created, it can also be examined to gain access to the input and output streams associated with it. Each of these methods can throw an IOException if the sockets have been invalidated by a loss of connection on the Net.
- InputStream getInputStream ( ) - Returns the InputStream associated with the invoking socket.
- OutputStream getOutputStream ( ) - Returns the OutputStream associated with the invoking socket.

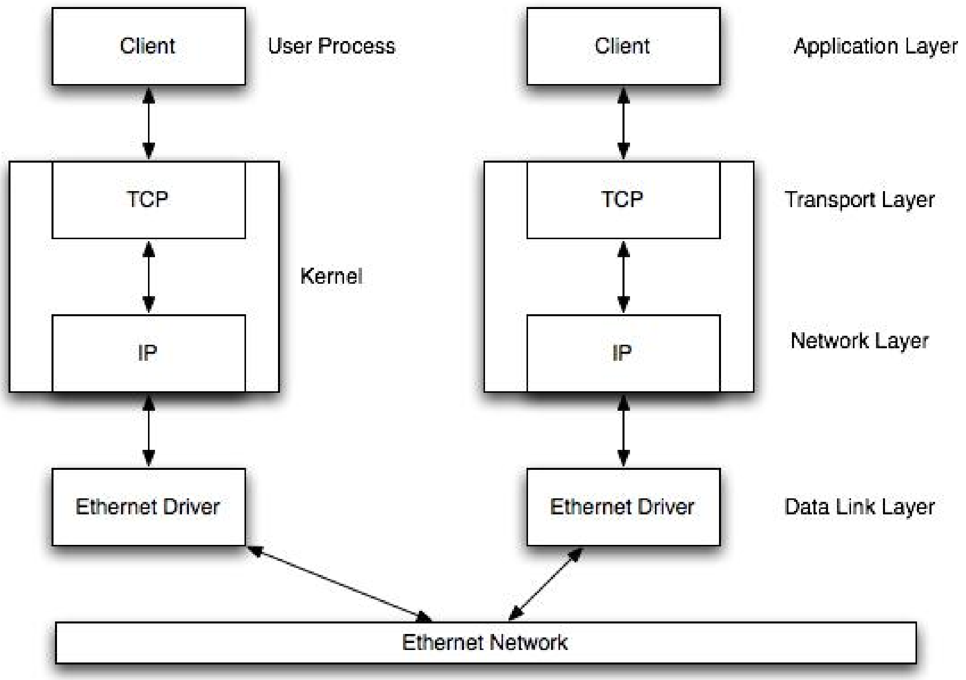


**Figure 1.3 Tcp\Ip Client Socket**

### 1.4.10 Tcp/Ip Server Sockets

Java has a different socket class that must be used for creating server applications. The ServerSocket class is used to create servers that listen for either local or remote client programs to connect to them on published ports. ServerSockets are quite different form normal Sockets. When the user create a ServerSocket, it will register itself with the system as having an interest in client connections.

- ServerSocket(int port) - Creates server socket on the specified port with a queue length 50.
- Serversocket(int port, int maxQueue) - Creates a server socket on the specified port with a maximum queue length of maxQueue.
- ServerSocket(int port, int maxQueue, InetAddress localAddress)-Creates a server socket on the specified port with a maximum queue length of maxQueue. On a multi homed host, localAddress specifies the IP address to which this socket binds.
- ServerSocket has a method called accept( ) - which is a blocking call that will wait for a client to initiate communications, and then return with a normal Socket that is then used for communication with the client.

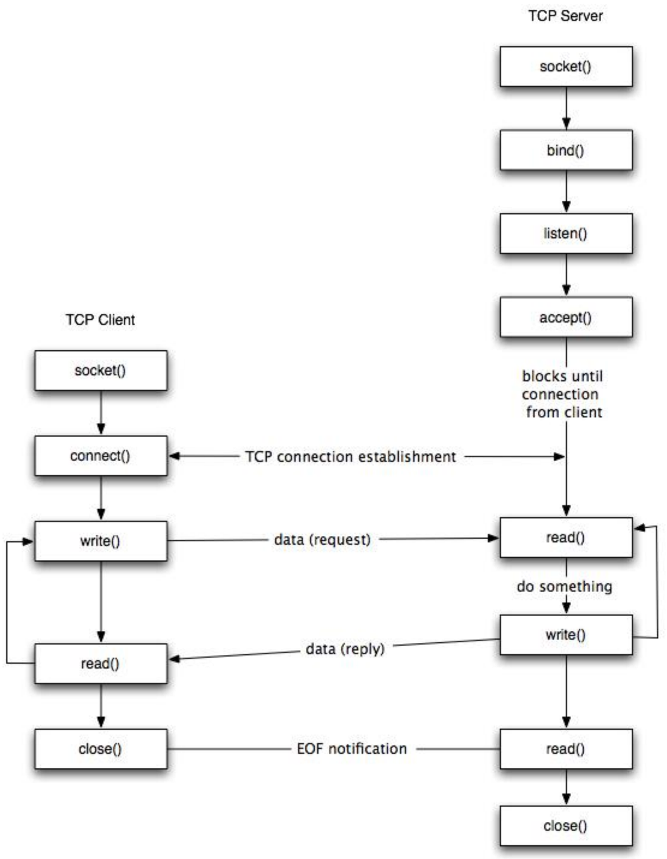


**Figure 1.4 Tcp\Ip Server Socket**

### 1.4.11 URL

The Web is a loose collection of higher-level protocols and file formats, all unified in a web browser. One of the most important aspects of the Web is that Tim Berners-Lee devised a scalable way to locate all of the resources of the Net. The Uniform Resource Locator (URL) is used to name anything and everything reliably.

# CHAPTER 2
# LITERATURE SURVEY

## 2.1 Literature Survey

The early detection of disease is a challenging problem, due to the structure of the cancer cells, where most of the cells are overlapped with each other, if lung nodules can be identified accurately at an early stage, the survival rate of the patients can be increased by a significant percentage. In the health industry, chest X-rays are considered to be the most widely used technique for the detection of lung cancer. However, because it is difficult to identify lung nodules using raw chest X-ray images, analysis of such medical images has become a tedious and complicated task. This can be corrected by using our proposed method.

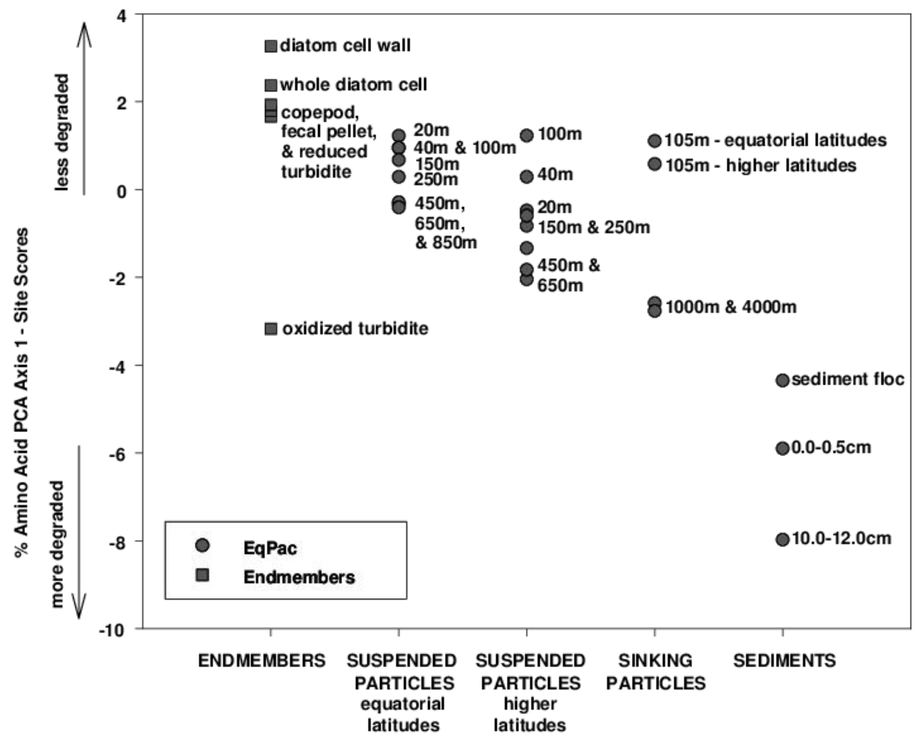


**Figure 2.1 Literature Survey Graph Diagram**

Automated disease classification using machine learning often relies on features derived from segmenting individual objects, which can be difficult to automate. Proposed model is a classification based an efficient approach in which machine learning concepts are used for the detection of cancer diseases. The algorithm obtained encouraging results but requires considerable computational expertise to execute. Furthermore, some benchmark sets have been shown to compare the proposed work model working.

## 2.2 Development

We developed user friendly disease prediction model based on PCA and LDA. To validate the method, the proposed method is applied in MATLAB 2014a to achieve high accuracy performance metric and then comparison has been made with ICA and SURF method. Conclusions: The proposed approach offers improved user-friendliness, as feature extraction is performed in an easily editable. As a direct implication, intermediate results are more easily accessible.

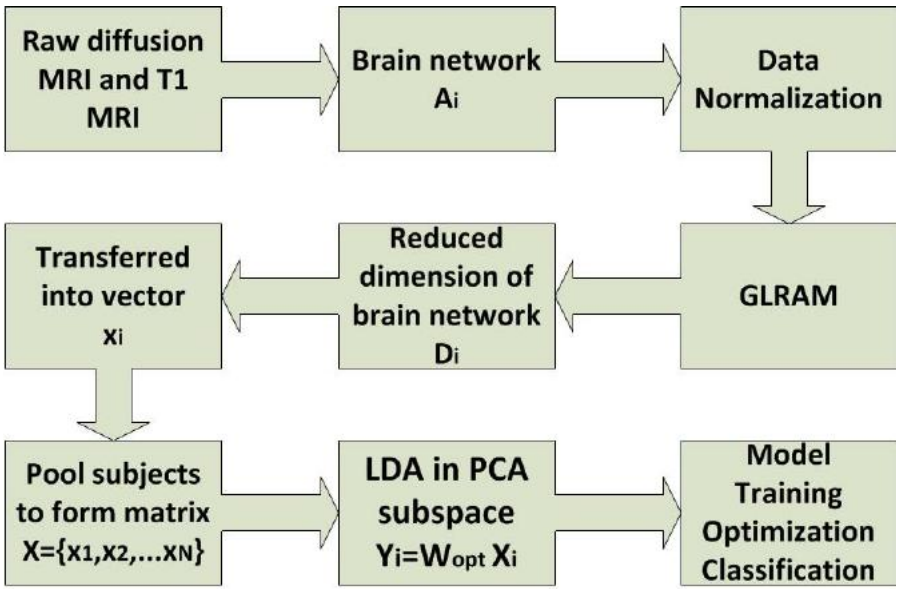

Raw diffusion MRI and T1 MRI
Brain network $A_i$
Data Normalization
GLRAM
Reduced dimension of brain network $D_i$
Transferred into vector $x_i$
Pool subjects to form matrix $X=\{x_1,x_2,...x_N\}$
LDA in PCA subspace $Y_i=W_{opt} X_i$
Model Training Optimization Classification

**Figure 2.2 Development of PCA**

Automated disease classification using machine learning often relies on features derived from segmenting individual objects, which can be difficult to automate. Proposed model is a classification based an efficient approach in which machine learning concepts are used for the detection of cancer diseases. The algorithm obtained encouraging results but requires considerable computational expertise to execute. Furthermore, some benchmark sets have been shown to compare the proposed work model working.

# CHAPTER 3
# SYSTEM ANALYSIS

## 3.1 Existing System

In existing system, this is impractical in our scenario for two reasons. First, it may have the undesirable effect of changing a cancer patient's existing test data groups, potentially undoing the Patient's own manual efforts in organizing her history. Second, it involves a high computational cost, since we would have to repeat a large number of attribute test data group similarity computations for every new test data.

As existing approaches to extract cancer branch selection feature extraction severity percentage suffer from scalability, it is imperative to address the scalability issue. Connections in cancer prediction are not homogeneous. This relation-type information, however, is often not readily available in cancer prediction. A direct application of collective inference or label propagation would treat connections in a branch selection feature extraction network as if they were homogeneous.

### 3.1.1 Draw Backs

- We motivate and propose a method to perform test data grouping in a dynamic fashion. Our goal is to ensure good performance while avoiding disruption of existing patient-defined test data groups.
- Less security.
- Poor performance.
- Complex data processing to find cancer prediction.
- The data retrieval based on user requirement is not done.

## 3.2 PROPOSED SYSTEM

The proposed framework based on branch selection feature extraction severity percentage is shown to be effective in addressing this heterogeneity. The framework suggests a novel way of network classification: first, capture the latent affiliations of actors by extracting branch selection feature extraction severity percentage based on network connectivity, and next, apply extant data mining techniques to classification based on the extracted severity percentage. The proposed framework based on branch selection feature extraction severity percentage is shown to be effective in addressing this heterogeneity

In the initial study, modularity maximization was employed to extract branch selection feature extraction severity percentage.The superiority of this framework over other representative relational learning methods has been verified with cancer prediction cancer data .we proposes an effective edge-centric approach to extract sparse branch selection feature extraction  severity percentage. We prove that with our proposed approach, sparsity of branch selection feature extraction severity percentage is guaranteed. We investigate how signals from search logs such as test data reformulations and clicks can be used together to determine the relevance among test data groups.

### 3.2.1 Advantages

- We show through comprehensive experimental evaluation the effectiveness and the robustness of our proposed search log-based method, especially when combined with approaches using other signals such as text similarity.
- We will focus on evaluating the effectiveness of the proposed algorithms in capturing test data relevance.
- Test data reformulation graph and the test data click graph into a single graph that we refer to as the test data fusion graph, and by expanding the test data set when classification relevance to also include other patient id.
- Relevance Measure.
- Online test data grouping process
- High Similarity functions provide good 3 types of cancer data classification.

## 3.3 Project Description

### 3.3.1 Data Visualization And Pre-Processing

In this module Wisconsin Prognostic Breast Cancer dataset is downloaded from the UCI Machine Learning Repository and saved as a text file. This file is then imported into Excel spreadsheet and the values are saved with the corresponding attributes as column headers. In this preprocessing state the missing values are replaced with appropriate values. The ID of the patient cases does not contribute to the classifier performance. Hence it is removed and the outcome attribute defines the target or dependant variable thus reducing the feature set size to 33 attributes. The algorithmic techniques applied for feature relevance analysis and classification are elaborately presented in the following sections.This file is then imported into Excel spreadsheet and the values are saved with the corresponding attributes as column headers. In this preprocessing state the missing values are replaced with appropriate values.

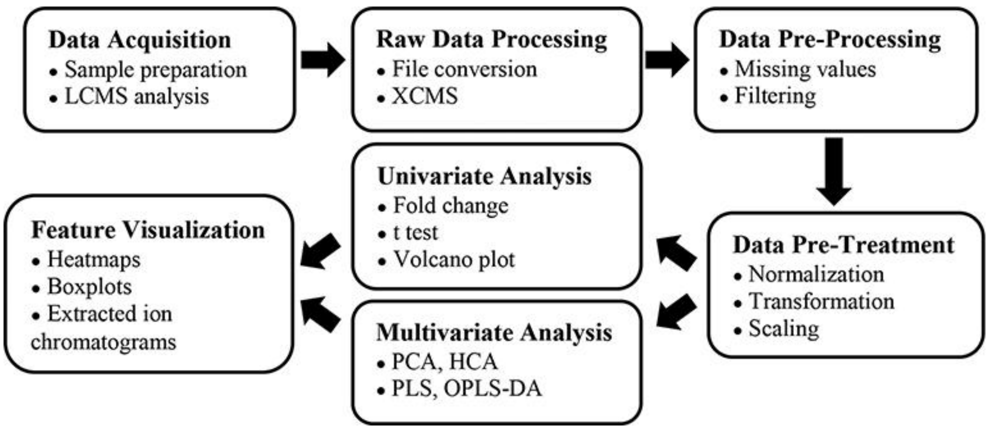


**Figure 3.1 Block Diagram of Data Visualization**

### 3.3.2 Feature Selection Algorithms

The generic problem of supervised feature selection PCA can be outlined as follows. we aim to find a feature subset of size m which contains the most informative features. The two well-performing feature selection algorithms on the WPBC dataset are briefly outlined below.

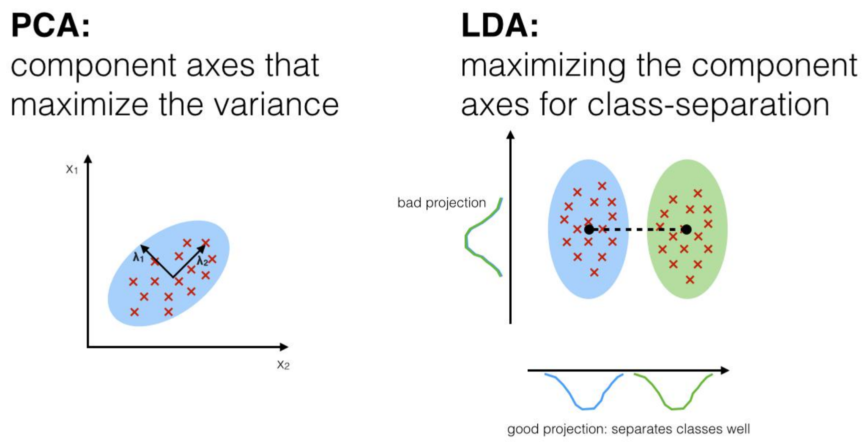

PCA:
component axes that maximize the variance
$x_1$
$x_2$
$\lambda_1$
$\lambda_2$
LDA:
maximizing the component axes for class-separation
bad projection
good projection: separates classes well

**Figure 3.2 Features of Selection Algorithm**

### 3.3.3 Fisher Filtering

This module is used to  filter that ranks the input attributes according to their relevance. A cutting rule enables the selection of a subset of these attributes. It is required to define the target attribute which in this domain of research applies to the nature of the breast cancer (recurrent/non-recurrent) and the predictor attributes. After computing the Fisher score for each feature, it selects the top-m ranked features with large scores. The next subsection directs focus on another technique of feature selection based on logistic regression.

### 3.3.4 Feature Reduction By Pca

Feature reduction applies a mapping of the multidimensional space into a space of lower dimensions. Feature extraction includes features construction, space dimensionality reduction, sparse representations, and feature selection all these techniques are commonly used as preprocessing to machine learning and statistics tasks of prediction, including pattern recognition. Although such problems have been tackled by researchers for many years, there has been recently a renewed interest in feature extraction. The feature space having reduced features truly Contributes to classification that cuts preprocessing costs and minimizes the effects of the 'peaking phenomenon' in classification.

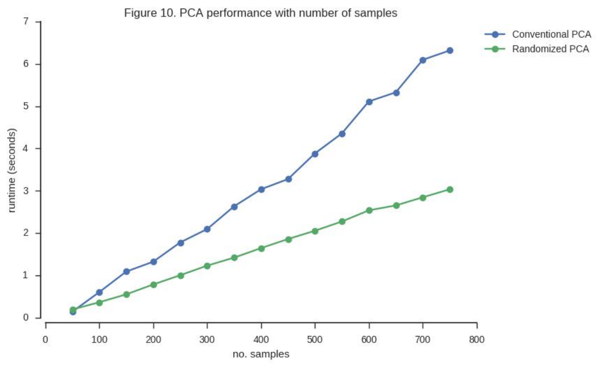


**Figure 3.3 PCA Performance**

### 3.3.5 Classification Scoring

When the number of descriptors is very large for a given problem domain, a learning algorithm is faced with the problem of selecting a relevant subset of features backward regression includes regression models in which the choice of predictor variables is carried out by an automatic procedure.

Step 1: The feature set with all 'ALL' predictors.
Step 2: Eliminate predictors one by one.
Step 3: 'ALL' models are learnt containing 'ALL-1' descriptor each.

These iterations are further continued till either a pre-specified target size is reached or the desired performance statistics is obtained. After feature relevance, we classify the nature of the breast cancer cases in the Wisconsin Prognostic Breast Cancer dataset using twenty classification algorithms.

# CHAPTER 4
# SYSTEM DESIGN

## 4.1 System Flow Diagram

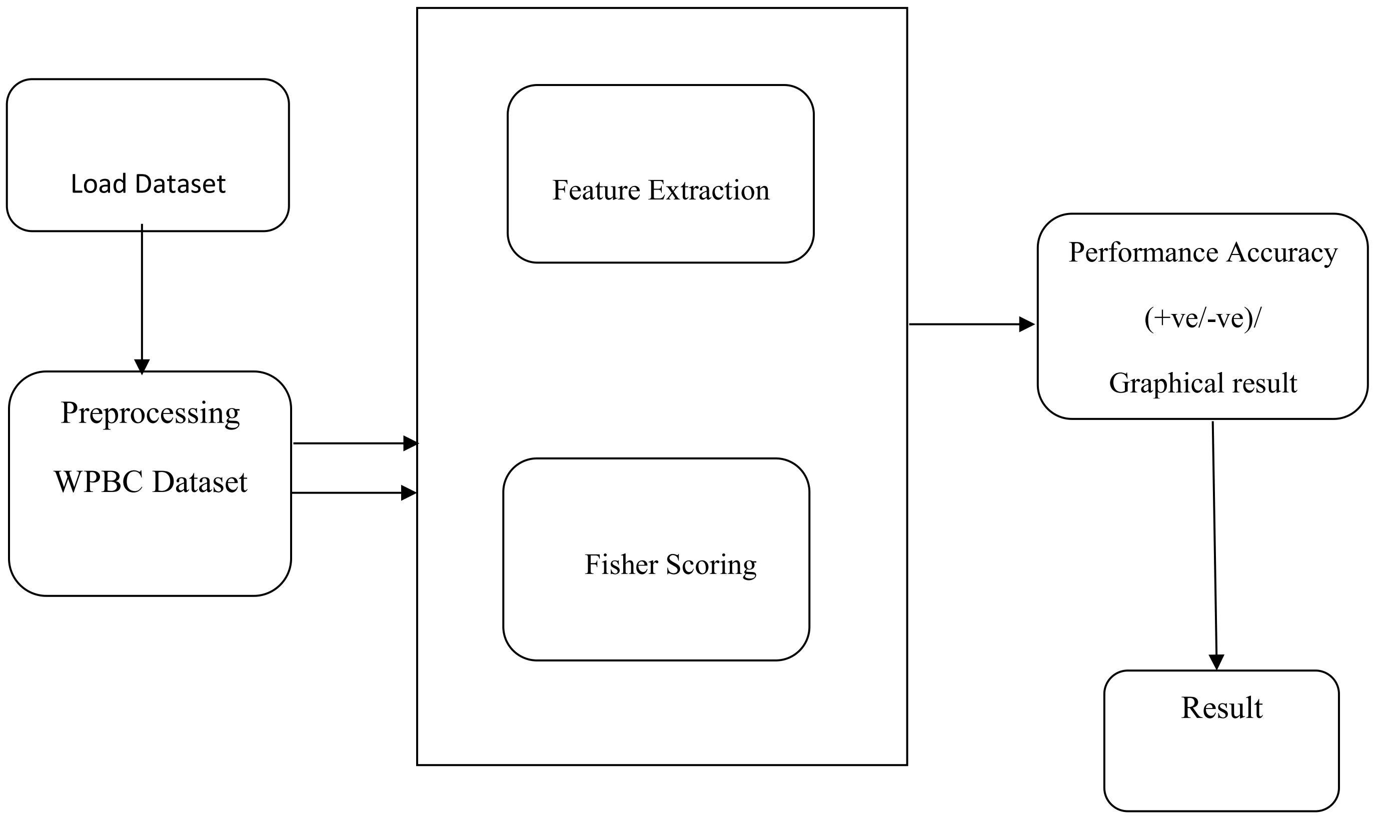


**Figure 4.1 Block Diagram of Flow Diagram**

## 4.2 Input Design

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system.

The input design consist two forms

- Pre-processing Form
- Attribute Selection Form

### 4.2.1 Pre-Processing Form

The amount of data decreased due to the existence of missing values, gross errors and dead band errors, which have been removed from the dataset. However, replacing the removed and missing values is beyond the scope of this research. Furthermore, the dataset does not include seasonal effects for accurate prediction of upcoming environmental variable.

In order to identify temporal outliers, are utilized to obtain temporal patterns and differences of the real-life data set, named Wisconsin Breast Cancer [8] is used. The data set is publicly available on UCI machine learning repository and consists of 699 instances with nine continuous attributes.

### 4.2.2 Attribute Selection Form

Concept-independent preprocessing and concept-specific sampling. The first step is concept-independent, thus allowing for learning different concepts at a later stage. The second step is concerned with the actual selection of a relevant subset of the instances once the concept of interest has been identified.

Following the selection of the subset, the relevant learning algorithm (EigFusion) is applied to obtain the classifier.The K-SVMs as the classifier algorithm, multiple hyperplanes would need to be learned, one for each class. Essentially, the second step dictates the reuse of the training dataset with varying labels. Such a scenario is common in data repositories where the data is available for pre-processing with periodic updates adding/removing instances.

## 4.3 Output Design

The output Design which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

Result Extraction that presented a novel clustering algorithm EigFusion is designed to perform clustering on rich structured multivariate datasets. it have shown that the applicability of K- Support Vector Machines are not limited to classification problems. Even in the absence of labelled training instances for K-SVM, generating labels on-the-fly effectively increases clustering performance.

# CHAPTER 5
# SYSTEM TESTING AND IMPLEMENTATION

## 5.1 System Testing

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work document. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished document. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

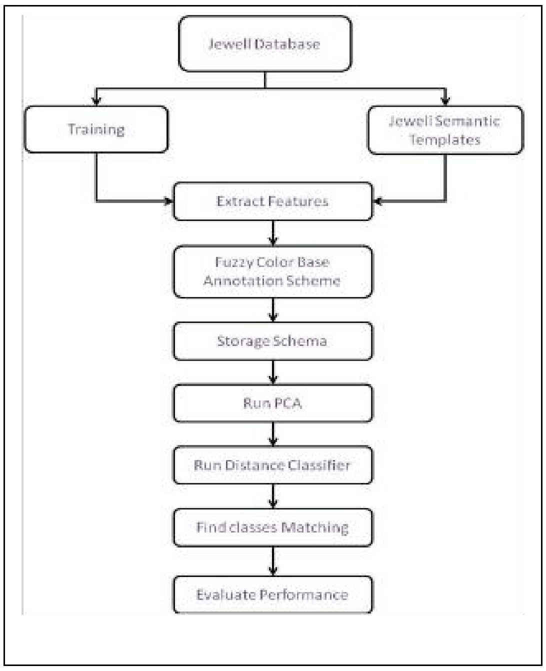


**Figure 5.1 System Testing**

### 5.1.1 Objectives Of Testing

- Testing is the process of executing a program with the intent of finding an error.
- A successful test is one that uncovers a discovered the error.

### 5.1.2 Types Of Testing

- Unit testing
- Validation testing
- Output Testing

### 5.1.2.1 Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration.

We implemented the Six algorithms implementation using the selecting text file generation optimization. Additionally, we implemented the Sequential Pattern, Hybrid and Inverse Term Frequency algorithms as presented in the previous section. All these algorithms were implemented in Java using several of the data structures provided by the JAVA Standard Template Library.

All experiments reported in this thesis were performed on a 1.40GHz Intel Processor with 1GB main memory, running Windows Xp. It shows the performance of the algorithms on each of the data sets described in Section 5 for varying minimal support thresholds. The first interesting behavior can be observed in the experiments for the Selected file data. Indeed, Sequential Pattern performs much worse than all other algorithms. Nevertheless, this behavior has been predicted since the number of frequent items in the Selected file data set is very large and hence, a huge amount of candidate 2-itemsets is generated.

The other algorithms all use dynamic candidate generation of 2-itemsets resulting in much better performance results. The Hybrid algorithm performed best when K-ROSE with Eig Fusion results on testing for a unit root in the presence of additive outliers. Two procedures are proposed. The first procedure is to ignore the possibility of additive outliers and use modified Eig-Fusion statistics.

The second procedure uses a new and very simple outlier detection statistic to identify outliers and then properly adjust standard K-ROSE unit root tests.

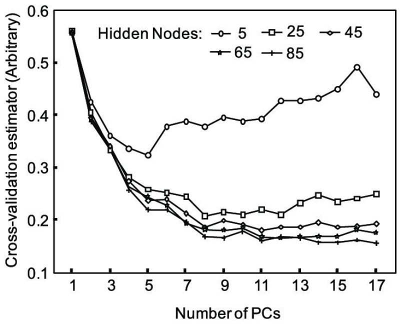


**Figure 5.2 Testing Graph**

#### 5.1.2.2 Validation Testing

The procedure of validation testing is to keep the system safe from errors. The basic purpose is to conform that the system satisfies the necessary conditions. This testing is planned when the incorrect data is given, the user receives the error message. A test plan output lines the classes of test to be conducted and test a procedure device specific test case that will be used to demonstrate conformity with the requirements. Being different from normal system behavior, intrusion detection is a candidate for applying outlier detection techniques.

The key challenges for outlier detection are :-

- Huge Data Volume: This calls for computationally efficient techniques.
- Streaming Data: This requires on-line analysis.
- False alarm rate: Smallest percentage of false alarms among millions of data objects can make be overwhelming for an analyst.
- Labeled data not usually available for Intrusions: This gives preference to Eig-Fusion and K-ROSE outlier detection techniques.

#### 5.1.2.3 Output Testing

After performing the validation testing the next step is the output testing of the proposed system. Since no system could be useful if it does not produce the required output format. Asking the users about format required by them test the output generated or displayed by the system under consideration.

Here the output form is considered on screen and in printed format. An important aspect for any outlier detection technique is the manner in which the outliers are reported. Typically, the outputs produced by outlier detection techniques are one of the following two types:

- Scores: Scoring techniques assign an outlier score to each instance in the test data depending on the degree to which that instance is considered an outlier. Thus the output of such techniques is a ranked list of outliers. An analyst may choose to either analyze top few outliers or use a cut-off threshold to select the outliers.
- Labels: Techniques in this category assign a label (normal or anomalous) to each test instances.

## 5.2 System Implementation

The implementation is the final phase. It involves user system testing and running the developed proposed system. The testing phase involves the testing of developed system using various kinds of data. While testing, errors are noted and corrections are made.

The implementation consists of the following steps:

- Testing the developed programs with sample data.
- Identification of any errors while running the project.
- Creating the files of the system with the actual data.
- Training of users to adapt them to the new operating system.

The software has been checked sample data. The changes being made as per the user requirement and will run parallel with the manual system to find out any errors. System Implementation is the most important part in the software development. Maintenance is the most crucial stage in achieving a successful system and giving the users confidence that the new system is workable and effective. The first step in system maintenance is demonstrating the working of the system to the user.

The method of implementation for the new system, is existing system is operated along with the new system. Later on existing system is replaced with the new system. This parallel method of implementation helps the user to get trained to the new system and replacing of the system takes place without affecting the current working of the system.

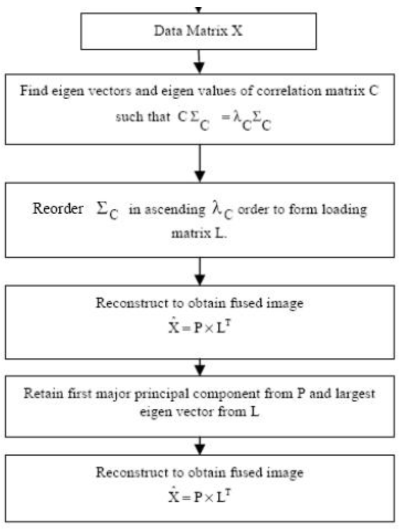

Data Matrix X
Find eigen vectors and eigen values of correlation matrix C such that $C\Sigma_C = \lambda_C \Sigma_C$
Reorder $\Sigma_C$ in ascending $\lambda_C$ order to form loading matrix L.
Reconstruct to obtain fused image $\dot{X} = P \times L^T$
Retain first major principal component from P and largest eigen vector from L
Reconstruct to obtain fused image $\dot{X} = P \times L^T$

**Figure 5.3 System Implementation Data**

Each program is tested individually at the time of development using the data and has verified that this program linked together in the way specified in the programs specification. The computer system and its environment are tested to the satisfaction of the user. Implementation is the process of converting a new system design into operation.

It is the phase that focuses on user training, site preparation and file conversion for installing a candidate system is the leading cause of cancer death worldwide, but techniques for effective early diagnosis are still lacking. In this project, a classification method was developed based on principal components of spectral data. This method was applied to SELDI spectral data from dataset. Unlike other peak-selection-based methods, this method takes each spectrum as a unity. The aim of this project was to demonstrate that this unity-based classification method is more robust and powerful as a method of diagnosis than peak-selection-based methods.

# CHAPTER 6
# SCOPE FOR FUTURE ENHANCEMENT

## 6. Scope For Future Enhancement

In future we have detect outlier information for the development of mobile device technology and localization technology makes the collection of spatio-temporal information from moving objects much easier than before, and outlier detection for spatio-temporal trajectory is becoming increasingly attractive to data mining community. However, there is a lack of serious studies in this area. Several existing trajectory outlier methods such as the partition-and-detect framework can only deal with the trajectory data which only includes spatial attributes. It cannot be applied to the spatio-temporal trajectory data which includes both spatial and temporal attributes. Future work, we propose an enhanced partition-and-detect framework to detect the outliers of spatio-temporal trajectory data.

# CHAPTER 7
# CONCLUSION

## 7.1 Conclusion

Traditional clustering algorithms do not handle rich structured data well by either focusing on a single homogeneous type or by disarding the interrelationships between the multiple aspects of the data find outliers. Hence, those algorithms are not sufficient to deal with the existing (and emerging) data that is heterogeneous in nature, where relationships between objects can be represented through multiple layers of connectivity and similarity. A remarkable note should be made for the definition of a new set, called kernel set, that has been demonstrated to be able to generate the "same" output results in terms of rough outlier set with time computational benefits. The experimental results on three real-world data sets prove that the performance of K-NN in methods.

In our work , we presented a novel clustering algorithm K-NNeans which is designed to perform clustering on rich structured multivariate datasets. We have shown that the applicability of Support Vector Machines are not limited to classification problems and SVM classification can greatly effect the performance of clustering algorithms for multivariate datasets. Even in the absence of labeled training instances for SVM, generating labels on-the-fly effectively increases clustering performance.

Our experimental results on the integration of authorship analysis with topical clustering of documents show significant improvements over traditional ROSE and confirms that there is great benefit in incorporating additional dimensions of similarity into a unified clustering solution. Since spatiotemporal outlier detection might turn out to be useful in many different research fields, we hope that this work will spark further interest in such problems that are challenging and relatively unexplored.

# APPENDIX 1
# SAMPLE CODE

## 8.1 Coding

**Classification.Java**

```
/*
* To change this template, choose Tools | Templates
* and open the template in the editor.
package wpbc;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.BufferedReader;
import java.io.FileReader;
import java.awt.BorderLayout;
import weka.core.Instances;
import weka.classifiers.trees.RandomTree;
import weka.classifiers.trees.REPTree;
import weka.gui.treevisualizer.PlaceNode2;
import weka.gui.treevisualizer.TreeVisualizer;
/**
*
* @author seabirds
*/
public class Classification
{
MainFrame mf;
PreprocessFrame pf;
Classification(MainFrame me,PreprocessFrame pe)
{
mf=me;
pf=pe;
}
```

```
public void getData1()
{
try
{
String ar="@relation wp\n";
for(int i=0;i<mf.colName.length-1;i++)
{
ar=ar+"@attribute "+mf.colName[i]+" numeric\n";
}
ar=ar+"@attribute class {";
for(int i=0;i<mf.cls.size();i++)
ar=ar+mf.cls.get(i).toString()+",";
ar=ar.substring(0, ar.lastIndexOf(","))+"}\n";
System.out.println(ar);
ar=ar+"@data\n";
for(int i=0;i<mf.data.length;i++)
{
String g1="";
for(int j=0;j<mf.data[0].length;j++)
{
g1=g1+mf.data[i][j].trim()+"\t";
}
ar=ar+g1+"\n";
}
File fe=new File("wp.arff");
FileOutputStream fos=new FileOutputStream(fe);
fos.write(ar.trim().getBytes());
fos.close();
}
catch(Exception e)
{
e.printStackTrace();
}
}
public void classify2()
```

```
{
try
{
String ar="@relation wp\n";
for(int i=0;i<pf.featName.size();i++)
{
ar=ar+"@attribute "+pf.featName.get(i).toString()+" numeric \n";
}
ar=ar+"@attribute class {";
for(int i=0;i<mf.cls.size();i++)
ar=ar+mf.cls.get(i).toString()+",";
ar=ar.substring(0, ar.lastIndexOf(","))+"}\n";
System.out.println(ar);
ar=ar+"@data\n";
for(int i=0;i<mf.data.length;i++)
{
String g1="";
for(int j=0;j<pf.featNo.size();j++)
{
int k=Integer.parseInt(pf.featNo.get(j).toString());
g1=g1+mf.data[i][k].trim()+"\t";
}
ar=ar+g1+mf.data[i][mf.data[0].length-1]+"\n";
}
File fe=new File("wp.arff");
FileOutputStream fos=new FileOutputStream(fe);
fos.write(ar.trim().getBytes());
fos.close();
REPTree tree=new REPTree();
Instances data = new Instances(new BufferedReader(new FileReader("wp.arff")));
data.setClassIndex(data.numAttributes() - 1);
tree.buildClassifier(data);
final javax.swing.JFrame jf = new javax.swing.JFrame("Decision Tree ");
jf.setSize(500,400);
jf.getContentPane().setLayout(new BorderLayout());
TreeVisualizer tv = new TreeVisualizer(null,tree.graph(), new PlaceNode2());
jf.getContentPane().add(tv, BorderLayout.CENTER);
jf.addWindowListener(new java.awt.event.WindowAdapter()
{
public void windowClosing(java.awt.event.WindowEvent e)
{
jf.dispose();
}
});
jf.setVisible(true);
tv.fitToScreen();
}
catch(Exception e)
{
e.printStackTrace();
}
}
public void classify1()
{
try
{
String ar="@relation wp\n";
for(int i=0;i<pf.featName.size();i++)
{
ar=ar+"@attribute "+pf.featName.get(i).toString()+" numeric \n";
}
ar=ar+"@attribute class {";
for(int i=0;i<mf.cls.size();i++)
ar=ar+mf.cls.get(i).toString()+",";
ar=ar.substring(0, ar.lastIndexOf(","))+"}\n";
System.out.println(ar);
ar=ar+"@data\n";
for(int i=0;i<mf.data.length;i++)
```

```
{
String g1="";
for(int j=0;j<pf.featNo.size();j++)
{
int k=Integer.parseInt(pf.featNo.get(j).toString());
g1=g1+mf.data[i][k].trim()+"\t";
}
ar=ar+g1+mf.data[i][mf.data[0].length-1]+"\n";
}
File fe=new File("wp1.arff");
FileOutputStream fos=new FileOutputStream(fe);
fos.write(ar.trim().getBytes());
fos.close();
RandomTree rt=new RandomTree();
Instances data = new Instances(new BufferedReader(new FileReader("wp1.arff")));
data.setClassIndex(data.numAttributes() - 1);
rt.buildClassifier(data);
final javax.swing.JFrame jf = new javax.swing.JFrame("Random Tree with Fisher Filtering");
jf.setSize(500,400);
jf.getContentPane().setLayout(new BorderLayout());
TreeVisualizer tv = new TreeVisualizer(null,rt.graph(), new PlaceNode2());
jf.getContentPane().add(tv, BorderLayout.CENTER);
jf.addWindowListener(new java.awt.event.WindowAdapter()
{
public void windowClosing(java.awt.event.WindowEvent e)
{
jf.dispose();
}
});
jf.setVisible(true);
tv.fitToScreen();
}
catch(Exception e)
e.printStackTrace();
}
```

**Main.Java**

```
/*
* To change this template, choose Tools | Templates
* and open the template in the editor.
*/
package wpbc;
/**
*
* @author seabirds
*/
public class Main {
/**
* @param args the command line arguments
*/
public static void main(String[] args) {
// TODO code application logic here
MainFrame mf=new MainFrame();
mf.setVisible(true);
mf.setResizable(false);
mf.setTitle("Efficient Classifier");
}
}
```

**Test Classify.Java**

```
/*
* To change this template, choose Tools | Templates
* and open the template in the editor.
*/
package wpbc;
import java.io.*;
import java.util.*;
import java.text.*;
import weka.core.*;
import weka.core.Instance;
import weka.core.Instances;
import weka.core.Attribute;
import weka.classifiers.*;
import weka.classifiers.Classifier;
import weka.classifiers.trees.*;
import weka.classifiers.trees.j48.*;
import weka.classifiers.bayes.*;
import weka.filters.unsupervised.attribute.StringToWordVector;
/**
*
* @author seabirds
*/
public class TestClassify
{
private String[]   inputText       = null;
private String[]   inputClasses    = null;
private String     classString     = null;
private Attribute  classAttribute  = null;
private Attribute  textAttribute   = null;
private FastVector attributeInfo   = null;
private Instances  instances       = null;
private Classifier classifier      = null;
private Instances  filteredData    = null;
private Evaluation evaluation      = null;
private Set        modelWords      = null;
private String     delimitersStringToWordVector = "\\s.,:'\\\"()?!";
TestClassify(String[] inputText, String[] inputClasses, FastVector attributeInfo, Attribute
textAttribute, Attribute classAttribute, String classString
{
this.inputText      = inputText;
this.inputClasses   = inputClasses;
this.classString    = classString;
this.attributeInfo  = attributeInfo;
this.textAttribute  = textAttribute;
this.classAttribute = classAttribute;
}
public StringBuffer classify() {
if (classString == null || "".equals(classString)) {
return(new StringBuffer());
}
return classify(classString);
}
public StringBuffer classify(String classString) {
this.classString = classString;
StringBuffer result = new StringBuffer();
instances = new Instances("data set", attributeInfo, 100);
instances.setClass(classAttribute);
try {
instances = populateInstances(inputText, inputClasses, instances, classAttribute, textAttribute);
result.append("DATA SET:\n" + instances + "\n");
filteredData = filterText(instances);
modelWords = new HashSet();
Enumeration enumx = filteredData.enumerateAttributes();
while (enumx.hasMoreElements()) {
Attribute att = (Attribute)enumx.nextElement();
String attName = att.name().toLowerCase();
modelWords.add(attName);
}
classifier = Classifier.forName(classString,null);
classifier.buildClassifier(filteredData);
evaluation = new Evaluation(filteredData);
evaluation.evaluateModel(classifier, filteredData);
result.append(printClassifierAndEvaluation(classifier, evaluation) + "\n");
int startIx = 0;
result.append(checkCases(filteredData, classifier, classAttribute, inputText, "not test", startIx)  +
"\n");
} catch (Exception e) {
e.printStackTrace();
result.append("\nException (sorry!):\n" + e.toString());
}
return result;
}
public StringBuffer classifyNewCases(String[] tests) {
System.out.println("----- classify new case "+tests.length);
StringBuffer result = new StringBuffer();
Instances testCases = new Instances(instances);
testCases.setClass(classAttribute);
String[] testsWithModelWords = new String[tests.length];
int gotModelWords = 0;
for (int i = 0; i < tests.length; i++) {
StringBuffer acceptedWordsThisLine = new StringBuffer();
String[] splittedText = tests[i].split("["+delimitersStringToWordVector+"]");
for (int wordIx = 0; wordIx < splittedText.length; wordIx++) {
String sWord = splittedText[wordIx];
if (modelWords.contains((String)sWord)) {
gotModelWords++;
acceptedWordsThisLine.append(sWord + " ");
}
}
testsWithModelWords[i] = acceptedWordsThisLine.toString();
}
if (gotModelWords == 0) {
result.append("\nWarning!\nThe text to classify didn't contain a single\nword from the modelled
words. This makes it hard for the classifier to\ndo something usefull.\nThe result may be
weird.\n\n");
 }
try {
String[] tmpClassValues = new String[tests.length];
for (int i = 0; i < tmpClassValues.length; i++) {
tmpClassValues[i] = "?";
}
}
}
```

```
testCases = populateInstances(testsWithModelWords, tmpClassValues, testCases, classAttribute,
textAttribute);
System.out.println("---- testcase "+testCases.numInstances());
Instances filteredTests = filterText(testCases);
int startIx = instances.numInstances();
System.out.println("--- classify new case"+startIx+" : "+filteredTests.numInstances());
result.append(checkCases(filteredTests, classifier, classAttribute, tests, "newcase", startIx) + "\n");
} catch (Exception e) {
e.printStackTrace();
result.append("\nException (sorry!):\n" + e.toString());
}
return result;
}
public static Instances populateInstances(String[] theseInputTexts, String[] theseInputClasses,
Instances theseInstances, Attribute classAttribute, Attribute textAttribute) {
System.out.println(theseInputTexts.length);
for (int i = 0; i < theseInputTexts.length; i++)
{
Instance inst = new Instance(2);
inst.setValue(textAttribute,theseInputTexts[i]);
if (theseInputClasses != null && theseInputClasses.length > 0) {
inst.setValue(classAttribute, theseInputClasses[i]);
}
theseInstances.add(inst);
}
System.out.println("populate instacnec "+theseInstances.numInstances());
return theseInstances;
}
public static StringBuffer checkCases(Instances theseInstances, Classifier thisClassifier, Attribute
thisClassAttribute, String[] texts, String testType, int startIx) {
StringBuffer result = new StringBuffer();
try {
result.append("\nCHECKING ALL THE INSTANCES:\n");
Enumeration enumClasses = thisClassAttribute.enumerateValues();
```

```
result.append("Class values (in order): ");
while (enumClasses.hasMoreElements())
{
String classStr = (String)enumClasses.nextElement();
result.append("'" + classStr + "'  ");
}
result.append("\n");
System.out.println("------------check case "+startIx+" : "+theseInstances.numInstances());
for (int i = startIx; i < theseInstances.numInstances(); i++) {
SparseInstance sparseInst = new SparseInstance(theseInstances.instance(i));
sparseInst.setDataset(theseInstances);
result.append("\nTesting: '" + texts[i-startIx] + "'\n");
double correctValue = (double)sparseInst.classValue();
double predictedValue = thisClassifier.classifyInstance(sparseInst);
String predictString = thisClassAttribute.value((int)predictedValue) + " (" + predictedValue + ")";
result.append("predicted: '" + predictString);
if (!"newcase".equals(testType)) {
String correctString = thisClassAttribute.value((int)correctValue) + " (" + correctValue + ")";
String testString = ((predictedValue == correctValue) ? "OK!" : "NOT OK!") + "!";
result.append("' real class: '" + correctString +  "' ==> " +  testString);
}
result.append("\n");
result.append("\n");
}
} catch (Exception e) {
e.printStackTrace();
result.append("\nException (sorry!):\n" + e.toString());
}
return result;
 }
public static Instances filterText(Instances theseInstances) {
System.out.println("---filertext "+theseInstances.numInstances());
StringToWordVector filter = null;
int wordsToKeep = 1000;
```

```
Instances filtered = null;
try {
filter = new StringToWordVector(wordsToKeep);
filter.setOutputWordCounts(true);
filter.setSelectedRange("1");
filter.setInputFormat(theseInstances);
filtered = weka.filters.Filter.useFilter(theseInstances,filter);
} catch (Exception e) {
e.printStackTrace();
}
return filtered;
}
public static StringBuffer printClassifierAndEvaluation(Classifier thisClassifier, Evaluation
thisEvaluation) {
StringBuffer result = new StringBuffer();
try {
result.append("\n\nINFORMATION ABOUT THE CLASSIFIER AND EVALUATION:\n");
result.append("\nclassifier.toString():\n" + thisClassifier.toString() + "\n");
result.append("\nevaluation.toSummaryString(title, false):\n" +
thisEvaluation.toSummaryString("Summary",false)  + "\n");
result.append("\nevaluation.toMatrixString():\n" + thisEvaluation.toMatrixString()  + "\n");
result.append("\nevaluation.toClassDetailsString():\n" +
thisEvaluation.toClassDetailsString("Details")  + "\n");
result.append("\nevaluation.toCumulativeMarginDistribution:\n" +
thisEvaluation.toCumulativeMarginDistributionString()  + "\n");
} catch (Exception e) {
e.printStackTrace();
result.append("\nException (sorry!):\n" + e.toString());
}
return result;
}
public void setClassifierString(String classString) {
this.classString = classString;
}
}
```

# APPENDIX 2
# SCREEN SHOTS

## 8.2 Screen shots

### 8.2.1 Input Data Set

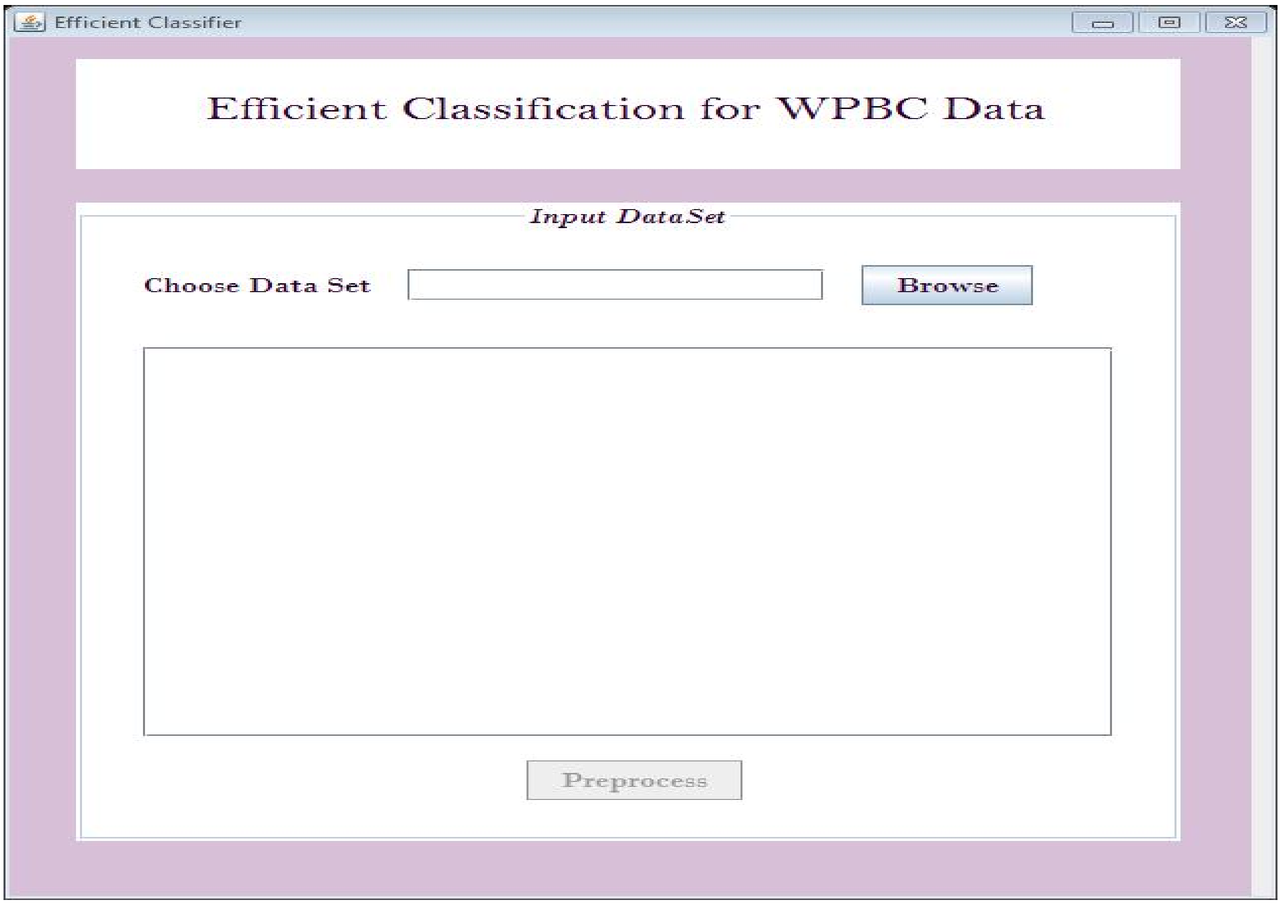


Figure 8.2.1 Input Data Set

### 8.2.2 Select File Path

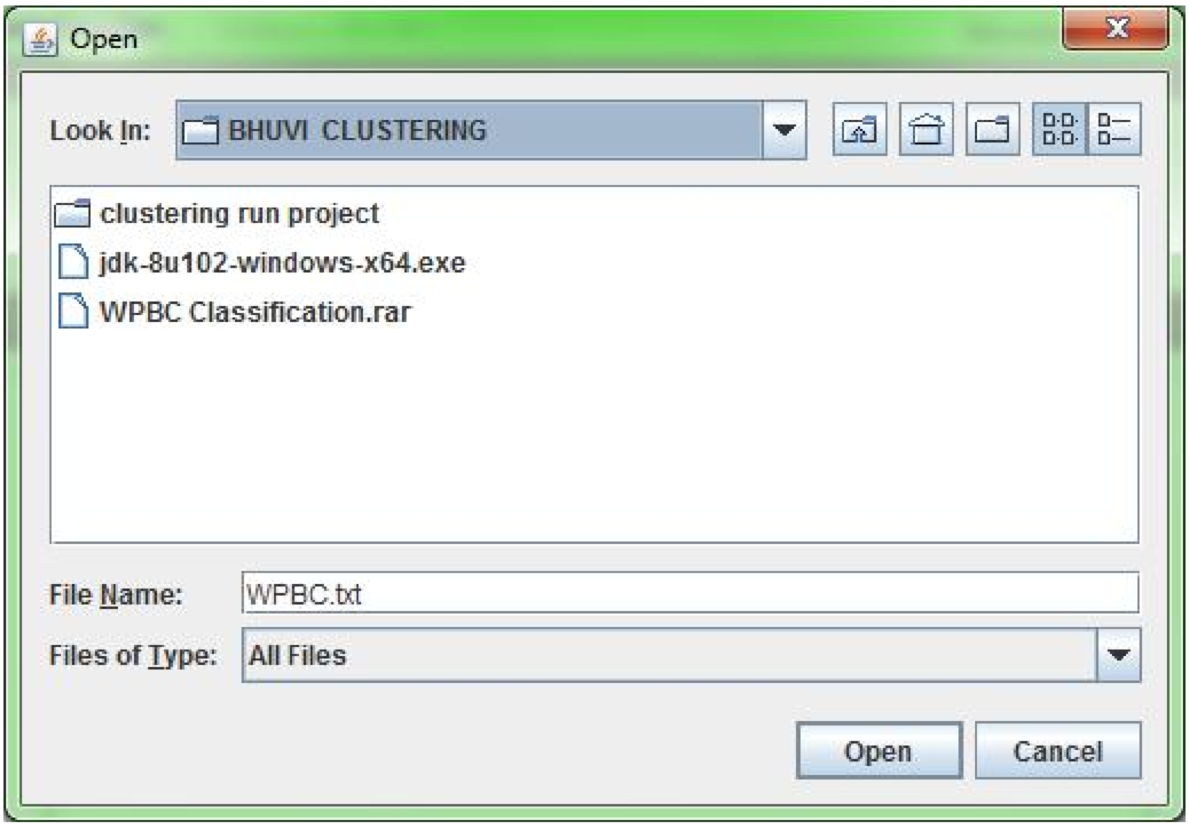
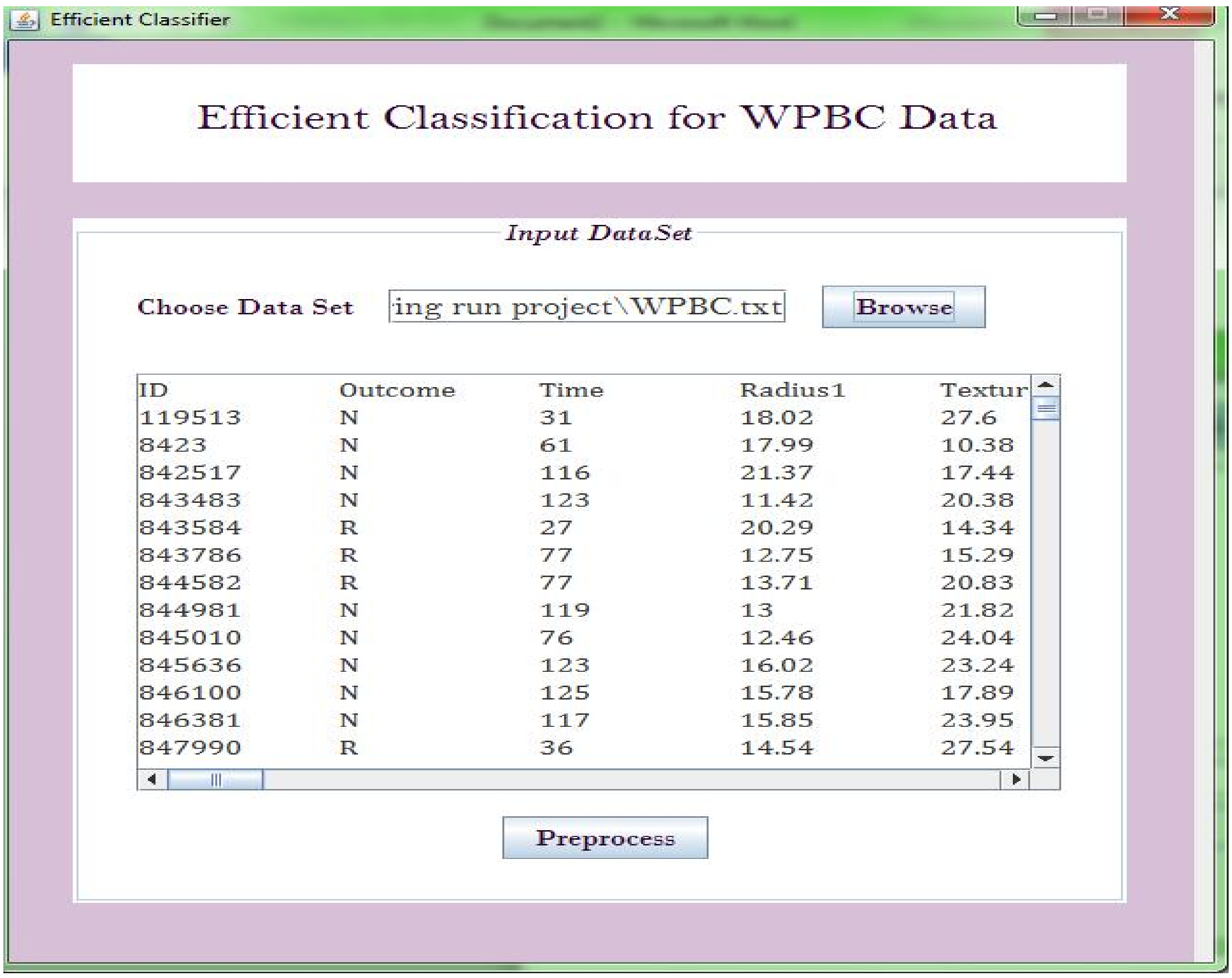


**Figure 8.2.2 File Path**

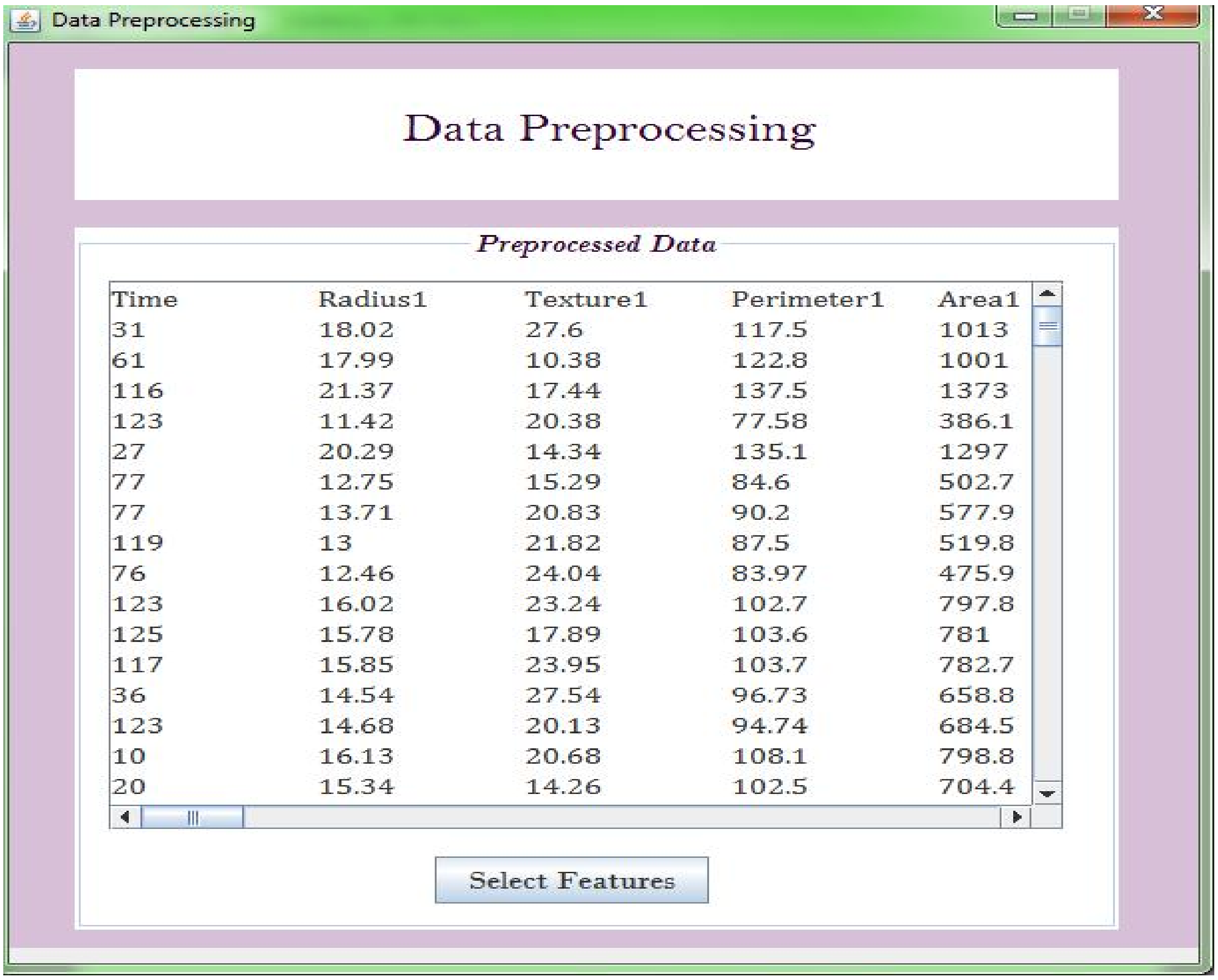
## 8.2.3 Preprocessing



| ID | Outcome | Time | Radius1 | Textur |
|---|---|---|---|---|
| 119513 | N | 31 | 18.02 | 27.6 |
| 8423 | N | 61 | 17.99 | 10.38 |
| 842517 | N | 116 | 21.37 | 17.44 |
| 843483 | N | 123 | 11.42 | 20.38 |
| 843584 | R | 27 | 20.29 | 14.34 |
| 843786 | R | 77 | 12.75 | 15.29 |
| 844582 | R | 77 | 13.71 | 20.83 |
| 844981 | N | 119 | 13 | 21.82 |
| 845010 | N | 76 | 12.46 | 24.04 |
| 845636 | N | 123 | 16.02 | 23.24 |
| 846100 | N | 125 | 15.78 | 17.89 |
| 846381 | N | 117 | 15.85 | 23.95 |
| 847990 | R | 36 | 14.54 | 27.54 |

Figure 8.2.3 Preprocessing

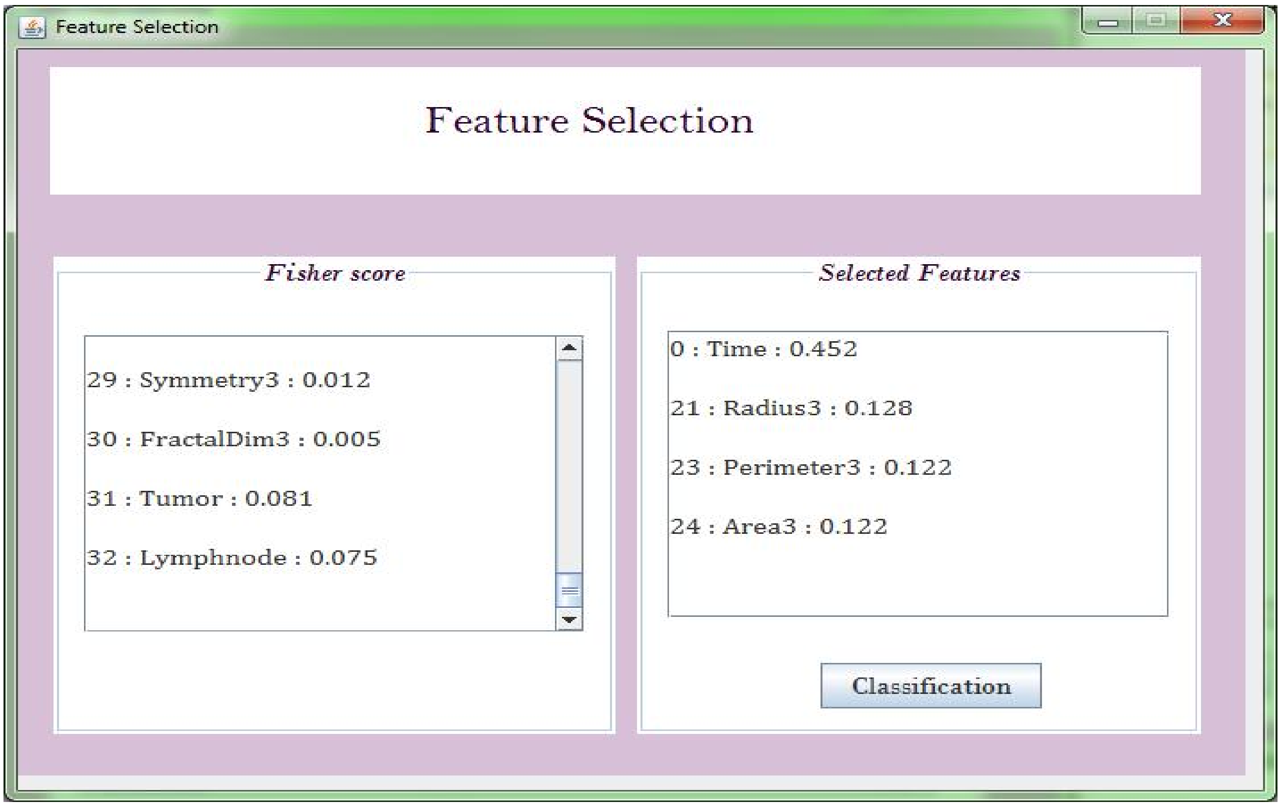
### 8.2.4 Select Feature



| Time | Radius1 | Texture1 | Perimeter1 | Area1 |
|---|---|---|---|---|
| 31 | 18.02 | 27.6 | 117.5 | 1013 |
| 61 | 17.99 | 10.38 | 122.8 | 1001 |
| 116 | 21.37 | 17.44 | 137.5 | 1373 |
| 123 | 11.42 | 20.38 | 77.58 | 386.1 |
| 27 | 20.29 | 14.34 | 135.1 | 1297 |
| 77 | 12.75 | 15.29 | 84.6 | 502.7 |
| 77 | 13.71 | 20.83 | 90.2 | 577.9 |
| 119 | 13 | 21.82 | 87.5 | 519.8 |
| 76 | 12.46 | 24.04 | 83.97 | 475.9 |
| 123 | 16.02 | 23.24 | 102.7 | 797.8 |
| 125 | 15.78 | 17.89 | 103.6 | 781 |
| 117 | 15.85 | 23.95 | 103.7 | 782.7 |
| 36 | 14.54 | 27.54 | 96.73 | 658.8 |
| 123 | 14.68 | 20.13 | 94.74 | 684.5 |
| 10 | 16.13 | 20.68 | 108.1 | 798.8 |
| 20 | 15.34 | 14.26 | 102.5 | 704.4 |

**Figure 8.2.4 Select Feature**

### 8.2.5 Classification



29 : Symmetry3 : 0.012

30 : FractalDim3 : 0.005

31 : Tumor : 0.081

32 : Lymphnode : 0.075

0 : Time : 0.452

21 : Radius3 : 0.128

23 : Perimeter3 : 0.122

24 : Area3 : 0.122

**Figure 8.2.5 Classification**

## 8.2.6 Test Data

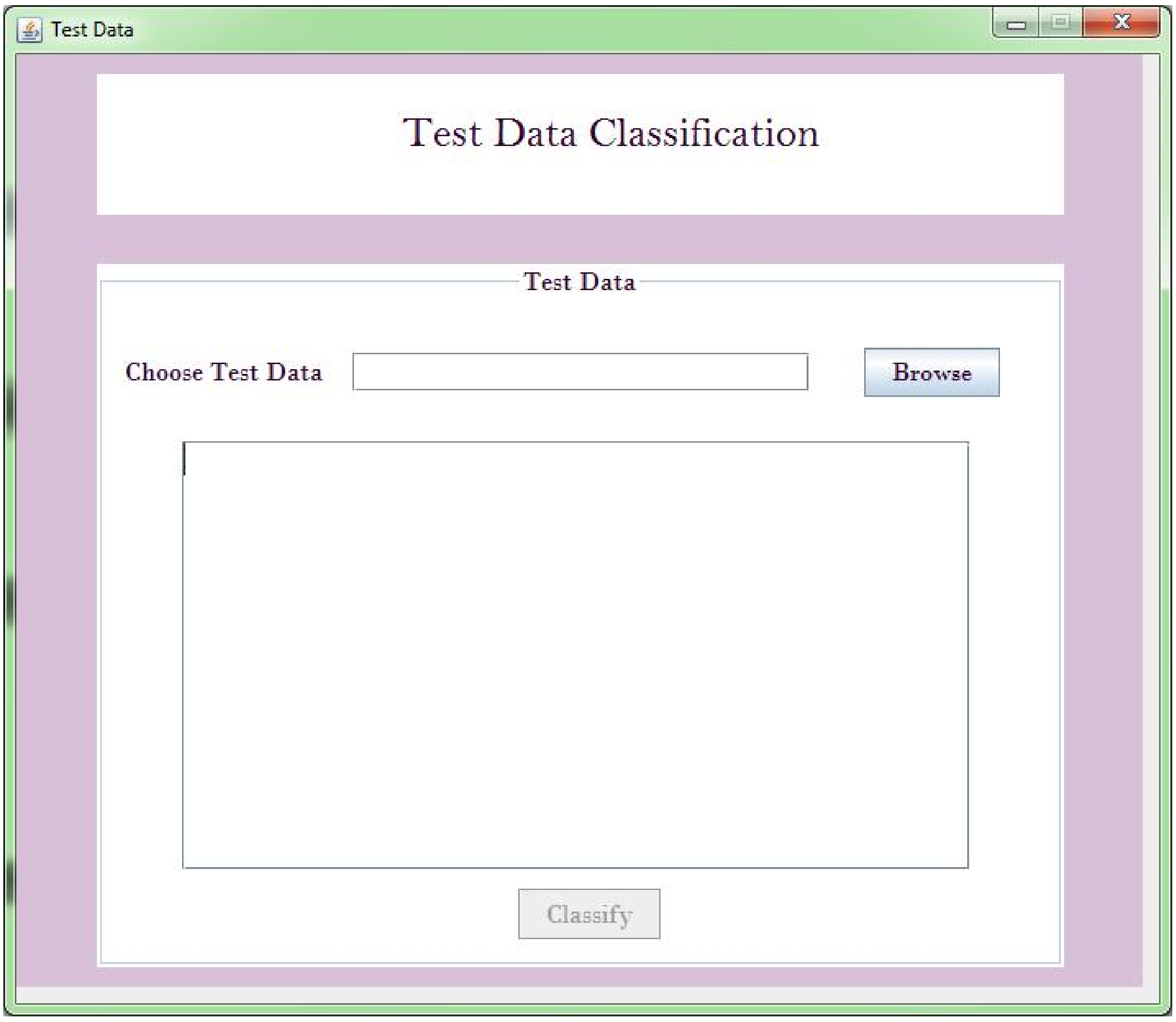


**Figure 8.2.6 Test Data**

## 8.2.7 Select Test Data

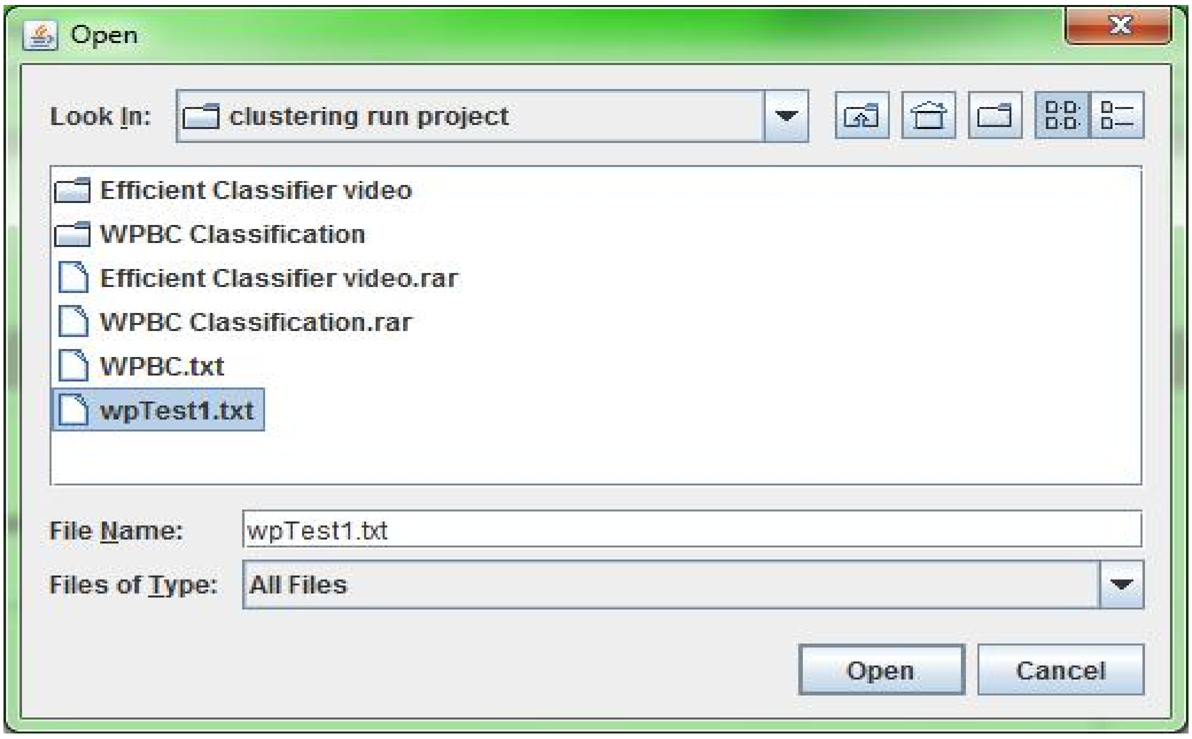
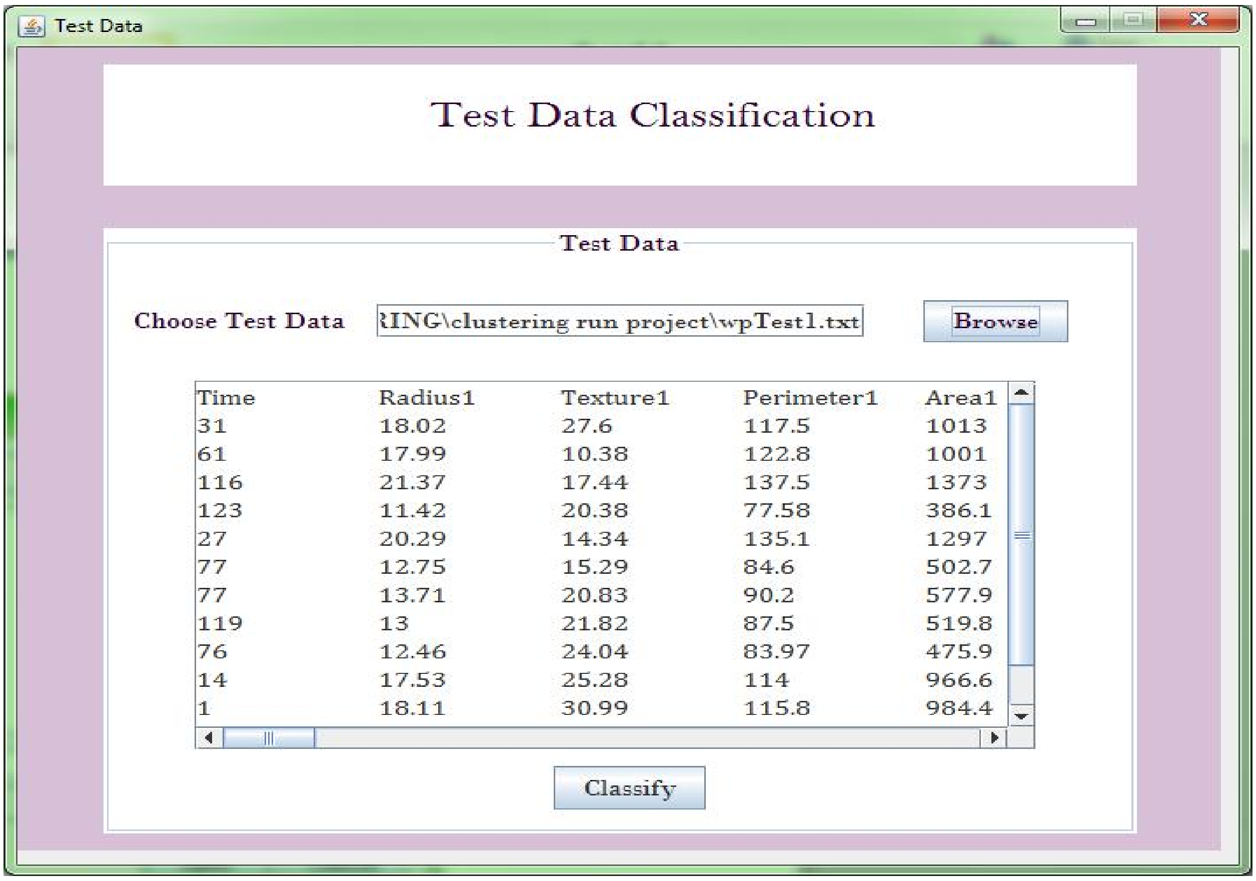


**Figure 8.2.7 Test Data**

## 8.2.8 Test Data Classification



RING\clustering run project\wpTest1.txt

| Time | Radius1 | Texture1 | Perimeter1 | Area1 |
|---|---|---|---|---|
| 31 | 18.02 | 27.6 | 117.5 | 1013 |
| 61 | 17.99 | 10.38 | 122.8 | 1001 |
| 116 | 21.37 | 17.44 | 137.5 | 1373 |
| 123 | 11.42 | 20.38 | 77.58 | 386.1 |
| 27 | 20.29 | 14.34 | 135.1 | 1297 |
| 77 | 12.75 | 15.29 | 84.6 | 502.7 |
| 77 | 13.71 | 20.83 | 90.2 | 577.9 |
| 119 | 13 | 21.82 | 87.5 | 519.8 |
| 76 | 12.46 | 24.04 | 83.97 | 475.9 |
| 14 | 17.53 | 25.28 | 114 | 966.6 |
| 1 | 18.11 | 30.99 | 115.8 | 984.4 |

Figure 8.2.8 Test Data Classification

## 8.2.9 Tree View

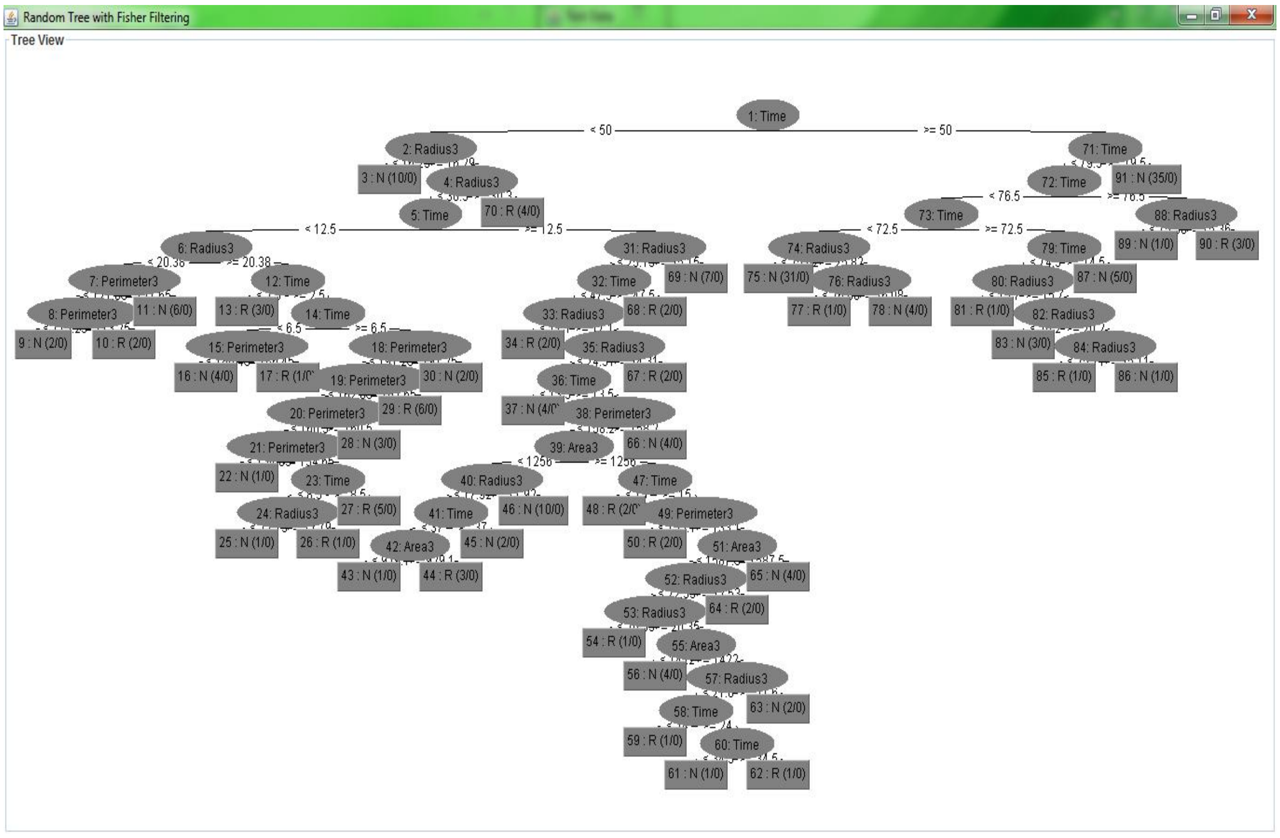


**Figure 8.2.9 Tree View**

## 8.2.10 DECISION TREE

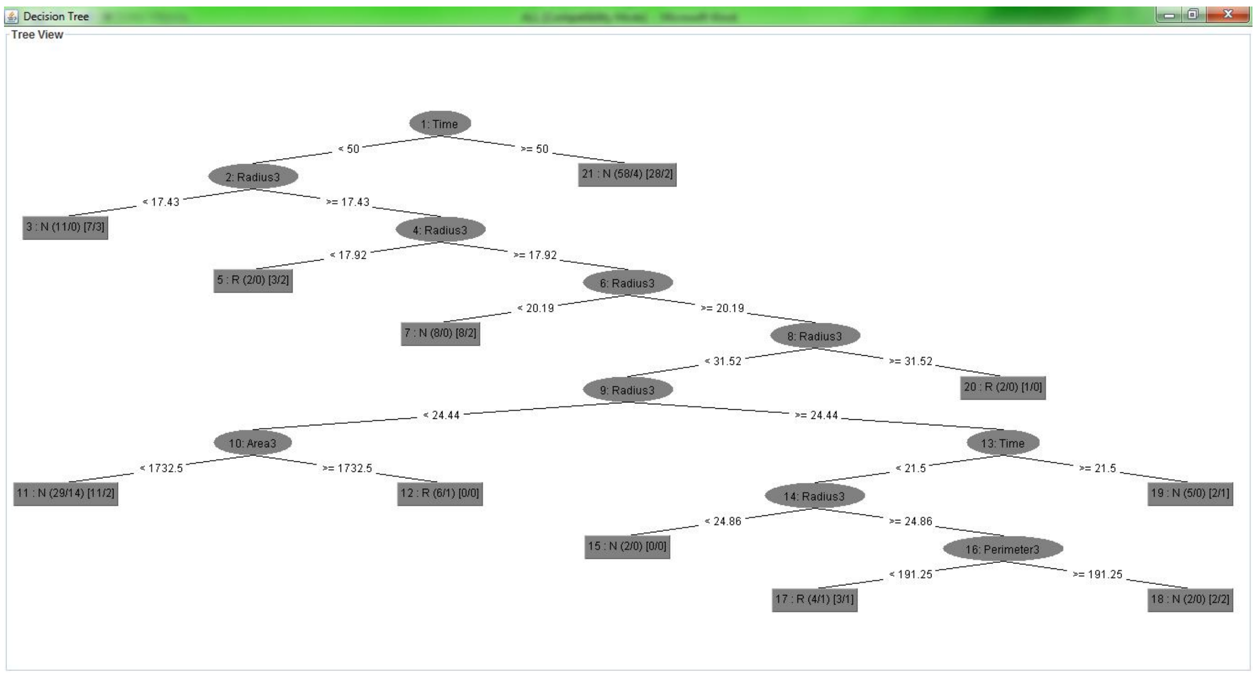
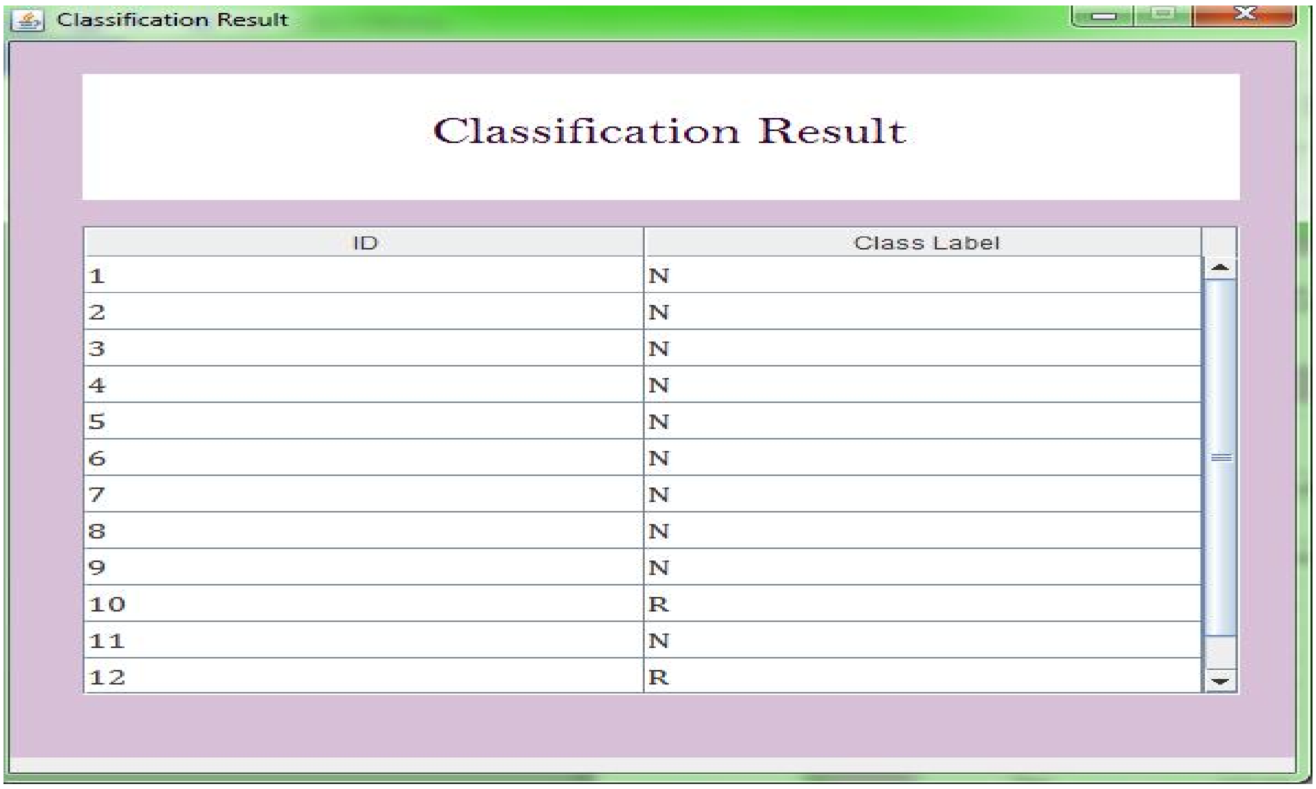


**Figure 8.2.10 Decision Tree**

## 8.2.11 CLASSFICATION RESULT


Classification Result

Classification Result

| ID | Class Label |
|---|---|
| 1 | N |
| 2 | N |
| 3 | N |
| 4 | N |
| 5 | N |
| 6 | N |
| 7 | N |
| 8 | N |
| 9 | N |
| 10 | R |
| 11 | N |
| 12 | R |

**Figure 8.2.11 Result**

# APPENDIX 3
# REFERENCE

[1] C.C. Aggarwal and P. Yu, “Finding Generalized Projected Clusters in High Dimensional Spaces,” Proc. ACM SIGMOD Int’l Conf. Management Data, pp. 70-81, 2000.

[2] C.C. Aggarwal and P.S. Yu, “An Effective and Efficient Algorithm for High-Dimensional Outlier Detection,” VLDB J., vol. 14, pp. 211-221, 2005.

[3] A. Albanese and A. Petrosino, “A Non Parametric Approach to the Outlier Detection in Spatio-Temporal Data Analysis,” Information Technology and Innovation Trends in Organizations, D’Atri, et al., eds., pp. 101-108, Springer Verlag, 2011.

[4] F. Angiulli and C. Pizzuti, “Outlier Mining in Large High- Dimensional Data Sets,” IEEE Trans. Knowledge and Data Eng., vol. 17, no. 2, pp. 203-215, Feb. 2005.

[5] F. Angiulli and F. Fassetti, “Distance-Based Outlier Queries in Data Streams: The Novel Task and Algorithms,” J. Data Mining and Knowledge Discovery, vol. 20, no. 2, pp. 290-324, 2010.