Time Series Data Mining Tool

Samir Sheriff, Satvik N, Vaishakh BN

May 20, 2013

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
Jnana Sangama, Belgaum - 590 014

2012-2013

Project Report on

Time Series Data Mining Tool

Submitted to RVCE (Autonomous institution affiliated to Visvesvaraya Technological University (VTU), Belgaum) in partial fulfillment of the requirements for the award of degree of

BACHELOR OF ENGINEERING
In
COMPUTER SCIENCE AND ENGINEERING
By

Samir Sheriff	**1RV09CS093**
Satvik N	**1RV09CS095**
Vaishakh BN	**1RV09CS114**

Under the Guidance of
Mrs. Shantha Rangaswamy
Assistant Professor
Computer Science and Engineering, RVCE

R.V. College of Engineering
(Autonomous institution affiliated to VTU)
Department of Computer Science and Engineering
Bangalore – 560059

DECLARATION

We, Samir Sheriff, Satvik N and Vaishakh B.N. bearing USN number 1RV09CS093 1RV09CS095 and 1RV09CS114 respectively, hereby declare that the project entitled "**Time Series Data Mining Tool**" completed and written by us, has not been previously formed the basis for the award of any degree or diploma or certificate of any other University.

Bangalore

Samir
Sheriff
1RV09CS093

Satvik N
1RV09CS095

Vaishakh B N
1RV09CS114

R V COLLEGE OF ENGINEERING

(Autonomous Institute Affiliated to VTU)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled, "**Time Series Data Mining Tool**", which is being submitted herewith for the award of B.E is the result of the work completed by **Samir Sheriff, Satvik N, Vaishakh B N** under my supervision and guidance.

Signature of Guide
(Mrs. Shanta R)

Signature of Head of Department
(Dr. N K Srinath)

Signature of the Principal
(Dr. B.S. Satyanarayana)

Name of Examiner	Signature of Examiner
1)	
2)	

ACKNOWLEDGEMENT

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped me in carrying out this project work. We would like to take this opportunity to thank them all.

First and foremost we would like to thank **Dr. B. S. Satyanarayana**, Principal, R.V.C.E, Bengaluru, for his moral support towards completing my project work.

We deeply express my sincere gratitude to our guides **Mrs. Shanta Rangaswamy**, Assistant Professor and **Dr. Shobha G**, Professor, Department of CSE, R.V.C.E, Bengaluru, for their able guidance, regular source of encouragement and assistance throughout this project.

We would like to thank **Dr. N. K. Srinath**, Head of Department, Computer Science & Engineering, R.V.C.E, Bengaluru, for his valuable suggestions and expert advice.

We thank our Parents, and all the Faculty members of Department of Computer Science & Engineering for their constant support and encouragement.

Last, but not the least, We would like to thank our peers and friends who provided us with valuable suggestions to improve our project.

Samir Sheriff
USN:1RV09CS093
Satvik N
USN:1RV09CS095
Vaishakh B N
USN:1RV09CS114

ABSTRACT

Data mining is the process of analyzing data from different perspectives and summarizing it into useful information. A time series is a sequence of data points, measured typically at successive points in time spaced at uniform time intervals. Time series analysis comprises methods for analyzing time series data in order to extract meaningful statistics and other characteristics of the data.In the context of statistics,the primary goal of time series analysis is forecasting. In the context of signal processing it is used for signal detection and estimation, while in the context of data mining, pattern recognition and machine learning time series analysis can be used for clustering, classification, query by content, anomaly detection as well as forecasting. This project is aimed making a time series data mining tool which can be used to accomplish the above goals.

This project mainly focuses on analyzing the sea and rainfall level time series. The data sets considered belong to the rainfall data collected over ten years in the six taluks of Chikkaballapura district of Karnataka. The tool developed can be used to perform anomaly detection, forecasting, similarity detection and temporal pattern detection. The performance of these algorithms were tested on the above data sets and the results are presented. The tool is developed using the model view control design pattern. The algorithms are coded using Java using the object oriented paradigm. A web based interface with chart visualisation is provided for the end user. This is done using Java Server pages and Servlets. The aid of Google Charts API is taken for plotting graphs. The tool used Git revision control system and Github for online collaboration and code hosting.

The algorithms implemented in this tool require a set of user-defined parameters that determine the accuracy of the results. The CUSUM and Statistical approach in the Anomaly-Detection module discover anomalies in the data sets. The Temporal Pattern Mining tool uses a fitness threshold set by the user and shows temporal patterns, and similarly, the Dynamic Time Warping tool in the Similarity Module shows similarities among the time series data sets. The Neural Network in the Forecasting module is the most accurate among the algorithms with 60% accuracy.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A time series is a set of observations X_t , each one being recorded at a specific time t. Discrete-time time series is one in which the set T of times at which observations are made is a discrete set. Continuous-time time series are obtained when observations are recorded continuously over some time interval, e.g., when T_0 belongs [0,1]. Examples of time series are the daily closing value of the stock market points and the annual flow volume of the Nile River at Aswan. Time series are very frequently plotted via line charts.[1,4]

Time series analysis comprises methods for analysing time series data in order to extract meaningful statistics and other characteristics of the data. Time series forecasting is the use of a model to predict future values based on previously observed values. [1]

1.1 Literature Survey

A time series is a collection of observations made sequentially through time. At each time point one or more measurements may be monitored corresponding to one or more attributes under consideration. The resulting time series is called univariate or multivariate respectively. In many cases the term sequence is used in order to refer to a time series, although some authors refer to this term only when the corresponding values are non-numerical.[1][2]

The most common tasks of time series data mining methods are: indexing, clustering, classification, novelty detection, motif discovery and rule discovery. In most of the cases, forecasting is based on the outcomes of the other tasks. A brief description of each task is given below.[3]

Indexing: Find the most similar time series in a database to a given query time series.
Clustering: Find groups of time series in a database such that, time series of the same group are similar to each other whereas time series from different groups are dissimilar to each other.
Classification: Assign a given time series to a predefined group in a way that is more similar to other time series of the same group than it is to time series from other groups.
Novelty detection: Find all sections of a time series that contain a different behavior than the expected with respect to some base model.
Motif discovery: Detect previously unknown repeated patterns in a time series database.
Rule discovery: Infer rules from one or more time series describing the most possible behavior that they might present at a specific time point (or interval).

The temporal aspect of data arises some special issues to be considered and imposes some restrictions in the corresponding applications[3]. First, it is necessary to define a similarity measure between two time series and this issue is very important in TSDM since it involves a degree of subjectivity that might affect the final result. Second, it is necessary to apply a representation scheme on the time series data. Since the amount of data may range from a few kilobytes to megabytes, an appropriate representation of the time series is necessary in order to manipulate and analyze it efficiently.[3,4] The desirable properties that this approach should hold are:

- the completeness of feature extraction
- the reduction of the dimensionality.

This project considers only single/uni-variate time series, dimentionality reduction is ignored.[5] In many cases also, the objective is to take advantage of the specific characteristics of a representation that make specific methods applicable (i.e. inducing rules, Markov models).

Novelty detection is a very important task in many areas. Several alternative terms for “noveltyḧave been used, such as, “anomaly,̇ “interestingness,̇ “surprising,̇ “faults.̇ Moreover, many problems of finding periodic patterns can be considered as similar problems.

1.2 Motivation

Time series data is a commonly found type of data in the nature. Everything changes over time and understanding why things change will help the mankind understand the nature in a better way and adapt according to the changes. Hence, discovering knowledge from time series and interpreting it is of utmost importance.

1.3 Problem Statement

A time series is a sequence of data points, measured typically at successive points in time spaced at uniform time intervals. Time series analysis comprises methods for analyzing time series data in order to extract meaningful statistics and other characteristics of the data.In the context of statistics,the primary goal of time series analysis is forecasting. In the context of data mining, pattern recognition and machine learning time series analysis can be used for clustering, classification, query by content, anomaly detection as well as forecasting.

To define the problem statement in one line, it is given a time series, design and develop a tool which can help in analyzing (forecasting/prediction, anomaly detection, temporal pattern detection, etc.) the time series and help the user visualize the results.

1.4 Objective

The ability to model and perform decision modelling and analysis is an essential feature of many real-world applications ranging from emergency medical treatment in intensive care units to military command and control systems. Existing formalisms and methods of inference have not been effective in real-time applications where trade-offs between decision quality and computational tractability are essential. The objective of this project is to fill the void that exists and help in proper analysis of time varying data.

1.5 Scope

The scope of a time series data mining tool is two fold. The first is to obtain an understanding of the underlying forces and structure that produced the observed data. The second is to fit a model and proceed to forecasting, monitoring or even feedback and feed forward control. The time series data mining tool can be used in the following fields:

- **Economic Forecasting**
- **Sales Forecasting**
- **Rainfall Analysis**
- **Stock Market Analysis**
- **Yield Projections**
- **Process and Quality Control**
- **Census Analysis**

1.6 Methodology

Time series analysis of data requires the user to able to view the different algorithms and the result obtained from each algorithm along with the graphs which help the user understand the time varying nature of the data. Hence, the representation of data becomes very important. Having understood this requirement in the early phase of the project, we adopted a methodology that will accomplish the objectives in a neat and intuitive way. A GUI was developed in the form of Java Server Pages and the back end was coded in Java which exploited the object oriented paradigm in designing the algorithms.

1.7 Organization of the report

Chapter 1 gave an introduction to the time series, the motivation, objective TSDM tool to be developed. Problem statement was discussed in brief.

Chapter 2 discusses the software requirements specifications considering product perspective, functional requirements, software and hardware requirements.

Chapter 3 discusses a high level design of the tool being developed. The data flow diagrams are discussed showing various levels (0, 1, 2).

Chapter 4 gives the detailed design of the tool, through Structured Charts that specify the high level design.

Chapter 5 gives the implementation details discussing the programming language selection, coding conventions used. A detailed description of the modules present in this project is presented under this chapter.

Chapter 6 discusses the testing strategies used, testing environment, and various test results are shown.

Chapter 7 explains the results of experiment and performance analysis of various modules of the TSDM tool. A comparative study is made for different algorithms under each module and inferences from the results are shown with a few snapshots for illustrations.

Chapter 8 summarizes the entire project, stating its limitations and puts forth the possible future enhancements for the TSDM Tool.

Chapter 2

Software Requirements Specifications

Software Requirement Specification (SRS) is an important part of software development process. It includes a set of use cases that describe all the interactions of the users with the software. Requirements analysis is critical to the success of a project.

2.1 Product Perspective

Time Series Data Mining tool is a unique product that makes use of different algorithms to predict, view similarities, and points out the anomalies in different time varying data sets. It is built in a pluggable fashion where the only requirement at the users end is the browser and a working internet connection.

2.2 Product Features

The time series data mining tool has many features that distinguishes it from the others available already in the open world. It provides accurate results using the similarity finding, anomaly finding algorithms. The back propagation neural network helps us predicting the future values. On the front end, the user has options to choose the algorithm of her/his choice. Also, the charts which depict the output are carefully plotted using the google charts API which has been made available by Google Inc. Also, Java beans along with servlets and java server pages and best practices of coding have been followed.

2.3 Constraints

During the development of this product, constraints were encountered. Some specific constraints under which the time series data mining tool has are :

- Memory consumption of the Tool on the server machine. Number of requests that can be handled by the server depends on the memory consumption of the tool.
- Internet speed, depending on the speed of the internet, data upload to the server, display of charts, loading of ajax, jquery apis are determined.

2.4 Assumptions and Dependencies

- It is assumed that the user of this tool has basic understanding of time series data mining.
- Also, the user must have a decent knowledge of the interpretation of line graphs.

2.5 Specific Requirements

This section shows the functional requirements that are to be satisfied by the system. All the requirements exposed here are essential to run this tool successfully.

2.5.1 Functional Requirements

The functionality requirements for a system describe the functionality or the services that the system is expected to provide. This depends on the type of software system being developed. The requirements that are needed for this project are :

- The data sets should be normalized so that the algorithms can be applied effectively.
- A good representation of the results should be made available to the users through proper representation media like graphs.

2.5.2 Software Requirements

Developerś Machine

- Operating System: Windows 7/8, Linux, Mac
- Software Tools : Java, JDK 7.0, Apache Tomcat Server version 7.0 Web Browser (Mozilla, IE8+, Chrome)
- IDE : Eclipse IDE for J2EE Developers
- API Libraries : JQuery UI and Ajax Libraries (Active Internet Connection)

End User Machine

- Java Enabled Browser
- Active Internet Connection

2.5.3 Hardware Requirements

- Processor: Intel Pentium 4 or higher version
- RAM: 512MB or more
- Hard disk: 5 GB

Software Requirements

The Java Runtime Environment (JRE) is required to run the software.

Chapter 3

High Level Design

The software development usually follows Software Development Life Cycle (SDLC). The second stage of SDLC is the design phase. The design stage involves two substages namely High level design and Detailed level design.

High level design gives an overview of how the system works and top level components comprising the system.

3.1 System Architecture

This section provides an overview of the functionality and the working of the time series data mining tool. The overall functionality of the application is divided into different modules in an efficient way. The system architecture is shown in Figure 3.1

3.2 Data Flow Diagrams

A DFD is a figure which shows the flow of data between the different processes and how the data is modified in each of the process. It is very important tool in software engineering that is used for studying the high level

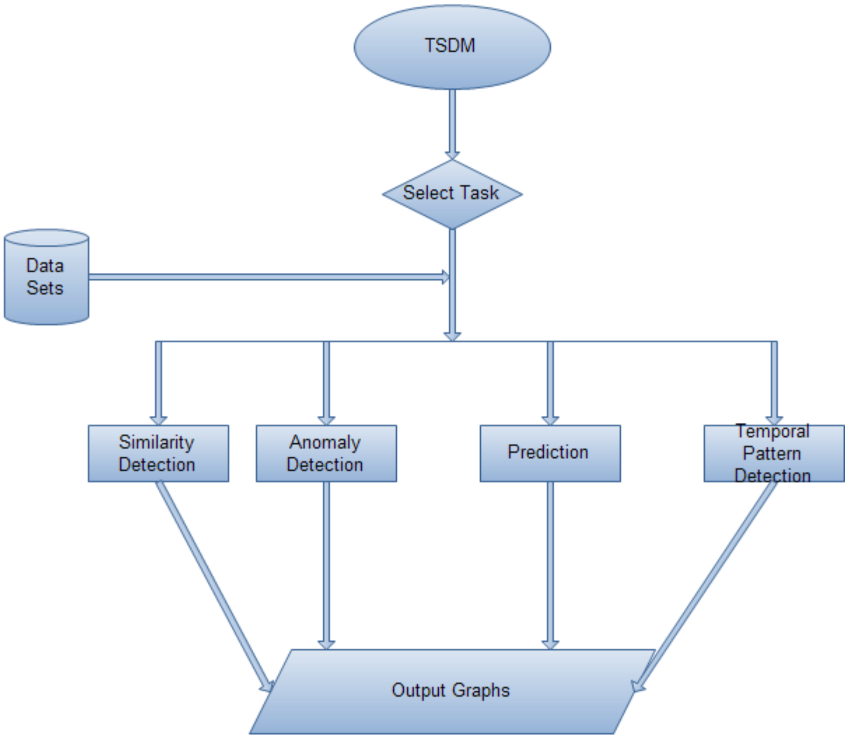


Figure 3.1: System Architecture

design.

There are many levels of DFDs. Level 0 gives the general description and level 1 gives the detailed description. Going higher in the level numbers greater description of the processes will be given.

3.2.1 DFD Level 0

The level 0 DFD is shown in Fig. 3.2 below which gives the general operation of the TSDM Tool. There are three major components. Two external entities called sender and receiver and the most important Time Series Data Mining Tool.

- Input Time Series Data : The user of this tool is the one responsible to send the data to be analyzed using the tool. This tool can analyze only uni-variate time series data.
- Analyzed Results: Depending the data sent by the user and the requested operations, the TSDM tool produces results in the form of graphs showing patterns, predictions, and other analyzed results.
- TSDM Tool : This the software tool which is used to analyze the input time series data and generate meaningful results, patterns etc. depending on what the user wants.

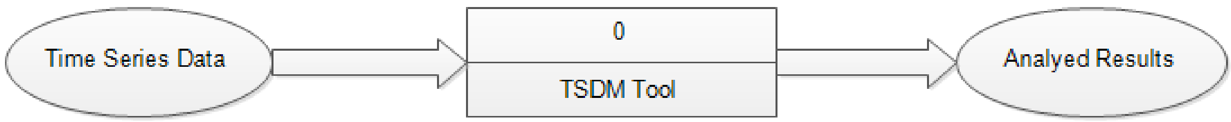


Figure 3.2: Data Flow Diagram Level 0

3.2.2 DFD Level 1

Major components of the TSDM Tool are shown in level 1 DFD as in Figure 3.3. The components are :
The description of the modules are as below :

- **Similarity Detection** : This module helps in finding similarity patterns (that occur at regular intervals in case of periodic time series), comparing different time series data. SAX and DTW are the main algorithms implemented/used in this module.
- **Forecasting and Prediction** : This module contain algorithms/models which can be trained from the past time series data and can be used to predict the future values of a time series.
- **Anomaly Detection** : This module contains algorithms that help in indicating anomalous patterns in the time series data analyzed. Anomalies are patterns in time series which deviate from the normal behavior and can indicate fraud/danger depending on the application. For example in an industry which produces the blades, the thickness of the blade can be monitored by a machine as a time series and any deviation from the normal error rate can signal an error in the manufacturing process.
- **Temporal Pattern Finder** : This module helps in finding hidden temporal patterns in a time series. This module can be further extended to implement clustering techniques.

Other components are

- **Input** - Input is the uni-variate time series data to be analyzed using the tool.
- **Output** - Output depends on the module selected by the user. This is visualized using the graphs/charts.

DFD Level 1.1

In DFD Level 1.1 shown in the figure 3.4, all the modules are expanded to show the algorithms implemented under them. The Input and the Output are same as level 1. Google charts are used for visualizing the results performed on the time series data. A brief description of the algorithms implemented under each module are below :

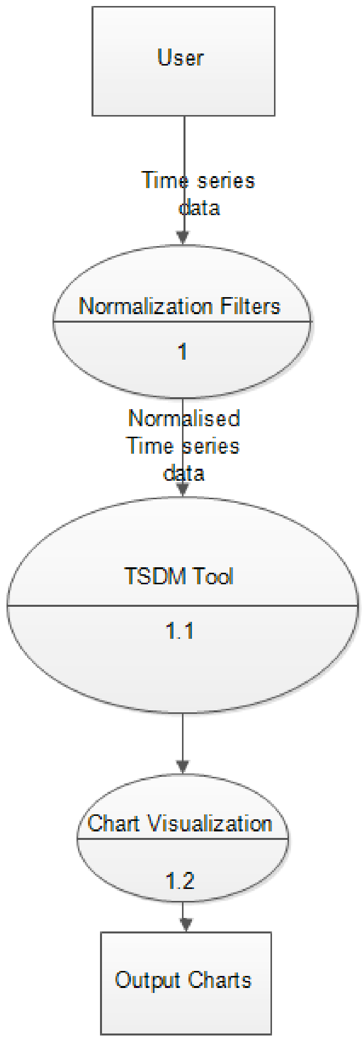


Figure 3.3: Data Flow Diagram Level 1

1. **Forecasting and Prediction**

   - NARX Neural Network : NARX is Non Linear Auto-Regressive with Extraneous Inputs. This is a neural network approach used for modeling a time series which depends on another time series. Back Propagation algorithm has been used to train the neural network. Sigmoid activation function is used for neurons.

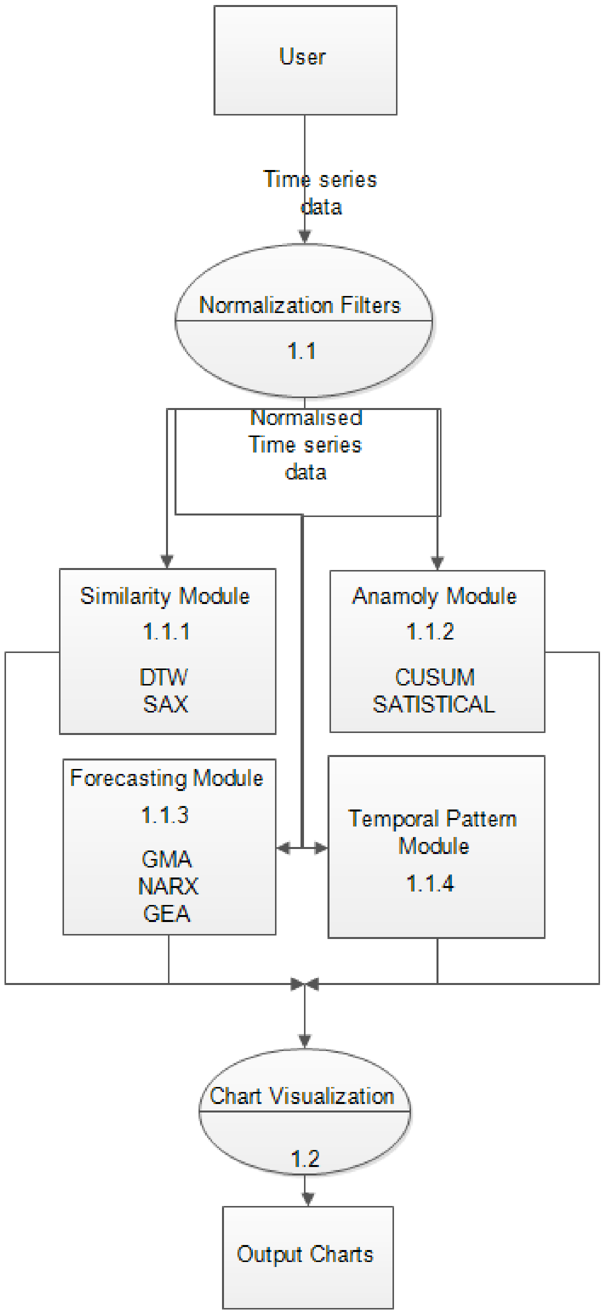


Figure 3.4: Data Flow Diagram Level 1.1

2. **Anomaly Detection**

   - Statistical Approach

- CUSUM Approach

These anomaly detection algorithms work in specific ways and help in determining the anomalous data points. The results are displayed in the output graphs.

3. **Similarity Detection**

   - DTW Algorithm : This algorithms helps in comparing two or more time series considering their euclidian distances and exhaustive comparing. This algorithm is computationally expensive but consumes less memory.
   - SAX Algorithm : This algorithm converts the time series to a string and helps in comparing different time series and shows how similar the time series are by giving a similarity distance.

4. **Temporal Pattern Finder** This module helps in determining interesting/temporal patterns in the time series data. The patterns are highlighted in the graphs.

3.2.3 DFD Level 2

The DFD Level 2 diagram is shown in the Figure 3.5 . The Output that comes out of the TSDM tool is divided into various categories depending on the module and is as shown in the figure 3.5. The output categories are :

- Comparision Results : The comparison results of the DTW or the SAX algorithm under the Similarity Detection Module are obtained and given as input to the charts api for visualization.

- Anomalous Data Points : The anomalous data points determined by the CUSUM or STATISTICAL algorithm under the Anomaly Detection Module are obtained and given as input to the charts application programmer interface for visualization.

- Forecast/Predicted Results : The predicted values of the input time series output by one of the NARX, GMA or GEA algorithms under the Forecasting Module are obtained and given as input to the charts api for visualization.

- Temporal Patterns : The temporal patterns found by the Temporal Pattern Finder module are obtained and given as input to the charts api for1 visualization.

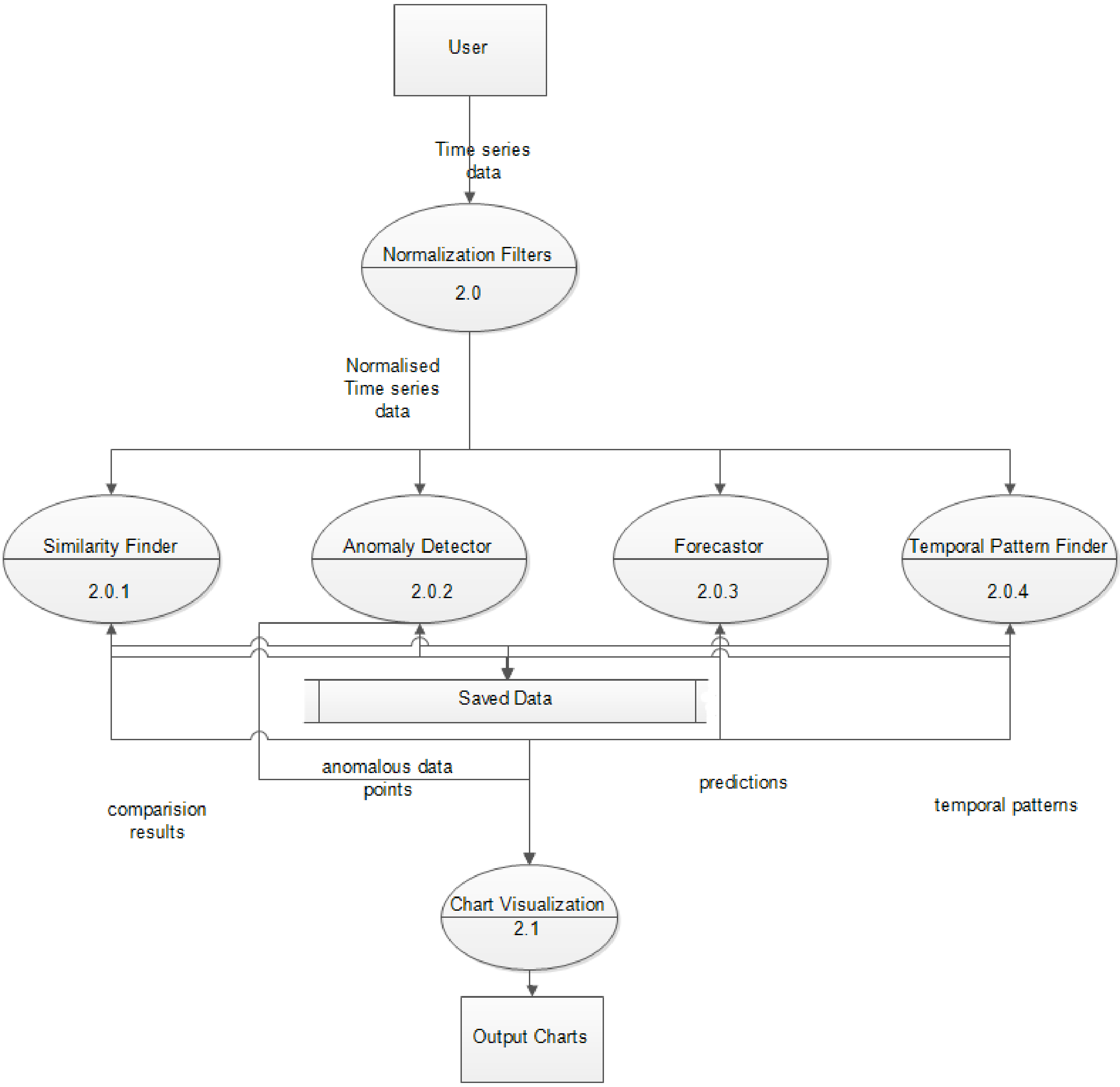


Figure 3.5: Data Flow Diagram Level 2

Chapter 4

Detailed Design

This chapter discusses the detailed design of the TSDM tool. In the section 4.1 the structured chart of the tool is explained.

In section 4.2 the details of the algorithms implemented under different modules of the tool are explained in detail.

4.1 Structured Chart

Structure charts are used to specify the high level design or architecture of a computer program. As a design tool, they help the programmer in dividing and conquering a large software problem, i.e. recursively breaking a problem down into parts that are small enough to be understood by a human brain. The process is called top-down design or functional decomposition.

Programmers use a structure chart to build a program in a manner similar to how an architect uses a blueprint to build a house. In the design stage, the chart is drawn and used as a method for the client and various software designers to communicate. During the actual building of the program, the chart is continuously referred to as master plan. Often, it is modified as programmers learn new details about the program. After a program is completed, the structured chart is used to fix bugs and to make changes.

The structured chart for the TSDM tool is shown in the figure 4.1 The Output that comes out of the TSDM tool is divided into various categories depending on the module and is as shown in the figure 4.1. The output categories are :

- Comparision Results : The comparison results of the DTW or the SAX algorithm under the Similarity Detection Module are obtained and given as input to the charts API for visualization.
- Anomalous Data Points : The anomalous data points determined by the CUSUM or STATISTICAL algorithm under the Anomaly Detection Module are obtained and given as input to the charts API for visualization.
- Forecast/Predicted Results : The predicted values of the input time series output by one of the NARX, GMA or GEA algorithms under the Forecasting Module are obtained and given as input to the charts API for visualization.
- Temporal Patterns : The temporal patterns found by the Temporal Pattern Finder module are obtained and given as input to the charts API for visualization.

4.2 Algorithm Details

In this section, the algorithms implemented in the TSDM Tool are explained in required detail.

As explained in the DFDs (figures 3.2 to 3.5), this tool mainly consists of four main modules. The modules are as below :

- Similarity Detection Module
- Forecasting and Prediction Module
- Anomaly Detection Module
- Temporal Pattern Finder

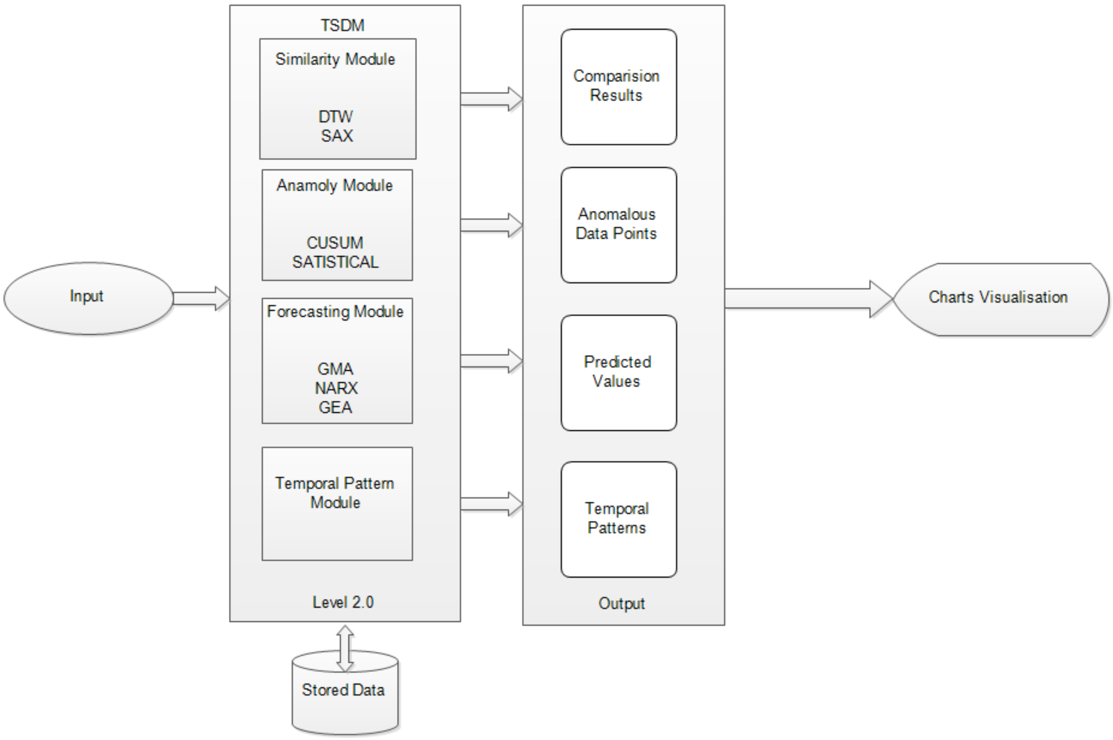


Figure 4.1: Structed Chart for the TSDM Tool

4.2.1 Similarity Detection Algorithms

Euclidian Distance

One of the simplest similarity measures for time series is the Euclidean distance measure.[1] Assume that both time sequences are of the same length n, this sequence can be viewed as a point in n-dimensional Euclidean space, and define the dissimilarity between sequences C and Q and D(C;Q) = $L_p(C;Q)$ i.e. the distance between the two points measured by the L_p norm (when p = 2, it reduces to the familiar Euclidean distance). Figure 4.2 shows a visual intuition behind the Euclidean distance metric.

Such a measure is simple to understand and easy to compute, which has ensured that the Euclidean distance is the most widely used distance measure for similarity search. However, one major disadvantage is that it is very brittle; it does not allow for a situation where two sequences are alike, but one has been “stretched” or “compressed” in the Y -axis.

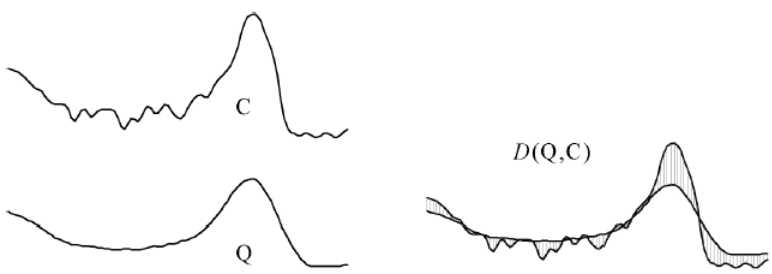


Figure 4.2: The intuition behind the Euclidean distance metric

DTW

Dynamic time warping (DTW) is a well-known technique to find an optimal alignment between two given (time-dependent) sequences under certain restrictions. Intuitively, the sequences are warped in a nonlinear fashion to match each other. Originally, DTW has been used to compare different speech patterns in automatic speech recognition. In fields such as data mining and information retrieval, DTW has been successfully applied to automatically cope with time deformations and different speeds associated with time-dependent data. The objective of DTW is to compare two (time-dependent) sequences

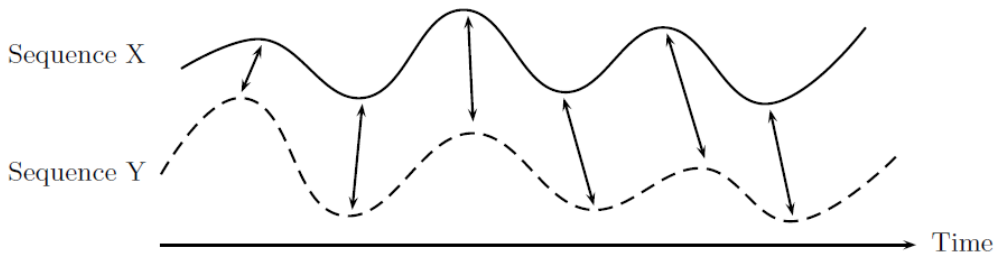


Figure 4.3: Time alignment of two time-dependent sequences.

$X := (x_1, x_2, ..., x_N)$ of length $N \in N$ and $Y := (y_1, y_2, ..., y_M)$ of length $M \in N$.
These sequences may be discrete signals (time-series) or, more generally, feature sequences sampled at equidistant points in time. Fix a feature space denoted by F. Then $x_n, y_m \in F$ for $n \in [1 : N]$ and $m \in [1 : M]$.
To compare two different features $x, y \in F$, one needs a local cost measure, sometimes also referred to as local distance measure, which is defined to be

a function c : $F \times F \rightarrow R$. Typically, $c(x, y)$ is small (low cost) if x and y are similar to each other, and otherwise $c(x, y)$ is large (high cost).

Evaluating the local cost measure for each pair of elements of the se-

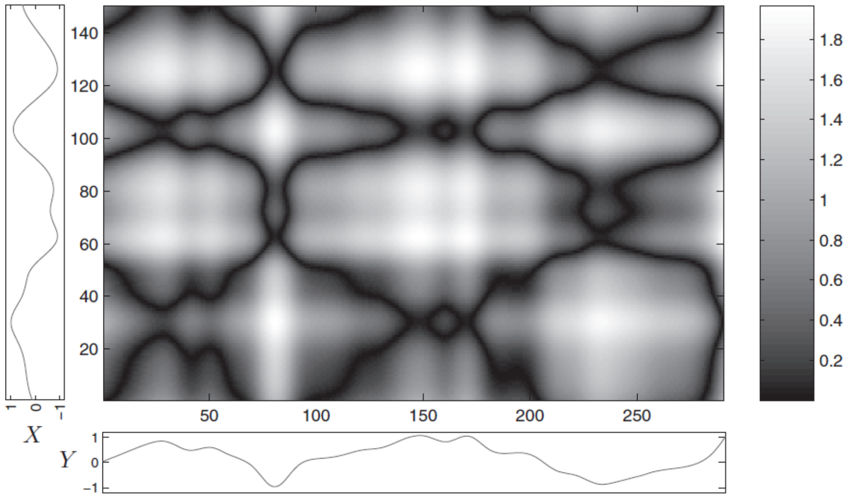


Figure 4.4: Cost matrix of the two real-valued sequences X (vertical axis) and Y (horizontal axis) using the Manhattan distance (absolute value of the difference) as local cost measure c. Regions of low cost are indicated by dark colors and regions of high cost are indicated by light colors

quences X and Y , one obtains the cost matrix $C \in R^{N \times M}$ defined by $C(n, m) := c(x_n, y_m)$. Then the goal is to find an alignment between X and Y having minimal overall cost. Intuitively, such an optimal alignment runs along a "valley" of low cost within the cost matrix C.

Basic Conditions to be satisfied: An (N,M)-warping path (or simply referred to as warping path if N and M are clear from the context) is a sequence $p = (p_1, ..., p_L)$ with $p_l = (n_l, m_l) \in [1 : N] \times [1 : M] forl \in [1 : L]$ satisfying the following three conditions.

1. Boundary condition: $p_1 = (1, 1)$ and $p_L = (N, M)$.
2. Monotonicity condition: $n_1 \leq n_2 \leq ... \leq n_L$ and $m_1 \leq m_2 \leq ... \leq m_L$.
3. Step size condition:$p_{l+1}p_l \in (1, 0), (0, 1), (1, 1)$ for $l \in [1 : L1]$.

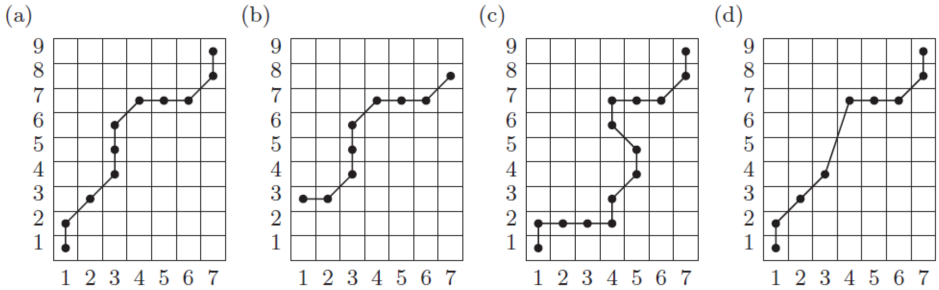


Figure 4.5: Illustration of paths of index pairs for some sequence X of length N = 9 and some sequence Y of length M = 7. (a) Admissible warping path satisfying the conditions (i), (ii), and (iii) of Definition 4.1. (b) Boundary condition (i) is violated. (c) Monotonicity condition (ii) is violated. (d) Step size condition (iii) is violated

SAX (Symbolic Aggregate approXimation) - Representation

Many high level representations of time series have been proposed for data mining, including Fourier transforms, wavelets, eigenwaves, piecewise polynomial models etc. While many symbolic representations of time series have been introduced over the past decades, they all suffer from two fatal flaws. Firstly, the dimensionality of the symbolic representation is the same as the original data, and virtually all data mining algorithms scale poorly with dimensionality. Secondly, although distance measures can be defined on the symbolic approaches, these distance measures have little correlation with distance measures defined on the original time series. SAX allows a time series of arbitrary length n to be reduced to a string of arbitrary length w, ($w<n$, typically $w<<n$). The alphabet size is also an arbitrary integer a, where a > 2. It uses an intermediate representation between the raw time series and the symbolic strings.

1. **Dimensionality Reduction Via PAA** - A time series C of length n can be represented in a w-dimensional space by a vector C,cw = 1 K . The i th element of C is calculated by the following equation:
   $\bar{c_i} = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} c_j$

2. **Obtaining Breakpoints** - Having transformed a time series database into the PAA we can apply a further transformation to obtain a dis-

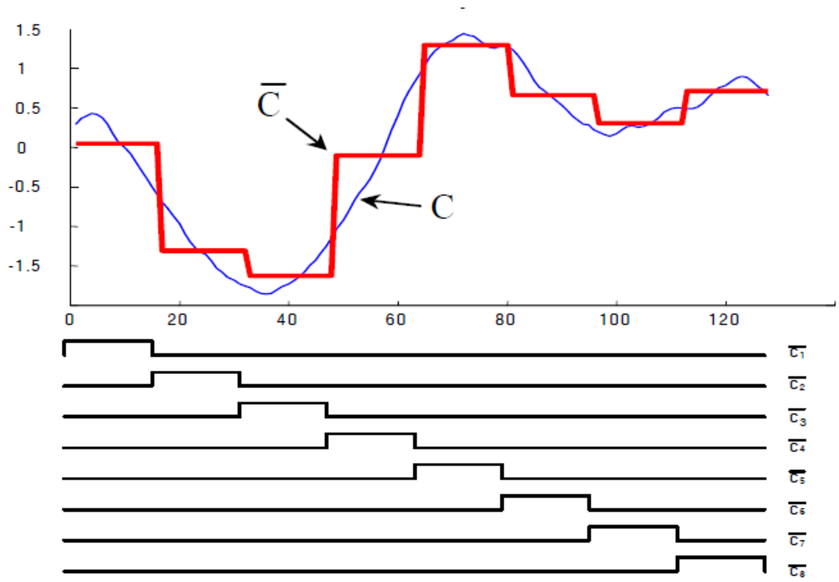


Figure 4.6: The PAA representation can be visualized as an attempt to model a time series with a linear combination of box basis functions. In this case, a sequence of length 128 is reduced to 8 dimensions

crete representation. It is desirable to have a discretization technique that will produce symbols with equiprobability. This is easily achieved since normalized time series have a Gaussian distribution.
We can simply determine the breakpoints that will produce a equal-sized areas under Gaussian curve. Breakpoints are a sorted list of numbers $= \beta_1, , \beta_{a-1}$ such that the area under a N(0,1) Gaussian curve from $\beta_i to \beta_{i+1} = 1/a(\beta_0$ and β_a are defined as $-\infty$ and ∞, respectively).
These breakpoints may be determined by looking them up in a statistical table shown in Figure 4.7

3. **Discretization** - Once the breakpoints have been obtained we can discretize a time series in the following manner. We first obtain a PAA of the time series. All PAA coefficients that are below the smallest breakpoint are mapped to the symbol a, all coefficients greater than or equal to the smallest breakpoint and less than the second smallest breakpoint are mapped to the symbol b, etc, as shown in Figure 4.8

4. **Distance Measure** - Having introduced the new representation of time series, we can now define a distance measure on it. The distance

β_i \ a	3	4	5	6	7	8	9	10
β_1	-0.43	-0.67	-0.84	-0.97	-1.07	-1.15	-1.22	-1.28
β_2	0.43	0	-0.25	-0.43	-0.57	-0.67	-0.76	-0.84
β_3		0.67	0.25	0	-0.18	-0.32	-0.43	-0.52
β_4			0.84	0.43	0.18	0	-0.14	-0.25
β_5				0.97	0.57	0.32	0.14	0
β_6					1.07	0.67	0.43	0.25
β_7						1.15	0.76	0.52
β_8							1.22	0.84
β_9								1.28

Figure 4.7: A lookup table that contains the breakpoints that divide a Gaussian distribution in an arbitrary number (from 3 to 10) of equiprobable regions

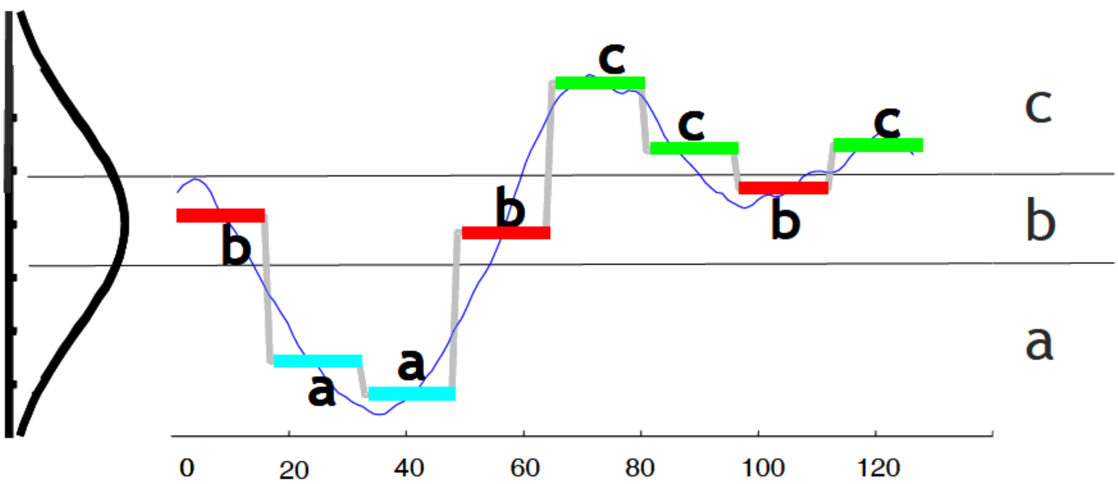


Figure 4.8: A time series is discretized by first obtaining a PAA approximation and then using predetermined breakpoints to map the PAA coefficients into SAX symbols.

between two SAX representations of a time series requires looking up the distances between each pair of symbols, squaring them, summing them, taking the square root and finally multiplying by the square root of the compression rate.

$MINDIST(\hat{Q},\hat{C}) \equiv \sqrt{\frac{n}{w}}\sqrt{\sum_{i=1}^{w}(dist(\hat{q}_i,\hat{c}_i))^2}$

The dist() function can be implemented using a table lookup, where the value in cell (r,c) for any lookup table can be calculated by the following expression.

$$cell_{r,c} = \begin{cases} 0, & if\ |r-c| \leq 1 \\ \beta_{\max(r,c)-1} - \beta_{\min(r,c)}, & otherwise \end{cases}$$

Figure 4.9: The value in cell (r,c) for any lookup table can be calculated by the above expression

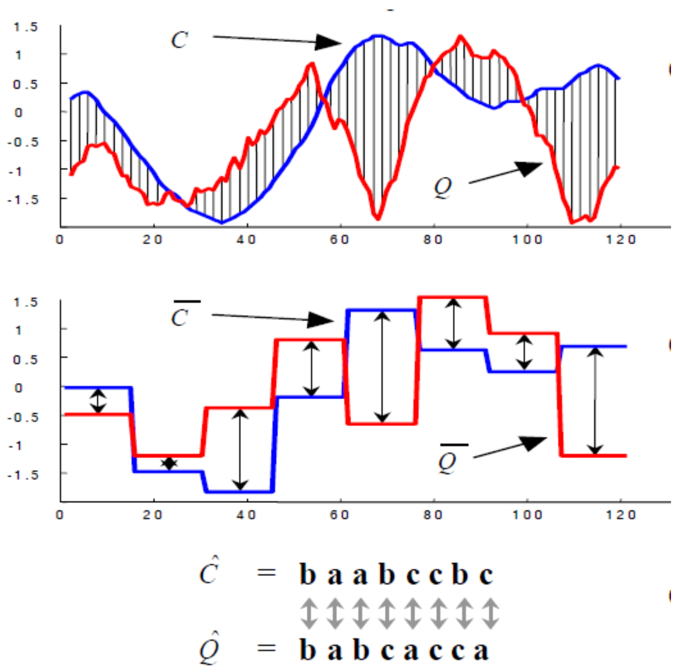


Figure 4.10: A visual intuition of the distance measures.

4.2.2 Forecasting and Prediction Algorithms

NARX Neural Network

The Nonlinear Autoregressive model with Exogenous inputs (NARX) [11,14] is an important class of discrete-time nonlinear systems that can be mathematically represented as

$$y(n+1) = f[y(n), ..., y(n-d_y+1); u(n), u(n-1), ..., u(n-d_u+1)]$$

where $u(n) \in R$ and $y(n) \in R$ denote, respectively, the input and output of the model at discrete time step n, while $du \geq 1$ and $dy \geq 1$, $du \leq dy$, are the input-memory and output-memory orders. In a compact vector form, the above equation can be written as

$$y(n+1) = f[y(n); u(n)];$$

where the vectors y(n) and u(n) denote the output and input regressors, respectively.

The nonlinear mapping f(.) is generally unknown and can be approximated, for example, by a standard multilayer Perceptron (MLP) network. The resulting connectionist architecture is then called a NARX network [11], a powerful class of dynamical models which has been shown to be computationally equivalent to Turing machines . In what concern training the NARX network, it can be carried out in one out of two modes: [15]

- *Series-Parallel (SP) Mode* - In this case, the output's regressor is formed only by actual values of the system's output:

$$y_b(n+1) = f_b[y_{sp}(n); u(n)];$$

- *Parallel (P) Mode* - In this case, estimated outputs are fed back and included in the output's regressor:

$$y_b(n+1) = f[y_p(n); u(n)];$$

$$y_b(n+1) = f[y(n), ..., y(n-d_y+1); u(n), u(n-1), ..., u(n-d_u+1);$$

In this tool, the SP mode of the NARX neural network is implemented. The implementation details are provided in the next chapter.

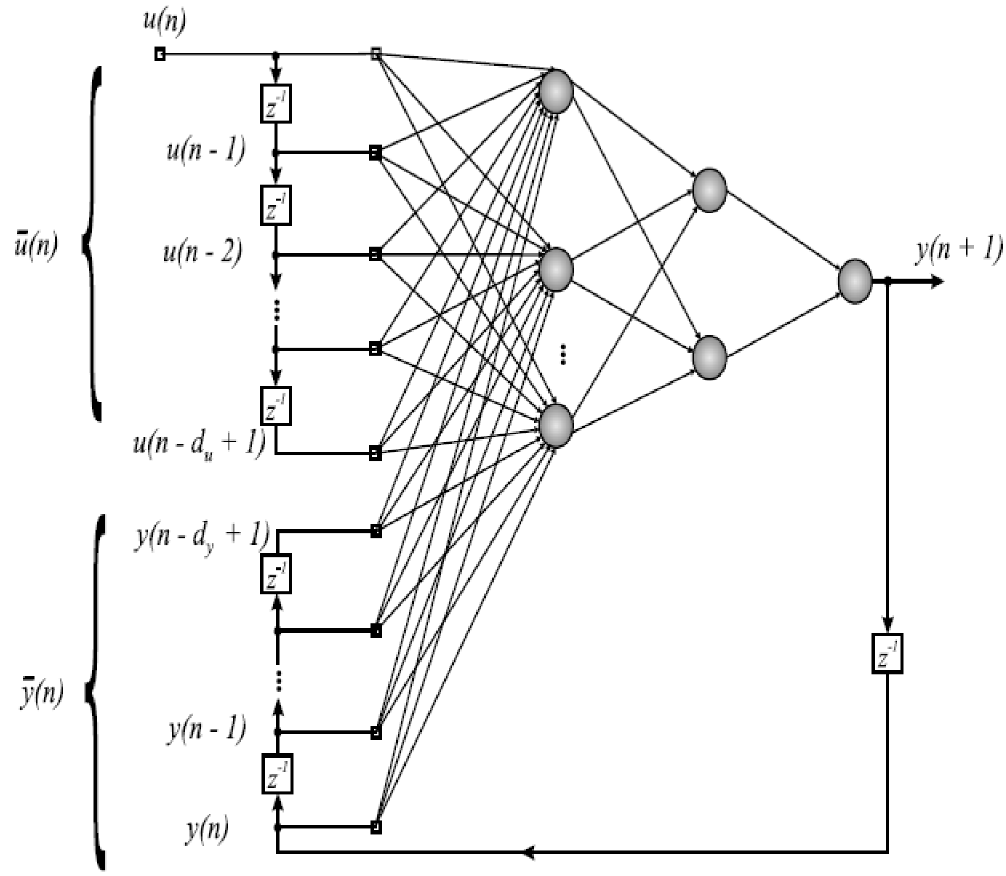


Figure 4.11: NARX network with d_u delayed inputs and $d_y$delayed outputs

Moving Exponential Average Method

Moving Exponential average, is a technique that can be applied to time series data to produce smoothed data for presentation, or to make forecasts. Exponential smoothing is commonly applied to discrete set of repeated measurements. This makes it a method feasible for usage in the TSDM tool. Exponential smoothing was first suggested by Robert Goodell Brown in 1956, the one used in this tool is known as " Brown's simple exponential smoothing".[14] The algorithm is a trivial and the forecasting accuracy is not as good as the Neural network approach.

Moving Geometric Average Method

A moving average of order k, MA(k) is the value of k consecutive observations. A moving geometric average is commonly used with time series data to smooth out short-term fluctuations and highlight longer-term trends or cycles. The method has disadvantages. One of the important disadvantage is that it is tough to extrapolate the values. This makes prediction inaccurate. [15]

4.2.3 Anomaly Detection Algorithms

Cumulative Sum (CUSUM) Approach

CUSUM algorithm works as follows [13]: A set of m samples, each of size n is collected, and compute the mean of each sample. Then the cumulative sum (CUSUM) control chart is formed by plotting the following quantity:

$$S_m = \sum_{i=1}^{m}(x_i - \mu_0)$$

against the sample number m, where is the estimate of the in-control mean and is the known (or estimated) standard deviation of the sample means. The choice of which of these two quantities is plotted is usually determined by the statistical software package. In either case, as long as the process remains in control centered at , the CUSUM plot will show variation in a random pattern centered about zero. If the process mean shifts upward, the charted CUSUM points will eventually drift upwards, and vice versa if the process mean decreases.[10]

A visual procedure proposed by Barnard in 1959, known as the V-Mask, is sometimes used to determine whether a process is out of control.

A V-Mask is an overlay shape in the form of a V on its side that is superimposed on the graph of the cumulative sums. The origin point of the V-Mask (see diagram below) is placed on top of the latest cumulative sum point and past points are examined to see if any fall above or below the sides of the V. As long as all the previous points lie between the sides of the V, the process is in control. Otherwise (even if one point lies outside) the process is suspected of being out of control.

In the figure above, the V-Mask shows an out of control situation because of the point that lies above the upper arm. By sliding the V-Mask backwards so that the origin point covers other cumulative sum data points, we can

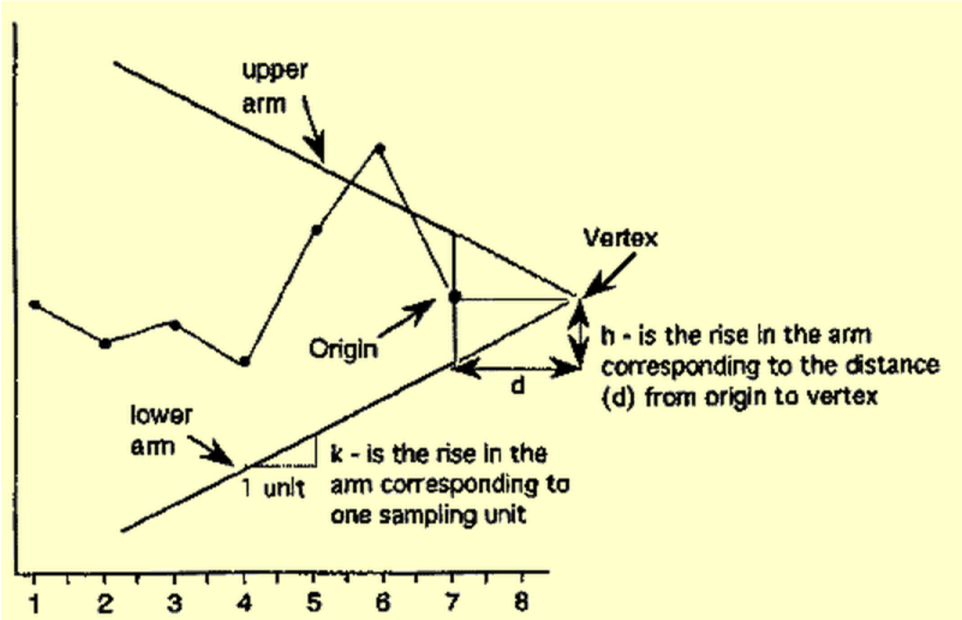


Figure 4.12: Sample V-Mask demonstrating an out of control process.

determine the first point that signaled an out-of-control situation. This is useful for diagnosing what might have caused the process to go out of control.

Statistical Approach

This approach is mainly dealt with by Statistical Quality Control process. In this approach the mean and standard deviation of the time series values of the process is recorded up to the the current time. If the time series value goes outside 3 sigmas (standard deviation) then an anomaly is signaled. [7]

This approach is explained in the figure 4.8.

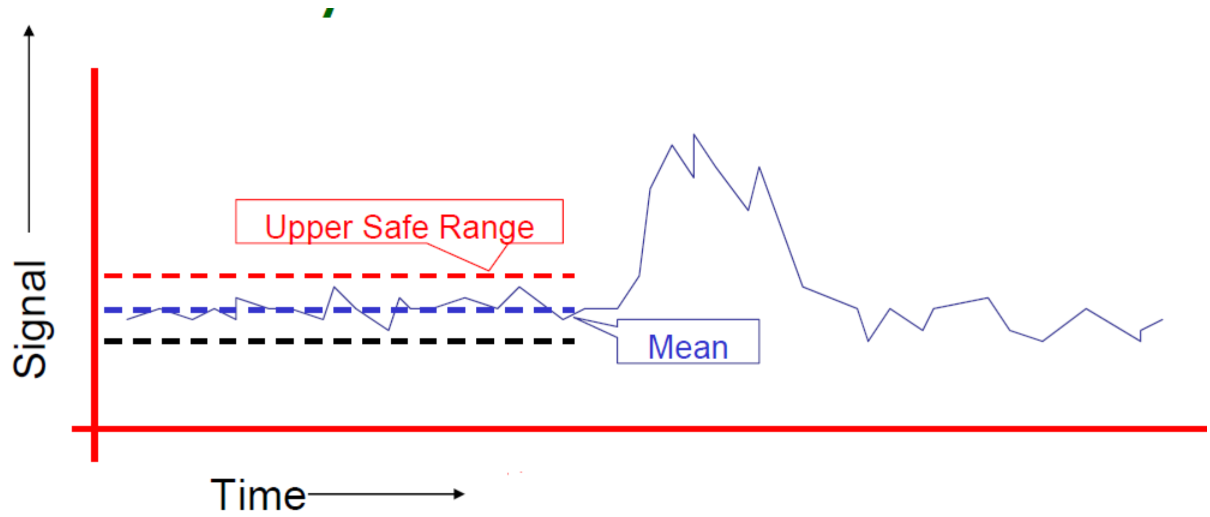


Figure 4.13: Statistical approach of anomaly detection.

Chapter 5

Implementation

The implementation phase of any project development is the most important phase and yields the final solution which solves the problem at hand.The implementation phase involves the actual materialization of the ideas, which are expressed in a suitable programming language. The factors concerning the programming language selection and platform chosen are described in the following sections.

5.1 Programming Language Selection

The programming language chosen must reflect the necessities of the project to be completely expressed in terms of the analysis and the design documents. Therefore before choosing the language, features to be included in the project are decided. The time series data mining project needs the following features in a language to be implemented. Some of the features required are stated as follows:

- J2EE provides us with servlets and JSP which help in dynamically constructing web pages.
- J2EE provides us with Java Beans which help in proper data manipulation.
- JSP and servlets make use of Java backend in a very optimal manner. They have special tags which help us exploit these features.
- Java's core classes are designed from scratch to meet the requirements of an object oriented system.

With these necessities in mind, J2EE is selected as the optimal programming language to implement the project.

5.2 Platform

The TSDM tool was built and designed on Windows Operating system family. They were specifically tested on Windows 7 with Google Chrome and Mozilla Firefox browsers. Because the product is browser based, any user with the browsers mentioned above will be able to run the tool. The product is hence platform independent in the true sense.

5.3 Code Conventions

The code standards for the Java programming Language document contains the standard conventions that follows. It includes file names, file organizations, indentation, comments, declarations, naming conventions and programming practices. Code conventions improve the readability of the software.

5.3.1 Naming Conventions

A naming convention is a set of rules for choosing the character sequence to be used for identifiers which denote variables, types, functions, and other entities in source code and documentation. There are several common elements that influence most if not all naming conventions in common use today. They are :

- Use mixed case to make names readable
- Avoid long names (15 characters maximum)
- Avoid names that are too similar or that differ only in case
- Capitalize the first letter of standard acronyms
- Use terminology applicable to the domain
- Use full descriptors that accurately describe the variable, field, or class

5.3.2 File Organization

As stated above, this project has been developed using the eclipse JEE IDE. This is a dynamic web project. The project has been organized as follows :

- **Java Resources** - This folder contains the java classes organized in packages. All the algorithms implemented in java are present in this folder under different packages.
- **JavaScript Resources** - This folder contains the javascript library files.
- **Web Content** - This folder mainly contains the jsp, js, html, css files which are used for developing the front webpages.

5.3.3 Comments

- **Implementation Comment Formats**
  - **Block Comments**

```
/*
 * Here is a block comment.
 */
```

  - **Single Line Comments**

```
if (condition) {

     /* Handle the condition. */
     ...
}
```

  - **Trailing Comments**

```
if (a == 2) {
     return TRUE;            /* special case */
} else {
```

```
    return isPrime(a);      /* works only for odd a */
}
```

– **End-of-Line Comments**

```
if (foo > 1) {

    // Do a double-flip. Harlem Style
     ...
}
else  {
     return false;          // Explain why here.
}
```

- **Documentation Comments**

```
/**
 * The Example class provides ...
 */
public class Example { ...
```

5.4 Difficulties encountered and Strategies used to tackle

There were a number of challenges that were faced while implementing the Time Series Data Mining tool.Some challenges were challenging and ended up in helping us think innovatively and come up with efficient solutions. Some major problems that were encountered have been stated in brief along with their solutions.

Problem 1

Initially the front end was designed in python using django web framework. But integrating java (back-end) with python had performance issues.(Using Jython interpreter)

Solution

JSP (Java server pages) was later used for the front end and this problem was solved.

Problem 2

In the initial stages of the project charts4j libraries were used to plot graphs. There were some internal problems with the URL rendering.

Solution

This Problem was solved later by making use of Google's Charts API and java script.

5.5 Module Design

Object-oriented programming (OOP) is a programming paradigm that represents concepts as "objects" that have data fields (attributes that describe the object) and associated procedures known as methods. Objects, which are instances of classes, are used to interact with one another to design applications and computer programs. In this chapter, we describe the different packages that were created by us to efficiently manage our code. Know first that our application consists of four diverse packages, namely:

1. *org.ck.sample*
2. *org.ck.smoother*
3. *org.ck.similarity*

4. *org.ck.forecaster.nn*
5. *org.ck.anomalifinder*
6. *org.ck.tsdm*
7. *org.ck.tsdm.ga*
8. *org.ck.beans*
9. *org.ck.servlets*
10. *org.ck.gui*

Each package is explained in detail, in the following sections. A bottom-up methodology is followed for explaining the layout of the classes.

5.5.1 org.ck.sample

This package allows us to efficiently manage and encapsulate the details of the data samples, provided by users, which are required for analysis. It is this class that allows our application to accept generic data sets. It consists of four classes:

1. **DataHolder** - This class keeps track of names of files that contain time series data; the fitness score threshold for the genetic algorithm. It provides this information, when required, to the front-end or back-end of our application. To make a long story short, this class acts like a middleman between the back-end and front-end of our application.

2. **Sample** - This class stores the values of a given time series in various forms - discrete and continuous; normalized and unnormalized; smooth and unsmooth; SAX Representation. It keeps track of all dimensions of a given time-series

5.5.2 org.ck.smoothers

This package consists of a number of smoothing filters that can be used to smoothen time series values stored in an object of the Sample class. These classes also double up as naive forecaster modules. They are based on the concept of Moving Averages.

1. **SmoothingFilter** - This is the parent class of all the other classes in this package. It is also an abstract class. Hence, all the other classes

that extend this class must provide implementations for the calculateSmoothedValues() and getAverage() methods compulsorily.

2. **ExponentialMovingAverageSmoother** - This class extends the SmoothingFilter class. Exponential smoothing is commonly applied to financial market and economic data, but it can be used with any discrete set of repeated measurements. The raw data sequence is often represented by x_t, and the output of the exponential smoothing algorithm is commonly written as s_t, which may be regarded as a best estimate of what the next value of x will be. When the sequence of observations begins at time t = 0, the simplest form of exponential smoothing is given by the formulae:

   $s_0 = x_0$ $s_t = \alpha x_t + (1 - \alpha) * s_{t-1}, t > 0$

   where α is the smoothing factor, and $0 < \alpha < 1$.

3. **SimpleMovingAverageSmoother** - This class extends the SmoothingFilter class. A simple moving average (SMA) is the unweighted mean of the previous n datum points.

   $SMA = (t_m + t_{m-1} + t_{m-2} + t_{m-3} + ... + t_{m-(n-1)}) \div n$

   where t_m is the value of the time series at the m^{th} instance of time.

4. **GeometricMovingAverageSmoother** - This class extends the SmoothingFilter class.The Geometric moving average calculates the geometric mean of the previous N bars of a time series.

5.5.3 org.ck.similarity

This module helps in finding similarity patterns (that occur at regular intervals in case of periodic time series), comparing different time series data. SAX and DTW are the main algorithms implemented/used in this module.

1. **DynamicTimeWarper** - Dynamic time warping (DTW) is an algorithm for measuring similarity between two sequences which may vary in time or speed. In general, DTW is a method that allows a computer to find an optimal match between two given sequences (e.g. time series) with certain restrictions. The sequences are "warped" non-linearly in the time dimension to determine a measure of their similarity independent of certain non-linear variations in the time dimension. This sequence alignment method is often used in time series classification. This class implements the aforementioned functionality.

2. **Discretizer** - This Singleton class is used to convert a time series represented by a PAA (Piecewise Aggregate Approximation), to a string representation (SAX - Symbolic Aggregate Approximation)

3. **Approximator** - This class averages out any time series containing continuous values using Piecewise Aggregate Approximation, allowing the time series to occupy as small a space as possible.

5.5.4 org.ck.forecaster.nn

This package contain algorithms/models which can be trained from the past time series data and can be used to predict the future values of a time series. In more practical terms neural networks are non-linear statistical data modeling or decision making tools. They can be used to model complex relationships between inputs and outputs or to find patterns in data.

1. **Neuron** - In a neural network model simple nodes (which can be called by a number of names, including "neurons", "neurodes", "Processing Elements" (PE) and "units"), are connected together to form a network of nodes hence the term "neural network".

2. **NetworkLayer** - Creating the neural network architecture therefore means coming up with values for the number of layers of each type and the number of nodes in each of these layers.

   - The Input Layer
   - The Hidden Layer
   - The Output Layer

3. **NeuralNetwork** - A neural network (NN), in the case of artificial neurons called artificial neural network (ANN) or simulated neural network (SNN), is an interconnected group of natural or artificial neurons that uses a mathematical or computational model for information processing based on a connectionistic approach to computation.

5.5.5 org.ck.anomalifinder

This package contains algorithms that help in indicating anomalous patterns in the time series data analyzed. Anomalies are patterns in time series which deviate from the normal behavior and can indicate fraud/danger depending on the application. For example in an industry which produces the blades, the thickness of the blade can be monitored by a machine as a time series and any deviation from the normal error rate can signal an error in the manufacturing process.

1. **Cusum_VmaskApproch**
2. **CusumAnomalyMethod**

5.5.6 org.ck.tsdm

This package contains a set of classes that reveal hidden patterns in time series data and overcome limitations of traditional time series analysis techniques. This Temporal Pattern Mining tool focuses on predicting events, which are important occurrences. This allows the TSDM methods to predict nonstationary, nonperiodic, irregular time series, including chaotic deterministic time series. It makes use of a genetic algorithm, internally.

1. **TSDM** - This class is to be used to find Temporal Patterns in Time Series and maintains status information about the algorithm.
2. **PhaseSpace** - Represents a Q-Dimensional Phase Space of points in the time series
3. **PhasePoint** - A point in a Q-Dimensional Phase Space

5.5.7 org.ck.tsdm.ga

This package takes care of all operations of the Genetic algorithm that is used by the TSDM class of the org.ck.tsdm package.

1. **Genome** - This class takes as input, a chromosome that encodes a given phase space cluster. It keeps track of this chromosome, and provides methods to manipulate this chromosome; to calculate the fitness score of this chromosome; and to throw an exception when the fitness value threshold has been crossed or when the best solution has been discovered.
2. **Population** - As defined earlier, a population is a collection of genomes. And this is exactly what this class is. Initially, the Population class

randomly initializes a large number of genomes, of which it keeps track. It provides methods such as roulette selection, reproduction, crossover, and mutation to operate on the population and discover the best genome, and hence, the best decision tree with the appropriate feature subset.

3. **OptimalScoreException** - This class is responsible for catching the best genome as soon as it is discovered, since the best genome should never be allowed to escape. It should be caught and nurtured for future use.

5.5.8 org.ck.beans

JavaBeans are reusable software components for Java. Practically, they are classes that encapsulate many objects into a single object (the bean). They are serializable, have a 0-argument constructor, and allow access to properties using getter and setter methods.

1. **TimeSeriesBean** - This bean stores information about requested values and results of calculations. The results produced by the Similarity, Forecaster, Anomaly Detection and Temporal Pattern Mining modules are stored in an object of this class, and the front-end reads the results from this bean. It is our very own custom class that allows any number of user-defined objects to be stored for communication.

5.5.9 org.ck.servlets

A Servlet is an object that receives a request and generates a response based on that request.

1. **MainController** - This servlet is the controller that gets requests from JSP pages, generates results through beans and forwards them to other jsp pages.

2. **AlgorithmUtils** - This class is a utility class that contains methods to run various algorithms in this tool.

5.5.10 org.ck.gui

This package allows each member of our team to test out the functionality of the back-end framework before connecting to the front-end to it.

1. **Constants** - A number of constants and enumerations used by all the other classes and packages.

2. **MainClass** - After a method is added to any class, the method is called from the respective team member's method to test it. If it works fine, then the tested method is used in the front-end.

Chapter 6

Software Testing

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

6.1 Types Of Testing

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provides a systematic demonstration that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items: Valid Input: identified classes of valid input must be accepted. Invalid Input: identified classes of invalid input must be rejected. Functions: identified functions must be exercised. Output: identified classes of application outputs must be exercised. Systems / Procedures: interfacing systems or procedures must be invoked. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identifying business process flows, data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

Performance Test

The Performance test ensures that the output be produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot see into it. The test provides inputs and responds to outputs without considering how the software works.

6.2 Test Environment

The testing of the modules was done on machine with the following features.

Operating System : Windows 7

JDK Version : 1.7

Browser : Google Chrome

Web server : Tomcat Version 7

Sl. Number of Test Case	1
Name Of the test	Connection between Tomcat and JDK.
Feature Being Tested	Tomcat Instance Working
Input	Tomcat Catalina Home batch file
Expected Output	Home Page Must Open
Output	Home Page Opens
Remarks	Test case passed

Table 6.1: Test Case 1 - System Test

Sl. Number of Test Case	2
Name Of the test	Working of JQuery and Ajax Libraries
Feature Being Tested	Presence of JQuery Libraries and Ajax Libraries
Input	Nil
Expected Output	Home Page Must Open with Jquery and Ajax Functionality
Output	Home Page Opens as expected
Remarks	Test case passed

Table 6.2: Test Case 2 - System Test

Sl. Number of Test Case	3
Name Of the test	Working of Google Charts API
Feature Being Tested	Charts
Input	Data Files
Expected Output	Scatter and Line Plots must appear
Output	Charts are plotted
Remarks	Test case passed

Table 6.3: Test Case 3 - System Test

Sl. Number of Test Case	4
Name Of the test	Network Testing
Feature Being Tested	Checking Network Connectivity
Input	IP address of any DNS server
Expected Output	Ping Data
Output	Ping Data Obtained
Remarks	Test case passed

Table 6.4: Test Case 4 - Integration Test

Sl. Number of Test Case	5
Name Of the test	Performance Testing
Feature Being Tested	Performance of various algorithms in the tool
Input	Time series data specified by the user
Expected Output	Load the graphs/results within the timeout specified.
Output	visualization graphs appear
Remarks	Test case passed

Table 6.5: Test Case 5 - Performance Test

Sl. Number of Test Case	6
Name Of the test	Load Testing
Feature Being Tested	Memory consumption of the tool per request
Input	Time series data and operations specified by the user
Expected Output	Process maximum requests from the users
Output	server computer handle multiple requests
Remarks	Test case passed

Table 6.6: Test Case 6 - Load Test

Sl. Number of Test Case	7
Name Of the test	DTW Algorithm Testing
Feature Being Tested	Working of the DTW Algorithm
Input	A null value time series is passed as a parameter
Expected Output	Similarity distance of -1
Output	A Java.NullPointer Exception is thrown
Remarks	Test case failed

Table 6.7: Test Case 7 - Similarity DTW Algorithm.

Sl. Number of Test Case	8
Name Of the test	DTW Algorithm Testing
Feature Being Tested	Working of the DTW Algorithm
Input	same Time series data for both parameters
Expected Output	Similarity distance of 0
Output	Similarity distance of 0
Remarks	Test case passed

Table 6.8: Test Case 8 - Similarity DTW Algorithm.

Sl. Number of Test Case	9
Name Of the test	SAX Algorithm Testing
Feature Being Tested	Working of the DTW Algorithm
Input	same Time series data for both parameters
Expected Output	Similarity distance of 0
Output	Similarity distance of 0
Remarks	Test case passed

Table 6.9: Test Case 9 - Similarity SAX Algorithm.

Sl. Number of Test Case	10
Name Of the test	Testing Temporal Pattern Detection Algorithm
Feature Being Tested	Temporal Pattern Detection Module
Input	Time series data with threshold value 5.0
Expected Output	Temporal patterns
Output	Improper patterns are shown
Remarks	Test case failed

Table 6.10: Test Case 10 : Testing Temporal Pattern Detection

Sl. Number of Test Case	11
Name Of the test	Testing Anomaly Detection Algorithm
Feature Being Tested	Statistical Approach
Input	Time series data with a low threshold value
Expected Output	Large number of anomalous data points
Output	Large number of anomalous points shown
Remarks	Test case passed

Table 6.11: Test Case 11 - Testing Anomaly Detection

Sl. Number of Test Case	12
Name Of the test	Testing Anomaly Detection Algorithm
Feature Being Tested	Statistical Approach
Input	Time series data with a higher threshold value
Expected Output	Number of anomalous data points decrease
Output	Number of anomalous points reduce
Remarks	Test case passed

Table 6.12: Test Case 12 - Testing Anomaly Detection

Sl. Number of Test Case	13
Name Of the test	SAX Algorithm Testing
Feature Being Tested	Working of the DTW Algorithm
Input	same Time series data for both parameters
Expected Output	Similarity distance of 0
Output	Similarity distance of 0
Remarks	Test case passed

Table 6.13: Test Case 13 - Testing Anomaly Detection.

Sl. Number of Test Case	14
Name Of the test	Testing Forecasting Module
Feature Being Tested	Working of the Moving Exponential Algorithm
Input	Time series and expected predictions as parameters
Expected Output	The predictions match as per the parameter sent
Output	predictions match
Remarks	Test case passed

Table 6.14: Test Case 14 - Forecasting Moving Exponential Average.

Sl. Number of Test Case	15
Name Of the test	Testing Forecasting Module
Feature Being Tested	Working of the Moving Exponential Algorithm
Input	An empty/null time series is passed
Expected Output	null value to be returned for predictions
Output	NullPointer Exception is thrown
Remarks	Test case failed

Table 6.15: Test Case 15 - Forecasting Moving Exponential Average.

Sl. Number of Test Case	16
Name Of the test	Testing Forecasting Module
Feature Being Tested	Working of the Moving Geometric Algorithm
Input	Time series and expected predictions as parameters
Expected Output	The predictions match as per the parameter sent
Output	predictions match
Remarks	Test case passed

Table 6.16: Test Case 16 - Forecasting Moving Geometric Average.

Sl. Number of Test Case	17
Name Of the test	Testing Forecasting Module
Feature Being Tested	Working of the Moving Geometric Algorithm
Input	An empty/null time series is passed
Expected Output	null value to be returned for predictions
Output	null value is returned
Remarks	Test case passed.

Table 6.17: Test Case 17 : Forecasting Moving Geometric Average.

Sl. Number of Test Case	18
Name Of the test	Testing Forecasting Module
Feature Being Tested	Working of the NARX Neural Network
Input	Time series data with learning rate 1.0
Expected Output	Prediction accuracy of 70%
Output	Prediction accuracy is 20%
Remarks	Test case failed.

Table 6.18: Test Case 18 - Forecasting with NARX Neural Network.

Chapter 7

Experimental Analysis and Results

In this chapter, all the performance and experimental analysis of the algorithms implemented under various different modules and the results obtained are presented.

7.1 Evaluation Metric

As explained earlier, major module present in the project are as follows :

- Similarity Detection Module
- Prediction and Forecasting Module
- Anomaly Detection Module
- Temporal Pattern Finder Module

Since these modules are independent of each other, they have different evaluation metrics. The evaluation metrics used for performance analysis of each module is explained in the following sub sections.

7.1.1 Metrics for Similarity Detection Module

The algorithms implemented under this module are :

- **Dynamic Time Wrapping (DTW) Algorithm**
- **SAX Algorithms**

The evaluation metrics are :

- **Euclidean Distance** In mathematics, the Euclidean distance or Euclidean metric is the "ordinary" distance between two points that one would measure with a ruler, and is given by the Pythagorean formula. By using this formula as distance, Euclidean space (or even any inner product space) becomes a metric space. The associated norm is called the Euclidean norm. Older literature refers to the metric as Pythagorean metric.

  In Cartesian coordinates, if p = $(p_1, p_2, ..., p_n)$ and q = $(q_1, q_2, ..., q_n)$ are two points in Euclidean n-space, then the distance from p to q, or from q to p is given by:

  $$d(p,q) = d(q,p) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2}$$

  $$d(p,q) = d(q,p) = \sqrt{\sum_{i=0}^{n} (p_i - q_i)^2}$$

  In the project, p and q can be visualized as two time series to be compared, where p_iś and q_iś are the time series values. Depending on the distance, the similarity of two or more time series with a give base series is found out. This metric is used in the DTW approach.

- **String Comparison** In SAX Algorithm, the time series is converted into a string of character as explained. Given two or more time series, which are represented by strings, a string comparison algorithm is run and the similarity is found out. There are various string comparison algorithms. In this project KMP algorithm has been used.

7.1.2 Metrics for Prediction and Forecasting Module

The algorithms implemented under this module for modeling and forecasting time series are :

- NARX-Neural Network
- Moving Average Forecaster

- Moving Geometric Average Forecaster
- Moving Exponential Average Forecaster

Modeling a time series is an regression problem, the evaluation metrics are :

- **Root-Mean-Square Deviation** - The root-mean-square deviation (RMSD) or root-mean-square error (RMSE) is a frequently used measure of the differences between values predicted by a model or an estimator and the values actually observed. These individual differences are called residuals when the calculations are performed over the data sample that was used for estimation, and are called prediction errors when computed out-of-sample. The RMSD serves to aggregate the magnitudes of the errors in predictions for various times into a single measure of predictive power. RMSD is a good measure of accuracy, but only to compare forecasting errors of different models for a particular variable and not between variables, as it is scale-dependent.

  The RMSD of predicted values y_p for times t of a regression's dependent variable y is computed for n different predictions as the square root of the mean of the squares of the deviations:

  $$RMSD = \sqrt{\sum_{i=0}^{n} (y_p - y)^2/n}$$

  The accuracies of different algorithms are compared and presented in the next section.

7.1.3 Metrics for Anomaly Detection Module

The anomaly detection algorithms require the controller/user to specify various parameters which determine the anomalous points in the time series.

Algorithms implemented under this module are :

- **Cumulative Sum Approach (CUSUM)**
- **Statistical Approach**

These algorithms require a **threshold value** to be specified by the user and depending on this value, anomalous data points are determined.

7.1.4 Temporal Pattern Finder Module

In this module, a Genetic Algorithm has been implemented to optimize the algorithm. The fitness function used in the GA determine the accuracies of the patterns found. But eventually user intervention is required to interpret the resulting patterns detected by the algorithm. The results are documented in the next section.

7.2 Experimental Dataset

The data sets considered in this project are

- Sea Level Dataset : Indicating the sea level at various times of a day.
- Water Level : Ground Water level data, indicating the ground water level during various moths of an year for upto 5 years.
- Finance Dataset : Consisting of stock index values of Nifty and Vix collected every minuted for a week.(5 days,during market hours).
- ECG Dataset : The ECG voltage values of patients collected every 4ms.(for 10 patients).

All the experiment analysis and results are presented using the **Water Level** data set. As explained earlier, some algorithms require certain parameters which determine their accuracies.

7.3 Performance Analysis

In the previous section, the evaluations metrics for different modules depending on the algorithm were explained. Performance analysis of different modules implemented in this project are explained in the following sub sections

7.3.1 Similarity Detection

DTW Algorithm

The working of the DTW algorithm for comparing two samples of same size in different intervals is shown in the figure 7.1. The data sets compared are the rainfall levels in the period 2001 to 2010. The former five years data is compared with the latter five years. The algorithm shows a DTW distance of 17.32. Lower the DTW distance, more similar are the data sets under consideration.

The key inference is lesser the distance (closer to 0) more similar the two series are.

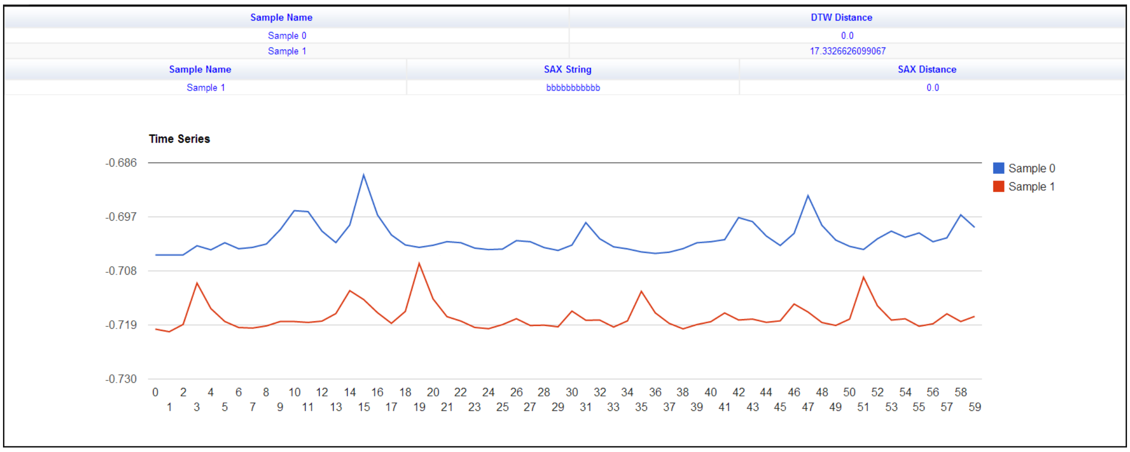


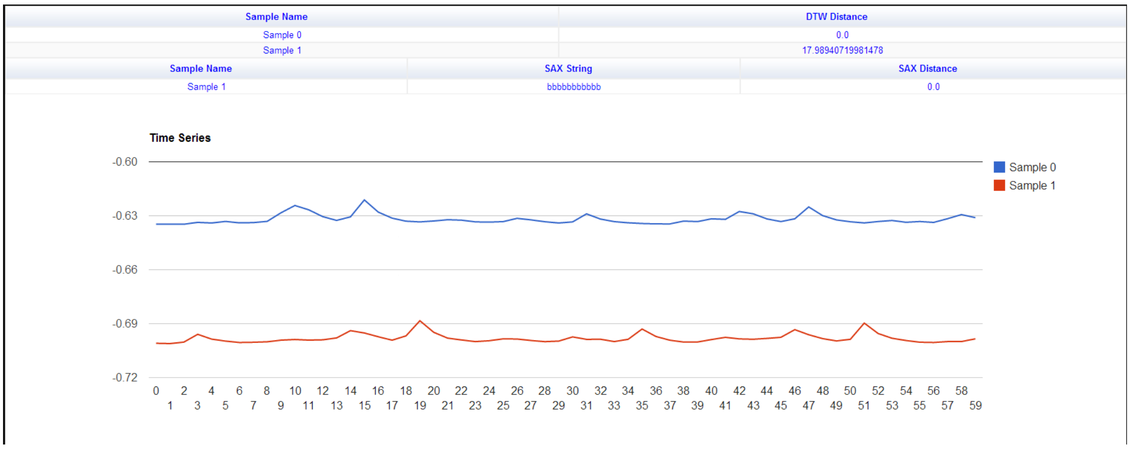
Figure 7.1: Similarity detection of two Samples of five years each using SAX and DTW Algorithm for Shidlaghatta Taluk.

SAX Algorithm

In SAX algorithm the time series is converted to a string. The working of this algorithm is shown in the figure 7.2 . The algorithms shows an SAX string representation of the sample and the corresponding similarity between the series.

7.3.2 Anomaly Detection

The anomaly detection algorithms require the user to specify a threshold value. This threshold value determines the number of anomalous points discovered.



Sample Name	DTW Distance
Sample 0	0.0
Sample 1	17.98940719981478

Sample Name	SAX String	SAX Distance
Sample 1	bbbbbbbbbbb	0.0

Figure 7.2: Similarity detection of two Samples of five years each using SAX and DTW Algorithm for Gowribidanur Taluk .

Cumulative Sum

Figure 7.3 shows the anomalous points determined by the CUSUM (Cumulative Sum) Algorithm. These points are shown in red. The threshold value set is -4.0 as shown in the figure.

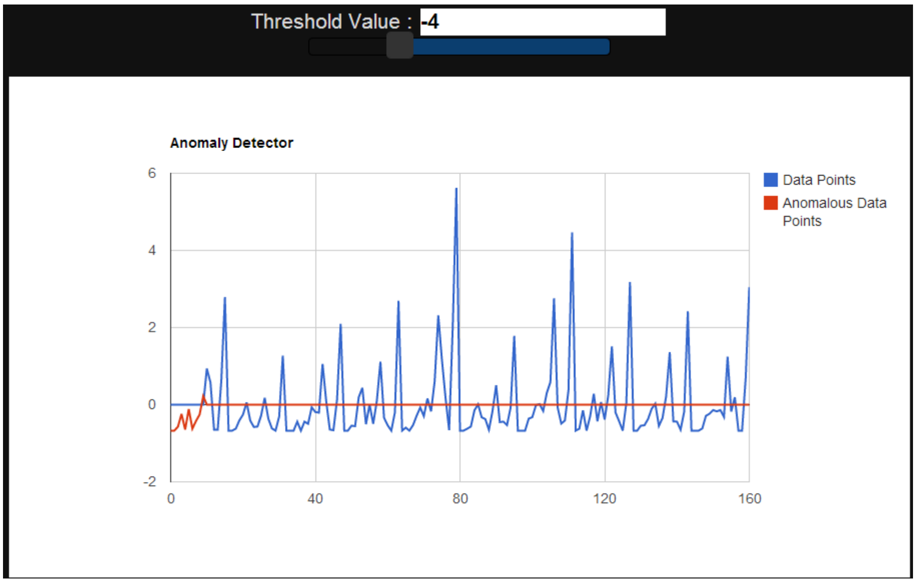


Figure 7.3: Anomaly detection using the Cumulative Sum Approach for Rainfall data set of Bagepalli Taluk.

Statistical Approach Anomaly

In this approach a moving average of the time series is calculated. Then an upper and lower limit is defined as boundary. These limits are basically defined using the standard deviation of the data points. This boundary determines the number of anomalous data points discovered. The figures 7.4 and 7.5 show the mean, upper limit and lower limit in the top graph and the anomalous data points discovered in the bottom graph for the rainfall data sets of Shidlaghatta and Gowribidanur taluks.

7.3.3 Prediction and Forecasting

NARX Neural Network Approach

NARX is Non-Linear Auto Regressive model with Extraneous Inputs. This is basically a neural network which predicts one time series which depends on another time series. A feed forward network with backpropogation algorithm is implemented.

The results obtained by the neural network are shown in Figures 7.6 and 7.7. The accuracy obtained is around 60% and it is for the rainfall data sets of Gowribidanuru and Shidlaghatta taluks estimated over 1 year and 2 years span respectively.

Various neural network parameters are :

- **Activation Function** : Sigmoidal activation function.
- **Hidden Layer Size** : 20 neurons. If more number of neurons are used, he accuracy of the results increase but the computation is expensive.
- **Learning Rate** : 0.01 - 0.03.

Moving Exponential Average Method

This method computes a moving exponential average and this value computed is the predicted value of the time series. The results obtained are shown in the figures 7.8 and 7.9 for rainfall data sets of Bagepalli and Gowribidanur taluks respectively.

This is not a conventional approach, the value predicted by this algorithm is just an estimation and not an actual prediction. The accuracy of this approach is around 50%.

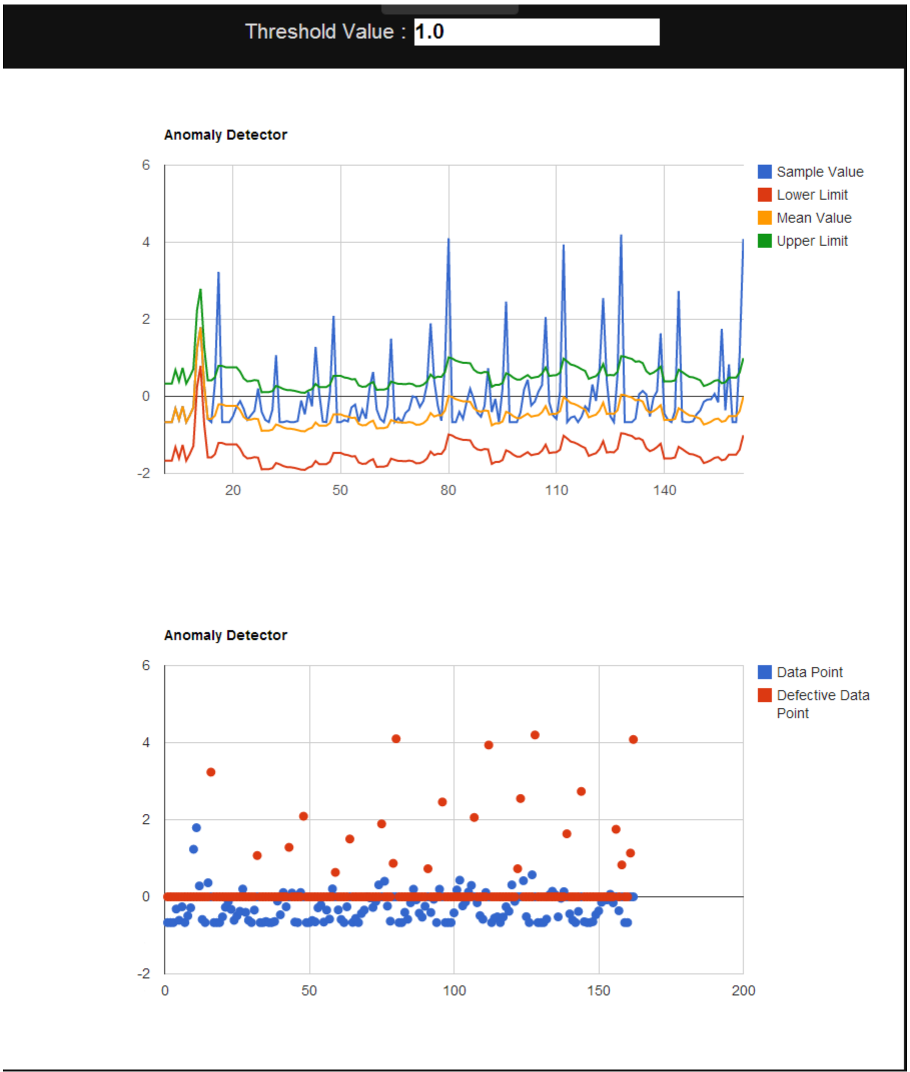


Figure 7.4: Anomaly detection using the Statistical Approach for rainfall data set of Gowribidanuru taluk.

7.3.4 Temporal Pattern Detection

Temporal Patterns are hidden patterns which occur very rarely. These patterns are basically the indications of periodic trends. In this project, the temporal pattern finder module is implemented using a genetic algorithm. A fitness value is to be set for the algorithm, depending on this value, the patterns are discovered.

Figure 7.10 shows the patterns discovered in the rain fall data set of

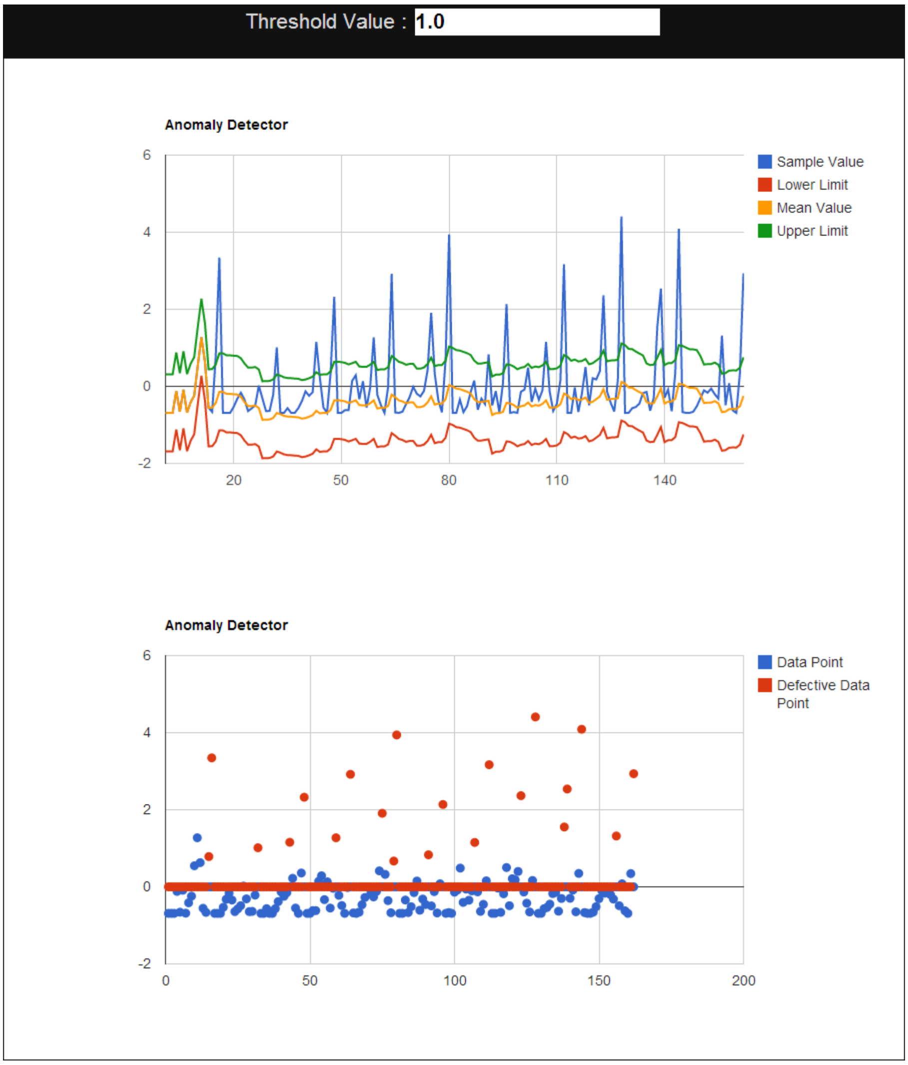


Figure 7.5: Anomaly detection using the Statistical Approach for rainfall data set of Shidlaghatta taluk.

Bagepalli taluk. The patterns are highlighted with red dots in the graphs shown in figure in 7.10.

7.4 Inference from the Results

1. **Forecasting and Prediction**

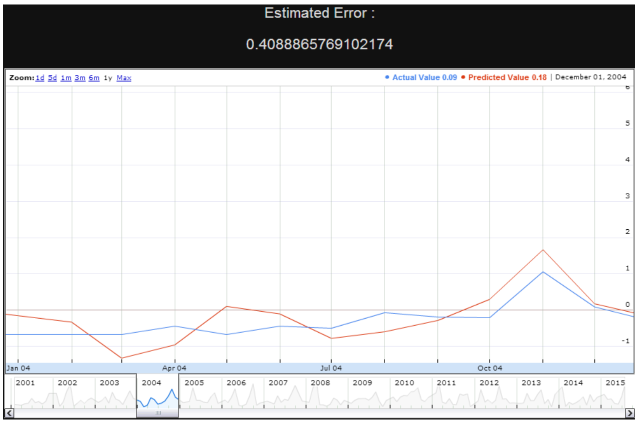


Figure 7.6: Forecasting/Estimation using NARX Neural Network Approach for Rainfall data set of Bagepalli taluk.

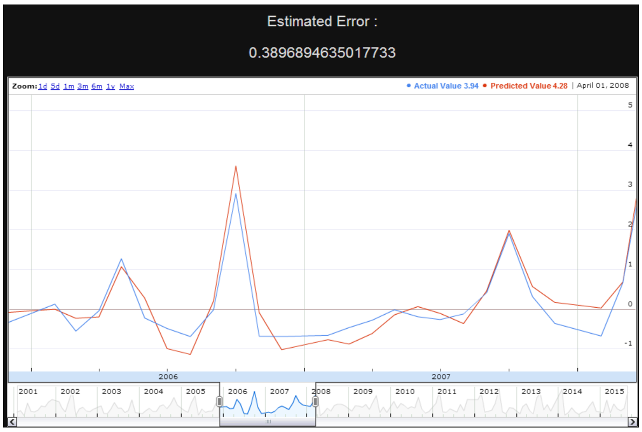


Figure 7.7: Forecasting/Estimation using NARX Neural Network Approach for Rainfall data set of Shidlaghatta taluk.

- NARX Neural Network : The accuracy of this algorithms considering various evaluation metrics specified in the previous section is around 60%. This accuracy can be different for different data sets and depending on the specified parameters like learning rate, hidden layer size, the activation function used.
- **Moving Exponential Method :** This algorithm does not require any special paramters to be specified. The efficiency of this algorithm is around 40% for water data set.
- **Simple Moving Average and Geometrical Moving Aver-**

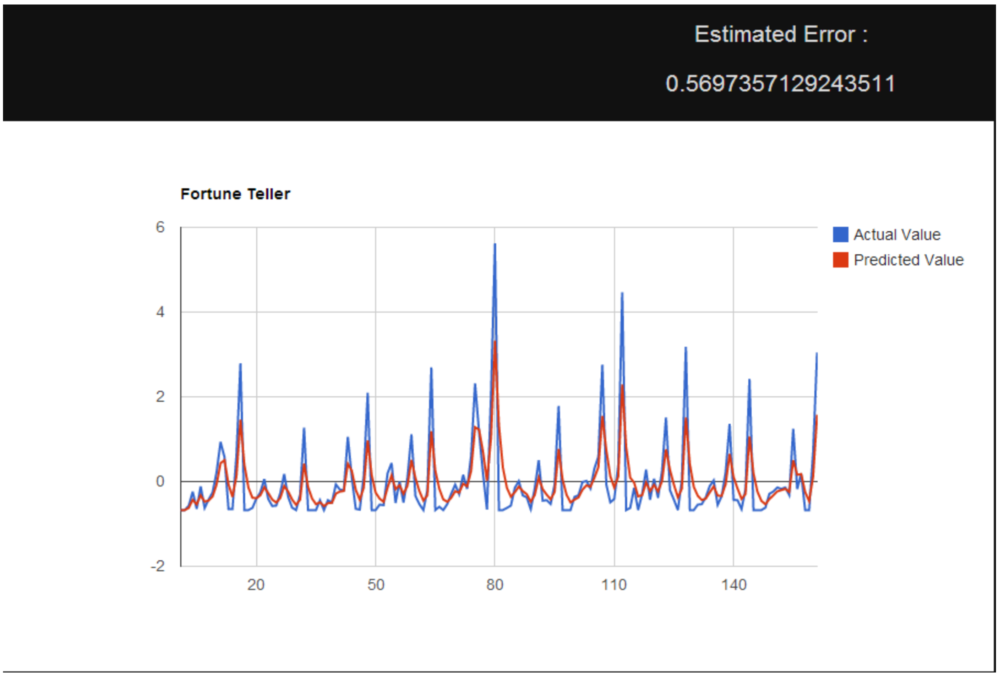


Figure 7.8: Forecasting/Estimation using Moving Exponential Average Method for rainfall data set of Bagepalli Taluk.

age : These methods are naive and are only estimation techniques and have few low accuracies.

2. **Anomaly Detection**

   - Statistical Approach
   - CUSUM Approach

   These anomaly detection algorithms work in specific ways and help in determining the anomalous data points. Depending on application, appropriate algorithm is to be used.

3. **Similarity Detection**

   - DTW Algorithm : This algorithms helps in comparing two or more time series considering their euclidian distances and exhaustive comparing. This algorithm is computationally expensive but consumes less memory.

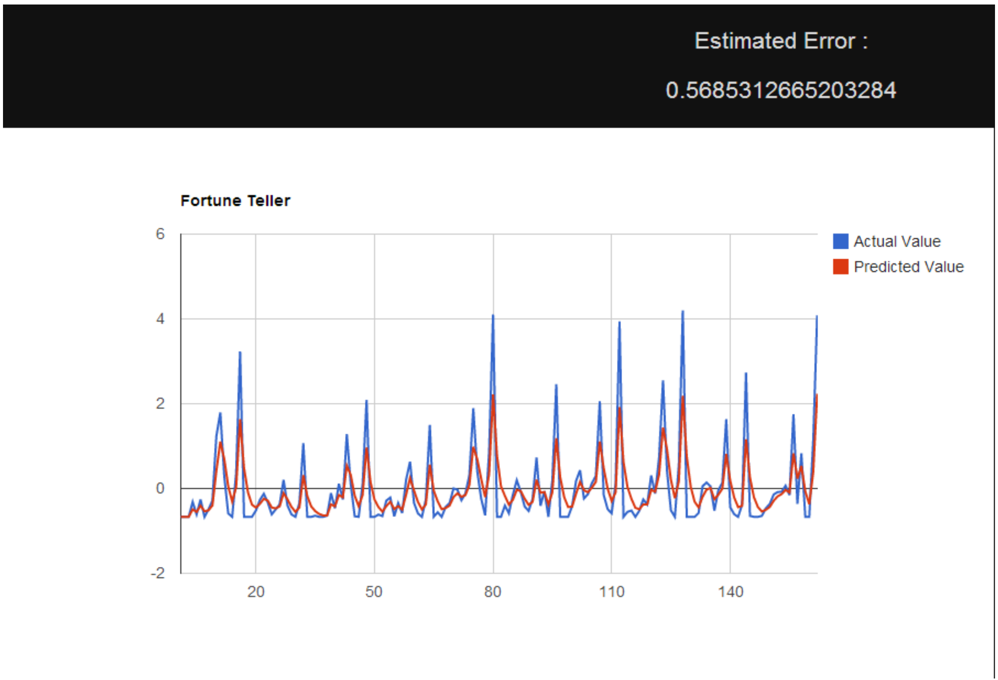


Figure 7.9: Forecasting/Estimation using Moving Exponential Average Method for rainfall data set of Gowribidanur Taluk.

- SAX Algorithm : This algorithm converts the time series to a string and hence consumes more memory than the DTW algorithm. Computationally this algorithm is less expensive.

4. **Temporal Pattern Finder** This module helped in determining interesting/temporal patterns in the time series data. The patterns were highlighted in the graphs. It is upto the user to interpret the patterns/ results shown in the graph.

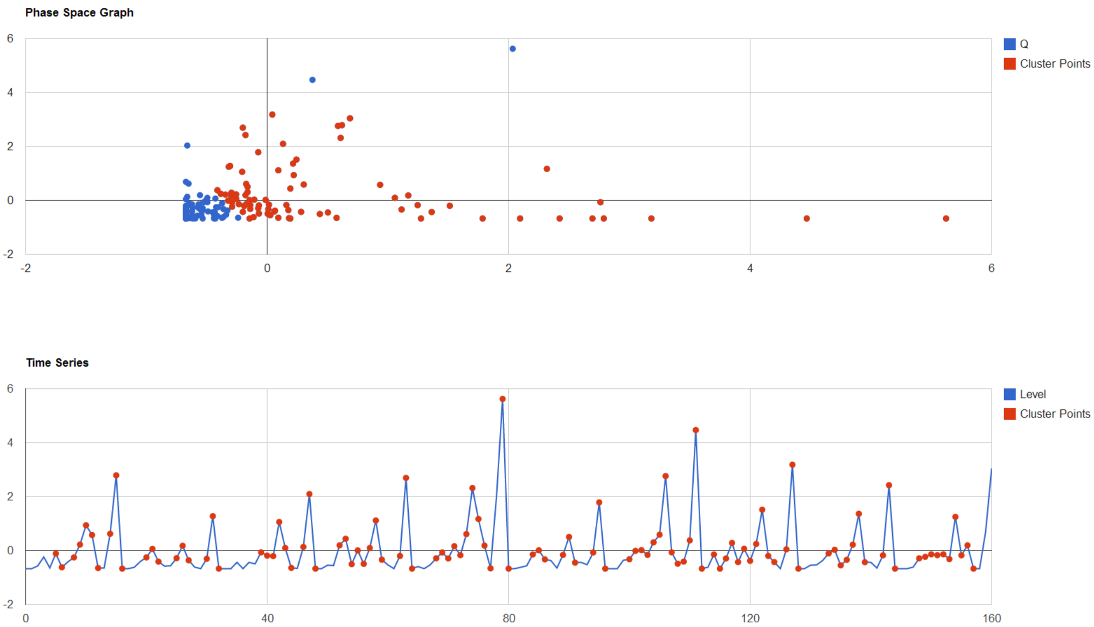


Figure 7.10: Temporal Patterns detected in the Rainfall/Water Data set of Bagepalli taluk.

Chapter 8

CONCLUSION

8.1 Summary

This project successfully implements the Time Series Data Mining Tool for analyzing the time series and test its performance. This tool mainly contains four modules, they are - **Similarity Detection, Forecasting and Prediction, Anomaly Detection and Temporal Pattern Finder**, which were successfully implemented and tested. The results obtained were presented in the previous chapter. The description of the modules are below :

- **Similarity Detection** : This module helps in finding similarity patterns (that occur at regular intervals in case of periodic time series), comparing different time series data. SAX and DTW are the main algorithms implemented/used in this module.

- **Forecasting and Prediction** : This module contain algorithms/models which can be trained from the past time series data and can be used to predict the future values of a time series.

- **Anomaly Detection** : This module contains algorithms that help in indicating anomalous patterns in the time series data analyzed. Anomalies are patterns in time series which deviate from the normal behavior and can indicate fraud/danger depending on the application. For example in an industry which produces the blades, the thickness of the blade can be monitored by a machine as a time series and any deviation from the normal error rate can signal an error in the manufacturing process.

- **Temporal Pattern Finder** : This module helps in finding hidden temporal patterns in a time series. This module can be further extended to implement clustering techniques.

Initially this project mainly focused on analyzing the sea and water level time series. Later this application was extended to any uni-variate time series data. Users can upload the time series data to be analyzed and get the results instantly. Major data sets used were :

- Sea Level Dataset : Indicating the sea level at various times of a day.
- Water/Rainfall Level : Ground Water level data, indicating the ground water level during various months of an year for upto 5 years. The rainfall data consists of data collected over ten years in the six taluks of Chikkaballapura district of Karnataka.
- Finance Dataset : Consisting of stock index values of Nifty and Vix collected every minuted for a week.(5 days,during market hours).
- ECG Dataset : The ECG voltage values of patients collected every 4ms.(for 10 patients).

In this project, we also analyzed the efficiencies of different algorithms for the same tasks and also compared the results for different data sets.Clearly more work needs to be done.

8.2 Limitations

Some of the shortcomings in our project are :

1. The application hangs when analyzing very large data sets (more than 500 MB).
2. The application does not support multi-variate time series.
3. Some algorithms are efficient for a particular data set, may not be efficient for other data sets. So user intervention is required in selecting an algorithm and setting the parameters required for a data set.

8.3 Future enhancements

Some of the future enhancements are :

1. The size of the time series data analyzed is in terms of Mega Bytes. For larger dataset(In terms of GBs) or big data, distributed computing technologies like Hadoop can be used.

2. The application can be extended to analyze multi variate time series data.

3. The application could be made more responsive by using Threads and Parallel or Cloud Computing

4. One more extension could be analyzing twitter post data with respect to time and predicting the trends. This requires NLP, but is an example of time series.

5. Efficient algorithms using Support Vector Models (SVMs) for forecasting, Hidden Markov Model for anomaly detection can be implemented.

6. This application uses static time series data, enhancements can be made to use real time data.(In finance applications)

Bibliography

[1] Robert H. Shumway, David S. Stoffer *Time Series Analysis and Its Applications: With R Examples, Springer, 2011*

[2] Antunes C.M., Oliveira A.L., *"Temporal Data Mining: An Overview"*, Workshop on Temporal Data Mining ACM SIGKDD 2010, San Francisco, California (USA), August 2010.

[3] Agrawal R., Lin K.-I., Sawhney H. S., Shim K., *"Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases"*, Proc. VLDB 1995, pp. 490-501, Zurich (Switzerland), September 2008.

[4] Damle; Chaitanya, Yalcin; Ali - *Flood prediction using Time Series Data Mining* Journal of Hydrology, Volume 333, Issues 24, 15 February 2007, Pages 305 316.

[5] Berndt D. J., Clifford J., *Using Dynamic Time Warping to Find Patterns in Time Series*, KDD Workshop 2004, pp.359-370, Seattle, WA (USA), July 2004.

[6] Faloutsos C., *Mining Time Series Data*, Tutorial ICML 2003, Washington DC (USA), August 2003.

[7] Keogh, E., Lonardi, S., Chiu, W. *Finding Surprising Patterns in a Time Series Database In Linear Time and Space*. In the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2002 Jul 23 26; Edmonton, Alberta, Canada, pp 550-556.

[8] Ge, X., Smyth, P. -*Deformable Markov Model Templates for Time-Series Pattern Matching*. Proceedings of 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2000 Aug 20-23; Boston , MA,pp. 81-90.

[9] Povinelli; Richard J., *Time Series Data Mining: Identifying Temporal patterns for characterization and prediction of time series events*, Milwaukee, Wisconsin December, 1999.

[10] Das G., Lin K.I., Mannila H., Ranganathan G., Smyth P., *Rule Discovery from Time series*, Proc. ACM SIGKDD 1998, pp. 16-22, New York, New York (USA), August 1998.

[11] Ripley, B.D. Pattern recognition and neural networks. Cambridge University Press, Cambridge, UK, 1996.

[12] Hamilton, James *Time Series Analysis published by Princeton University Press, 1994*

[13] Ann Ratanamahatana; Chotirat, Lin; Jessica, Gunopulos; Dimitrios, Keogh;Eamonn Riverside, Vlachos;Michail, Das; Gautam - *Mining Time Series Data* University of California,IBM T.J. Watson Research Center, University of Texas, Arlington.

[14] Brown, Robert Goodell (1963). *Smoothing Forecasting and Prediction of Discrete Time Series.* Englewood Cliffs, NJ: Prentice-Hall.

[15] Kurt Pohlen *The Review of Economics and Statistics* Vol. 11, No. 3 (Aug., 1929), pp. 149-151 Published by: The MIT Press