OUTLIER DETECTION ALGORITHM SUIT FOR WEKA

Enrollment No: 101240

Name of Student: Shivantika Thakur

Supervisor's Name: Dr. Sakshi Babbar

May-2014

Submitted in partial fulfillment of the Degree of Bachelor of Technology

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING & INFORMATION TECHNOLOGY

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

WAKNAGHAT, SOLAN (H.P)

Table of Contents

CERTIFICATE

This is to certify that the work titled “**OUTLIER DETECTION ALGORITHM SUIT FOR WEKA”**submitted by **SHIVANTIKA THAKUR** in partial fulfillment for the award of degree of B.Tech Computer Science Engineering of Jaypee University of Information Technology, Waknaghat has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

____

(Signature of Supervisor)

Name of Supervisor: Dr. Sakshi Babbar

Designation: Assistant Professor (senior grade), Dept. of CSE

Date:

ACKNOWLEDGEMENT

I take this opportunity to express my profound gratitude and deep regards to Dr. Sakshi Babbar , my Project Guide, for guiding and correcting me at every step of my work with attention and care. He has taken pain to go through the project and make necessary correction as and when needed. Thanks and appreciation to the helpful people at college for their support. I would also thank my university and my faculty members without whom this project would have been a distant reality. I also extend my heartfelt thanks to my family and friends for their undaunted support and faith in me.

Signature of the Student……………………………….

Name of the Student – Shivantika Thakur

Date -

ABSTRACT

WEKA (Waikato Environment for Knowledge Analysis) is open source which is a collection of machine learning algorithms for data mining tasks. WEKA contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. The workbench also provides a graphical user interface for easy access to these functionalities. There are several advantages to this software that includes probability and usability. Many researchers in industry and academia including students use this software because of it support the standard data mining tasks with good usability .Also, one of the best features of this software is that, it is well suited for developing new machine learning schemes. Taking advantage of this feature, I extended this software to contain in the outlier detection module as well .I have developed set of outlier algorithms and incorporated them under a new module in the weka software.

Currently the weka software includes different interfaces like Simple Command Line interface, Explorer, Experimenter and knowledge flow. Explorer is the main interface where the same functionality can also be obtained through the command line or knowledge flow. Experimenter is used for comparisons on different types of panels which provide access to main component of this workbench.

1. Preprocess panel is used to take the input data and analyze the data .Also we can preprocess the data by using certain filtering algorithm thereby processing the data according to the requirements.
2. Classify panel consists of different classification algorithms. This panel provides a facility to apply the classification algorithms to the given dataset, estimate the accuracy of the resulting predictive model, to generate ROC curves and also to visualize.
3. Associate panel consists of association rule mining algorithms. This panel helps in identifying the interesting rules from the given dataset.
4. Cluster panel consists of clustering algorithms that helps in cluster the given dataset.
5. Select attributes panel provide algorithms to identify the most predictive attributes in the given dataset.
6. Visualize panel helps in providing a scatter plot matrix which consists of individual scatter plots for the given dataset.

As my contribution, I have created a new panel that consists of most popular outlier detection algorithms. This panel enables users to apply outlier detection algorithms to the given dataset.

List of Figures

CERTIFICATE

This is to certify that the work titled "**OUTLIER DETECTION ALGORITHM SUIT FOR WEKA**"submitted by **SHIVANTIKA THAKUR** in partial fulfillment for the award of degree of B.Tech Computer Science Engineering of Jaypee University of Information Technology, Waknaghat has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Sakshi
25/5/14.

(Signature of Supervisor)

Name of Supervisor: Dr. Sakshi Babbar

Designation: Assistant Professor (senior grade), Dept. of CSE

Date: 25 May 2014

ACKNOWLEDGEMENT

I take this opportunity to express my profound gratitude and deep regards to Dr. Sakshi Babbar , my Project Guide, for guiding and correcting me at every step of my work with attention and care. He has taken pain to go through the project and make necessary correction as and when needed. Thanks and appreciation to the helpful people at college for their support. I would also thank my university and my faculty members without whom this project would have been a distant reality. I also extend my heartfelt thanks to my family and friends for their undaunted support and faith in me.

Signature of the Student.. Shivantika Thakur

Name of the Student – Shivantika Thakur

Date - 24 May 2014.

Chapter 1: INTRODUCTION

Chapter 1: INTRODUCTION

1.1 Introducing Outliers

Anomalies/ outliers are patterns in data that do not conform to a well defined notion of normal behaviour. Outlier detection refers to the task of finding anomalous patterns in given data according to a particular definition of anomalous behaviour. An outlier will refer to these anomalous patterns in the data. An outlier detection technique is a specific solution to an outlier detection problem. A normal pattern refers to a pattern in the data which is not an outlier. The output of an outlier detection technique could be labelled patterns (outlier or normal). Some of the outlier detection techniques also assign a score to a pattern based on the degree to which the pattern is considered an outlier. Such a score is referred to as outlier Score [VipinKumar2009].

Anomaly detection finds extensive use in a wide variety of applications such as fraud detection for credit cards, insurance, or health care, intrusion detection for cyber-security, fault detection in safety critical systems, and military surveillance for enemy activities.
The importance of anomaly detection is due to the fact that anomalies in data translate
to significant, and often critical, actionable information in a wide variety of application domain. For example, an anomalous traffic pattern in a computer network could mean that a hacked computer is sending out sensitive data to an unauthorized destination. An anomalous MRI image may indicate the presence of malignant tumours. Anomalies in credit card transaction data could indicate credit card or identity theft, or anomalous readings from a space craft sensor could signify a fault in some component of the space craft.

Detecting outliers or anomalies in data has been studied in the statistics community as early as the 19th century [VipinKumar2009]. Over time, a variety of anomaly detection techniques have been developed in several research communities. Many of these techniques have been specifically developed for certain application domains out of which distance and density based techniques are very popular because of its advantages, while others are more generic.

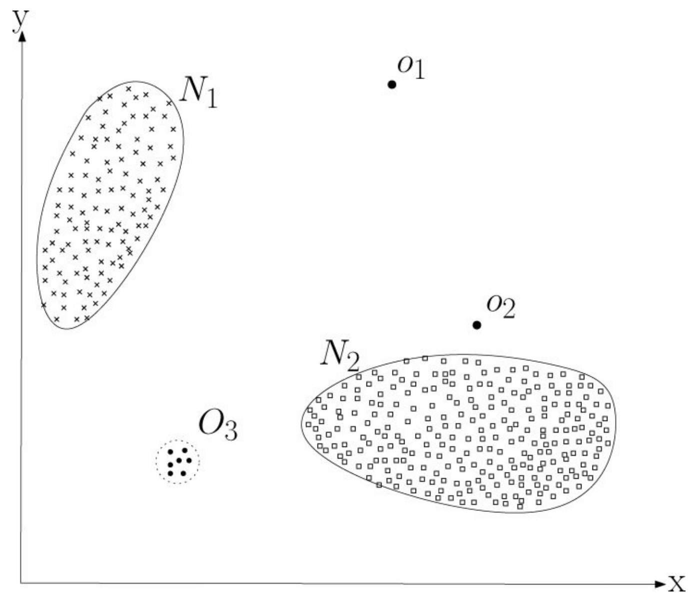


Fig.1. A simple example of anomalies in a two-dimensional data set.

1.2 Problem Statement

Development of outlier detection algorithm suit and incorporating it in a new module in the WEKA software.

Chapter 2: BACKGROUND MATERIAL

1.1 Background

Outlier detection has been a very important concept in the realm of data analysis. Recently, several application domains have realized the direct mapping between outliers in data and real world anomalies that are of great interest to an analyst. Outlier detection has been researched within various application domains and knowledge disciplines. [VipinKumar2009]

2.2 Challenges

A key challenge in outlier detection is that it involves exploring the unseen space. As mentioned earlier, at an abstract level, an outlier can be defined as a pattern that does not conform to expected normal behavior. A straightforward approach will be to define a region representing normal behaviour and declare any observation in the data which does not belong to this normal region as an outlier. But several factors make this apparently simple approach very challenging.

1. Defining a normal region which encompasses every possible normal behaviour is very difficult.
2. Often times normal behaviour keeps evolving and an existing notion of normal behaviour might not be sufficiently representative in the future.
3. The boundary between normal and outlying behavior is often fuzzy. Thus an outlying observation which lies close to the boundary can be actually normal and vice versa.
4. The exact notion of an outlier is different for different application domains. Every application domain imposes a set of requirements and constraints giving rise to a specific problem formulation for outlier detection.
5. Availability of labelled data for training/validation is often a major issue while developing an outlier detection technique.
6. In several cases in which outliers are the result of malicious actions, the malicious adversaries adapt themselves to make the outlying observations appear like normal, thereby making the task of defining normal behaviour more difficult.
7. Often the data contains noise which is similar to the actual outliers and hence is difficult to distinguish and remove.

2.3 Types of Anomalies/Outliers

An important aspect of an anomaly detection technique is the nature of the desired anomaly. Anomalies can be classified into following three categories:

2.3.1. Point Anomalies

If an individual data instance can be considered as anomalous with respect to the rest of data, then the instance is termed a point anomaly. This is the simplest type of anomaly and is the focus of majority of research on anomaly detection.
For example, in Figure 1, points o1 and o2, as well as points in region O3, lie outside the boundary of the normal regions, and hence are point anomalies since they are different from normal data points.

2.3.2. Contextual Anomalies

If a data instance is anomalous in a specific context, but not otherwise, then it is termed a contextual anomaly (also referred to as conditional anomaly [Song et al. 2007]). Each data instance is defined using the following two sets of attributes:

(1) Contextual attributes. The contextual attributes are used to determine the context (or neighbourhood) for that instance. For example, in spatial data sets, the longitude and latitude of a location are the contextual attributes. In time-series data, time is a contextual attribute that determines the position of an instance on the entire sequence.

(2) Behavioural attributes. The behavioral attributes define the noncontextual characteristics of an instance. For example, in a spatial data set describing the average rainfall of the entire world, the amount of rainfall at any location is a behavioural attribute. The anomalous behaviour is determined using the values for the behavioural attributes within a specific context. A data instance might be a contextual anomaly in a given context, but an identical data instance (in terms of behavioural attributes) could be considered normal in a different context. This property is key in identifying contextual and behavioural attributes for a contextual anomaly detection technique.

Contextual anomalies have been most commonly explored in time-series data [Weygand et al. 1995; Salvador and Chan 2003] and spatial data [Kou et al. 2006; mShekhar et al. 2001]. Figure 2 shows one such example for a temperature time-series that shows the monthly temperature of an area over the last few years. A temperature of 35◦F might be normal during the winter (at time t1) at that place, but the same value during the summer (at time t2) would be an anomaly. The choice of applying a contextual anomaly detection technique is determined by the meaningfulness of the contextual anomalies in the target application domain. Another key factor is the availability of contextual attributes. In several cases defining a context is straightforward, and hence applying a contextual anomaly detection technique makes sense. In other cases, defining a context is not easy, making it difficult to apply such techniques.

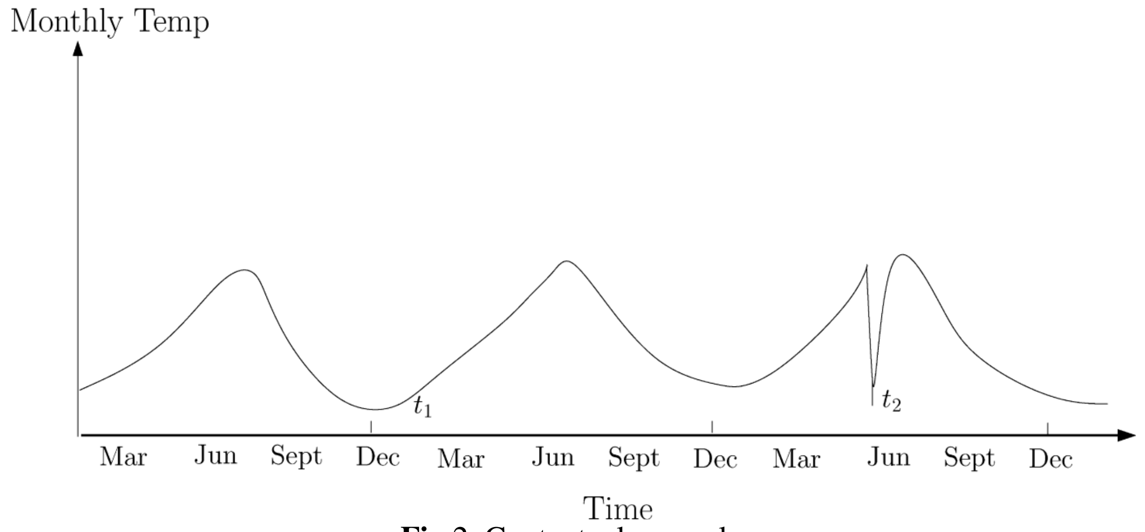


Fig 2. Contextual anomaly

Contextual anomaly *t*2 in a temperature time-series. Note that the temperature at time *t*1 is same as that at time *t*2 but occurs in a different context and hence is not considered as an anomaly.

2.3.3. Collective Anomalies

If a collection of related data instances is anomalous with respect to the entire data set, it is termed a collective anomaly. The individual data instances in a collective anomaly may not be anomalies by themselves, but their occurrence together as a collection is anomalous. Figure 3 is an example that shows a human electrocardiogram output [Goldberger et al. 2000]. The highlighted region denotes an anomaly because the same low value exists for an abnormally long time (corresponding to an Atrial Premature Contraction). Note that that low value by itself is not an anomaly.

It should be noted that this collection of events is an anomaly, but the individual events are not anomalies when they occur in other locations in the sequence. Collective anomalies have been explored for sequence data [Forrest et al. 1999; Sun et al. 2006], graph data [Noble and Cook 2003], and spatial data [Shekhar et al. 2001]. It should be noted that while point anomalies can occur in any data set, collective anomalies can occur only in data sets in which data instances are related.

In contrast, occurrence of contextual anomalies depends on the availability of context attributes in the data. A point anomaly or a collective anomaly can also be a contextual anomaly if analyzed with respect to a context. Thus a point anomaly detection problem or collective anomaly detection problem can be transformed to a contextual anomaly detection problem by incorporating the context information. The techniques used for detecting collective anomalies are very different than the point and contextual anomaly detection techniques, and require a separate detailed discussion. Hence I chose to not cover them in this project.

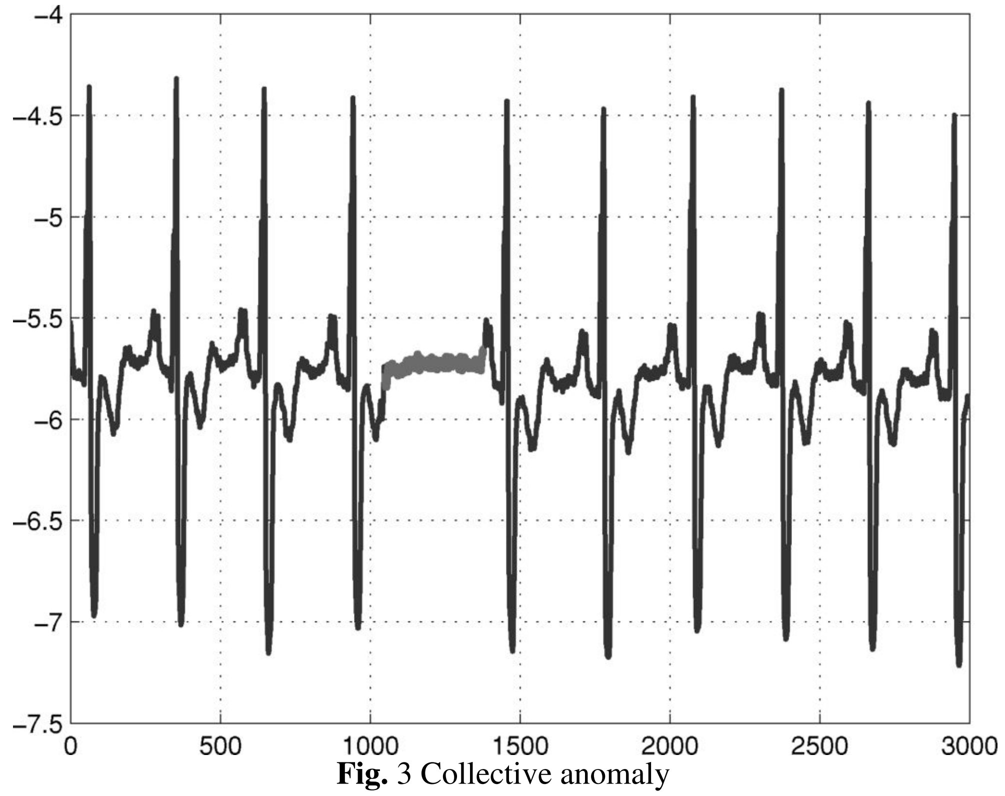


Fig. 3 Collective anomaly

Collective anomaly corresponding to an *Atrial Premature Contraction* in a human electrocardiogram output.

2.4 Approaches for outlier detection

2.4.1 Classification based Anomaly Detection Technique

Classification [Tan et al. 2005; Duda et al. 2000] is used to learn a model (classifier) from a set of labelled data instances (training) and then, classify a test instance into one of the classes using the learned model (testing). Classification-based anomaly detection techniques operate in a similar two-phase fashion. The training phase learns a classifier using the available labelled training data. The testing phase classifies a test instance as normal or anomalous, using the classifier.

2.4.2 Nearest Neighbour-Based Anomaly Detection Technique

The concept of nearest neighbor analysis has been used in several anomaly detection techniques. Nearest neighbor-based anomaly detection techniques require a distance or similarity measure defined between two data instances. Distance (or similarity) between two data instances can be computed in different ways. For continuous attributes, Euclidean distance is a popular choice, but other measures can be used [Tan et al. 2005, Chapter2]. For categorical attributes, a simple matching coefficient is often used but more complex distance measures can also be used [Boriah et al. 2008; Chandola et al. 2008]. For multivariate data instances, distance or similarity is usually computed for each attribute and then combined [Tan et al. 2005, Chapter 2]. My project is based on this technique.

2.4.3 Clustering-Based Anomaly Detection Technique

Clustering [Jain and Dubes 1988; Tan et al. 2005] is used to group similar data instances into clusters. Clustering is primarily an unsupervised technique though semi supervised clustering [Basu et al. 2004] has also been explored lately. Even though clustering and anomaly detection appear to be fundamentally different from each other, several clustering-based anomaly detection techniques have been developed.

2.4.4 Statistical Anomaly Detection Technique

The underlying principle of any statistical anomaly detection technique is: “An anomaly is an observation which is suspected of being partially or wholly irrelevant because it is not generated by the stochastic model assumed” [Anscombe and Guttman 1960].

2.4.5 Nonparametric Techniques

The anomaly detection techniques in this category use nonparametric statistical models, such that the model structure is not defined a priory, but is instead determined from given data. Such techniques typically make fewer assumptions regarding the data, such as smoothness of density, when compared to parametric techniques.

2.4.6 Information Theoretic Anomaly Detection Techniques

Information theoretic techniques analyze the information content of data set using different information theoretic measures such as Kolomogorov Complexity, entropy, relative entropy, and so on.

2.4.7 Special Anomaly Detection Technique

Spectral techniques try to find an approximation of the data using a combination of attributes that capture the bulk of the variability in the data.

2.5 Distance Based Approaches

The approach used in this project is distance based approach. A basic nearest neighbour anomaly detection technique is based on the following definition: *The anomaly score of a data instance is defined as its distance to its kth nearest neighbor in a given data set.*

This technique is based on the following key assumption:
Assumption: Normal data instances occur in dense neighbourhoods, while anomalies occur far from their closest neighbours
.
Using *K* nearest neighbor distance based approach the distance is calculated using one of the following measures:

•Euclidian Distance

•Mahalanobis Distance

What is *K* -nearest neighbour approach?

Briefly, if m of the k nearest neighbors (where m < k) lie within a specific distance threshold d then the exemplar is deemed to lie in a sufficiently dense region of the data distribution to be classified as normal. However, if there are less than m neighbors inside the distance threshold then the exemplar is an outlier.

Nearest neighbor-based anomaly detection techniques can be broadly grouped into two categories:
(1) Techniques that use the distance of a data instance to its kth nearest neighbor as the anomaly score;
(2) Techniques that compute the relative density of each data instance to compute its anomaly score.

There are various flavours of k-Nearest Neighbour (k-NN) algorithm for outlier detection but all calculate the nearest neighbours of a record using a suitable distance calculation Metric such as Euclidean distance or Mahalanobis distance.
Euclidean distance is given by equation 1 and is simply the vector distance

$$\text{Dist}(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

- (1)

Whereas the Mahalanobis distance given by equation 2

$$\text{Dist}(x, y) = \sqrt{(x - y)^T \Sigma^{-1} (x - y)}$$

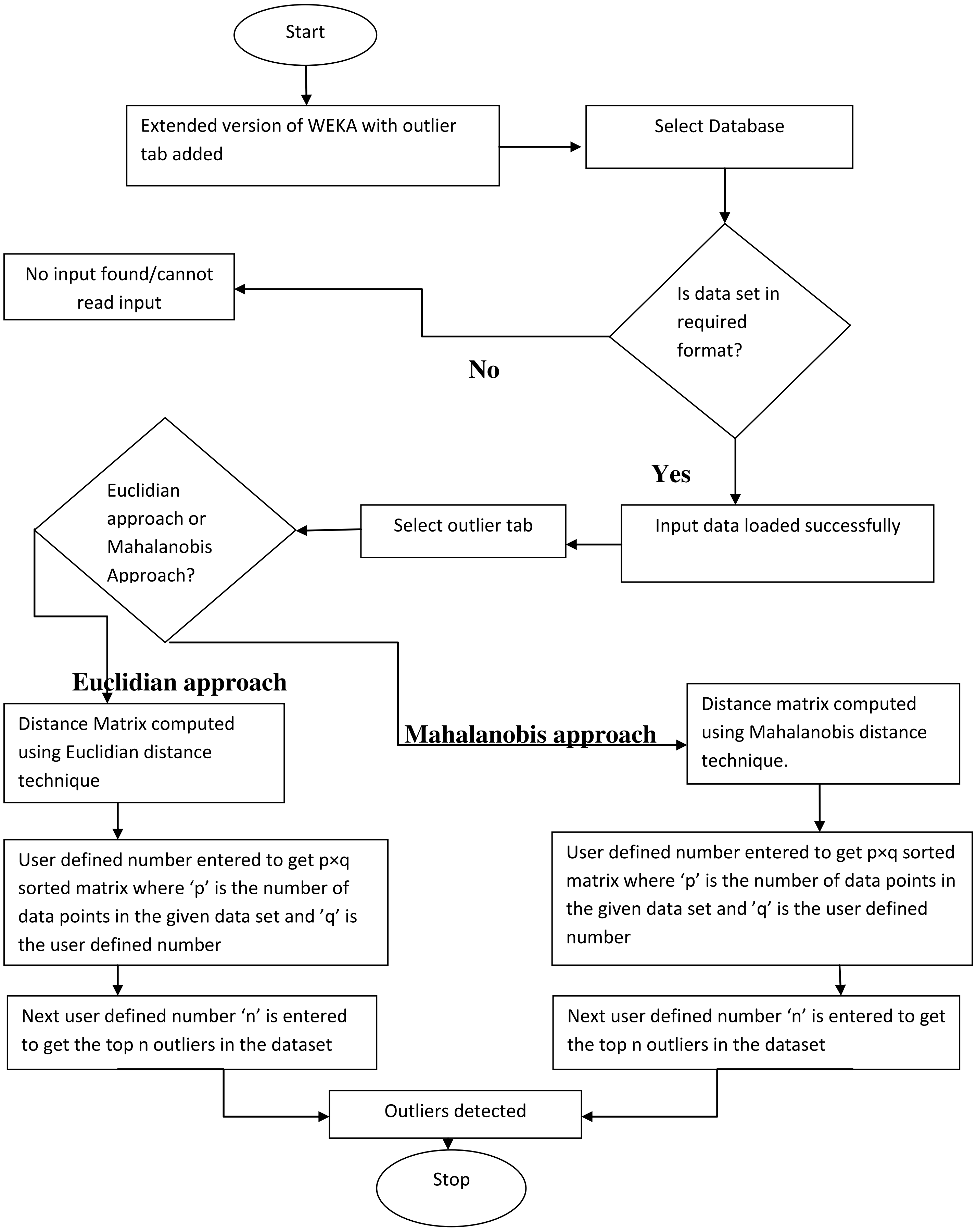
- (2)

Mahalanobis distance is computationally expensive to calculate for large high dimensional Data sets compared to the Euclidean distance as it requires a pass through the entire data set to identify the attribute correlations.

Chapter 3: DESIGN AND ANALYSIS

3.1 Design

Following is the flowchart representation of how the extended version of WEKA works.


Start
Extended version of WEKA with outlier tab added
Select Database
Is data set in required format?
No
No input found/cannot read input
Yes
Input data loaded successfully
Select outlier tab
Euclidian approach or Mahalanobis Approach?
Euclidian approach
Mahalanobis approach
Distance Matrix computed using Euclidian distance technique
Distance matrix computed using Mahalanobis distance technique.
User defined number entered to get p×q sorted matrix where 'p' is the number of data points in the given data set and 'q' is the user defined number
User defined number entered to get p×q sorted matrix where 'p' is the number of data points in the given data set and 'q' is the user defined number
Next user defined number 'n' is entered to get the top n outliers in the dataset
Next user defined number 'n' is entered to get the top n outliers in the dataset
Outliers detected
Stop

3.2 Implementation Details

3.2.1 Software

- All the programming code is written in Java using inbuilt functions.
- Net Beans as an integrated development environment
- WEKA version 3.6.8
- WEKA add on classes
- The project is supposed to run both on windows or Mac

3.2.2 Hardware

- No hardware implementations required.

Chapter 4: IMPLEMENTATION

4.1 Input phase

4.1.1 Inputting Data into WEKA

We need to input our dataset into WEKA. We start the process by clicking on "Open Url" while in the "Preprocess" tab of the Explorer (this is the tab that we are initially in when WEKA starts). We can then browse to the location of the data file on the PC (we can use "Open URL" if the dataset is on the web). WEKA ideally would like an .arff file, which contains a header that describes the variables and the data types of the variables, followed by the data itself. The format of the .arff file is available from the various WEKA manuals. It also accepts .csv format.

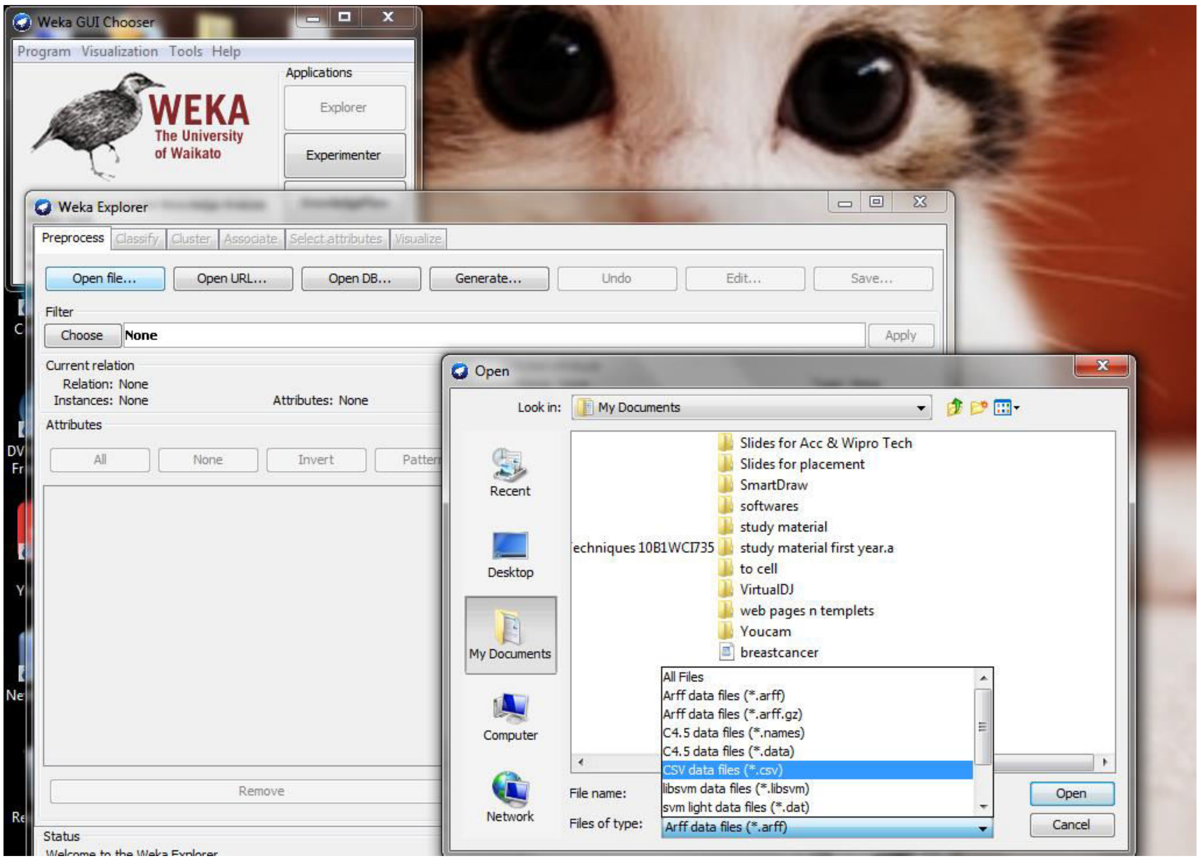
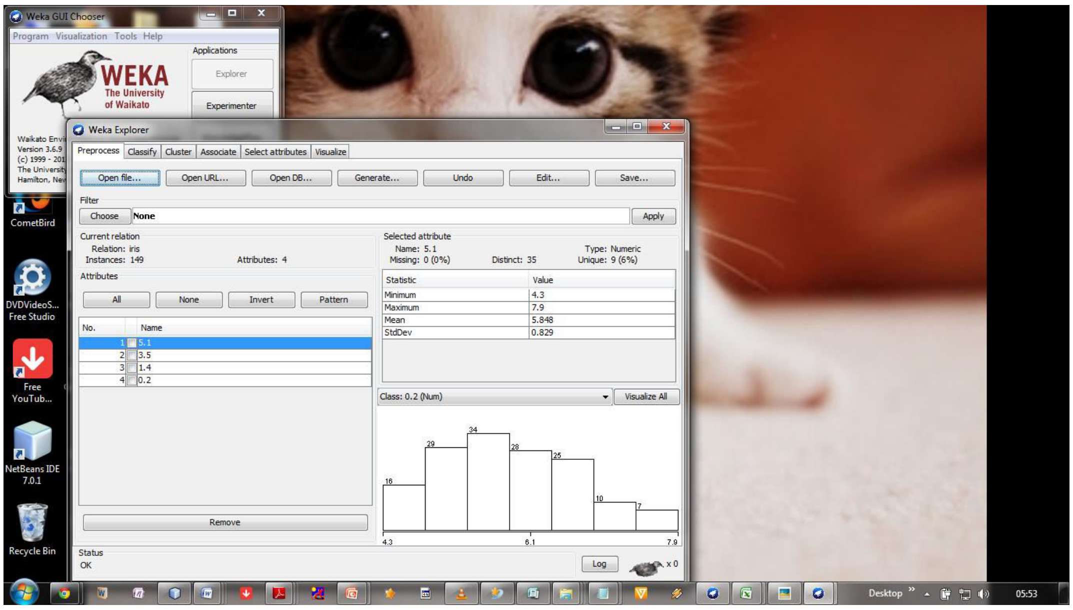


Fig 5: Inputting dataset into WEKA

4.2 WEKA without outlier panel

Once the input has been taken by the software, all the panels perform different action. The algorithms can either be applied directly to a dataset or called from our own Java code. WEKA contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization.



Filter: None

Current relation
Relation: iris
Instances: 149 Attributes: 4

Selected attribute
Name: 5.1 Type: Numeric
Missing: 0 (0%) Distinct: 35 Unique: 9 (6%)

No.	Name
1	5.1
2	3.5
3	1.4
4	0.2

Statistic	Value
Minimum	4.3
Maximum	7.9
Mean	5.848
StdDev	0.829

Class: 0.2 (Num)

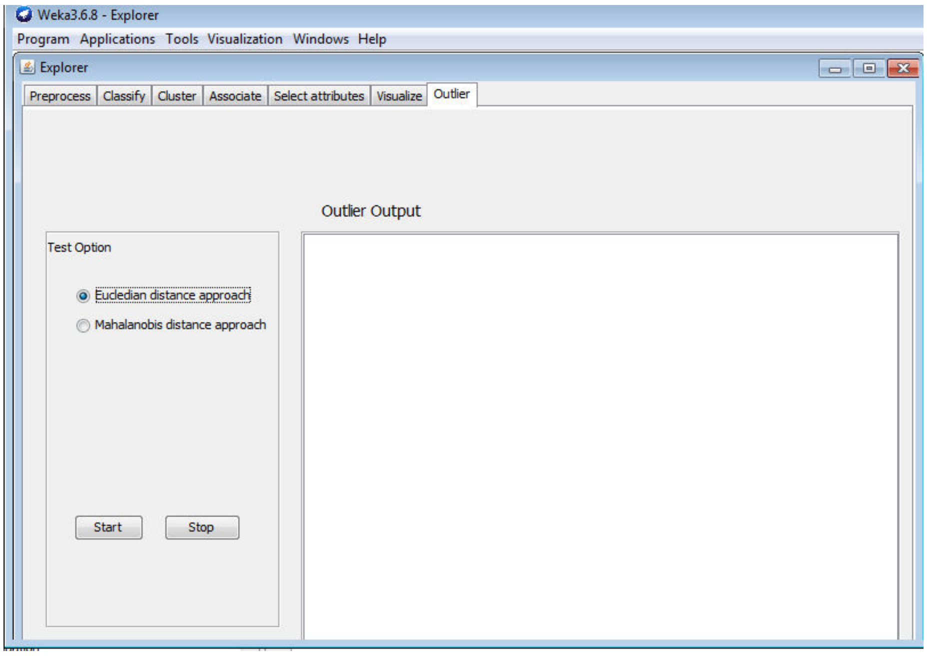
Status
OK

Fig 4: WEKA with Iris dataset as an input

4.3 WEKA with outlier panel added

Adding the outlier panel, though procedure of addition is shown in the code implementation part .The outlier panel contains outlier output for computational and final results.

Two radio buttons for Euclidian and Mahalanobis distance approach each .A start button to trigger the algorithms and stop to terminate the process are added in this new panel.



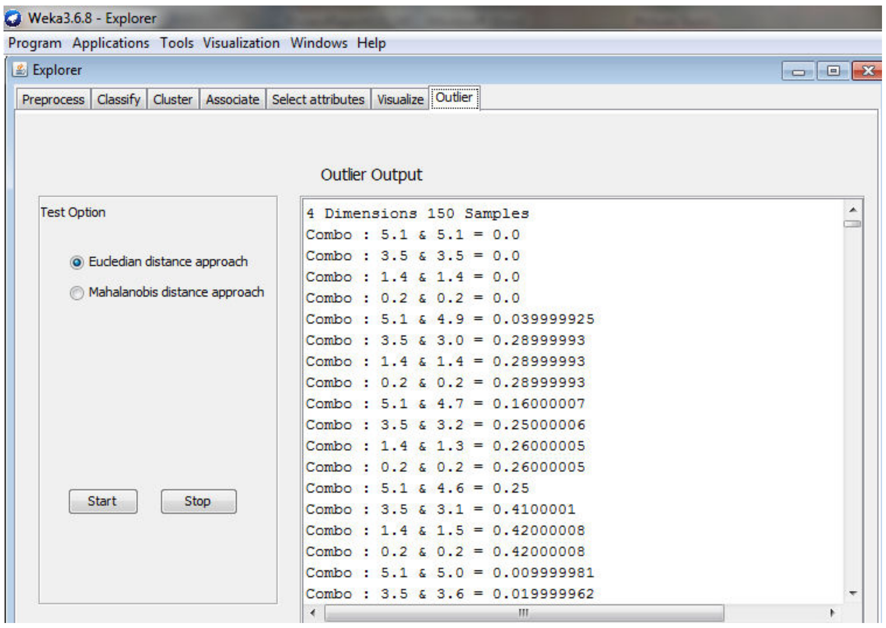
Test Option
[✓ Euclidean distance approach]
Mahalanobis distance approach

Fig 6: Outlier panel in WEKA

4.4 Operation carried out by Outlier panel

4.4.1 Using Euclidian distance approach

Dataset used for the operations is Iris dataset. The following figures show how Euclidian distance approach is carried out for detecting anomalies in the Iris dataset



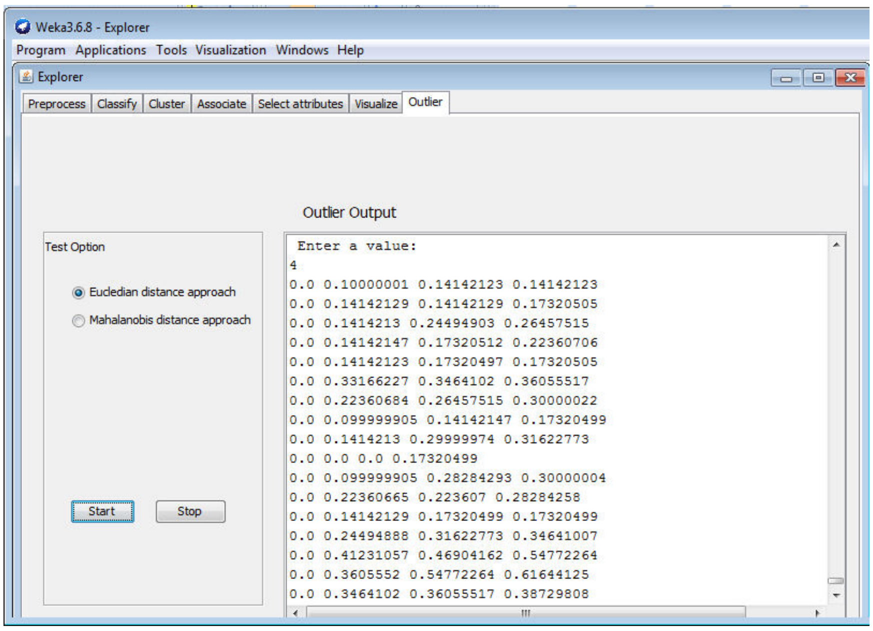
Test Option

[✓ Eucledian distance approach]
Mahalanobis distance approach

```
4 Dimensions 150 Samples
Combo : 5.1 & 5.1 = 0.0
Combo : 3.5 & 3.5 = 0.0
Combo : 1.4 & 1.4 = 0.0
Combo : 0.2 & 0.2 = 0.0
Combo : 5.1 & 4.9 = 0.039999925
Combo : 3.5 & 3.0 = 0.28999993
Combo : 1.4 & 1.4 = 0.28999993
Combo : 0.2 & 0.2 = 0.28999993
Combo : 5.1 & 4.7 = 0.16000007
Combo : 3.5 & 3.2 = 0.25000006
Combo : 1.4 & 1.3 = 0.26000005
Combo : 0.2 & 0.2 = 0.26000005
Combo : 5.1 & 4.6 = 0.25
Combo : 3.5 & 3.1 = 0.4100001
Combo : 1.4 & 1.5 = 0.42000008
Combo : 0.2 & 0.2 = 0.42000008
Combo : 5.1 & 5.0 = 0.009999981
Combo : 3.5 & 3.6 = 0.019999962
```

Fig 7: Computation of distance matrix (Euclidian Approach)

Here the algorithm is designed in such a way that it gets the dimensions of any data set chosen and calculates the samples present in it. Then it form the combinations such that distance of each sample from another is computed (here distance is calculated using Euclidian approach)

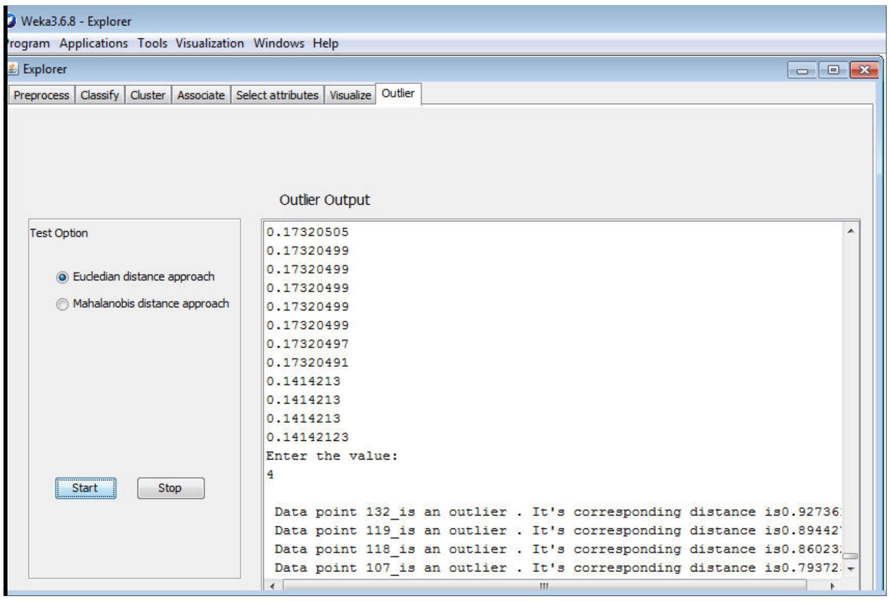

Weka3.6.8 - Explorer
Program Applications Tools Visualization Windows Help
Explorer
Preprocess Classify Cluster Associate Select attributes Visualize Outlier
Outlier Output
Test Option
Start
Stop

[✓ Eucledian distance approach] Mahalanobis distance approach

```
 Enter a value:
4
0.0 0.10000001 0.14142123 0.14142123
0.0 0.14142129 0.14142129 0.17320505
0.0 0.1414213 0.24494903 0.26457515
0.0 0.14142147 0.17320512 0.22360706
0.0 0.14142123 0.17320497 0.17320505
0.0 0.33166227 0.3464102 0.36055517
0.0 0.22360684 0.26457515 0.30000022
0.0 0.099999905 0.14142147 0.17320499
0.0 0.1414213 0.29999974 0.31622773
0.0 0.0 0.0 0.17320499
0.0 0.099999905 0.28284293 0.30000004
0.0 0.22360665 0.223607 0.28284258
0.0 0.14142129 0.17320499 0.17320499
0.0 0.24494888 0.31622773 0.34641007
0.0 0.41231057 0.46904162 0.54772264
0.0 0.3605552 0.54772264 0.61644125
0.0 0.3464102 0.36055517 0.38729808
```

Fig 8: User selected values from Sorted Distance matrix (Euclidian Approach)

The result of computations form a matrix of distances (distance matrix) such that the distance of sample from itself will be 0.0 and from some other sample it will be some value greater than zero. Whole matrix is then sorted in ascending order by the algorithm and then a user defined number picks up an array which is then again sorted in descending order during run time.


Weka3.6.8 - Explorer
rogram Applications Tools Visualization Windows Help
Explorer
Preprocess Classify Cluster Associate Select attributes Visualize Outlier
Outlier Output
Test Option
Start
Stop

[✓ Eucledian distance approach] Mahalanobis distance approach

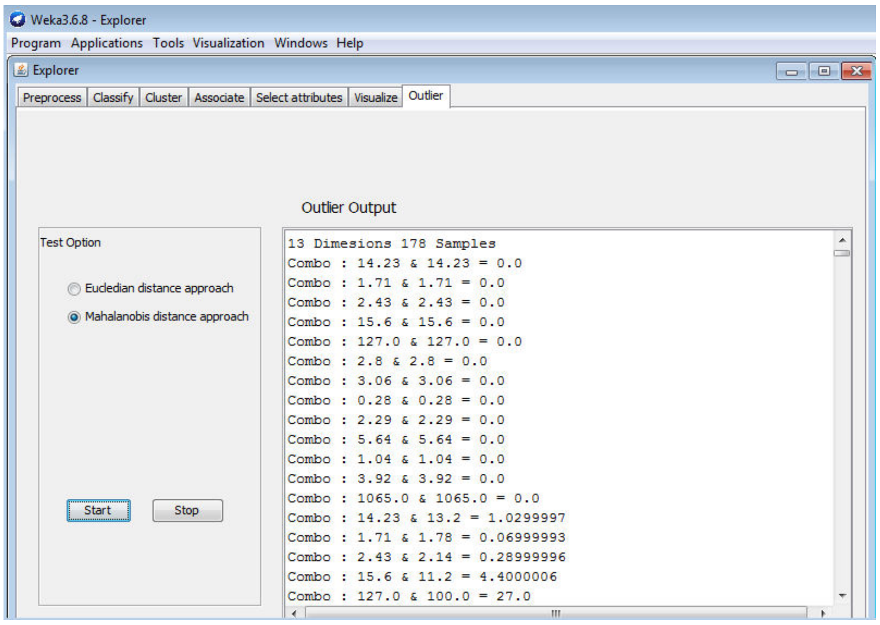
```
0.17320505
0.17320499
0.17320499
0.17320499
0.17320499
0.17320499
0.17320497
0.17320491
0.1414213
0.1414213
0.1414213
0.14142123
Enter the value:
4

 Data point 132_is an outlier . It's corresponding distance is0.92736
 Data point 119_is an outlier . It's corresponding distance is0.89442
 Data point 118_is an outlier . It's corresponding distance is0.86023
 Data point 107_is an outlier . It's corresponding distance is0.79372
```

Fig 9: Output (Euclidian Approach)

Another user defined number can then fetch the result from the sorted array derived from the ordered distance matrix.

4.4.2 Using Mahalanobis distance approach

Similar steps are followed but only the algorithms running in the backend differ. Here distance is computed by Mahalanobis Distance formula. Taking chemical analysis of wine data set as an example.

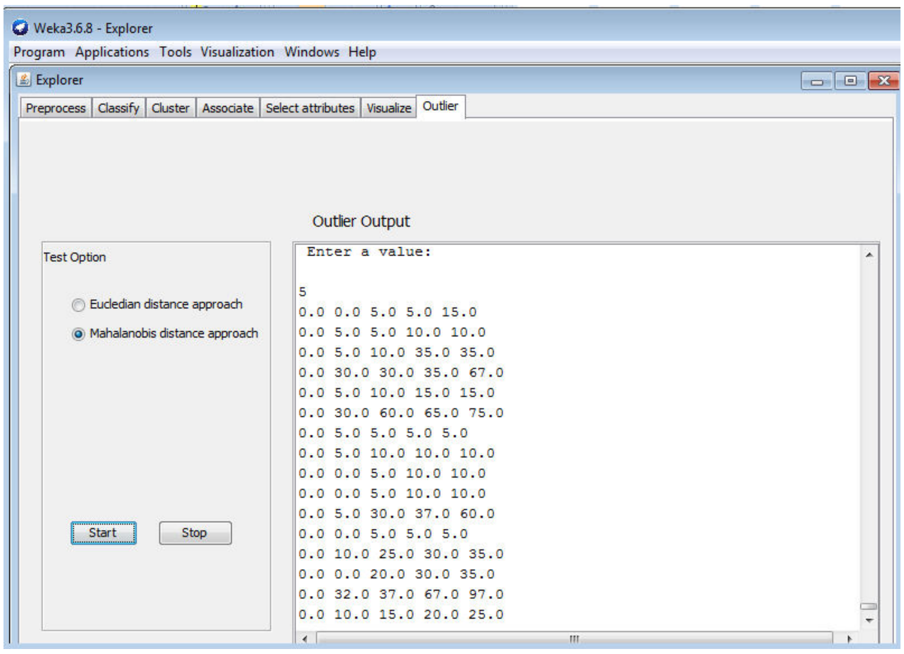

Weka3.6.8 - Explorer
Program Applications Tools Visualization Windows Help
Explorer
Preprocess Classify Cluster Associate Select attributes Visualize Outlier
Outlier Output
Start
Stop

Test Option

( ) Eucledian distance approach
[✓ Mahalanobis distance approach]

```
13 Dimesions 178 Samples
Combo : 14.23 & 14.23 = 0.0
Combo : 1.71 & 1.71 = 0.0
Combo : 2.43 & 2.43 = 0.0
Combo : 15.6 & 15.6 = 0.0
Combo : 127.0 & 127.0 = 0.0
Combo : 2.8 & 2.8 = 0.0
Combo : 3.06 & 3.06 = 0.0
Combo : 0.28 & 0.28 = 0.0
Combo : 2.29 & 2.29 = 0.0
Combo : 5.64 & 5.64 = 0.0
Combo : 1.04 & 1.04 = 0.0
Combo : 3.92 & 3.92 = 0.0
Combo : 1065.0 & 1065.0 = 0.0
Combo : 14.23 & 13.2 = 1.0299997
Combo : 1.71 & 1.78 = 0.06999993
Combo : 2.43 & 2.14 = 0.28999996
Combo : 15.6 & 11.2 = 4.4000006
Combo : 127.0 & 100.0 = 27.0
```

Fig 10: Computation of distance matrix (Mahalanobis approach)

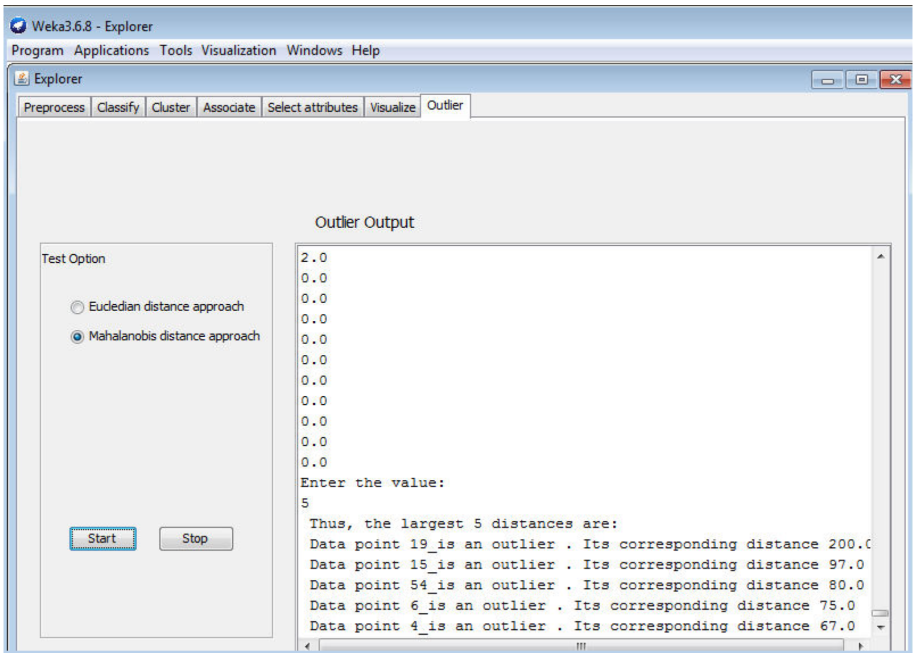

Weka3.6.8 - Explorer
Program Applications Tools Visualization Windows Help
Explorer
Preprocess Classify Cluster Associate Select attributes Visualize Outlier
Outlier Output
Start
Stop

Test Option

( ) Eucledian distance approach
[✓ Mahalanobis distance approach]

```
 Enter a value:

5
0.0 0.0 5.0 5.0 15.0
0.0 5.0 5.0 10.0 10.0
0.0 5.0 10.0 35.0 35.0
0.0 30.0 30.0 35.0 67.0
0.0 5.0 10.0 15.0 15.0
0.0 30.0 60.0 65.0 75.0
0.0 5.0 5.0 5.0 5.0
0.0 5.0 10.0 10.0 10.0
0.0 0.0 5.0 10.0 10.0
0.0 0.0 5.0 10.0 10.0
0.0 5.0 30.0 37.0 60.0
0.0 0.0 5.0 5.0 5.0
0.0 10.0 25.0 30.0 35.0
0.0 0.0 20.0 30.0 35.0
0.0 32.0 37.0 67.0 97.0
0.0 10.0 15.0 20.0 25.0
```

Fig 11: User selected values from Sorted Distance matrix (Mahalanobis approach)


Weka3.6.8 - Explorer
Program Applications Tools Visualization Windows Help
Explorer
Preprocess Classify Cluster Associate Select attributes Visualize Outlier
Outlier Output
Start
Stop

Test Option

( ) Eucledian distance approach
(●) Mahalanobis distance approach

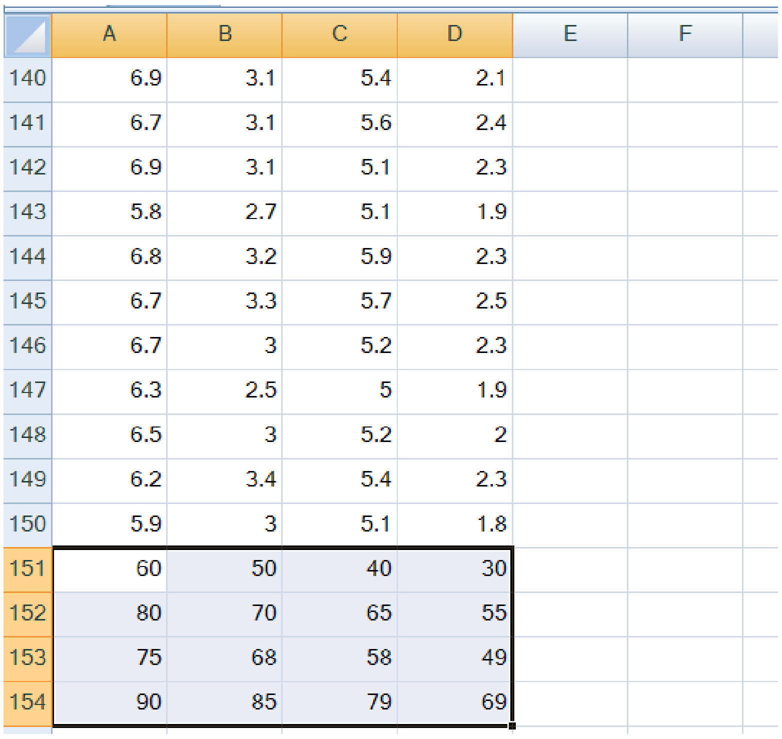
```
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
Enter the value:
5
 Thus, the largest 5 distances are:
 Data point 19_is an outlier . Its corresponding distance 200.
 Data point 15_is an outlier . Its corresponding distance 97.0
 Data point 54_is an outlier . Its corresponding distance 80.0
 Data point 6_is an outlier . Its corresponding distance 75.0
 Data point 4_is an outlier . Its corresponding distance 67.0
```

Fig 12: Output (Mahalanobis approach)

The complexity of both the algorithms is quadratic i.e. O(n^2).

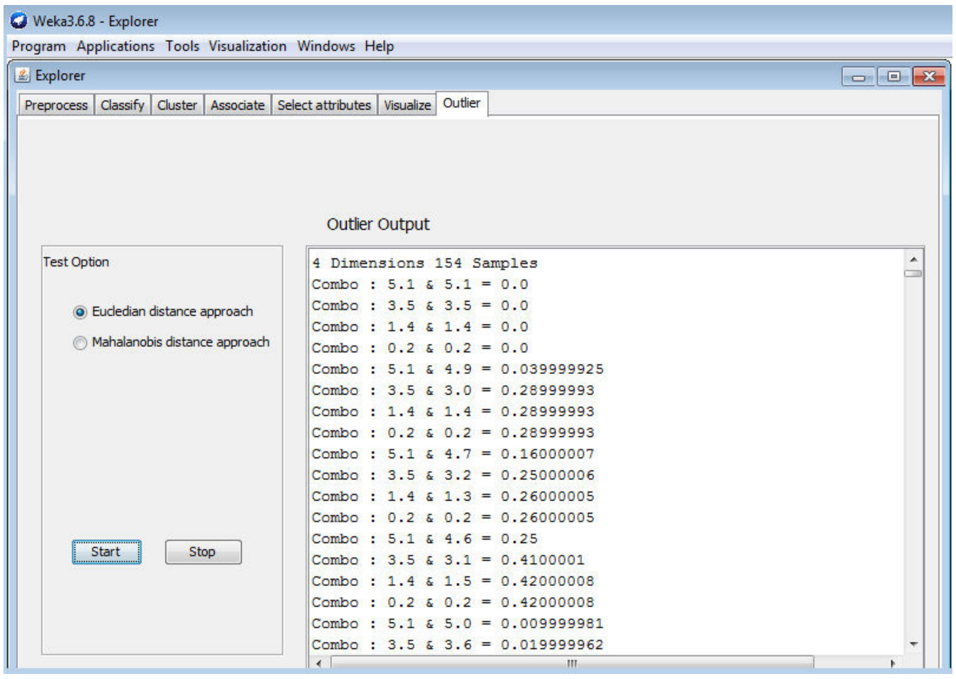
4.5Evaluation

To evaluate four outliers are deliberately added to iris data set**.**



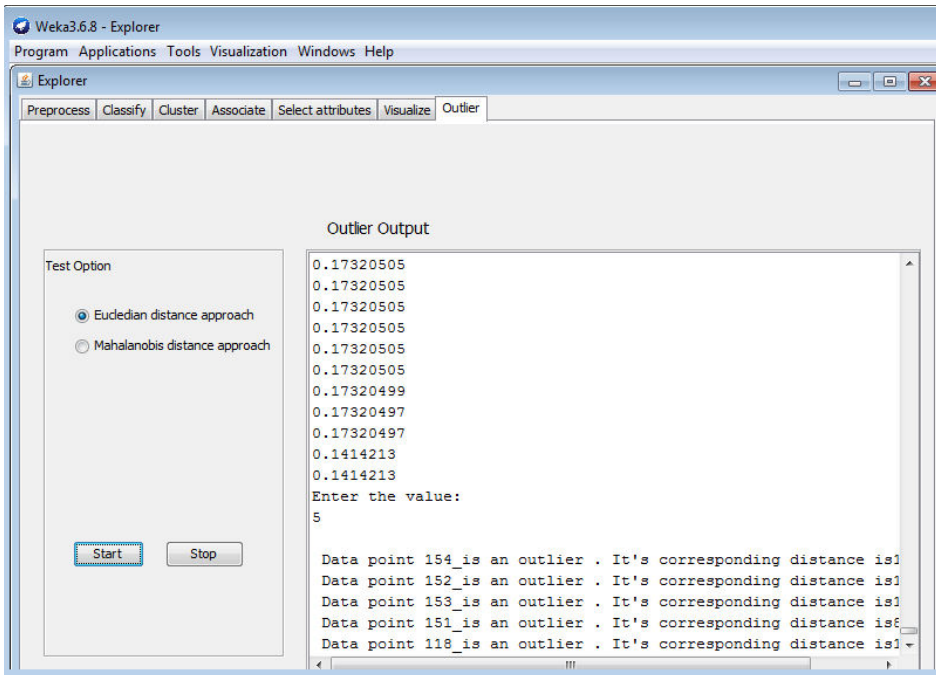
	A	B	C	D	E	F
140	6.9	3.1	5.4	2.1		
141	6.7	3.1	5.6	2.4		
142	6.9	3.1	5.1	2.3		
143	5.8	2.7	5.1	1.9		
144	6.8	3.2	5.9	2.3		
145	6.7	3.3	5.7	2.5		
146	6.7	3	5.2	2.3		
147	6.3	2.5	5	1.9		
148	6.5	3	5.2	2		
149	6.2	3.4	5.4	2.3		
150	5.9	3	5.1	1.8		
151	60	50	40	30		
152	80	70	65	55		
153	75	68	58	49		
154	90	85	79	69		

Fig 13: Adding Outliers in iris data set


Weka3.6.8 - Explorer
Program Applications Tools Visualization Windows Help
Explorer
Preprocess Classify Cluster Associate Select attributes Visualize Outlier
Outlier Output
Test Option
Eucledian distance approach
Mahalanobis distance approach
Start
Stop

```
4 Dimensions 154 Samples
Combo : 5.1 & 5.1 = 0.0
Combo : 3.5 & 3.5 = 0.0
Combo : 1.4 & 1.4 = 0.0
Combo : 0.2 & 0.2 = 0.0
Combo : 5.1 & 4.9 = 0.039999925
Combo : 3.5 & 3.0 = 0.28999993
Combo : 1.4 & 1.4 = 0.28999993
Combo : 0.2 & 0.2 = 0.28999993
Combo : 5.1 & 4.7 = 0.16000007
Combo : 3.5 & 3.2 = 0.25000006
Combo : 1.4 & 1.3 = 0.26000005
Combo : 0.2 & 0.2 = 0.26000005
Combo : 5.1 & 4.6 = 0.25
Combo : 3.5 & 3.1 = 0.4100001
Combo : 1.4 & 1.5 = 0.42000008
Combo : 0.2 & 0.2 = 0.42000008
Combo : 5.1 & 5.0 = 0.009999981
Combo : 3.5 & 3.6 = 0.019999962
```

Fig 14: Data set with outliers


Weka3.6.8 - Explorer
Program Applications Tools Visualization Windows Help
Explorer
Preprocess Classify Cluster Associate Select attributes Visualize Outlier
Outlier Output
Test Option
Eucledian distance approach
Mahalanobis distance approach
Start
Stop

```
0.17320505
0.17320505
0.17320505
0.17320505
0.17320505
0.17320505
0.17320499
0.17320497
0.17320497
0.1414213
0.1414213
Enter the value:
5

 Data point 154_is an outlier . It's corresponding distance is1
 Data point 152_is an outlier . It's corresponding distance is1
 Data point 153_is an outlier . It's corresponding distance is1
 Data point 151_is an outlier . It's corresponding distance is8
 Data point 118_is an outlier . It's corresponding distance is1
```

Fig 15: Evaluation output

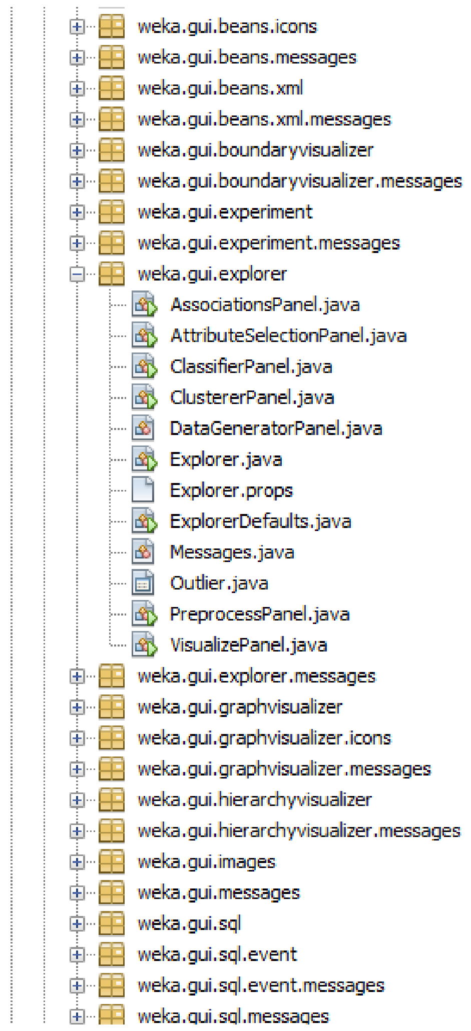
Results clearly state out those four specific data points as outliers.

.

Chapter5: CODE IMPLEMENTATION

5.1 Project code

5.1.1Adding new tab in explorer


weka.gui.beans.icons
weka.gui.beans.messages
weka.gui.beans.xml
weka.gui.beans.xml.messages
weka.gui.boundaryvisualizer
weka.gui.boundaryvisualizer.messages
weka.gui.experiment
weka.gui.experiment.messages
weka.gui.explorer
AssociationsPanel.java
AttributeSelectionPanel.java
ClassifierPanel.java
ClustererPanel.java
DataGeneratorPanel.java
Explorer.java
Explorer.props
ExplorerDefaults.java
Messages.java
Outlier.java
PreprocessPanel.java
VisualizePanel.java
weka.gui.explorer.messages
weka.gui.graphvisualizer
weka.gui.graphvisualizer.icons
weka.gui.graphvisualizer.messages
weka.gui.hierarchyvisualizer
weka.gui.hierarchyvisualizer.messages
weka.gui.images
weka.gui.messages
weka.gui.sql
weka.gui.sql.event
weka.gui.sql.event.messages
weka.gui.sql.messages

For adding outlier panel in WEKA, a java file has to be added in the package weka.gui.explorer in the source package.

Source code for outlier .java is

package weka.gui.explorer; import dmm.Dmm;

import dmm.Mahalanobis;

import dmm.dmm_new;

import java.io.;*

import java.util.Arrays;

import java.util.Scanner;

import weka.gui.explorer.;*

```
import javax.swing.ButtonGroup;

import javax.swing.JFileChooser;

import javax.swing.JOptionPane;

import javax.swing.JTextArea;

import weka.core.Instances;

import static weka.gui.explorer.ClustererPanel.MODEL_FILE_EXTENSION;

 public class Outlier extends javax.swing.JPanel implements Explorer.ExplorerPanel {
ButtonGroup bg;

          File fl;

          String pos;

          JFileChooser jc = new JFileChooser();

    public Outlier() {

        initComponents();

  bg = new ButtonGroup();

        bg.add(jRadioButton1);

     bg.add(jRadioButton2);

    }

       @SuppressWarnings("unchecked")

    // <editor-fold defaultstate="collapsed" desc="Generated Code">

    private void initComponents() {

buttonGroup1 = new javax.swing.ButtonGroup();

        jPanel1 = new javax.swing.JPanel();

        jScrollPane1 = new javax.swing.JScrollPane();

        jTextArea1 = new javax.swing.JTextArea();

        jLabel1 = new javax.swing.JLabel();

        jPanel2 = new javax.swing.JPanel();

        jLabel2 = new javax.swing.JLabel();
        jRadioButton1 = new javax.swing.JRadioButton();
        jRadioButton2 = new javax.swing.JRadioButton();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
    jPanel1.setBorder(javax.swing.BorderFactory.createEtchedBorder());
    jTextArea1.setColumns(20);
        jTextArea1.setRows(5);
        jScrollPane1.setViewportView(jTextArea1);
     javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jScrollPane1, javax.swing.GroupLayout.Alignment.TRAILING)
        );
        jPanel1Layout.setVerticalGroup(
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 363,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(0, 0, Short.MAX_VALUE))
        );
   jLabel1.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
        jLabel1.setText("Outlier Output");
     jPanel2.setBorder(javax.swing.BorderFactory.createEtchedBorder());
   jLabel2.setText("Test Option");
   jRadioButton1.setText("Eucledian distance approach");


        jRadioButton2.setText("Mahalanobis distance approach");
        jButton1.setText("Start");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });
        jButton2.setText("Stop");
        javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 101,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(0, 0, Short.MAX_VALUE))
            .addGroup(jPanel2Layout.createSequentialGroup()
.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
                    .addGroup(jPanel2Layout.createSequentialGroup()
                        .addGap(21, 21, 21)
.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG, false)
                            .addComponent(jRadioButton2,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                            .addComponent(jRadioButton1,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)))
    .addGroup(jPanel2Layout.createSequentialGroup()
                        .addGap(23, 23, 23)
```

```
                  .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 61,
javax.swing.GroupLayout.PREFERRED_SIZE)

                  .addGap(18, 18, 18)

                  .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 67,
javax.swing.GroupLayout.PREFERRED_SIZE)))

         .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

      );

      jPanel2Layout.setVerticalGroup(

         jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

         .addGroup(jPanel2Layout.createSequentialGroup()

            .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 25,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addGap(18, 18, 18)

            .addComponent(jRadioButton1)

            .addComponent(jRadioButton2)

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 155,
Short.MAX_VALUE)
.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASEL
INE)

               .addComponent(jButton1)

               .addComponent(jButton2))

            .addGap(75, 75, 75))

      );

      javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);

      this.setLayout(layout);

      layout.setHorizontalGroup(

         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

         .addGroup(layout.createSequentialGroup()

            .addGap(19, 19, 19)
```

```
                .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

                .addGap(18, 18, 18)

                .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

                .addContainerGap())

            .addGroup(layout.createSequentialGroup()

                .addGap(261, 261, 261)

                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 111,
javax.swing.GroupLayout.PREFERRED_SIZE)

                .addContainerGap(472, Short.MAX_VALUE))

        );

        layout.setVerticalGroup(

            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()

                .addContainerGap(53, Short.MAX_VALUE)

                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 25,
javax.swing.GroupLayout.PREFERRED_SIZE)

                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)


.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

                    .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

                    .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

                .addGap(26, 26, 26))

        );

    }// </editor-fold

   private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
```

```
  if(jRadioButton1.isSelected())
       {
          jTextArea1.setText(null);
   try
        {    {
          {
          JFileChooser jc = new JFileChooser();
          jc.showOpenDialog(null);
          fl=jc.getSelectedFile();
   BufferedReader br1 = new BufferedReader(new FileReader(fl));
          String line1;
          line1= br1.readLine();
int count=1,count2=1;
          for(int i=0;i<line1.length();i++)
          {
              if(line1.charAt(i)==',')
                  count++;
          }
          for(int i=0;;i++)
          {
              line1=br1.readLine();
              if(line1==null)
                  break;
              count2++;
          }
        jTextArea1.append(count+" Dimensions ");
          jTextArea1.append(count2+" Samples ");
```

```
BufferedReader br = new BufferedReader(new FileReader(fl));
String line;
float[][] col0 = new float[count2][count];
float[][] res = new float[count2][count2];
float shi[]=new float[count2];
for (int i = 0; i < count2; i++) {
   line = br.readLine();


      String[] cols = line.split(",");
      for(int j=0;j<count;j++)
      {
      col0[i][j] = Float.valueOf(cols[j]);
      }
}
float a;
for(int i=0;i<count2;i++)
{
      for(int j=0;j<count2;j++)
      {
            a=0;
            for(int v=0;v<count;v++)
            {
                   a+= ((col0[i][v]-col0[j][v])*(col0[i][v]-col0[j][v]));
                  String ares = ("Combo : "+col0[i][v]+ " & "+col0[j][v]+" = "+a);
                  jTextArea1.append("\n" +ares);
                              }
            res[i][j]=(float) Math.sqrt(a) ;
```

```
        }
}
jTextArea1.append("\n Enter a value:");
int val = Integer.parseInt(JOptionPane.showInputDialog("\n Enter a value\n"));
String pr = Integer.toString(val);
jTextArea1.append("\n" +pr+"\n");
for(int i=0;i<count2;i++)
Arrays.sort(res[i]);
int s=count2-1;
  String arr[] = new String[count2];
for(int i=0;i<count2;i++)
{
        for(int j=0;j<val;j++)
        {
              String sss =Float.toString(res[i][j]);
              jTextArea1.append(sss +" ");
               arr[i] = new String((i+1) + "_" );
        }
        jTextArea1.append("\n");
        }
         for (int i = 0; i < arr.length; i++) {
    }
for(int i=0;i<count2;i++)
{
shi[i]=res[i][val-1];
}
for(int i=0;i<s;i++)
```

```
          {
             for(int j=i+1;j<count2;j++)
             {
                if(shi[i]<shi[j])
                {
                   float temp=shi[i];
                   shi[i]=shi[j];
                   shi[j]=temp;

                   String s1 = arr[i];
                     arr[i] = arr[j];
                     arr[j] = s1;
                }
             }
          }
        jTextArea1.append("\nFollowing is the sorted (Descending) array, considering \n the
largest distances corresponding to each data point:\n");
          for(int i=0;i<count2;i++)
          {
             String cc=Float.toString(shi[i]);
            jTextArea1.append("\n"+cc);
    }
    jTextArea1.append("\nEnter the value:\n");
      int vall = Integer.parseInt(JOptionPane.showInputDialog("\nEnter a value :\n"));
          String prn = Integer.toString(vall);
          jTextArea1.append(prn+"\n");
          for(int i=0;i<vall;i++)
        {
            String ch =Float.toString(shi[i]);
         jTextArea1.append("\n Data point "+arr[i] + "is an outlier . It's corresponding
distance is" + shi[i]);
                }
        br.close();
    }
         }}
    catch (FileNotFoundException e1) {
        e1.printStackTrace();
    }
    catch (IOException e) {
        e.printStackTrace();
    }
  }
  if(jRadioButton2.isSelected())
      {
          jTextArea1.setText(null);
        try {
            jc.showOpenDialog(null);
            fl = jc.getSelectedFile();
  BufferedReader br1 = new BufferedReader(new FileReader(fl));
        String line1;
        line1= br1.readLine();

        int count=1,count2=1;
        for(int i=0;i<line1.length();i++)
```

```
{
    if(line1.charAt(i)==',')
        count++;
}
for(int i=0;;i++)
{
    line1=br1.readLine();
    if(line1==null)
        break;
    count2++;
}
jTextArea1.append(count+" Dimesions ");
jTextArea1.append(count2+" Samples ");
BufferedReader br = new BufferedReader(new FileReader(fl));
String line;
float[][] col0 = new float[count2][count];
float[][] res = new float[count2][count2];
float shi[]=new float[count2];
for (int i = 0; i < count2; i++) {
    line = br.readLine();
        String[] cols = line.split(",");
        for(int j=0;j<count;j++)
        col0[i][j] = Float.valueOf(cols[j]);
}
float a = 0.0f;
Mahalanobis mb=new Mahalanobis(1);
for(int i=0;i<count2;i++)
```

```
{
        for(int j=0;j<count2;j++)
        {
                a=0;
                for(int v=0;v<count;v++)
                {
                            a=mb.getDistance(col0[i][v], col0[j][v]);
                            String atr =("Combo : "+col0[i][v]+ " & "+col0[j][v]+" = "+a);
                        jTextArea1.append("\n"+atr);
                  String pos =     Float.toString(col0[i][v]);
                }
                res[i][j]=a;
        }
}
jTextArea1.append("\n Enter a value:\n");
int nm = Integer.parseInt(JOptionPane.showInputDialog("\n Enter a value :\n"));
String ln = Integer.toString(nm);
jTextArea1.append("\n" +ln +"\n");
for(int i=0;i<count2;i++)
Arrays.sort(res[i]);
int s=count2-1;
 String arr[] = new String[count2];
for(int i=0;i<count2;i++)
{
        for(int j=0;j<nm;j++)
        {
              String mn =Float.toString(res[i][j]);
              jTextArea1.append(mn + " ");
                arr[i] = new String((i+1) + "_"  );
         }
     jTextArea1.append("\n");
 }
 for (int i = 0; i < arr.length; i++) {
   }
 for(int i=0;i<count2;i++)
 {
 shi[i]=res[i][nm-1];
 }
 for(int i=0;i<s;i++)
 {
    for(int j=i+1;j<count2;j++)
    {
        if(shi[i]<shi[j])
        {
            float temp=shi[i];
            shi[i]=shi[j];
            shi[j]=temp;

            String s1 = arr[i];
              arr[i] = arr[j];
              arr[j] = s1;
        }
    }
 }
```

```
        jTextArea1.append("\nFollowing is the sorted (Descending) array, considering the
\nlargest distances corresponding to each data point:\n");

        for(int i=0;i<count2;i++)

        {

            String asd = Float.toString(shi[i

    jTextArea1.append("\n"+asd);

        }

        jTextArea1.append("\nEnter the value:\n");

         int nmm = Integer.parseInt(JOptionPane.showInputDialog("\nEnter a value :\n"));

        String lnn = Integer.toString(nmm);

          jTextArea1.append(lnn);

        jTextArea1.append("\n Thus, the largest "+nmm+" distances are:");

        for(int i=0;i<nmm;i++)

        {

            String cd = Float.toString(shi[i]);

 jTextArea1.append("\n Data point "+arr[i] +"is an outlier . Its corresponding distance "+
"" + shi[i]);

        }

        br.close();

    }

    catch (FileNotFoundException e1) {

        e1.printStackTrace();

    }

    catch (IOException e) {

        e.printStackTrace();

    }

         }

   }
```

```
// Variables declaration - do not modify
private javax.swing.ButtonGroup buttonGroup1;
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel2;
private javax.swing.JRadioButton jRadioButton1;
private javax.swing.JRadioButton jRadioButton2;
private javax.swing.JScrollPane jScrollPane1;
public javax.swing.JTextArea jTextArea1;
// End of variables declaration
@Override
public void setExplorer(Explorer parent) {
}
@Override
public Explorer getExplorer() {
   return null;
}
@Override
public void setInstances(Instances inst) {
}
@Override
public String getTabTitle() {
     return Messages.getInstance().getString("OutlierPanel_GetTabTitle_Text");
}
```

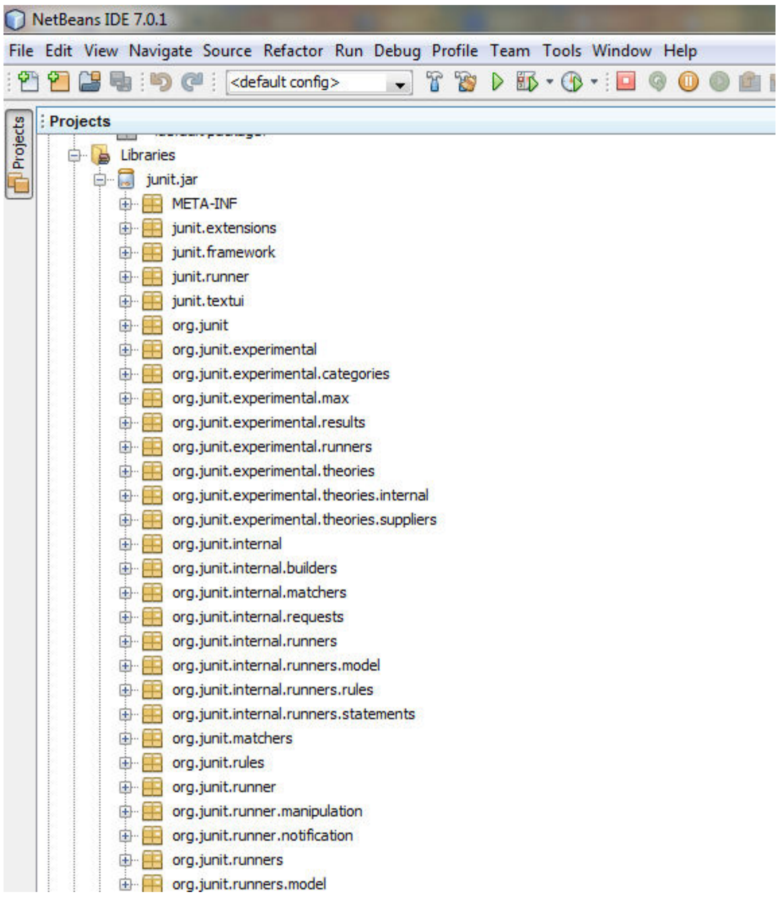
```
  @Override
  public String getTabTitleToolTip() {
      return ";AAA";
  }
}
```

5.2 Designe of outlier panel

The outlier output panel remains the same as for another output panels. Left side comprises of two radio buttons with a stat option at below to run the algorithms incorporated behind the radio buttons one with Eucidian distance and another with Mahalanobis distance approach.

5.3 Additional libraries in source code

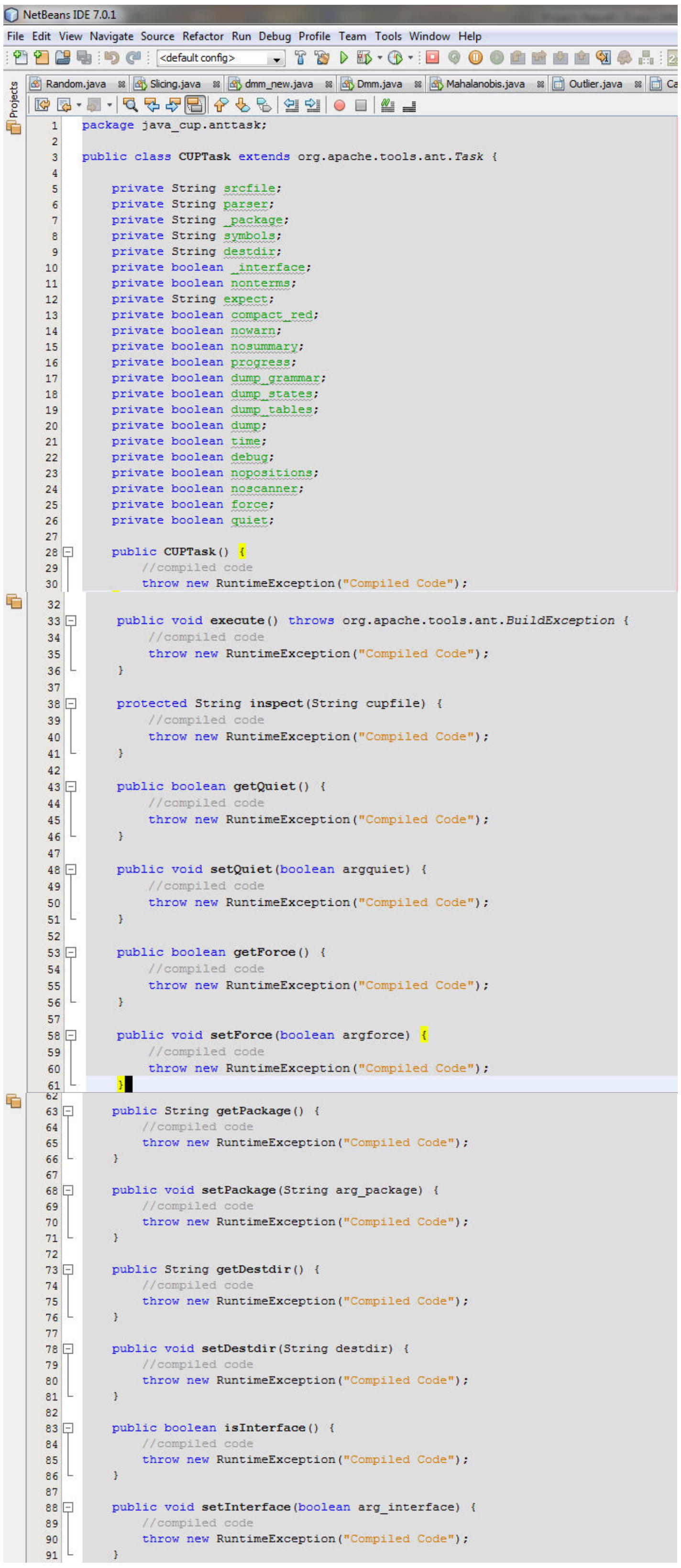
Junit.jar library which must contain all the packages mentioned ahead.


NetBeans IDE 7.0.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Projects

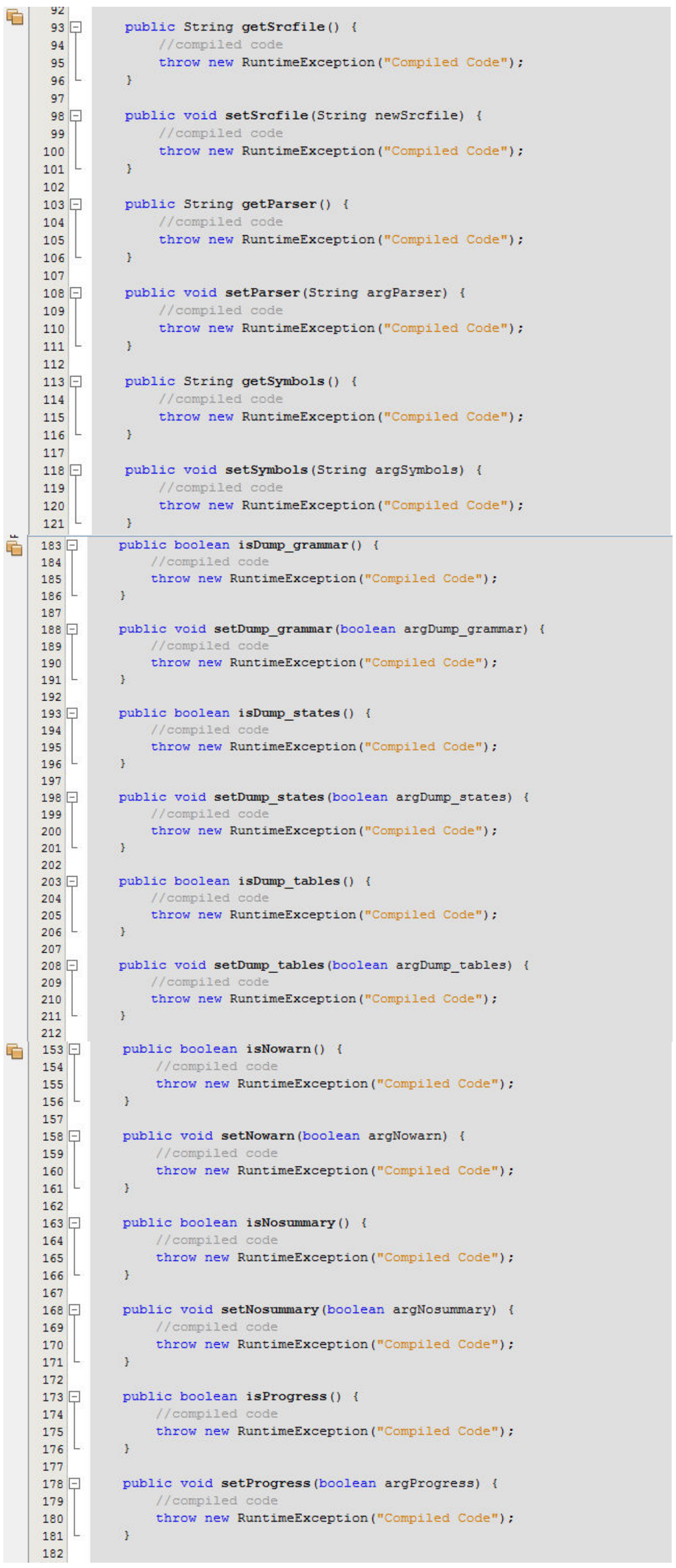
```
Libraries
  junit.jar
    META-INF
    junit.extensions
    junit.framework
    junit.runner
    junit.textui
    org.junit
    org.junit.experimental
    org.junit.experimental.categories
    org.junit.experimental.max
    org.junit.experimental.results
    org.junit.experimental.runners
    org.junit.experimental.theories
    org.junit.experimental.theories.internal
    org.junit.experimental.theories.suppliers
    org.junit.internal
    org.junit.internal.builders
    org.junit.internal.matchers
    org.junit.internal.requests
    org.junit.internal.runners
    org.junit.internal.runners.model
    org.junit.internal.runners.rules
    org.junit.internal.runners.statements
    org.junit.matchers
    org.junit.rules
    org.junit.runner
    org.junit.runner.manipulation
    org.junit.runner.notification
    org.junit.runners
    org.junit.runners.model
```

Java_cup library containing a default package and CUPTask.class

Java code for CUPTask.class



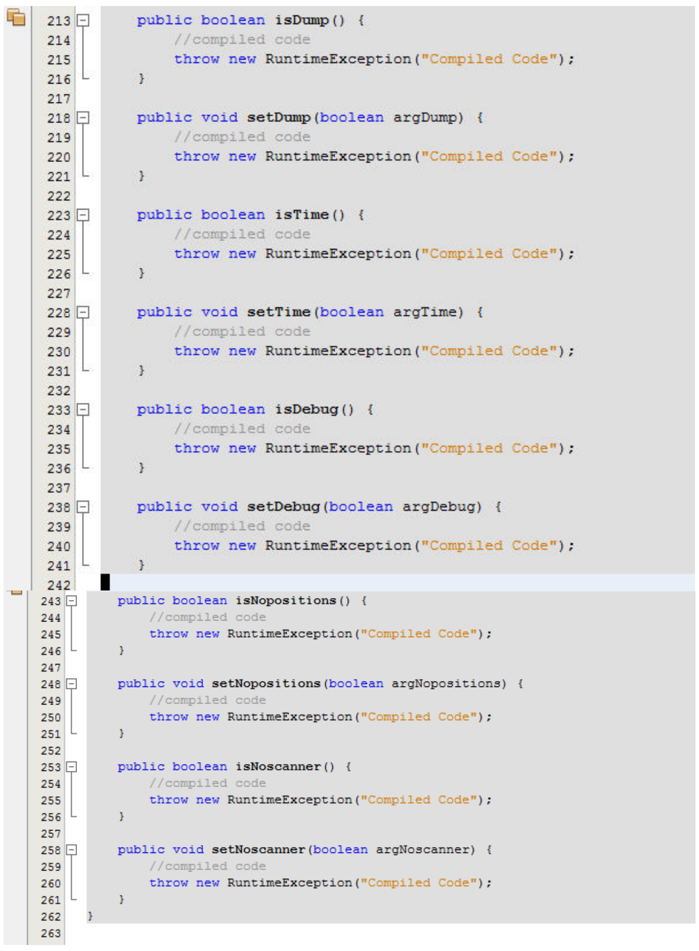
```java
package java_cup.anttask;

public class CUPTask extends org.apache.tools.ant.Task {

    private String srcfile;
    private String parser;
    private String _package;
    private String symbols;
    private String destdir;
    private boolean _interface;
    private boolean nonterms;
    private String expect;
    private boolean compact_red;
    private boolean nowarn;
    private boolean nosummary;
    private boolean progress;
    private boolean dump_grammar;
    private boolean dump_states;
    private boolean dump_tables;
    private boolean dump;
    private boolean time;
    private boolean debug;
    private boolean nopositions;
    private boolean noscanner;
    private boolean force;
    private boolean quiet;

    public CUPTask() {
        //compiled code
        throw new RuntimeException("Compiled Code");

    public void execute() throws org.apache.tools.ant.BuildException {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    protected String inspect(String cupfile) {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public boolean getQuiet() {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public void setQuiet(boolean argquiet) {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public boolean getForce() {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public void setForce(boolean argforce) {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public String getPackage() {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public void setPackage(String arg_package) {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public String getDestdir() {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public void setDestdir(String destdir) {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public boolean isInterface() {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public void setInterface(boolean arg_interface) {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }
```



```
    public String getSrcfile() {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public void setSrcfile(String newSrcfile) {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public String getParser() {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public void setParser(String argParser) {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public String getSymbols() {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public void setSymbols(String argSymbols) {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }
```

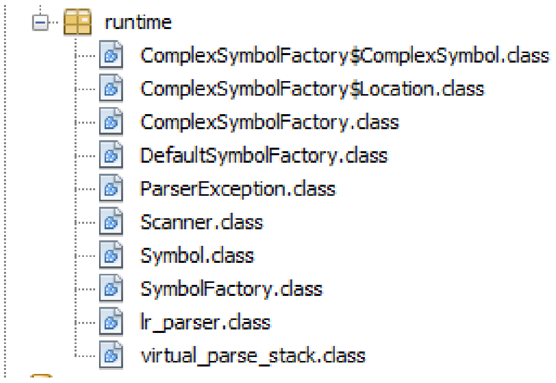
```
    public boolean isDump_grammar() {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public void setDump_grammar(boolean argDump_grammar) {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public boolean isDump_states() {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public void setDump_states(boolean argDump_states) {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public boolean isDump_tables() {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public void setDump_tables(boolean argDump_tables) {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

```

```
    public boolean isNowarn() {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public void setNowarn(boolean argNowarn) {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public boolean isNosummary() {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public void setNosummary(boolean argNosummary) {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public boolean isProgress() {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public void setProgress(boolean argProgress) {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

```



```
    public boolean isDump() {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public void setDump(boolean argDump) {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public boolean isTime() {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public void setTime(boolean argTime) {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public boolean isDebug() {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public void setDebug(boolean argDebug) {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public boolean isNopositions() {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public void setNopositions(boolean argNopositions) {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public boolean isNoscanner() {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }

    public void setNoscanner(boolean argNoscanner) {
        //compiled code
        throw new RuntimeException("Compiled Code");
    }
}
```

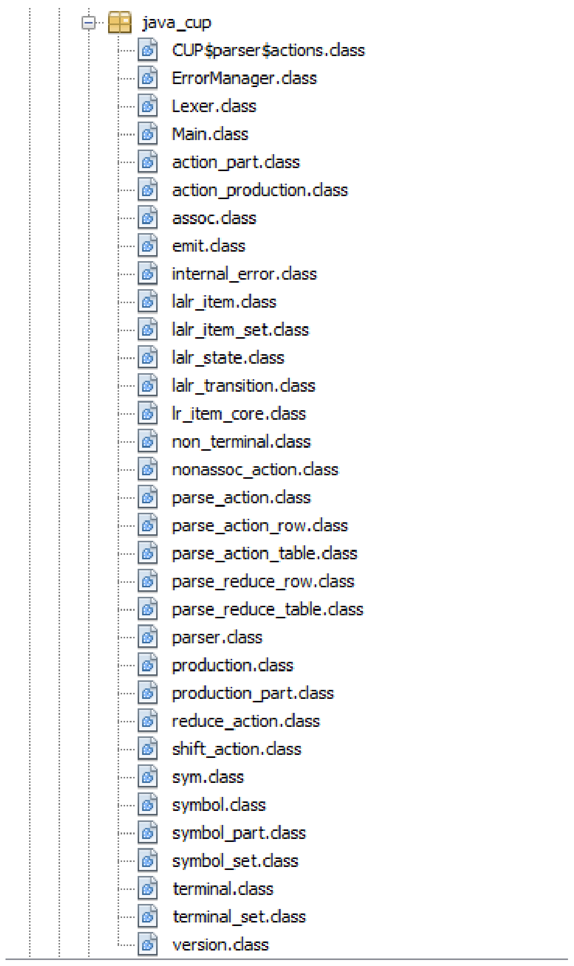
Further under it a runtime package following classes are required to run the application



```
runtime
    ComplexSymbolFactory$ComplexSymbol.class
    ComplexSymbolFactory$Location.class
    ComplexSymbolFactory.class
    DefaultSymbolFactory.class
    ParserException.class
    Scanner.class
    Symbol.class
    SymbolFactory.class
    lr_parser.class
    virtual_parse_stack.class
```

Last additional library is cup_11.jar having a packages java-cup , java_cup.runtime and java_cup.anttask

a). java-cup

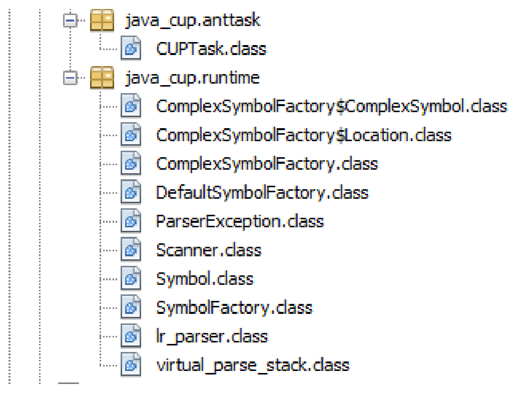


```
java_cup
    CUP$parser$actions.class
    ErrorManager.class
    Lexer.class
    Main.class
    action_part.class
    action_production.class
    assoc.class
    emit.class
    internal_error.class
    lalr_item.class
    lalr_item_set.class
    lalr_state.class
    lalr_transition.class
    lr_item_core.class
    non_terminal.class
    nonassoc_action.class
    parse_action.class
    parse_action_row.class
    parse_action_table.class
    parse_reduce_row.class
    parse_reduce_table.class
    parser.class
    production.class
    production_part.class
    reduce_action.class
    shift_action.class
    sym.class
    symbol.class
    symbol_part.class
    symbol_set.class
    terminal.class
    terminal_set.class
    version.class
```

b). java_cup.runtime and

c). java_cup.anttask

CUPTask.class code is mentioned here



```
java_cup.anttask
    CUPTask.class
java_cup.runtime
    ComplexSymbolFactory$ComplexSymbol.class
    ComplexSymbolFactory$Location.class
    ComplexSymbolFactory.class
    DefaultSymbolFactory.class
    ParserException.class
    Scanner.class
    Symbol.class
    SymbolFactory.class
    lr_parser.class
    virtual_parse_stack.class
```

5.4 Algorithm using Euclidian distance

Java code for Dmm.java

```
package dmm;

import java.io.*;

import java.util.Arrays;

import java.util.Scanner;

public class Dmm {

   public static void main(String[] args) throws Exception {

       try {

 BufferedReader br1 = new
BufferedReader(newFileReader("C:\\Users\\hp\\Documents\\iris.csv"));

            String line1;

            line1 = br1.readLine();

            int count = 1, count2 = 1;

            for (int i = 0; i < line1.length(); i++) {

                if (line1.charAt(i) == ',') {

                    count++;

                }

            }

            for (int i = 0;; i++) {

                line1 = br1.readLine();

                if (line1 == null) {

                    break;

                }

                count2++;

            }
```

```
        System.out.println(count + " Dimensions");

        System.out.println(count2 + " Samples");

 BufferedReader br = new
BufferedReader(newFileReader("C:\\Users\\hp\\Documents\\iris.csv"));

        String line;

        float[][] col0 = new float[count2][count];

        float[][] res = new float[count2][count2];

        float shi[] = new float[count2];

        for (int i = 0; i < count2; i++) {

           line = br.readLine();

           String[] cols = line.split(",");

           for (int j = 0; j < count; j++) {

               col0[i][j] = Float.valueOf(cols[j]);

           }

        }

        float a;

        for (int i = 0; i < count2; i++) {

           for (int j = 0; j < count2; j++) {

               a = 0;

               for (int v = 0; v < count; v++) {

                   a += ((col0[i][v] - col0[j][v]) * (col0[i][v] - col0[j][v]));

               System.out.println("Combo : " + col0[i][v] + " & " + col0[j][v] + " = " + a);

               }

               res[i][j] = (float) Math.sqrt(a);

           }

        }

        int c;
```

```
Scanner in = new Scanner(System.in);
System.out.println("Enter a value");
c = in.nextInt();
for (int i = 0; i < count2; i++) {
   Arrays.sort(res[i]);
}
int s = count2 - 1;
String arr[] = new String[count2];
for (int i = 0; i < count2; i++) {
   for (int j = 0; j < c; j++) {
      System.out.print(res[i][j]);
      arr[i] = new String((i+1) + "_");
   }
   System.out.println();
}
for (int i = 0; i < arr.length; i++) {
}
for (int i = 0; i < count2; i++) {
   shi[i] = res[i][c - 1];
}
for (int i = 0; i < s; i++) {
   for (int j = i + 1; j < count2; j++) {
      if (shi[i] < shi[j]) {
         float temp = shi[i];
         shi[i] = shi[j];
         shi[j] = temp;
         String s1 = arr[i];
```

```
                    arr[i] = arr[j];

                    arr[j] = s1;

                }

            }

        }

 System.out.println("\nFollowing is the sorted (descending) array, considering the largest
distances corresponding to each data point:\n");

          System.out.println(shi[i]);

           int choice;

           System.out.println("\nEnter the value:");

           choice = in.nextInt();

           System.out.println("Thus, the largest " + choice + " distances are:");

           //for(int i=s-choice;i<count2;i++)

           for (int i = 0; i < choice; i++) {

 System.out.println("data point "+arr[i] + is an outlier . Distance corresponding to the point
is  " + shi[i]);

           }

           br.close();

        } catch (FileNotFoundException e1) {

           e1.printStackTrace();

        } catch (IOException e) {

           e.printStackTrace();

        }

    }

}
```

5.5 Algorithm using Mahalanobis Distance approach

a). Java code for dmm_new.java

```
package dmm;
```

```
import java.io.*;

import java.util.Arrays;

import java.util.Scanner;

public class dmm_new {

   public void show1()

   {

   try {

 BufferedReader br1 = new
BufferedReader(newFileReader("C:\\Users\\hp\\Documents\\iris.csv "));

        String line1;

        line1= br1.readLine();

        int count=1,count2=1;

        for(int i=0;i<line1.length();i++)

        {

            if(line1.charAt(i)==',')

                count++;

        }

        for(int i=0;;i++)

        {

            line1=br1.readLine();

            if(line1==null)

                break;

            count2++;

        }

        System.out.println(count+" Dimesions");

        System.out.println(count2+" Samples");

  BufferedReader br = new
BufferedReader(newFileReader("C:\\Users\\hp\\Documents\\iris.csv "));
```

```
String line;
float[][] col0 = new float[count2][count];
float[][] res = new float[count2][count2];
float shi[]=new float[count2];
for (int i = 0; i < count2; i++) {
    line = br.readLine();
        String[] cols = line.split(",");
        for(int j=0;j<count;j++)
        col0[i][j] = Float.valueOf(cols[j]);
}
float a = 0.0f;
Mahalanobis mb=new Mahalanobis(1);
for(int i=0;i<count2;i++)
{
        for(int j=0;j<count2;j++)
        {
                a=0;
                for(int v=0;v<count;v++)
                {
                        a=mb.getDistance(col0[i][v], col0[j][v]);
                System.out.println("Combo : "+col0[i][v]+ " & "+col0[j][v]+" = "+a);
                }
                res[i][j]=a;
        }
}
int c;
Scanner in = new Scanner(System.in);
```

```
System.out.println("Enter a value");
c = in.nextInt();
for(int i=0;i<count2;i++)
Arrays.sort(res[i]);
int s=count2-1;
for(int i=0;i<count2;i++)
{
        for(int j=0;j<c;j++)
        {
              System.out.print(res[i][j]);
              System.out.print(' ');
        }
     System.out.println();
}
for(int i=0;i<count2;i++)
{
shi[i]=res[i][c-1];
}
for(int i=0;i<s;i++)
{
    for(int j=i+1;j<count2;j++)
    {
        if(shi[i]<shi[j])
        {
            float temp=shi[i];
            shi[i]=shi[j];
            shi[j]=temp;
```

```
                }
            }
        }
        System.out.println("\nFollowing is the sorted (descending) array, considering the
largest distances corresponding to each data point:\n");
        for(int i=0;i<count2;i++)
            System.out.println(shi[i]);
        int choice;
        System.out.println("\nEnter the value:");
        choice=in.nextInt();
        System.out.println("Thus, the largest "+choice+" distances are:");
      //for(int i=s-choice;i<count2;i++)
        for(int i=0;i<choice;i++)
            System.out.println(shi[i]);
  br.close();
    }
    catch (FileNotFoundException e1) {
        e1.printStackTrace();
    }
    catch (IOException e) {
        e.printStackTrace();
    }
  }

  }
```

b).Java code for Mahalanobis.java

```
package dmm;
import dmm.org.math.array.LinearAlgebra;
public class Mahalanobis {
    // the covariance matrix
    private double[][] S;
    public static void main(String[] args) {
        double x = 5.1;
        double y = 5.1;
        Mahalanobis mah = new Mahalanobis(1);
        System.out.println(mah.getDistance(x, y));
    }
    public Mahalanobis(int dim) {
        S = new double[dim][dim];
        for(int i=0; i<dim; i++)
            for(int j=0; j<dim; j++)
                if(i == j)
                    S[i][j] = 1.0;
                else
                    S[i][j] = 0.0;
    }
    public float getDistance(double x, double y) {
        double[][] diff = new double[1][1];
        for(int i=0; i<1; i++)
            diff[0][i] = x - y;
        double result[][] = LinearAlgebra.times( diff, LinearAlgebra.inverse(S) );
        result = LinearAlgebra.times( result, LinearAlgebra.transpose(diff) );
    return (float)Math.sqrt(result[0][0]);
}
public double getSimilarity(double x, double  y) {
    return 1.0 / (1.0 + getDistance(x, y));
}   }
```

Chapter6 :CONCLUSION

CONCLUSION AND FUTURE SCOPE.

I have created a new panel that consists of most popular outlier detection algorithms. This panel enables users to apply outlier detection algorithms to the given dataset.
Standard techniques were used in process.
The relatively low percentage of detection rate as compared to other forms of outlier detection suites indicates that the algorithms used are not very robust and is vulnerable to effects like scaling and elastic deformations.
Also a major challenge in outlier detection is defining a normal region which encompasses every possible normal behavior. It is very difficult to achieve hundred percent accuracy in anomaly detection and this also adds to the low detection rate.
As a future scope we can take a large database and make outlier detection module with more improvised algorithms to detect anomalies correctly to the maximum extent.
Such large systems are used in various fields like airlines and traffic pattern monitoring and control, medical fields, for an anomalous MRI image may indicate the presence of malignant tumors, in credit card identity and thefts, in space research and other areas of research.

REFERENCES

1. Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM Comput. Surv.* 41, 3, Article 15 (July 2009), 58 pages.DOI=10.1145/1541880.1541882 http://doi.acm.org/10.1145/1541880.1541882

2. Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. LOF: identifying density-based local outliers. *SIGMOD Rec.* 29, 2 (May 2000),93-104.DOI=10.1145/335191.335388 http://doi.acm.org/10.1145/335191.335388

3. Xiuyao Song, Mingxi Wu, Christopher Jermaine, and Sanjay Ranka. 2007. Conditional Anomaly Detection. *IEEE Trans. On Knowl. And Data Eng.* 19, 5 (May2007),631-645.DOI=10.1109/TKDE.2007.1009 http://dx.doi.org/10.1109/TKDE.2007.1009

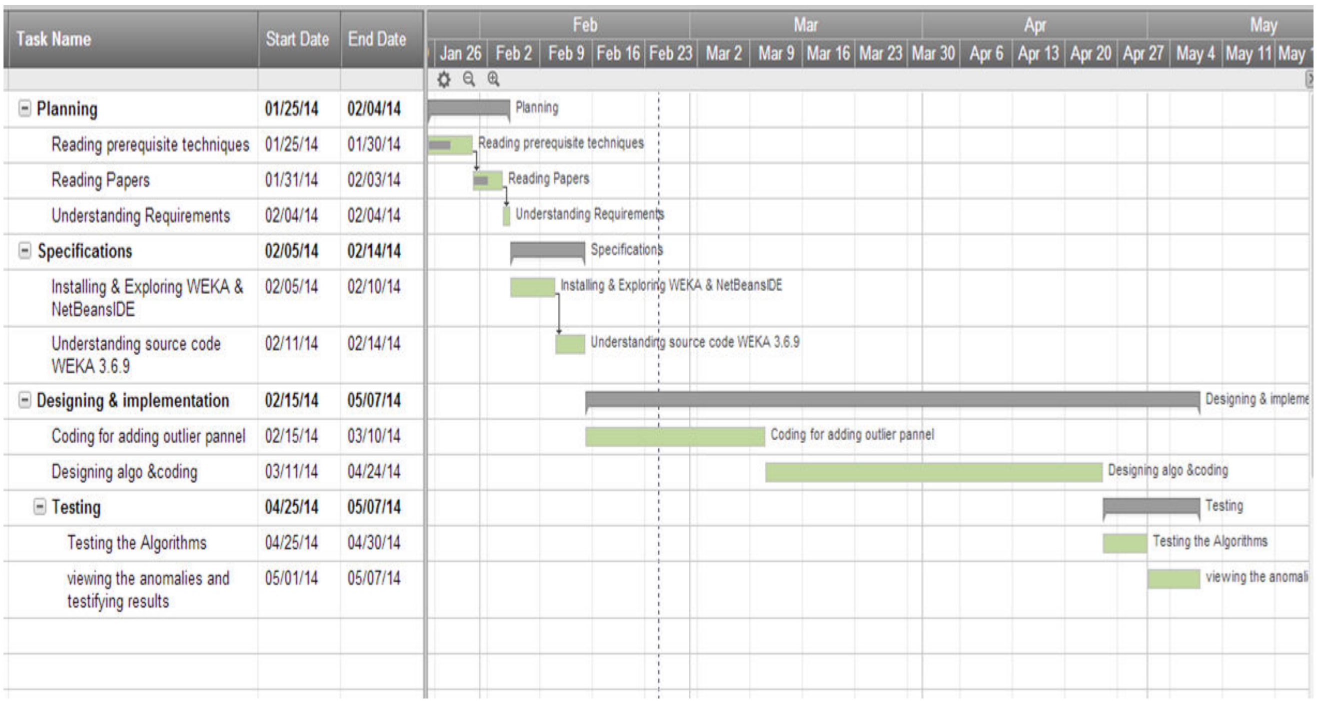
4. KNORR, E. M., NG, R. T., AND TUCAKOV, V. 2000. Distance-based outliers: Algorithms and applications. *VLDB*

J. 8, 3-4, 237–253.

5. Victoria J. Hodge and Jim. 2004. A Survey of Outlier Detection Methodologies. AIRE381

APPENDICES

APPENDIX A: Work Plan



Task Name	Start Date	End Date
Planning	**01/25/14**	**02/04/14**
Reading prerequisite techniques	01/25/14	01/30/14
Reading Papers	01/31/14	02/03/14
Understanding Requirements	02/04/14	02/04/14
Specifications	**02/05/14**	**02/14/14**
Installing & Exploring WEKA & NetBeansIDE	02/05/14	02/10/14
Understanding source code WEKA 3.6.9	02/11/14	02/14/14
Designing & implementation	**02/15/14**	**05/07/14**
Coding for adding outlier pannel	02/15/14	03/10/14
Designing algo &coding	03/11/14	04/24/14
Testing	**04/25/14**	**05/07/14**
Testing the Algorithms	04/25/14	04/30/14
viewing the anomalies and testifying results	05/01/14	05/07/14

APPENDIX B: Tools Description

NetBeansIDE 7.0.1

The NetBeans IDE is an award-winning integrated development environment available for Windows, Mac, Linux, and Solaris. The NetBeans project consists of an open-source IDE and an application platform that enable developers to rapidly create web, enterprise, desktop, and mobile applications using the Java platform, as well as PHP, JavaScript and Ajax, Groovy and Grails, and C/C++.

The Net Beans project is supported by a vibrant developer community and offers extensive documentation and training resources as well as a diverse selection of third-party plugins.

NetBeans IDE 7.0 introduces language support for development to the Java SE 7 specification with JDK 7 language features. The release also provides enhanced integration with the Oracle WebLogic server, as well as support for Oracle Database and GlassFish 3.1. Additional highlights include Maven 3 and HTML5 editing support; a new GridBagLayout designer for improved Swing GUI development; enhancements to the Java editor, and more.

NetBeans IDE 7.0 is available in English, Brazilian Portuguese, Japanese, Russian, and Simplified Chinese.

NetBeans IDE 7.0.1 includes the following notable changes:

- Full JDK 7 support: Running NetBeans IDE on top of JDK 7 and support for the final version of the JDK 7 language features
- Integration of the recent patches
- Performance improvements

APPENDIX C: Quality Assurance

Quality assurance, or QA for short, is the systematic monitoring and evaluation of the various aspects of a project, service or facility to maximise the probability that minimum standards of quality are being attained by the production process. Two principles included in QA are “Fit for Purpose” – the product should be suitable for the intended purpose; and “Right first time”- mistakes should be eliminated.

Fit for purpose:
This project is fit for the purpose of detecting outliers in any dataset carrying float type samples.

Right first time:
Till now 75% of accuracy is achieved.

Failure Testing:
I will definitely consider the test cases on it so as to increase its accuracy and make it a better project than existing ones.

Programming style and testing:
I have only used java for coding of all the algorithms and Netbeans development environment.

APPENDIX D: Data Sets

Iris Data Set

Data Set Description

Database: from Fisher, 1936

Data Set Characteristics:	Multivariate	Number of Instances:	150	Area:	Life
Attribute Characteristics:	Real	Number of Attributes:	4	Date Donated	1988-07-01
Associated Tasks:	Classification	Missing Values?	No	Number of Web Hits:	564556

Creator: R.A. Fisher

Data Set Information:

This is perhaps the best known database to be found in the pattern recognition literature. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other. Predicted attribute: class of iris plant.

Attribute Information:

1. Sepal length in cm
2. Sepal width in cm
3. Petal length in cm
4. Petal width in cm
5. Class:

-- Iris Setose
-- Iris Versicolour
-- Iris Virginica

Wine Data Set

Data Set Description

Using chemical analysis determine the origin of wines

Data Set Characteristics:	Multivariate	Number of Instances:	178	Area:	Physical
Attribute Characteristics:	Integer, Real	Number of Attributes:	13	Date Donated	1991-07-01
Associated Tasks:	Classification	Missing Values?	No	Number of Web Hits:	338591

Original Owners:
Forina, M. et al, PARVUS -
An Extendible Package for Data Exploration, Classification and Correlation.
Institute of Pharmaceutical and Food Analysis and Technologies, Via Brigata Salerno, 16147 Genoa, Italy.

Data Set Information:

These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

The attributes are

1) Alcohol
2) Malic acid
3) Ash
4) Alcalinity of ash
5) Magnesium
6) Total phenols
7) Flavanoids
8) Nonflavanoid phenols
9) Proanthocyanins
10)Color intensity
11)Hue
12)OD280/OD315 of diluted wines
13)Proline

Attribute Information: All attributes are continuous

ABSTRACT

WEKA (Waikato Environment for Knowledge Analysis) is open source which is a collection of machine learning algorithms for data mining tasks. WEKA contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. The workbench also provides a graphical user interface for easy access to these functionalities. There are several advantages to this software that includes probability and usability. Many researchers in industry and academia including students use this software because of it support the standard data mining tasks with good usability .Also, one of the best features of this software is that, it is well suited for developing new machine learning schemes. Taking advantage of this feature, I extended this software to contain in the outlier detection module as well .I have developed set of outlier algorithms and incorporated them under a new module in the weka software.

Currently the weka software includes different interfaces like Simple Command Line interface, Explorer, Experimenter and knowledge flow. Explorer is the main interface where the same functionality can also be obtained through the command line or knowledge flow. Experimenter is used for comparisons on different types of panels which provide access to main component of this workbench.

1. Preprocess panel is used to take the input data and analyze the data .Also we can preprocess the data by using certain filtering algorithm thereby processing the data according to the requirements.
2. Classify panel consists of different classification algorithms. This panel provides a facility to apply the classification algorithms to the given dataset, estimate the accuracy of the resulting predictive model, to generate ROC curves and also to visualize.
3. Associate panel consists of association rule mining algorithms. This panel helps in identifying the interesting rules from the given dataset.
4. Cluster panel consists of clustering algorithms that helps in cluster the given dataset.
5. Select attributes panel provide algorithms to identify the most predictive attributes in the given dataset.
6. Visualize panel helps in providing a scatter plot matrix which consists of individual scatter plots for the given dataset.

As my contribution, I have created a new panel that consists of most popular outlier detection algorithms. This panel enables users to apply outlier detection algorithms to the given dataset.