

Bayesian Network Based Early Disease Outbreak Detection Software

Project Report Submitted in partial fulfillment of the requirement

For the degree of

Bachelor of Technology

In

Computer Science Engineering

Under the Supervision of

Dr. Sakshi Babbar

By

Kumar Vaibhav 101255

To



JAYPEE UNIVERSITY OF INFORMATION AND TECHNOLOGY

WAKNAGHAT, SOLAN-173215, HIMACHAL PRADESH

JAYPEE UNIVERSITY OF INFORMATION AND TECHNOLOGY

WAKNAGHAT, SOLAN-173215, HIMACHAL PRADESH

CERTIFICATE

This is to certify that the work titled “**Bayesian Network Based Early Disease Outbreak Detection Software using What’s strange about recent events**” Submitted By **Kumar Vaibhav (101255)** in partial fulfillment for the award of degree of B.Tech Computer Science Engineering of Jaypee University of Information Technology, Wagnaghat 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, Dept. of CSE and ICT

Jaypee University of Information Technology

Date:

ACKNOWLEDGEMENT

I would like to express my greatest gratitude to all the people who have helped and supported me throughout the project. Thanks and appreciation to the helpful people at college for their support. I would also thank our 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 well wishers.

A special thanks to all the fellow students who helped me in completing the project and exchanged their interesting ideas, thoughts and made this project easy and accurate.

KUMAR VAIBHAV

101255

B.Tech-CSE

DATE:

Table of Contents

Certificate

Acknowledgement

Table of Contents

Abstract

References

Chapters

1 Introduction	2-5
1.1 Introduction	
1.2 Problem Statement	
1.3 Motivation	
1.4 Deliverables of the Project	
2 Literature Survey	6-20
2.1 What's strange about recent events	
2.1.1 One component rule	
2.1.2 Two component rule	
2.1.3 Finding the p-value for a rule	
2.1.4 WSARE(3.0)	
2.2 Introduction to Bayesian Networks	
3 Requirement Analysis	21-24
3.1 Netbeans	
3.2 Netica	
3.3 B Course	

4 Bayesian Network based disease outbreak detection software	25-36
4.1 Implementation of the algorithm	
4.1.1 WSARE (2.0)	
4.1.2 WSARE (3.0)	
5 Coding	37-54
6 Conclusion	55

Abstract

Traditional bio surveillance algorithms detect disease outbreaks by looking for peaks in a uni-variate time series of health-care data. Current health-care surveillance data, however, are no longer simply uni-variate data streams. Instead, a wealth of spatial, temporal, demographic and symptomatic information is available. Here is an early disease outbreak detection algorithm called What's Strange About Recent Events (WSARE), which uses a multivariate approach to improve its timeliness of detection. WSARE employs a rule-based technique that compares recent health-care data against data from a baseline distribution and finds subgroups of the recent data which shows trend. In addition, health-care data also pose difficulties for surveillance algorithms because of inherent temporal trends such as seasonal effects and day of week variations. WSARE approaches this problem using a Bayesian network to produce a baseline distribution that accounts for these temporal trends.

Chapter – 1

Introduction

1.1) Introduction

Detection systems inspect routinely collected data for anomalies and raise an alert upon discovery of any significant deviations from the norm. Here is how to tackle the problem of early disease outbreak detection, in which the disease outbreak can be due to either natural causes or a bioterrorist attack.

In this surveillance infrastructure, the database which was used was from the emergency department (ED) cases from several hospitals in a city. Each record in this multivariate database contains information about the individual who is admitted to the ED. This information includes fields such as age, gender, symptoms exhibited, home zip code, work zip code, and time of arrival at the ED. When a severe epidemic sweeps through a region, there will obviously be extreme perturbations in the number of ED visits. While these dramatic upswings are easily noticed during the late stages of an epidemic, the challenge is to detect the outbreak during its early stages and mitigate its effects.

Traditional anomaly detection algorithms are inappropriate for this domain. In the traditional approach, a probabilistic model of the baseline data is built. Anomalies are identified as individual data points with a rare attribute or rare combination of attributes. If we apply traditional anomaly detection to the ED data, the results would be, for example, a patient that is over a hundred years old living in a sparsely populated region of the city. These isolated outliers in attribute space are not at all indicative of a disease outbreak.

This technique works well if we know beforehand which disease to monitor. In this situation, a non-specific disease monitoring is to performed because there is no knowledge about the disease to expect.

The approach to early disease outbreak detection uses a rule-based anomaly pattern detector called What's Strange About Recent Events (WSARE). WSARE operates on discrete, multidimensional data sets with a temporal component.

The 2.0 and 3.0 of the WSARE algorithm are presented here. These algorithms only differ in how they create the baseline distribution; all other steps in the WSARE framework remain identical. WSARE 2.0 uses raw historical data from selected days as the baseline while WSARE 3.0 models the baseline distribution using a Bayesian network.

1.2) Problem Statement

Detection of an “epidemic” as early as possible and to mine anomalous pattern of specific characteristics whose recent pattern of illness is anomalous relative to historical patterns. Where historical patterns are captured using Bayesian network.

1.3) Motivation

What motivated me to choose this project was the concern about the detection of an epidemic as early as possible. The idea was to differentiate among the diseases whose early symptoms are similar.

1.4) Deliverables of the Project

The deliverables of the project is an algorithm which detects an epidemic at an early stage using the non Bayesian approach as well as the Bayesian Network approach.

Chapter – 2

Literature Survey

2.1) What's Strange About Recent Events

The basic question asked by all detection systems is whether anything strange has occurred in recent events. This question requires defining what it means to be recent and what it means to be strange. WSARE algorithm considers all patient records falling on the current day under evaluation to be recent events. Note that this definition of recent is not restrictive – the approach is fully general and recent can be defined to include all events within some other time period such as over the last six hours. In order to define an anomaly, we need to establish the concept of something being normal.

In WSARE version 2.0, baseline behaviour is assumed to be captured by raw historical data from the same day of the week in order to avoid environmental effects such as weekend versus weekday differences in the number of ED cases. This baseline period must be chosen from a time period similar to the current day. This can be achieved by being close enough to the current day to capture any seasonal or recent trends. On the other hand, the baseline period must also be sufficiently distant from the current day.

If the baseline period is too close to the current day, the baseline period will quickly incorporate the outbreak cases as time progresses. In the description of WSARE 2.0, the assumption is that the baseline behaviour is captured by records that are in the set *baseline days*. Typically, *baseline days* contains the days that are 35, 42, 49, and 56 days prior to the day under consideration. Later in the report the illustration of the version 3.0 of WSARE automatically generates the baseline using a Bayesian network. The events that fit a certain rule for the current day are termed as C_{recent} . Similarly, the number of cases matching the same rule from the baseline period will be called $C_{baseline}$. As an example, suppose the current day is Tuesday December 30, 2003. The baseline used for WSARE 2.0 will then be November 4, 11, 18 and 25 of 2003 as seen in Figure 1. These dates are all from Tuesdays in order to avoid day of week variations.

Here is the schematic overview of the algorithm which will be used in the implementation of the project and the table 1 shows the main parameters of WSARE.

WSARE first finds the best scoring rule over events occurring on the current day using a greedy search. The score of a rule is determined by comparing the events on the current day against events in the past. More specifically, the comparison is if the ratio between certain events on the current day and the total number of events on the current day differ dramatically between the recent period and the past. Following the score calculation, the best rule for that day has its p-value estimated by a randomization test. The p-value for a rule is the likelihood of finding a rule with as good a score under the hypothesis that the date and the other attributes are independent. If the algorithm is ran on a day-by-day basis we would end at this step. However, if we are looking at a history of days and we want to control for some level of false discoveries over this group of days, we would need the additional step of using the False Discovery Rate (FDR) method to determine which of the p-values are significant. The days with significant p-values are returned as the anomalies.

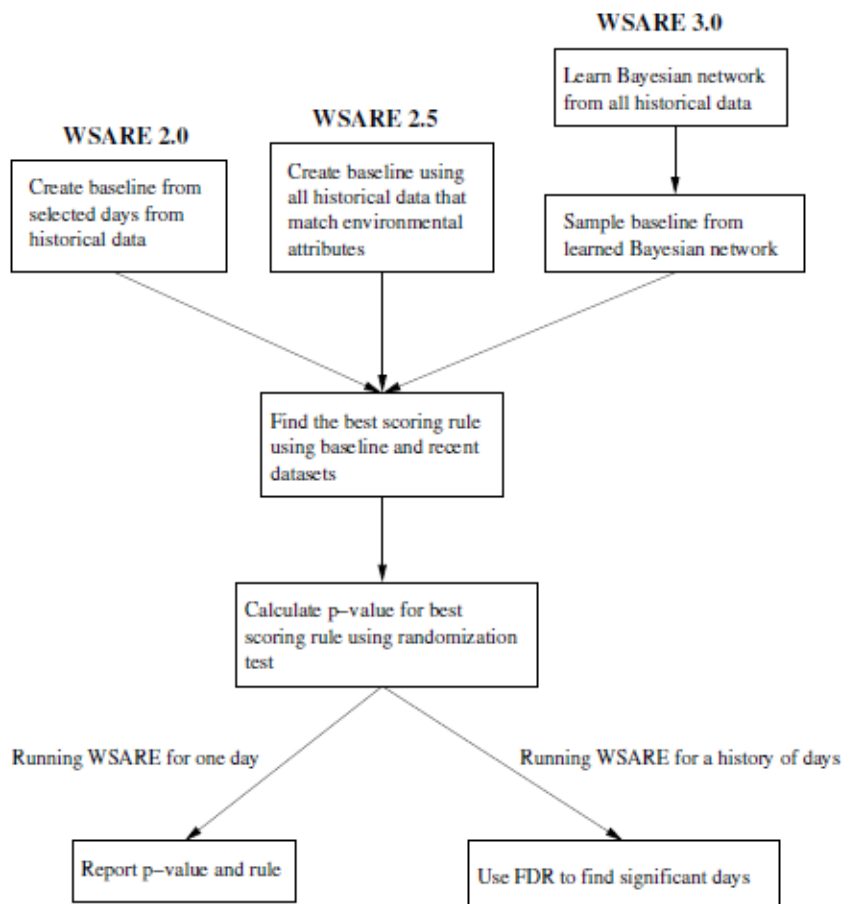


Figure 1: A schematic overview of the steps involved in the WSARE algorithms.

Parameter Name	Description	Default value
<i>max_rule_components</i>	Maximum number of components to a rule	2
<i>num_randomizations</i>	Number of iterations to the randomization test	1000
α_{FDR}	The significance level of the False Discovery Rate	0.05
<i>baseline_days</i> (WSARE 2.0 only)	Days to be used for the baseline	35, 42, 49, and 56 days prior to current date
<i>environmental_attributes</i> (WSARE 2.5 and 3.0)	Attributes that account for temporal trends	Not applicable
<i>num_baseline_samples</i> (WSARE 3.0 only)	The number of sampled records from the baseline Bayesian network	10000

Table 1: The main parameters in WSARE.

2.1.1) WSARE (2.0)

The algorithms differ only in how they create the baseline while all of the other steps remain identical. Figure 2 shows how the baseline of WSARE 2.0 is created. The other steps are described in the later sections.

<i>November 2003</i>								
<i>Su</i>	<i>Mo</i>	<i>Tu</i>	<i>We</i>	<i>Th</i>	<i>Fr</i>	<i>Sa</i>	<i>1</i>	<i>2</i>
			3	4	5	6	7	8
9	10	11	12	13	14	15		
16	17	18	19	20	21	22		
23	24	25	26	27	28	29		
30								

<i>December 2003</i>								
<i>Su</i>	<i>Mo</i>	<i>Tu</i>	<i>We</i>	<i>Th</i>	<i>Fr</i>	<i>Sa</i>		
	1	2	3	4	5	6		
7	8	9	10	11	12	13		
14	15	16	17	18	19	20		
21	22	23	24	25	26	27		
28	29	30	31					

Figure 2: The baseline for WSARE 2.0 if the current day is December 30, 2003

2.1.2) One Component Rule

In order to illustrate this algorithm, suppose we have a large database of 1,000,000 ED records over a two-year span. This database contains roughly 1370 records a day. Suppose we treat all records within the last 24 hours as “recent” events. In addition, we can build a baseline data set out of all cases from exactly 35, 42, 49, and 56 days prior to the current day. We then combine the recent and baseline data to form a record subset called DB_i , which will have approximately 5000 records.

The algorithm proceeds as follows. For each day i in the surveillance period, retrieve the records belonging to DB_i . We first consider all possible one-component rules. For every possible

attribute value combination, obtain the counts C_{recent} and $C_{baseline}$ from the data set DB_i . As an example, suppose the attribute under consideration is Age for the ED case. There are 3 possible values for Age (for making the computation easy), ranging between child, senior and working persons. We start with the rule $Age = child$ and count the number of cases for the current day i that have $Age = child$ and those that have $Age \neq child$. The cases from five to eight weeks ago are subsequently examined to obtain the counts for the cases matching the rule and those not matching the rule.

Later in the report the code for calculating one component rule is given. The result of the code of one component rule is given in Figure 3, in the form of a two-by-two contingency table.

	C_{recent}	$C_{baseline}$
$Age = child$	6	496
$Age \neq child$	40	9504

Table 2: A Sample 2x2 Contingency Table

2.1.3) Two Component Rule

At this point, the best one component rule for a particular day has been found. We will refer to the best one component rule for day i as $BR1_i$. The algorithm then attempts to find the best two component rule for the day by adding on one extra component to $BR1_i$ through a greedy search. This extra component is determined by supplementing $BR1_i$ with all possible attribute-value pairs, except for the one already present in $BR1_i$, and selecting the resulting two component rule with the best score.

Scoring is performed in the exact same manner as before, except the counts C_{recent} and $C_{baseline}$ are calculated by counting the records that match the two component rule. The best two-component rule for day i is subsequently found and we will refer to it as $BR2_i$. Suppose $BR1_i$ has as its first component the attribute-value pair $C1 = V1$. Furthermore, let $BR2_i$'s components be $C1 = V1$ and $C2 = V2$. Adding the component $C2 = V2$ to $BR1_i$ may not result in a better scoring

rule. During the search for the best scoring two component rule, we only consider two component rules in which adding either component has a significant effect. Determining if either component has a significant effect can be done through two hypothesis tests. In the first hypothesis test, use Fisher's exact test to determine the score of adding $C_2 = V_2$ to the one component rule $C_1 = V_1$. Similarly, in the second hypothesis test, we use Fisher's exact test to score the addition of the component $C_1 = V_1$ to $C_2 = V_2$. The 2-by-2 contingency tables used by the two hypothesis tests are shown in Table 2.

Records from Today with $C_1 = V_1$ and $C_2 = V_2$	Records from Other with $C_1 = V_1$ and $C_2 = V_2$
Records from Today with $C_1 \neq V_1$ and $C_2 = V_2$	Records from Other with $C_1 \neq V_1$ and $C_2 = V_2$
Records from Today with $C_1 = V_1$ and $C_2 \neq V_2$	Records from Other with $C_1 = V_1$ and $C_2 \neq V_2$
Records from Today with $C_1 \neq V_1$ and $C_2 \neq V_2$	Records from Other with $C_1 \neq V_1$ and $C_2 \neq V_2$

Table 2: 2x2 Contingency Tables for a Two Component Rule

Once the scores for both tables are determined, we need to determine if they are significant or not. A score is considered significant if the result of a hypothesis test is significant at the $\alpha = 0.05$ level. If the scores for the two tables are both significant, then the presence of both components has an effect. As a result, the best rule overall for day i is $BR2_i$. On the other hand, if any one of the scores is not significant, then the best rule overall for day i is $BR1_i$.

2.1.4) Finding the p-value for a Rule

The algorithm above for determining scores is prone to over fitting due to multiple hypothesis testing. Even if data were generated randomly, most single rules would have insignificant p-values but the best rule would be significant if we had searched over 1000 possible rules. In order to illustrate this point, suppose we follow the standard practice of rejecting the null hypothesis when the p-value is $< \alpha$, where $\alpha = 0.05$. In the case of a single hypothesis test, the probability of a false positive under the null hypothesis would be α , which equals 0.05. On the other hand, if we perform 1000 hypothesis tests, one for each possible rule under consideration, then the probability of a false positive could be as bad as $1 - (1 - 0.05)^{1000} \approx 1$, which is much greater than 0.05 (Miller et al., 2001). Thus, if the algorithm returns a significant p-value, we cannot accept it at face value without adding an adjustment for the multiple hypothesis tests we performed.

The use of a randomization test, under the null hypothesis of this randomization test, the date and the other ED case attributes are assumed to be independent. Consequently, the case attributes in the data set DB_i remain the same for each record but the date field is shuffled between records from the current day and records from five to eight weeks ago. The P-value was calculated by the following method:

C-recent-> C-baseline	Home location= NW	Home location!= NW	<i>Row Total</i>
Home location= NW	<i>a</i>	<i>b</i>	<i>a + b</i>
Home location!= NW	<i>c</i>	<i>d</i>	<i>c + d</i>
<i>Column Total</i>	<i>a + c</i>	<i>b + d</i>	<i>a + b + c + d</i> <i>(=n)</i>

$$p = \frac{\binom{a+b}{a} \binom{c+d}{c}}{\binom{n}{a+c}} = \frac{(a+b)! (c+d)! (a+c)! (b+d)!}{a! b! c! d! n!}$$

2.1.5) WSARE 3.0

Detection algorithms operate by subtracting away the baseline from recent data and raising an alarm if the deviations from the baseline are significant. The challenge facing all such systems is to estimate the baseline distribution using data from historical data. In general, determining this distribution is extremely difficult due to the different trends present in surveillance data.

Seasonal variations in weather and temperature can dramatically alter the distribution of surveillance data. For example, flu season typically occurs during mid-winter, resulting in an increase in ED cases involving respiratory problems. Day of week variations make up another

period choosing the wrong baseline distribution can have dire consequences for an early detection system. The system would consider high counts of flu-like symptoms to be normal. If an anthrax attack occurs, it would be detected late, if at all.

2.1.6) Creating the Baseline Distribution

Learning the baseline distribution involves taking all records prior to the past 24 hours and building a Bayesian network from this subset. During the structure learning, we differentiate between environmental attributes, which are attributes that cause trends in the data, and *response attributes*, which are the remaining attributes. The environmental attributes are specified by the user based on the user's knowledge of the problem domain. If there are any latent environmental attribute that are not accounted for in this model, the detection algorithm may have some difficulties.

While learning the structure of the Bayesian network, environmental attributes are prevented from having parents because we are not interested in predicting their distributions, but rather, we want to use them to predict the distributions of the response attributes. In general, any structure learning algorithm can be used in this step as long as it follows this restriction. In fact, the structure search can even exploit this constraint by avoiding search paths that assign parents to the environmental attributes.

Optimal Reinsertion is a larger scale search operator that is much less prone to local optima. Optimal Reinsertion first picks a target node T from the DAG, disconnects T from the graph, and efficiently finds the optimal way to reinsert T back into the graph according to the scoring function.

Environmental attributes, however, can also include any source of information that accounts for recent changes in the data. For example, suppose we detect that a botulism outbreak has occurred and we would still like to be on alert for any anthrax releases. Incorporating such knowledge into the Bayesian network allows WSARE to treat events due to the botulism outbreak as part of the baseline.

Once the Bayesian network is learned, we have a joint probability distribution for the data. We would like to produce a conditional probability distribution, which is formed by conditioning on the values of the environmental attributes. Suppose that today is February 21, 2003. If the environmental attributes were *Season* and *Day of Week*, we would set *Season* = *Winter* and *Day of Week* = *Weekday*. Let the response attributes in this example be X_1, \dots, X_n . We can then obtain the probability distribution $P(X_1, \dots, X_n \mid \textit{Season} = \textit{Winter}, \textit{Day of Week} = \textit{Weekday})$ from the Bayesian network. For simplicity, we represent the conditional distribution as a data set formed by sampling a large number of records from the Bayesian network conditioned on the environmental attributes. The number of samples is specified by the parameter *num baseline samples*, which has to be large enough to ensure that samples with rare combinations of attributes will be present. In general, this number will depend on the learned Bayesian network's structure and the parameters of the network. We chose to sample 10000 records because we determined empirically that this number is a reasonable compromise between running time and accuracy on our data. We will refer to this sampled data set as $DB_{baseline}$. The data set corresponding to the records from the past 24 hours of the current day will be named DB_{recent} .

We used a sampled data set instead of using inference mainly for simplicity. Inference might be faster than sampling to obtain the conditional probability $P(X_1, \dots, X_n \mid \textit{Environmental Attributes})$, especially when the learned Bayesian networks are simple. However, if inference is used, it is somewhat unclear how to perform the randomization test. With sampling, on the other hand, we only need to generate $DB_{baseline}$ once and then we can use it for the randomization test to obtain the p-values for all the rules. In addition, sampling is easily done in an efficient manner since environmental attributes have no parents. While a sampled data set provides the simplest way of obtaining the conditional distribution.

2.2) Introduction to Bayesian Networks

- ▶ Bayesian networks are DAGs whose nodes represent random variables.
- ▶ Edges represent *conditional dependencies* (or can be treated as *causal*)
- ▶ Nodes that are not connected represent variables that are conditionally independent of each other.
- ▶ Each node has a probability function (assuming discrete setting).

Inference in Bayesian Networks:

- ▶ Causal
- ▶ Diagnostic

Why Bayesian Network?

The graphical model has several advantages for data analysis:

- ▶ A Bayesian network can be used to learn causal relationships, and hence can be used to gain understanding about a problem domain and to predict the consequences of intervention.
- ▶ As the model has both a causal and probabilistic semantics, it is an ideal representation for combining prior knowledge (which often comes in causal form) and data.

How Bayesian framework is useful in solving an epidemic problem?

- ▶ It helps in modeling the domain which is can prove useful in comparing the recent events (current observations) with the historical data (learnt Bayesian model) using Bayesian inference.

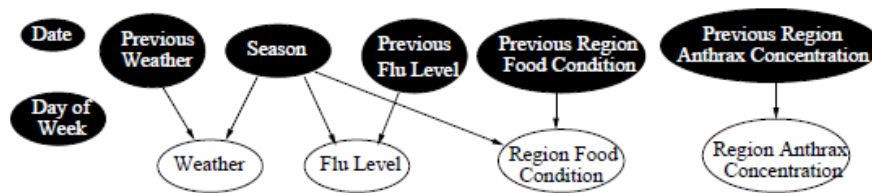


Figure 3: The city Bayesian Network used.

Chapter – 3
REQUIREMENT ELICITATION
AND ANALYSIS

3.1) NetBeans

NetBeans IDE is the official IDE for Java 8. With its editors, code analysers, and converters, you can quickly and smoothly upgrade your applications to use new Java 8 language constructs, such as lambdas, functional operations, and method references.

Batch analysers and converters are provided to search through multiple applications at the same time, matching patterns for conversion to new Java 8 language constructs. With its constantly improving Java Editor, many rich features and an extensive range of tools, templates and samples, NetBeans IDE sets the standard for developing with cutting edge technologies out of the box.

In this project Netbeans was used for all the coding part of one component rule as well of the two component rule.

3.2) Netica

Netica was useful in learning the desired Bayesian network with the help of the given data, the conditional probability tables were developed by Netica. Which were very useful in the later course. Netica is a powerful, easy-to-use, complete program for working with Bayesian networks and influence diagrams. It has an intuitive and smooth user interface for drawing the networks. Once a network is created, the knowledge it contains can be transferred to other networks by cutting and pasting, or saved in modular form by creating a library of nodes with disconnected links. Netica can use the networks to perform various kinds of inference using the fastest and most modern algorithms. Given a new case of which we have limited knowledge, Netica will find the appropriate values or probabilities for all the unknown variables. These values or probabilities may be displayed in a number of different ways, including bar graphs and meters. The case may conveniently be saved to a file, and later brought back into the network (or a

different network) for further querying, or to take into account new information about the case. Netica can use influence diagrams to find optimal decisions which maximize the expected values of specified variables. Netica can construct conditional plans, since decisions in the future can depend on observations yet to be made, and the timings and inter-relationships between decisions are considered. Netica can be used to transform a network in a number of ways. Variables that are no longer of interest may be removed without changing the overall relationships between the remaining variables (technically, the probabilities are "summed out" when we don't know the variable's value, and a more complex operation is used when we do). Probabilistic models may be explored by such operations as reversing individual links of the network, removing or adding causal influences, optimizing one decision at time, etc. These operations may be done with just a click of the mouse, which makes Netica very suitable for easy exploring, and for teaching belief network and influence diagram concepts.

3.3) B Course

B course is a web based data analysis tool for Bayesian modeling, in particular dependence and classification modeling. It can also be used as an interactive tutorial which provides you with data sets that have been prepared in advance. B-Course can be used as an analysis tool for any research where dependence or classification modeling based on data is of interest.

Chapter – 4

Bayesian Network Based Early Disease Outbreak Detection Software

4.1) Implementation of the Algorithm

In order to implement the algorithm a given data set was used. Each line in a data set represents a health care record where a person in the simulation city did one of three things:

Visit an ED department, purchase medication, or was absent from school/work.

The fields in the data set can take on the following values:

1. XY: The region the case comes from (NW, N, NE, W, C, E, SW, S, SE)
2. age: child, working or senior
3. gender: male or female
4. flu: none, low, high or decline
5. day_of_week: sat, sun or weekday
6. weather: hot or cold
7. season: winter, spring, summer or fall
8. action: purchase, evisit or absent
9. reported_symptom: none, respiratory, nausea or rash
10. drug: none, nyquil, aspirin or vomit-b-gone
11. date: The date on which he/she visited the ED department.
12. anthrax: Whether anthrax broke on that particular date or not.

	A	B	C	D	E	F	G	H	I	J	K	L
1	XY	age	gender	flu	day_of_w	weather	season	action	reported_symptom	drug	date	anthrax
2												
3	NW	child	male	high	weekday	cold	winter	absent	nausea	none	JAN_01_2002	no
4	NW	child	female	high	weekday	cold	winter	evisit	nausea	none	JAN_01_2002	no
5	NW	working	female	high	weekday	cold	winter	evisit	nausea	none	JAN_01_2002	no
6	N	working	female	high	weekday	cold	winter	absent	nausea	nyquil	JAN_01_2002	no
7	N	senior	male	high	weekday	cold	winter	absent	respiratory	none	JAN_01_2002	no
8	N	senior	male	high	weekday	cold	winter	purchase	none	nyquil	JAN_01_2002	no
9	N	senior	male	high	weekday	cold	winter	evisit	respiratory	none	JAN_01_2002	no
10	N	working	female	high	weekday	cold	winter	evisit	respiratory	none	JAN_01_2002	no
11	N	child	female	high	weekday	cold	winter	absent	respiratory	aspirin	JAN_01_2002	no
12	NE	child	female	high	weekday	cold	winter	purchase	none	nyquil	JAN_01_2002	no
13	NE	working	female	high	weekday	cold	winter	evisit	respiratory	none	JAN_01_2002	no
14	NE	working	female	high	weekday	cold	winter	purchase	none	aspirin	JAN_01_2002	no
15	NE	working	female	high	weekday	cold	winter	absent	none	nyquil	JAN_01_2002	no
16	NE	working	female	high	weekday	cold	winter	absent	respiratory	none	JAN_01_2002	no
17	NE	child	female	high	weekday	cold	winter	purchase	none	aspirin	JAN_01_2002	no
18	NE	working	female	high	weekday	cold	winter	purchase	none	vomit_b_gone	JAN_01_2002	no
19	NE	working	male	high	weekday	cold	winter	evisit	respiratory	none	JAN_01_2002	no
20	NE	child	female	high	weekday	cold	winter	evisit	none	none	JAN_01_2002	no
21	NE	senior	female	high	weekday	cold	winter	purchase	none	vomit_b_gone	JAN_01_2002	no
22	NE	working	female	high	weekday	cold	winter	purchase	none	nyquil	JAN_01_2002	no
23	NE	child	male	high	weekday	cold	winter	purchase	none	nyquil	JAN_01_2002	no
24	W	senior	female	high	weekday	cold	winter	purchase	none	nyquil	JAN_01_2002	no
25	C	working	male	high	weekday	cold	winter	evisit	respiratory	none	JAN_01_2002	no

Figure 4: The data set used.


```
read the cell (c,i)

if cell.getContents.equals(str)
```

```
    increment a by 1
```

```
else
```

```
    increment b by 1
```

```
calculate pval
```

```
p=calp()
```

```
display pval, a, b, c, d
```

```
function calp(n)
```

```
    if(n<0)
```

```
        display error
```

```
    else
```

```
        for c=1 to n
```

```
            fact= fact+ log©
```

```
            k=(long) exp(fact)
```

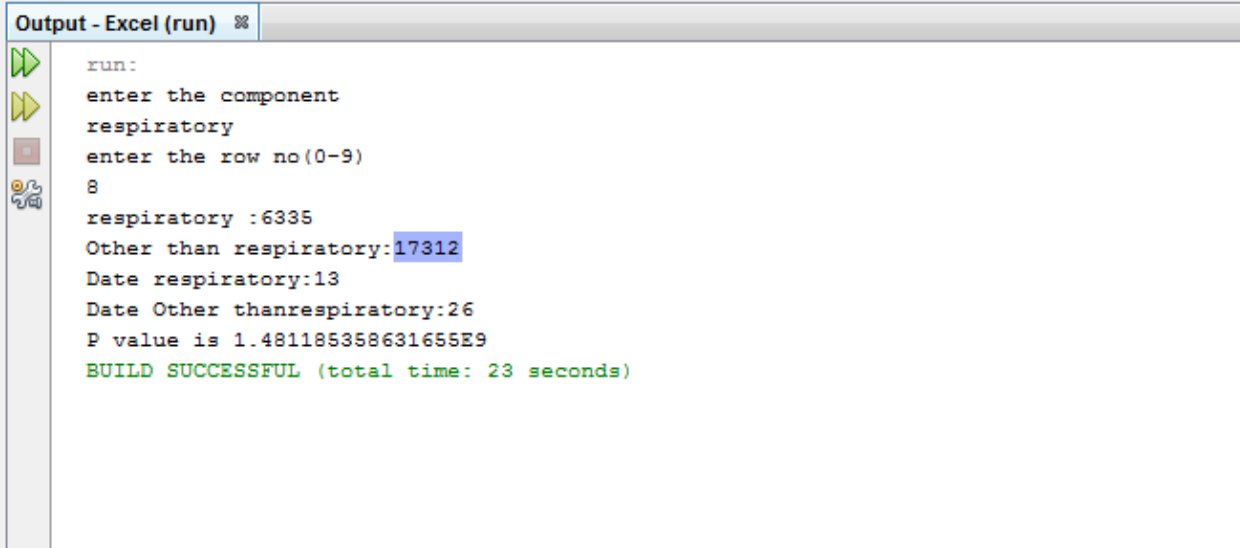
```
return k.
```

Expected result:

	C_{recent}	$C_{baseline}$
<i>Reported symptom = respiratory</i>	13	6335
<i>Reported symptom! = respiratory</i>	26	17312

Table 3: A expected 2x2 Contingency Table.

Real time result:



```
run:
enter the component
respiratory
enter the row no(0-9)
8
respiratory :6335
Other than respiratory:17312
Date respiratory:13
Date Other thanrespiratory:26
P value is 1.481185358631655E9
BUILD SUCCESSFUL (total time: 23 seconds)
```

Figure 5: One-component result

The given dataset was used in case of two component rule. The day in which anthrax=yes was treated as the current day C_{recent} and the remaining dataset was used as $C_{baseline}$ the algorithm was ran.

Calculating one-component rule:

Algorithm:

function read

initialise the current date as string s

initialise string 'bb' to all values of data

for i=0 to 10

 for a=0 to 9

 for j=i to 10

 for p=0 to 9

 if bb[i][a].equals '0' or bb[i][p].equals '0'

 break

 else

 increment flag by 1

 ss[flag][k]=bb[i][p]

 ss[flag][k1]=bb[i][j]

 for i=0 to sheet.getcol()

 for a=2 to sheet.getrow()

 for j=i+1 to sheet.getcol()

 for p=0 to sheet.getrow()


```
cell=sheet.getcell(i,0)
```

```
cell1=sheet.getcell(j,p)
```

```
cell2=sheet.getcell(s,0)
```

```
q=cell.getcontents()
```

```
qw=cell.setcontents()
```

```
for m=0 to flag+1
```

```
if(q equals (ss[a][0]) and qw(ss[a][1])
```

```
if(cell.getcontent equals (s))
```

```
increment a by 1
```

```
increment b[x] by 1
```

```
else
```

```
increment c[x] by 1
```

```
increment d[x] by 1
```

```
end
```

```
for m=0 to flag+1
```

```
calculate p[x] and cell
```

```
find max of p[]
```

```
for m=0 to flag+1
```

```
display ss[m][a] + ss[a][0]
```

```

display a[m]+b[m]+c[m]+d[m]+p[m]

calculate pval

p=calp()

display pval, a[], b[], c[], d[]

function calp(n)

    if(n<0)

        display error

    else

        for c=1 to n

            fact= fact+ log©

            k=(long) exp(fact)

return k.

end.

```

Expected result:

	C_{recent}	$C_{baseline}$
<i>Reported symptom = respiratory XY=NW</i>	4	335
<i>Reported symptom! = respiratory XY!=NW</i>	44	1254

Table 4: A expected 2x2 Contingency Table

Real time result:

```
NW respiratory  
a =0  
b =0  
c =0  
d =0  
p =1.0
```

Figure 6: Two-component result

The real time results were not same as the expected the expected results in the case of two-component rule.

4.1.2) WSARE (3.0)

In order to implement WSARE (3.0) the above dataset was used and the Bayesian network was learned using B Course without the condition probability table. Netica was used to learn the same Bayesian network with the conditional probability tables. The data was used to learn these Bayesian networks.

Bayesian Network without the probability tables :

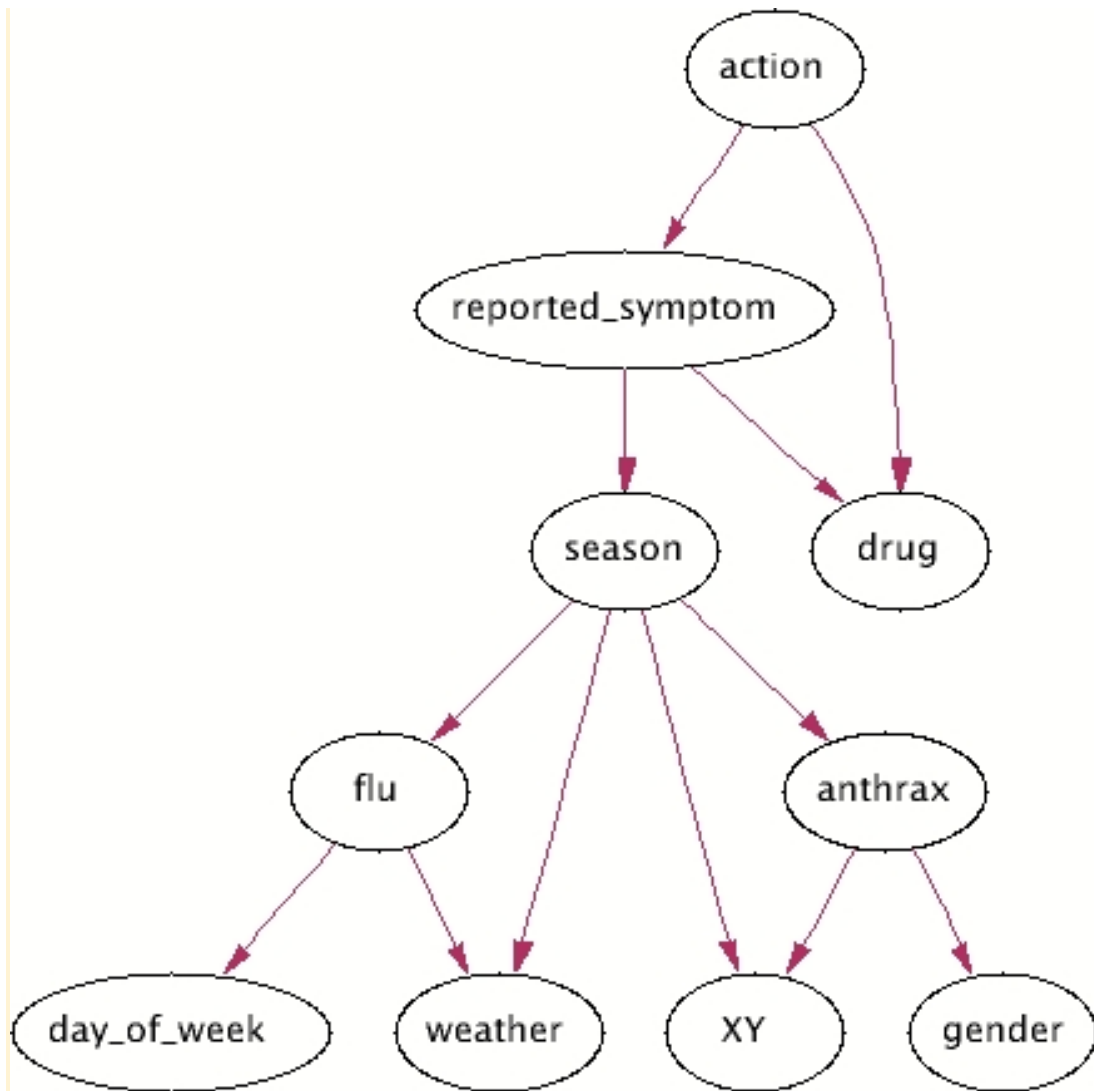


Figure 7: Bayesian Network using B course

Bayesian Network with the probability tables:

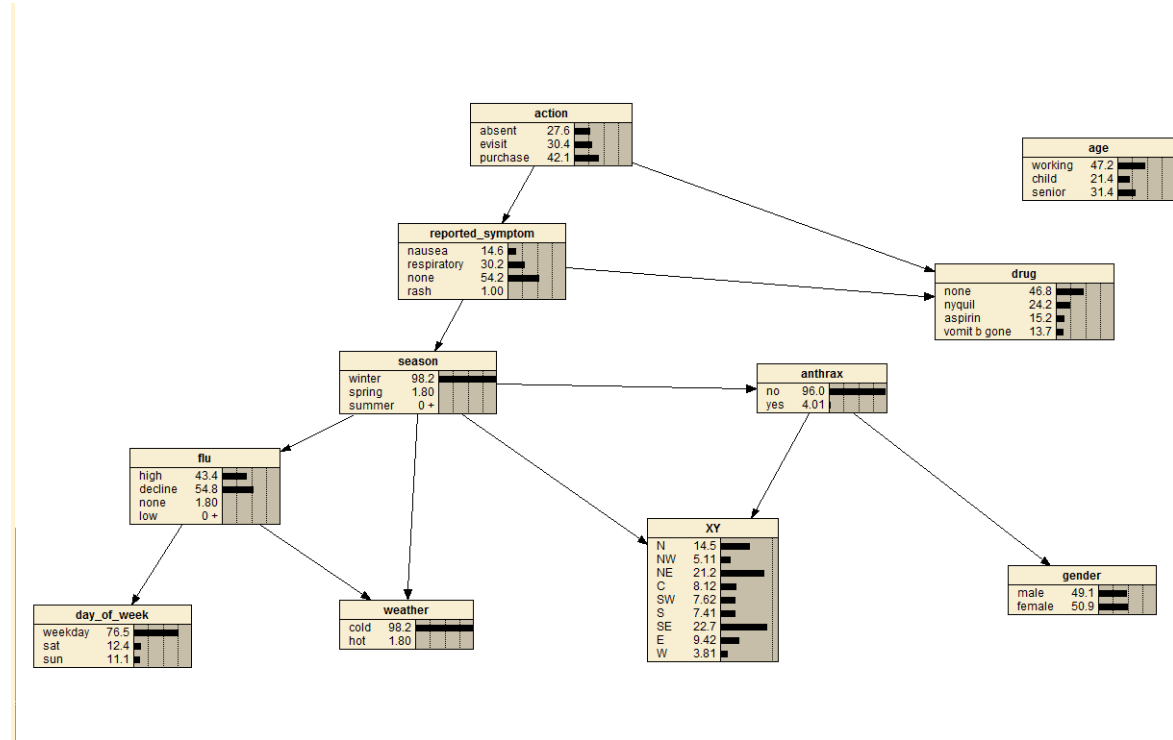


Figure 8: Bayesian Network using Netica.

Using Netica the following data set was found which was used as the baseline data for further calculations:

NumCases	A	XY
25	t	t
25	t	f
25	f	t
25	f	f

Chapter – 5

Coding

5.1) Code to calculate the p-value and one component rule:

```
package excel;

import java.io.File;

import java.io.IOException;

import static java.lang.Math.exp;

import static java.lang.Math.log;

import java.util.Scanner;

import jxl.Cell;

import jxl.CellType;

import jxl.Sheet;

import jxl.Workbook;

import jxl.read.biff.BiffException;

public class one_component

{

    private String inputFile;

    public void setInputFile(String inputFile)

    {

        this.inputFile = inputFile;

    }

    public void read() throws IOException

    {

        String s = "JAN-01-2002";

        String str;
```

```

Scanner key=new Scanner(System.in);

System.out.println("enter the component");

str=key.next();

System.out.println("enter the row no(0-9)");

int e=key.nextInt();

int c = 0, r = 0, none = 0, nau = 0;

File inputWorkbook = new File(inputFile);

Workbook w;

try

{

    w = Workbook.getWorkbook(inputWorkbook);

    Sheet sheet = w.getSheet(0);

    //Total cases

    for (int i = 0; i < sheet.getRows(); i++)

    {

        Cell cell = sheet.getCell(e, i);

        CellType type = cell.getType();

        if (cell.getContents().equals(str))

        {

            r++;

        }

        else

        {

```



```

        none++;
    }
}

System.out.println(str+" :"+ r);

System.out.println("Other than "+str+" :"+ none);

//Date wise

for (int i = 0; i < sheet.getRows(); i++)
{
    Cell cell = sheet.getCell(e, i);
    Cell cell1 = sheet.getCell(10, i);
    if (cell1.getContents().equals(s))
    {
        CellType type = cell.getType();
        if (cell.getContents().equals(str))
        {
            c++;
        }
        else
        {
            nau++;
        }
    }
}
}

```

```

        System.out.println("Date "+str+":" + c);

        System.out.println("Date Other than"+str+": "+ nau);

    }

    catch (BiffException ef)

    {

        ef.printStackTrace();

    }

    double p;

    p = (cal(c + nau) * cal(r + none) * cal(c + r) * cal(nau + none)) / (cal(r) * cal(c) *
cal(none) * cal(nau));

    System.out.println("P value is " + p);

}

public double cal(double n)

{

    double c, fact = 0;

    long k = 0;

    if (n < 0)

    {

        System.out.println("Number should be non-negative.");

    }

    else if(n==0)

        k=1;

```

```
else  
  
{  
  
    for (c = 1; c <= n; c++)  
  
        {  
  
            fact += log(c);  
  
            k = (long) exp(fact);  
  
        }  
  
    }  
  
    return k;  
  
}
```

```
public static void main(String[] args) throws IOException  
  
{  
  
    one_component test = new one_component();  
  
    test.setInputFile("C:\\Users\\sony\\Documents\\NetBeansProjects\\ExcelI\\aa1.xls");  
  
    test.read();  
  
}  
  
}
```

5.2) Code to calculate the p-value and two component rule:

```
import java.io.File;

import java.io.IOException;

import static java.lang.Math.exp;

import static java.lang.Math.log;

import java.util.Scanner;

import jxl.Cell;

import jxl.CellType;

import jxl.Sheet;

import jxl.Workbook;

import default_package.*;

import jxl.read.biff.BiffException;

public class excelf extends NewClass

{

    private String inputFile;

    public void setInputFile(String inputFile)

    {

        this.inputFile = inputFile;

    }

    public void read() throws IOException

    {

        String s = "JAN-01-2002";
```

```

long flag2 = 0;

int a1=0,b1=0,c1=0,d1=0;

String ss[][] = new String[599][2];

int i = 0, j = 0, k = 0, a = 0, flag = 0,m=0;

String q, qw;

int aa[] = new int[599];

int b[] = new int[599];

int c[] = new int[599];

int d[] = new int[599];

String bb[][] = {

    {"NW", "N", "NE", "W", "C", "E", "SW", "S", "SE"},

    {"WORKING", "SENIOR", "CHILD", "0", "0", "0", "0", "0", "0"},

    {"male", "female", "0", "0", "0", "0", "0", "0", "0"},

    {"none", "high", "decline", "low", "0", "0", "0", "0", "0"},

    {"weekday", "sat", "sun", "0", "0", "0", "0", "0", "0"},

    {"cold", "hot", "0", "0", "0", "0", "0", "0", "0"},

    {"winter", "summer", "fall", "spring", "0", "0", "0", "0", "0"},

    {"evisit", "purchase", "absent", "0", "0", "0", "0", "0", "0"},

    {"none", "respiratory", "nausea", "0", "0", "0", "0", "0", "0"},

    {"none", "vomit-b-gone", "aspirin", "nyquill", "0", "0", "0", "0", "0"}

};

String str,str1;

Scanner key=new Scanner(System.in);

```

```

System.out.println("enter the first component");

str=key.next();

System.out.println("enter the row no(0-9)");

int e=key.nextInt();

System.out.println("enter the second component");

str1=key.next();

System.out.println("enter the row no(0-9)");

int e1=key.nextInt();

File inputWorkbook = new File(inputFile);

Workbook w;

for (int y = 0; y < 599; y++)

{

    aa[y] = 0;

    b[y] = 0;

    c[y] = 0;

    d[y] = 0;

}

//Cases from 2d Array

for (i = 0; i < 10; i++)

{

    for (a = 0; a < 9; a++)

    {

```

```

for (j = i + 1; j < 10; j++)
{
    for (k = 0; k < 9; k++)
    {
        if (bb[i][a].equals("0") || bb[j][k].equals("0"))
        {
            break;
        }
        else
        {
            flag++;
            ss[flag][0] = bb[i][a];
            ss[flag][1] = bb[j][k];
        }
    }
}

//Reading of excel file and comparing and calculating aa,b,c,d,p
try
{
    w = Workbook.getWorkbook(inputWorkbook);
    Sheet sheet = w.getSheet(0);

```

```

for (i = 0; i < sheet.getColumns(); i++)
{
    for (a = 2; a < sheet.getRows(); a++)
    {
        for (j = i + 1; j < sheet.getColumns(); j++)
        {
            for (k = 2; k < sheet.getRows(); k++)
            {
                Cell cell = sheet.getCell(i, a);
                Cell cell1 = sheet.getCell(j, k);
                Cell cell2 = sheet.getCell(10, a);
                //System.out.println(cell2.getContents());
                CellType type = cell.getType();
                CellType type1 = cell1.getType();
                q = cell.getContents();
                qw = cell1.getContents();
                //flag2++;
                //for (m = 0; m < 599; m++)
                //{
                    if ((q.equals(str)) && (qw.equals(str)))
                    {
                        System.out.println("kk");
                        if (cell2.getContents().equals(s))

```



```

}

double p[] = new double[599];

double pmax;

//Calculating p value

/*for(m=0;m<599;m++)

{

    p[m] =
(cal(aa[m]+b[m])*cal(c[m]+d[m])*cal(aa[m]+c[m])*cal(b[m]+d[m]))/(cal(aa[m])*cal(b[m])
*cal(c[m])*cal(d[m]));

}*/

//for(i=0;i<599;i++)

//p[i]=(Math.random())/18;

double p1;

p1 = (cal(a1+b1)*cal(c1+d1)*cal(a1+c1)*cal(b1+d1))/(cal(a1)*cal(b1)*cal(c1)*cal(d1));

/*pmax = p[0];

for (i=1;i < 599;i++)

{

    if (p[i] > pmax)

    {

        pmax = p[i];

        j = i;

    }

}*/

```

```

/*for (i = 0; i < 599; i++)
{
    System.out.println(ss[i][0] + " " + ss[i][1]);

    System.out.println("a =" + aa[i]);

    System.out.println("b =" + b[i]);

    System.out.println("c =" + c[i]);

    System.out.println("d =" + d[i]);

    System.out.println("p =" + p[i]);
}*/

System.out.println(str + " " + str1);

    System.out.println("a =" + a1);

    System.out.println("b =" + b1);

    System.out.println("c =" + c1);

    System.out.println("d =" + d1);

    System.out.println("p =" + p1);

//System.out.println("\nMax p value is =" + pmax1);

//System.out.println(ss[j][0]+" "+ss[j][1]);
}

//Calculating the factorial
public double cal(double n)
{
    double c, fact = 0;

    long k=0;

```

```

if (n<0)

    System.out.println("Number should be non-negative.");

else if(n==0)

    k=1;

else

{

    for ( c = 1 ; c <= n ; c++ )

    {

        fact+=log(c);

        k=(long) exp(fact);

    }

}

return k;

}

public static void main(String[] args) throws IOException

{

    excelf test = new excelf();

    test.setInputFile("C:\\Users\\sony\\Documents\\NetBeansProjects\\Final_Project\\ioi.xls");

    test.read();

}

}

```

Conclusion:

The aim of the project was to find an epidemic on a selected day with the help of the algorithm called What's Strange About Recent Events (WSARE). This algorithm used a multivariate approach to improve the timeliness of detection. WSARE employed a rule-based technique that compared recent health-care data against data from a baseline distribution and found the subgroups of the recent data which showed trends. WSARE approached this problem using a Bayesian network to produce a baseline distribution that accounts for these temporal trends.

References:

Paper:

What's Strange About Recent Events (WSARE): An Algorithm for the Early Detection of Disease Outbreaks.

Weng-Keen Wong WONG@EECS.OREGONSTATE.EDU

School of Electrical Engineering and Computer Science

Oregon State University

Corvallis, OR 97330, USA

Andrew Moore AWM@CS.CMU.EDU

School of Computer Science

Carnegie Mellon University

Pittsburgh, PA 15213, USA

Software used:

Netica: http://www.norsys.com/tutorials/netica/tut_refs.htm

B course: <http://b-course.cs.helsinki.fi/obc>

Baysian Network:

Introduction to Graphical Models : David Barber
(from the web)

