BOĞAZİÇİ UNIVERSITY

DEPARTMENT OF MANAGEMENT INFORMATION SYSTEMS

MATLAB AS A DATA MINING ENVIRONMENT

June, 2007

MATLAB AS A DATA MINING ENVIRONMENT

by

Submitted to Department of Management Information Systems

as a requirement for the degree of

Bachelor of Science

in

Management Information Systems

Boğaziçi University

June, 2004

MATLAB AS A DATA MINING ENVIRONMENT

APPROVED BY:

Bertan Badur, Ph.D., Assistant Professor (Advisor)	…………….............
Name of Jury Member1	………………….............
Name of Jury Member 2	………………….............

DATE OF APPROVAL: .... …………………….............

ABSTRACT

The subject of this study is to create a data mining environment. Through the content of the "Developing a Data Mining Platform" study (2006) and the analysis of the necessities, the tool is designed in MATLAB with the interfaces.

The study contains a detailed analysis of three existing data mining tools and the functionalities and graphical user interface capabilities of MATLAB. According to the preliminary study, the functionalities of the proposed data mining tool and the structure of interfaces determined.

The study began by investigating the need and the analysis of the tool that will have the ability to perform necessary data mining functions and resulted with design of the tool.

The phases of the study are literature survey, investigation of MATLAB, determining data mining methodology for the project and the design of the tool.

ÖZET

Bu çalışmanın amacı bir veri madenciliği aracı yaratmaktır. 2006 yılında "Veri Madenciliği Platformu Geliştirme" projesi ve bu konudaki ihtiyaçlar analiz edilerek, kullanıcı arayüzleri ile birlikte araç MATLAB' ta dizayn edilmiştir.

Çalışma, veri madenciliğinde kullanılan üç programın ve MATLAB' ın fonksiyonlarının ve arayüz tasarlama becerilerinin detaylı bir şekilde incelenmesini içermektedir. Bu analizden yola çıkılarak dizayn edilen veri madenciliği aracının fonksiyonları ve arayüzlerin yapısı belirlenmiştir.

Çalışma, ihtiyaçların analizi ile başlamış ve gerekli veri madenciliği fonksiyonlarını gerçekleştirecek yetenekte bir aracın dizaynıyla sonuçlanmıştır. Projenin fazları literature araştırması, MATLAB' ın incelenmesi, veri madenciliği methodolojisinin belirlenmesi ve aracın dizaynından oluşmaktadır.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

There has been a good contribution to the creation of this study besides a good team work of the project team members.

Besides the successful example of teamwork of the project team members, there have been precious contributions to the project.

Especially, MIS Department Instructor Mr. Bertan BADUR has contributed and collaborated a lot to this project by his professional guidance.

We would also like to thank the evaluation committee for their time and evaluation efforts.

LIST OF FIGURES

1. INTRODUCTION

The main purpose of this study is designing a data mining environment within MATLAB that combines many data mining functionalities.

MATLAB is one of the commonly used data mining tools. It is a high-level language and interactive environment that enables user to perform computationally intensive tasks in fastest way. MATLAB can be used in a wide range of applications, including signal and image processing, communications, control design, test and measurement, financial modeling and analysis, and computational biology.

However, it does not include a data mining toolbox. So the last year, in the "Developing Data Mining Platform" project data mining functions added to MATLAB. This project is the continuation of the last year study that is preparing GUIs for data mining functions added to MATLAB and providing usage of all these functions from interfaces.

MATLAB has the command line environment, all applications are made with commands. It provides flexibility and power to expert users. They easily manipulate data and make analysis. Nevertheless, this structure stress novice users, they may confuse while using commands and this requires serious learning of usage of commands.

MATLAB have many different toolboxes includes statistics and neural network control etc that provides user interfaces for specific functions. They do not exactly cover all data mining models. Furthermore, there is need to reach all data mining functions through one main window.

MATLAB has its own GUI Design Environment (GUIDE). The project is developed within the MATLAB environment through the Layout Editor of Guide.

Before the design step, the methodology of the project is determined. CRISP-DM is used to determine the scope of the project and the steps of design phase. According to the developed data mining functions in MATLAB, the structure of the main screen, the menus and their contents, the interfaces and their design options determined. While designing the user interfaces, usability principles are considered to make user-centered design.

The tool contains data manipulating, function preprocessing and data modeling. In the data module, there are functions that allow creating lists, adding variables to list, setting metadata of variables and modification of them and descriptives.

In the preprocessing module, there are functions let user make handling of missing values, sampling, transformation and discretization.

In the modeling module, there are association, classification, clustering, regression and sequention modeling designs that are designed with the basic algorithms.

This tool is also designed to ease the use for novice users. The user without any data mining background can easily learn to use this tool with strong help and user guide.

2. MOTIVATION

The project aims to design a Data Mining environment using MATLAB with many functionalities. The project is shaped on the idea of developing data mining capabilities of MATLAB and designing a Data Mining tool based on these capabilities.

Another aim of the project was to design a Data Mining tool that will be easier to understand and use with user-friendly interfaces. Command line environment of MATLAB is so powerful but it is not easy to use, designing a tool using graphical user interfaces is essential to provide easiness.

Detailed literature survey studies are made in order to get information about interface designs and functionalities of available tools and about MATLAB graphical user interface design. The functionalities of the tool are designed with the information gained via previous study made in MATLAB and these surveys.

This project is prepared to become a continuation of "Developing a Data Mining Platform" project and a basis for further additions and studies. Also, all of the interfaces of the tool that will be used in the future are prepared. With this final project, a base has been prepared for further projects that will be formed in order to improve this tool and to increase usability of graphical user interfaces of MATLAB.

3. LITERATURE SURVEY

As the first phase of the study, the existing data mining tools are investigated. The purpose of this investigation is to have an outline of tools together with the functions they focus on and to examine user interface design of them. These tools will be used as a model for this study. After the construction of the list, detailed information about tool is given.

3.1 Data Mining Tools Used As Model

1 SPSS [www.spss.com]

SPSS for Windows:

Statistical and data management package

Add-on modules and stand-alone software

Linear Regression

Factor Analysis

Two-step Cluster Analysis

K-Means Cluster Analysis

Hierarchical Cluster Analysis

Ordinal Regression (PLUM)

2 MIS Data Miner

MIS DM:

Data Mining Environment

Preprocessing Functions

Association

Classification

Clustering

Prediction Modeling

3 WEKA [www.cs.waikato.ac.nz/ml/weka]

Weka:

Machine learning algorithms for data mining tasks

Open source software

Data pre-processing

Classification

Regression

Clustering

Association rules

Visualization

3.2 Detailed Investigation of Data Mining Tools

3.2.1. SPSS

Name of the Product:

SPSS for Windows

Version: 14.0

Vendor of the product:

SPSS Inc.

What does the SPSS for Windows do:

SPSS for Windows supports wide range of capabilities for the entire analytical process. It helps to make decision by using powerful statistics, to understand and effectively display results with high-quality tabular and graphical output. It enables to make smarter decisions more quickly by uncovering key facts, patterns, and trends.

Platform& Architecture:

Microsoft® Windows XP or 2000

Data Types:

- String
- Numeric
- Comma
- Dot
- Date
- Scientific notation
- Dollar
- Custom currency

Data Interface:

ODBC-compliant databases

OLE DB data sources

Text Wizard

Microsoft Excel® Data

SAS® Data

Data Mining Capabilities:

Descriptive statistics

- Crosstabulations
- Frequencies
- Descriptives
- Explore
- Descriptive ratio statistics

Bivariate statistics

- Means
- *t* tests
- ANOVA
- Correlation
    - Bivariate
    - Partial
    - Distances
- Non-parametric tests

Prediction for numerical outcomes and identifying groups

- Factor analysis
- TwoStep cluster analysis
- K-means cluster analysis
- Hierarchical cluster analysis
- Discriminant
- Linear and ordinal regression
- Ordinal regression
- Principal components analysis

Data Manipulation:

SPSS for Windows lets the user defining, entering, editing, and displaying data for modeling. The functions are:

- Identify duplicate copies
- Discard or replace missing values
- Derive new fields: Adds new fields to further analyses.
- Create bins or bands (for example, break income into "bins" of 10,000 or break ages into groups)

User Interface:

There is Data Editor with spreadsheet-like system, it simplify evaluating and modifying data.

The menu system of the SPSS is well- designed, all functions grouped under the related menus and the content of the menus are wide. There are data preparation features and tools to clean and prepare data for modeling. SPSS provides write back to databases from SPSS or export data to CSV (comma-separated value) file format.

Representation of Outputs:

SPSS creates easy-to-understand results from data analysis with variety of charts and graphs so decision makers can quickly understand the information. The results can be summarized by plots or charts and the relationships among data sets can be found out easily.

There are some output object types used for display of information like tables, logs, charts, trees and headings. They let the user to see the results in standard and organized format.

SPSS can export outputs to Excel, Word, PDF, PowerPoint format, more over it can report OLAP cubes.

Users can also automatically export all output or user-specified types of output as text, HTML, XML, or SPSS-format data files.

Conclusion:

SPSS for Windows provides user to make statistical analyses with large databases. It helps to prepare data for analysis and explore data. It provides to use different models and find different relationships to extract and reach meaningful results. It lets user to present results with high-quality tabular and graphical output, and share them with others using a variety of reporting methods, including secure Web publishing.

3.2.2. MIS Data Miner

<u>Name of the Product:</u>
MIS Data Miner
Version:1.0 (2003)

<u>Vendor of the product:</u>
Boğaziçi University Department of MIS

<u>What does the MIS DM do:</u>
It is a data mining environment that combines many data mining functionalities.
It serves all preprocessing methods such as outliners, discretization, and transformation so on. In the modeling phase, association, classification, discretization and other models are built with basic algorithms. It is helpful for novice users to make data mining applications.

<u>Platform& Architecture:</u>
Windows 98/200/XP

<u>Data Types:</u>
Nominal / Ordinal
Date/Time
Categorical

<u>Data Interface:</u>
Comma Separated Values File (*.csv)
Tab or return separated text files (*.txt)
Microsoft Excel
ARFF files
Databases through ODBC connection

Data Mining Capabilities:

- Modeling:
    - Classification
    - Association
    - Clustering
    - Prediction
- New models and algorithms can be added
- Generate Text Design: Percentages of Training and Test can be selected by the user.
- Build Model: Model description will display statistics, ratios, misclassification matrices, etc.

Data Manipulation:

Outliners provide to exclude the record, to make the value missing and set the value to threshold or user defined value.
Preprocessing functions like missing value handling
Sampling operations
Data transformation
Data integration can be done by appending rows and columns to a table from another one.
Data discretization

User Interface:

There are three main windows in MIS DM. Those are the Project Window, the Workspace Window and the Process Window. At the Project Window the tree structure is used for project displays. At the Workspace data sets are displayed. At the Process Window information about selected item is given.

Representation of Outputs:

MIS DM provides see the results in report format, user can add graphs or charts to summarize data. User can save outputs as many ways with merging two or more reports or saving single one. Reports can be saves as .doc files. Graphs in the reports can be saved separately in picture format. User can edit reports and add bookmarks to reach some headers in the report.

Conclusion:

MIS DM combines many data mining tools in one package. It is useful for novice users too. Through preparing data for analysis to data modeling the program includes all steps to make analyses and reach accurate and meaningful information. Moreover it lets to see results and efficiently evaluate the by reporting options.

3.2.3. WEKA

Name of the Product:

WEKA- Waikato Environment for Knowledge Analysis

Version: 3.4.10 (includes Java Runtime Environment)

Vendor of the product:

WEKA is the open source data mining tool developed by Waikato University Computer Science Department in New Zealand

What does the WEKA do:

Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes.

Platform& Architecture:

Java 1.4 (or later) is required to run Weka 3.4.x and older versions. It has been tested under Windows, Linux operating systems.

Data Types:

Numeric (Real & Integer)

Nominal

String

Date

Data Interface:

- *.arff
- CSV (comma separated) or tab separated
- Spreadsheet

Data Mining Capabilities:

Classification algorithms include Decision Stump, Decision Table, ZeroR, HyperPipes, ID3, IB1, IBk, J48, J48-Part, KernelDensity, Kstar, Logistic, NaiveBayes, Regression, Id3 Decision tree, OneR, Prism, SMO, VFI, AdaBoostM1, M5Rules, Part, Ridor, Meta Cost Neural Network and many other ones.

Clustering algorithms include FarthestFirst, DensityBasedClusterer, SimpleKMeans, EM, Cobweb algorithms.

Association algorithms include Apriori and PredictiveApriori.

Attributes include evaluation algorithms such as, InfoGainAttributeEval, CfsSubSetEval, ChiSquaredAttributeEval, GainRatioAttributeEval, OneRAttributeEval, ReliefFAttributeEval, SVMAttributeEval, SymmetricalUncertAttributeEval, UnsupervisedAttributeEvaluator

Data Manipulation:

WEKA has filtering algorithms for sampling, missing value handling and transformations.

WEKA provides to preprocess a dataset, feed it into a learning scheme, and analyze the resulting classifier and its performance—all without writing any program code at all. Tools for preprocessing a dataset, which we call filters, are an important component of Weka. Like classifiers, filters have a standardized command-line interface, and there is a basic set of command-line options that they all have in common.

User Interface:

WEKA displays each data mining function on a different tab in the main screen. The interface leads user through rich functionality allowing them to preprocess data, perform machine learning on their dataset, and visualize the results. However the learnability of the interface is low and it is not easy to use for novice users and users without Linux experience.

Representation of Outputs:

Displays statistical measures for each attributes in preprocess tab. Before applying the data mining algorithms, it gives a visual overview to user and if the user accepts it applies the algorithm to the data and displays the results in the text format. It visualizes data only in 2D scatter plot diagram.

Conclusion:

WEKA is an open code, university- developed, free data mining tool. It works in Java environment and because of open code and object- oriented structure, it lets user to add new models to the tool. It is flexible and strong.

3.3 Online Sources Used For Developing GUI

1. www.mathworks.com

It is the web site of MATLAB. It gives support to users with documentations, user communities, reports, downloads, user stories and additional resources. The site has search engine for support, users can search anything according to product or they can specify where to look for search.
Moreover, in the website there are demos for toolboxes or other applications, users can load them to their computer and work. The site gives technical support to users.

2. www.mathworks.com/matlabcentral/fileexchange

In this web page the users share their works, reports about MATLAB. There are some categories like graphics, aerospace, optimization etc. The users add .m files or documents under the related topic and serve them to others' use. The web page provides search options to reach required information quickly.

3. newsreader.mathworks.com

In the web page like forums, people write problems they face when using MATLAB and give response to coming problems. There is not any restriction about the subject, it may be so little problem about user interface design or complex problem related with engineering disciplines. This web page provides search options too.

4. mathforum.org

It is the web site of Drexel University in Texas about math learning, teaching, and communication. Under the discussion titles there is software topic that includes some programs. MATLAB is one of them, under this topic people share their problems about programming in MATLAB and search for help of the other people.

3.4 Design Rules for GUI

It is important problem to how interfaces of the tool should be designed to increase usability. Here, we require design rules should be followed to increase usability of software product. There are general principles, which can be applied to the design of interactive system in order to promote its usability. While we are preparing the interfaces of tool we take account of them for effective design. In the following, the usability principles will be mentioned in categorical way.

Learnability – the ease with which new users can begin effective interaction and achieve maximal performance.

Flexibility – the multiplicity of ways in which the user and the system exchange information.

Robustness – the level of support provided to the user in determining successful achievement and assessment of goals.

Learnability

It allows novice users to understand how to use the tool initially and then how to attain a maximal level of performance.

1. Predictability- Support for the user in determining the effect of future action based on past interaction history
2. Synthesizability - Support for user in assessing the effect of past operations on current system state
3. Familiarity - Does UI task leverage existing real-world or domain knowledge? – guessability, affordance
4. Generalizability- Can knowledge of one system/UI be extended to other similar ones?
5. Consistency- Likeness in behavior between similar tasks/ operations/ situations/ terminology

Flexibility

It refers to multiplicity of ways that users and system exchange information.

1. Dialog Initiative- Allow the user freedom from artificial constraints on the input dialog imposed by the system, user pre-emptiveness is preferable.

2. Multithreading- Allowing user to perform more than one task at a time. It can be concurrent or interleaving.

3. Task migratability- Ability to move performance of task to the entity (user or system) that can do it better

4. Substitutivity- Equivalent actions/values can be substituted for each other- representation multiplicity

5. Customizability- Modifiability of the user interface by the user or the system- adaptivity and adaptability

Robustness

It covers supporting user in determining successful achievement and assessment of goals.

1. Observability- Ability the user to evaluate the internal state of the system from its perceivable representation- browsability, reachability, persistence

2. Error Prevention- Make it hard for the user to make an error. Don't let the user do something that will lead to an error message

3. Recoverability- Ability to take corrective action upon recognizing error

4. Responsiveness- Users perception of rate of communication with system- stability

5. Task Conformance- Does system support all tasks user wishes to perform in expected ways? It includes task completeness and task adequacy [1]

3.4.1 Shneiderman's Principles of Human-Computer Interface Design

In the design of the interface user types should be taken account from novice user, knowledgeable but intermittent user and expert frequent user. Each type of user expects the screen layout to accommodate their desires, novices needing extensive help, experts wanting to get where they want to go as quickly as possible. the differences in users can be addressed by including both menu or icon choices as well as commands (i.e. Command or Control P for Print as well as an icon or menu entry), or providing an option for both full descriptive menus and single letter commands. Shneiderman's eight golden rules provide a convenient and succinct summary of interface design.

The Eight Golden Rules of Interface Design:

1. Strive for consistency
    - consistent sequences of actions should be required in similar situations
    - identical terminology should be used in prompts, menus, and help screens
    - consistent color, layout, capitalization, fonts, and so on should be employed throughout.
2. Enable frequent users to use shortcuts
    - to increase the pace of interaction use abbreviations, special keys, hidden commands, and macros
3. Offer informative feedback
    - for every user action, the system should respond in some way (in web design, this can be accomplished by DHTML - for example, a button will make a clicking sound or change color when clicked to show the user something has happened)
4. Design dialogs to yield closure
    - Sequences of actions should be organized into groups with a beginning, middle, and end. The informative feedback at the completion of a group of actions shows the user their activity has completed successfully
5. Offer error prevention and simple error handling
    - design the form so that users cannot make a serious error; for example, prefer menu selection to form fill-in and do not allow alphabetic characters in numeric entry fields
    - if users make an error, instructions should be written to detect the error and offer simple, constructive, and specific instructions for recovery
    - segment long forms and send sections separately so that the user is not penalized by having to fill the form in again - but make sure you inform the user that multiple sections are coming up
6. Permit easy reversal of actions
7. Support internal locus of control
    - Experienced users want to be in charge. Surprising system actions, tedious sequences of data entries, inability or difficulty in obtaining

necessary information, and inability to produce the action desired all build anxiety and dissatisfaction

8. Reduce short-term memory load
    - A famous study suggests that humans can store only 7 (plus or minus 2) pieces of information in their short term memory. You can reduce short term memory load by designing screens where options are clearly visible, or using pull-down menus and icons. [2]

3.5 Previous Studies

3.5.1. Preliminary Study of a Data Mining Environment (2003)

The main purpose of this study is designing a data mining environment which combines many data mining functionalities. The name of the tool is MIS Data Miner (MIS DM). To provide flexibility to the user, object- oriented programming was used in the tool. The scope of the tool design covers whole preprocessing functions and modeling of data is taken into consideration. In the preprocessing module, there are functions that allow the handling of the missing values and the outliners, discretization, transformation of the data, filtering and sampling. In the modeling module, there are association, classification, clustering and prediction modeling designs that are constructed with basic algorithms. Also this tool designed for novice users.

In the project the team applied AHP for selecting the programming language. They evaluated languages according to performance, visualization, being thought at school or not, maturity of language, being commonly used or not and time necessary to code and test. They selected Java, C++, Visual C++, Visual Basic and C#. At the end of the rank analysis VC++ selected for coding.

The team used CRISP-DM (Cross Industry Standard Process for Data Mining) process model as a data mining methodology. In the project, the team was interested in data understanding, data preparation, modeling and evaluation phases of CRISP-DM model related with data assessments and models together with outputs of MIS DM.

In the tool design the team used three windows for overall control in the main screen. One window is the project window, in the tree structure, it displays data, model and output branches. The other one is workspace, the item selected from project window is displayed in here. The last one is the process window; it gives information about the item selected in the workspace. Both workspace and process window are in grid format. All function interfaces were designed with using buttons, text boxes, list boxes etc. and their basic design increases usability of novice users. Moreover in the preprocessing function interfaces the type of the selected column is shown, so it helps user to determine the next step in the interface. After all functions the outputs is displayed in the main page workspace. [3]

3.5.2. Developing a MATLAB Platform (2006)

The main purpose of this project is developing a data mining platform. The project covers developing single data mining tool that has all data mining functionalities to meet requirements of user. So developing a data mining platform on an existing tool is more reasonable. The study contains a detailed analysis of five existing data mining tools and selection of the most suitable tool to develop a data mining platform.

After deciding to develop a data mining platform on an existing tool, the team made an analysis of five existing data mining tools namely SPSS, MATLAB, S-Plus, SQL Server 2005, and Weka. They applied AHP on them according to the KDD methodology and they evaluated the tools under 10 categories. These categories are data, sampling and missing value, transformation, graphics, programming, time series handling, data mining algorithms availability, data mining algorithms reliability, environment, and learnability.

The team determined priorities of categories in terms of their vitality for the project. The programming capabilities and deficiencies of these five tools were determined as the most important parameter that is essential for developing a data mining platform project. In addition, being able to develop new transformation functions easily in addition to the built-in ones is another important parameter. Then, the availability and reliability of the functions of these tools for various data mining algorithms and the easiness of the learnability of the programming languages that they use are other important parameters. Data retrieval capacity and supported data sources of these tools also have importance for such a project. On the other hand, sampling capabilities of these tools are not so important, because sampling functions can be written easily if the tool has a powerful programming language. Similarly, time series functions can be coded with such powerful languages. Because of the powerful programming capabilities, ease of use and learn the programming language, and the reliability of its functions, MATLAB was to develop a data mining platform.

The team used Knowledge Discovery and Data mining (KDD) methodology to list data mining functionalities, to determine which functionalities exist in MATLAB and which ones do not, and to make a list of functionality requirements for MATLAB. The methodology used to extract valuable information and knowledge in large volumes of data. The goal of knowledge discovery and data mining is to find interesting patterns and/or models that exist in databases but hidden among the volumes of data
Knowledge discovery in databases is the process of identifying valid, novel, potentially useful, and ultimately understandable patterns/models in data. Data mining is a step in the knowledge discovery process consisting of particular data mining algorithms that, under some acceptable computational efficiency limitations, finds patterns or models in data. the goal of knowledge discovery and data mining is to find interesting patterns and/or models that exist in databases but are hidden among the volumes of data []

Within the project, twenty-six MATLAB functions developed to meet most of the data mining functionality requirements. Most of them were data retrieval, pre-processing, transformation, and sampling functionalities. In addition to that, the team made extension on three existing data mining methods in MATLAB: kmeans, regression, and neural network. The team wrote codes with using M-file Editor of MATLAB, and the codes were applied at command line environment of MATLAB. [4]

4. PROBLEM DEFINITION

- **System Overview**

MATLAB is a suitable environment for function developers. Because, MATLAB is a powerful software development language which provides almost all the functionalities for the developers. Coders can access data, define complex functions and build complicated algorithms for complex calculations. MATLAB also provide the opportunity for building a platform which can be used in all the functionalities of data mining operations. Developers could use existing libraries of MATLAB for different purposes. Online developer communities, forums and portals provide m-files which are built by other developers and uploaded to web. All the m-files in the online communities are ready for execution. A MATLAB user can reach MATLAB files by making Google searches. For illustration, a Google search for "clustering MATLAB" gives the clustering algorithms which are developed by other coders around the world. Also developers can see the code of functions and the coders can make changes on the file because MATLAB files are open source files. In addition, MATLAB is also fit with the needs of power users who can use command line interface. The Expert user learns the commands and makes the execution instead of dealing with interfaces. Thus the power user doesn't waste time with interfaces and the user can also make alternative operations by using command line. Also for frequent users, command line interface and its language provide a strong feeling of having control over the system. Users learn the syntax and can often express complex possibilities rapidly, without having to read distracting prompts. The Power or expert users can use the functions after reading documentation of the functions or help files written by developers. Most of the engineering students use MATLAB in their projects for developing functions or they use already built functions for getting results by entering different values. However it is difficult to memorize commands and documentation notes. Also ordinary users can not use MATLAB command line and they are reluctant to read the user manuals and documentation files.

- **Problems of using MATLAB as a Data mining Environment**

1. MATLAB Requires Command Line Interface Usage:

Although there are a lot of ready libraries available in the internet, almost all the resources found in internet for execution of data mining operations are available in M-file format and they can be used by command line interface. In fact, it is a fundamental problem of MATLAB. That is, the MATLAB platform mostly used by academicians or technical users because of command line interface usage. Engineering, MIS and other technical departments can use the command line easily but students from other departments like physiology, sociology and management could not use the command line properly. Also most of the non-academic users can not use command line interface and they want to use graphical user interfaces which are familiar with other daily applications in the computer.

2. Graphical User Interface Development is neglected in MATLAB:

MATLAB usage is increased continuously since it was firstly released in late 1970s. Most of the students use MATLAB in their final projects in recent years. Although Mathworks provide new releases of MATLAB environment regularly, user interface side of MATLAB is not developed in the environment. New functions are added and toolboxes are expanded in all the new releases of MATLAB but user interface development is not regarded as precious as other sides. MATLAB is now a powerful language which provides all the libraries for developers and academicians. It also provides some powerful coding tips which are not available high level languages like Java or C#. However, all the good functionalities of MATLAB are available in command line screen. User interface side of MATLAB environment is very weak compared to other software development environment because the developers of MATLAB give priority to coding or development side more than graphical user interface side. User interface considered as plotting graphics of functions to the screen by most of the MATLAB users rather than building interfaces with buttons and list boxes.

3. Weakness of Graphical User Interfaces in MATLAB:

Building user interfaces in MATLAB requires calling functions of M-files from the user interface. Firstly, string operations are performed in background and then the strings are sent to the functions as inputs. Then the result of the functions are converted to strings and displayed to the users. String operations like adding strings to an array, removing strings, comparing strings and adding variables to user interface components are very difficult. Communication between different user interfaces is also hard. Dealing with dropdown menus, list boxes and text boxes are difficult than other languages. Also user interfaces operation is very slow and user wait times are longer than other languages. In brief, graphical user interface side of MATLAB is in the process of improvements and the development environment needs more effort and investment for building suitable user interfaces.

4. Resource Availability Problem:

Online user communities and books published for MATLAB are mostly about building functions and writing code in MATLAB. There are e few user interface examples found in the internet and user communities only explains basic user interface concepts in MATLAB. Interface design is covered only in one or two chapters as basic level and books published for MATLAB explains displaying different types of graphics for functions from command line, instead of using user interfaces while using MATLAB environment. The interface developer must find interface building tips and trips by his or her own research and trial and error methods. After developing the software, it is necessary to increase usage of the software. Since most of the computer users are not power user, user interfaces should be developed for ordinary users of MATLAB. In forthcoming years, comprehensive interface development books will be published and resource availability of MATLAB user interfaces expected to be increased.

4.1 CRISP- DM Methodology

When we were determining our methodology for the project, we examined many models then we selected commonly used one, CRISP- DM because of its applicability to our project.

CRISP-DM stands for CRoss Industry Standard Process for Data Mining. It is a data mining process model that describes commonly used approaches that expert data miners use to tackle problems. CRISP-DM is an industry-, tool-, and application-neutral model. This model encourages best practices and offers organizations the structure needed to realize better, faster results from data mining.

CRISP-DM organizes the data mining process into six phases: business understanding, data understanding, data preparation, modeling, evaluation, and deployment. These phases help organizations understand the data mining process and provide a road map to follow while planning and carrying out a data mining project.

Put simply, CRISP-DM is a comprehensive data mining methodology and process model that provides anyone—from novices to data mining experts—with a complete blueprint for conducting a data mining project. [5]

The Phases CRISP-DM Model:

- **Business Understanding**

This initial phase focuses on understanding the project objectives and requirements from a business perspective, and then converting this knowledge into a data mining problem definition, and a preliminary plan designed to achieve the objectives. Our aim is developing graphical user interface for widely used data-mining platform, so our project is independent from business understanding. We skip this phase in our project.

- **Data Understanding**

The data understanding phase starts with an initial data collection and proceeds with activities in order to get familiar with the data, to identify data quality problems, to discover first insights into the data, or to detect interesting subsets to form hypotheses for hidden information. The data understanding phase involves four steps, including the collection of initial data, the description of data, the exploration of data, and the verification of data quality. In the collection of initial data we get the data from many sources like text file or ODBC. In the description of data we examine issues such as the format of the data, the quantity of the data, the number of records and fields in each table, the identities of the fields, and any other surface features of the data. The data can be loaded inside the tool as fully or with limited numbers. After the data has been loaded, user can create list and modify them, or user can define the metadata of the variables and modify them.

- **Data Preparation**

The data preparation phase covers all activities to construct the final dataset (data that will be fed into the modeling tool(s)) from the initial raw data. Data preparation tasks are likely to be performed multiple times, and not in any prescribed order. Tasks include table, record, and attribute selection as well as transformation and cleaning of data for modeling tools. The five steps in data preparation are the selection of data, the cleansing of data, the construction of data, the integration of data, and the formatting of data.

In this phase, we prepare data for data- mining through preprocessing functions and improve the dataset. Our preprocessing design includes:

- Missing Value: Missing values in the data are assumed to be in the following formats; “Nan”, “NA”, “” or any format user defined. They can be handled as follows:
  1. Replace the value with mean, mode or median of the column
  2. Replace the value with user defined value.
  3. Remove row if the number of the missing values in the row equal or more than user defined number.

- Data Selection: Sampling operations can be performed with this tool as follows:
  1. Sampling
  2. Rest Sampling
  3. Stratified Sampling

- Data Transformation: This tool helps to prepare data to regression analysis.

- Data Discretization: Both numeric and categorical variables can be discretized.

- **Modeling**

In this phase, various modeling techniques are selected and applied, and their parameters are calibrated to optimal values. Typically, there are several techniques for the same data mining problem type. Some techniques have specific requirements on the form of data. Therefore, stepping back to the data preparation phase may be necessary. . Modeling steps include the selection of the modeling technique, the generation of test design, the creation of models, and the assessment of models.
In the selection of modeling technique step, we determine which models are used in the tool. There are association, classification, clustering, regression and sequention models placed in the tool. In the test design generation step, we determine that the user can select percentage of training and test data. At last step, model description will display statistics, ratios etc.

- **Evaluation**

At this stage in the project you have built a model (or models) that appears to have high quality, from a data analysis perspective. Before proceeding to final deployment of the model, it is important to more thoroughly evaluate the model, and review the steps executed to construct the model, to be certain it properly achieves the business objectives. A key objective is to determine if there is some important business issue that has not been sufficiently considered. At the end of this phase, a decision on the use of the data mining results should be reached.

We determine that, the outputs will be displayed in the new window within the spreadsheet format. It gives opportunity to user to edit the data. The results can be saved as .doc files.

- **Deployment**

Depending on the requirements, the deployment phase can be as simple as generating a report or as complex as implementing a repeatable data mining process. The key steps here are plan deployment, plan monitoring and maintenance, the production of the final report, and review of the project. In our project we skip this phase too we only consider the phases related with our project.

5. MATLAB AS A GLOBAL DEVELOPMENT ENVIRONMENT

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include

- Math and computation
- Algorithm development
- Data acquisition
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows users to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar noninteractive language such as C or Fortran.

The name MATLAB stands for *matrix laboratory*. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects. Today, MATLAB engines incorporate the LAPACK and BLAS libraries, embedding the state of the art in software for matrix computation.

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

MATLAB features a family of add-on application-specific solutions called *toolboxes*. Very important to most users of MATLAB, toolboxes allow you to *learn* and *apply* specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

5.1 The Matlab System

The MATLAB system consists of these main parts:

Desktop Tools and Development Environment

This is the set of tools and facilities that help users use MATLAB functions and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and Command Window, a command history, an editor and debugger, a code analyzer and other reports, and browsers for viewing help, the workspace, files, and the search path.

The MATLAB Mathematical Function Library

This is a vast collection of computational algorithms ranging from elementary functions, like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigen values, Bessel functions, and fast Fourier transforms.

The MATLAB Language

This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create large and complex application programs.

Graphics

MATLAB has extensive facilities for displaying vectors and matrices as graphs, as well as annotating and printing these graphs. It includes high-level functions for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level functions that allow you to fully customize the appearance of graphics as well as to build complete graphical user interfaces on your MATLAB applications.

The MATLAB External Interfaces/API

This is a library that allows you to write C and Fortran programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

KEY FEATURES of MATLAB

- High-level language for technical computing
- Development environment for managing code, files, and data
- Interactive tools for iterative exploration, design, and problem solving
- Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration
- 2-D and 3-D graphics functions for visualizing data
- Tools for building custom graphical user interfaces
- Functions for integrating MATLAB based algorithms with external applications and languages, such as C, C++, Fortran, Java, COM, and Microsoft Excel

5.1.1 Developing Algorithms and Applications

MATLAB provides a high-level language and development tools that let users quickly develop and analyze their algorithms and applications.

The MATLAB Language

The MATLAB language supports the vector and matrix operations that are fundamental to engineering and scientific problems. It enables fast development and execution.

With the MATLAB language, programming and developing algorithms are faster than with traditional languages due to absence of need to perform low-level administrative tasks, such as declaring variables, specifying data types, and allocating memory. In many cases, MATLAB eliminates the need for 'for' loops. As a result, one line of MATLAB code can often replace several lines of C or C++ code.

At the same time, MATLAB provides all the features of a traditional programming language, including arithmetic operators, flow control, data structures, data types, object-oriented programming (OOP), and debugging features.

MATLAB lets user execute commands or groups of commands one at a time, without compiling and linking, enabling user to quickly iterate to the optimal solution.

For fast execution of heavy matrix and vector computations, MATLAB uses processor-optimized libraries. For general-purpose scalar computations, MATLAB generates machine-code instructions using its JIT (Just-In-Time) compilation technology.

This technology, which is available on most platforms, provides execution speeds that rival those of traditional programming languages.

Development Tools

MATLAB includes development tools that help users implement their algorithms efficiently. These include the following:

MATLAB Editor - Provides standard editing and debugging features, such as setting breakpoints and single stepping.

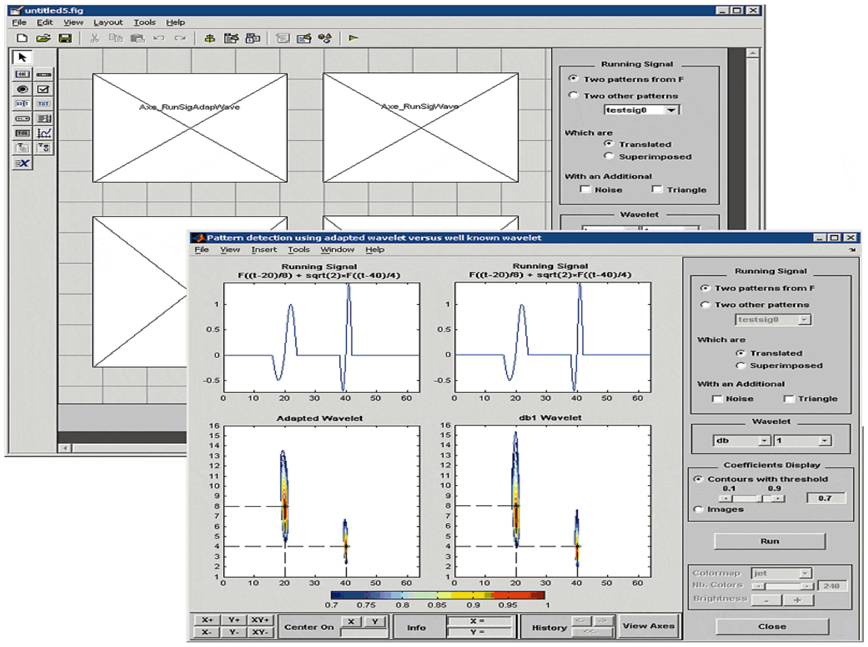
M-Lint Code Checker - Analyzes code and recommends changes to improve its performance and maintainability.

MATLAB Profiler - Records the time spent executing each line of code

Directory Reports - Scan all the files in a directory and report on code efficiency, file differences, file dependencies, and code coverage

Designing Graphical User Interfaces

The interactive tool GUIDE (Graphical User Interface Development Environment) can be used to lay out, design, and edit user interfaces. GUIDE lets you include list boxes, pull-down menus, push buttons, radio buttons, and sliders, as well as MATLAB plots and ActiveX controls. Alternatively, you can create GUIs programmatically using MATLAB functions.



GUIDE layout of a wavelet analysis GUI (top) together with the completed interface (bottom).

Figure 5.1: Wavelet analysis GUI

5.1.2 Analyzing and Accessing Data

MATLAB supports the entire data analysis process, from acquiring data from external devices and databases, through preprocessing, visualization, and numerical analysis, to producing presentation-quality output.

Data Analysis

MATLAB provides interactive tools and command-line functions for data analysis operations, including:

- Interpolating and decimating
- Extracting sections of data, scaling, and averaging
- Thresholding and smoothing
- Correlation, Fourier analysis, and filtering
- 1-D peak, valley, and zero finding
- Basic statistics and curve fitting
- Matrix analysis

Plot showing curve fitted to the monthly averaged atmospheric pressure differences between Easter Island and Darwin, Australia.

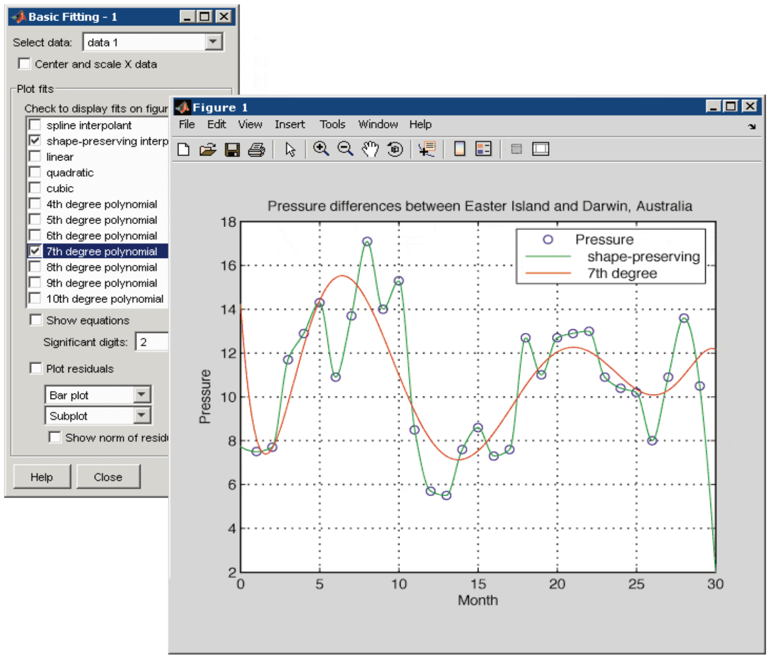


Figure 5.2: Plot- Curve, GUI

Data Access

MATLAB is an efficient platform for accessing data from files, other applications, databases, and external devices. It can be used to read data from popular file formats, such as Microsoft Excel; ASCII text or binary files; image, sound, and video files; and scientific files, such as HDF and HDF5. Low-level binary file I/O functions let users work with data files in any format.

Other applications and languages, such as C, C++, COM objects, DLLs, Java, Fortran, and Microsoft Excel can be called, and FTP sites and Web services can be accessed from MATLAB. Using the Database Toolbox helps accessing data from ODBC/JDBC-compliant databases.

Moreover MATLAB provides acquiring data from hardware devices, such as user's computer's serial port or sound card. Using the Data Acquisition Toolbox helps streaming live, measured data directly into MATLAB for analysis and visualization. The Instrument Control Toolbox (available separately) enables communication with GPIB and VXI hardware.

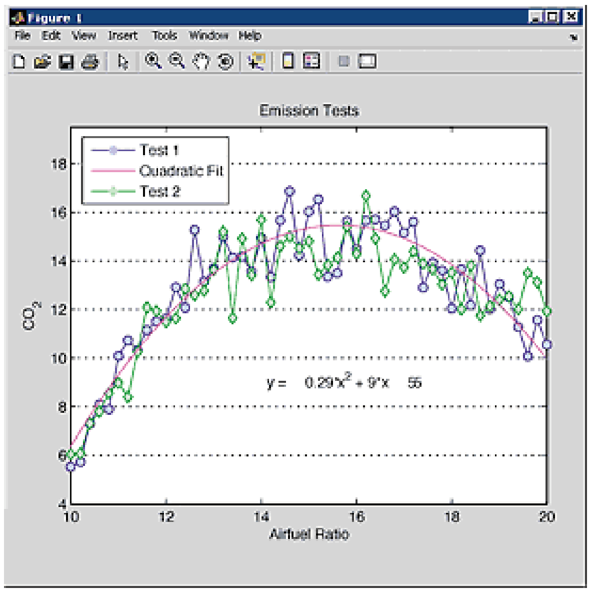
5.1.3 Visualizing Data

All the graphics features that are required to visualize engineering and scientific data are available in MATLAB. These include 2-D and 3-D plotting functions, 3-D volume visualization functions, tools for interactively creating plots, and the ability to export results to all popular graphics formats. You can customize plots by adding multiple axes; changing line colors and markers; adding annotation, LaTEX equations, and legends; and drawing shapes.

2-D Plotting

MATLAB provides functions for visualizing vectors of data with 2-D plotting functions that create:

- Line, area, bar, and pie charts
- Direction and velocity plots
- Histograms
- Polygons and surfaces
- Scatter/bubble plots
- Animations



Line plots of multiple engine emission test results, with a curve fitted to the raw data.

Figure 5.3: Line Plot, GUI

3-D Plotting and Volume Visualization

MATLAB provides functions for visualizing 2-D matrices, 3-D scalar, and 3-D vector data. these functions can be used to visualize and understand large, often complex, multidimensional data. Plot characteristics can be specified, such as camera viewing angle, perspective, lighting effect, light source locations, and transparency. 3-D plotting functions include:

- Surface, contour, and mesh
- Image plots
- Cone, slice, stream, and isosurface

A 3-D isosurface plot revealing the geodesic dome structure of a carbon-60 fullerene molecule.

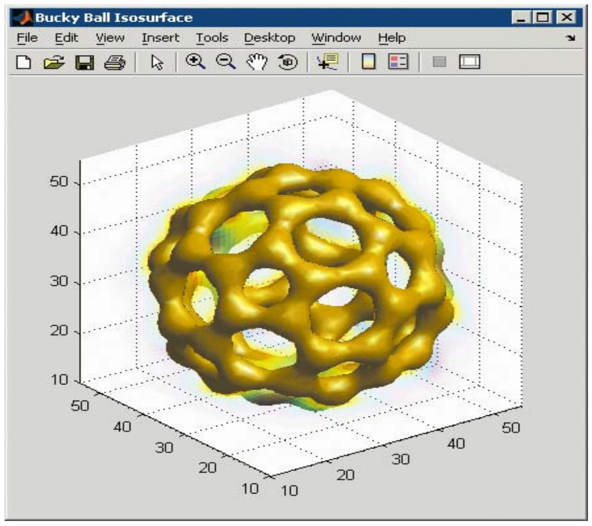


Figure 5.4: 3-D Isosurface Plot, GUI

Creating and Editing Plots Interactively

MATLAB provides interactive tools for designing and modifying graphics. From a MATLAB figure window, the following tasks can be performed:

- Drag and drop new data sets onto the figure
- Change the properties of any object on the figure
- Zoom, rotate, pan, and change camera angle and lighting
- Add annotations and data tips
- Draw shapes

- Generate an M-code function that can be reused with different data

Importing and Exporting Graphic Files

MATLAB lets you read and write common graphical and data file formats, such as GIF, JPEG, BMP, EPS, TIFF, PNG, HDF, AVI, and PCX. As a result, you can export MATLAB plots to other applications, such as Microsoft Word and Microsoft PowerPoint, or to desktop publishing software. Before exporting, you can create and apply style templates, covering characteristics such as layout, font, and line thickness, to meet publication specifications.

5.1.4 Performing Numeric Computation

MATLAB contains mathematical, statistical, and engineering functions to support all common engineering and science operations. The core math functions use the LAPACK and BLAS linear algebra subroutine libraries and the FFTW Discrete Fourier Transform library. Because these processor-dependent libraries are optimized to the different platforms that MATLAB supports, they execute faster than the equivalent C or C++ code.
MATLAB provides the following types of functions for performing mathematical operations and analyzing data:

- Matrix manipulation and linear algebra
- Polynomials and interpolation
- Fourier analysis and filtering
- Data analysis and statistics
- Optimization and numerical integration
- Ordinary differential equations (ODEs)
- Partial differential equations (PDEs)
- Sparse matrix operations

MATLAB can perform arithmetic on a wide range of data types, including doubles, singles, and integers.

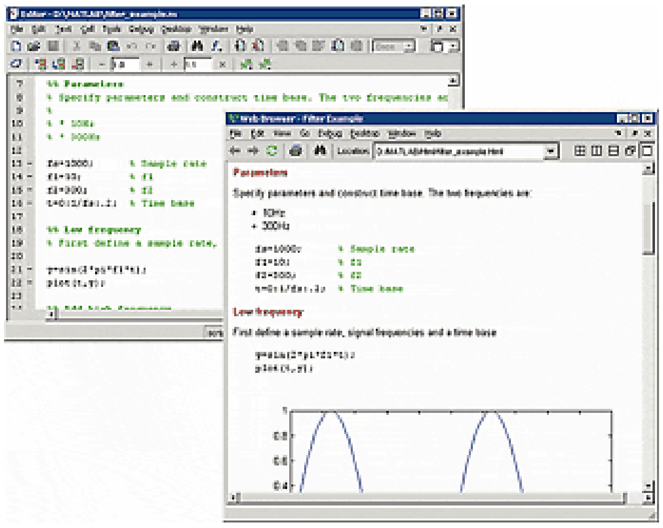
Add-on toolboxes (available separately) provide specialized mathematical computing functions for areas including signal processing, optimization, statistics, symbolic math, partial differential equation solving, and curve fitting.

5.1.5 Publishing Results and Deploying Applications

MATLAB provides a number of features for documenting and sharing users' work. MATLAB code can be integrated with other languages and applications and MATLAB algorithms and applications can be deployed as stand-alone programs or software modules.

Publishing Results

MATLAB lets users export their results as plots or as complete reports. Users can export plots to all popular graphics file formats and then import the plots into other packages, such as Microsoft Word or Microsoft PowerPoint. Using the MATLAB Editor helps automatically publishing MATLAB code in HTML, Word, LaTEX, and other formats.



```
%% Parameters
% Specify parameters and construct time base. The two frequencies ar
%
% * 10Hz
% * 300Hz

fs=1000;        % Sample rate
f1=10;          % f1
f2=300;         % f2
t=0:1/fs:.2;    % Time base

%% Low frequency
% First define a sample rate,

y=sin(2*pi*f1*t);
plot(t,y);
```

Parameters

Specify parameters and construct time base. The two frequencies are:

- 10Hz
- 300Hz

```
fs=1000;        % Sample rate
f1=10;          % f1
f2=300;         % f2
t=0:1/fs:.2;    % Time base
```

Low frequency

First define a sample rate, signal frequencies and a time base

```
y=sin(2*pi*f1*t);
plot(t,y);
```

M-code program (left) published to HTML (right) using the MATLAB Editor. Results output to the command window or to plots are captured and included, and the comments are turned into section headings and body text in the HTML.

Figure 5.5: M-code Program, HTML view

Integrating MATLAB Code with Other Languages and Applications

MATLAB provides functions for integrating C and C++ code, Fortran code, COM objects, and Java code with users' applications. DLLs, Java classes, and ActiveX controls can be called. Using the MATLAB engine library helps to call MATLAB from C, C++, or Fortran code.

Deploying Applications
MATLAB supports creating algorithm in its environment and distributing it to other MATLAB users as M-code. Using the MATLAB Compiler (available separately) helps deploying algorithm, as a stand-alone application or as a software module, to users who do not have MATLAB.[6]

5.2 Interface Usage In Matlab

MATLAB is originally entirely a command-line environment, but through the needs of the users and developments of programmers, functionalities of graphical interfaces have developed. GUI will start to use widely instead of command line. GUI is preferable because of making programs easier to use by providing them with a consistent appearance and with intuitive controls like pushbuttons, list boxes, sliders, menus, and so forth. GUI behaves in an understandable and predictable manner and helps user to know what to expect when he or she performs an action.

Nowadays, GUI developments focus on specific functions like association, clustering, computational aids in engineering, there is not any tool for full functions like SPSS. However due to open source features of MATLAB, the program provides to program and develop GUI in different fields such as engineering, data mining, and bioinformatics.

MATLAB GUI is similar to RAD such as C++ builder and VB, interfaces can work across platforms. Moreover MATLAB GUI can perform most functions as traditional GUI through tricks.

Nonetheless, interface usage in MATLAB is developing recently so the features of GUI do not meet all needs of users. MATLAB GUI is not as flexible to program. Cross platform appearance may not be the same. Most of the time tricks and unfriendly techniques must be used for programming because of insufficient features. [7]

5.2.1 What is GUIDE?

GUIDE is the MATLAB Graphical User Interface development environment that provides a set of tools for creating graphical user interfaces (GUIs). These tools greatly simplify the process of designing and building GUIs. GUIDE allows a programmer to layout the GUI by clicking and dragging GUI components such as panels, buttons, text fields, sliders, menus, and so on into the layout area. Once the components are in place, the programmer can edit their properties: name, color, size, font, text to display, and so forth.

GUIDE automatically generates an M-file that controls how the GUI operates. The M-file initializes the GUI and contains a framework for all the GUI callbacks which the commands that are executed when a user clicks a GUI component. M-file editor provides to add code to the callbacks to perform the functions when it requires.

The three principal elements required to create a MATLAB Graphical User Interface are:

1. Components. Each item on a MATLAB GUI (pushbuttons, labels, edit boxes, etc.) is a graphical component. The types of components include:

- Graphical controls (pushbuttons, edit boxes, lists, sliders, etc.)- created by the function uicontrol
- Static elements (frames and text strings)- created by the function uicontrol
- Menus- created by the functions uimenu and uicontextmenu
- Axes- created by the function axes

2. Figures. A figure, which is a window on the computer screen, must be used to arrange the components of a GUI. MATLAB command line environment uses figures to plot data so figures are not new within GUI elements. However, with the GUIDE empty figures can be created with the function figure and can be used to hold any combination of components.

3. Callbacks. They are control actions that perform an action if a user clicks a mouse on a button or types information on a keyboard. The MATLAB program must respond to each event if the program is to perform its function. For example, if a user clicks on a button, that event must cause the MATLAB code that implements the function of the button to be executed. The code executed in response to an event is known as a call back. There must be a callback to implement the function of each graphical component on the GUI. [8]

5.2.2 Components of MATLAB GUI

Component	Icon	Description
Push Button	OK	Push buttons generate an action when clicked. For example, an **OK** button might apply settings and close a dialog box.
Toggle Button	TGL	Toggle buttons generate an action and indicate whether they are turned on or off. When toggle button is clicked, it appears depressed, showing that it is on.
Radio Button	●	Radio buttons are similar to check boxes, but radio buttons are typically mutually exclusive within a group of related radio buttons. That is, when one button is selected the previously selected button is deselected.
Check Box	☑	Check boxes can generate an action when checked and indicate their state as checked or not checked. Check boxes are useful when providing the user with a number of independent choices, for example, displaying a toolbar.
Edit Text	EDIT	Edit text components are fields that enable users to enter or modify text strings.
Static Text	TXT	Static text controls display lines of text. Static text is typically used to label other controls, provide directions to the user, or indicate values associated with a slider. Users cannot change static text interactively.
Slider		Sliders accept numeric input within a specified range by enabling the user to move a sliding bar, which is called a slider or thumb. Users move the slider by clicking the slider and dragging it, by clicking in the trough, or by clicking an arrow. The location of the slider indicates the relative location within the specified range.
List Box		List boxes display a list of items and enable users to select one or more items.
Pop-Up Menu		Pop-up menus open to display a list of choices when users click the arrow.
Axes		Axes enable your GUI to display graphics such as graphs and images. Like all graphics objects, axes have properties that can set to control many aspects of its behavior and appearance.
Panel	T	Panels arrange GUI components into groups. By visually grouping related controls, panels can make the user interface easier to understand. A panel can have a title and various borders. Panel children can be user interface controls and axes as well as button groups and other panels. The position of each component within a panel is interpreted relative to the panel. If user moves the panel, its children move with it and maintain their positions on the panel.
Button Group	T	Button groups are like panels but are used to manage exclusive selection behavior for radio buttons and toggle buttons.
ActiveX Component	X	ActiveX components enable to display ActiveX controls in your GUI. They are available only on the Microsoft Windows platform. An ActiveX control can be the child only of a figure, i.e., of the GUI itself. It cannot be the child of a panel or button group.

5.2.3 The Layout Editor

When GUI in GUIDE is opened, it is displayed in the Layout Editor, which is the control panel for all of the GUIDE tools. The following figure shows the Layout Editor with a blank GUI template. GUI can be laid out by dragging components, such as push buttons, pop-up menus, or axes, from the component palette, at the left side of the Layout Editor, into the layout area. The top of the window has a toolbar with a series of useful *tools* that allow the user to distribute and align GUI components, modify the properties of GUI components, add menus to GUIs, and so on.

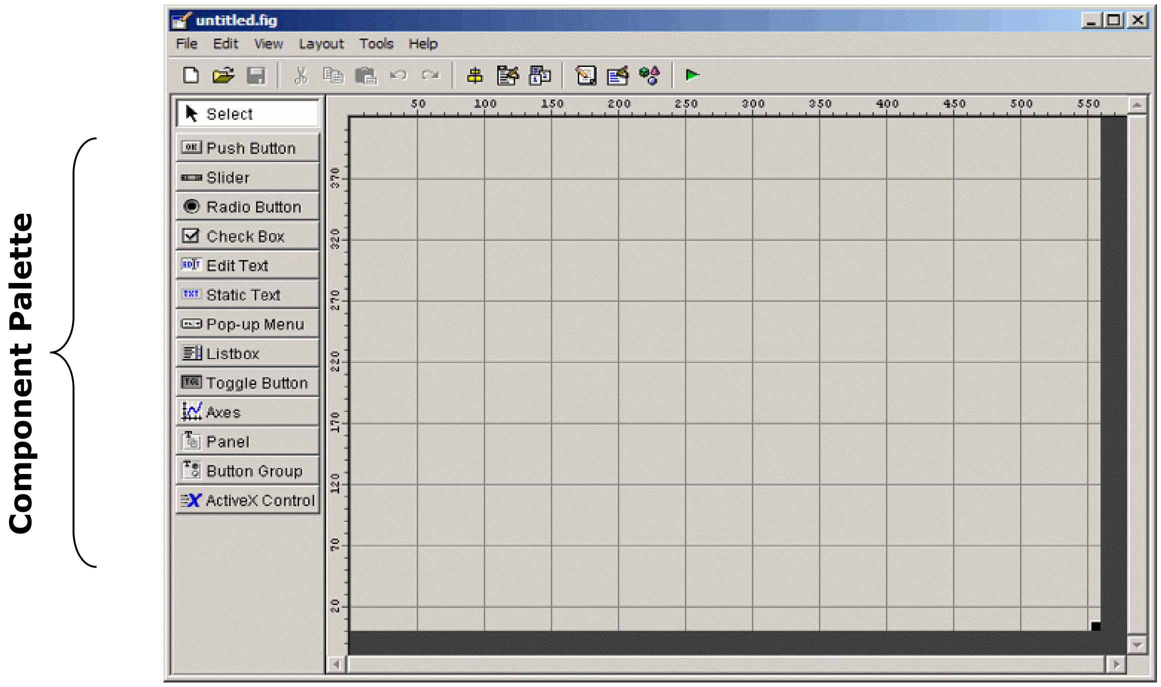


Figure 5.6: Layout Editor

After adding components to layout area their properties can be changed with the help of the Property Inspector. The name, size, color, font and many other features can be changed through the inspector.

Moreover, menus can be created for user interface with using the Menu Editor. There are two types of menus, menus for the menu bar and context menus. The Menu Editor provides to add submenus to menus and modify when required.

The other feature is Alignment Tool that facilitates alignment and distribution of components with respect to each other and adjusts the spacing between selected objects.

In MATLAB there is not any toolbar tool, user can use codes to create toolbar or can add prepared toolbars for specified functions, such as plot edit tool bar.

5.2.4 Interface Development Tips

- Callbacks should be short and call functions.

When MATLAB evals a string, it must interpret it statement by statement. MATLAB must do the same with a script m-file. It behaves differently with a function: the first time MATLAB encounters the function, it compiles it. MATLAB does not re-interpret the function on subsequent calls. Thus of the three possibilities (function m-file, script m-file, and long string), making the callback a function m-file is by far the fastest, especially if the callback is long.

- Switchyard concept of GUI programming.

This involves coding an application with a GUI as a single function. Inside the function is a large if-then-else construct which executes portions of the function based on an input switch. Callback functions of UI objects (uicontrols and uimenus) and mouse Button functions (down/motion/up) simply call the function with the appropriate input argument. Callbacks are therefore very short and editing them is easier than changing multi-line string definitions. A call to the function with no input argument could by default start the program. Program specific data and UI object handles are stored either as globals to the function or in the UserData of the objects. This makes the application "clear-proof". For an example of this sort of programming, see "sigdemo1".

- Draw moveable or changing objects with the 'erasemode' property set to 'xor' or background'. This prevents reentering the axes when changing these objects. 'erasemode' is a property of all objects which are children of axes (line, text, surface, image, patch).

- Set 'drawmode' (an axes property) to 'fast'. This prevents MATLAB from sorting 3D objects, which can speed things up significantly. The side-effect is that 3D surface plots will not work in this mode.

- Set 'backingstore' (a figure property) to 'off'. This should give roughly a factor of two speed-ups in normal drawing but turns off the instantaneous update that normally ocurrs when windows are uncovered.

- Set 'nextplot' (a figure property) to 'new' when creating a GUI. That way when you make plots from the command window they don't appear in the axes of the GUI's figure window.

- Wherever possible, recycle figure windows by using the 'visible' property of figures instead of creating and destroying them. When done with the window, set 'visible' to 'off'; when you need the window again, make any changes to the window and THEN set 'visible' to 'on'. Creating figure windows involves much more overhead than unhiding them.

- Set a uicontrol's style upon creation. E.g. uicontrol ('Style', 'edit') is much faster than h=uicontrol; set (h, 'Style', 'edit').

- Set a figure's position upon creation. E.g. figure ('Pos',[10 10 200 100]) is much faster than f=figure; set (f, 'Pos',[10 10 200 100]). [9]

6. DESIGN

6.1 Menu Structure

1 File

- 1.1 Read
  - 1.1.1 Read from File
  - 1.1.2 Read from ODBC
- 1.2 Save
- 1.3 Exit

2 Data

- 2.1 Run Matlab Command
- 2.2 Create List
- 2.3 Add List
- 2.4 Remove List
- 2.5 Set Meta
- 2.6 Add Meta
- 2.7 List Meta
- 2.8 Descriptives

3 Preparation

- 3.1 Missing Value
- 3.2 Sampling
- 3.3 Transformation
- 3.4 Discretization

4 Functionality

- 4.1 Association
- 4.2 Clustering
- 4.3 Classification
  - 4.3.1 Decision Tree
    - 4.3.1.1 CHAID

4.3.1.2 EX-CHAID

4.3.2 Neural Network

4.4 Regression

5 Graphics

5.1 Bar_Chart

5.2 Pie_Chart

5.3 Line_Graph

5.4 Box_Plot

5.5 Scatter_Plot

6.2 Menu Commands

1 File

1.1 Read

1.1.1 Read from File: Imports data from text files into grid format at main window,

1.1.2 Read from ODBC: Imports data from .mdb files into grid format at main window.

1.2 Save: Saves the current data to .mat files.

1.3 Exit: Exits the tool.

2 Data

2.1 Run Matlab Command: Opens window to write command and run it without using Matlab Command Window.

2.2 Create List: Creates lists of variables with user defined names.

2.3 Add List: Adds variables to created lists.

2.4 Remove List: Removes variables from created lists.

2.5 Set Meta: Sets metadata value of a variable.

2.6 Add Meta: Add new values to the metadata of a variable.

2.7 List Meta: Lists metadata values of variables.

2.8 Descriptives: Displays statistics of selected lists.

3 Preparation

3.1 Missing Value: Replaces the missing values in the data set or removes rows according to number of missing values in the row.

3.2 Sampling: Selects samples from specified data set with selected sampling method.

3.3 Transformation: Transforms the columns into specified ranges.

3.4 Discretization: Discretizes the values according to given intervals.

4 Functionality

4.1 Association: Extracts association rules from specified data set.

4.2 Clustering: Performs clustering operation on the defined data set using the K-Means algorithm.

4.3 Classification

4.3.1 Decision Tree

4.3.1.1 CHAID: Builds decision trees using the CHAID algorithm.

4.3.1.2 EX-CHAID: Builds decision trees using the EX- CHAID algorithm.

4.3.2 Neural Network: Builds neural networks.

4.4 Regression: Performs multiple linear regression with bootstrap and cross validation.

5 Graphics

5.1 Bar_Chart: Draws bar chart according to defined variables.

5.2 Pie_Chart: Draws pie chart according to defined variables.

5.3 Line_Graph: Draws line graph according to defined variables.

5.4 Box_Plot: Draws box plot according to defined variables.

5.5 Scatter_Plot: Draws scatter plot according to defined variables.

6.3 Screen Designs and Flows

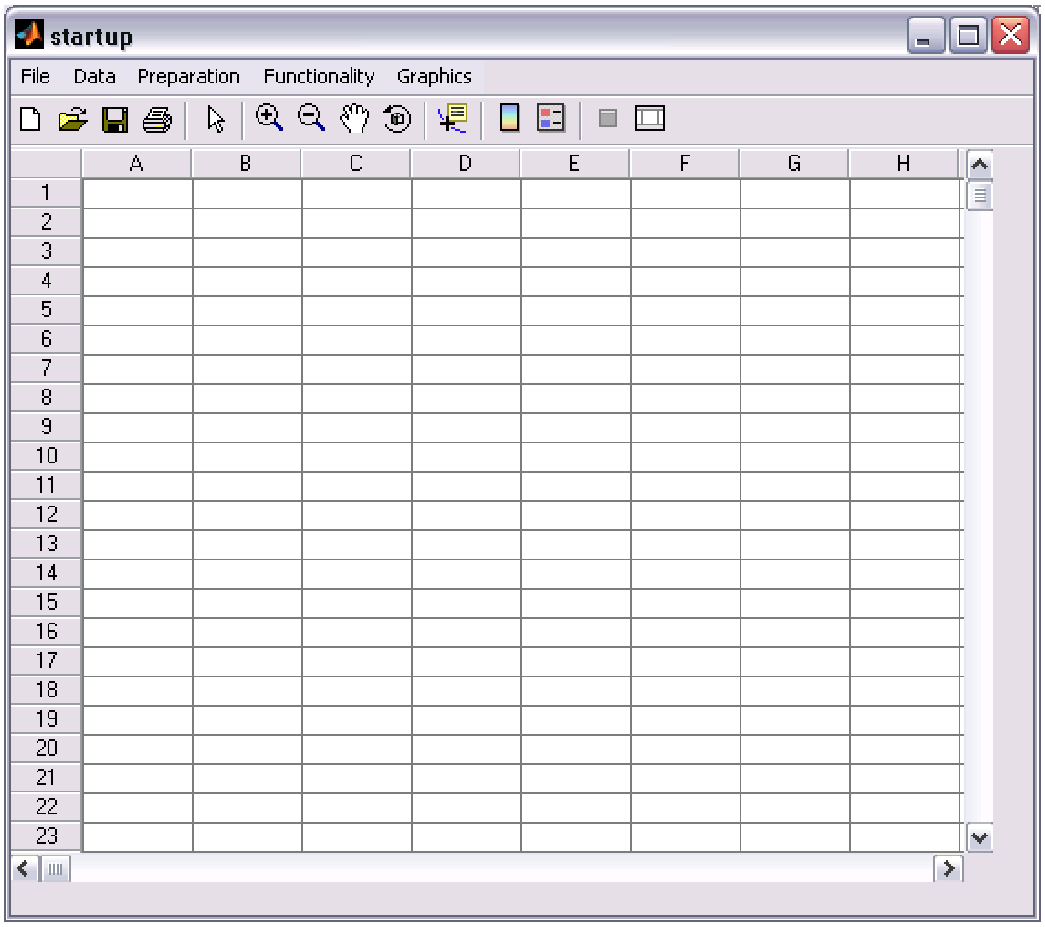


Figure 6.1: Main Screen of the tool

Figure 6.1 shows the main screen of the tool. There is spreadsheet format to display data set efficiently. When the user selects one of the Read functions from File menu the data is imported to the tool and displayed in the grids. Each variable is shown at top as a column name, and entries of variables are listed under the variables.

6.3.1 FILE

6.3.1.1 Read

6.3.1.1.2 Read_From_File

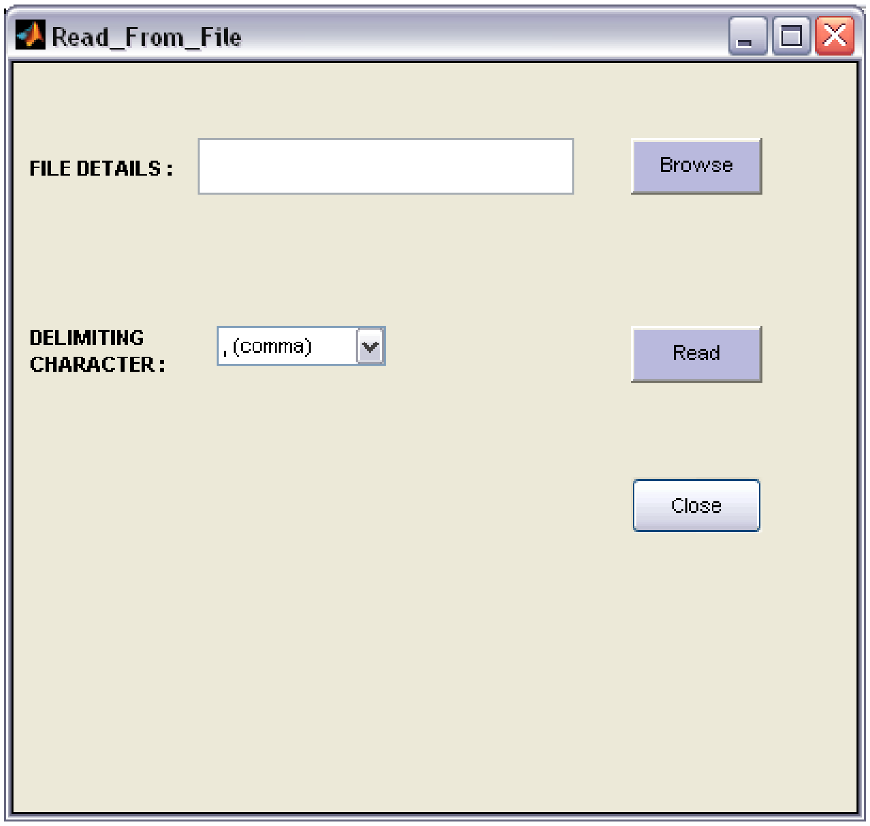


Figure 6.2: Read_From_File

When the user selects Read/ Read_From_File from File menu, Figure 7.2 appears.

Browse: In this window, the user browses the text files to place the data source.

Delimiter Character: After that s/he specifies the delimiter character of the data. It can be comma (,), semi-colon (;), colon (:), full-stop (.) or space.

After making selection the user clicks to Read button.

Constraints:

1. The user has to define a data source.

Otherwise when Read button is clicked error message occur.

2. The user has to select delimiter.

Otherwise the system displays all variables and attributes in one column.

6.3.1.1.2 Read_From_ODBC

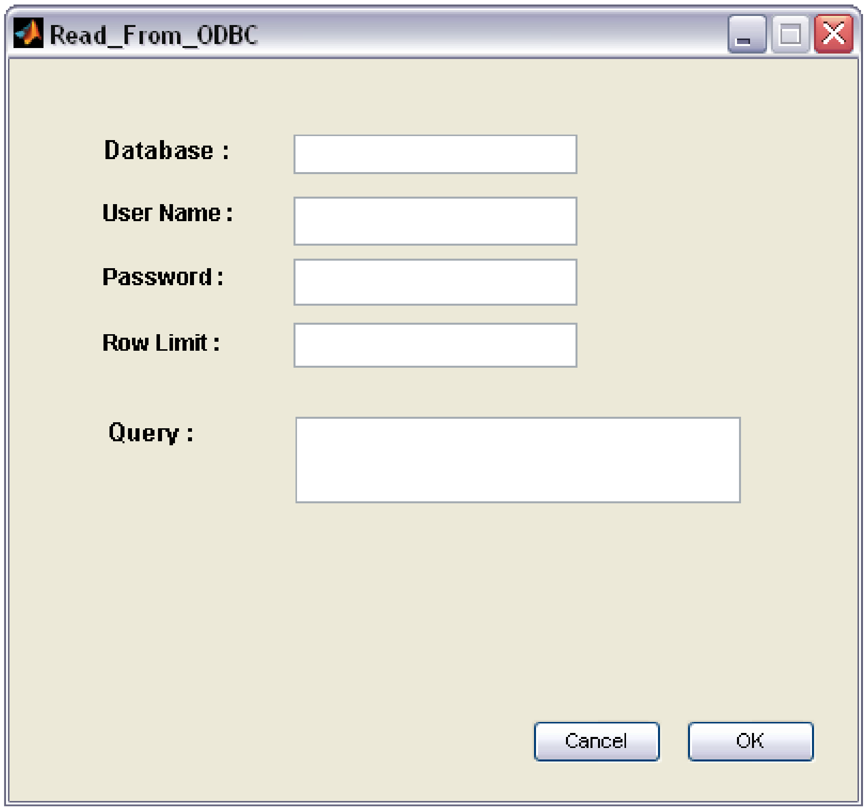


Figure 6.3: Read_From_ODBC

When the user selects "Read_from_ODBC" under the Read at File menu, Figure 6.3 appears.

Database: User enters the name of the database to text box.

User Name: If the file is protected the user enters the user name in this field, otherwise s/he skip this field.

Password: If the file is protected the user enters the password to reach data. If it is not, user leaves blank this field.

Row Limit: This field provides user to limit the data set. User can import file with limited number of records.

Query: It is the SQL query box, user can write any query and get data in desired format. The query text can be basically:

select * from "SURGERY CATALOG"

Constraints:

1. The user has to enter database and query, otherwise, the "OK" button is not worked.

Use Case: **Read_From_File**
ID: 1.1.1
Actor(s): User
Preconditions:
Flow of Events: **1.** The user selects “Read” and “Read_from_File” from File menu. **2.** At the opened screen, user enters file path or uses “Browse” button to select file. **3.** The user selects delimiter character from popup menu. **4.** The user clicks “Read” button.
Post Conditions: **1.** The data set is shown in the spreadsheet at the main screen.

Figure 6.4: Use Case- Read_From_File

Use Case: **Read_From_ODBC**
ID: 1.1.2
Actor(s): User
Preconditions:
Flow of Events: **1.** The user selects “Read” and “Read_from_ODBC” from File menu. **2.** At the opened screen, user enters the name of the database. **3.** If the file is protected 3.1 The user enters user name 3.2 The user enters password. **4.** The user can determine row limit of the data. **5.** The user writes query. **6**. The user clicks “OK” button.
Post Conditions: **1.** The data set is shown in the spreadsheet at the main screen.

Figure 6.5: Use Case- Read_From_ODBC

6.3.1.2 Save

Use Case: **Save**
ID: 1.2
Actor(s): User
Preconditions: The data set is imported.
Flow of Events: **1.** The user selects “Save” from the File menu.   **2.** If the user did not previously save the project   2.1 <include> Use Case 1.3   **3.** The system saves the current data in the spreadsheet.
Post Conditions:

Figure 6.6: Use Case- Save

6.3.1.3 Save As

Use Case: **Save As**
ID: 1.3
Actor(s): User
Preconditions: The data set is imported.
Flow of Events: **1.** The user selects “Save As…” from the File Menu.   **2.** The system prompts for the path where the data set will be saved.   **3.** The user specifies the path.   **4.** the system saves the current situation of the project to the specified path.
Post Conditions:

Figure 6.7: Use Case- Save

6.3.1.4 Exit

Use Case: **Exit**
ID: 1.4
Actor(s): User
Preconditions:
Flow of Events: **1.** The user selects "Exit" from the File menu. **2.** The system asks the user if he wants to quit or not. **3.** If the user clicks "Yes" button, the system closes the tool **4.** If the user clicks "No" button, the system returns to main screen.
Post Conditions:

Figure 6.8: Use Case- Exit

6.3.2 DATA

6.3.2.1 Run Matlab Command

When the user selects “Run Matlab Command” from the Data menu, Figure 6.9 appears. This window is used to write any command to MATLAB without using Command Window of MATLAB.

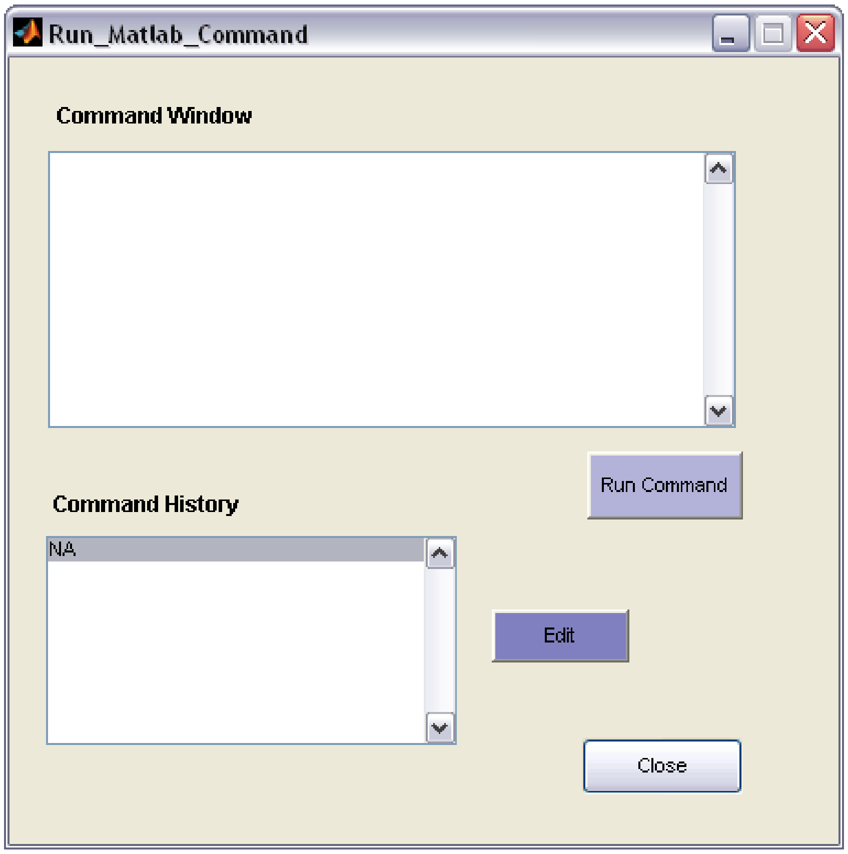


Figure 6.9: Run Matlab Command

While using the tool, the user may need to use command features of MATLAB, closing the tool, writing the commands to MATLAB and executing them complicates the work of user.

This window works as main screen of MATLAB, when the user enters any command and use Run Matlab button the system behaves as the command is written in command window of MATLAB.

Command Window: The user enters any command to this box. The user can use this field to do whatever he wants to do with using MATLAB commands. The system writes commands MATLAB command history too, when user closes the tool he can reach written commands from main screen of MATLAB. The user can create many commands in one time.

Run Command: When the user clicks on this button, the written commands in the Command Window is executed. In addition the Command Windows is cleaned.

Command History: After the use of Run Command button, the commands are displayed in this list box. While the MATLAB open, all the commands created by Run Matlab Command screen, is displayed in this field.

Edit: If the user wants to edit the enlisted commands, s/he selects it from Command History and uses Edit button. The command is transferred to Command Window.

Close: The user closes the window using Close button. The user can make countless application in the window until s/he uses Close button.

Use Case: **Run Matlab Command**
ID: 2.1
Actor(s): User
Preconditions:
Flow of Events: **1.** The user enters the commands to Command Window. **2.** The user clicks on the "Run Command" button. **3.** The system displays written commands in the Command History list box **4.** The system cleans the Command Window box. **5.** If the user wants to edit enlisted commands; 5.1 He selects the command from Command History 5.2 He uses "Edit" button. 5.3 The system displays the command in Command Window edit box. **6.** The user clicks on "Close" button to quit the window.

Figure 6.10: Use Case- Run Matlab Command

6.3.2.2 Create List

When the user selects "Create List" under the Data menu, Figure 6.11 appears.

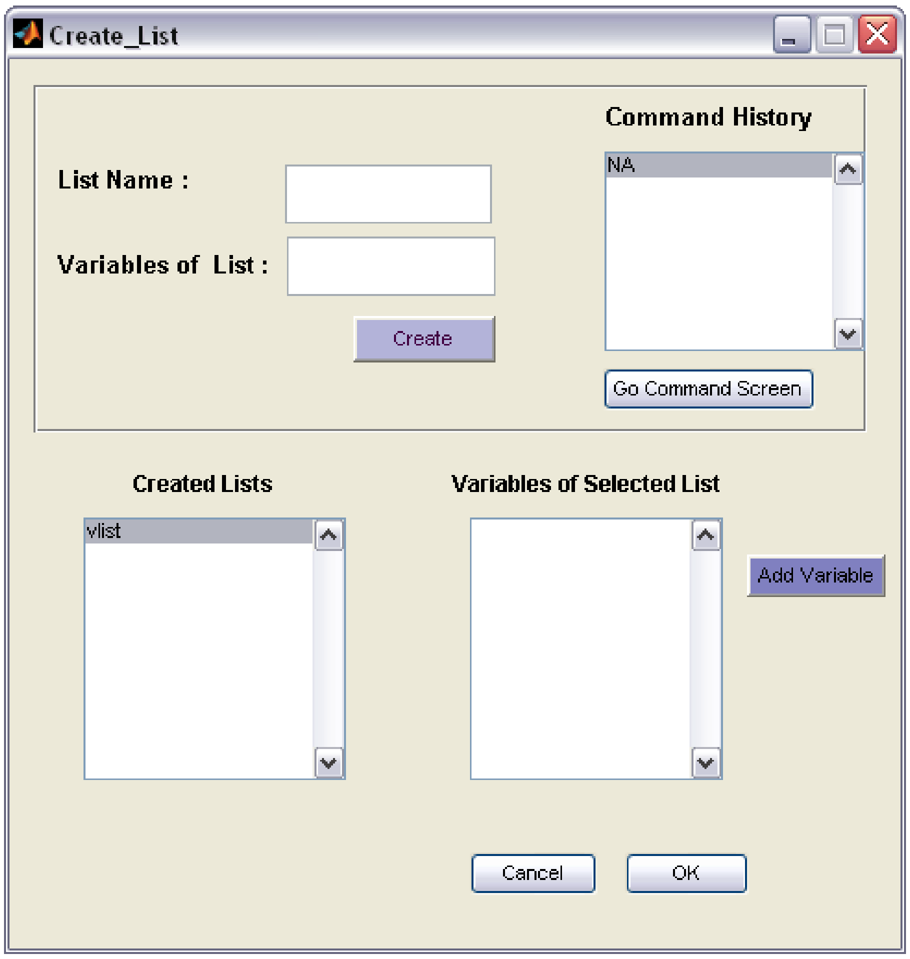


Figure 6.11: Create List

The tool provides to create lists with variables from imported data set and variables created by Run Matlab Command. These lists can be used in preparation of data and data analysis.

List Name: The user enters a name for the list.

Variables of List: Then, the user writes the variables that will be in the list. The user can select variables from “Variables of Selected List” and using “Add Variable” button add them to field.

Command History: This field provides user to see created variables using Command Window. While entering the variables of the list, it helps user to see all created variables and write them to field correctly.

Create: After entering the name and the variables of the user clicks on “Create” button to create list.

Created Lists: After using “Create” button, the created list appears in the “Created Lists” list box. When the window is opened the variable list “vlist” is displayed in the box, it includes all variables of data imported from text file or ODBC.

Variables of Selected List: When user clicks on any list name in “Created Lists”, the list of the variables in the selected list is displayed in this field.

Add Variable: While entering the variables of the list user can add them from “Variables of Selected List”. The user selects the variable and uses “Add Variable” button and the variable is added to “Variables of List” box.

Go Command Screen: While creating lists, the user may want to create new variables to add list. When s/he clicks on the “Go Command Screen” button the “Run Matlab Command” in Figure 6.8 screen appears. The user adds commands in this window and when he closes the window, the created commands are displayed in “Command History” in Create List screen.

Use Case: **Create List**
ID: 2.2
Actor(s): User
Preconditions:
Flow of Events: **1.** The user enters the name of the list. **2.** The user enters the variables of list. 2.1 The user can write the variables. 2.2 The user can select variables from list 2.2.1 The user selects list from Created Lists 2.2.2 Then he selects variable from Variables of Selected List. 2.2.3 The user clicks on Add Variable button. 2.2.4 The system adds variable to Variables of List box. **3.** If the user wants to create new variables 3.1 He clicks the Go Command Screen button and <include> Run Matlab Command 3.2 The system displays created variables in Command History list box. **4.** The user clicks on Create button **5.** The system adds the list to Created Lists
Post Conditions: The created lists updated.

Figure 6.12: Use Case- Create List

6.3.2.3 Add List

When the user selects "Add List" from the Data menu, the Figure 6.13 appears.

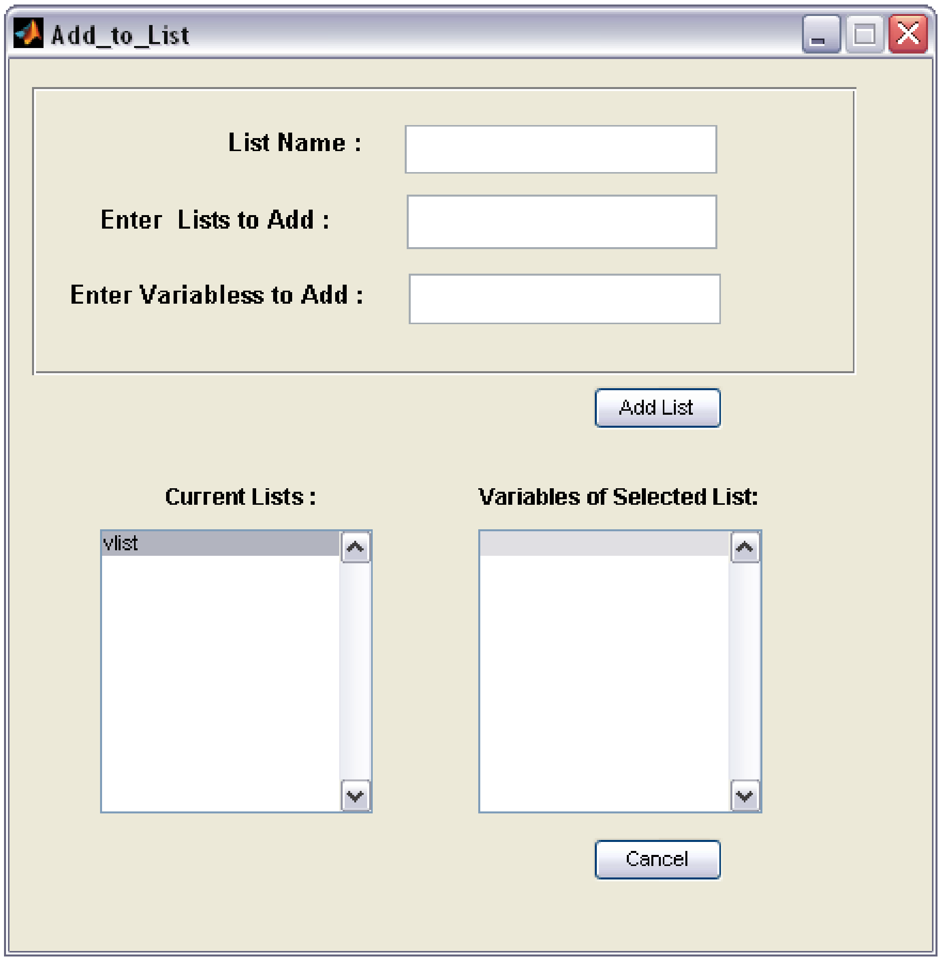


Figure 6.13: Add List

This window is used to add new variables to existing lists. In this window, the user can merge lists and give new names to them too.

List Name: The user enters the list name to this field. The name can be the current name of the created list or new name to merge lists in new list.

Enter Lists to Add: The user enters names of the lists to add to list. The user can add the list to current list or can create new list with different name.

Enter Variables to Add: The user enters variables to add the list defined at "List Name".

Add List: After filling required fields, the user clicks on "Add List" button for execution.

Current Lists: The user see created lists in this list box. When the "Add List" button is used the system updates the list box. Created lists and modifications can be seen immediately.

Variables of Selected List: After selecting the list from "Current Lists", the variables of it is shown in this list box. The user can see variables of each list though this field. It is updated when the "Add List" button is used. When the lists merged, the system puts their names in this field instead of variables of each list.

Use Case: **Add List**
ID: 2.3
Actor(s): User
Preconditions: The lists should be created in “Create List” screen.
Flow of Events: **1.** The user enters the name of the list.   1.1 If the user adds variable to current list, enters name of the existing list.   1.2 If the user merges lists under the new name, he enters new name to List Name.   **2.** If the user adds lists to list written in List Name, enters the names of the lists.   **3.** If the user adds variables to current list enters the variable names.   **4.** The user clicks on Add List button   **5.** The system updates both Current Lists and Variables of Selected List fields.   **6.** If the user wants to see new forms of lists;   6.1 He selects the list from Current Lists.   6.2 The system displays variables in the Variables of Selected List.
Post Conditions: The created lists updated.

Figure 6.14: Use Case- Add List

6.3.2.4 Remove List

When the user selects "Remove List" under the Data Menu, the Figure 6.15 appears.

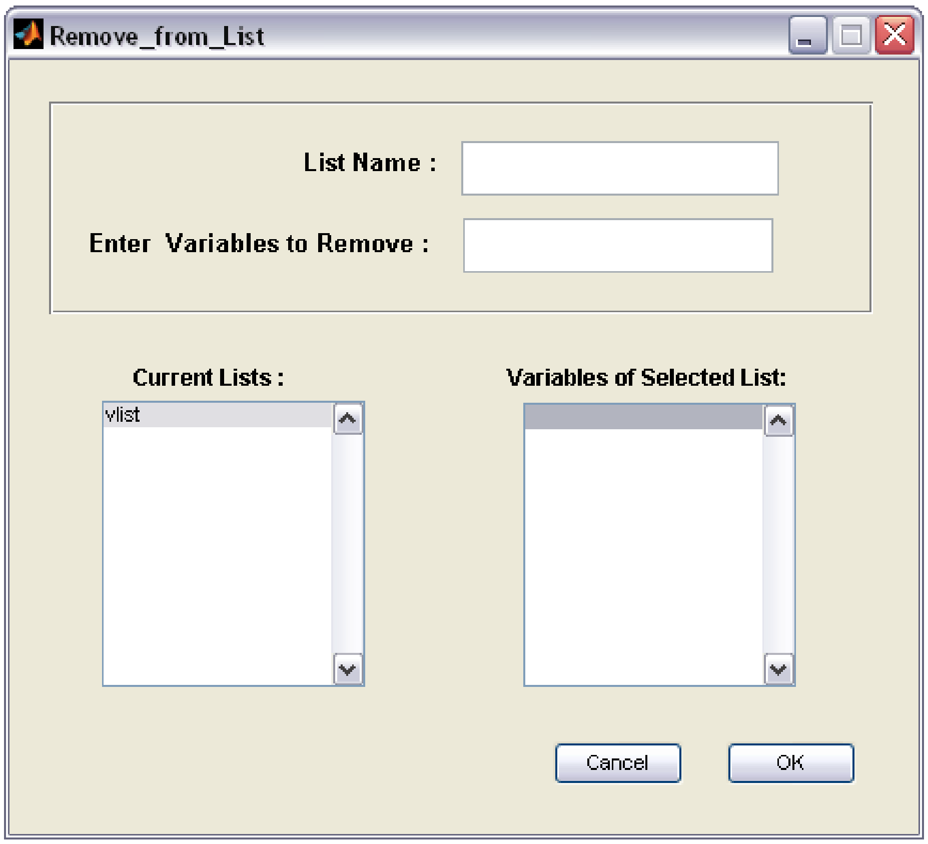


Figure 6.15: Remove From List

List Name: The user enters the name of the list to remove variables from this list.

Enter Variables to Remove: Then, the user enters the variables to "enter Variables to Remove" field. In this point, the content of lists formed with merge of lists is perceived as variables. Each list behaves as variable.

OK: When the user clicks on OK button the variable is removed from the list.

Current Lists: The user see created lists in this list box. When the "OK" button is used the system updates the list box. Modifications can be seen immediately.

Variables of Selected List: After selecting the list from “Current Lists”, the variables of it is shown in this list box. The user can see variables of each list though this field. It is updated when the “OK” button is used. The entered variables to “Enter Variables to Remove” are removed from the lists and disappear from this field.

Use Case: **Remove List**
ID: 2.4
Actor(s): User
Preconditions: The lists should be created in “Create List” screen.
Flow of Events: 1. The user enters the name of the list to remove variable from this list. 2. The user enters the variables to remove from list. 3. The user clicks on OK button. 4. The system updates both Current Lists and Variables of Selected List fields. 5. If the user wants to see new forms of lists; 5.1 He selects the list from Current Lists. 5.2 The system displays variables in the Variables of Selected List.
Post Conditions: The created lists updated.

Figure 6.16: Use Case- Remove List

6.3.2.5 Set Meta

When the user selects “Remove List” under the Data Menu, the Figure 6.17 appears.

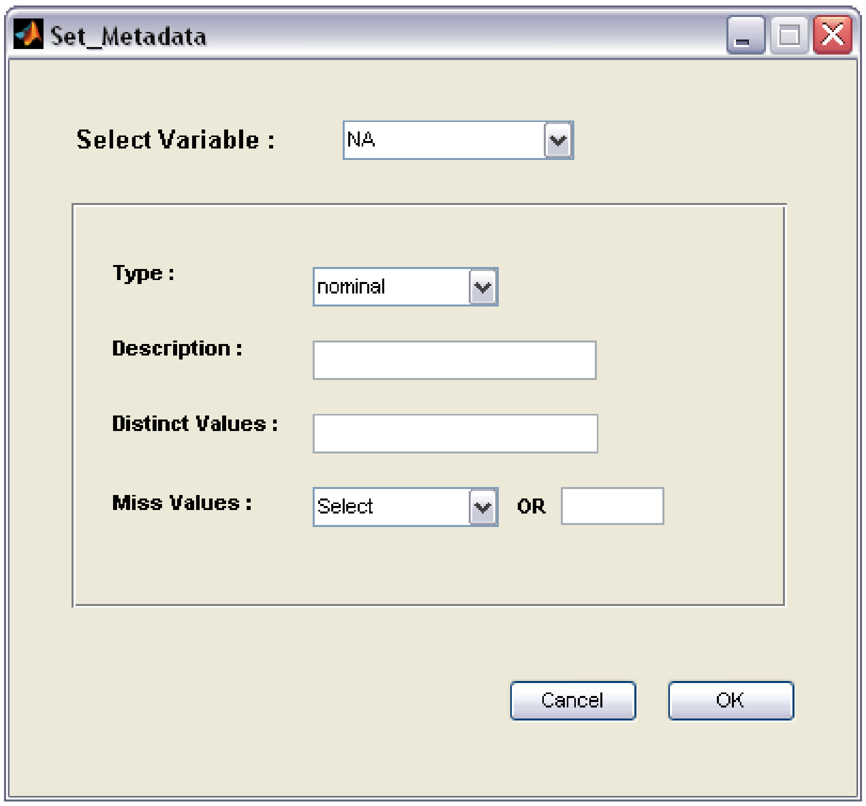


Figure 6.17: Set_Metadata

This window is used to set the metadata of variables. If metadata of a variable does not exist, first the system creates metadata of this variable, then sets metadata value of the variable. The user can set the metadata of both imported variables from different sources and created variables in Command Window.

Select Variable: The system displays all variables created after the program is opened. The user selects one of the variables. The user can specify all features of the variable or any of them. Some fields can be leaved blank.

Type: After selecting the variable, the user determines the type of the variable from pop up menu. The types are nominal, ordinal, numeric, categorical, date/time.

Description: The user enters the description of the variable.

Distinct Values: The user determines the distinct values of the variable using this field. It can be low, medium, high etc.

Miss Values: The user selects missing value of the variable. If the missing value is not in the pop up menu, user enters it to text box in the line of “Miss Values”.

Use Case: **Set Metadata**

ID: 2.5

Actor(s): User

Preconditions: The variables have to be imported or created.

Flow of Events:

1. The user selects “Set Meta” submenu from Data menu.
2. <<include>> select variable.
3. The user selects the variable from combo box that contains all variables.
4. The user determines the type of the variable from Type combo box.
5. If the user wants, he enters description for the variable.
6. If the user wants, he enters distinct variables for variable.
7. If the user wants, he enters missing value of the variable
   - 7.1 The user can selects the missing value from Miss Values combo box
   - 7.2 If the missing values is absent in the combo box, he can enters the character to text box.
8. The user clicks on OK button save this record.

Post Conditions:

1. The system creates metadata for selected variable.

Figure 6.18: Use Case- Set Metadata

6.3.2.6 Add Meta

When the user selects "Add Meta" from Data menu, the Figure 6.19 appears.

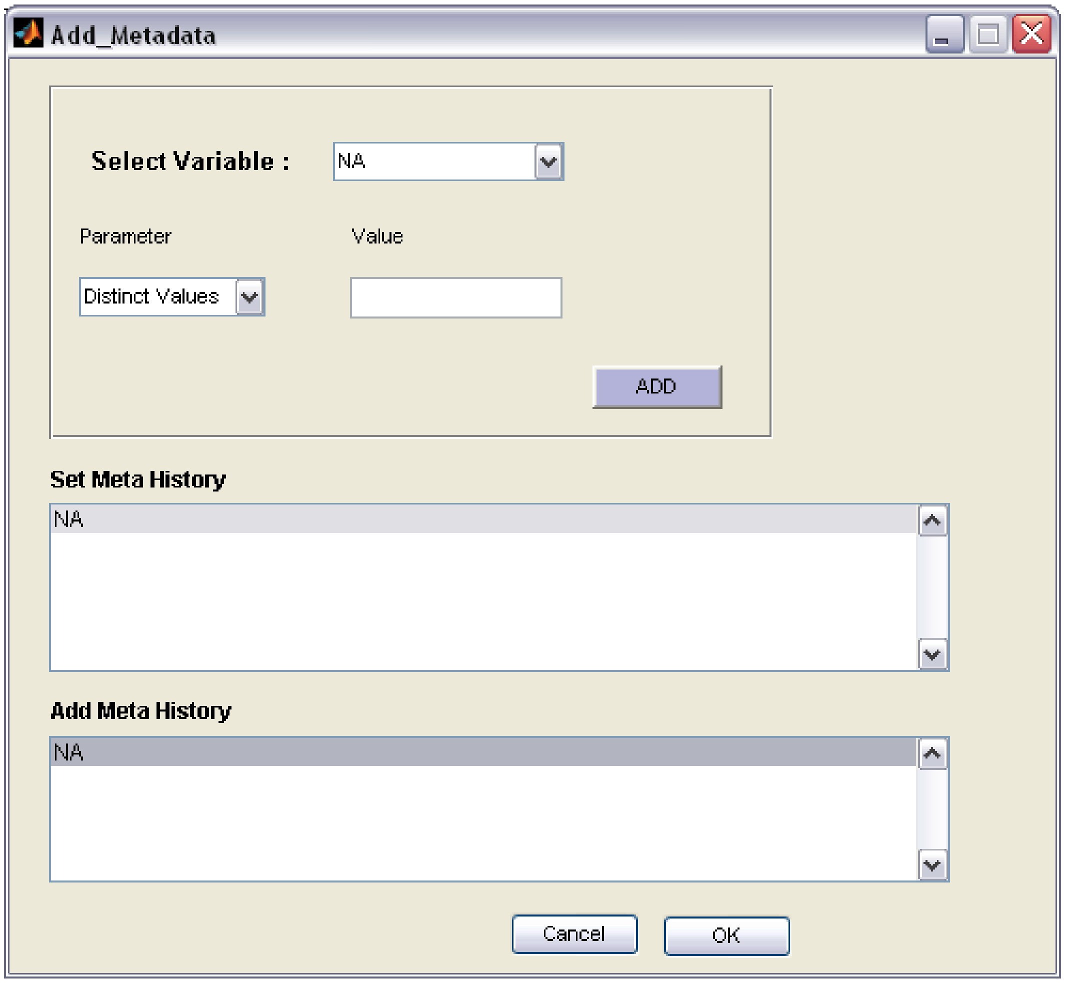


Figure 6.19: Add Metadata

This window is used to add new values to metadata of variable.

Select Variable: The system displays all variables created after the program is opened. The user selects one of the variables.

Parameter: After selecting the variable, the user selects the parameter to add value into them. The parameter can be distinct values or missing values.

Value: The user enters value for the selected parameter.

Add: When user clicks on "Add" button, the system adds them to metadata of the variable. The variable should have metadata to add values. Otherwise this button does not work.

Set Meta History: In this field, the user sees the variables and their metadata created by Set Meta. It helps user to see, which variables have metadata and which parameters how defined.

Add Meta History: In this field, the user sees added parameters and their values to variables. When the user clicks on "Add" button, the system writes the code to this field.

Use Case: **Add Metadata**
ID: 2.6
Actor(s): User
Preconditions: **1.** The variables have to be imported or created. **2.** The metadata of variables should be created.
Flow of Events: **1.** The user selects "Add Meta" submenu from Data menu **2.** <<include>> select variable **3.** The user selects variable from combo box. **4.** The user selects the parameter. **5.** The user enters the value of the parameter. **6.** The user clicks on "Add" button. **7.** The system displays metadata of variables created in Set Meta. **8.** The system displays added values of metadata.
Post Conditions: The system added new values to metadata of the selected variable.

Figure 6.20: Use Case- Add Metadata

6.3.2.7 List Meta

When the user selects “List Meta” from the Data menu, Figure 6.21 appears.

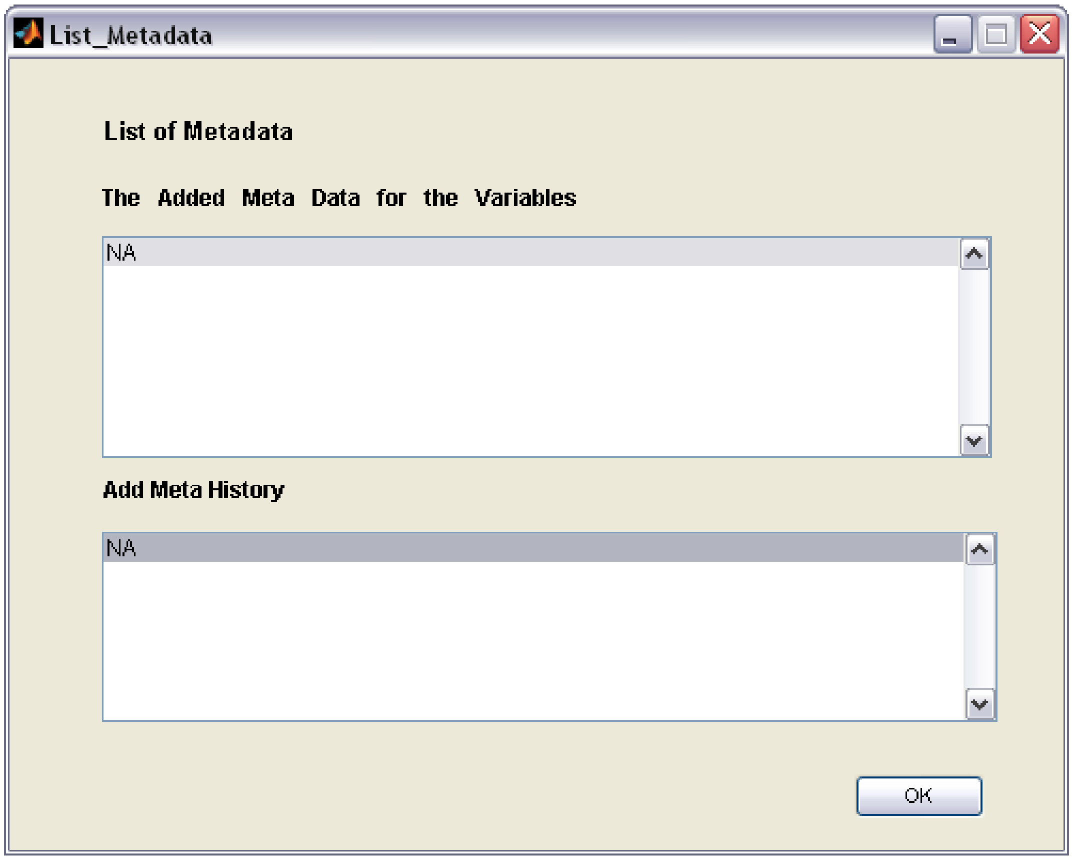


Figure 6.21: List Metadata

In this window metadata of variables are listed.

Set Meta History: In this field, the user sees the variables and their metadata created by Set Meta. It helps user to see, which variables have metadata and which parameters how defined.

Add Meta History: In this field, the user sees added parameters and their values to variables.

Use Case: **List Metadata**
ID: 2.7
Actor(s): User
Preconditions: 1. The variables have to be imported or created. **2.** The metadata of variables should be created.
Flow of Events: **1.** The user selects “List Meta” submenu from Data menu. **2.** The system displays metadata values of the variables created with setmeta command in List of Metadata **3.** The system displays metadata values of the variables created with addmeta command in Add Meta History.
Post Conditions:

Figure 6.22: Use Case- List Metadata

6.3.2.8 Descriptives

When the user selects “Descriptives” from the Data menu, Figure 6.23 appears.

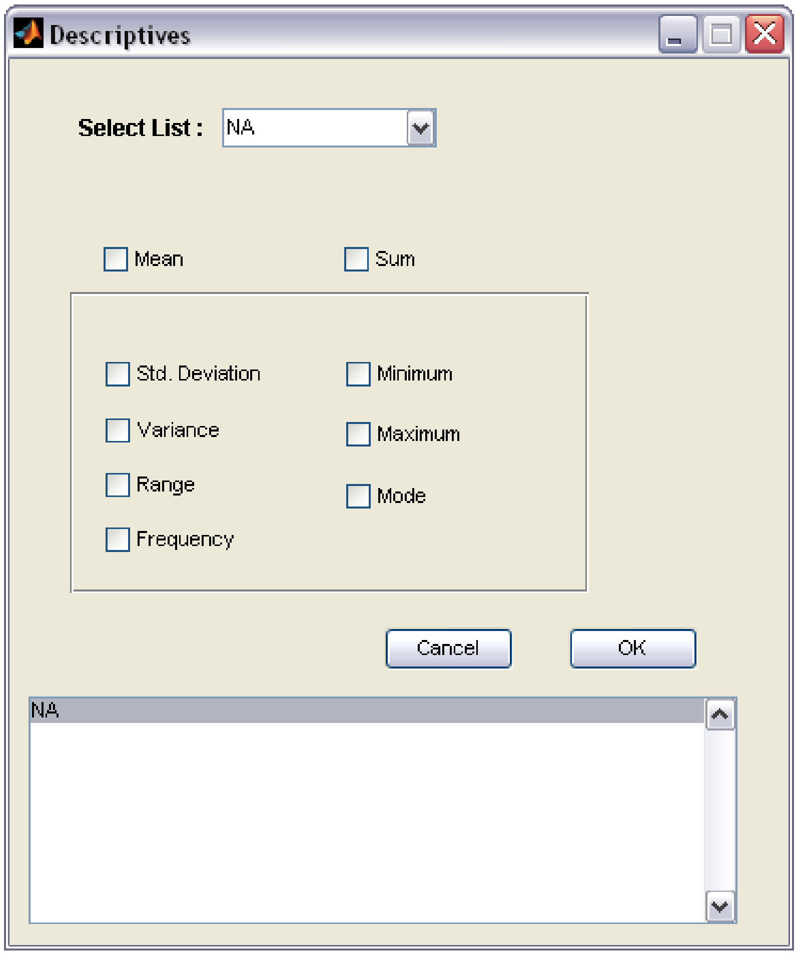


Figure 6.23: Descriptives

Select List: In this window, the user selects variable from the Select List combo box. The Select List combo box contains all variables created or imported to the tool.

The user selects one of the statistical functions and uses “OK” button. The result is listed at the bottom of the table.

The user can see only one result each time. For the same variable, to see different statistics s/he should repeat the application.

Use Case: **Descriptives**
ID: 2.8
Actor(s): User
Preconditions: The variables have to be imported or created.
Flow of Events: **1.** The user selects "Descriptives" submenu from Data Menu. **2.** <<include>> select list **3.** The user selects variable from select list. **4.** If the user wants to see sum of the data 4.1 He selects Sum check button. 4.2 He clicks on OK button. 4.3 The system displays the result at the bottom of the screen. **5.** If the user wants to see mean of the data 5.1 He selects Mean check button. 5.2 He clicks on OK button. 5.3 The system displays the result at the bottom of the screen. **6.** If the user wants to see standard deviation of the data 6.1 He selects Std. Deviation check button. 6.2 He clicks on OK button. 6.3 The system displays the result at the bottom of the screen. **7.** If the user wants to see sum of the data 7.1 He selects Variance check button. 7.2 He clicks on OK button. 7.3 The system displays the result at the bottom of the screen. **8.** If the user wants to see minimum value of the data 8.1 He selects Minimum check button. 8.2 He clicks on OK button. 8.3 The system displays the result at the bottom of the screen. **9.** If the user wants to see maximum value of the data 9.1 He selects Maximum check button. 9.2 He clicks on OK button. 9.3 The system displays the result at the bottom of the screen.

10. If the user wants to see range of the data 10.1 He selects Range check button. 10.2 He clicks on OK button. 10.3 The system displays the result at the bottom of the screen. **11.** If the user wants to see mode of the data 11.1 He selects Mode check button. 11.2 He clicks on OK button. 11.3 The system displays the result at the bottom of the screen. **12.** If the user wants to see frequencies of the data 12.1 He selects Frequency check button. 12.2 He clicks on OK button. 12.3 The system displays the result at the bottom of the screen.
Post Conditions:

Figure 6.24: Use Case- Descriptives

6.3.3 PREPARATION

6.3.3.1 Missing Values

When the user selects “Missing Value” from the Preparation menu, Figure 6.25 appears.

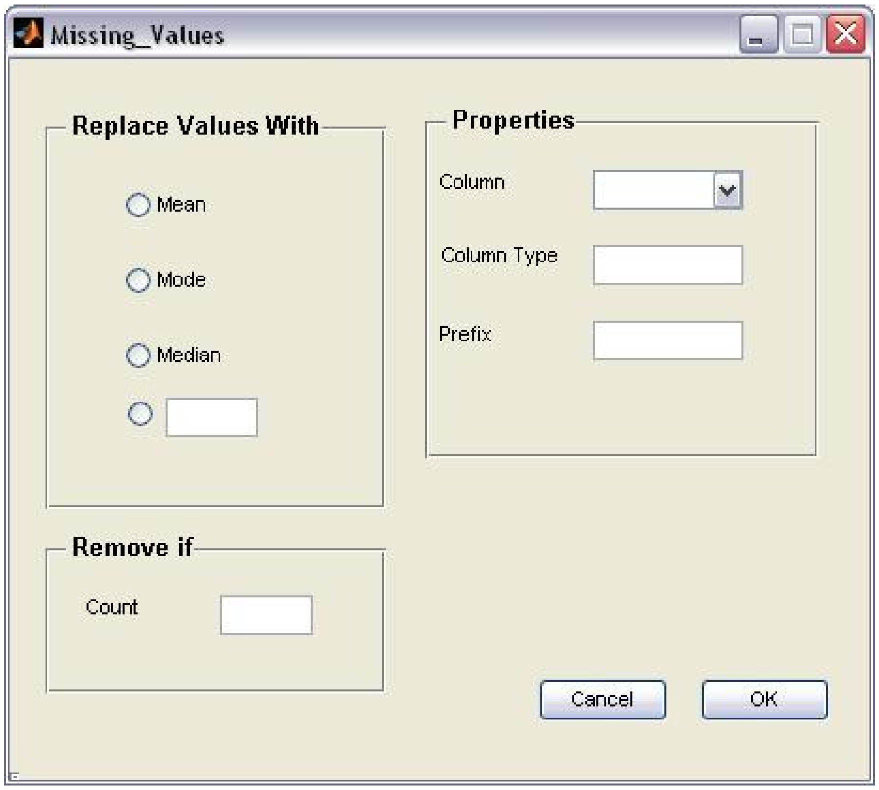


Figure 6.25: Missing_Values

Replace Values With: The user determines the missing handling method with using radio buttons. If the user wants to replace the missing values with any value, s/he can select the radio button at last and enter the value to the text box.

Properties: For the replacements of missing values the user selects the column from Column combo box. If the metadata of the variable was defined, the type of the variable is shown at Column Type box.

If the column data type is numeric, the user can handle missing values with the methods of Mean, Median, Mode or the user specified value.

If the column data type is nominal, the user can handle missing values with the methods of Mode or user specified value.

If the column data type is nominal, the user can handle missing values with the methods of user specified value.

Prefix: The user can define prefix for new variables with using Prefix text box. If the user wants to add new variable to data set with handled missing values and do not touch the original data, s/he uses this field.

Remove if: If the user wants to remove rows that have number of missing values, s/he uses this part. The user enters the number to text box. It means, if the number of the missing values at the row is equal or more than that number remove the row.

OK: when the user clicks on OK button, the handling missing process begins according to selected parameters.

Use Case: **Missing Values**

ID: 3.1

Actor(s): User

Preconditions: The variables have to be imported or created.

Flow of Events:

1. The user selects "Missing Value" submenu from Preparation menu.

2. <<include>> column

3. For the data set is ready to process, the user selects the column from the column combo box.

4. If the variable has been selected is Numeric,

  4.1 The user determines the replacement type (Mean, Mode, Median or a value specified by user).

Else if the column has been selected is Nominal;

  4.1 The user determines the replacement type (Mode or a value specified by user).

Else if the column has been is Ordinal,

  4.1 The user replace the missing values with a values specified by himself.

5. If the user wants to save new values with different name,

  5.1 The user enters the prefix for value.

6. The user clicks on OK button.

Post Conditions: The user sees the handled data.

Figure 6.26: Use Case- Missing Values

6.3.3.2 Sampling

When the user selects “Sampling” from the Preparation menu, Figure 6.27 appears.

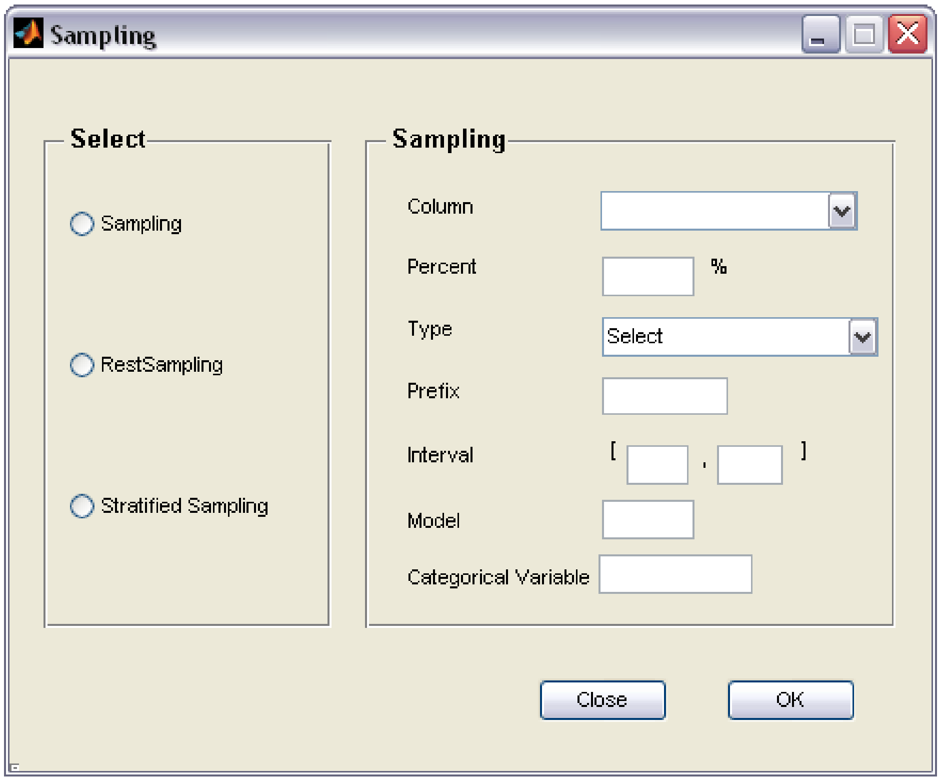


Figure 6.27: Sampling

The user selects the variable or list from Column combo box to make sampling.

Select: In this window, the user selects the sampling method from the radio buttons at left side.

If user makes sampling with/without replacement, he uses Sampling radio button.

If the user wants to create new variables with values which are not selected in a previously performed sampling, he selects RestSampling radio button. In this type the user only defines model and prefix, if he deserves.

If the user makes stratified sampling, he selects Stratified Sampling radio button.

Percent: The user defines the percent for sampling. If the user leaves blank this field the system takes all data in defined intervals.

Type: Then the user selects the type for with/without replacement.

Prefix: If the user wants to create new variables, he enters prefix.

Interval: The user determines the interval to take sample of the data between these intervals. .For example, if interval is an array of two elements for example [30,70] then the rows between 30 and 70 will be selected.

Model: Model is an array of integer numbers which are the indices of the rows that user want to be selected and be entered manually. If the wants to save the use the properties of this sampling for another sampling or if there will be necessity to add new variables to sampling, user can store the properties of the sampling to in this variable entered to Model text box as a model. Another time, when he enters the model name he can leave blank the percent, type, and interval parameters.

Categorical Variable: For the Stratified Sampling the user specifies the variable to consider the ratios of distinct values of column.

Use Case: **Sampling**
ID: 3.2
Actor(s): User
Preconditions: The variables have to be imported or created and lists created.
Flow of Events: **1.** The user selects "Sampling" submenu from Preparation menu. **2.** <<include>> column **3.** For the data set is ready to process, the user selects the column or list from the column combo box. **4.** The user selects sampling type. If the type is Sampling 4.1 The user enters the percent. 4.2 The user determines the type of replacement. 4.3 If the user wants to save the result in new variables enters prefix. 4.4 If the user wants to take data in defined intervals for sampling, enters the intervals. 4.5 The user enters name for model. Else if the type is RestSampling, 4.1 The user enters name for model. 4.2 If the user wants to save the result in new variables enters prefix. Else if the type is Stratified Sampling, 4.1 The user enters the percent. 4.2 The user determines the type of replacement. 4.3 If the user wants to save the result in new variables enters prefix. 4.4 If the user wants to take data in defined intervals for sampling, enters the intervals. 4.5 The user enters name for model. 4.6 The user enters the categorical variable to make sampling according to ratios of this variable. **5.** The user clicks on OK button.
Post Conditions: The sampling results are displayed.

Figure 6.28: Use Case- Sampling

6.3.3.3 Transformation

When the user selects "Transformation" from the Preparation menu, Figure 6.29 appears.

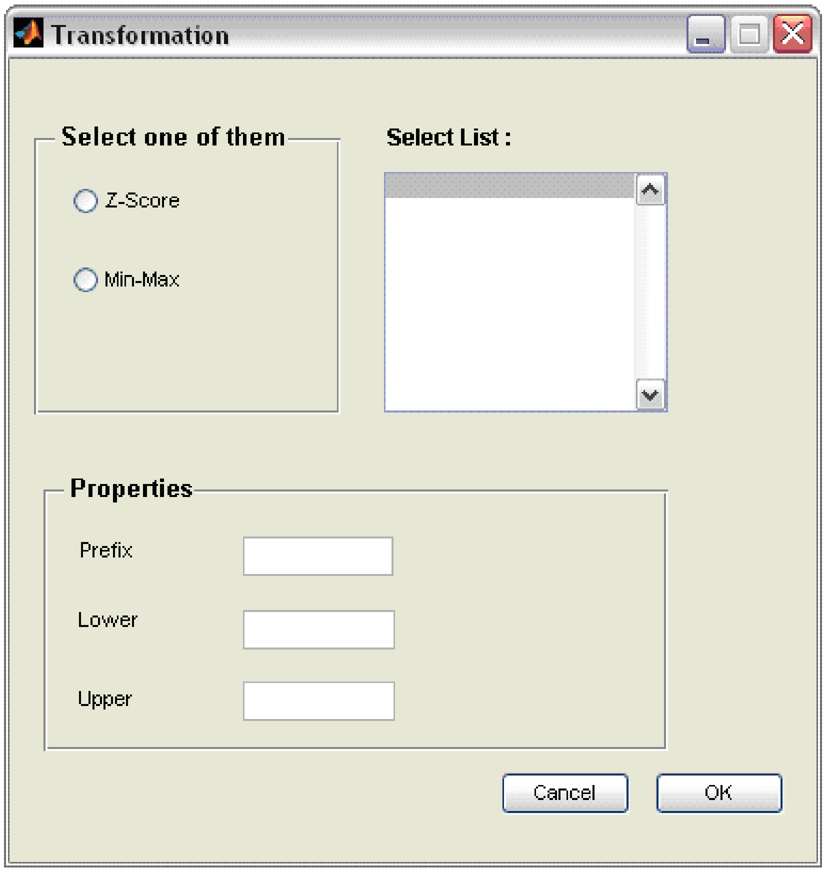


Figure 6.29: Transformation

Select List: In this window, the user selects the lists and variables from the Select List list box.

Then the user selects the transformation method, he can apply z-score normalization or min-max normalization.

Prefix: The user can define prefix for new variables with using Prefix text box. If the user wants to add new variable to data set with transformed values and do not touch the original data, s/he uses this field.

Lower: The user determines the lower limit of the range for Min-max normalization.

Upper: The user determines the lower limit of the range for Min-max normalization.

OK: The user clicks on the OK button to begin the process and the data set is changed.

Use Case: **Transformation**

ID: 3.3

Actor(s): User

Preconditions: The variables have to be imported or created and lists created.

Flow of Events:

1. The user selects “Transformation” submenu from Preparation menu.

2. <<include>> select list

3. For the data set is ready to process, the user selects the column or list from the list box named Select List.

4. The user selects transformation method.

If the method is z-score,

4.1 If the user wants to save the result in new variables enters prefix.

Else if the method is min-max,

4.1 If the user wants to save the result in new variables enters prefix.

4.2 The user defines lower limit of range.

4.3 The user defines upper limit of range.

5. The user clicks on OK button.

Post Conditions: The transformed variables are displayed.

Figure 6.30: Use Case- Transformation

6.3.3.4 Discretization

When the user selects “Discretization” from the Preparation menu, Figure 6.31 appears.

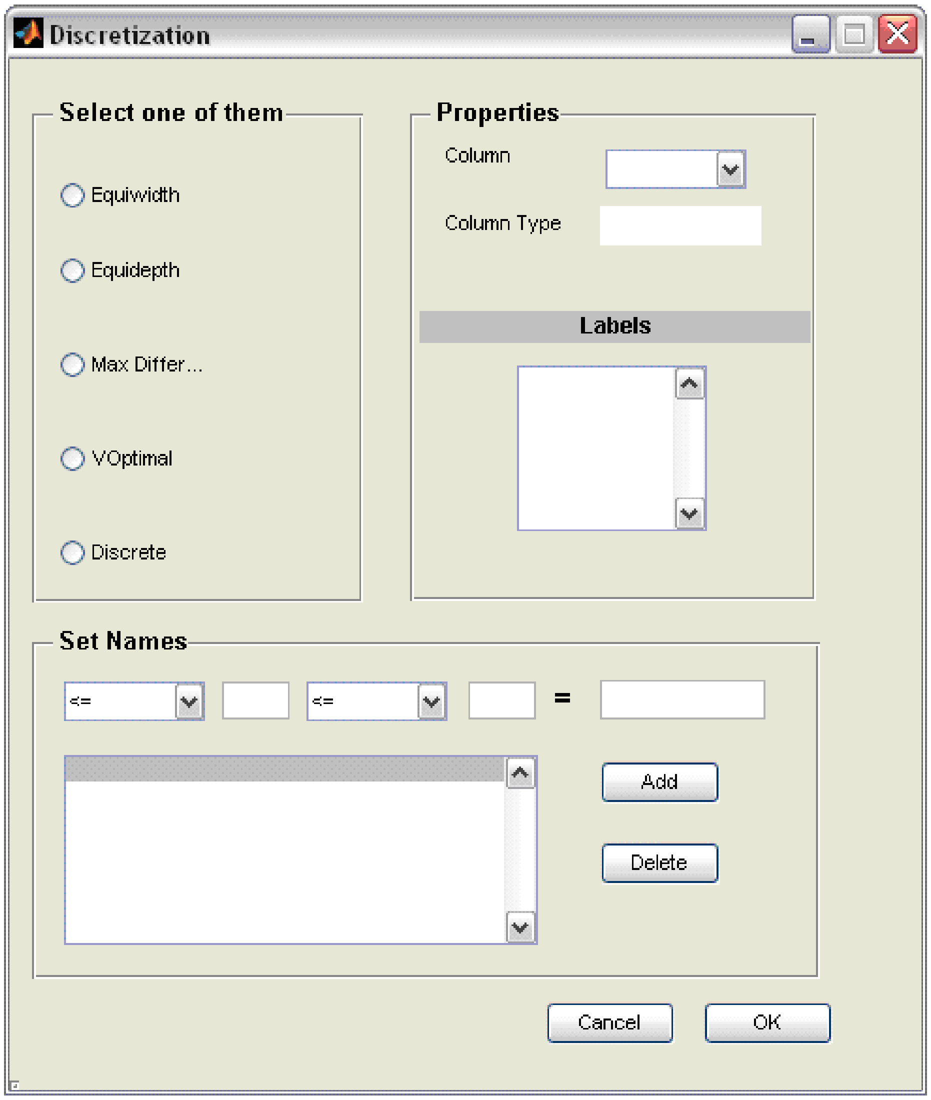


Figure 6.31: Discretization

Column: In this window, the user selects the column from Column combo box, that contains all variables created or imported to the tool. After selection the system displays the type of the variable if its metadata is created.

According to the selected variable, the user selects the method using radio button.

If the user determines the method of equiwidth, equidepth, max difference or v-optimal, he enters labels using Labels text box. The user enters each label to one line and use enter to pass other line

Set Names: If the user selects discrete method, he has to define discrete values and their intervals. When user determines intervals and enter value uses "Add" button to add it discrete values. The system displays the value and interval at the list box. When user selects the discrete value and uses "Delete" button, the line is erased from discrete values.

OK: The user clicks on OK button to process begins and the data set changes.

Use Case: **Discretization**

ID: 3.4

Actor(s): User

Preconditions: The variables have to be imported or created and lists created.

Flow of Events:

1. The user selects "Discretization" submenu from Preparation menu.

2. <<include>> column

3. For the data set is ready to process; the user selects the column or list from the Column combo box.

4. The user selects discretization method.

If the method is equiwidth, equidepth, max difference or v-optimal,

4.1 The user enters labels

Else if the method is discrete,

4.1 The user enters discrete values and intervals

4.2 To add values, he uses Add button.

4.3 If the user wants to remove discrete value from list, he uses Delete button.

5. The user clicks on OK button.

Post Conditions: The result is displayed.

Figure 6.32: Use Case- Discretization

6.3.4 FUNCTIONALITY

6.3.4.1 Association:

When the user selects “Association” from “Functionality” menu, Figure 6.33 appears.

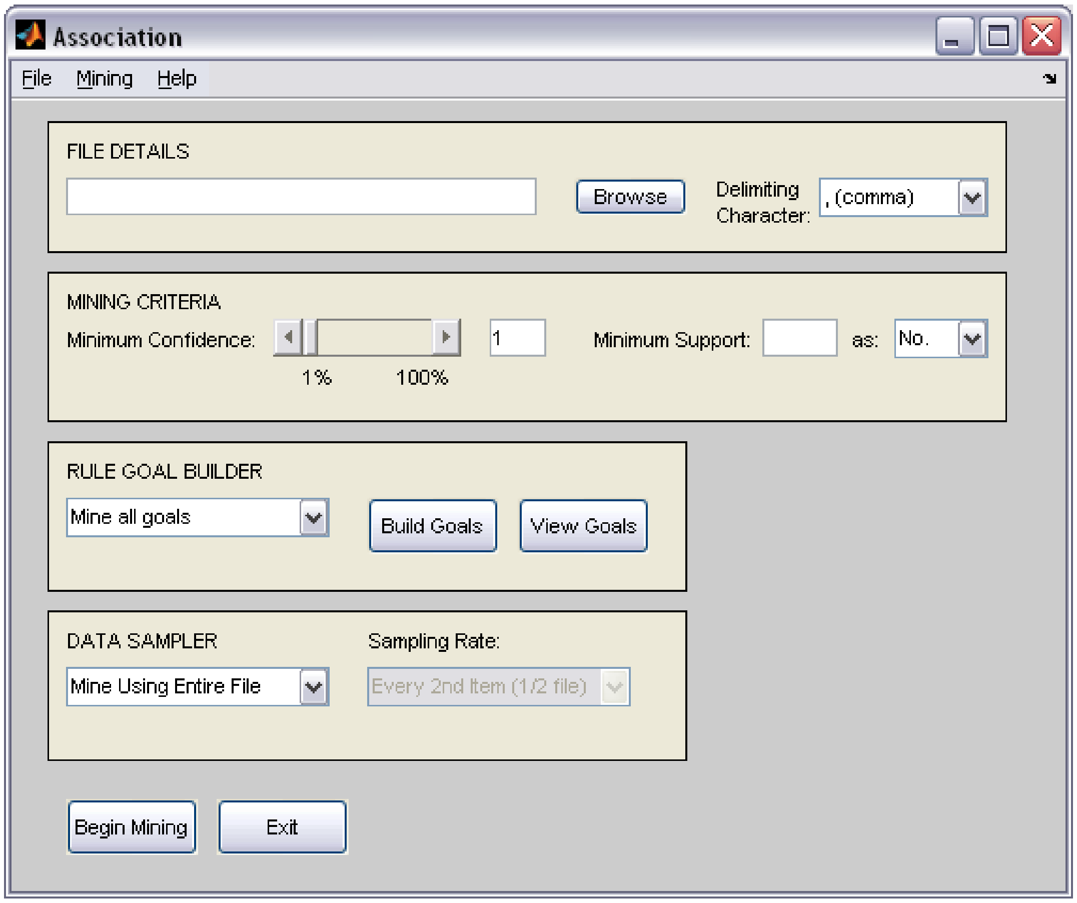


Figure 6.33: Association

This screen provides Association processes to the user. It is taken from Armada association toolbox which is developed by James Malone. The program integrates several mining methods which allow the efficient extraction of rules, while allowing the thoroughness of the mine to be specified at the user’s discretion. The program also allows the results to be displayed through various graphical representations, such as bar charts and line graphs. This open source and free toolbox is added to our project for allowing user to reach Association operations in one screen.

Browse: Browse button provide user to navigate on the disk for finding files for mining. When user clicks on browse button the current directory which contains Armada file will be opened. Then user can change directories and highlight a file for execution.

Delimiting Character: From the "Delimiting Character" popup menu user select the character which separates the fields in the text file. So that algorithms understand that the cursor is reading another data for data import.

Minimum Confidence: User determines the minimum confidence limit for association analysis. That is the probability that selected items appear together in a subset given that the subset contains one or more selected items. User can change the value by pressing left or right arrows or scrolling the button to the left or right or entering the percentage to the textbox.

Minimum Support: User enters the minimum support percentage to the textbox for defining the minimum level will be regarded by the algorithm in the data set. It identifies that in the all data set the minimum percent of item or items appear in all the data.

As: From the list box user selects percentage or No values after entering the data. If user does not enter any data to the textbox it will be regarded as no.

Rule Goal Builder: The user can select whether using his own built goals or mining all the goals by clicking the drop down menu.

Build Goals: The user can build his own goals by clicking the build goals button. In the Goal builder screen user selects variables and the rules associated with the variables.

View Goals: When the user clicks on view goals button the system displays the rules built by the user.

Data Sampler: User determines whether to use all the file or using a cluster of the file by selecting appropriate choice from the dropdown menu. The reason why user needs sampling is to test the result of an association in a sample on the data. So that result of the mining will be appropriate.

Sampling Rate: If user selects using samples, sampling rate will be available and the user selects the percentage of the sample from the list.

Begin Mining: After identifying all the necessary fields, user can begin mining by clicking Begin mining button. Results of the association will be displayed in a new screen.

Exit: If user clicks exit button the program will be closed.

Use Case: **Association**
ID: 4.1
Actor(s): User
Flow of Events: 1. The user selects “Association” submenu from Functionalities menu **5.** The user presses the browse button, finds the text data and selects the data base. **6.** The user selects delimiting character from dropdown menu. **7.** The user determines the minimum confidence percentage. **8.** The user could define minimum support percent. **9.** The user determines mining criteria. 9.1 User could select to mine all the goals. 9.2 User could select to mine own built goals. 6.3. User Clicks the Build Goal button to define new goals. **10.** The user decides whether to use a data sample or all the data. 7.1 If user wants to use a sample, he selects sampling rate from the dropdown menu. **11.** The system displays added association results to the user on a new screen.
Post Conditions:

Figure 6.34: Use Case- Association

6.3.4.2 Classification:

6.3.4.2.1 Decision Tree

6.3.4.2.1.1 CHAID

When the user selects "CHAID" from "Decision Tree" under "Classification" Title in "Functionality" menu, Figure 6.35 appears.

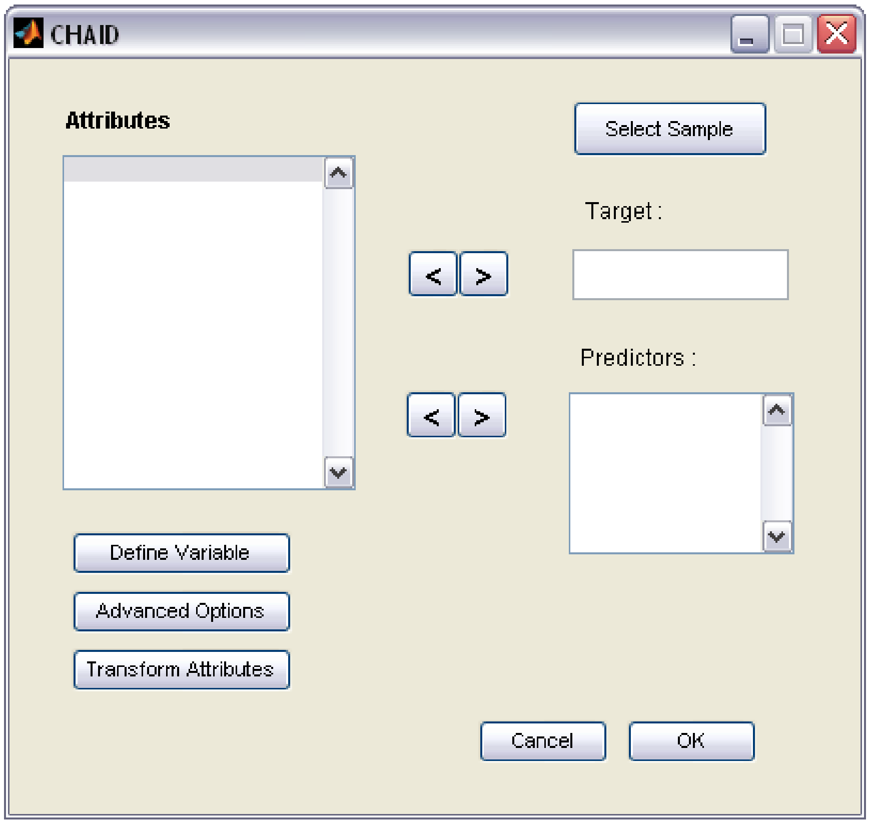


Figure 6.35: CHAID

This Window is used for Chaid decision tree technique which is short for CHi-squared Automatic Interaction Detector.

Attributes: Attributes are listed on the list box. User can select, add and remove attributes to the list box.

Select Sample: When user clicks select sample button, he can choose an already defined sample or can define a new sample.

Target: The variable which is constitutes the basis of the analysis. It can be regarded as dependent variable.

Predictors: The variables which high lights the target variable. It is also considered as independent variable.

Define Variable: The user can define new variables by using the define variable button.

Advanced Options: The user can change the options and critical values like from the Advanced Operations Menu.

Transform Attributes: The user also can transform attributes to different format like Z-score and min-max normalization.

Use Case: **CHAID**
ID: 4.2.1.1
Actor(s): User
Flow of Events: 1. The user selects “CHAID” menu from “Decision Tree” submenu under Classification Title in Functionalities menu. 2. The user can define new variables or add variables to the list. 3. The user selects target variable by selecting attributes list box. 4. The user defines predictor values by selecting from list box. 5. The user can change Advanced Options for optimizing algorithm by changing critical values. 6. The user can transform the attributes to different format from transformation menu. 7. When user clicks OK button, the program generates the Decision Tree by using Chaid technique and displays the result on the screen.
Post Conditions:

Figure 6.36: Use Case- CHAID

6.3.4.2.1.2 EX CHAID

When the user selects "EX CHAID" from "Decision Tree" under "Classification" Title in "Functionality" menu, Figure 6.37appears.

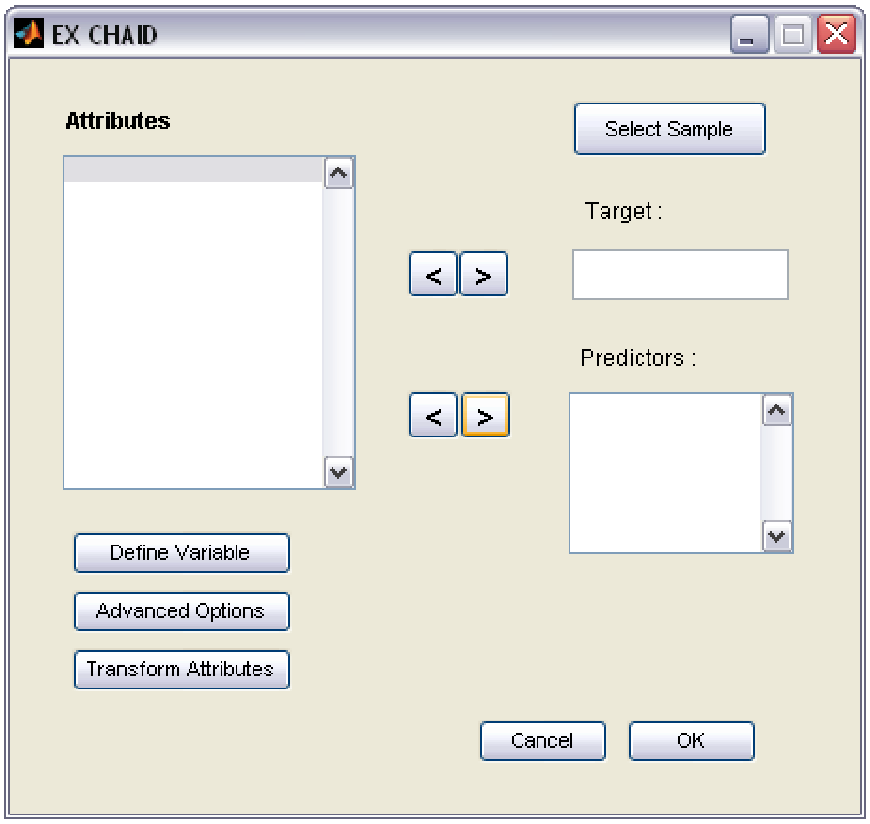


Figure 6.37: EX- CHAID

This screen provides building decision trees by using EX CHAID technique. Screen is similar to CHAID screen but the algorithms are different for making classification.

Attributes: Attributes are listed on the list box. User can select, add and remove attributes to the list box.

Select Sample: When user clicks select sample button, he can choose an already defined sample or can define a new sample.

Target: The variable which is constitutes the basis of the analysis. It can be regarded as dependent variable.

Predictors: The variables which high lights the target variable. It is also considered as independent variable.

Define Variable: The user can define new variables using "Define Variable" button.

Advanced Options: The user can change the options and critical values like from the Advanced Operations Menu.

Transform Attributes: The user also can transform attributes to different format like Z-score and min-max normalization.

Use Case: **EX CHAID**
ID: 4.2.1.2
Actor(s): User
Flow of Events: 1. The user selects "CHAID" menu from "Decision Tree" submenu under Classification Title in Functionalities menu. 2. The user can define new variables or add variables to the list. 3. The user selects target variable by selecting attributes list box. 4. The user defines predictor values by selecting from list box. 5. The user can change Advanced Options for optimizing algorithm by changing critical values. 6. The user can transform the attributes to different format from transformation menu. 7. When user clicks OK button, the program generates the Decision Tree by using Chaid technique and displays the result on the screen.
Post Conditions:

Figure 6.38: Use Case- CHAID

6.3.4.2.2 Neural Network

When the user selects “Neural Network” from under “Classification” Title in “Functionality” menu, Figure 6.39 appears.

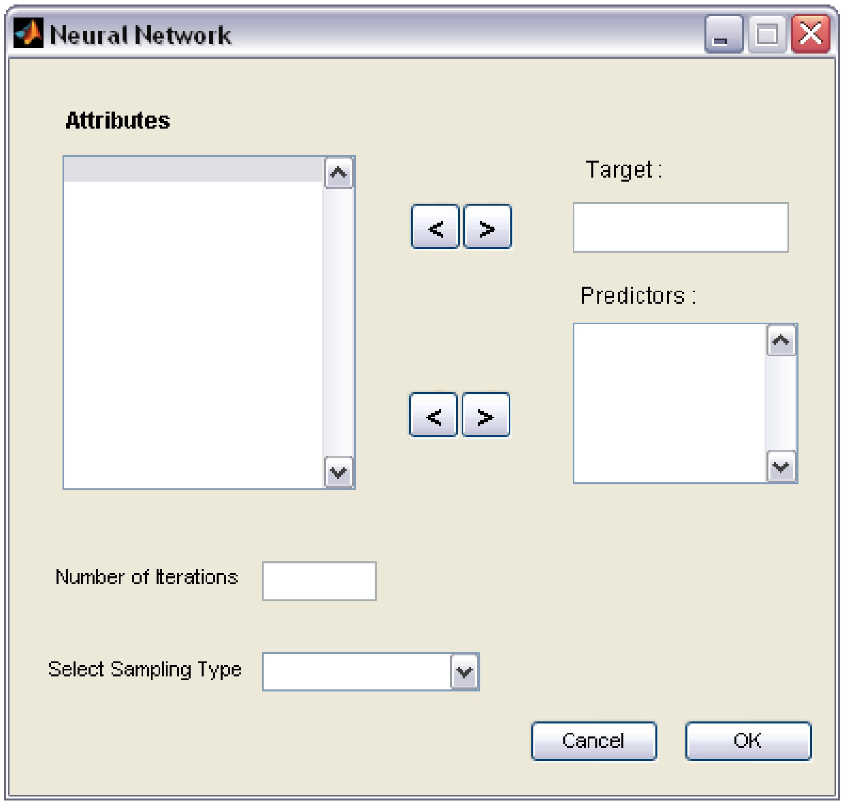


Figure 6.39: Neural Network

This window provides the user to run the neural network algorithms.

Attributes: Attributes are listed on the list box. User can select, add and remove attributes to the list box.

Number of iterations: The user determines the number of iterations will be performed in the execution according to data set size and number of nodes.

Target: The variable which is constitutes the basis of the analysis. It can be regarded as dependent variable.

Predictors: The variables which high lights the target variable. It is also considered as independent variable.

Select Sampling Type: The user can select the sampling type from the list box. It can be either Cross Validation or Bootstrap according to the data mining aim.

Use Case: **Neural Network**
ID: 4.2.2
Actor(s): User
Flow of Events: 1. The user selects "Neural Network" menu from Classification Title under "Functionalities" menu. 2. The user selects target variable by selecting attributes list box. 3. The user defines predictor values by selecting from list box. 4. The user enters number of iterations to the text box. 5. The user selects sampling type from the list box for Neural Network operations. 6. When user clicks OK button, the program begin execution and generates the Neural Network according to data set and given parameters.
Post Conditions:

Figure 6.40: Use Case- Neural Network

6.3.4.3. Clustering

When the user selects "Clustering" from "Functionality" menu, Figure 6.41 appears.

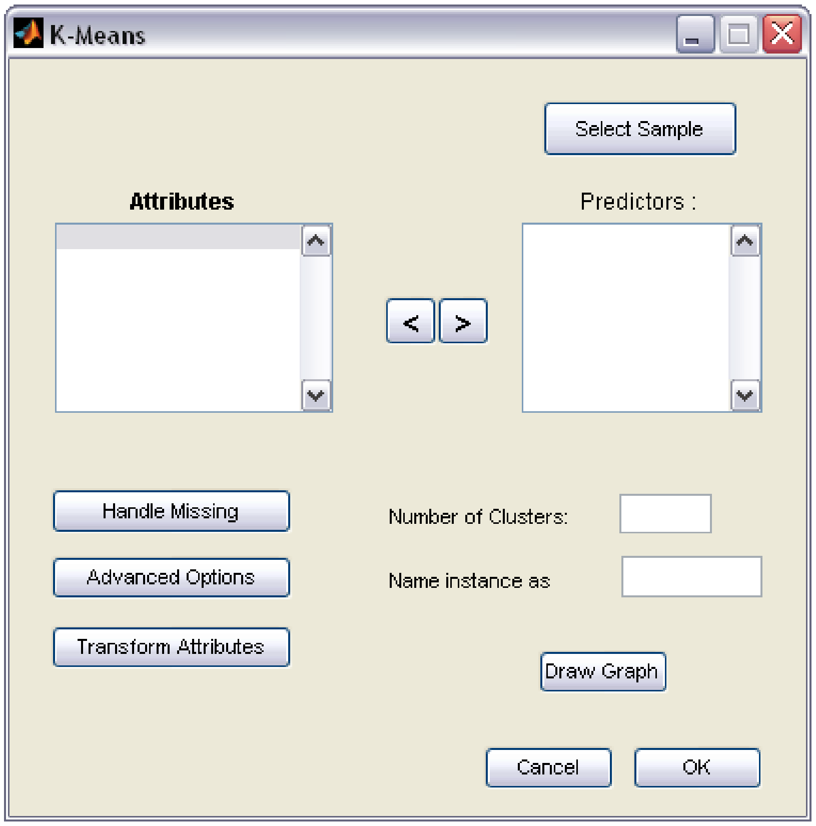


Figure 6.41: K-Means , Clustering

Clustering window enables user to run k-means algorithm for shape clusters in the data set.

Select Sample: The user can select pre defined data sample for clustering. The user can also create samples for operations.

Attributes: Attributes are listed in the list box for explaining the user available values for operations.

Predictors: The user can select variables for defining clusters. The variables in predictors list constitute the cluster after applying K-means algorithms.

Handle Missing: The user can remove or change missing values in the field by using Handle missing menu on the screen.

Advanced Operations: The user can change advanced settings to optimize or test algorithms in clustering processes.

Transform Attributes: The user can also transform attributes to Z-score or min-max by using transform menu.

Number of Clusters: The user enters the cluster numbers. The algorithms take the number as input and divide data to the given number of clusters.

Name instance as: The user defines a name for the operations and records the application history.

Draw Graph: The user can also see the graphs of the operation by selecting graph options in the screen.

Use Case: **Clustering**
ID: 4.3
Actor(s): User
Flow of Events: 1. The user selects “Clustering” menu from under “Functionalities” menu. 2. The user could select samples from the select sample menu. 3. The user selects predictor variable by selecting from attributes list box. 4. The user defines could replace missing values. 5. The user enters number of clusters to the text box. 6. The user can change advanced options values from the menu. 7. The user can transform the attributes before beginning to the operations. 8. The user can give name to the operation to record the operation. 9. When user clicks OK button, the program begin clustering execution and creates clusters by using K-means algorithms.
Post Conditions:

Figure 6.42: Use Case- K-means, Clustering

6.3.4.4. Regression

When the user selects “Regression” from “Functionality” menu, Figure 6.43 appears.

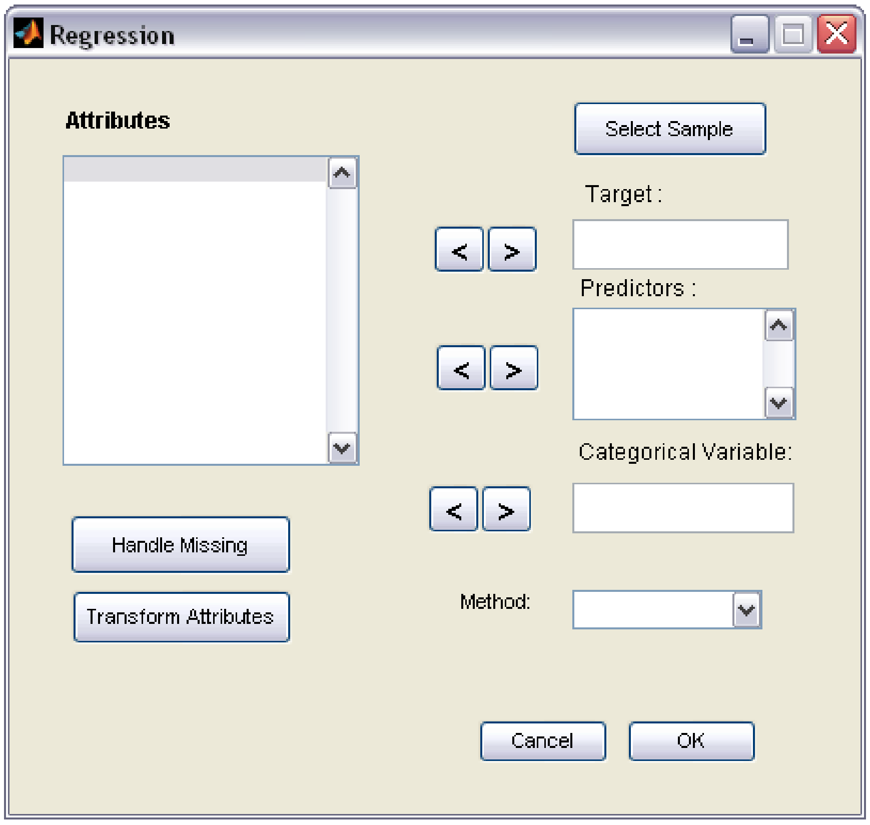


Figure 6.43: Regression

This window enables user to perform regression analysis between variables.

Attributes: Related attributes are listed in the Attributes list box for easy operations. The user can select the attributes, add and drop to other places.

Select Sample: The user can select sample for regression operations by clicking select sample button.

Target: The user can determine the dependent variables from the attributes list box.

Predictors: The user selects independent variables which explain the dependent variable.

Categorical Variable: The user could also define Categorical variables for the regression operation.

Handle Missing: The user can deal with missing values for better results. He can replace missing values with different methods or remove the missing values.

Transform Attributes: The user can transform attributes to different formats before execution for optimization and better results.

Method: The user determines the regression method from the dropdown menu. It can be simple, stepwise, backward or forward.

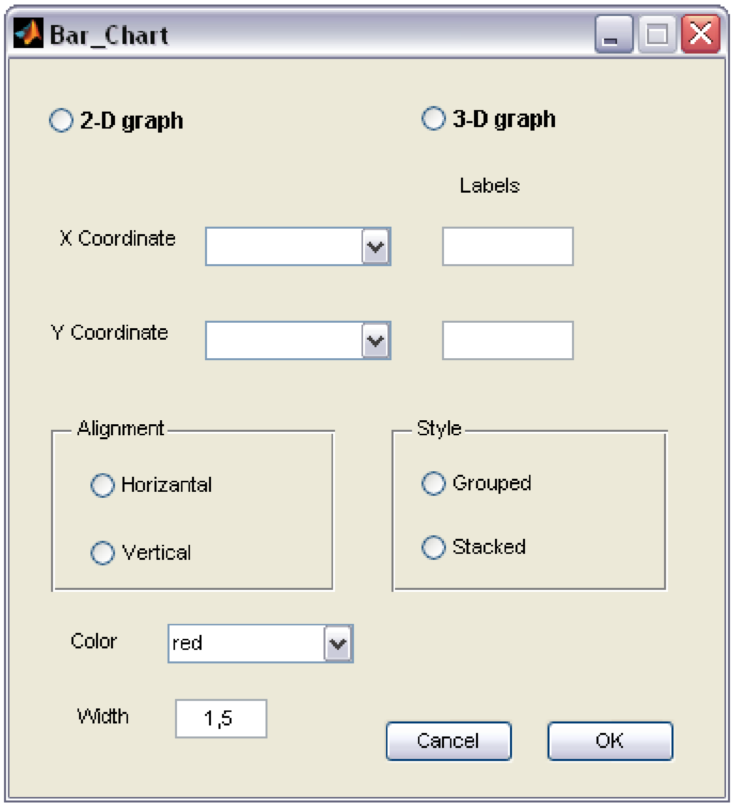
Use Case: **Regression**
ID: 4.4
Actor(s): User
Flow of Events: 1. The user selects “Regression” menu from under “Functionalities” menu. 2. The user could select samples from the select sample menu. 3. The user defines predictor variable by selecting variables from Attributes list box. 3. The user selects predictor variable by selecting from attributes list box. 4. The user defines categorical values for the operation. 4. The user could replace missing values. 5. The user transforms attributes. 6. The user can change advanced options values from the menu. 7. The user can transform the attributes before beginning to the operations. 9. When user clicks OK button, the program begin Regression and displays result on the screen.
Post Conditions:

Figure 6.44: CHAID

6.3.5 GRAPHICS

6.3.5.1 Bar Chart

When the user selects “Bar_Chart” from the Graphics menu, Figure 6.45 appears.



Color red

Width 1,5

Figure 6.45: Bar Chart

In this window, the user firstly determines the dimension of the graph using radio buttons.

Then using combo boxes he selects the variables to draw up coordinates. If the user wants he can enter labels for coordinates.

Alignment: The user determines the alignment of the graph using radio buttons.

Style: Moreover, the user determines the style of the bars.

Color: If the bar chart is drawn with one color the system let user to select the color. Otherwise for multicolor bars the system automatically colors the bars.

Width: The user defines the width between bars, the default number is 1,5

OK: The user clicks on OK button to see the chart.

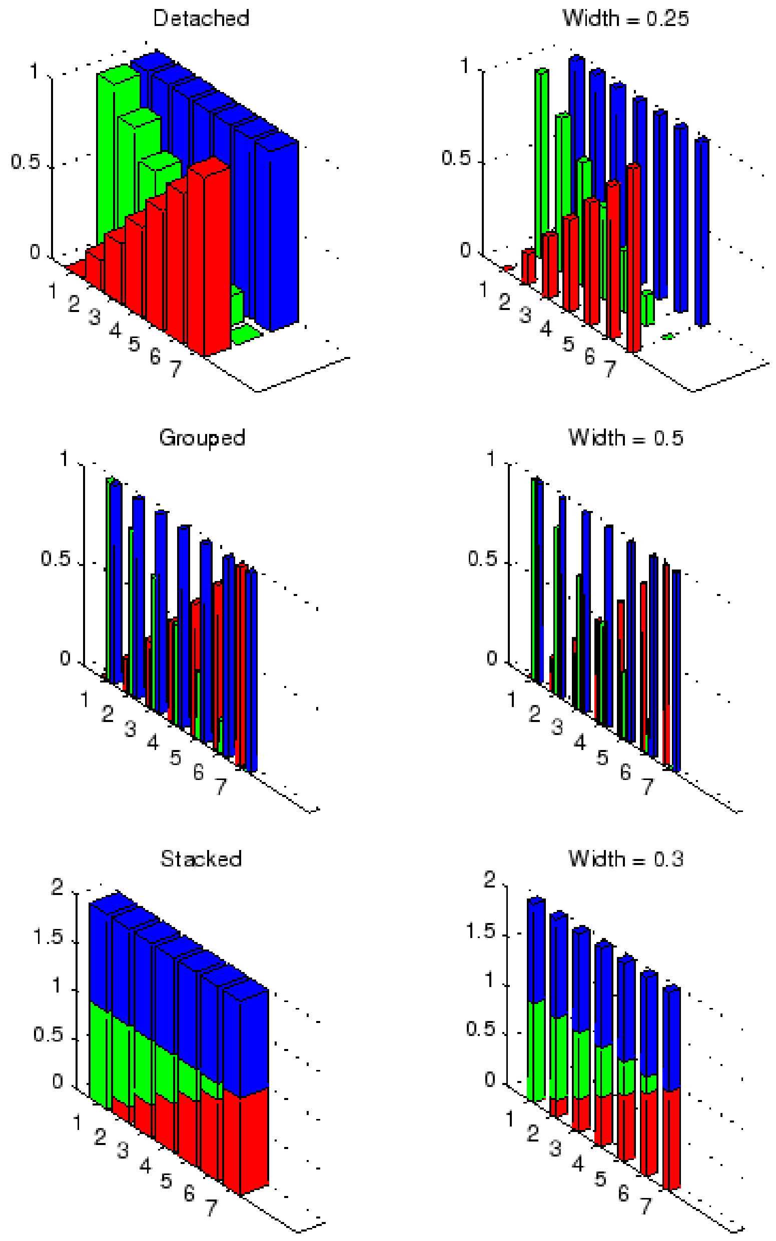


Figure 6.46: Examples of Bar Charts

Use Case: **Bar Chart**
ID: 5.1
Actor(s): User
Preconditions: The variables have to be imported or created and lists created.
Flow of Events: **1.** The user selects “Bar_Chart” submenu from Graphics menu. **2.** The user select 2-D or 3-D graph options **3.** The user selects the variables for X and Y coordinates. **4.** The user enters labels for coordinates. **5.** The user determines the alignment of the chart. **6.** The user determines the style of the function. **7.** For one color charts, the user selects the color for bar color. **8.** The user enters the width between the bars. **9.** The user clicks on OK button.
Post Conditions: The chart is displayed.

Figure 6.47: Use Case- Bar Chart

6.3.5.2 Pie Chart

When the user selects “Pie_Chart” from the Graphics menu, Figure 6.48 appears.

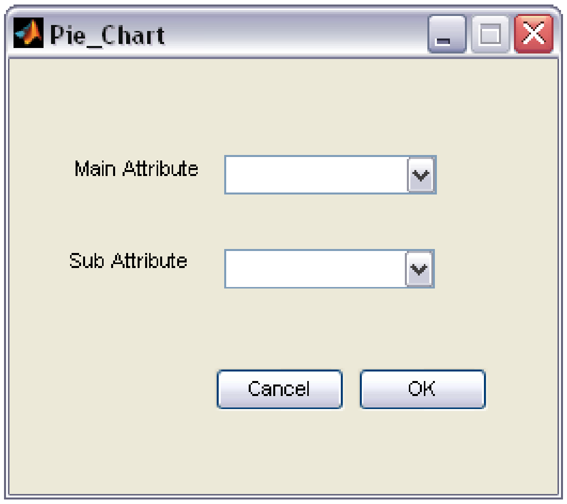


Figure 6.47: Pie Chart

In this window, the user enters the main attribute to see in the pie chart. If the user wants to analyze the main variable according to the sub attribute, he selects variable from Sub Attribute combo box too.

Then the user clicks on OK button to see the chart.

Use Case: **Pie Chart**
ID: 5.2
Actor(s): User
Preconditions: The variables have to be imported or created and lists created.
Flow of Events: 1. The user selects “Bar_Chart” submenu from Graphics menu. 2. The user select main variable from Main Attribute combo box. 3. The user selects sub variable to analyze data in terms of it. The user clicks on OK button.
Post Conditions: The chart is displayed.

Figure 6.49: Use Case- Pie Chart

6.3.5.3 Line Graph

When the user selects "Pie_Chart" from the Graphics menu, Figure 6.50 appears.

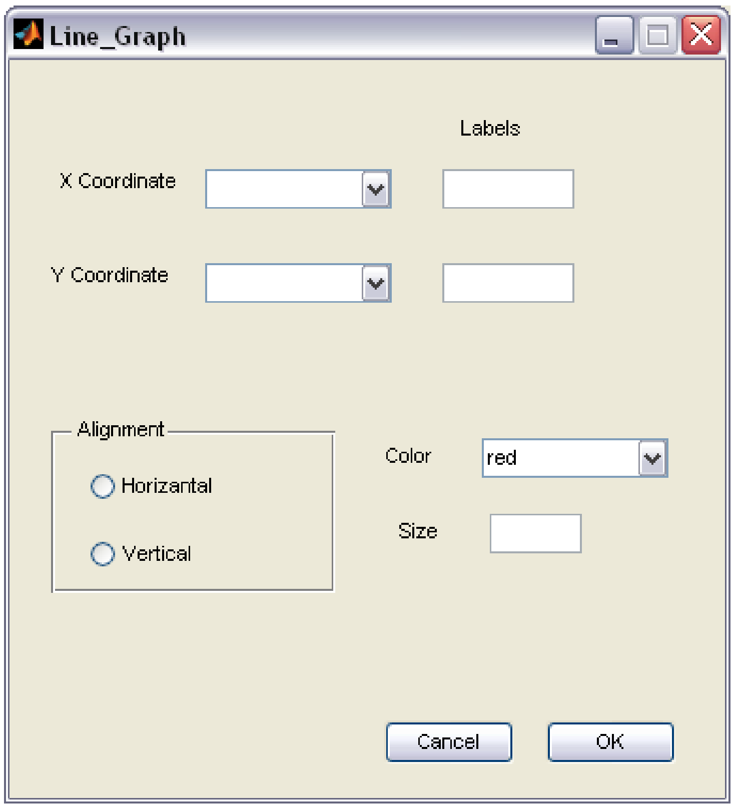


Figure 6.50: Line Graph

In this window, the user selects the variables for X and Y coordinated from combo boxes. He can enter labels for coordinates.

Alignment: The user determines the alignment of the graph using radio buttons.

Color: The user can determine the color of the line from Color combo box.

Size: The user can define the size of the line.

Use Case: **Line Graph**
ID: 5.3
Actor(s): User
Preconditions: The variables have to be imported or created and lists created.
Flow of Events: **1.** The user selects “Line_Graph” submenu from Graphics menu. **2.** The user selects the variables for X and Y coordinates. **3.** The user enters labels for coordinates. **4.** The user determines the alignment of the chart. **5.** The user selects the color of line. **6.** The user defines the size of the line **7.** The user clicks on OK button.
Post Conditions: The graph is displayed.

Figure 6.51: Use Case- Line Graph

6.3.5.4 Box Plot

When the user selects “Box_Plot” from the Graphics menu, Figure 6.52 appears.

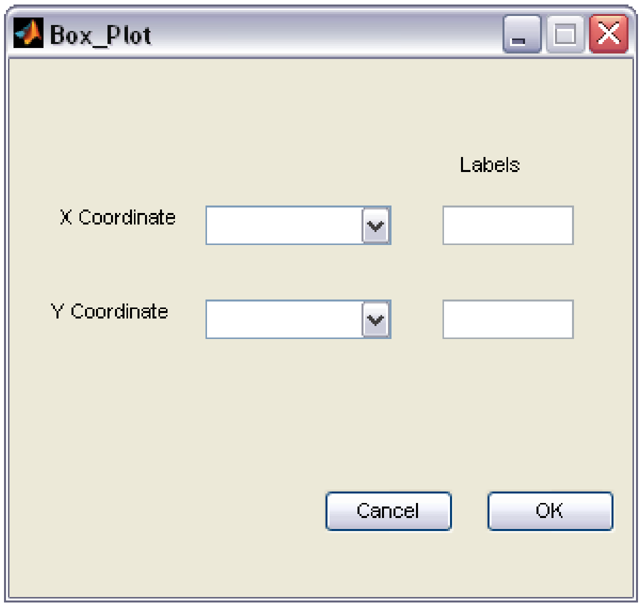


Figure 6.52: Box Plot

In this window, the user selects the variables for X and Y coordinated from combo boxes. He can enter labels for coordinates.

When the user clicks on OK button the box plot is shown.

Use Case: **Box Plot**
ID: 5.4
Actor(s): User
Preconditions: The variables have to be imported or created and lists created.
Flow of Events: **1.** The user selects “Box_Plot” submenu from Graphics menu. **2.** The user selects the variables for X and Y coordinates. **3.** The user enters labels for coordinates. **4.** The user clicks on OK button.
Post Conditions: The box plot is displayed.

Figure 6.53: Use Case- Box Plot

6.3.5.5 Scatter Plot

When the user selects "Scatter_Plot" from the Graphics menu, Figure 6.54 appears.

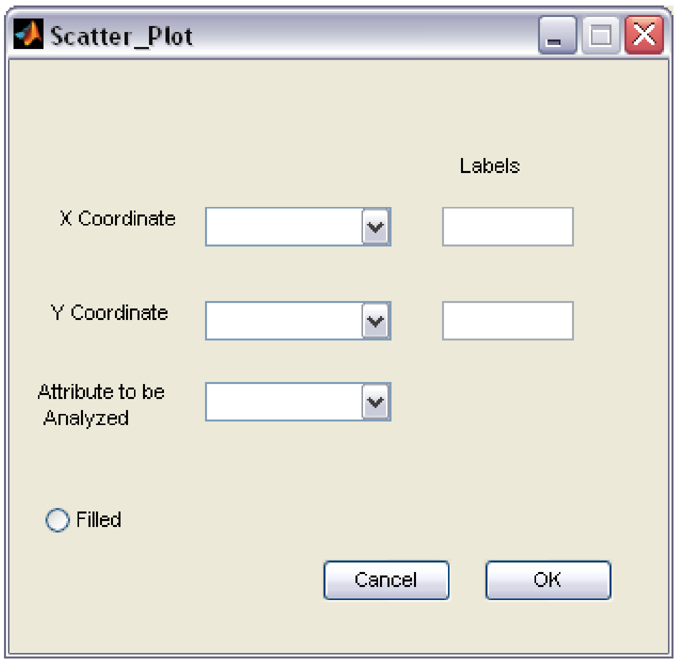


Figure 6.54: Scatter Plot

In this window, the user selects the variables for X and Y coordinated from combo boxes. He can enter labels for coordinates.

The user selects the variable from combo box for analysis.

If the user wants the view of plots as filled select the Filled radio button.

The user clicks on OK button to see the graph.

Use Case: **Scatter Plot**
ID: 5.5
Actor(s): User
Preconditions: The variables have to be imported or created and lists created.
Flow of Events: 1. The user selects “Line_Graph” submenu from Graphics menu. 2. The user selects the variables for X and Y coordinates. 3. The user enters labels for coordinates. 4. The user selects variable from combo box for analysis. 5. The user selects the Filled radio button to see plots filled. 6. The user clicks on OK button.
Post Conditions: The graph is displayed.

Figure 6.55: Use Case- Scatter Plot

7. CONCLUSIONS

This study's aim is to develop a data mining environment in MATLAB which covers all necessary data mining capabilities and it is easy to use for all type of computer users. Also the platform is designed for new changes and improvements

MATLAB is one of the commonly used data mining tools. It is a high-level language and interactive environment that enables user to perform computationally intensive tasks in fastest way. Although it has wide area of usage, it is not commonly used for data mining functionalities. So the last year, in the "Developing Data Mining Platform" project data mining functions added to MATLAB. This project is the continuation of the last year study that is preparing GUIs for data mining functions added to MATLAB and providing usage of all these functions from interfaces.

Through our study, the user interfaces designed for data mining functions. This study handles some pre-processing functions and data models, like association different from previous work.

The strong side of this tool is that it provides visuality to data mining functions and increasing user flexibility with embedding different data mining functions and models into the tool. The tool retrieves the data in spreadsheet format, this provides to see the data in organized format. More over the tool targets both novice and expert users. The power users can prefer the tool for basic applications and they can develop the algorithms of tool according to their needs.

As a further study, association tool can be embedded to the tool with modification and other data models and data mining functions can be extended. When the user interface capabilities of MATLAB will improve, the tool's user interfaces may be redesigned for better user interaction. Moreover the report capabilities of the tool can be improved and the functions and reports can serve from internet by using web services.

It is hoped that this effort of initialization will give its product in the near future, serving the needs of its users.

REFERENCES

[1] HCLAB, Design Principles, Guidelines, Heuristics
www.sis.uncc.edu/~clatulip/ITIS6400/Usability_Principles_2_6400.pdf

[2] Shneiderman, B. (1998). Designing the user interface: Strategies for effective human-computer interaction (3rd ed.) . Reading, MA: Addison-Wesley Publishing.

[3] Baysal, D., Buruncuk, G., Demirci, H. (2003), “Preliminary Study of a Data Mining Environment”, Boğaziçi University

[4] Çubuk, S., Karakoç, V., Özkan, S. (2006) “Developing a Data Mining Platform”, Boğaziçi University

[5] Shearer, Colin (2000) The CRISP-DM Model: The New Blueprint for Data Mining, Journal of Data Warehousing, Volume 5, Number 4
http://www.crisp-dm.org/News/86605.pdf

[6] www.mathworks.com

[7] Graphical User Interfaces with MATLAB
http://dsp.ucsd.edu/students/present-students/mik/matlabgui/

[8] Ashi, R, Ameri, A “Introduction to Graphical User Interface (GUI) MATLAB 6.5”, Uae Unıversıty, College Of Engıneerıng Electrıcal Engıneerıng Department, IEEE UAEU Student Branch

[9] An Effectıve Guı Programmıng Methodology
http://www.ece.gatech.edu/research/DSP/courses/ee2200/matlab/5.graphics/uiguide.htm
l

REFERENCES NOT CITED:

www.crisp-dm.org/Process/index.htm

http://www.cs.queensu.ca/home/cisc333/tutorial/Weka.html

http://www.cs.bham.ac.uk/~jxl/Tutorial/classification.php

http://www.cs.utexas.edu/~shawnyu/weka_docs/weka/classifiers/rules/

http://www.cs.waikato.ac.nz/~ml/weka/index.html

http://cs.nyu.edu/courses/fall00/G22.3033-001/weka/weka-3-0-2/Tutorial.pdf

http://www.matlabgui.com/

www.spss.com

SPSS Inc.(2006) “SPSS 15.o for Windows Tutorial”, USA

D. J. Mayhew, PRinciples and Guidelines in Software User Interface Design, Prentice Hall, 1992

http://www.kdnuggets.com/polls/2002/methodology.htm