Linked2Safety

FP7-288328

A Next-Generation, Secure Linked Data Medical Information Space for Semantically-Interconnecting Electronic Health Records and Clinical Trials Systems Advancing Patients Safety in Clinical Research

---

Deliverable D5.1

Data Analysis Space Design

Editor(s):	David Tian (UNIMAN), Athos Antoniades (UCY), Aristos Aristodemou (UCY), Hasapis Panagiotis (INTRASOFT), Ann Gledson (UNIMAN), John Keane (UNIMAN), Xiao-jun Zeng (UNIMAN), Goran Nenadic (UNIMAN)
Responsible Partner:	UNIMAN
Status-Version:	Final - v1.0

Date:	31/12/2012
EC Distribution:	Public

Project Number:	FP7-288328
Project Title:	Linked2Safety

Title of Deliverable:	D5.1 - Data Analysis Space Design
Date of Delivery to the EC:	31/12/2012

Workpackage responsible for the Deliverable:	WP5 – Linked2Safety Data Analysis Space
Contributor(s):	UNIMAN, UCY, INTRASOFT, LUH
Reviewer(s):	CERTH
Approved by:	All Partners

Abstract:	This deliverable includes the detailed specifications of Data Analysis Space software components which implements subject counting, statistical hypothesis testing, data mining, knowledge filtering and acquisition mechanism and adverse event early detection mechanism and so on.
Keyword List:	data cube, quality control, feature selection, subject count, single hypothesis testing, genetic association, data mining, association rules, adverse events detection, Galaxy server

Document Description

Document Revision History

Version	*Date*	*Modifications Introduced*	
		Modification Reason	*Modified by*
V0.1	14/12/12	rephrase of sections, expansions of sections, typos	UCY, INTRASOFT, CERTH, DERI, LUH
V1.0	23/12/12	Update of figures, rephrase of sentences, typos	UCY,CERTH, UNIMAN
V1.0	28/12/12	Formatting	UNIMAN

Contents

List of Figures

List of Tables

Definitions, Acronyms and Abbreviations

Table 1: Definitions, Acronyms and Abbreviations

Acronym	Title
API	Application Package Interface
RDF	Resource Description Framework
SPARQL	SPARQL Protocol and RDF Query Language
CSV	Comma Separated Values
HPC	High Performance Computing
QC	Quality Control
HWE	Hardy-Weinberg Equilibrium
SNP	Single Nucleotide Polymorphism
MAF	Minor Allele Frequency
REST	Representational State Transfer
RSFS	Rough Set Feature Selection
SEHR	Semantic Electronic Health Record
LD	Linkage Disequilibrium
RR	Relative Reporting Ratio
ER	Entity Relationship
PMML	Predictive Model Markup Language

1 Executive Summary

The present document is Deliverable D5.1 “Data Analysis Space Design” of the Linked2Safety project. The Data Analysis Space is responsible for subject selection, single hypothesis testing, data mining, pattern discovery, knowledge extraction and filtering, and notification of safety alerts. The design of the Data Analysis Space will be used to implement the Data Analysis Space in deliverables D5.3.1 and D5.3.2.

This deliverable presents the design of Data Analysis Space in terms of software components. Initially, the requirements of the Data Analysis Space are derived from D1.1 Requirements Analysis and other requirements such as security and speed and scalability are analysed. Then the workflow of the Data Analysis Space is designed taking into account the RDF data cubes, which are the input of this space and are provided by the Linked Medical Data Space (WP4). The workflow of the possible analyses of the space consists of the following steps: input, pre-processing, processing, post-processing and output. Later on, the components of the Data Analysis Space are identified based on the workflow of the Data Analysis Space. Each component implements a step of the workflow and the integration of the components is designed using the Galaxy web server [1,2,3]. Each component is designed in a top-down fashion as a set of modules and each module is designed in a top-down fashion as a set of algorithms. A database is designed to store the results and the workflow followed by the performed data analyses. The database is a module of the ‘Processing’ component.

2 Introduction

2.1 Document Scope

The present document is Deliverable D5.1 "Data Analysis Space Design" of the Linked2Safety project. This deliverable includes the specifications of the Data Analysis Space software components which implement the following modules: data input mechanisms, data pre-processing algorithms, subject count, statistical hypothesis testing algorithms, data mining algorithms, knowledge filtering and acquisition mechanism, a database to store such knowledge, adverse event early detection mechanism and a results visualization mechanism. Moreover the way that these components interact and will be applied is presented.

The data input mechanism is responsible for retrieving data cubes from the Linked Medical Data Space, by sending SPARQL queries to the Linked Medical Data Space. The data pre-processing techniques improve the quality and utility of the input data cube by providing quality control algorithms and feature selection algorithms. The quality control algorithms remove data that do not conform to the expected quality and handle missing values from the input data cubes, and feature selection algorithms remove noisy dimensions from the input data cubes. The subject count mechanism counts the number of subjects from a clinical site who satisfy subject selection criteria. Statistical hypothesis testing algorithms test the statistical significance of selected associations, whereas data mining algorithms are used to mine pattern-based rules and to detect adverse events from input data. The knowledge filtering and acquisition mechanism removes rules not considered to be important and acquires human knowledge related to adverse event detection. The database stores all details of data analyses. The adverse event early detection mechanism notifies the users of safety alerts relating to patient' profiles. Finally, the results visualization mechanism provides visualization functionalities for adverse events, associations and association rules.

2.2 Methodology

The Data Analysis Space was designed using the following methodology:

- **Requirements Analysis:** The functional requirements of the Data Analysis Space were derived from D1.1 Requirements Analysis [41]. Some non-functional requirements are also analysed.

- **Workflow Design of the Data Analysis Space:** The workflow of the Data Analysis Space was designed taking into account the input RDF data cubes which are provided by the Linked Medical Data Space (WP4). The workflow consists of the following steps: input, pre-processing, processing, post-processing and visualization.
- **Identification of components:** The components of Data Analysis Space were identified based on the workflow of the Data Analysis Space. Each component implements a step of the workflow.
- **Integration of Components:** The components integration approach was designed based on the Galaxy web server.
- **Design of Components:** Each component was designed in a top-down fashion as a set of modules. Each module is designed in a top-down fashion as a set of algorithms.
- **Design of Database:** A database is designed to store the details of data analysis. The database is a module of the 'Processing' component.

The document is structured as follows: section 3 analyses the requirements of the Data Analysis Space; section 4 presents the design of the Input, Pre-processing and Processing components of the Data Analysis Space; section 5 presents the design of the database; section 6 presents the design of the Post-processing component; section 7 presents the Visualization component; and section 8 presents the conclusion.

3 Requirements Analysis

This section presents the functional and non-functional requirements of the Data Analysis Space.

3.1 Functional Requirements

The functional requirements of the Data Analysis Space are derived from D1.2 Reference Architecture [1] as follows:

- F3: Implement data mining algorithms such as decision trees, random forests, association rule mining algorithms, etc. to detect safety signals.
- F4: Perform hypothesis testing for statistical evidence (e.g. to test an association between a genetic marker and a phenotype or to test a specific scenario of adverse drug events (ADEs) in order to "direct" data mining results).
- F5: Implement statistical analysis techniques such as χ^2, regression analysis, permutation testing, etc. to provide independently replicated results.
- F6: Support manual selection of association rules (e.g. among a set of drugs and a set of AEs) that have more than a given, user-defined, minimum "support" value.
- F7: Support manual selection of association rules by selecting (from a list of all possible association rules) those that have more than a given, user-defined, minimum "confidence" value.
- F8: Filter rules obtained from data mining both automatically and by visual inspection.
- F9: Manually add rules considered to be as mandatory although they were not discovered by the data mining process (e.g. because the conditions of the rules never occur or because the conditions occur but not lead to any outcome).
- F10: Manually remove rules that do not apply to general population, but to the specific healthcare data analyzed.
- F12: Search the Knowledge Base according to i) Drugs (e.g. molecular structures, detected AEs, associations with other medicines) and ii) Detected associations/indications (e.g. reports on the identified AEs, associations between molecular structures and AEs)

- F13: Insert inclusion/exclusion criteria in order to identify the number of subjects and/or the respective site(s) for participating in a clinical trial
- F14: Receive recommendations in inclusion/exclusion criteria based on statistics to obtain a better population.
- F17: Support chemoinformatic analysis to detect associations between molecular sub-structures and phenotyping, especially AEs.
- F18: Customize the signaling threshold/level based on advanced criteria (e.g. high/low sensitivity, specific drug, type of disease).
- F19: Visualize results (view statistical graphs and measures).
- F20: Identify situations at risk and generate safety alerts (based on the analysis performed on Linked2Safety data).
- F21: Implement a quality control mechanism that will identify (and remove) markers or subjects that are most likely erroneous or impossible to contribute to a true positive result due to low statistical power.

3.2 Non-functional Requirements

3.2.1 Data Security

The data security requirements of the Data Analysis Space are derived from Deliverable D1.2 Reference Architecture [1] as follows:

- NF1: Respect patients'anonymity, Linked2Safety data's ownership and privacy.
- NF2: All tasks and analyses performed within the platform will comply with legislative, regulatory and ethical requirements (European and national).

3.2.2 Speed and Scalability

Based on the size and complexity of data volume that may need to be analyzed, the following non-functional requirement concerning the speed and the scalability of the Data Analysis Space was identified:

- NF26: Perform complex and analytic tasks in acceptable times;

In addition, a new non-functional requirement has become evident from the work done as part of this deliverable:

- NF27: The Data Analysis Space should be capable of executing multiple data analysis algorithms simultaneously, as appropriate and where required.

4 Data Analysis Space Design

The section presents the design of the Data Analysis Space. Firstly, the components of the Data Analysis Space are identified. Thereafter, each of the components is described and further details such as the class diagrams and the web interfaces of the components are provided. Moreover the integration of the components is taken into consideration and the approach that will be followed is described.

4.1 Components of Data Analysis Space

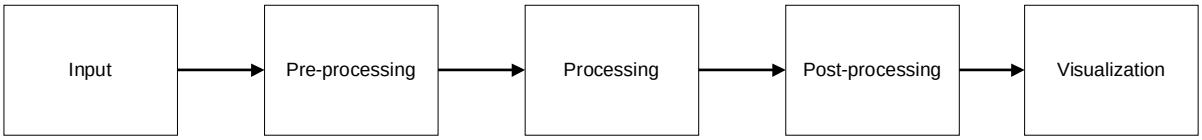

Figure 1: Workflow of Data Analysis Space

Figure 1 illustrates the workflow of The Data Analysis Space. Each step of the workflow corresponds to a software component. The Data Analysis Space (Figure 1) is composed of the following components:

- Input: this component consists of one module:
  - RDF Data Cube Retrieval: the user sends a SPARQL query to the Linked Medical Data Space (WP4) which retrieves a RDF data cube and converts the data cube to comma separated values (CSV) format.
- Pre-processing: this component consists of the following modules:

- Quality Control: this module includes analytical measures that need to be performed on the values of all subjects for a specific marker. This component comprises the following algorithms: Hardy-Weinberg Equilibrium, Allele Frequencies Test, Phenotypic, Environmental and Genetic Missing Data Test;
- Feature Selection: this module includes feature selection algorithms such as manual feature selection, rough set feature selection, information gain feature selection and Chi-squared feature selection.

- Processing: this component consists of the following modules:
  - Subjects Count: this module computes the number of subjects meeting the specified search criteria as well as displaying additional information about the recruitment sites/hospitals. Moreover, the users can receive recommendation on how to improve search results as well as view related associations;
  - Single Hypothesis Testing: this module comprises the following algorithms: Pearson's Chi Square Test, Fisher's Exact Test, Logistic Regression, Permutation Testing, Linkage Disequilibrium, Genetic Region Based Association Test, Confidence and Support and Odds Ratio;
  - Data Mining: this module comprises the following algorithms: C4.5 decision trees, Random Forest, Apriori association rules and data format transformation algorithms;
  - Adverse Event Early Detection Mechanism: this module detects adverse events of patient profiles early using the rules output by data mining.
  - PMML Generation: the mined rules are represented in PMML format;
  - Database: association rules, data analysis results and PMML files are stored in a central database. The database can be accessed by all the modules using SQL;

- Post-processing: this component consists of one module:
  - Knowledge Extraction and Filtering Mechanism: this module filters rules based on statistical tests and expert knowledge. The user can add or remove rules using the Linked2Safety platform.

- Visualization: this component visualizes outputs in the following formats:
  - Histograms;
  - Bar Charts.

During data analysis, Quality Control, Feature Selection, Single Hypothesis Testing and Data Mining components can be applied one after another in a workflow (Figure 2). The quality controls subspace and the feature selection subspace take as input data cubes and return data cubes, whereas the other two subspaces take as input data cubes and return the actual results of their respective analyses. This interaction among these four subspaces was taken into consideration when designing the data analysis space and for the selection of the Galaxy workflow management system, which is described in the following section and will be used for the integration of the components.

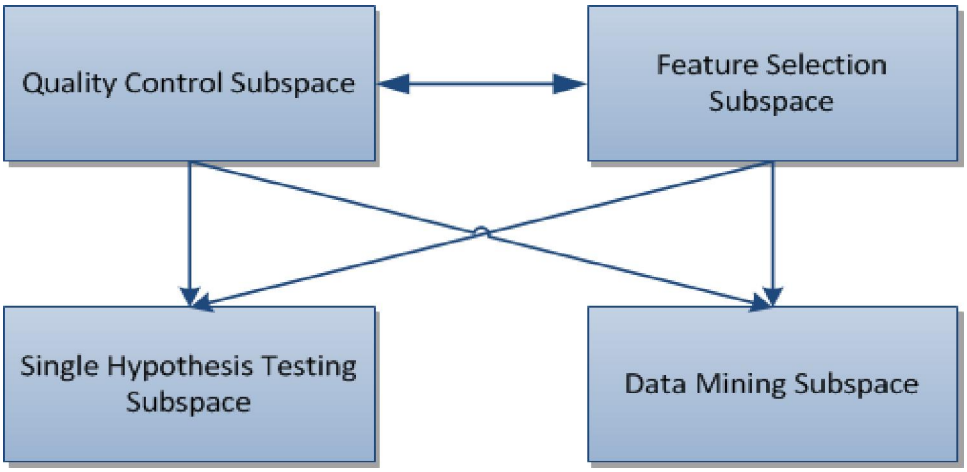

Figure 2: Workflow of pre-processing and processing components

4.2 Integration of Components

4.2.1 Galaxy Workflow Management System

Galaxy is an open, web based workflow management system that comes preloaded with a big selection of tools designed for data intensive biomedical research. It is open source and specially designed for the needs of bioinformaticians. It allows analysing data using multiple tools which are incorporated to the system through the use of workflows that can be stored for rerunning similar analyses on different datasets. Moreover Galaxy is web based, hence its operations can be performed using nothing more than a web browser. Galaxy can also be instantiated on cloud computing infrastructures or can be interfaced with grid clusters, which is an important feature for creating scalable systems that can exploit high performance computing infrastructures. The Python programming language is the primary implementation language of the Galaxy framework.

Galaxy provides a framework for easily integrating computational tools. Any tool that can be run from the command line can be wrapped in a structured well defined interface. Moreover, any scripting tool that runs from the command line or has a web interface can be wrapped up to run by Galaxy. Thus if a tool can be run from the command line it can be incorporated in Galaxy regardless of the programming language that was created.

Galaxy's principal design considerations were accessibility, reproducibility and transparency. Galaxy is accessible by scientists with no programming knowledge through the use of Galaxy tools. It facilitates reproducible computational analysis by generating metadata for each analysis step through the automated production of Galaxy history items. It promotes transparency by enabling sharing of data, tools, workflows, results and report documents.

Galaxy's web based character is accompanied by the support of user authentication that enables each user to own a space for datasets, histories and workflows. Moreover, users can belong to groups as defined by the administrative user and can set permissions on Galaxy objects like datasets, workflows and pages. Roles define how users and groups associate with permissions and datasets. Sharing also is either user based or entirely public. Administrative users are defined in the environment setup files and have the right to set permissions on library datasets, create roles and groups and associate users and groups with roles. An Administrator is able to modify the access rights and permissions for other categories of users.

Galaxy is set up as a free, public, internet accessible resource at UseGalaxy.org. Alternatively local Galaxy servers can be set up by downloading and customizing the Galaxy application. Galaxy is available as a standalone package only for Linux environments. It includes an embedded web server and an SQL database.

Finally Galaxy has a wiki page with information about the system and a large community that interacts through forums and mailing lists providing support to Galaxy users.

A screenshot of a prototype of the Linked2Safety Galaxy web interface is provided in Figure 3 below. All available analytical tools are provided on the left side of the page and as can be seen tools can be grouped in different categories. When selecting a tool, its parameters are presented to the user in the centre of the page. The right side of the web page contains the Galaxy history, where the inputs and outputs of the analyses run are stored.

As mentioned earlier, Galaxy allows the use and sharing of workflows. An example of a workflow is illustrated in Figure 4 below. The users can easily create workflows in the graphic interface of the workflow editor provided by Galaxy. This is done by dragging and dropping the tools that are needed and by connecting them according to the flow of the analysis that want to perform. In the example shown in Figure 4, the dataset will first be processed by the Hardy Weinberg Equilibrium tool and

then the output of this tool will be used as input for running Fisher's exact test and Pearson's chi square test.

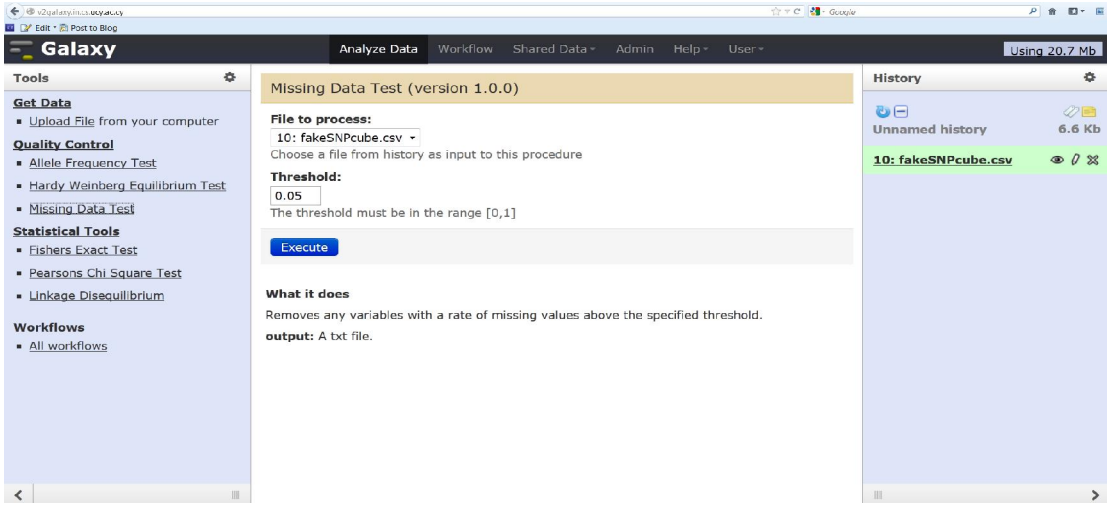

Figure 3: Galaxy Web Interface

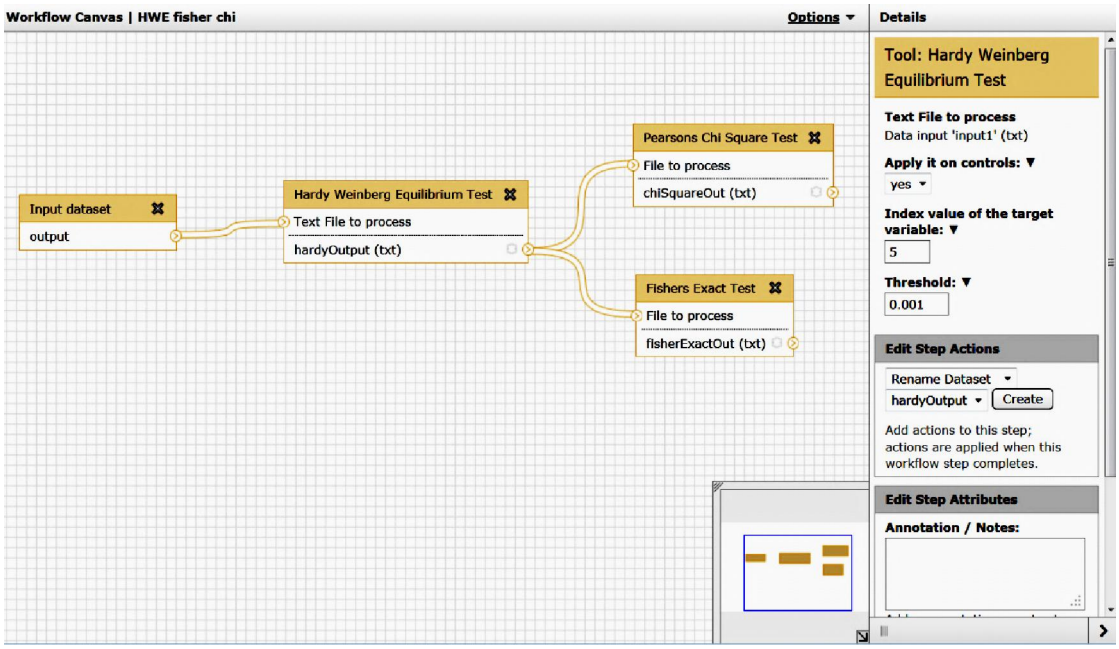

Figure 4: Galaxy workflow

4.2.2 Integration of Components into Galaxy

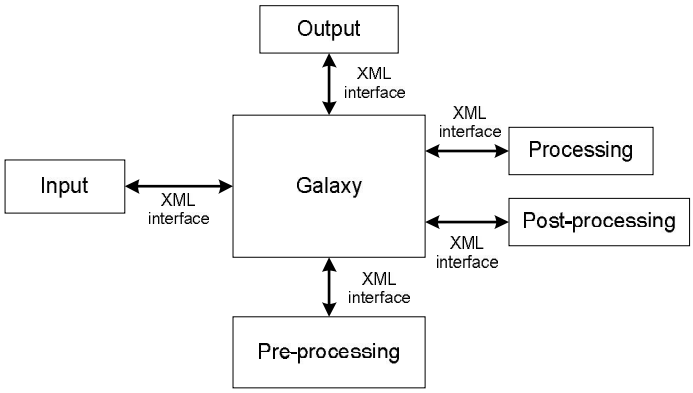

Figure 5: Components Integration into Galaxy

The Galaxy server has been implemented in Python and is extensible. Galaxy allows integration of external tools implemented in other programming language such as java, C++ etc. In order to integrate the components into Galaxy server, the following tasks must be done:

- Galaxy XML interface files are defined for the components (Figure 5);
- Each component is designed to consist of a number of modules with each module having both a command line interface and a web interface.

4.3 Input

The Input component is responsible for retrieving RDF data cubes using SPARQL queries. The output of the SPARQL query can be viewed and downloaded in various formats, such as CSV, XML, N3, JSON, Text and TSV. The RDF data cube creation and retrieval is depicted in Figure 6. The bottom layer (Figure 6) consists of a number of data sets shown in various formats which are transformed aligned based on a common electronic health record (EHR) schema (Common EHR Schema) and then to RDF data cubes based on RDF data cube vocabulary (middle layer in Figure 6) [41]. A common reference model is also needed to query these data cubes in an integrated fashion. The Linked2Safety Semantic EHR Model (SEHR), developed in Task 1.4 [3], acts as a shared and consistent ontological reference point for the data cubes originating from various clinical partners. The SEHR is a domain level ontology representing the clinical trial research domain. The goal of building the Semantic EHR Model is to enable seamless sharing and linking pieces of healthcare, i.e., EHRs and clinical

data/knowledge among the authorized stakeholders. To preserve the anonymization and security of patients (and related stakeholders) these heterogeneous resources are tailored as data cubes within the Linked2Safety data space.

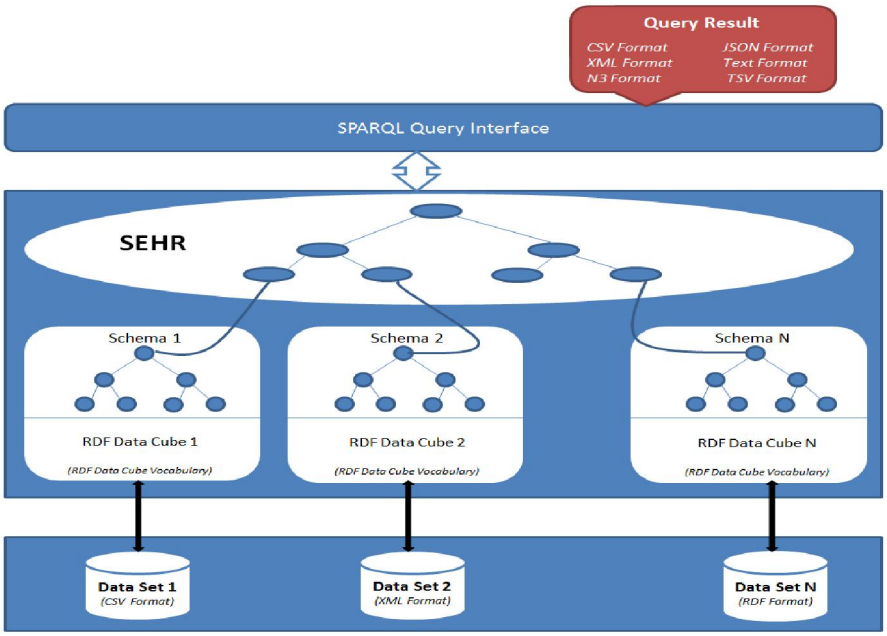

Figure 6: Linked Medical Data Space Layers: *Query, Schema and Clinical Data Cubes*

An example data set is used to illustrate the creation of RDF data cubes and then querying these data cubes with SPARQL.

Figure 7 shows a snapshot of a data set describing the total number of patients exhibiting any of four adverse events (BMI, Dyslipidaemia, Headache, and Rash). This data set is transformed to RDF data cubes using the RDF extension to Google Refine. Figure 8 shows the snapshot of a tabular representation of the RDF data cube created from the given data set. Only two of the adverse events are in the RDF data cube for simplicity. The data cube shown in Figure 8 is stored in RDF format using the RDF data cube vocabulary.

In this example there are four dimensions, namely Drug, Sex, Adverse Event 1 (BMI), and Adverse Event 2 (Dyslipidaemia). Each observation in the data cube represents the total number of patients exhibiting a particular adverse event (the measure). This data is then queried using SPARQL to get the desired results. An example SPARQL query along with the results it retrieves from the RDF data cubes is shown in Figure 9.

1	Sex	Drug	BMI>25	Dyslipidemia	Headache	Rash	Positives
2	Male	Insulin	0	0	0	0	99
3	Male	Insulin	0	0	0	1	9
4	Male	Insulin	0	0	0	9	62
5	Male	Insulin	0	0	1	0	80
6	Male	Insulin	0	0	1	1	10
7	Male	Insulin	0	0	1	9	71
8	Male	Insulin	0	0	9	0	89
9	Male	Insulin	0	0	9	1	84
10	Male	Insulin	0	0	9	9	28

Figure 7: Example dataset

	Sex	Male			Female		
Drug	BMI > 25 Dyslipidemia	0	1	9	0	1	9
Insulin	0	24	35	4	23	12	1
	1	16	32	0	20	18	2
	9	3	8	0	1	3	0
Digrin	0	12	24	3	15	25	0
	1	19	17	1	22	18	4
	9	4	2	0	1	2	1

Figure 8: Tabular representation of data cube

SPARQL Query

```
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX l2s-data:<http://linked2safety.org/data/>
PREFIX l2s-prop:<http://linked2safety.org/properties/>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX sdmx:<http://purl.org/linked-data/sdmx#>
PREFIX l2s:<http://linked2safety.org/>
PREFIX qb:<http://purl.org/linked-data/cube#>
PREFIX sdmx-code:<http://purl.org/linked-data/sdmx/2009/code#>
PREFIX sdmx-dimension:<http://purl.org/linked-data/sdmx/2009/dimension#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

SELECT ?sex ?drug ?bmi25 ?dyslipidemia ?headache ?rash ?positives FROM l2s-data:clinical-trials WHERE {
?instance a qb:Observation.                    ?instance sdmx-dimension:sex ?sex.
?instance l2s-prop:drug ?drg.                  ?drg rdfs:label "Insulin".
?drg rdfs:label ?drug.                         ?instance l2s-prop:bmi25 ?bmi25.
?instance l2s-prop:dyslipidemia ?dyslipidemia. ?instance l2s-prop:headache ?headache.
?instance l2s-prop:rash ?rash.                 ?instance l2s-prop:positives ?positives . }
```

Query Result

sex	drug	BMI > 25	dyslipidemia	Headache	Rash	Postives
male	insulin	0	0	0	0	99
male	insulin	0	0	0	1	9
male	insulin	0	0	0	9	62

Figure 9: Example SPARQL query and results returned

This query retrieves all the observations from the data cubes where the patients have been administered the drug insulin. The *positives* column in the *Query Results* shows the number of patients who received insulin with or without any side effect, depending on the values in four columns, namely *BMI>25*, *dyslipidemia*, *Headache*, and *Rash* in the *Query Result*. The value pattern in these four columns determines what type of side effect patients have or patients with no side effect.

The Input Component is summarized in Table 2. The class diagram of the Input component is shown in Figure 10. The method create_SPARQL_query takes as input the dataset selection criteria specified by the user and translates them into a SPARQL query. The method execute_SPARQL_query takes the SPARQL query and sends it to the Linked Medical Data Space to retrieve a RDF data cube which is returned as a CSV file.

Table 2: Input Component

Name	Input Component
Description	The Input Component enables users to select data cubes by constructing SPARQL queries and submit the queries to the Linked Medical Data Space (LMDS).
Responsibilities	This component is responsible for retrieving RDF data cubes using SPARQL queries from the LMDS.
Constraints	Assumption is that there are data cubes in the LMDS.
Composition	• Node to construct a SPARQL query. • Node to return the data cubes from LMDS.
Inputs	A SPARQL query or a data selection criteria.
Outputs	A data cube in a CSV file.
Interactions/ Interfaces	This component interacts with the LMDS to obtain data cubes and also interacts with the pre-processing and processing components to provide inputs for data analysis.

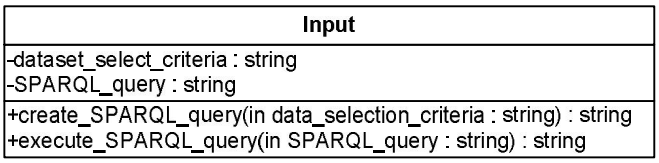

Figure 10: Class diagram of Input component

4.3.1 User Interface Design

The screenshot below (Figure 11: User Interface mock-up of the Input component) illustrates the type of "technical" interface that may be used to query data within the Linked2Safety Platform. The envisaged interface will include a query builder, which will use concepts from the SEHR, and will guide those who are not SPARQL experts through the process of building a federated query across the clinical data sets. This is in addition to an envisaged REST API which will reflect the Linked2Safety ontology and allow a developer to program against the LMDS. Additionally, a less technical interface, which will allow visual exploration of the ontology, will be designed. This visualization/navigation will result in the generation of a SPARQL query in the background, again working against the query engine. This type of user will only require knowledge and understanding of the model which describes the data – i.e. the SEHR. These interfaces will be designed as part of deliverable D4.2.1 Linked Medical Data Space.

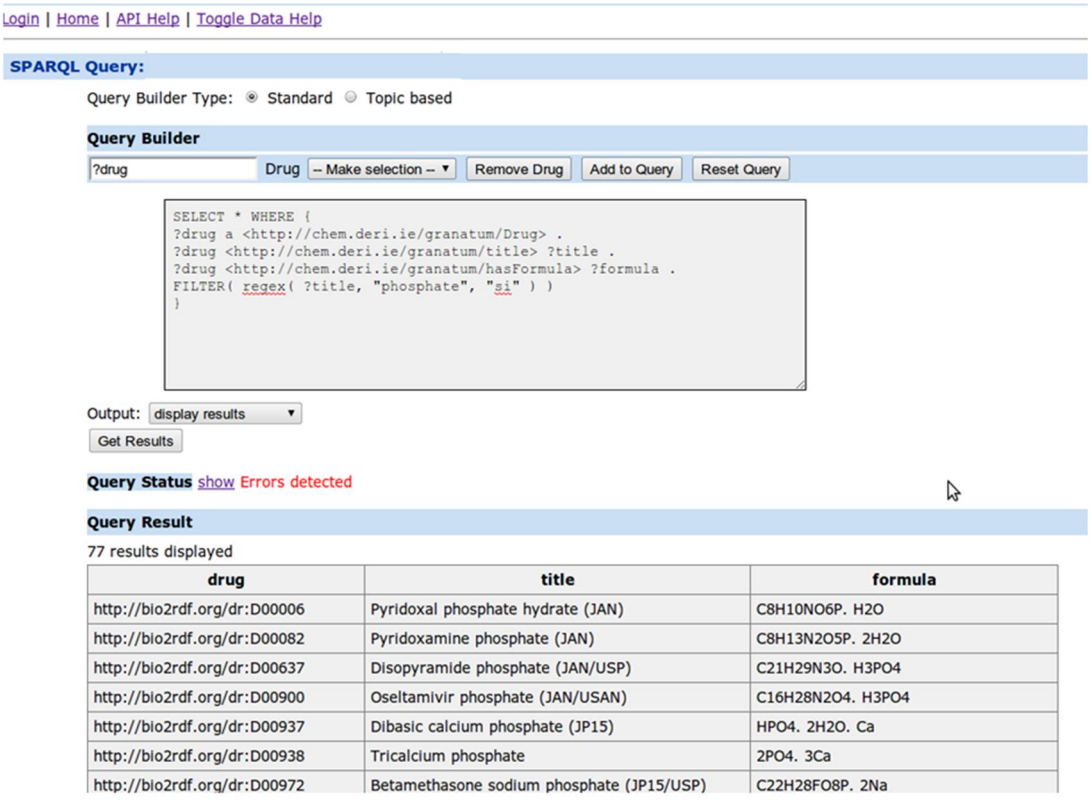

Figure 11: User Interface mock-up of the Input component

4.4 Pre-processing

The Pre-processing component is composed of the following two modules: Quality Control and Feature Selection.

4.4.1 Quality Control (QC)

4.4.1.1 Module Design

This module will provide functionalities for the exclusion of variables that do not conform to certain QC tests [42]. The QC module consists of the following algorithms:

- Hardy-Weinberg Equilibrium Test
- Allele Frequency Test
- Missing Data Test

The input of the algorithms is an RDF data cube in CSV format. The output of each algorithm will be in the same format as their input and can subsequently be used as input to the components of the pre-processing and processing steps.

4.4.1.1.1 Hardy-Weinberg Equilibrium (HWE)

This component is responsible for identifying genotyping errors within population-based datasets. The test is only applied on the genotypic markers of the input file and may use only data of the subjects with a control status or the data of all subjects for performing the test. If the algorithm only considers subjects with a control status the target variable will also be specified. The algorithm also takes as input the significance level which will be used as a threshold for rejecting the hypothesis that a genetic marker is in HWE. An exact test of HWE will be used and any markers that fail the test will be excluded from the output file.

4.4.1.1.1.1 Class Diagram

The class diagram of the Hardy-Weinberg Equilibrium component is shown in Figure 12Figure 12. The component uses the Datacube class to retrieve a data cube and then store the updated data cube once the HWE test is performed on the data.

The pseudocode of the method performHWE is provided in Figure 13Figure 13 where initially the number of homozygous minor alleles, homozygous major alleles and heterozygous alleles is calculated using only the controls or all of the subjects. Once these values are available, the pValue is calculated and any variables with a pValue less than the threshold defined by the user are removed from the data cube.

Table 3: Hardy-Weinberg Equilibrium Component

Name	Hardy-Weinberg Equilibrium component
Description	The Hardy-Weinberg Equilibrium component provides functions to identify genotyping errors within population-based datasets.
Responsibilities	The primary responsibility of this component is to identify if a marker is erroneous so that it can be removed from an analysis and reduce the probability of having erroneous results.
Constraints	The algorithm can only be applied on genotypic markers.
Composition	Node for applying the Hardy-Weinberg Equilibrium test on the input data.
Inputs	An RDF data cube in CSV format, the threshold value and whether it will be applied on all subjects or only subjects with a control status. If it is applied on controls the target variable will also be specified
Outputs	An RDF data cube in CSV format.
Interactions/ Interfaces	This component interacts with the RDF data cube retrieval to get its input and with the components of the pre-processing and processing steps to provide input for further analysis.

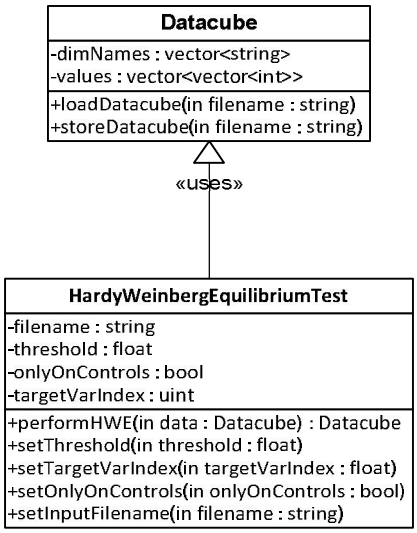

Figure 12: Hardy-Weinberg Equilibrium class diagram

performHWE

Initialize *datacube*
Require: *onlyOnControls*, *threshold*, *targetVarIndex*, *filename*
datacube.loadDatacube(filename)
for each genotype variable *datacube.values*[*i*] **do**
minor = 0,*major* = 0,*heterozygous* = 0
if onlyOnControls **then**
calculate *minor*, *major*, *heterozygous* using controls data
else
calculate *minor*, *major*, *heterozygous* using all data
end if
calculate *pvalue*
if *pvalue* < *threshold* **then**
remove *datacube.values*[*i*]
end if
end for
datacube.storeDatacube(*filename*)

Figure 13: Pseudocode of method performHWE

4.4.1.1.1.2 User Interface Design

The user interface mock-up of the HWE is illustrated in Figure 14. The user selects the input file and whether the test will use only subjects with control status. The index of the target variable will also be needed if the test will be applied only on controls. The user will have to enter the threshold value before executing the test.

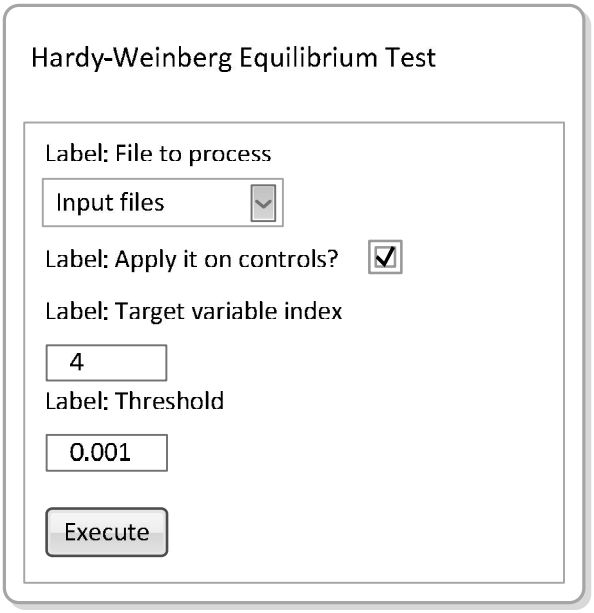

Figure 14: Hardy-Weinberg Equilibrium user interface mock-up

4.4.1.1.2 Allele Frequency Test

The Allele Frequencies Test component is responsible for identifying if a Single Nucleotide Polymorphism (SNP) has enough statistical power to be used in an analysis. The component gets as input an RDF data cube in CSV format and the minimum acceptance frequency for the Minor Allele Frequency (MAF). The test can only be applied on genotypic markers and any markers with a MAF below the specified threshold are excluded from the output file.

4.4.1.1.2.1 Class Diagram

The class diagram of the Allele Frequency Test component is shown in Figure 15Figure 15. The component uses the Datacube class to retrieve a data cube and then store the updated data cube once the test is performed on the data.

Table 4: Allele Frequency Test Component

Name	Allele Frequency Test Component
Description	The Allele Frequencies Test component identifies if a SNP has enough statistical power to be used in an analysis.
Responsibilities	The primary responsibility of this component is to identify if a SNP has enough statistical power to be used in an analysis. By removing SNPs with low statistical power, the analysis execution time is reduced.
Constraints	The algorithm can only be applied on genotypic markers.
Composition	Node for applying the Allele Frequency test on the input data.
Inputs	An RDF data cube in CSV format and the threshold value.
Outputs	An RDF data cube in CSV format.
Interactions/ Interfaces	This component interacts with the RDF data cube retrieval for getting its input and with the components of the pre-processing and processing steps to provide input for further analysis.

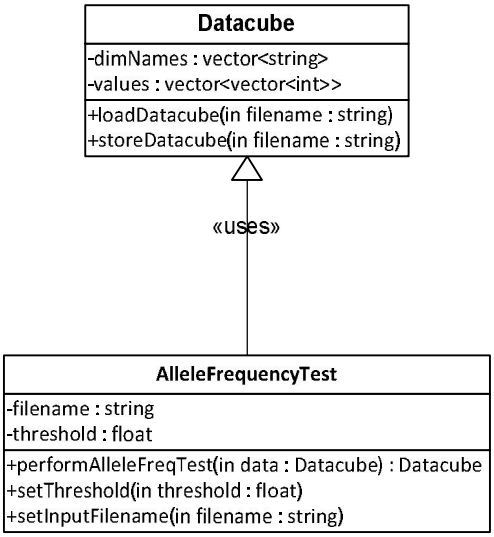

Figure 15: Allele Frequency Test class diagram

In Figure 16Figure 16 the pseudocode of the method performAlleleFreqTest is provided. Once the data cube is loaded the number of occurrence of each allele is calculated. Then the minimum frequency of the two alleles is set as the MAF and if it is above the user defined threshold, the variable is removed from the data cube. Once all genotypic variables are tested the final data cube is stored.

performAlleleFreqTest

Initialize *datacube*
Require: *threshold*, *filename*
datacube.loadDatacube(filename)
for each genotype variable *datacube.values*[*i*] **do**
$numOfAllele1 = 0, numOfAllele2 = 0$
calculate $numOfAllele1, numOfAllele2$
$total = numOfAllele1 + numOfAllele2$
$MAF = min(\frac{numOfAllele1}{total}, \frac{numOfAllele2}{total})$
if $MAF > threshold$ **then**
remove *datacube.values*[*i*]
end if
end for
datacube.storeDatacube(*filename*)

Figure 16: Pseudocode of method performAlleleFreqTest

4.4.1.1.2.2 User Interface Design

The user interface mock-up of the component is illustrated in Figure 17Figure 17. The user selects the input file and sets the minimum accepted threshold of the MAF. Once these values are added the test can be executed.

4.4.1.1.3 Missing Data Test

The Missing Data Test component (Table 5Table 5) is responsible for removing variables with a missing data rate above a user defined threshold. The component gets as input an RDF data cube in CSV format and the maximum accepted missing data rate for the variables. It then calculates the frequency of the missing values for each variable and removes any variables with a frequency above the threshold. Such variables are removed because a high frequency of missing values may indicate an error during the data collection. The output of the component is an RDF data cube in CSV format.

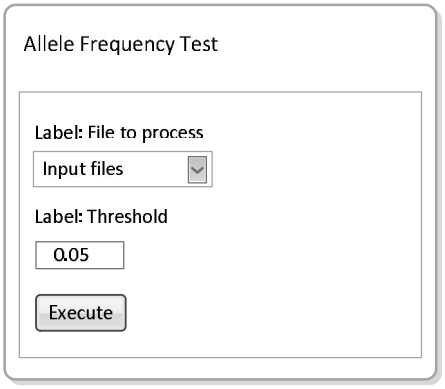

Figure 17: Allele Frequency Test user interface mock-up

Table 5: Missing Data Component

Name	Missing Data Test component
Description	The Missing Data Test component provides functions to identify if the missing data rate of a variable is acceptable or not.
Responsibilities	The primary responsibility of this component is to identify if the missing data rate of a variable is acceptable or not. By removing variables with too many missing values, the probability of getting erroneous results due to data collection errors is reduced.
Constraints	The missing values must be represented with a pre-specified sentinel value for all variables.
Composition	Node for applying the Missing Data test on the input data.
Inputs	An RDF data cube in CSV format, the threshold value and whether it will be applied on all subjects or only subjects with a control status. If it will be applied on controls the target variable will also be specified
Outputs	An RDF data cube in CSV format.
Interactions/ Interfaces	This component interacts with the RDF data cube retrieval to get its input and with the components of the pre-processing and processing steps to provide input for further analysis.

4.4.1.1.3.1 Class Diagram

Figure 18 presents the class diagram of the component. It uses the Datacube class to retrieve a data cube and then store the updated data cube once the Missing Data Test is performed on the data.

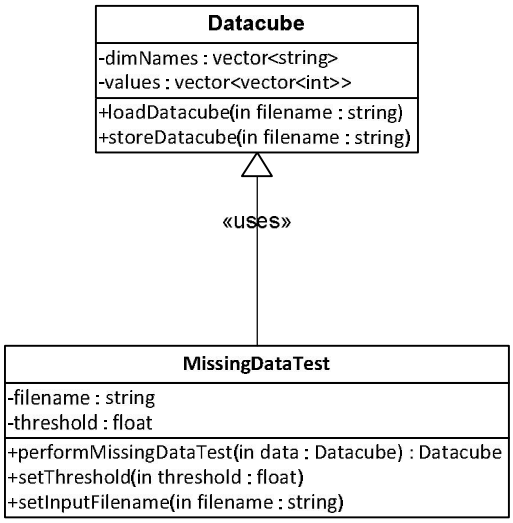

Figure 18: Missing Data test class diagram

The pseudocode of the method performMissingDataTest is shown in Figure 19. The data cube is initially loaded and the frequency of missing values for a variable is calculated. If the frequency is above the user defined threshold the variable is removed from the final data cube.

```
performMissingDataTest
  Initialize datacube
Require: threshold, filename
  datacube.loadDatacube(filename)
  for each variable datacube.values[i] do
    numOfMissing = 0, total = 0
    calculate numOfMissing, total
    freq = numOfMissing / total
    if freq > threshold then
      remove datacube.values[i]
    end if
  end for
  datacube.storeDatacube(filename)
```

Figure 19: Pseudocode of method performMissingDataTest

4.4.1.1.3.2 User Interface Design

The user interface mock-up of the component is shown in Figure 20. The user selects the input file and the maximum percentage of missing values for the variables. Once these values are entered the user can execute the test.

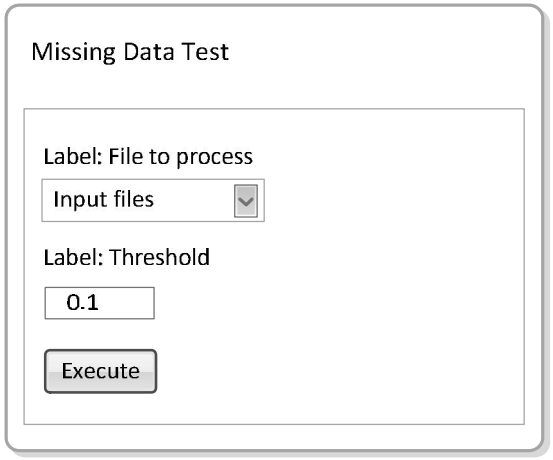

Figure 20: Missing Data Test user interface mock-up

4.4.2 Feature Selection

This module takes as input a high dimensional data cube returned by a SPARQL query and outputs a lower dimensional data cube by removing its noisy dimensions.

4.4.2.1 Module Design

This module is composed of the following algorithms:

- Manual feature selection which enables a user to select features manually;
- Rough set feature selection (RSFS) which selects features automatically using rough set theory [21-24];
- Information gain feature selection which selects a user-specified number of features which are the highest-ranked using the information gain measure [20];
- Chi-squared feature selection which selects a user-specified number of features which are the highest-ranked using the Chi-Squared measure [20].

4.4.2.2 Manual Feature Selection

The manual feature selection algorithm (Table 6Table 6) takes a subset of features from the user and removes the remaining features from the data cube. The reduce_data_cube method of the ManualFeatureSelection class (Figure 21) takes a list of features specified by the user and a data cube in CSV format. Then, the method removes those features which are not specified by the user from the data cube to give a lower dimensional data cube in CSV format.

Table 6: Manual Feature Selection

Name	Manual Feature Selection
Description	The user specifies the features (dimensions) of the data cube to select.
Responsibilities	This algorithm enables the user to manually reduce the dimensionalities of a data cube.
Constraints	The data cube should have more than one dimension.
Composition	Node to select relevant features and delete features which are not selected by the user from the data cube.
Inputs	A data cube in a CSV file and a list of features to select
Outputs	A data cube consisting of the list of features in a CSV file
Interactions/ Interfaces	This component interacts with the RDF data cube retrieval to get its input and with the components of the pre-processing and processing steps to provide input for further analysis.

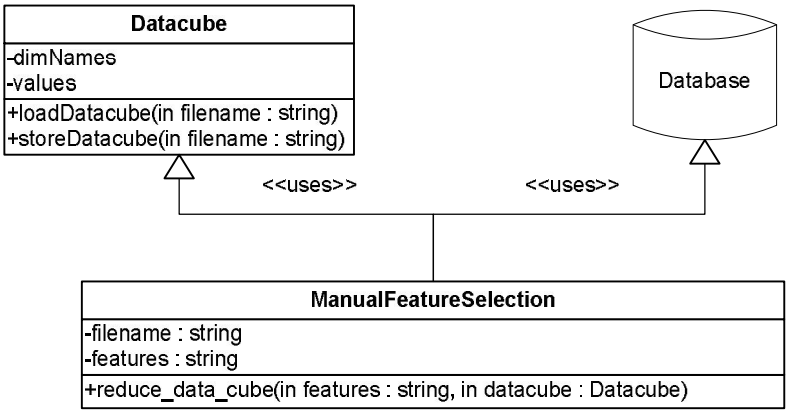

Figure 21: ManualFeatureSelection class diagram

4.4.2.2.1 User Interface Design

Figure 22Figure 22 illustrates the user interface (UI) mock-up of the manual feature selection algorithm. The indices of all features of the input data cube are listed on the right hand side of the UI. The user selects features by inputting the indices of the features at the text field "Input features indices" and clicks on the button "Reduce Data Cube" to output a lower dimensional data cube.

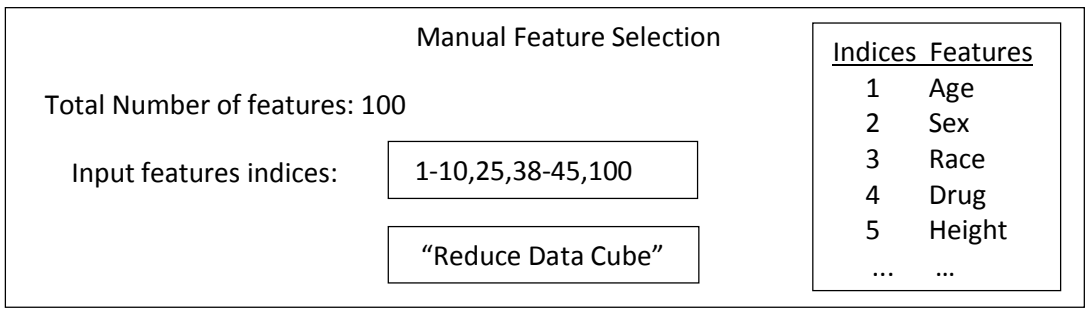

Figure 22: User Interface Mock-up of Manual Feature Selection

4.4.2.3 Rough Set Feature Selection

Rough set feature selection [21-24] outputs a data cube of lower dimensionality automatically. Firstly, a number of features are selected using rough set theory (RST) [21-24]. Then, the data cube is reduced to a lower dimensional data cube which consists of the selected features only (Table 7). The select_features method of the RoughSetFeatureSelection class (Figure 23) selects features using RST and returns the selected features as a String variable. The reduce_data_cube method takes the selected features and the data cube; then, the method outputs a lower dimensional data cube consisting of the selected features only.

Table 7: Rough Set Feature Selection

Name	Rough Set Feature Selection
Description	Given a data cube, RSFS selects a subset of features automatically using RST and returns the data cube consisting of the selected features only.
Responsibilities	RSFS automatically reduces the dimensionality of a data cube.
Constraints	The data cube should have more than one dimension.
Composition	Node to apply RSFS to a data cube.
Inputs	A data cube in a CSV file.
Outputs	A data cube of lower dimensionality in a CSV file.
Interactions/ Interfaces	This component interacts with the Input component to get input data cubes and with the components of the pre-processing and processing steps to provide input for further analysis.

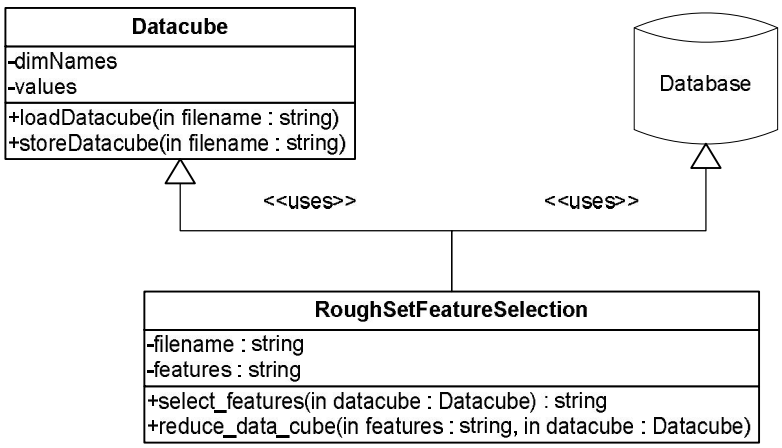

Figure 23: RoughSetFeatureSelection class diagram

4.4.2.3.1 User Interface Design

After selecting an input data cube, the user clicks on the "Reduce Data Cube" button on the rough set feature selection UI (Figure 24); then, a lower dimensional data cube is created and the selected features are also displayed on the UI.

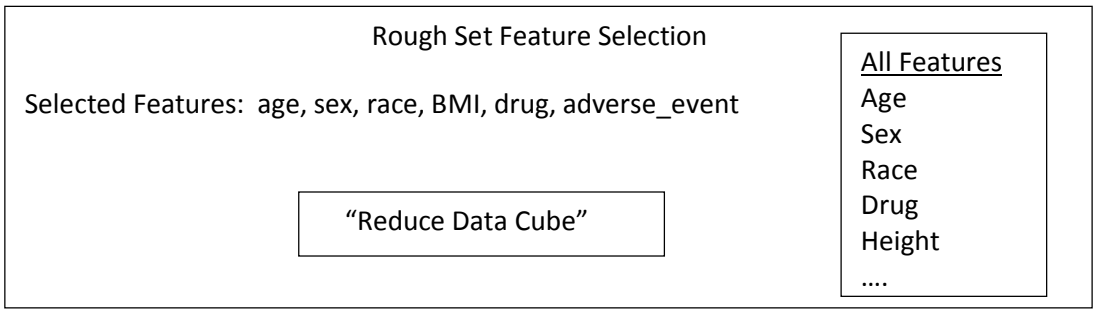

Figure 24: UI mock-up of rough set feature selection algorithm

4.4.2.4 Information Gain Feature Selection

Information gain feature selection (IGFS) takes the number K of features to select from the user and ranks all features of a data cube based on the information gain feature significance measure; then IGFS outputs the lower dimensional data cube consisting of the K highest-ranked only (Table 8Table 8). The rank_features method of the InfoGainFeatureSelection class (Figure 25) ranks all features of the data cube and returns the list of ranked features in a string variable. The reduce_data_cube method takes k the number of features to select from the user, the ranked features and the data cube; then, the method outputs the lower dimensional data cube consisting of the K highest-ranked features only.

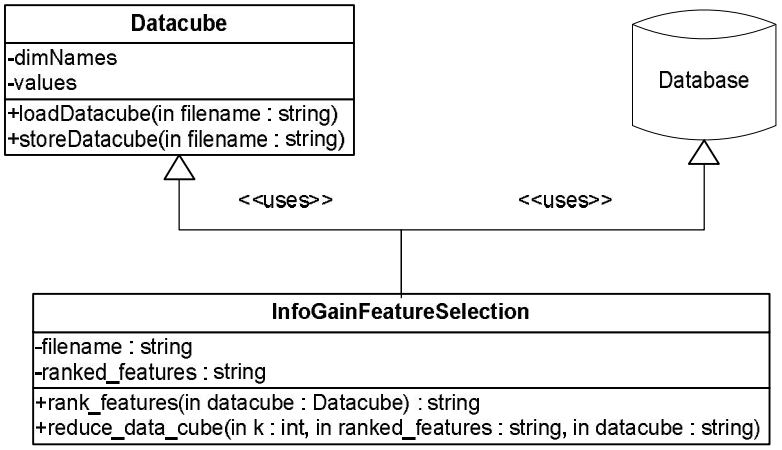

Figure 25: InfoGainFeatureSelection class diagram

Table 8: Information Gain Feature Selection

Name	Information Gain Feature Selection
Description	Given a data cube, IGFS asks the user to specify the number K of features to select; then IGFS ranks all features automatically according to the Information Gain (IG) feature significance measure and reduces the data cube so that it consists of the K highest-ranked features.
Responsibilities	IGFS reduces the dimensionality of a data cube.
Constraints	The data cube should have more than one dimension.
Composition	Node to apply IGFS to a data cube.
Inputs	A data cube in a CSV file and the number of features to select
Outputs	A data cube of lower dimensionality in a CSV file.
Interactions/ Interfaces	This component interacts with the Input component to get input data cubes and with the components of the pre-processing and processing steps to provide input for further analysis.

4.4.2.4.1 User Interface Design

Having selected an input dataset using the Input component, the user inputs the number K of features to select in the text field “Input number of features” of the Information Gain Feature Selection UI (Figure 26). Then, clicking on the button “Reduce data cube” outputs a lower dimensional data cube which consists of the K highest-ranked features only.

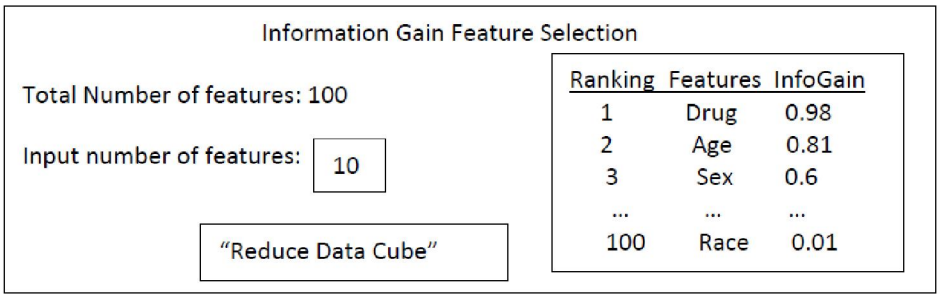

Figure 26: UI mock-up of information gain feature selection algorithm

4.4.2.5 Chi-squared Feature Selection

Chi-squared feature selection (CSFS) takes the number K of features to select from the user and ranks all the features of a data cube based on the chi-squared feature significance measure; then CSFS outputs the lower dimensional data cube consisting of the K highest-ranked only (Table 9). The rank_features method of the ChiSquaredFeatureSelection class (Figure 27) ranks all the features of the data cube and returns the list of ranked features in a string variable. The reduce_data_cube method takes k, the number of features to select from the user, the ranked features and the data cube; then, the method returns the lower dimensional data cube consisting of the K highest-ranked features only.

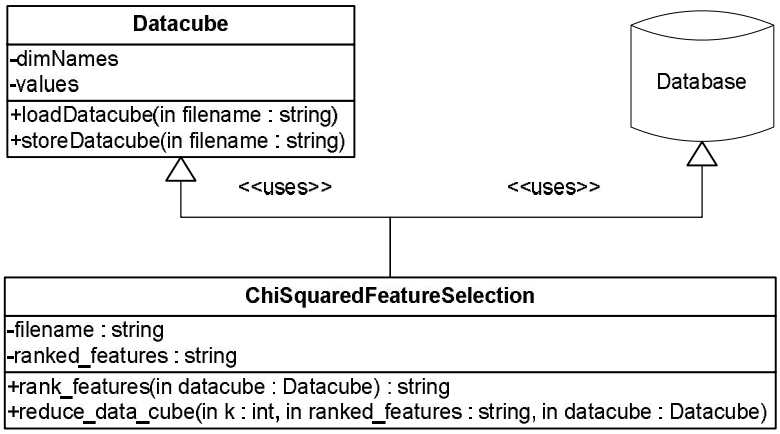

Figure 27: ChiSquareFeatureSelection class diagram

Table 9: Chi-squared Feature Selection

Name	Chi-Squared Feature Selection
Description	Given a data cube, CSFS asks the user to specify the number K of features to select; then CSFS ranks all the features automatically according to the Chi-Squared (CS) feature significance measure and reduces the data cube so that it consists of the K highest-ranked features.
Responsibilities	CSFS reduces the dimensionality of a data cube.
Constraints	The data cube should have more than one dimension.
Composition	Node to apply CSFS to a data cube.
Inputs	A data cube in a CSV file and the number of features to select.
Outputs	A data cube of lower dimensionality in a CSV file.
Interactions/ Interfaces	This component interacts with the Input component to get input data cubes and with the components of the pre-processing and processing steps to provide input for further analysis.

4.4.2.5.1 User Interface Design

Having selected an input dataset using the Input component, the user inputs the number K of features to select in the text field “Input number of features” of the Chi-Squared Feature Selection UI (Figure 28Figure 28). Then, clicking on the button “Reduce data cube” outputs a lower dimensional data cube which consists of the K highest-ranked features only.

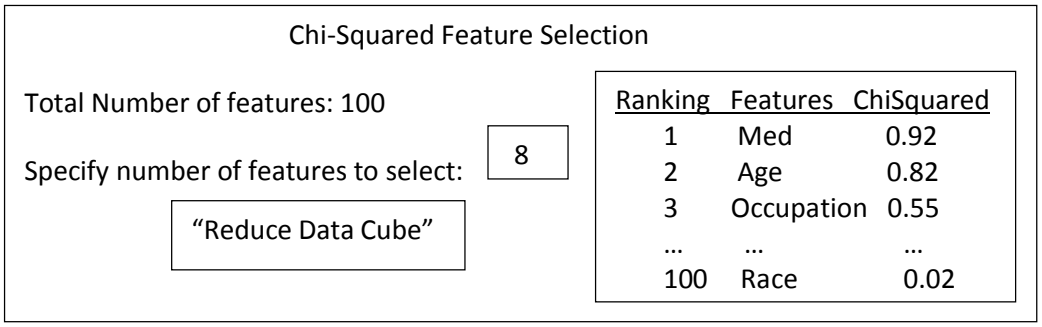

Figure 28: Chi-squared feature selection user interface

4.5 Processing

4.5.1 Subject Count

This module returns the number of subjects at a specific hospital/site that match the user's search criteria and the details related to that hospital/site.

4.5.1.1 Module Design

The count of subjects can be obtained by querying RDF cubes using SPARQL (Figure 6). The output of the SPARQL query can be viewed and downloaded in various formats, such as CSV, XML, N3, JSON, Text and TSV. Subject count module is summarized in Table 10.

The class diagram for subject counting is shown in Figure 29. The method create_SPARQL_query takes as input the subject selection criteria specified by the user and translates the selection criteria to a SPARQL query. The method execute_SPARQL_query sends to the Linked Medical Data Space and returns the count of the subjects and any information related to the selection sites.

Table 10: Subject Count module

Name	Subject Count Component
Description	The Subject Count component enables a user to count the number of subjects of a clinical site who satisfy certain inclusion/exclusion criteria.
Responsibilities	This component is responsible for count subjects using by querying RDF data cubes stored in LMDS using SPARQL queries.
Constraints	Assumption is that data cubes exist in the LMDS.
Composition	• Node to construct a SPARQL query to count the number of subjects who satisfy certain selection criteria. • Node to return the data cubes from LMDS.
Inputs	Subject selection criteria
Outputs	Count of subjects and any related information of the clinical site
Interactions/ Interfaces	This component interacts with the LMDS to obtain count of subjects and insert results into the database.

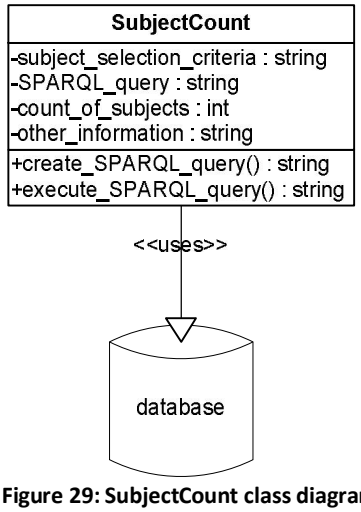

Figure 29: SubjectCount class diagram

4.5.1.2 User Interface Design

Figure 30 illustrates the user interface mock-up of the Subject Count module.

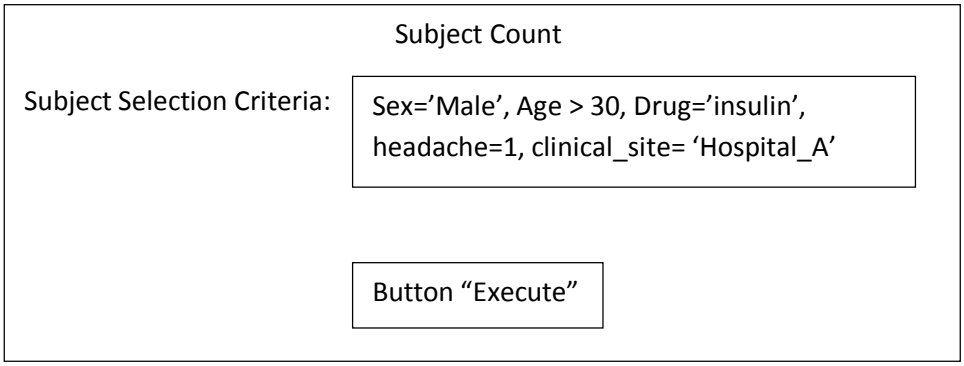

Figure 30: Subject Count User Interface Mock-up. The user specifies subjects selection criteria of a clinical site as a list of variable-value pairs and clicks the "Execute" button; then, a count of the subjects satisfying the selection criteria is displayed on the UI "Results of Subject Count" (Figure 31); any other information related to the clinical site are also displayed; clicking on the "Save" button stores the results into the database.

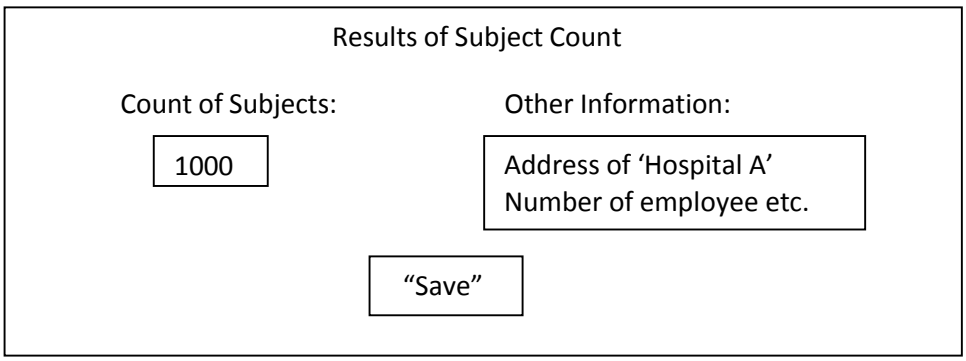

Figure 31: Results of subject count UI mock-up

4.5.2 Single Hypothesis Testing

4.5.2.1 Module Design

This module provides a set of algorithms that can be used for single hypothesis testing [26-30]. It comprises the following algorithms:

- Pearson's Chi Square Test
- Fisher's Exact Test
- Logistic Regression
- Linkage Disequilibrium
- Genetic Region Based Association Test
- Permutation Test
- Odds Ratio

The input of the algorithms is an RDF data cube in CSV format. The output of each algorithm can be stored in the database.

4.5.2.2 Pearson's Chi Square Test

This component is responsible for testing the association between variables using Pearson's Chi Square test [31-35] (see Table 11). The component gets as input an RDF data cube in CSV format, the identifier of the target variable and some parameters regarding the test. Then the chi square value is calculated and using the degrees of freedom that are automatically calculated, a p-value is computed and returned to the user.

4.5.2.2.1 Class Diagram

The class diagram of the Pearson's Chi Square Test component is shown in Figure 32. The component uses the Datacube class to load a data cube when performing the test. The parameters of the class are initialized with the method setParameters(). The method performChiSquareTest() gets as input the data cube and calculates the p-value for the association that is tested. Initially a contingency table is created from the data cube and the chi square value is calculated using that table. If "yatesCorrection" is true, Yate's correction is applied when calculating the chi square and if "removeZeroCR" is true, any rows or columns with only zeros are removed from the table before calculating the chi square. Finally using the chi square value and the degrees of freedom the p-value is calculated. The output and the parameter settings can be stored in the database.

4.5.2.2.2 User Interface Design

The user interface mock-up of the component is illustrated in Figure 33. The user selects the input file and the target variable of the association. Then the user specifies if rows/columns with only zeros should be removed and if Yate's correction will be applied. Once these values are added the test can be executed.

Table 11: Pearson's Chi Square Test Component

Name	Pearson's Chi Square Test component
Description	The Pearson's Chi Square Test component provides the tools to test the association between variables. The association is tested using Pearson's chi square test.
Responsibilities	The primary responsibility of this component is to test the association between the variables using Pearson's chi square test. The p-value returned by this component is used by the user to accept or reject the null hypothesis.
Constraints	• This is an approximation test and if the expected frequencies are too low the approximation is not reliable. • If we have 1 degree of freedom, the approximation is not reliable if we have expected frequencies below 10. If this is so then Yates Correction can be used for coping with this problem. • If we have more than 1 degree of freedom then no more than 20% of the expected frequencies should be below 5. In this case Yates correction cannot be used and the chi square test should be avoided. • If any rows or columns in the contingency table created for the test contain only zero values the test cannot be performed.
Composition	Node for applying Pearson's chi square test on the input data.
Inputs	An RDF data cube in CSV format, the identifier of the target variable, a Boolean value that specifies if Yate's correction should be applied and a Boolean value that specifies if any rows/columns with only zero values should be removed before running the test.
Outputs	The association tested and the p-value.
Interactions/ Interfaces	This component interacts with the RDF data cube retrieval to get its input and with the components of the pre-processing steps as their output can be used as input for this component. The output of the component and its parameter settings can be stored in the database.

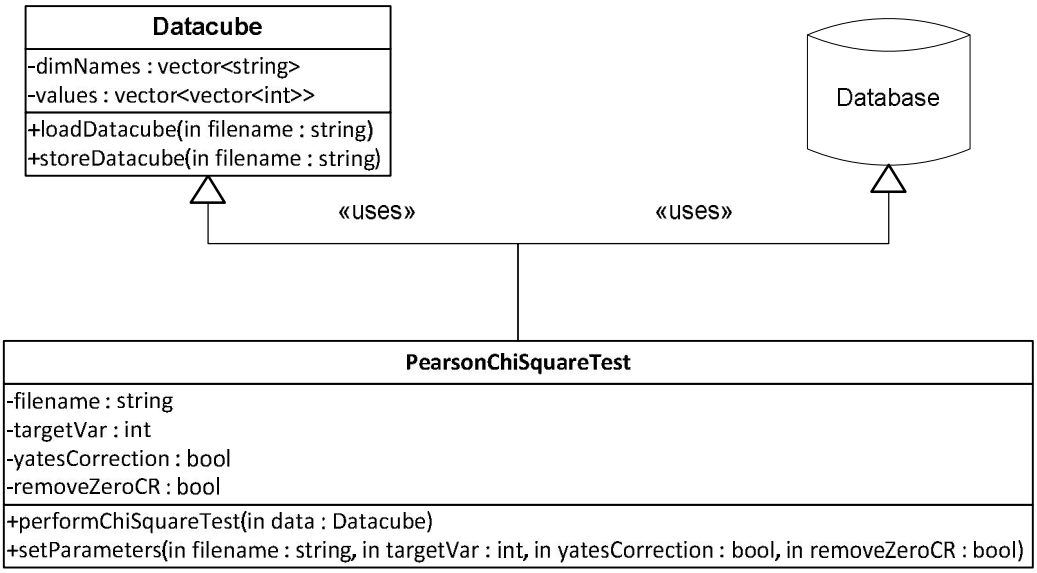

Figure 32: Pearson's Chi Square Test class diagram

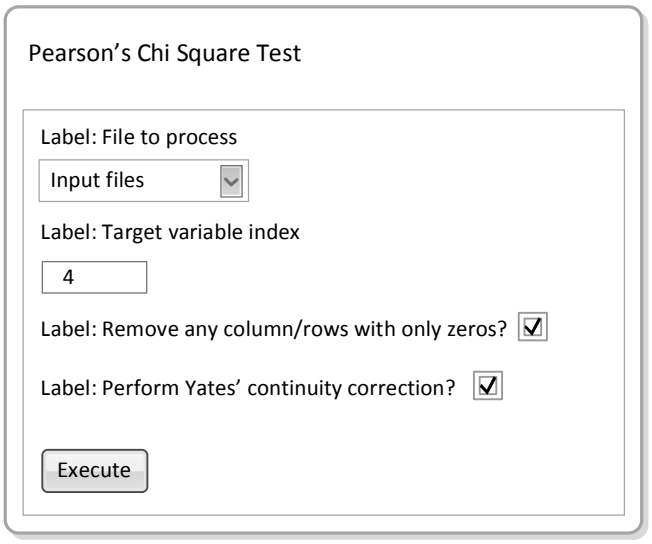

Figure 33: Pearson's chi square test user interface mock-up

4.5.2.3 Fisher's Exact Test

This component is responsible for testing the association between variables using Fisher's Exact test [34-40]. The component gets as input an RDF data cube in CSV format and the identifier of the target variable. Then a p-value is computed and returned to the user.

Table 12: Fisher's Exact Test Component

Name	Fisher's Exact Test component
Description	The Fisher's Exact Test component provides the tools to test the association between variables. The association is tested using Fisher's exact test.
Responsibilities	The primary responsibility of this component is to test the association between variables using Fisher's Exact test. The p-value returned by this component is used by the user to accept or reject his/her null hypothesis.
Constraints	Fisher's Exact test uses factorials hence it should not be used in cases where the values in a contingency table are very large.
Composition	Node for applying Fisher's exact test on the input data.
Inputs	An RDF data cube in CSV format and the identifier of the target variable.
Outputs	The association tested and the p-value.
Interactions/ Interfaces	This component interacts with the RDF data cube retrieval to get its input and with the components of the pre-processing steps as their output can be used as input for this component. The output of the component and its parameter settings can be stored in the database.

4.5.2.3.1 Class Diagram

Figure 34 illustrates the class diagram of the component. The Datacube class is used to load a data cube and use it when performing Fisher's exact test. The index of the target variable is stored in the

variable “targetVar” using the method setTargetVar(). The method performFishersExactTest() gets as input the data cube and creates a contingency table based on the target variable. Then Fisher’s exact test is applied on the contingency table and the p-value is calculated. The output and the parameter settings can be stored in the database.

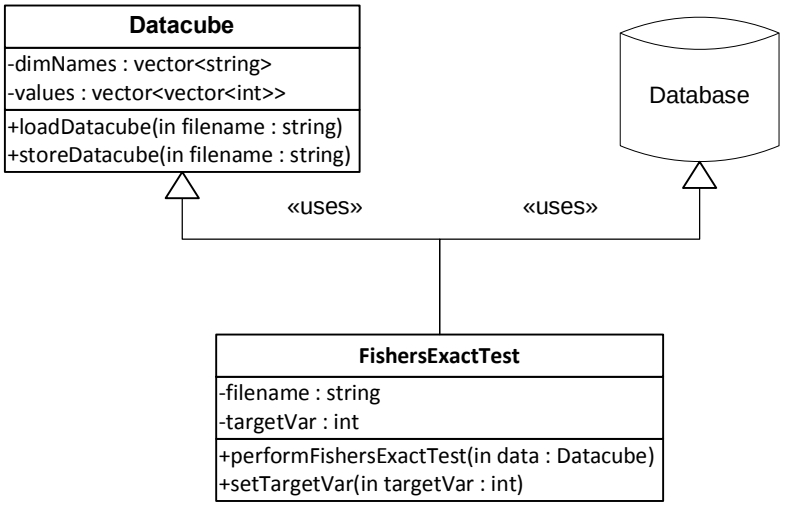

Figure 34: Fisher’s exact test class diagram

4.5.2.3.2 User Interface Design

The user interface mock-up of the component is shown in Figure 35. The user selects the input file and the target variable of the association and can then execute the test.

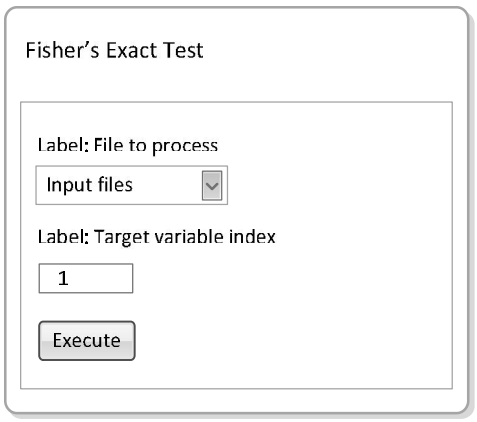

Figure 35: Fisher’s exact test user interface mock-up

4.5.2.4 Logistic Regression

The Logistic Regression [34-40] component is responsible for testing for the association between the predictor variables and the target variable. The component gets as input an RDF data cube in CSV format, the identifier of the target variable and the maximum number of iterations in case there is no convergence. The component tries to predict the values of the target variable by adjusting the coefficients of the predictor variables. Once the training of the classifier is finished, the coefficients of the model and the p-values of the predictor variables are returned.

Table 13: Logistic Regression Component

Name	Logistic regression component
Description	The logistic regression component provides the tools to test for the association between the predictor variables and the target variable.
Responsibilities	The primary responsibility of this component is to test for the association between the predictor variables and the target variable. It can also be used as a classifier and predict whether an AE may or may not occur to a subject.
Constraints	• If the algorithm does not converge and the maximum number of iterations is large it may take a long time. • The target variable needs to be binary. • Subjects with missing values must be removed from the dataset.
Composition	Node for applying logistic regression on the input data.
Inputs	An RDF data cube in CSV format, the identifier of the target variable and the maximum number of iterations.
Outputs	The coefficients of the model and the p-values of the predictor variables.
Interactions/ Interfaces	This component interacts with the RDF data cube retrieval to get its input and with the components of the pre-processing steps as their output can be used as input for this component. The output of the component and its parameter settings can be stored in the database.

4.5.2.4.1 Class Diagram

The class diagram of the logistic regression component is shown in Figure 36. The component uses the Datacube class to load a data cube when performing the test and the parameters of the class are initialized with the method setParameters(). The method performLogisticRegression() gets as input the data cube and fits a model on the data. Then the coefficients of the model and the p-values of the predictor variables are calculated. The results and the parameter settings of the algorithm can be stored in the database.

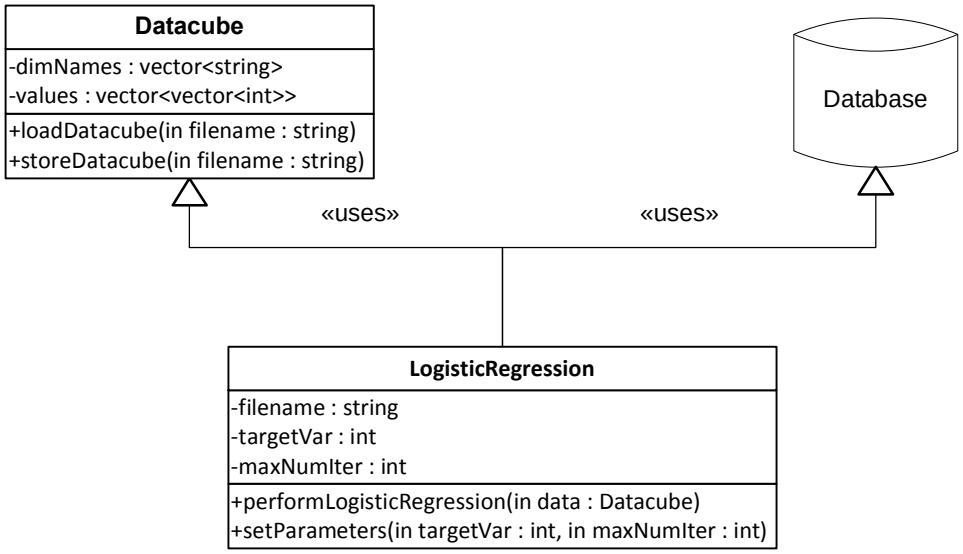

Figure 36: Logistic regression class diagram

4.5.2.4.2 User Interface Design

The user interface mock-up of the component is illustrated in Figure 37. The user selects the input file and the identifier of the target variable. Then the user specifies the maximum number of iterations of the algorithm in case there is no convergence. Once these values are added the test can be executed.

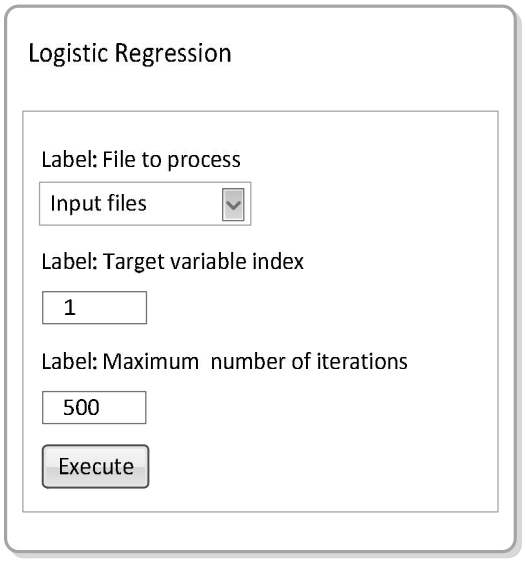

Figure 37: Logistic regression user interface mock-up

4.5.2.5 Linkage Disequilibrium

This component is responsible for calculating the linkage disequilibrium (LD) [34-40] value between two SNPs. The component gets as input the data cube that contains the SNPs and the identifiers of the two SNPs that will be tested. Moreover the LD measure that will be used is selected and then the LD value is calculated.

4.5.2.5.1 Class Diagram

Figure 38 illustrates the class diagram of the linkage disequilibrium component. The component uses the Datacube class to load a data cube when performing the test. Then the frequencies of the two SNP combinations are calculated and their LD value is provided to the user. The output and the parameters of the algorithm can be stored in the database.

4.5.2.5.2 User Interface Design

The user interface mock-up of the component is presented in Figure 39. The user can select the input file that contains the SNPs that he/she wants to test. Then the user specifies the index value of the two SNPs and the LD measure that will be used. Once these values are added the test can be executed.

Table 14: Linkage Disequilibrium Component

Name	Linkage disequilibrium component
Description	The linkage disequilibrium component provides the tools for calculating the LD value between two SNPs.
Responsibilities	The main responsibility of this component is to calculate the LD value between two SNPs.
Constraints	The combination of the two SNPs must exist in a data cube for this analysis to run. It cannot combine SNPs from different data cubes.
Composition	Node for applying linkage disequilibrium on the input data.
Inputs	An RDF data cube in CSV format, the identifier of the two SNPs and the LD measure that will be used.
Outputs	The two SNPs that were tested and their LD.
Interactions/ Interfaces	This component interacts with the RDF data cube retrieval tp get get its input and with the components of the pre-processing steps as their output can be used as input for this component. The output of the component and its parameter settings can be stored in the database.

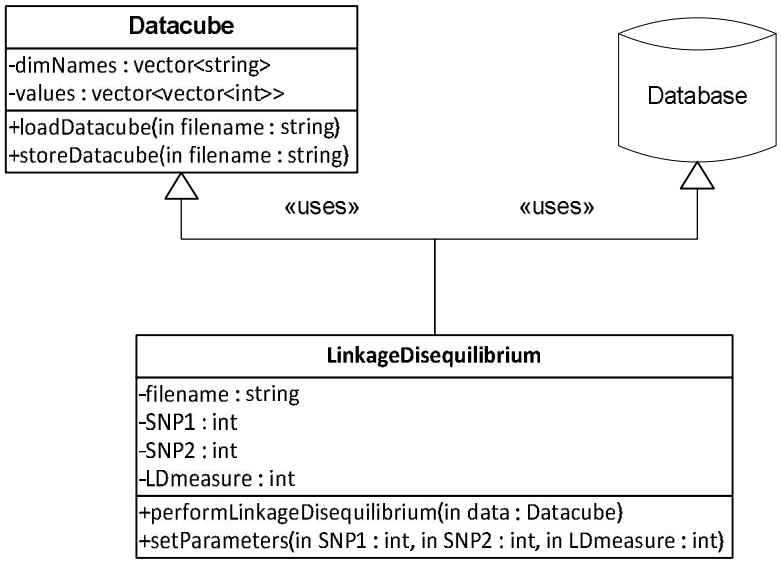

Figure 38: Linkage disequilibrium class diagram

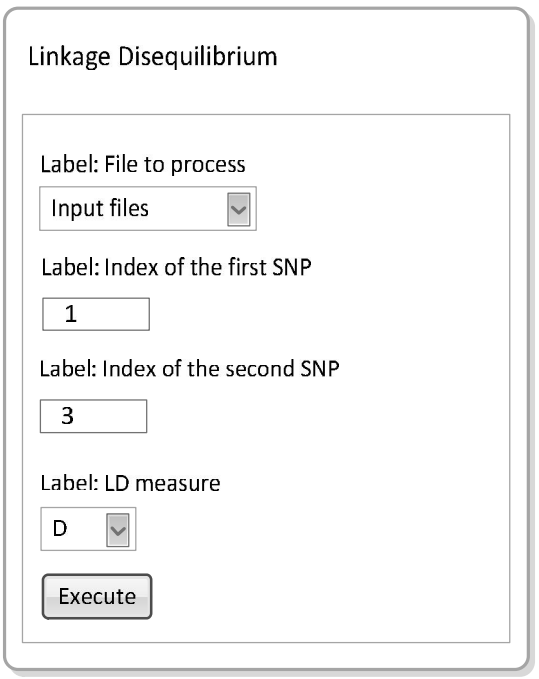

Figure 39: Linkage disequilibrium user interface mock-up

4.5.2.6 Genetic Region Based Association Test

The Genetic Region Based Association Test [34-40] component is responsible for identifying the association between a genetic region and a specific variable. The component gets as input the start and finish chromosomal position (given as Chromosome, first base pair, last base pair) and the variable that the region is tested for association with. It then returns the estimated p-value for the association test between the defined genetic region and the variable of interest.

Table 15: Genetic Region Based Association Test Component

Name	Genetic Region Based Association Test component
Description	The Genetic Region Based Association Test component identifies the association between a genetic region and a specific variable.
Responsibilities	This component tests for association between a genetic region and a specific variable. This component identifies SNPs within the region of interest, performs association tests with the relevant component and then this component merges the results together to output a single p-value that represents the estimated association between the genetic region and the target variable.
Constraints	• Genetic data must be available in the region of interest. • If the region specified contains a large number of SNPs the process might have a high execution time.
Composition	Node for applying genetic region based association test on the input data.
Inputs	An RDF data cube in CSV format, the chromosome, the starting base pair position, the ending base pair position and the target variable.
Outputs	The association tested and the p-value.
Interactions/ Interfaces	This component interacts with the RDF data cube retrieval to get its input and with the components of the pre-processing steps as their output can be used as input for this component. The output of the component and its parameter settings can be stored in the database.

4.5.2.6.1 Class Diagram

In Figure 40 the class diagram of the component is presented. The Datacube class is used to load a data cube whereas the setParameters() method is used to initialize the variables of the GeneticRegionBasedAssociationTest class. The method performTest() retrieves all necessary data cubes and calculates the p-value for the association among each SNP and the target variable. It then combines the results to calculate a p-value for the association of the genetic region and the target variable. The output and the parameters settings of the algorithm can be stored in the database.

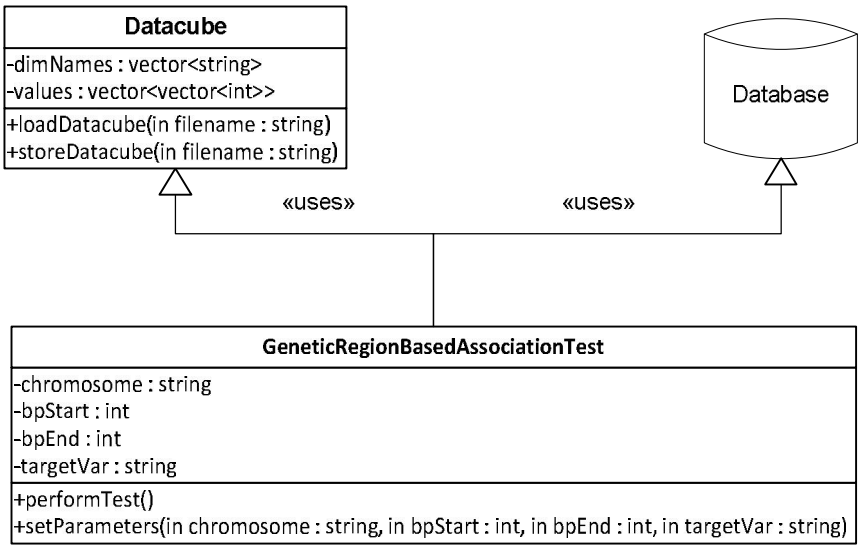

Figure 40: Genetic region based association test class diagram

4.5.2.6.2 User Interface Design

The user interface mock-up of the component is shown in Figure 41. To execute the test, the user needs to specify the region of interest. This is done by providing the chromosome, the starting base pair position and the ending base pair position of the region. Then the target variable is defined and the test can be executed.

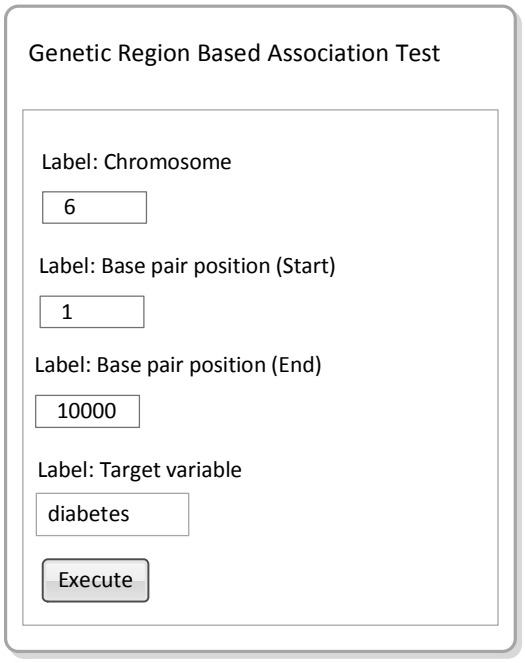

Figure 41: Genetic region based association test user interface mock-up

4.5.2.7 Permutation Test

This component is responsible for performing a permutation test [34-40] on the association that needs to be tested. The component takes as input the RDF data cube in CSV format and the identifier of the target variable of the association tested. The component then outputs the p-value of the association tested.

4.5.2.7.1 Class Diagram

Figure 42 illustrates the class diagram of the permutation test component. The component uses the Datacube class to load a data cube when performing the test. Once the data are available, the p-value of the association tested is calculated. Then the values of the target variables are shuffled many times in such a way that the initial distribution of its values is according to the distribution in the original dataset. Each time the dataset's target values are rearranged, the p-value of the association in the rearranged dataset is calculated. Finally the p-values obtained by the original dataset and the rearranged datasets are used to calculate the final p-value of the association. The results and the parameters values can be stored in the database.

Table 16: Permutation Test Component

Name	Permutation test component
Description	Permutation tests are non-parametric tests that use random shuffles of the data to get the correct distribution of a test statistic under a null hypothesis. The shuffling is done in a way such that the initial distribution of the values of the target variable is the same as in the original dataset.
Responsibilities	The main responsibility of this component is to test a hypothesis using the permutation test approach.
Constraints	This method is computationally expensive.
Composition	Node for applying the permutation test on the input data.
Inputs	An RDF data cube in CSV format and the target variable.
Outputs	The association tested and the p-value.
Interactions/ Interfaces	This component interacts with the RDF data cube retrieval to get its input and with the components of the pre-processing steps as their output can be used as input for this component. The output of the component and its parameter settings can be stored in the database.

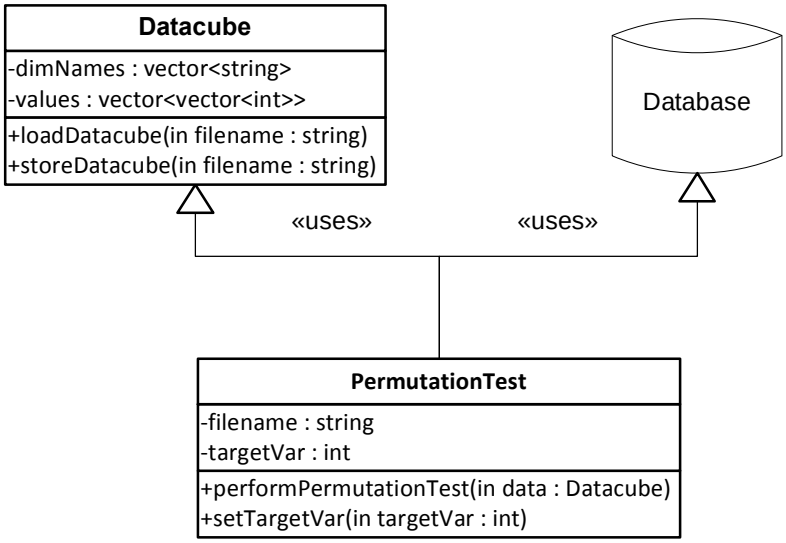

Figure 42: Permutation test class diagram

4.5.2.7.2 User Interface Design

The user interface mock-up of the component is presented in Figure 43Figure 43. The user specifies the input file and the identifier of the target variable. Then the user can execute the test by pressing the "Execute" button.

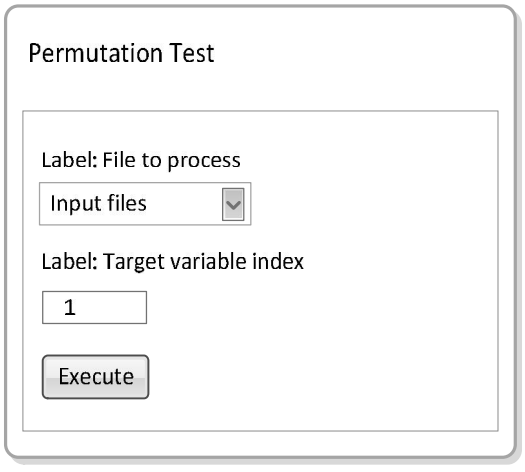

Figure 43: Permutation test user interface mock-up

4.5.2.8 Confidence and Support

This component is responsible for calculating the confidence [42] and support [42] of specific variables' values. The component gets as input an RDF data cube in CSV format and the values of the variables that are tested and outputs the confidence and support.

Table 17: Confidence and Support Component

Name	Confidence and support component
Description	The Confidence and Support component provides the tools that are needed to compute the confidence and support for specific variables' values.
Responsibilities	This component is responsible for computing the confidence and support for specific variables' values.
Constraints	The combinations of variables' values must exist in the dataset.
Composition	Node for applying confidence and support on the input data.
Inputs	An RDF data cube in CSV format and the variables' values.
Outputs	Confidence and support
Interactions/ Interfaces	This component interacts with the apriori algorithm.

4.5.2.8.1 Class Diagram

The class diagram of this component is shown in Figure 44. The component uses the Datacube class for obtaining the values of the data to be analyzed and the values of interest for each variable that is being tested. It then calculates the confidence and support and returns it. The result can be stored in the database.

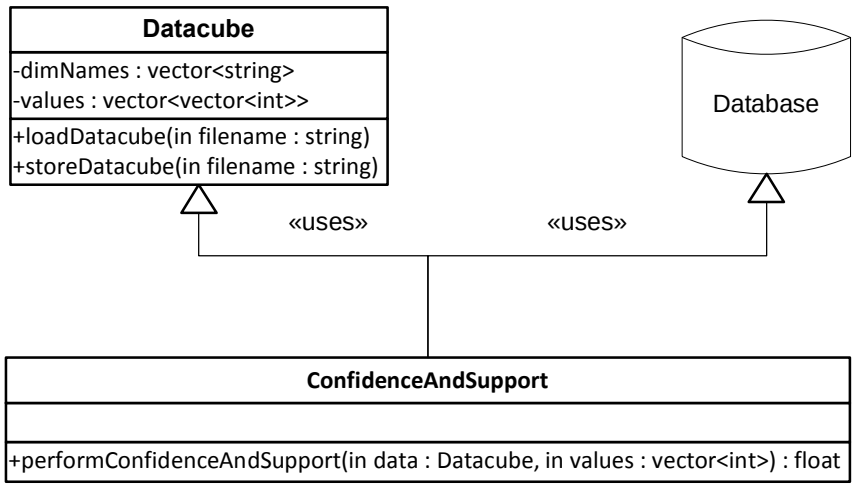

Figure 44: Confidence and support class diagram

4.5.2.8.2 User Interface Design

The user interface mock-up of confidence and support is illustrated in Figure 45. The user selects the file that will be processed and enters the values of interest for each variable of the input file. Once these values are provided the confidence and support may be calculated.

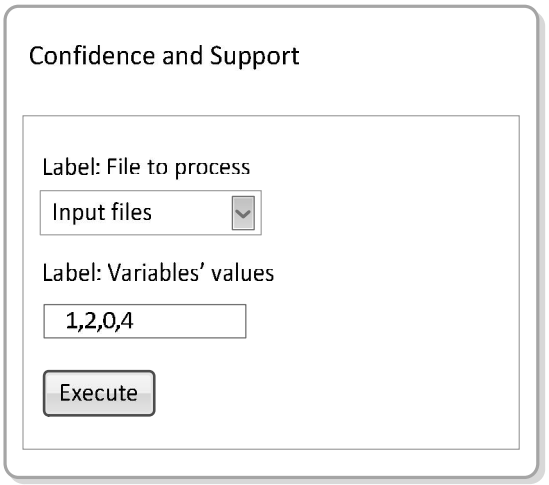

Figure 45: Confidence and support user interface mock-up

4.5.2.9 Odds Ratio

This component is responsible for calculating the odds ratio [42] between two binary variables. The component gets as input the data cube and the identifiers of the two variables that will be tested. Moreover an option for ignoring any missing data is provided.

Table 18: Odds Ratio Component

Name	Odds Ratio component
Description	The Odds Ratio component provides the tools to compute the odds ratio between two binary variables for identifying the association between them.
Responsibilities	This component is responsible for computing the odds ratio between two binary variables for identifying the association between them.
Constraints	Can only be applied on binary variables.
Composition	Node for applying odds ratio on the input data.
Inputs	An RDF data cube in CSV format, the identifiers of the two variables and whether missing data will be ignored.
Outputs	The association tested and the odds ratio.
Interactions/ Interfaces	This component interacts with the RDF data cube retrieval to get its input and with the components of the pre-processing steps as their output can be used as input for this component. The output of the component and its parameter settings can be stored in the database.

4.5.2.9.1 Class Diagram

The class diagram of the component is shown in Figure 46. To initialize the parameters of the OddsRatio class the method setParameters() is used. For performing the odds ratio test, the Datacube class is used to load the specified data cube and then the odds ratio is calculated based on the combinations of the values of the two variables. If missing variables are not ignored, then each

instance with a missing value is replaced by instances that contain all of the possible combinations for that value before calculating the odds ratio. The output and the parameters of the algorithm can be stored in the database.

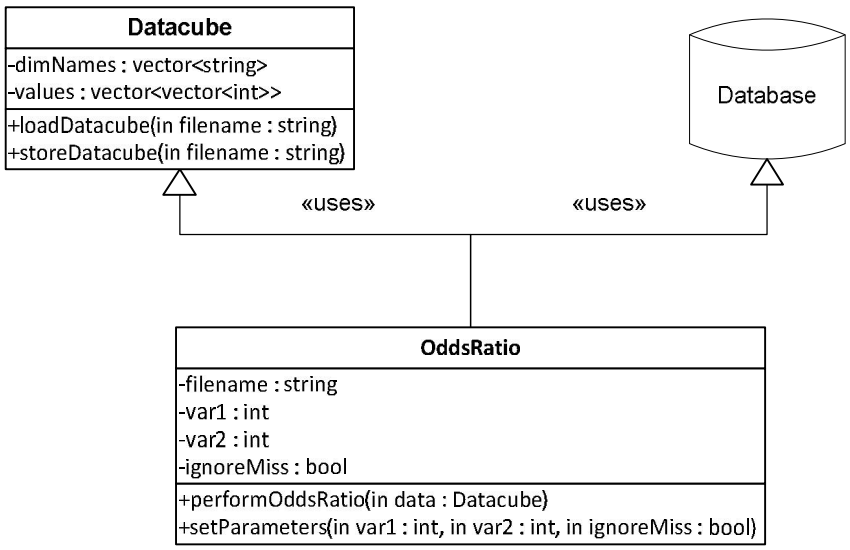

Figure 46: Odds ratio class diagram

4.5.2.9.2 User Interface Design

Figure 47 illustrates the user interface mock-up of the component. The user can select the input file and set the identifiers of the two variables. An option for ignoring instances with missing values is provided to the user. Once these values are added the test can be executed.

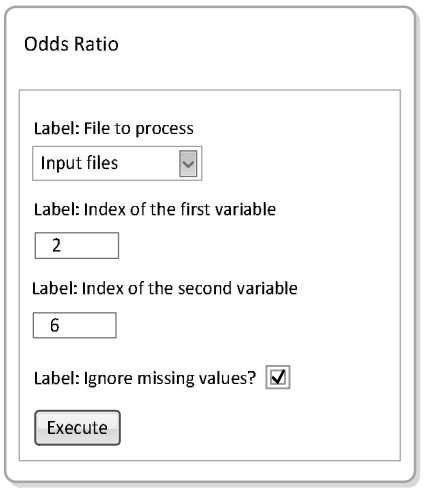

Figure 47: Odds ratio user interface mock-up

4.5.3 Data Mining

Rule-based data mining [6-20,25] techniques that are thought to be promising will be implemented in the Linked2Safety platform.

4.5.3.1 Module Design

The Data Mining module is composed of following algorithms:

- Decision trees: A decision tree algorithms takes as input data cubes and output a decision tree [9, 18-20];
- Random forest: A random forest algorithm [18-20] takes as input data cubes and outputs a forest of decision trees.
- Association rules mining: An association rules mining algorithm takes as input data cubes and outputs association rules [18-20].

It should be noted that each of the above categories is an abstraction of a particular data mining approach; a number of algorithms exist in each category.

4.5.3.2 Decision Trees

A decision tree algorithm is described in Table 19. The Decision trees algorithm take a data cube which can be output by either the Quality Control or the Feature Selection modules; then, the algorithm outputs a decision tree which is capable of detecting adverse events. Rules are extracted from the decision tree. The rules and the parameters setting of a decision tree algorithm are stored inthe database. The construct_decision_tree method of the DecisionTree class (Figure 48) takes a data cube in CSV format; returns a decision tree as a string object. The extract_rules method extracts rules from the decision tree. The save_output method stores the rules and the parameters setting of the decision tree in the database.

4.5.3.2.1 User Interface Design

Figure 49 illustrates the UI mock-up of Decision tree.

Table 19: Decision Tree Algorithm

Name	Decision Tree Algorithm
Description	The decision tree algorithm provides functions to build a decision tree from data. The decision tree is capable of detecting adverse events. Rules are extracted from the decision tree and stored in the database.
Responsibilities	The decision tree algorithm is responsible for creating a decision tree from a data cube which is capable of detecting adverse events.
Constraints	• The input data contains more than one subject. • The data set must contain a class variable.
Composition	Node for applying a decision tree algorithm to the input data.
Inputs	A data cube in CSV format.
Outputs	• A decision tree to detect adverse events • A set of rules extracted from the decision tree
Interactions/ Interfaces	This component interacts with the RDF data cube retrieval to get its input and with the components of the pre-processing steps as their output can be used as input for this component. The output of the component and its parameter settings can be stored in the database.

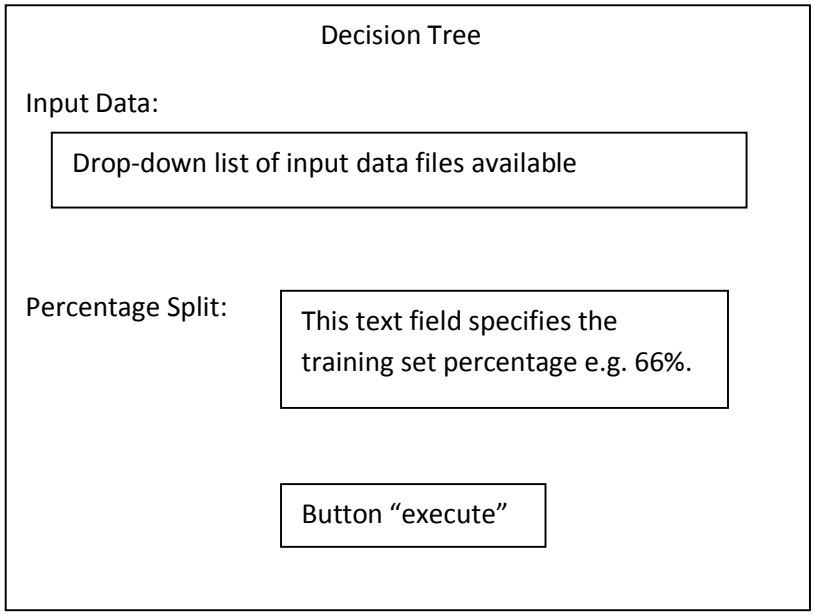

Figure 48: Input user interface of decision tree algorithm; user selects an input data file and specifies the training set percentage and clicks “Execute” button; then, the outputs are displayed in a new window (Figure 49).

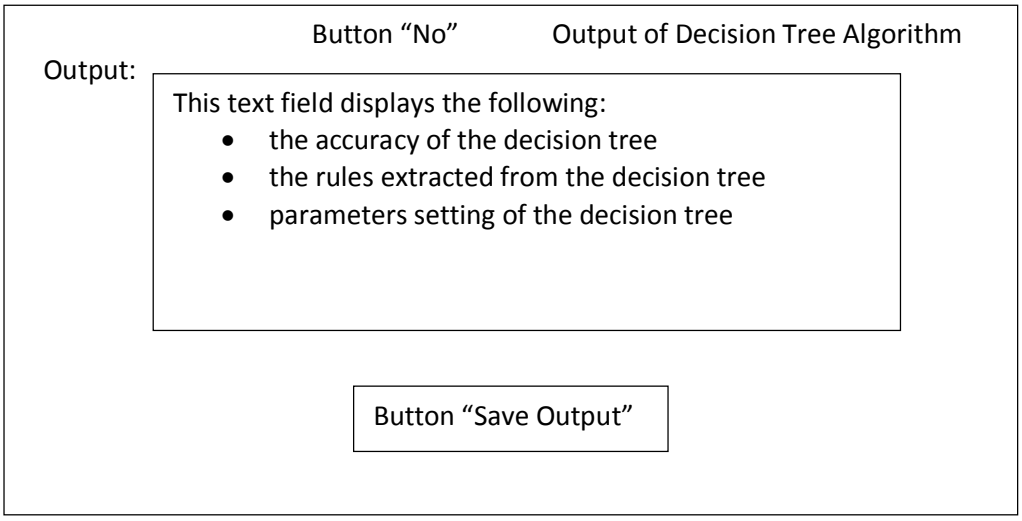

Figure 49: Output user interface of decision tree algorithm; clicking on the “Save Output” button stores the outputs into the database.

4.5.3.3 Random Forest

The random forest algorithm is described in Table 20. The random forest algorithm takes a data cube which can be output by either the Quality Control or the Feature Selection modules; then, the algorithm outputs a number of randomly-generated decision trees which are capable of detecting adverse events. Rules are extracted from the decision trees. The rules and the parameters setting of random forest are stored in the database. The construct_decision_tree method of the RandomForest class (Figure 50) takes a data cube in CSV format; returns a number of decision trees as a string object. The extract_rules method extracts rules from all the decision trees. The save_output method stores the rules and the parameters setting of the random forest in the database.

Table 20: Random Forest

Name	Random Forest Algorithm
Description	The random forest algorithm provides functions to build a number of decision trees from data. Each decision tree is capable of detecting adverse events. Rules are extracted from all the decision trees and stored into the database.
Responsibilities	The random forest algorithm is responsible of creating a number of decision trees from a data cube which are capable of detecting adverse events.
Constraints	• The input data contains more than one subject. • The input dataset must have a class variable.
Composition	Node for applying RFA i.e. constructing a number of decision trees randomly for a given dataset.
Inputs	A data cube in CSV format
Outputs	• A number of decision trees to detect adverse events • A set of rules extracted from the decision trees
Interactions/ Interfaces	This component interacts with the RDF data cube retrieval to get its input and with the components of the pre-processing steps as their output can be used as input for this component. The output of the component and its parameter settings can be stored in the database.

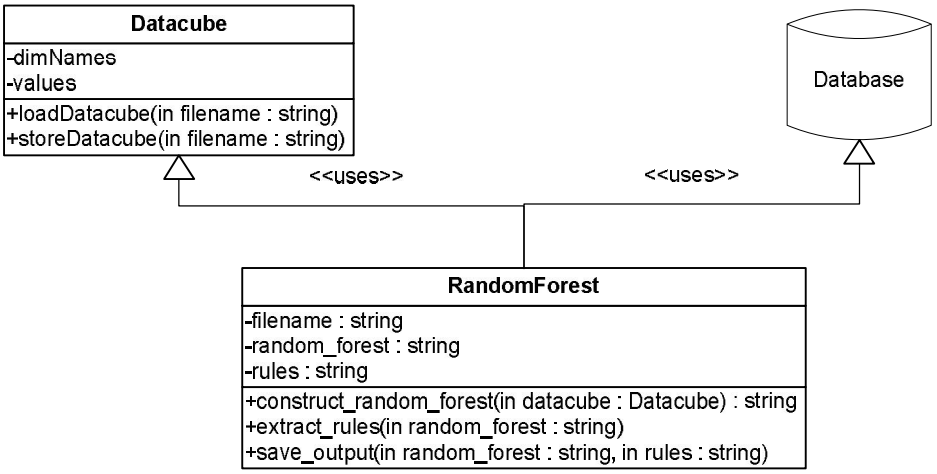

Figure 50: Class diagram of random forest algorithm

4.5.3.3.1 User Interface Design

Figure 51 illustrates the user interface mock-up of a random forest algorithm.

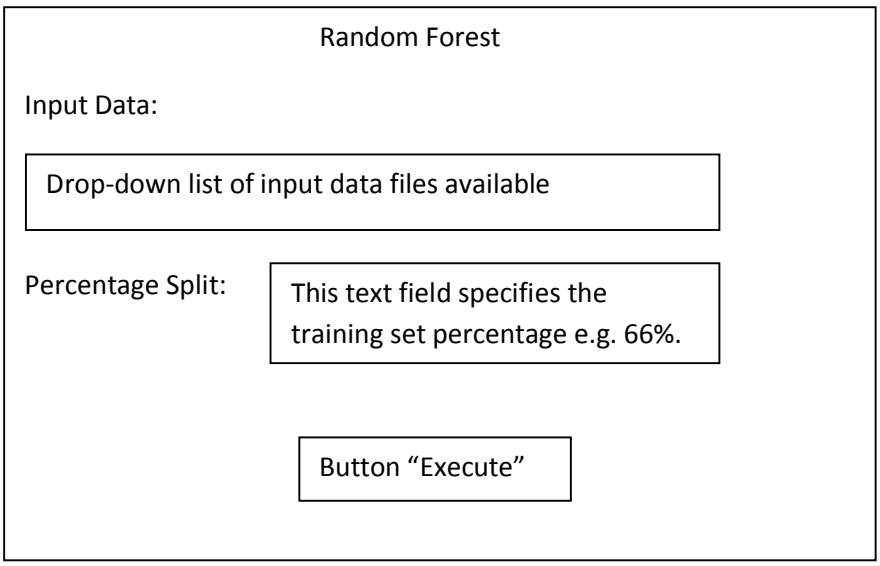

Figure 51: Input user interface of a random forest algorithm; user selects an input data file and specifies the training set percentage and clicks “Execute” button; then, the outputs are displayed in a new window (Figure 52).

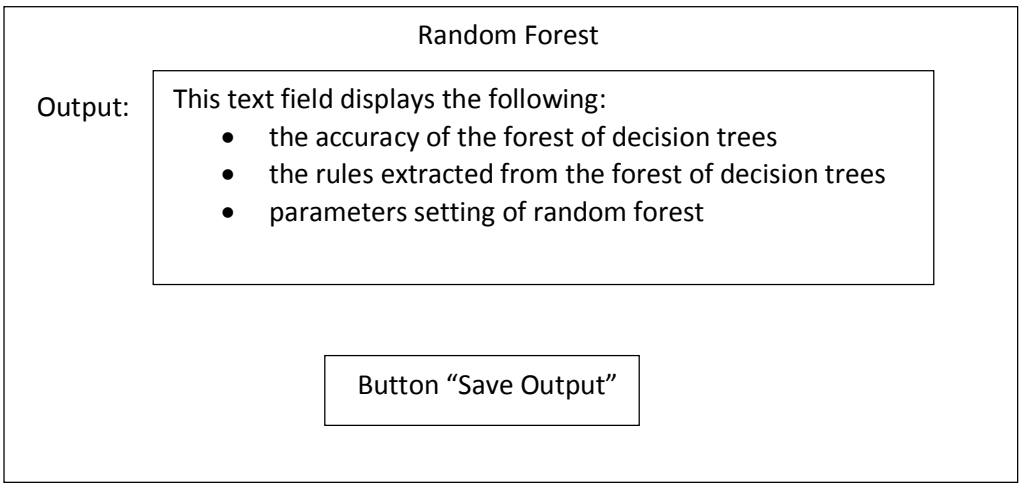

Figure 52: Output user interface of a random forest algorithm; clicking on the “Save Output” button stores the outputs in the database.

4.5.3.4 Association Rules

An association rules algorithm is described in Table 21.Table 21 An association rules algorithm takes a data cube which can be output by either the Quality Control or the Feature Selection modules; then, the algorithm outputs a number of association rules which are capable of detecting adverse events. The rules and the parameters setting of the association rule algorithm are stored in the database. The mine_rules method of the AssociationRules class (Figure 53) takes a data cube in CSV format; returns a number of rules as a string object. The save_output method stores the rules and the parameters setting of the association rules algorithm in the database.

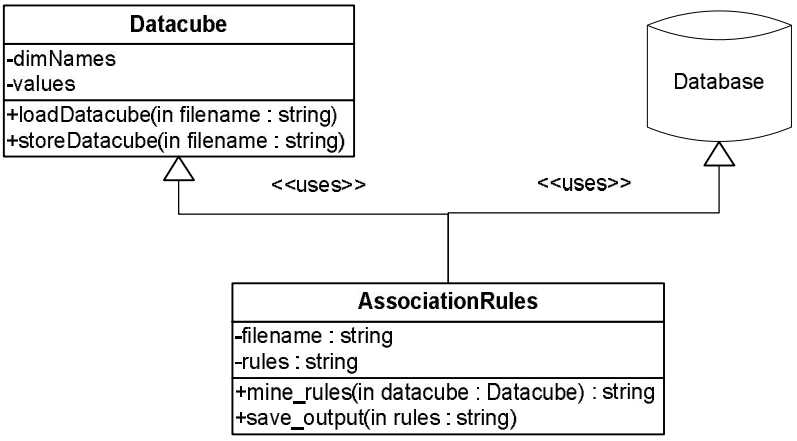

Figure 53: Class diagram of an association rules algorithm

Table 21: Association Rules Algorithm

Name	Association Rules Algorithm
Description	The association rules algorithm mines association rules from a data cube and outputs the rules to detect adverse events.
Responsibilities	The basic procedure of the association rules algorithm is to mine association rules from input data. The selection of the association rules is performed using rule quality metrics such as support, confidence and lift etc.
Constraints	The input data contains more than one subject.
Composition	A node for mining association rules from a given data cube using rule quality metrics.
Inputs	A data cube in CSV format.
Outputs	Association rules to detect adverse events.
Interactions/ Interfaces	This component interacts with the RDF data cube retrieval to get its input and with the components of the pre-processing steps as their output can be used as input for this component. The output of the component and its parameter settings can be stored in the database.

4.5.3.4.1 User Interface Design

Figure 54 illustrates the UI mock-up of an association rules algorithm.

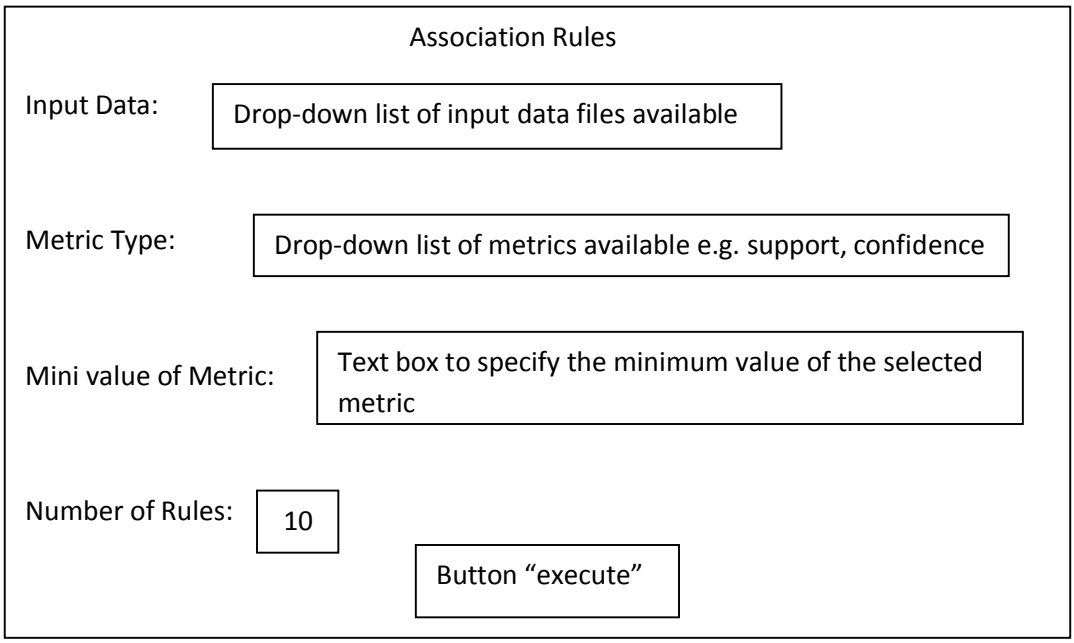

Figure 54: Input user interface of an association rules algorithm; the user selects an input data file, a metric type e.g. confidence, sets a minimum value of the selected metric type and sets the number of rules to output; then, the button “Execute” is clicked and the outputs are displayed in a new UI (Figure 55).

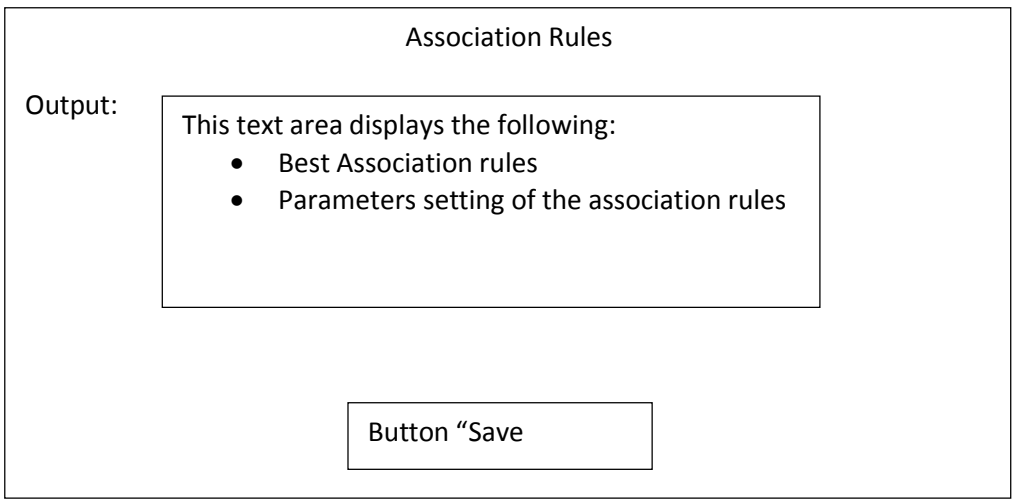

Figure 55: Output user interface of an association rules algorithm; clicking on “Save Outputs” button stores the outputs in the database.

4.6 PMML File Creation

4.6.1 XML and PMML

The Predictive Model Markup Language (PMML) [52] is a universally accepted, open standard and specification for describing data-mining models. It was written by the Data Mining group to enable different data-mining applications to easily share their respective models. PMML is a subset of XML [53], and as such, each model description instance is written in XML mark-up language and the PMML specification is written using the XML Schema language.

XML is designed to transport data between applications and for data storage and is described by the W3C as 'the most common tool for data transmissions between all sorts of applications'. Several XML-based technologies are available, for example to explore and manipulate document instances (Document Object Model (DOM)), transforming the document into HTML or XML (XSLT) and for querying XML documents (XML Query Language (XQuery)). In addition, as XML is a W3C recommendation, many 3rd party XML tools are available for displaying and editing document instances and many software technologies and browsers have built-in XML parsers and other functionalities.

It is clear therefore, that PMML will facilitate the storage, querying and visualisation of the models produced by the Linked2Safety platform both within the current project specifications and for future developments, including integration with new components both internal to Linked2Safety and external, if required.

4.6.2 Components of a PMML File

PMML follows an intuitive structure to describe a data mining model. A PMML file consists of the following components:

- **Header:** Header contains general information about the PMML document, such as copyright information for the model, its description, and information about the application used to generate the model such as name and version. It also contains an attribute for a timestamp which can be used to specify the date of model creation;
- **Data Dictionary:** Data dictionary contains definitions for all the possible fields used by the model. It is here that a field is defined as continuous, categorical, or ordinal (attribute optype). Depending on this definition, the appropriate value ranges are then defined as well as the data type (such as, string or double);

- **Data Transformation:** Data transformations allow for the mapping of user data into a more desirable form to be used by the mining model. PMML defines several kinds of simple data transformations:
  - Normalization: map values to numbers, the input can be continuous or discrete;
  - Discretization: map continuous values to discrete values;
  - Value mapping: map discrete values to discrete values;
  - Functions (custom and built-in): derive a value by applying a function to one or more parameters;
  - Aggregation: used to summarize or collect groups of values.
- **Model:** Model contains the definition of the data mining model such as association rules;
- **Mining Schema:** Mining schema lists all fields used in the model. This can be a subset of the fields as defined in the data dictionary. It contains specific information about each field, such as:
  - Name (attribute name): must refer to a field in the data dictionary;
  - Usage type (attribute usageType): defines the way a field is to be used in the model. Typical values are: active, predicted, and supplementary. Predicted fields are those whose values are predicted by the model;
  - Outlier Treatment (attribute outliers): defines the outlier treatment to be use. In PMML, outliers can be treated as missing values, as extreme values (based on the definition of high and low values for a particular field), or as is;
  - Missing Value Replacement Policy (attribute missingValueReplacement): if this attribute is specified then a missing value is automatically replaced by the given values.
  - Missing Value Treatment (attribute missingValueTreatment): indicates how the missing value replacement was derived (e.g. as value, mean or median).
- **Targets**: allow for post-processing of the predicted value in the format of scaling if the output of the model is continuous. Targets can also be used for classification tasks. In this case, the attribute prior Probability specifies a default probability for the corresponding target category. It is used if the prediction logic itself did not produce a result. This can happen, e.g., if an input value is missing and there is no other method for treating missing values;

- **Output**: this element can be used to name all the desired output fields expected from the model. These are features of the predicted field and so are typically the predicted value itself, the probability, cluster affinity (for clustering models), standard error, etc. PMML 4.1, the latest release of PMML, extended output to allow for generic post-processing of model outputs. In PMML 4.1, all the built-in and custom functions that were originally available for pre-processing only are now also available for post-processing.

An example PMML file is presented in Appendix 10.

4.6.3 Module Design

The PMML File creation module (Table 22) represents association rules in PMML format and stores the PMML files in the database. The create_PMML_file method of the PMMLCreator class (Figure 56Figure 56) retrieves rules from the database and creates a PMML file. The store_PMML_file method stores the PMML file in the database. Note this functionality may also be automatically invoked following analysis.

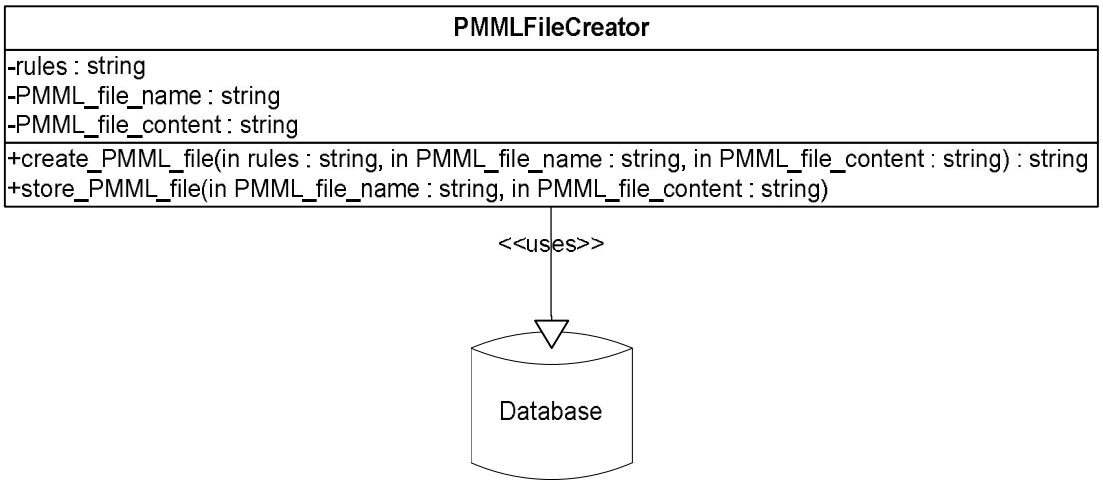

Figure 56: Class diagram of PMML File Creation

Table 22: PMML File Creation Module

Name	PMML File Creation
Description	Association rules are retrieved from the database and converted to PMML files which are stored into the database.
Responsibilities	This component is responsible for converting association rules to PMML format and storing the PMML file into the database
Constraints	There is at least one rule in the database.
Composition	• A node to retrieve association rules from the database. • A node to convert association rules to the PMML format.
Inputs	Association rules and their details such as support, confidence etc.
Outputs	A PMML file containing the rules.
Interactions/ Interfaces	This component interacts with the database to retrieve association rules from it and to insert PMML files into it.

4.6.4 User Interface Design

Figure 57 illustrates the user interface mock-up of PMML File Creation module.

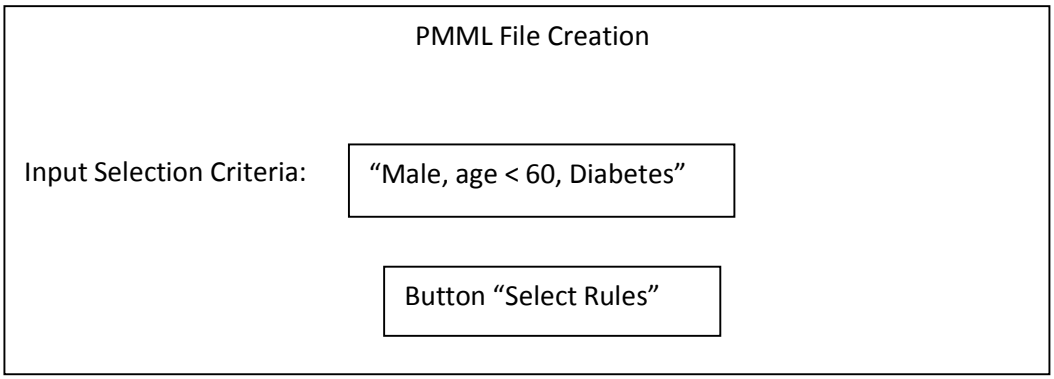

Figure 57: User interface mock-up of selecting rules for PMML file creation; the user enters the rule selection criteria and clicks on the button "Select Rules"; then, the rules matching the criteria are displayed on the UI in Figure 58.

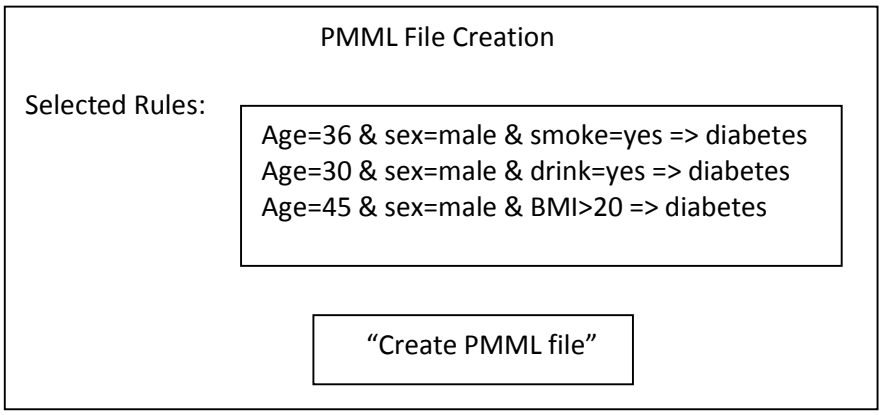

Figure 58: User interface mock-up of PMML File Creation; the user reads the displayed rules and deletes any rules not to be converted to PMML format; then clicking on the button "Create PMML File" creates a PMML file and stores it into the database.

4.7 Adverse Event Early Detection Mechanism

The adverse event early detection mechanism is an alerting tool, running in real time. Whenever data cubes are updated (for example, because a new patient is recruited or an existing one is diagnosed with a new condition or adverse response event to a drug), the specific mechanism matches the profile of the patient from the EHR with the knowledge that has been produced by the data mining algorithms or provided by the users. If the matching shows complications regarding adverse events, an alert will be provided. The alert will provide specific details related to the rule that defines the specific association between a patient group and an adverse event, as well as a detailed distribution of the patients on the drug and their association to the adverse effect.

4.7.1 Module Design

The Adverse event early detection mechanism (Figure 59Figure 59) consists of the database and the three classes: AlertNotificationMechanism, KnowledgeBase and PatientProfile. AlertNotificationMechanism is the main program which instantiates a KnowledgeBase object and a PatientProfile object. Then, AlertNotificationMechanism delegates the following operations to the objects:

- Subscribe to alert notification: after subscription to alert notification, each time the user logs into the adverse event early detection mechanism, it generates safety alerts for the user's patient profiles and presents it to the user;
- Unsubscribe to alert notification;
- Create patients profiles: creates patients profiles and stores them into the database;
- Get safety alerts: this generates safety alerts;
- Update patients profiles: this saves changes made to patient profiles and stores them into the database.

Table 23: Adverse Event Early Detection Mechanism

Name	Adverse Event Early Detection Mechanism
Description	This module runs in the background as an alerting tool whenever patient profiles are subscribed to the adverse event early detection mechanism to provide safety alerts on newly discovered potential adverse events related to specific medication or treatment options.
Responsibilities	To identify and report to specific expert users potentially newly identified adverse events associated with a subgroup of population.
Constraints	Rules and association rules must exist in the database.
Composition	• Node to create new patient profiles or updates existing ones. • Node to matches the new/updated patient profiles and raise an appropriate alert where relevant.
Inputs	Patient profiles, rules and associations stored in the database.
Outputs	Safety alerts of patient profiles.
Interactions/ Interfaces	This component interacts with the database to retrieve data mining rules and patient profiles to perform appropriate matching and raise relevant alerts.

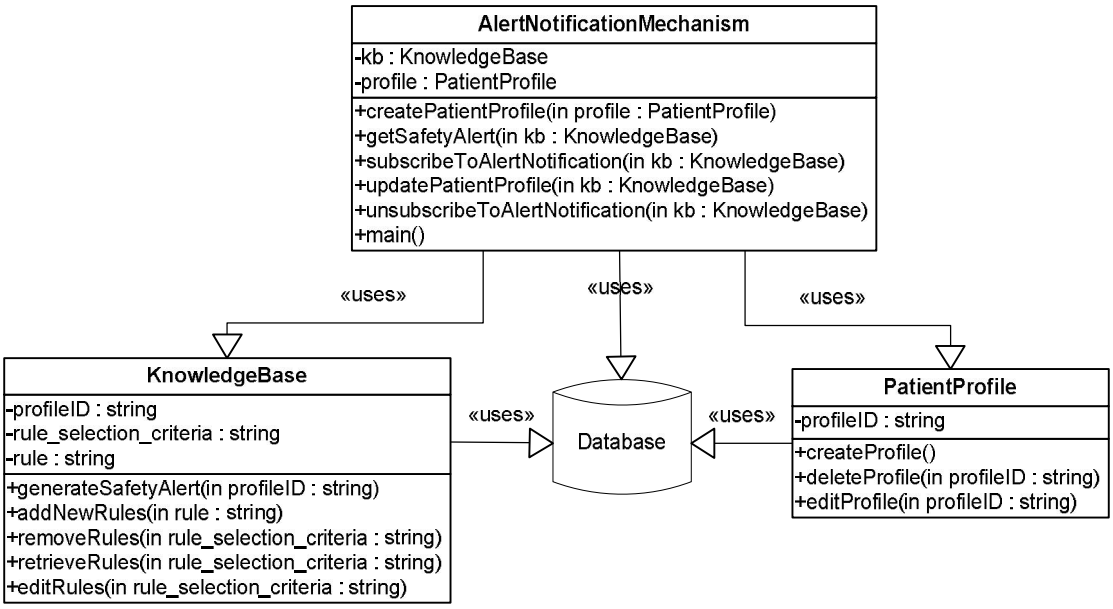

Figure 59: Class Diagram of Adverse Event Early Detection Mechanism

4.7.2 User Interface Design

The user interfaces are shown in figures 61-68.

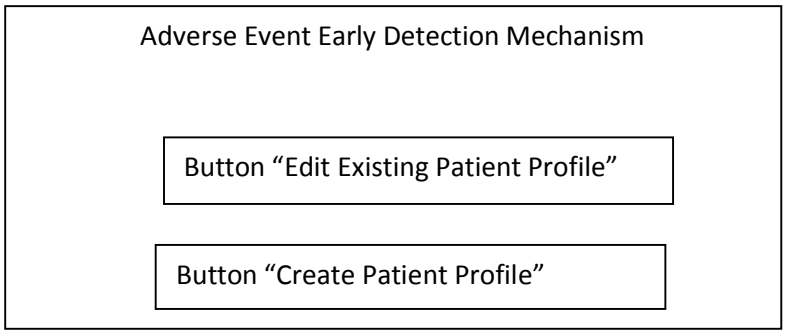

Figure 60: Adverse Event Early Detection Mechanism Interface: when the button “Create Patient Profile” is clicked, “Create Patient Profile” user interface pops up (Figure 61).

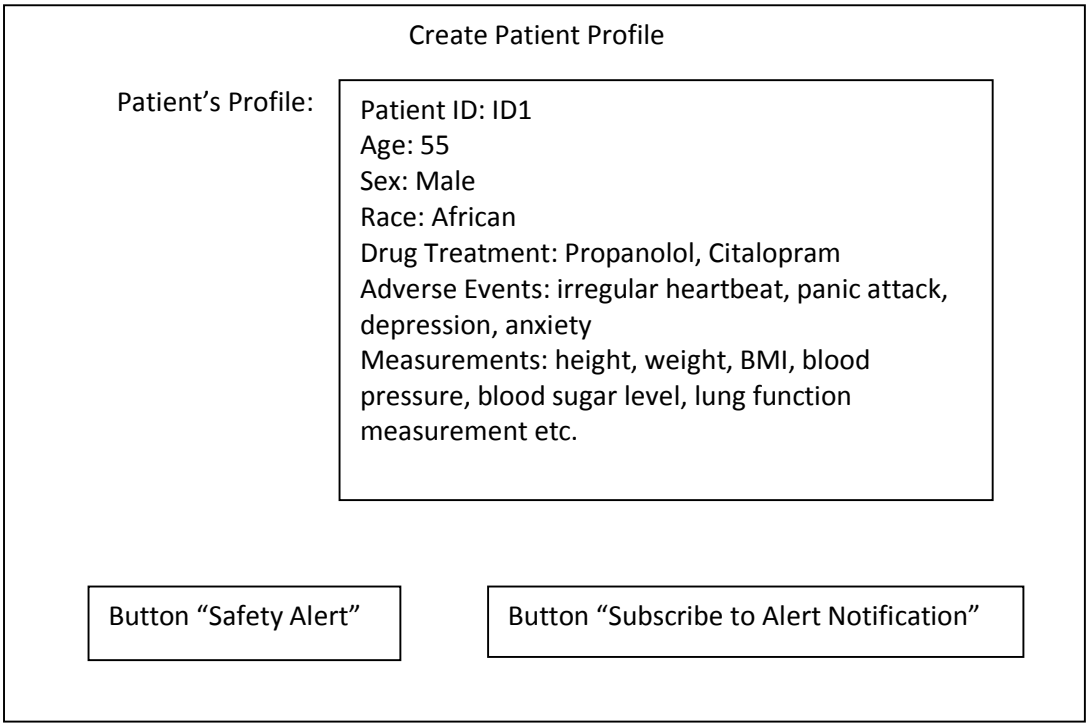

Figure 61: Create patient profile User Interface Mock-up: clicking on the button "Safety Alert" pops up the "Safety Alert of Patient Profile" (Figure 62) pops up to display the safety alert. Clicking on the "Subscribe to Alert Notification" button, the profile is subscribed to the alert notification mechanism, so that when the user logs into the Linked2Safety platform the safety alerts are displayed (Figure 65).

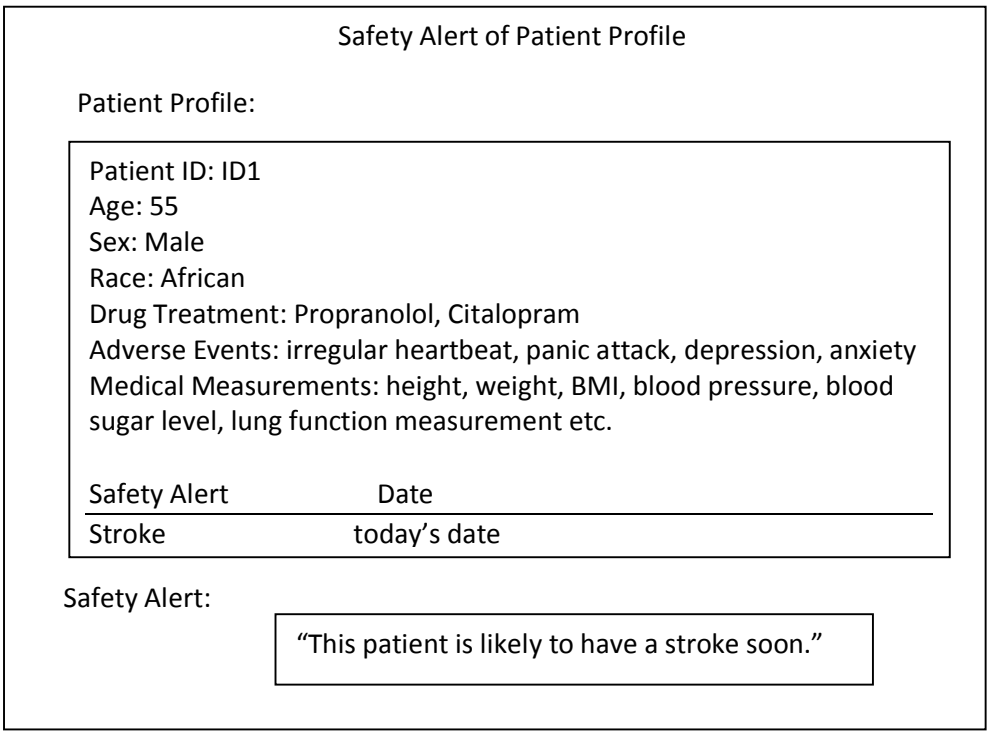

Figure 62: Safety alert of patient profile

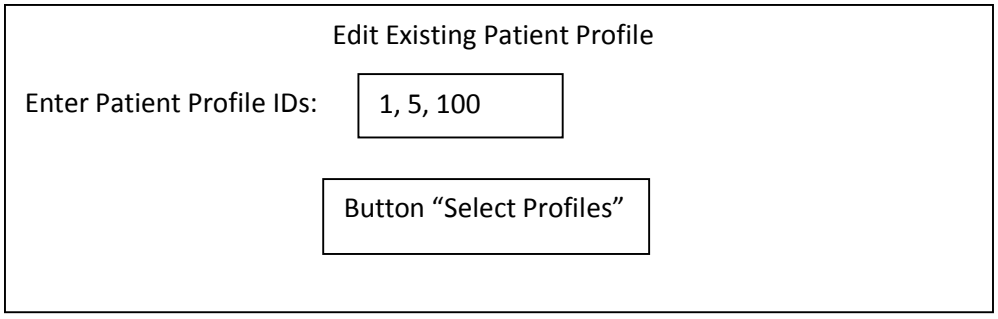

Figure 63: Edit Existing Patient Profile Interface; the user enters profile ID and press "Select"; then the Options window (Figure 64) pops up.

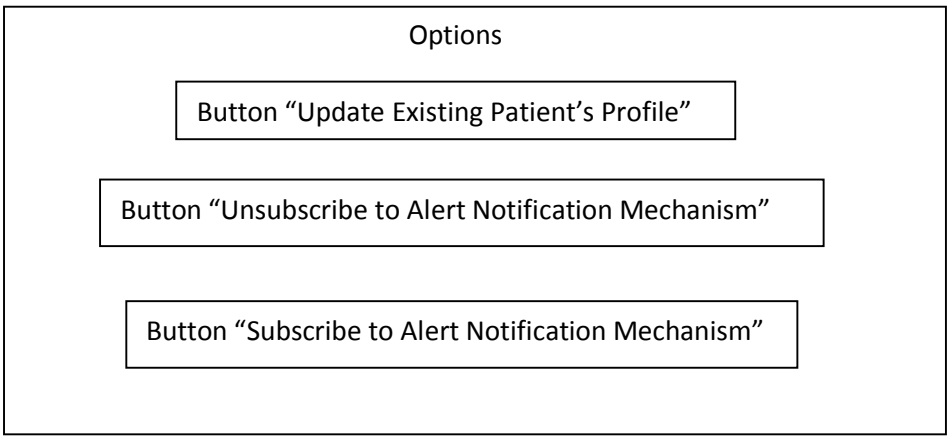

Figure 64: Options

Alert Notification

Profile ID	Adverse Events
ID1	Stroke
ID3	Diabetes
ID11	Heart Attack
…	…

Figure 65: Alert Notification

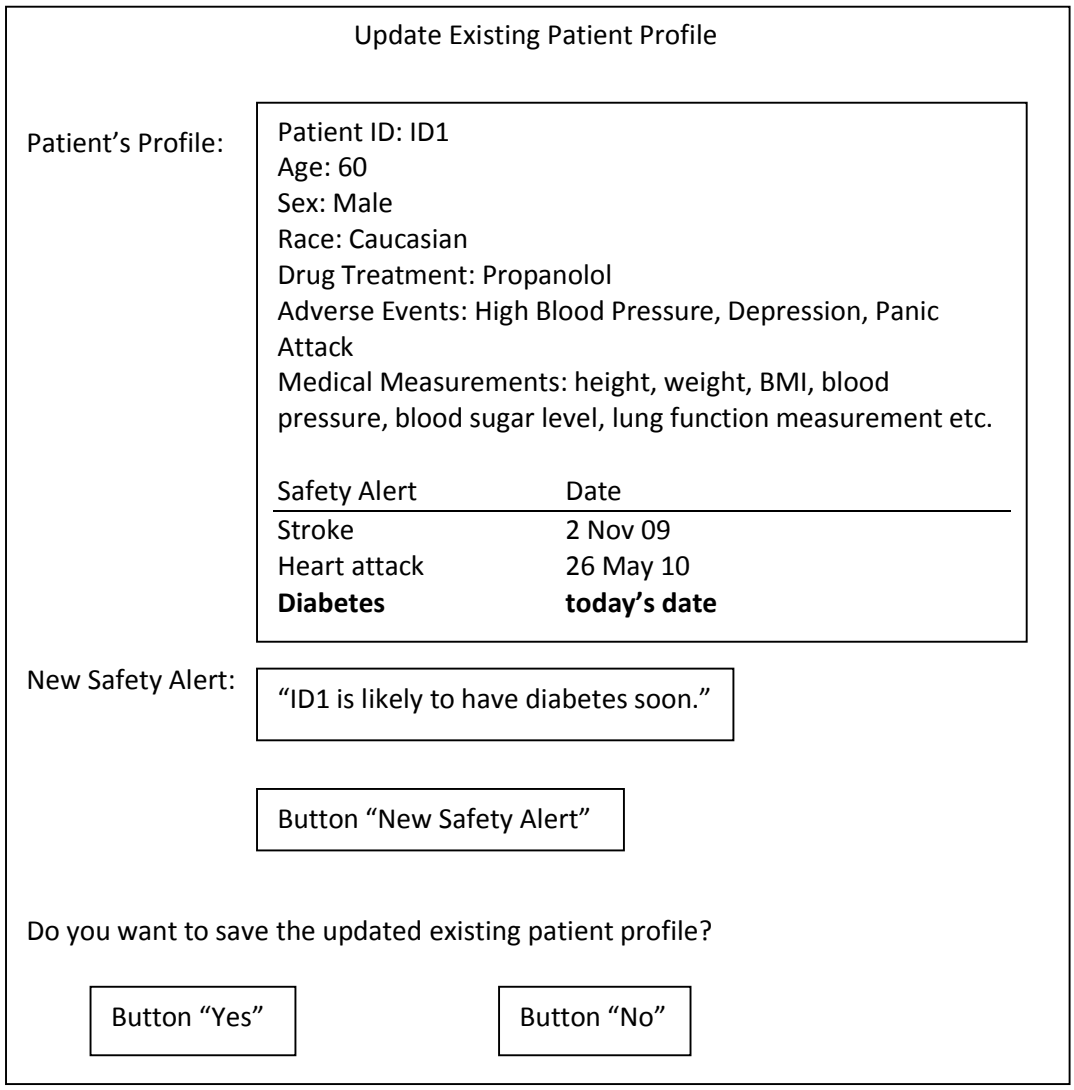

Figure 66: Update Existing Patient Profile User Interface Mock-up

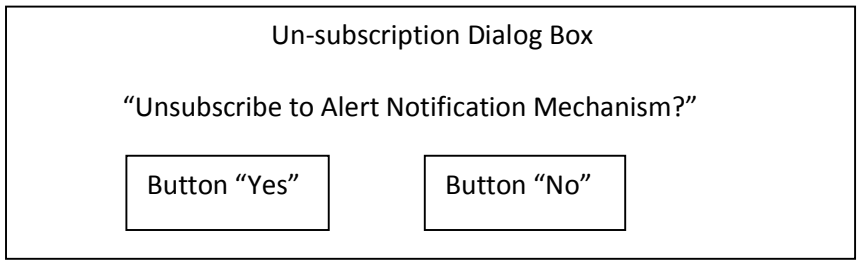

Figure 67: Un-subscription Dialog Box

5 Database Design

5.1 The purpose of the Database

A database is designed to store all the information related to results from the data analyses so that they can be used by the knowledge extraction and filtering and the safety alert notification mechanisms.

The user uses the Knowledge Extraction and Filtering Mechanism to remove any non-interesting rules and associations from the database based on experts' knowledge and the significance of the rules and the associations. The user can also remove from the database any results that should not be recorded because of violation of security requirements. Additionally, the user can insert rules and associations from the literature into the database.

The Adverse Event Early Detection Mechanism is capable of automatically generating safety alerts for patient profiles. The Adverse Event Early Detection Mechanism searches rules which match a patient profile. Then, a safety alert is created based on the outcomes of the matched rules and output to the user.

The modules of the Data Analysis Space are interfaced with the database (Figure 68Figure 68), so that the outputs of the modules can be inserted into the database and the Knowledge Extraction and Filtering, Adverse Event Early Detection Mechanisms modules and the Visualization component can use the information stored in the database.

5.2 Data contained in the database

Based on the objective, the database should contain the following data:

- **Users' details:** name, occupation, organization, role, subscription date, working area, country, email address;
- **Details of data analysis experiments:** during an experiment, the user applies one or more algorithms on one or more datasets and the results are output by the algorithms used. The details of experiments are stored in the database so that the experiments can be repeated later;
- **Results of data analysis experiments:** association rules, results of subjects count, results of hypothesis testing, results of permutation testing;

- **Association rules in PMML format;**
- **Safety alert subscription:** information related to safety alert subscription of patient profile.

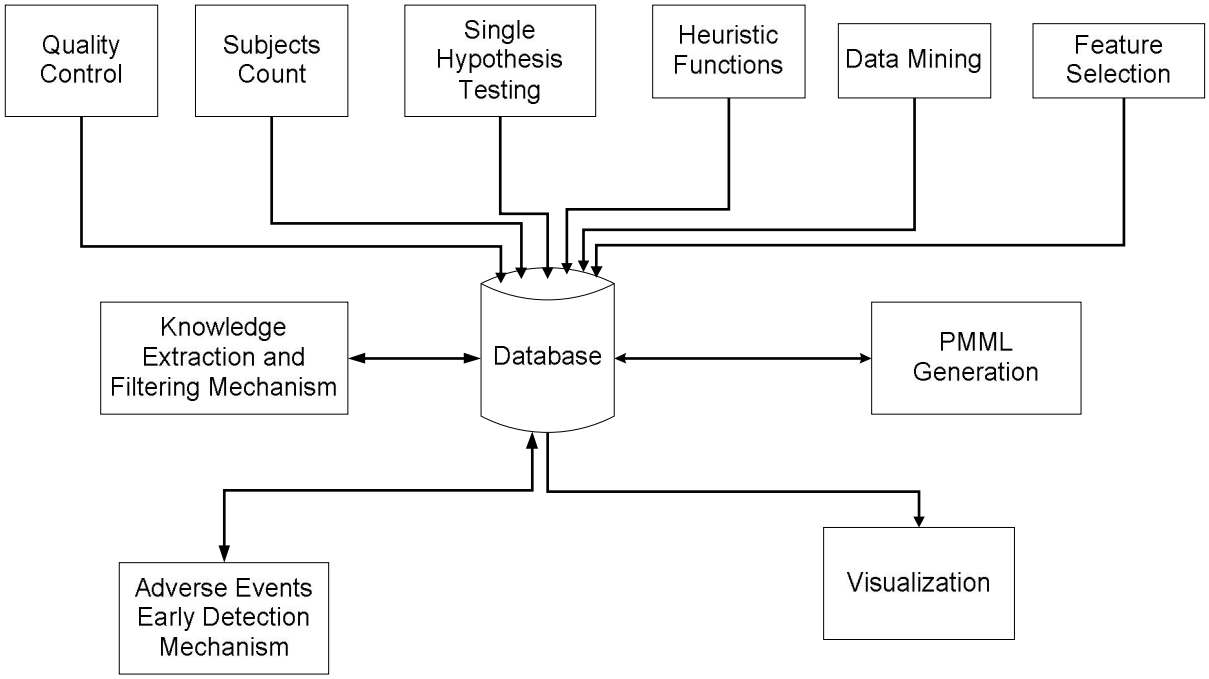

Figure 68: Database interaction with modules; arrows indicate storing/retrieving data to/from the database

Fields such as "Role", "Subscription Date", "Working Area", "Country" are related to the Access Policy Model, as described in "D1.4 Linked Medical Data Space". The Access Policy Model of the Linked Medical Data Space deals with the access control of the RDF data cubes that are requested though federated queries from the distributed RDF sources located in the closed rooms of the data providers. The policies that are applied to the RDF data cubes allow or deny access to expert users based on the data providers' preferences. Data providers use this model to represent and assign access policies to the RDF data cubes. Among other things, the access policies define the requirements that should be satisfied by the expert users in order to access the RDF data cubes. As expert users are authenticated, access to data is permitted or prohibited to him/her based on their profile attributes/characteristics (e.g. role, purpose, working area, etc.). In order to confirm if a user can access the data cubes, the respective user's attributes need to be validated with those defined in the access policies rules.

Moreover, when a user creates a data analysis experiment he/she should declare the purpose of the specific experiment. This information is essential to be kept in the database as it will determine whether a user is permitted to access the data cubes he/she requests. This constitutes a non-functional requirement derived from the review of D2.1 Legal and Ethical Requirements[1].

All the above information needs to be stored in the database for potential audit.

5.3 Entities and Relationships

Based on the data to be stored in the database, the following entities can be identified where the entity fields are listed after each entity:

- **User:** userID, name, occupation, organization, role, subscription_date, working_area, country, email_address;
- **Experiment:** userID, datasetID, experimentID, date_of_experiment, algorithm_name;
- **Dataset:** datasetID, SPARQL_query, time;
- **SubjectCount:** experimentID, datasetID, count_of_subjects, SPARQL_query, other_information;
- **DataMiningAlgorithm:** experimentID, algorithm_name, algorithmParamsSetting, datasetID, results;
- **QualityControl**: experimentID, algorithm_name, algorithmParamsSetting, datasetID, results;
- **FeatureSelectionAlgorithm:** experimentID, algorithm_name,

  algorithmParamsSetting, datasetID, reducedDataset;

[1] Deliverable 2.1 p. 76"Particular conditions for the Expert user and his registering to the Linked2Safety platform through which he/she will be able to access the data will have to be developed. As far as data processing is concerned, even though the data will be already in anonymous form, Linked2Safety will still have to make sure that the processing is in line with the original patient's consent. The patient could for instance provide consent only for particular studies related to cardiovascular diseases. In such a case, Linked2Safety would need to make sure that the Expert user will use data only for studies which are in line with the consent. This might be done by concluding End user agreement through registration process or by accepting specific rules that regard specific data from the specific data provider."

- **StatisticalAlgorithm:** experimentID, algorithm_name, algorithmParamsSetting, datasetID, results;
- **AssociationRule:** RuleID, Antecedent, Consequent, Support, Confidence, RRRatio, Lift;
- **Association:** AssociationID, Association;
- **Pattern:** PatternID, Pattern;
- **PMMLFile:** PMMLFile_name, dataset_name, algorithm_name, algorithmParamsSetting, PMMLFileContent;
- **AlertNotificationSubscription:** profileID, PatientProfile, AdverseEvents, dateOfAlert.

The description of the fields is presented in the following sections. During implementation of the database, the entities may be modified so that some of the fields are deleted from the entities or/and new fields are added to the entities.

5.3.1 User Entity

Records of the User entity represent users of the Linked2Safety platform. The fields of the User entity are described in Table 24.

Table 24: User Entity

Field	***Description***
userID	User Identification Number
Name	User's name
Occupation	User's occupation
Organization	User's organization
Email_address	User's email address
Role	User's role (the user can log-in the system as an expert user being either Clinical Researcher or Clinical Trial Coordinator)
Subscription_date	The date when the user has subscribed in the system
Working_area	User's working area
Country	User's country

The following is an example record of User:

userID: 1
Name: 'Tom'
Occupation: 'clinical expert'
Organization: 'Drug Company'
Role: 'Expert User'
Subscription date: '21-11-2012'
Working area: 'Oncology'
Country: 'Greece'
Email: 'tom@DrugCompany.com'

5.3.2 Dataset Entity

Records of the Dataset entity represent datasets used by the users. The fields of the Dataset entity are described in Table 25.

Table 25: Dataset Entity

Field	Description
datasetID	Dataset identification number
SPARQL_query	The SPARQL query to retrieve the dataset
Time	time the SPARQL query was run

The following is an example record of Dataset:

datasetID: 1
SPARQL_query: ' SELECT ?instance ?positives
FROM l2s-data:clinical-trials
WHERE {
?instance a qb:Observation .
?instance l2s-prop:drug ?drug .
?drug rdfs:label "Insulin" .
?instance l2s-prop:bmi25 "1"^^xsd:int .
?instance l2s-prop:dyslipidemia "1"^^xsd:int .
?instance l2s-prop:positives ?positives .
}'
Time: 2012-12-26 05:10:06

5.3.3 Experiment Entity

Records of the Experiment entity represent data analysis experiments performed by the users. The fields of Experiment entity are described in Table 26.

Table 26: Experiment Entity

Field	***Description***
userID	User Identification Number
datasetID	Dataset Identification Number
experimentID	Experiment Identification Number
Workflow	the sequence of the algorithms applied
time_of_experiment	time of doing the experiment
Purpose	Purpose for executing the specific experiment

The following is an example record of Experiment:

userID: 100
datasetID: 99
experimentID: 5
workflow: 'input data →rough set feature selection →Apriori → output'
time_of_experiment: 10:12:05 20 Nov 2012
purpose: 'studies for cardiovascular diseases'

5.3.4 DataMiningAlgorithm Entity

Records of the DataMiningAlgorithm entity represent data mining algorithms performed by the users. The fields of Experiment entity are described in Table 27.

Table 27: DataMiningAlgorithm Entity

Field	**Description**
Algorithm_name	Name of algorithm
algorithmParamsSetting	Parameters Setting of algorithm in the following format: <param1>=<value1>,…,<paramK=valueK>
datasetID	Dataset identification number
experimentID	Experiment identification number

The following is an example record of DataMiningAlgorithm:

Algorithm_name: 'Apriori'
AlgorithmParamsSetting: 'number of rules=10, rule quality measure=confidence'
datasetID: 1
experimentID: 5

5.3.5 QualityControlAlgorithm Entity

Records of QualityControlAlgorithm entity represent Raw Data Quality Control algorithms and Information Loss Quality Control algorithms performed by the users. The fields of QualityControlAlgorithm are described in Table 28.

Table 28: QualityControlAlgorithm Entity

Field	Description
Algorithm_name	Name of algorithm
algorithmParamsSetting	Parameters Setting of algorithm
datasetID	Dataset identification number
experimentID	Experiment Identification number

5.3.6 SubjectCount Entity

Records of the SubjectCount entity represent counts of the subjects who are selected by the users. The fields of the entity are described in Table 29.

Table 29: SubjectCount Entity

Field	Description
experimentID	Experiment Identification number
datasetID	Dataset identification number
count_of_subjects	Count of subjects
SPARQL_query	SPARQL query to count subjects
Other_information	Other information related to subject count e.g. information related to clinical sites

5.3.7 FeatureSelectionAlgorithm Entity

Records of the FeatureSelectionAlgorithm entity represent feature selection algorithms performed by the users. The fields of the entity are described in Table 30.

Table 30: FeatureSelectionAlgorithm

Field	Description
Algorithm_name	Name of algorithm
algorithmParamsSetting	Parameters Setting of algorithm
datasetID	Dataset identification number
experimentID	Experiment Identification number

The following is an example record of the FeatureSelectionAlgorithm entity:

Algorithm_name: 'information gain feature selection'
algorithmParamsSetting: 'number of features=10'
datasetID: 11
experimentID: 2

5.3.8 StatisticalAlgorithm Entity

Records of the StatisticalAlgorithm entity represent statistical hypothesis testing algorithms which are performed by the users. The fields of the StatisticalAlgorithm entity are described in Table 31Table 31.

Table 31: StatisticalAlgorithm Entity

Field	Description
Algorithm_name	Name of algorithm
algorithmParamsSetting	Parameters Setting of algorithm
datasetID	Dataset identification number
experimentID	Experiment Identification number

The following is an example record of the StatisticalAlgorithm entity:

Algorithm_name: 'Fisher Exact Test'
algorithmParamsSetting: 'target_variable =diabetes'
datasetID: 25

experimentID: 8

5.3.9 AssociationRule Entity

A record of the AssociationRule entity represents an association rule. The fields of the AssociationRule entity are described in Table 32**Table 32**.

Table 32: AssociationRule Entity

Field	Description
RuleID	Rule identification Number
Antecedent	condition of rule
Consequent	outcome of rule
Support	Support of rule
Confidence	Confidence of rule
Lift	Lift of rule
RRRatio	Relative Report Ratio of rule

The following is an example record of the AssociationRule entity:

RuleID: 1
Antecedent: sex='m' & smoke='yes' & drug='Evista'
Consequent: AE='respiratory failure' & outcome='death'
Support: 900
Confidence: 0.9
Lift: 0.9
RRRatio: 0.8

5.3.10 Association Entity

A record of the Association entity represents an association output by Statistical Hypothesis Testing algorithms. The fields of the Association entity are described in Table 33.

Table 33: Association Entity

Field	Description
AssocatiationID	Association Identification Number
Association	An association e.g. Height is associated with gender

pValue	The p-value of the association
oddsRatio	The odds ratio value

The following is an example record of the Association entity:

AssociationID: 1
Association: 'Height & Weight => gender'
pValue: '0.0005'
oddsRatio: ''

5.3.11 Pattern Entity

A record of the Pattern entity represents a pattern output by Statistical Hypothesis Testing algorithms. The fields of the Pattern entity are described in Table 34Table 34.

Table 34: Pattern Entity

Field	Description
PatternID	Pattern Identification Number
Pattern	A pattern e.g. Height over 1.70m ->male

The following is an example record of the Pattern entity:

PatternID: 1
Pattern: 'Height over 1.70m ->male'

5.3.12 PMMLFile Entity

Records of the PMMLFile entity represent PMML association rules files which are output by the PMML File Generation module. The fields of the PMMLFile entity are described in Table 35.

Table 35: PMMLFile Entity

Field	Description
PMMLFile_name	Name of PMML file
PMMLFileContent	Content of PMML file
Algorithm_name	the algorithm which outputs the association rules
algorithmParamsSetting	Parameters setting of the algorithm

The following is an example record of the PMMLFile entity. Value of the PMMLFileContent field is presented in Appendix 10.1. PMML file format is described in Section 4.6.

PMMLFile_name: 'PMMLfile1'
Algorithm_name: 'Apriori'
algorithmParamsSetting: 'number of rules=10,rule quality measure=confidence'

5.3.13 SafetyAlertNotificationSubscription Entity

Records of the SafetyAlertNotificationSubscription entity represent safety alert notification subscription. The fields of the SafetyAlertNotificationSubscription entity are described in Table 36Table 36.

Table 36: SafetyAlertNotificationSubscription Entity

Field	Description
profileID	Profile identification number
timeofAlert	Time when the safety alert (adverse events) is generated
patientProfile (patient profile includes information on demographics, drugs being taken, medical measurements e.g. BMI, blood pressure, height, weight etc.)	Patient profile is in following format: <variable1>=<value1>;…;<variable>=<valueK> e.g. weight=90kg; if a variable takes numerous values, they are separated by commas e.g. adverse events='vomiting (12 Oct 2012), respiratory depression (5 Aug 2010)'.
AdverseEvents	Predicted adverse event(s) separated by commas.
HealthExpertID	Identification number of the expert who subscribed this profile to the alert notification mechanism.
Subscription	Subscription (Yes/No) to safety alert notification.

The following is an example record of the SafetyAlertNotificationSubscription entity:

profileID : 101
timeOfAlert: 14:10:05 on 20 Nov 2012
patientProfile: 'age=60;sex=male;race=Caucasian;medication_being_taken=citapol(1 Oct 2012);medication_history=ibuprofen(1 May 2012),antibiotics(5 June 2012);BMI=2;

height=1.78m;weight=90kg;heart rate=150;BMI=2;adverse_events_history=vomiting(12 Oct 2012), respiratory depression(5 Aug 2010);outcomes_history=hospitalization(13 Oct 2012),life-threatening (8 Aug 2010)'
AdverseEvents: 'heart attack, hypertension, diabetes'
HeathExpertID: 8
Subscription: 'Yes'

The relationships between the entities are presented in the following sections.

5.3.14 User-Experiment Relationship Entity

The userID field is the primary key of User entity. The experimentID field is the primary key of Experiment entity. There is a one-to-many relationship between User and Experiment entities because a user can perform numerous experiments. Therefore, userID is defined as a foreign key in the Experiment entity to link it with User (Figure 69).

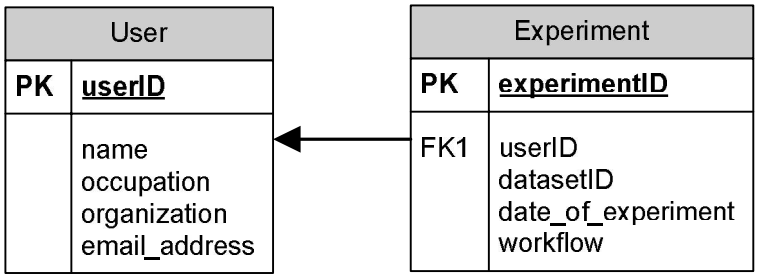

Figure 69: User-Experiment Relationship

5.3.15 Experiment-QualityControlAlgorithm Relationship

The combination of datasetID, algorithmName and algorithmParamsSetting fields can be used as a primary key of QualityControlAlgorithm entity because the combination identifies different application of Quality Control algorithms. There is a one-to-many relationship between Experiment and QualityControlAlgorithm entities because an experiment can run numerous Quality Control algorithms. Therefore, experimentID is defined as a foreign key in the QualityControlAlgorithm entity to link it with Experiment (Figure 70).

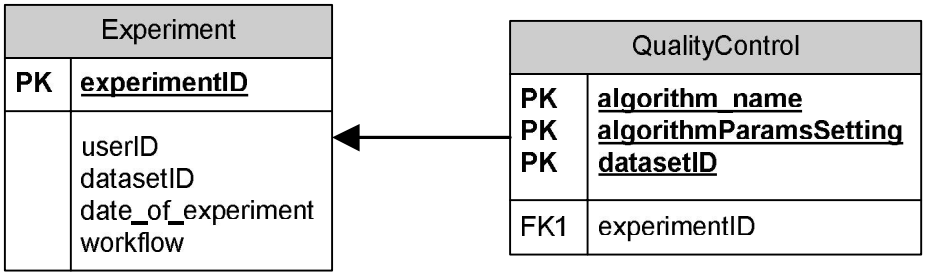

Figure 70: Experiment-QualityControl Relationship

5.3.16 Experiment-SubjectCount Relationship

The combination of experimentID and datasetID can be used as a primary key of SubjectCount entity because the combination identifies different subject count analyses. There is a one-to-many relationship between Experiment and SubjectCount entities because an experiment can perform numerous counts of subjects. Therefore, experimentID is defined as a foreign key in the SubjectCount entity to link it with Experiment (Figure 71).

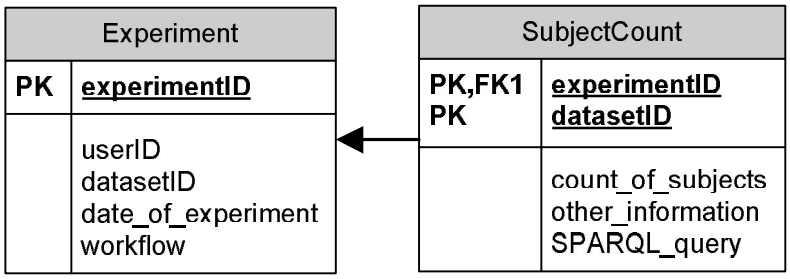

Figure 71: Experiment-SubjectCount Relationship

5.3.17 Experiment-DataMiningAlgorithm Relationship

The combination of datasetID, algorithmName and algorithmParamsSetting fields can be used as a primary key of DataMiningAlgorithm entity because the combination identifies different uses of the same data mining algorithm on different datasets. There is a one-to-many relationship between Experiment and DataMiningAlgorithm entities because an experiment can run numerous data mining algorithms. Therefore, experimentID is defined as a foreign key in the DataMiningAlgorithm entity to link it with Experiment (Figure 72).

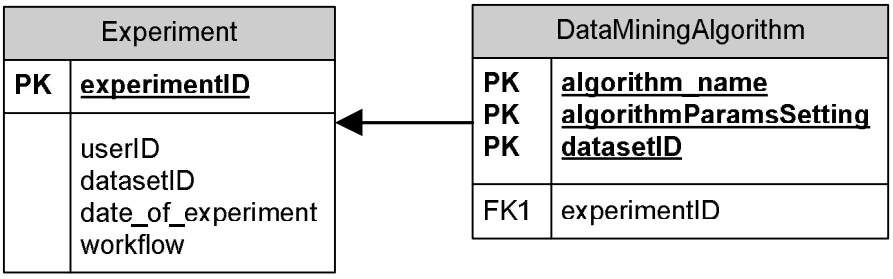

Figure 72: Experiment-DataMiningAlgorithm Relationship

5.3.18 Experiment-FeatureSelectionAlgorithm Relationship

The combination of datasetID, algorithmName and algorithmParamsSetting fields can be used as a primary key of FeatureSelectionAlgorithm entity. There is a one-to-many relationship between Experiment and FeatureSelectionAlgorithm entities because numerous feature selection algorithms can be used in an experiment. Therefore, experimentID is defined as a foreign key in the FeatureSelectionAlgorithm entity to link it with Experiment (Figure 73).

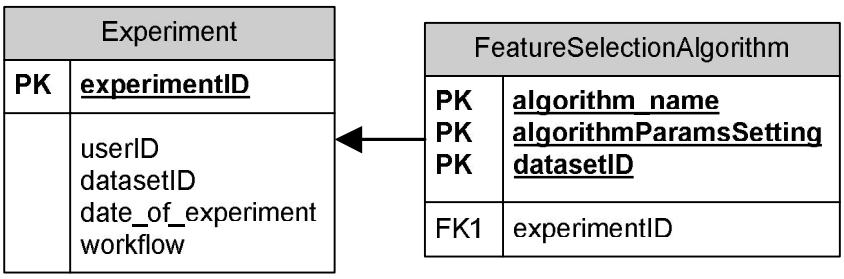

Figure 73: Experiment-FeatureSelectionAlgorithm Relationship

5.3.19 DataMiningAlgorithm-AssociationRule Relationship

The ruleID field is the primary key of the AssociationRule entity. There is a one-to-many relationship between DataMiningAlgorithm and AssociationRule entities because a data mining algorithm outputs numerous rules. Therefore, the combination of algorithm_name and algorithmParamsSetting fields is defined as a foreign key in the AssociationRule entity to link it with DataMiningAlgorithm (Figure 74).

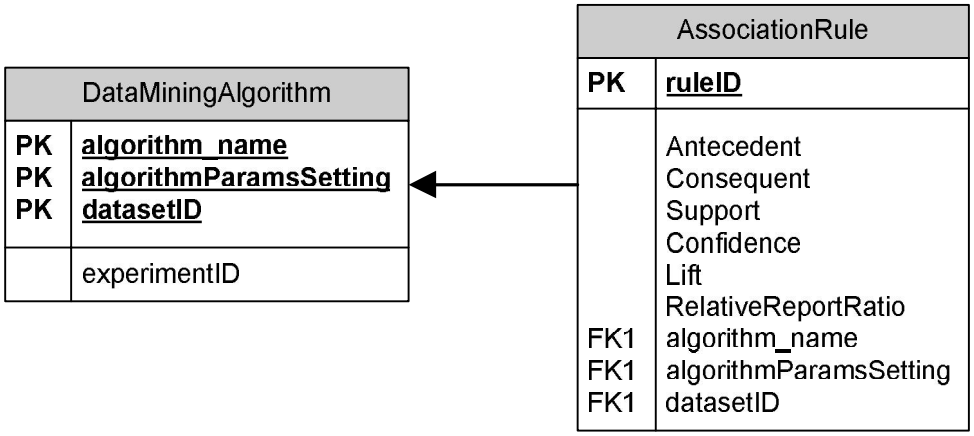

Figure 74: DataMiningAlgorithm-AssociationRule Relationship

5.3.20 StatisticalAlgorithm-Association Relationship

The AssociationID field is the primary key of the Association entity. There is a one-to-many relationship between StatisticalAlgorithm and Association entities because a statistical hypothesis testing algorithm outputs one or numerous associations. Therefore, the combination of algorithm_name and algorithmParamsSetting fields is defined as a foreign key in the Association entity to link it with StatisticalAlgorithm (Figure 75).

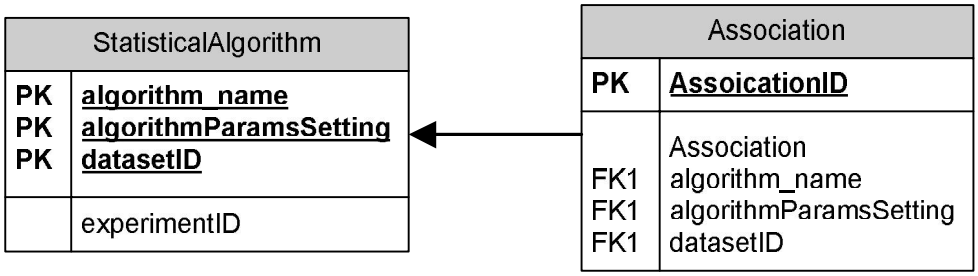

Figure 75: StatisticalAlgoritm-Associatioon Relationship

5.3.21 Experiment-StatisticalAlgorithm Relationship

The combination of algorithmName and algorithmParamsSetting fields can be used as a primary key of StatisticalAlgorithm entity. There is a one-to-many relationship between Experiment and

StatisticalAlgorithm entities because an experiment can run numerous statistical algorithms. Therefore, experimentID is defined as a foreign key in the StatisticalAlgorithm entity to link it with Experiment (Figure 76).

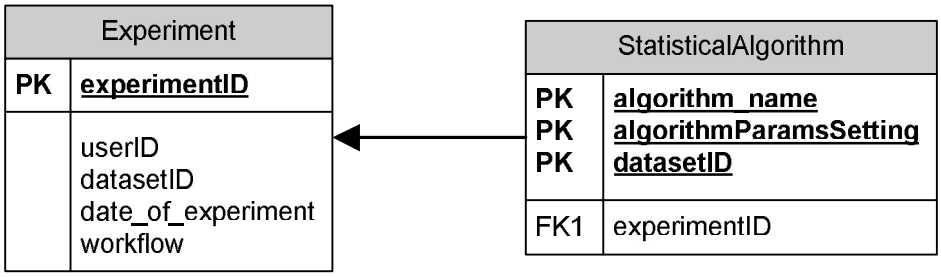

Figure 76: Experiment-StatisticalAlgorithm Relationship

5.3.22 Experiment-Dataset Relationship

The datasetID field is the primary key of the Dataset entity. There is a one-to-many relationship between Experiment and Dataset entities because numerous datasets can be used in an experiment. Therefore, experimentID is defined as a foreign key in the Dataset entity to link it with Experiment (Figure 77).

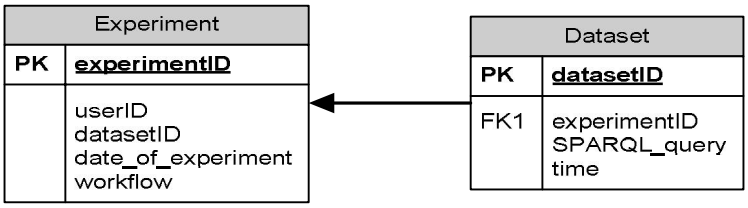

Figure 77: Experiment-Dataset Relationship

5.3.23 Dataset-PMMLFile Relationship

The PMMLFileName is the primary key of the PMMLFile entity. There is a one-to-many relationship between Dataset and PMMLFile because numerous PMML files can be created for a dataset. Therefore, datasetID is defined as a foreign key in the PMMLFile entity to link it with Dataset (Figure 78).

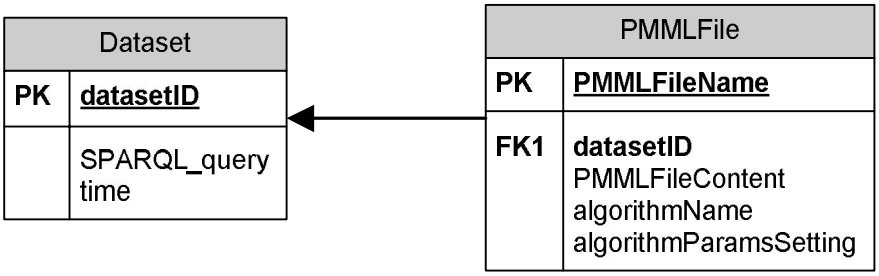

Figure 78: Dataset-PMMLFile Relationship

5.3.24 AssociationRule-SafetyAlertNotificationSubscription Relationship

The combination of ProfileID and date_of_alert fields is the primiary key of the SafetyAlertNotificationSubscription entity. There is a many-to-one relationship between AssociationRule and SafetyAlertNotificationSubscription because given a patient's profile, one or more rules could match the profile; if one rule matches the profile, the safety alert is the output of the rule; if numerous rules match the profile, the safety alert consists of the outputs of the matched rules. This is illustrated by the following example:

patientProfile: 'age=60;sex=male;race=Caucasian;medication=ibuprofen,antibiotics;BMI=2; height=1.78m;weight=90kg;heart rate=150;BMI=2;adverse events=vomiting(12 Oct 2012), respiratory depression(5 Aug 2010);patient profile outcomes=hospitalization(13 Oct 2012),life-threatening (8 Aug 2010)'

Matched Rules: 'sex=male & medication=ibuprofen => AE=heart attack
Age > 50 & medication=antibiotics => AE=hypertention
Sex=male & height > 1.70m & weight > 88kg => AE=diabetes'

Safety Alert: 'heart attack, hypertension, diabetes'

The combination of time_of_alert and ProfileID is defined as a foreign key in the AssociationRule entity to link it with SafetyAlertNotificationSubscription (Figure 79). The SafetyAlertNotificationSubscription table is updated by the Adverse Event Early Detection Mechanism.

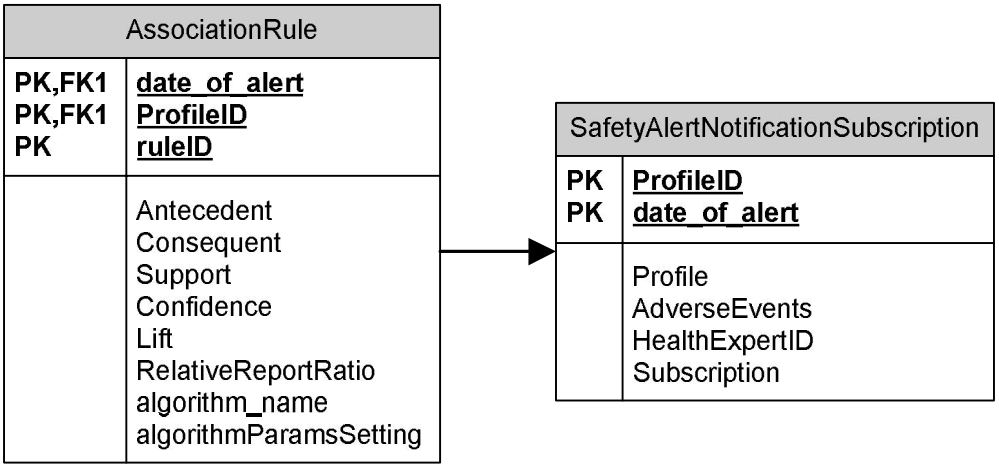

Figure 79: AssociationRule-SafetyAlertNotificationSubscription Relationship

The Entity Relationship (ER) schema of the database is illustrated in Figure 80.

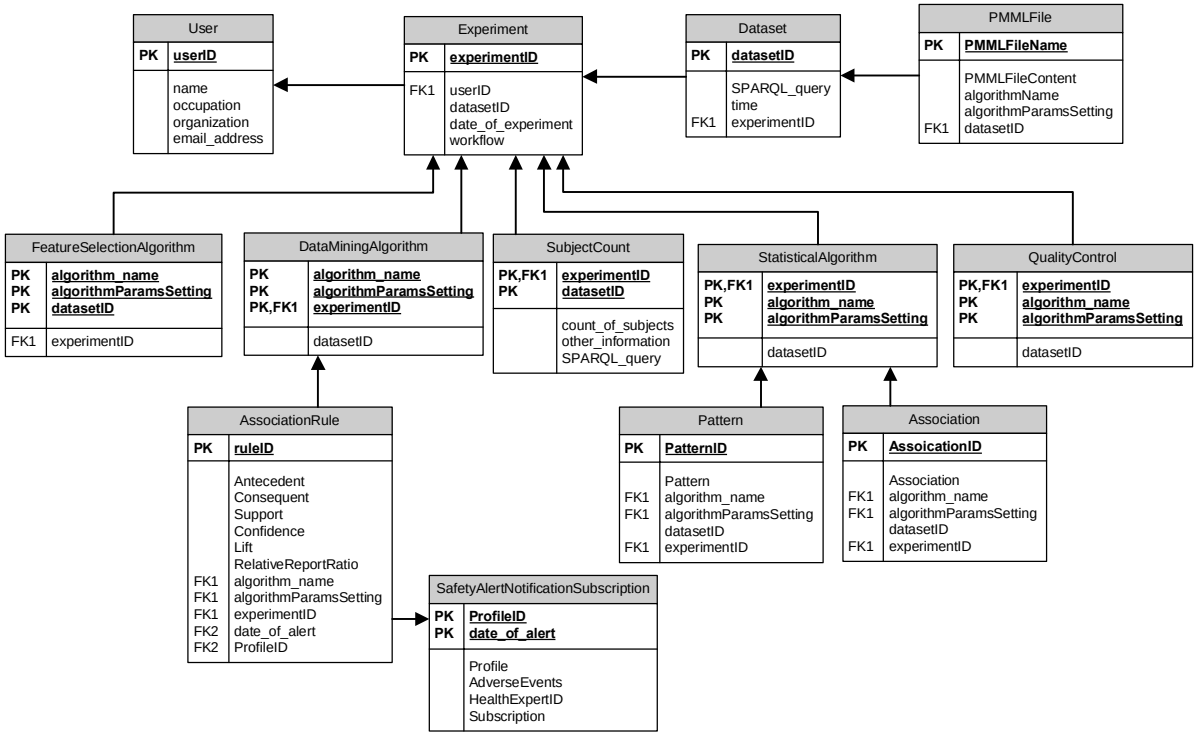

Figure 80: ER Schema of the database

5.4 Choosing a Database System

During database design, a number of database systems such as MySQL, Oracle, SQL Server, PostgreSQL and MS Access were considered in order to select the most appropriate database system to implement the database design. Because all the components are integrated into the Galaxy server and PostgreSQL works very well with Galaxy server, PostgreSQL is selected for use. Moreover PostgreSQL provides interfaces to be used in different languages such as C++ and Java. Each entity of the ER schema is implemented as a table.

5.5 Choosing Data Types of Table Fields

Having chosen PostgreSQL to implement the database design, the various data types of PostgreSQL are reviewed in order to choose the appropriate ones to implement the table fields. Identification Numbers such as userID, datasetID are implemented as serial data typeswhich represent auto-incrementing integers. Other data fields which hold text are implemented as text data type. Date fields which represent date of experiments and date of generating safety alerts are implemented as date data type.

5.5.1 User Table

userID is a serial data type (an auto-incrementing integer) because userID is the primary key of User table. For example, the fields: Name, Occupation, Organization and Email_address are text data types.

Table 37: Example Data types of User table

Field	***Data type***	***Description***
userID	serial	auto-incrementing integer
Name	text	string
Occupation	text	string
Organization	text	string
Email_address	text	string
Role	set	It can receive more than one options
Subscription_date	date	Date of subscription
Working_area	text	string
Country	text	string

5.5.2 Experiment Table

Table 38: Data types of Experiment table

Field	*Data type*	*Description*
userID	serial	auto-incrementing integer
datasetID	serial	auto-incrementing integer
experimentID	serial	auto-incrementing integer
time_of_experiment	Timestamp without time zone	time when the experiment started
Workflow	text	string
purpose	text	string

5.5.3 Dataset Table

Table 39: Data types of Dataset table

Field	*Data type*	*Description*
DatasetID	Serial	auto-incrementing integer
SPARQL_query	Text	string
ExperimentID	Serial	auto-incrementing integer
Time	timestamp without time zone	The local time when the SPARQL query was run

5.5.4 SubjectCount Table

Table 40: Data types of SubjectCount table

Field	*Data type*	*Description*
datasetID	serial	auto-incrementing integer
experimentID	serial	auto-incrementing integer
Count_of_subjects	integer	integer
Other_information	text	string
SPARQL_query	Text	string

5.5.5 FeatureSelectionAlgorithm Table

Table 41: Data types of FeatureSelectionAlgorithm table

Field	*Data type*	*Description*
Algorithm_name	Text	string
AlgorithmParamsSetting	Text	string
ExperimentID	serial	auto-incrementing integer
DatasetID	serial	auto-incrementing integer

5.5.6 DataMiningAlgorithm Table

Table 42: Data types of DataMiningAlgorithm table

Field	*Data type*	*Description*
Algorithm_name	text	String
AlgorithmParamsSetting	text	string
ExperimentID	serial	auto-incrementing integer
DatasetID	serial	auto-incrementing integer

5.5.7 StatisticalAlgorithm Table

Table 43: Data types of StatisticalAlgorithm table

Field	*Data type*	*Description*
Algorithm_name	text	string
AlgorithmParamsSetting	text	string
ExperimentID	serial	auto-incrementing integer
DatasetID	serial	auto-incrementing integer
p-value	Numeric	Real number

5.5.8 QualityControl Table

Table 44: Data types of QualityControlAlgorithm table

Field	*Data type*	*Description*
Algorithm_name	text	string
AlgorithmParamsSetting	text	string
ExperimentID	serial	auto-incrementing integer
DatasetID	serial	auto-incrementing integer

5.5.9 PMMLFile Table

Table 45: Data types of PMMLFile table

Field	*Data type*	*Description*
PMMLFileName	text	string
PMMLFileContent	text	string
Algorithm_name	text	string
AlgorithmParamsSetting	text	string
DatasetID	serial	auto-incrementing integer

5.5.10 AssociationRule Table

Table 46: Data types of AssociationRule table

Field	*Data type*	*Description*
RuleID	serial	auto-incrementing integer
Antecedent	text	string
Consequent	text	string
Support	integer	integer
Confidence	numeric	number with any precision and scale
Lift	numeric	number with any precision and scale
RRRatio	numeric	number with any precision and scale
ExperimentID	serial	auto-incrementing integer
Algorithm_name	text	string
AlgorithmParamsSetting	text	string

5.5.11 Association Table

Table 47: Data types of Association table

Field	*Data type*	*Description*
AssociationID	serial	Auto-incrementing integer
Association	text	string
ExperimentID	serial	auto-incrementing integer
DatasetID	serial	auto-incrementing integer
Algorithm_name	Text	string
algorithmParamsSetting	Text	string

5.5.12 Pattern Table

Table 48: Data types of Pattern table

Field	*Data type*	*Description*
PatternID	serial	Auto-incrementing integer
Pattern	text	string
ExperimentID	serial	auto-incrementing integer
DatasetID	serial	auto-incrementing integer
Algorithm_name	Text	string
algorithmParamsSetting	Text	string

5.5.13 SafetyAlertNotificationSubscription Table

Table 49: Data types of SafetyAlertNotificationSubscription table

Field	*Data type*	*Description*
ProfileID	serial	Auto-incrementing integer
Time	timestamp without time zone	Local time when the safety alert was generated
Profile	text	String
AdverseEvents	text	String
HealthExpertID	serial	Auto-incrementing integer
Subscription	varchar(3)	Variable-length string of maximum 3 characters

Tables can be altered so that new fields can be added to the tables and existing fields can be deleted from the tables. This can be achieved using the Alter statement [51] in SQL.

5.6 Database Backup and Restore

The database should be backed up on a regular basis e.g. daily or weekly. This can be done using the commands pg_dump and psql available in PostgreSQL. pg_dump creates a dump file (text file) which contains SQL statements to recreate the database. pg_dump takes the dump file as input and recreates the database.

5.7 Database Interfaces with Components

The components can be implemented in a wide variety of languages such as Python, R, Perl, Java and C/C++ etc. Database interfaces in different languages are used to connect the components with the database. The following are PostgreSQL interface libraries for applications in different languages:

- R: RPostgreSQL [45]package;
- Java: JDBC driver [46];
- C/C++: libpq-C [47] and libpqxx [48];
- Perl: Perl DBI [49];
- Python: Psycopg [50].

6 Post-processing

This section includes functions that aim to process previously generated results that exist in the previously described database and filtering, re-processing, or annotating them with external results or information as appropriate.

6.1 Knowledge Extraction and Filtering Mechanism

The results of the analyses of the processing steps are stored in the database described in Section 4.6. The knowledge extraction and filtering mechanism allows filtering of rules and associations from the database that are considered as outliers and do not apply in the general population. Filtering will be performed based on statistical tests of the extracted rules and based on expert's knowledge. Moreover it will allow the insertion of rules and associations that are not listed in the database but exist in the literature.

6.1.1 Module Design

Further details of the module are provided in Table 50.

Table 50: Knowledge Extraction and Filtering Mechanism

Name	Knowledge Extraction and Filtering Mechanism
Description	This module allows the user to insert new rules in the database from the literature or delete rules from the database that are considered as outliers and do not apply in the general population. It also allows the filtering of existing rules in the database.
Responsibilities	Enables insertion/deletion of rules in the database and the filtering of existing rules in the database.
Constraints	Experts knowledge is needed for inserting new rules
Composition	Node for inserting/deleting/filtering rules and associations from the database.
Inputs	Associations or association rules
Outputs	Filtered associations or association rules
Interactions/ Interfaces	This component interacts with the database to retrieve associations and association rules and to insert /delete rules and associations into/from the database.

The knowledge extraction and filtering mechanism consists of the database and the class KnowledgeExtractionAndFiltering as shown in Figure 81.

The class KnowledgeExtractionAndFiltering uses the database to retrieve and filter existing rules and associations. The attributes of the class are the thresholds that are used for filtering any rules and associations in the database. The class has a method for initializing the threshold values called setThresholds(). The methods addNewRule() and removeRule() are used to insert new rules or remove rules from the database respectively, whereas the filterRules() method filters all of the rules in the database.

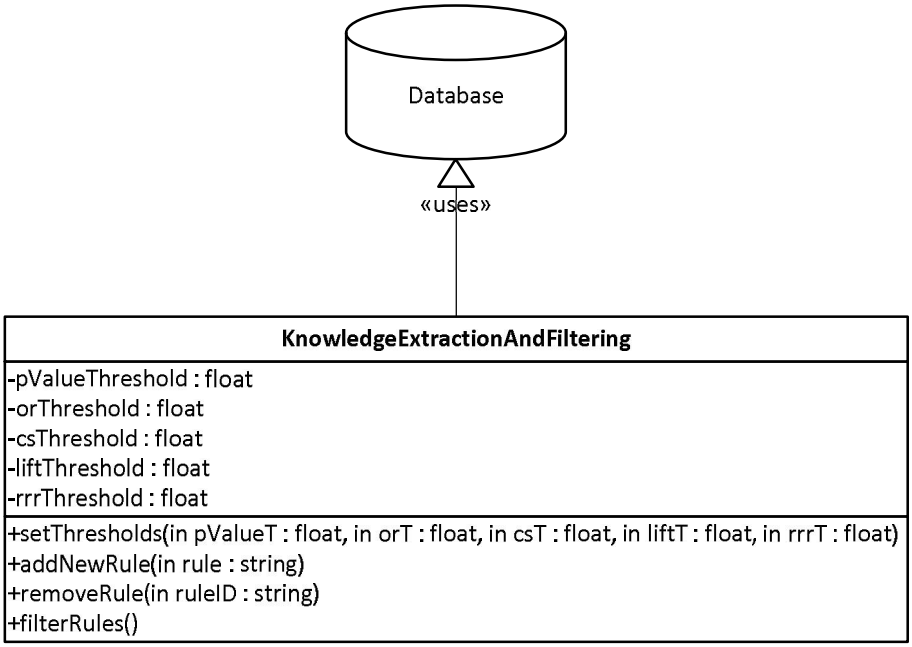

Figure 81: Class diagram of the knowledge extraction and filtering mechanism

6.1.2 User Interface Design

The user interface of the subcomponents of the knowledge extraction and filtering mechanism are illustrated in Figure 82.

Figure 82: The add new rule user interface mock-up

For adding a new rule, the user needs to type in the rule and press the add rule button, whereas for removing a rule, the user specifies the rule that needs to be deleted and then presses the “Remove Rule” button (Figure 83).

Figure 83: The remove rule user interface mock-up

For filtering the rules in the database, the user needs to specify the threshold values of the different evaluation metrics. Once these are specified, the rules can be filtered by pressing the "Filter Rules" button (Figure 84).

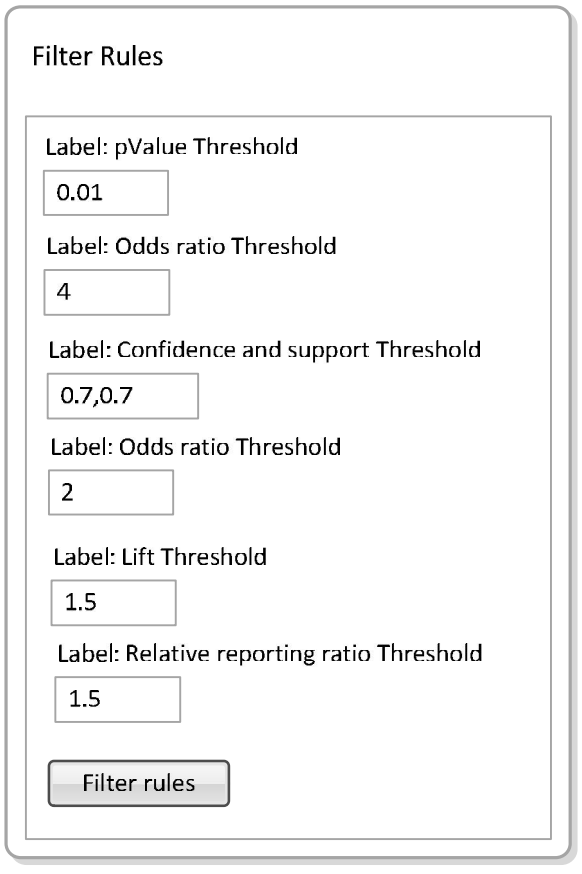

Figure 84: The filter rules user interface mock-up

7 Visualization

This component visualizes the association rules, adverse events and associations of the data analysis. The following sections will present the visualizations that will be used.

7.1 Visualization of Association Rules

7.1.1 Visualizing Number of Rules involving Each Adverse Event

The number of rules for each adverse event can be visualized using a histogram. Figure 85Figure 85 is an example histogram illustrating the number of rules of each adverse event. The x-axis represents the adverse events; the y-axis represents the number of rules predicting the adverse events.

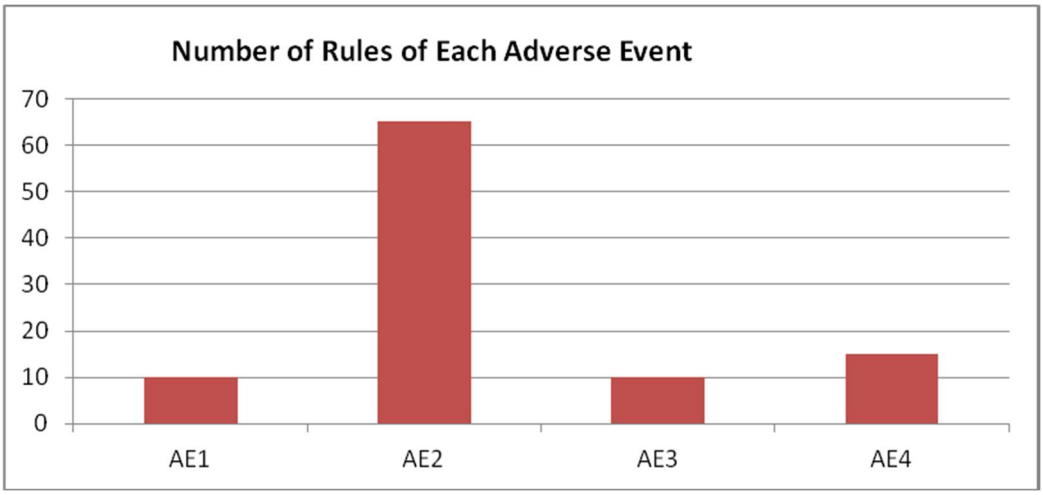

Figure 85: Number of rules of each adverse event

7.1.2 Visualizing Rule Performance

The performance of each association rule such as support, confidence, lift and Relative Report Ratio can be visualized on a bar chart to identify the strongest rules and weakest rules. The quality of each rule can be retrieved from the AssociationRule table. Figure 86 is an example bar chart of the measure of confidence of rules derived from data mining. The x-axis represents the rules; the y-axis represents the confidence of each rule.

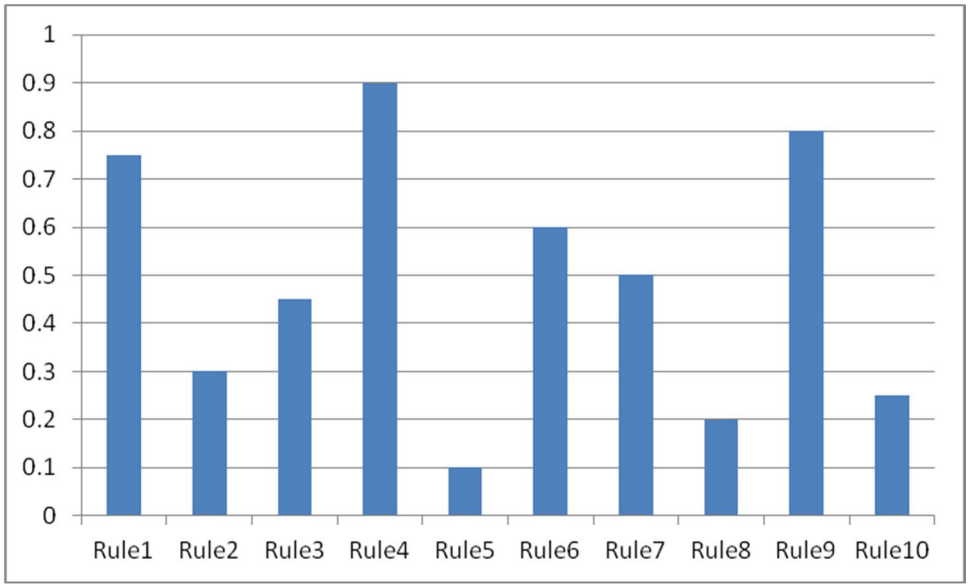

Figure 86: Confidence of association rules

7.2 Visualization of Adverse Events

The number of adverse events of each profile can be visualized using a histogram (Figure 87) is an example histogram illustrating the number of adverse events of each profile. The x-axis represents the profiles; the y-axis represents the number of adverse events.

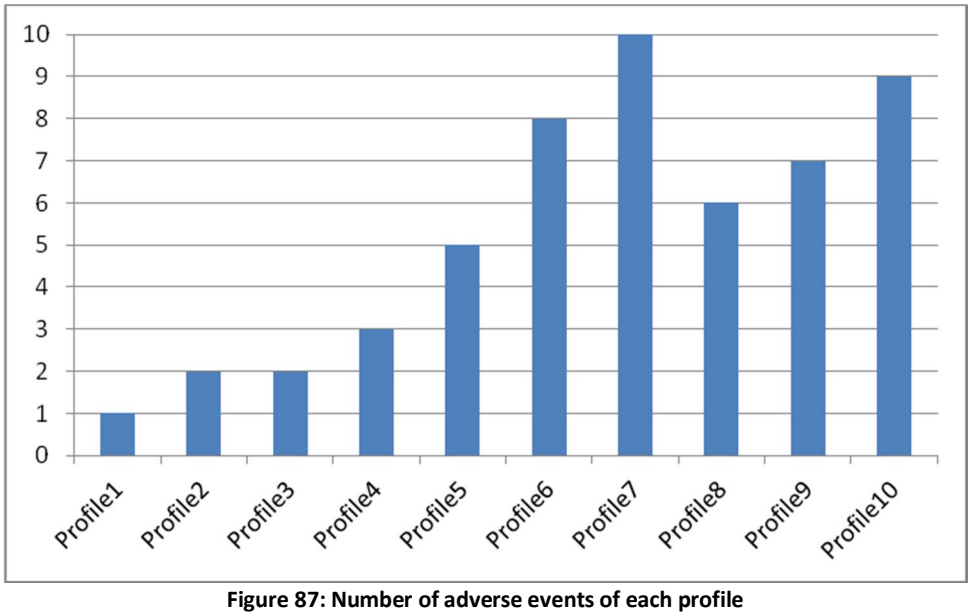

Figure 87: Number of adverse events of each profile

7.2.1 Visualizing the Occurrences of Adverse Events

The number of occurrences of each adverse event can be visualized using a histogram to identify the most common adverse event existing in each profile.

Figure 88 illustrates the number of profiles for each adverse event. The x-axis represents adverse events; the y-axis represents the number of occurrences of adverse events.

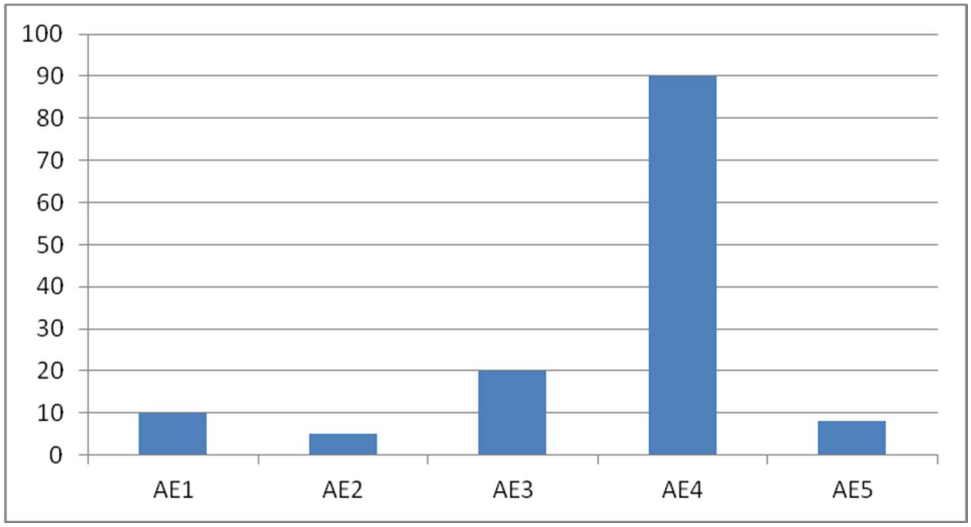

Figure 88: Occurrences of adverse events

7.2.2 Visualizing Number of Adverse Events of Each Medication

The number of adverse events of each medication can be visualized using a histogram to identify the drugs causing most adverse events. Figure 89 illustrates the number of adverse events of each medication. The x-axis represents the drugs; the y-axis represents the number of adverse events of drugs.

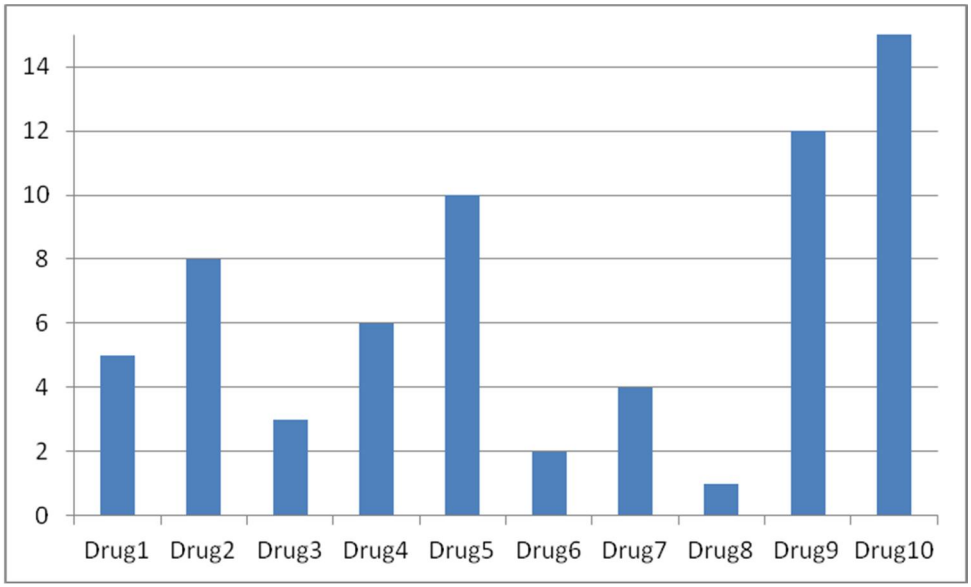

Figure 89: Number of adverse events of each medication

7.3 Visualization of Associations

7.3.1 Visualizing the Number of Associations of Each Adverse Event

The number of associations of each adverse event can be visualized in the same way as the visualization of the number of rules of each adverse event. Therefore, this will provide information of which adverse events were identified having associations with other factors and how many associations were identified for each adverse event. Figure 90Figure 90 illustrates a histogram with the number of associations for each adverse event. The x axis represents the adverse events, whereas the y axis represents the number of associations related to the adverse events.

7.3.2 Visualizing the Associations of a Specific Variable

It may be important for the user to be able to see which factors were found to be associated with a specific variable and the significance of the associations. Because the same association might be identified by different algorithms, the significance value will probably not be the same. Thus it is important to be able to see the diversion among the significance values of a specific association. For this reason a boxplot chart will be used for visualising the factors that were found associated with a specific variable and the variance in significance value for each factor. An example is illustrated in

Figure 91Figure 91. The x axis represents the factors associated with the variable of interest, whereas the y axis represents the –log(p-value) of the associations identified.

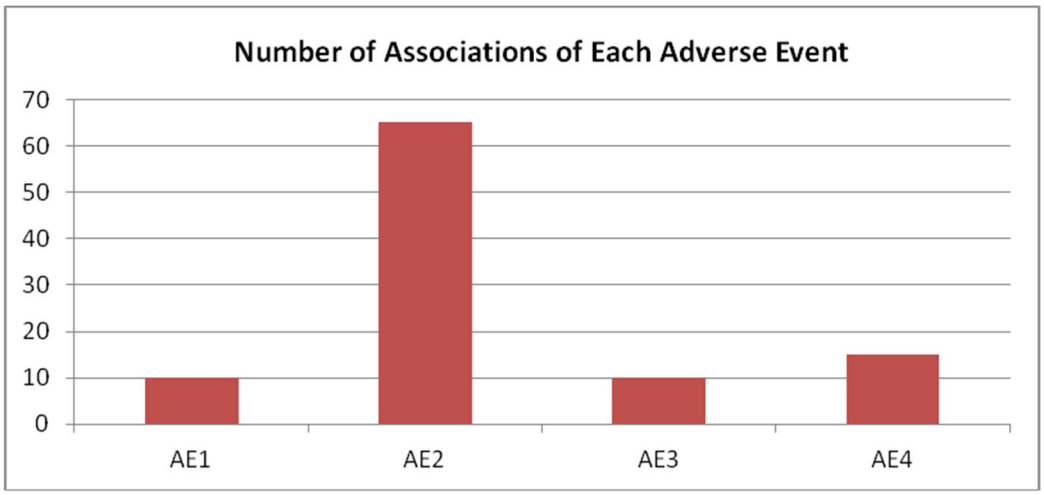

Figure 90: Number of associations of each adverse event

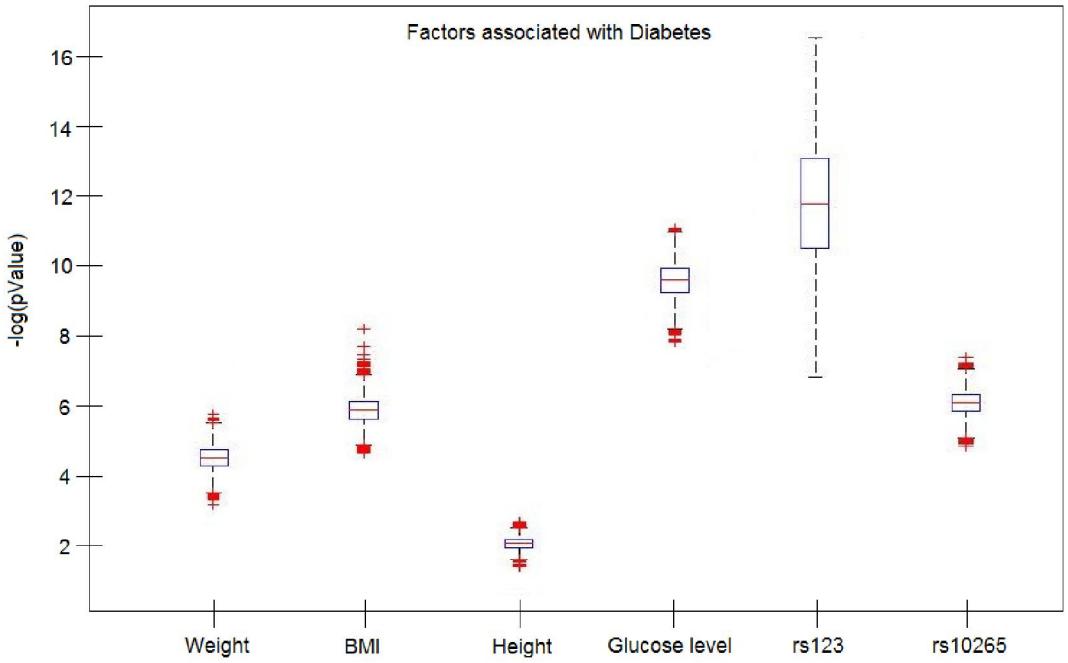

Figure 91: Association of a specific variable

8 Conclusion

The functional requirements of the Data Analysis Space have been derived from D1.1 “Requirements Analysis”. Some non-functional requirements of the Data Analysis Space have also been analyzed. Thereafter, the Data Analysis Space was designed to consist of the following software components:

1. Input component which retrieves RDFized data cubes from the Linked Medical Data Space;
2. Pre-processing component which includes quality control and feature selection algorithms;
3. Processing component which consists of subject counting, single hypothesis testing, data mining algorithms, adverse event early notification mechanism and a database to store the details of data analyses;
4. Post-processing component which includes the knowledge extraction and filtering mechanism;
5. Visualization component which includes visualization functionalities for genetic associations, association rules and adverse events.

The components will be integrated into the Galaxy server. All requirements in section 3 have been addressed (see Table 51).

Table 51: How Requirements Have Been Addressed

Requirement	How Requirement Has Been Addressed
F3	Data mining algorithms are developed (Section 4.5.3)
F4	Single hypothesis testing algorithms are developed (Section 4.5.2)
F5	Statistical analysis techniques are developed (Section 5.3)
F6	Manual selection of association rules is supported (Section 6)
F7	Manual selection of association rules is supported (Section 6)
F8	Rules Filtering is supported (Section 6)
F9	Manual addition of rules is supported (Section 6).
F10	Manual removal of rules is supported (Section 6).
F14	Recommendations in subject selection inclusion/exclusion criteria are supported (Section 4.5.1)
F17	Chemoinformatic analysis is supported by statistical hypothesis testing and data mining (Section 4.5.3)
F18	Customization of Signaling threshold/level is supported (sections 4.5.3.4 and 6)
F19	Visualization of results is supported (Section 7)
F20	Safety alerts notification is supported (Section 4.7)
F21	Quality control techniques are developed (Section 4.4.1)
NF1	RDF Data cubes are retrieved from LMDS (Section 4.3)
NF2	RDF Data cubes are retrieved from LMDS (Section 4.3)
NF26	The Galaxy web server can be instantiated on cloud computing infrastructures or can be interfaced with grid clusters (Section 4.2.1)
NF27	The Galaxy web server can be instantiated on cloud computing infrastructures or can be interfaced with grid clusters (Section 4.2.1)

9 References

[1] Goecks, J.; Nekrutenko, A.; Taylor, J.; Galaxy Team, T. "Galaxy: A comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences". *Genome Biology* 11 (8), 2010

[2] Taylor, J.; Schenck, I.; Blankenberg, D.; Nekrutenko, A. "Using Galaxy to Perform Large-Scale Interactive Data Analyses". Current Protocols in Bioinformatics. Current protocols in bioinformatics / editoral board, Andreas D. Baxevanis ... [et al.] Chapter 10: UNIT 10.5, 2007

[3] Blankenberg, D.; Coraor, N.; Von Kuster, G.; Taylor, J.; Nekrutenko, A.; Galaxy, T. "Integrating diverse databases into an unified analysis framework: A Galaxy approach". *Database* 2011

[4] "Resource Description Framework (RDF) Model and Syntax Specification" http://www.w3.org/TR/PR-rdf-syntax/

[5] RDF Tutorial http://www.w3schools.com/rdf/default.asp

[6] R. Bellazzi and B. Zupan, "Predictive data mining in clinical medicine: current issues and guidelines," *International Journal of Medical Informatics,* vol. 77, pp. 81–97, 2008.

[7] D. Regori, R. Bigi, L. Cortigiani, F. Bovenzi, C. Fiorentini, and E. Picano, "Non-invasive risk stratification of coronary artery disease: an evaluation of some commonly used statistical classifiers in terms of predictive accuracy and clinical usefulness," *Journal of Evaluation in Clinical Practice,* vol. 15, pp. 777-781, 2009.

[8] F. Demichelis, P. Magni, P. Piergiorgi, M. A. Rubin, and R. Bellazzi, "A hierarchical Naive Bayes model for handling sample heterogeneity in classification problems: an application to tissue microarrays," *BMC Bioinformatics,* vol. 7, 2006.

[9] J. R. Quinlan, "C4.5: Programs for Machine Learning," San Mateo, CA: Morgan Kaufmann, 1993.

[10] M. J. Glasgow and P. J. Kaboli, "Detecting adverse drug events through data mining,"*American Journal of Health-System Pharmacy,* 2010.

[11] E. Chazard, G. Ficheur, S. Bernonville, M. Luyckx, and R. Beuscart, "Data Mining to Generate Adverse Drug Events Detection Rules," *IEEE Transactions on Information Technology In Biomedicine,* vol. 15, 2011

[12] M. Rouane-Hacene, Y. Toussaint, and P. Valtchev, "Mining Safety Signals in Spontaneous Reports Database Using Concept Analysis," in *Proceedings of the 12th Conference on Artificial Intelligence in Medicine: Artificial Intelligence in Medicin (AIME '09)*, pp. 285-294,2009.

[13] R. Harpaz, H. S. Chase, and C. Friedman, "Mining multi-item drug adverse effect associations in spontaneous reporting systems," *BMC Bioinformatics,* vol. 11 p. S7, 2010.

[14] Bate and S. J. Evans, "Quantitative signal detection using spontaneous ADR reporting," *Pharmacoepidemiol Drug Safety,* vol. 18, pp. 427-436, 2009.

[15] Szarfman, S. G. Machado, and R. T. O'Neill, "Use of screening algorithms and computer systems to efficiently signal higher-than-expected combinations of drugs and events in the US FDA's spontaneous reports database," *Drug Safety,* vol. 25, pp.381-392, 2002.

[16] W. P. Stephenson and M. Hauben, "Data mining for signals in spontaneous reporting databases: proceed with caution," *Pharmacoepidemiol Drug Safety,* vol. 16, pp. 59-65, 2007.

[17] T. Bharat, S. Grundschober, and L. Doessegger, "Detecting signals of drug-drug interactions in a spontaneous reports database," *British Journal of Clinical Pharmacology,* vol. 64, pp. 489-495, 2007.

[18] J. Han and M. Kamber, "Data Mining: Concepts and Techniques," San Francisco, CA: Morgan Kaufmann Publishers Inc., 2000.

[19] D. Hand, H. Mannila, and P. Smyth, "Principles of Data Mining," Cambridge, MA: MIT Press, 2001.

[20] H. Witten and E. Frank, "Data mining: practical machine learning tools and techniques, 2nd ed," San Francisco: Morgan Kaufmann, 2005.

[21] L. Polkowski, "Rough Sets: Mathematical Foundations," Berlin: Physica-Verlag, 2002.

[22] D. Tian, X. Zeng, and J. Keane, "Core-generating Approximate Minimum Entropy Discretization for Rough Set Feature Selection in Pattern Classification," *International Journal of Approximate Reasoning,* vol. 52, pp. 863-880, 2011.

[23] D. Tian, X. Zeng and J. Keane, Core-generating Discretization for Rough Set Feature Selection, Transactions on Rough Sets XIII, LNCS6499, pp.135-158, 2011

[24] D. Tian, J. Keane, and X. Zeng, "Evaluating the effect of rough set feature selection on the performance of decision trees," in *IEEE International Conference on Granular Computing*, pp. 57-62, 2006

[25]G. Karegowda, S. A. Manjunath, and A. M. Jayaram, "Application of Genetic lgorithm Optimized Neural Network Connection Weights fpr Medical Diagnosis of Pima Indians Diabetes," *International Journal on Soft Computing (IJSC),* vol. 2, 2011.

[26]Antoniades, L. Loizou, A. Aristodimou, and C. S. Pattichis, "A binary format for genetic data designed for large whole genome studies that enable both marker and strand based analyses," in *8th IEEE International Conference on BioInformatics and BioEngineering (BIBE 2008)*, pp. 1-4, 2008.

[27]G. R. Abecasis, S. S. Cherny, W. O. Cookson, and L. R. Cardon, "Merlin-rapid analysis of dense genetic maps using sparse gene flow trees," *Nature genetics,* vol. 30, pp.97–101, 2002.

[28]S. Purcell, B. Neale, K. Todd-Brown, L. Thomas, M. A. Ferreira, D. Bender, J. Maller, P.Sklar, P. I. de Bakker, M. J. Daly, and P. C. Sham, "PLINK: a tool set for whole-genome association and population-based linkage analyses," *The American Journal of Human Genetics,* vol. 81, pp. 559–575, 2007.

[29]L. Hosking, S. Lumsden, K. Lewis, A. Yeo, L. McCarthy, A. Bansal, J. Riley, I. Purvis, and C. F. Xu, "Detection of genotyping errors by Hardy-Weinberg equilibrium testing," *European Journal of Human Genetics,* vol. 12, pp. 395–399, 2004.

[30]W. M. Liu, X. Di, G. Yang, H. Matsuzaki, J. Huang, R. Mei, T. B. Ryder, T. A. Webster, S. Dong, G. Liu, K. W. Jones, G. C. Kennedy, and D. Kulp, "Algorithms for large-scale genotyping microarrays," *Bioinformatics,* vol. 19, pp. 2397–2403, 2003.

[31]B. Antonisamy, S. Christopher, and P. P. Samuel, "Biostatistics: principles and practice," Tata McGraw-Hill Education, 2010.

[32]E. Vittinghoff, "Regression methods in biostatistics: linear, logistic, survival, and repeated measures models," Springer Verlag, 2005.

[33]D. S. Falconer and T. F. C. Mackay, "Introduction to quantitative genetics," Longman, 1996.

[34]The International HapMap Consortium and T. I. H. Consortium, "A haplotype map of the human genome," *Nature genetics,* vol. 437, pp. 1299–1320, 2005.

[35]Adeyemo, R. Cooper, R. Ward, E. S. Lander, M. J. Daly, and D. Altshuler, "The structure of haplotype blocks in the human genome," *Science,* vol. 296, p. 2225, 2002.

[36]H. J. Cordell, "Detecting gene-gene interactions that underlie human diseases," *Nature Reviews Genetics,* vol. 10, pp. 392–404, 2009.

[37]S. Sherry, M. Ward, M. Kholodov, J. Baker, L. Phan, E. Smigielski, and K. Sirotkin, "dbSNP: the NCBI database of genetic variation," *Nucleic acids research,* vol. 29, pp.308-311, 2001.

[38]Gibbs R.A. et al., "The international HapMap project," *Nature,* vol. 426, pp. 789–796, 2003.

[39]J. C. Barrett, B. Fry, J. Maller, and M. J. Daly, "Haploview: analysis and visualization of LD and haplotype maps," *Bioinformatics,* vol. 21, pp. 263 -265, 2005.

[40]Deliverable D1.1 Requirements Analysis

[41]Deliverable D1.2 Reference Architecture

[42]C. Ritchie, Database Principles and Design, 3rd Edition, Thomson Learning, 2008

[43]T.M. Connolly, C. E. Begg, Database systems : a practical approach to design, implementation, and management, 5th edition, Addison Wesley, 2010

[44]RPostgreSQL: http://code.google.com/p/rpostgresql/

[45]JDBC: http://www.postgresql.org/docs/7.4/static/jdbc.html

[46]Libpq-C: http://www.postgresql.org/docs/8.0/static/libpq.html

[47]libpqxx: http://www.postgresql.org/docs/8.2/static/external-interfaces.html

[48]Perl DBI: http://www.postgresql.org/docs/8.2/static/external-interfaces.html

[49]Psycopg: http://www.postgresql.org/docs/8.2/static/external-interfaces.html

[50]dALTER: http://www.postgresql.org/docs/9.2/static/ddl-alter.html

[51]PMML: http://en.wikipedia.org/wiki/PMML

[52]XML: http://en.wikipedia.org/wiki/XML

10 Appendix

10.1 PMMLFileContent Field of PMMLFile Table


```
<PMML version="4.0" xsi:schemaLocation="http://www.dmg.org/PMML-4_0
http://www.dmg.org/v4-0/pmml-4-0.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns="http://www.dmg.org/PMML-4_0">
 <Header copyright="Copyright (c) 2011 Zementis Inc. (www.zementis.com)"
description="Association Rules Model">
  <Timestamp>Apr 8, 2011</Timestamp>
 </Header>
 <DataDictionary numberOfFields="18">
  <DataField dataType="string" name="apple" optype="categorical"/>
  <DataField dataType="string" name="orange" optype="categorical"/>
  <DataField dataType="string" name="banana" optype="categorical"/>
  <DataField dataType="string" name="cracker" optype="categorical"/>
  <DataField dataType="string" name="tofu" optype="categorical"/>
  <DataField dataType="string" name="tomato" optype="categorical"/>
  <DataField dataType="string" name="beer" optype="categorical"/>
  <DataField dataType="string" name="milk" optype="categorical"/>
  <DataField dataType="string" name="bread" optype="categorical"/>
  <DataField dataType="string" name="juice" optype="categorical"/>

  <DataField dataType="string" name="spinach" optype="categorical"/>
  <DataField dataType="string" name="water" optype="categorical"/>
  <DataField dataType="string" name="egg" optype="categorical"/>
  <DataField dataType="string" name="coke" optype="categorical"/>
  <DataField dataType="string" name="beef" optype="categorical"/>
  <DataField dataType="string" name="pork" optype="categorical"/>
  <DataField dataType="string" name="cabbage" optype="categorical"/>
  <DataField dataType="string" name="chicken" optype="categorical"/>
 </DataDictionary>
 <AssociationModel functionName="associationRules" minimumConfidence="0.630434782608696"
minimumSupport="0.2" modelName="Rectangular_AR" numberOfItems="18"
numberOfItemsets="292" numberOfRules="998" numberOfTransactions="435">
  <MiningSchema>
   <MiningField name="apple" usageType="active"/>
   <MiningField name="orange" usageType="active"/>
   <MiningField name="banana" usageType="active"/>
   <MiningField name="cracker" usageType="active"/>
   <MiningField name="tofu" usageType="active"/>
   <MiningField name="tomato" usageType="active"/>
   <MiningField name="beer" usageType="active"/>
```

```
    <MiningField name="milk" usageType="active"/>
    <MiningField name="bread" usageType="active"/>
    <MiningField name="juice" usageType="active"/>
    <MiningField name="spinach" usageType="active"/>
    <MiningField name="water" usageType="active"/>
    <MiningField name="egg" usageType="active"/>
    <MiningField name="coke" usageType="active"/>
    <MiningField name="beef" usageType="active"/>
    <MiningField name="pork" usageType="active"/>
    <MiningField name="cabbage" usageType="active"/>
    <MiningField name="chicken" usageType="active"/>
   </MiningSchema>
   <Output>
    <OutputField feature="predictedValue" name="predictedValue" dataType="string"
optype="categorical"/>
   </Output>
   <Item id="1" value="apple"/>
   <Item id="2" value="orange"/>
   <Item id="3" value="banana"/>
   <Item id="4" value="cracker"/>
   <Item id="5" value="tofu"/>
   <Item id="6" value="tomato"/>
   <Item id="7" value="beer"/>
   <Item id="8" value="milk"/>
   <Item id="9" value="bread"/>
   <Item id="10" value="juice"/>
   <Item id="11" value="spinach"/>
   <Item id="12" value="water"/>
   <Item id="13" value="egg"/>
   <Item id="14" value="coke"/>
   <Item id="15" value="beef"/>
   <Item id="16" value="pork"/>
   <Item id="17" value="cabbage"/>
   <Item id="18" value="chicken"/>
   <Itemset id="12" numberOfItems="1">
    <ItemRef itemRef="3"/>
   </Itemset>
         <Itemset id="13" numberOfItems="1">
    <ItemRef itemRef="2"/>
   </Itemset>
   <Itemset id="14" numberOfItems="1">
    <ItemRef itemRef="1"/>
   </Itemset>
   <Itemset id="271" numberOfItems="4">
```

```
    <ItemRef itemRef="2"/>
    <ItemRef itemRef="3"/>
    <ItemRef itemRef="5"/>
    <ItemRef itemRef="9"/>
   </Itemset>
        <Itemset id="274" numberOfItems="4">
    <ItemRef itemRef="1"/>
    <ItemRef itemRef="2"/>
    <ItemRef itemRef="5"/>
    <ItemRef itemRef="9"/>
   </Itemset>
        <Itemset id="292" numberOfItems="4">
    <ItemRef itemRef="1"/>
    <ItemRef itemRef="3"/>
    <ItemRef itemRef="5"/>
    <ItemRef itemRef="9"/>
   </Itemset>
        <AssociationRule antecedent="271" confidence="0.9296875" consequent="14"
lift="1.59847455533597" support="0.273563218390805"/>
   <AssociationRule antecedent="292" confidence="0.952" consequent="13"
lift="1.71123966942149" support="0.273563218390805"/>
   <AssociationRule antecedent="274" confidence="0.86231884057971" consequent="12"
lift="1.56949245042751" support="0.273563218390805"/>
        </AssociationModel>
</PMML>
```