European Security in Health Data Exchange

Deliverable D5.7
Data Sensitivity Analysis Tool

<table>
<thead>
<tr>
<th>Editor(s):</th>
<th>Sigal Asaf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Responsible Partner:</td>
<td>IBM</td>
</tr>
<tr>
<td>Status-Version:</td>
<td>Final – v1.0</td>
</tr>
<tr>
<td>Date:</td>
<td>28/06/2018</td>
</tr>
<tr>
<td>Distribution level (CO, PU):</td>
<td>CO</td>
</tr>
</tbody>
</table>
Abstract: This report describes the Data Sensitivity Analysis tool prototype. It describes the tool functionality, its input and output, and its APIs.

Keyword List: Data privacy, sensitivity, sensitive data, classification

Disclaimer: This document reflects only the author’s views and neither Agency nor the Commission are responsible for any use that may be made of the information contained therein.
## Document Revision History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Modifications Introduced</th>
<th>Modification Reason</th>
<th>Modified by</th>
</tr>
</thead>
<tbody>
<tr>
<td>v0.1</td>
<td>29/05/2018</td>
<td>First draft version</td>
<td></td>
<td>Sigal Asaf (IBM)</td>
</tr>
<tr>
<td>V0.2</td>
<td>04/06/2018</td>
<td>Internal comments and suggestions</td>
<td></td>
<td>Sigal Asaf (IBM)</td>
</tr>
<tr>
<td>v0.3</td>
<td>19/06/2018</td>
<td>Comments and suggestions received by consortium partners</td>
<td></td>
<td>Stefanie Wiegand (IT Innovation)</td>
</tr>
<tr>
<td>V1.0</td>
<td>28/06/2018</td>
<td>Incorporating IT Innovation review</td>
<td></td>
<td>Sigal Asaf (IBM)</td>
</tr>
</tbody>
</table>
# Table of Contents

Table of Contents .................................................................................................................................................. 4  
List of Figures ...................................................................................................................................................... 5  
List of Tables ....................................................................................................................................................... 5  
Terms and abbreviations ........................................................................................................................................ 6  
Executive Summary .............................................................................................................................................. 7  
1 Introduction .......................................................................................................................................................... 8  
1.1 About this deliverable ......................................................................................................................................... 8  
1.2 Document structure ........................................................................................................................................ 8  
2 Software Release ................................................................................................................................................ 9  
2.1 Functional description ......................................................................................................................................... 9  
2.1.1 Technical specifications ................................................................................................................................ 9  
2.2 User Manual – The Tool APIs ......................................................................................................................... 10  
2.2.1 DataStore .................................................................................................................................................... 11  
2.2.1.1 DatabaseId ................................................................................................................................................ 12  
2.2.1.2 Credentials .............................................................................................................................................. 12  
2.2.1.3 RelationalDatabase ............................................................................................................................... 12  
2.2.2 Classifying a DataStore .............................................................................................................................. 13  
2.2.2.1 DataClassifier .......................................................................................................................................... 13  
2.2.2.2 ColumnClassifierFactory ...................................................................................................................... 14  
2.2.2.3 CompositionClassifier .......................................................................................................................... 14  
2.2.2.4 DataStoreClassifier ............................................................................................................................. 15  
2.2.2.5 DataStoreClassification ........................................................................................................................ 15  
2.2.3 Creating a PostgreSQL Database from csv file ............................................................................................ 17  
2.2.3.1 ColumnInfo ............................................................................................................................................. 17  
2.2.3.2 CsvInfo .................................................................................................................................................. 17  
2.2.3.3 PostgreSqlDatabase ............................................................................................................................. 18  
APPENDIX A: Tool Design Changes ..................................................................................................................... 19
List of Figures

FIGURE 1: DATA CLASSIFIER COMPOSITION ................................................................................. 9
FIGURE 2: DATA STORE CLASSIFICATION CODE EXAMPLE .................................................... 11
FIGURE 3: CLASSIFICATION RESULT ......................................................................................... 11
FIGURE 4: RELATIONAL DATABASE CLASS DIAGRAM ............................................................. 12
FIGURE 5: DATA STORE CLASSIFICATION PROCESS ................................................................. 13
FIGURE 6: LOADING CSV FILE CODE EXAMPLE ....................................................................... 17

List of Tables

TABLE 1 - COMPOSITION CLASSIFIER LIBRARY ...................................................................... 15
## Terms and abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC</td>
<td>European Commission</td>
</tr>
<tr>
<td>GDPR</td>
<td>General Data Protection Regulation</td>
</tr>
</tbody>
</table>
Executive Summary

The process of identifying sensitive data is a necessary step to be able to address EU GDPR regulation which aims primarily to give control to EU citizens and residents over their personal data. The first step to address the GDPR regulation is to find the sensitive/personal data in the organization data stores. Once the sensitive data has been identified, the organizations can provide their customers/users the ability to control their personal data (delete, modify, provides permissions, etc.). The Data Sensitivity Analysis Tool address this step. It finds the sensitive/personal data in relational databases.

This document represents deliverable D5.7 - “Data Sensitivity Analysis Tool (prototype)”. The deliverable focuses on a prototype version of the tool and this document describes its functionality and its API. The prototype will support WP6 by helping SHiELD to identify sensitive/personal information. Personal and sensitive information should be cared carefully to protect patient privacy as required by GDPR.

For each column in the database, the tool indicates if the column is sensitive or not and provides a confidence score (a value between 0.0 to 1.0). The confidence score indicates how much the tool is confident that the specific column is indeed sensitive. In addition, the tool provides explanations why a specific column is considered as sensitive. This is done by displaying additional categories the column belongs too.

The tool itself is configurable. The tool contains a library of data classifiers, each finds if a column belongs to a specific category such as date, email, birth, city, and birth-date. In addition, it enables adding additional categories by adding corresponding data classifiers. The user configures the sensitive classification problem by selecting which categories are related to the problem, and how they relate to the sensitivity category. For example, a user may decide that a column is sensitive if it is either email, or social-id. In addition, the user declares a threshold. A column belongs to a specific category only if its confidence score is above the threshold.

Following this release, we plan to continue developing the tool, to test it on additional representative patient data, and to provide its final version. Its development may include expanding the classifier library, improving category explanations, and enhancing the tool APIs. In addition, we may provide support to additional relational database types to support SHiELDs' data. Finally, we plan to run the tool on our EU partners representative data to help in discovering the data that should be considered as sensitive/personal information.

The document is structured as follows: in Section 1, we provide an overview of the tool. In Section 2, we describe the prototype, its functionality and its APIs.
1 Introduction

This document describes D5.7 – Data Sensitivity Analysis Tool – prototype.

The Data Sensitivity Analysis Tool finds sensitive data in relational databases. The process of identifying sensitive data is a necessary step to be able to address EU GDPR regulation which aims primarily to give control to EU citizens and residents over their personal data.

For each column in the database, the tool indicates if the column is sensitive or not and provides a confidence score (a value between 0.0 to 1.0). The confidence score indicates how much the tool is confident that the specific column is indeed sensitive. In addition, the tool provides explanations why a specific column is considered as sensitive. This is done by displaying additional categories the column belongs too.

The sensitivity problem is configurable. The tool contains a library of data classifiers, each finds if a column belongs to a specific category such as date, email, birth, city, and birth-date. In addition, it enables adding additional categories by adding corresponding data classifiers (see Section 2.2.2). The user configures the sensitive classification problem by selecting which categories are related to the problem, and how they relate to the sensitivity category. For example, a user may decide that a column is sensitive if it is either email, or social-id.

1.1 About this deliverable

This deliverable describes the prototype of the Data Sensitivity Analysis Tool, including functionality and APIs. The prototype enables initial evaluation of the tool architecture and functionalities, and it aims to provide support for WP6 by helping SHIELD to identify sensitive/personal information. Personal and sensitive information should be cared carefully to protect patient privacy as required by GDPR.

1.2 Document structure

This document is structured as follows: Section 1 provides a short overview of the current release. Section 2 describes the tool’s functionality (see Section 2.1) and its APIs (see Section 2.2). Finally, Appendix A describes changes we have made to the design while developing the prototype.
2 Software Release

2.1 Functional description

The Data Sensitivity Analysis Tool finds sensitive data in relational databases. For each column in the database, the tool indicates if the column is sensitive or not and provides a confidence score (a value between 0.0 to 1.0). The confidence score indicates how much the tool is confident that the specific column is indeed sensitive. In addition, the tool provides explanations why a specific column is considered as sensitive. This is done by displaying additional categories the column belongs to. Each category is also attached with confidence score. A column belongs to a specific category if the corresponding confidence score is above a threshold. The user configures which categories to use, how sensitive category is defined, and the threshold value.

The tool contains a library of data classifiers, each finds if a column belongs to a specific category such as date, email, birth, city, and birth-date. In addition, it enables adding additional categories by adding corresponding data classifiers. When configuring the tool, the user selects the corresponding data classifiers, add additional classifier, define the sensitive classifier, and finally, define the confidence threshold.

We present two types of data classifier: Elementary Data Classifier and Composition Data Classifier. The Elementary Data Classifier uses the column title, its instances and the database schema to decide if a column belongs to a specific category. Composition Data Classifier bases its decision on the other categories the column belongs to. For example, the Birth classifier searches for the keyword birth in the column title and is an Elementary Data Classifier. While the BirthDate Data Classifier searches for columns that are classified as both “Date” and “Birth” and is a Composition Data Classifier. The BirthDate composition is: Birth && Date → BirthDate. A composition data classifier also uses an expression which calculates the confidence score. For the BirthDate example, the confidence score is calculated using the expression: $\frac{\text{score}(\text{Date}) + \text{score}(\text{Birth})}{2}$

The user defines which columns should be considered as sensitive by defining the sensitive data classifier. For example, the user can define that a column is sensitive if it has either a Birth Date, a Personal City, or a Personal Identifier flag. The column is Personal Identifier if it has either the Italian Fiscal Code flag or Personal Health Index flag. Same idea applies to Personal City and Birth Date composition. Figure 1 shows this concept.

Figure 1: Data Classifier Composition

| Personal Identifier: Italian Fiscal Code || Personal Health Index |
| Personal City: City & (Birth || Address) |
| Birth Date: Birth & date |
| Sensitive: PersonIdentifier || PersonalCity || BirthDate |

In addition, the user defines a threshold score. A column belongs to a specific category only if the corresponding confidence score is above this threshold.

2.1.1 Technical specifications

We use Java 8 for the prototype implementation.
The prototype uses PostgreSQL database. We also developed a mechanism which enables uploading tables from csv files into the database. Using this mechanism, we can support other databases. First, data is exported as csv file. Then, the Data Sensitivity Analysis Tool uploads the data into the PostgreSQL database. Now, the classification analysis, which uses PostgreSQL can run on the data in the database.

2.2 User Manual – The Tool APIs

To find the sensitive columns in a relational database and to identify their categories, the user codes a short program which uses the tool APIs. In this program the user defines the classification problem: the database to classify, the classifiers that are relevant to the problem, and the sensitivity classifier.

Figure 2 provides an example of such a program. The code contains three main parts: Lines 1 to 11 define the database to classify its data; Lines 13-17 define the classification problem; Lines 19 to 26 do the actual classification and outputs the results. See below for more details on each of these sections.

Lines 1 to 11 define the database to explore. For this, a new RelationalDatabase object is created (Line 11). To create a RelationalDatabase object the user provides parameters that direct the tool how to connect to the database. The parameters are: the database type (at this stage, it must be “PostgreSQL”), the database host, the database port, the database name, and the corresponding user-name and password.

Lines 13-17 define the classification problem. This is done by creating a new DataStoreClassifier object. The DataStoreClassifier gets in its constructor the list of classifiers that are relevant to the problem. In our example, the relevant categories are "Primary Key", "Uniqueness", and "Personal Info".

Finally, in lines 19-26, the actual classification is done. First, the command dsClassifier.classify(datastore) classifies the data. Then, the example code filters the results such that only categories with confidence score above 0.5 remained. Last, in line 26, the code outputs the data classification (for each column in the database, the categories it belongs to and the corresponding confidence score).

Figure 3 displays the classification result. Lines 1 to 6 start with a column name (e.g., birthPlace_birthCity) followed by the column categories. Each category is attached with a score between 0.0 and 1.0. This score provides an indication on how confident the tool is that the column indeed belongs to the specific category. The last line in the output displays all the columns that have not been classified. The displayed categories may contain categories that are not directly asked by the user. These additional categories provide additional information on the column and are the results of dependency between categories. For example, a column belongs to the "Personal Info" category if the column is classified as either "Birth Date", "City", "Ip", "Email", or "Italian Social Id". A column belongs to "Birth Date" category if the column is also classified as "Birth" and "Date". For example, see the column "Identity_birthDate " in Figure 3 which is classified as "Birth Date" because it belongs to the categories "Birth" and "Date".

The rest of this section describes in more details the API classes.
2.2.1 DataStore

The Data Sensitivity Analysis tool classifies a DataStore object. A DataStore object represents a data we would like to classify. A RelationalDatabase object is a DataStore which represents a data located in a relational database. The constructor of RelationalDatabase required DatabaseId and Credentials as explained in Sections 2.2.1.1 and 2.2.1.2.

Figure 4 provides the corresponding class diagram.
2.2.1.1 DatabaseId
The DatabaseId object contains the information required to connect to the database.

**Constructor**

DatabaseId(String dbms, String host, int port, String name)

**Parameters:**
- dbms – a database type. This should be PostgreSQL.
- host – a name of the host. E.g., localhost.
- port – a database port number.
- name – a database name.

2.2.1.2 Credentials
Credentials describe database’s user and password.

**Constructor**

Credentials(String username, String password)

**Parameters:**
- username – the database’s username.
- password – the corresponding password.

2.2.1.3 RelationalDatabase
A class which represents a Relational Database. The Data Sensitivity Analysis tool classifies RelationalDatabase columns.

**Constructor**

RelationalDatabase(DatabaseId dbId, Credentials credentials)

**Parameters:**
- dbId – the database type and location.
- credentials – user name and its password.
2.2.2 Classifying a DataStore

The class DataStoreClassifier enables the classification of a data in a relational database. As illustrated in Figure 5, it receives in its constructor a list of DataClassifiers the user is interesting in. The method classify(dataStore) classifies the data in the DataStore and returns a DataStoreClassification. The DataStoreClassification contains the categories of each column. As described previously, each category has a confidence score. The user can filter the classification result by providing a threshold to the DataStoreClassification filter(...) method.

![DataStore classification process diagram]

**2.2.2.1 DataClassifier**

The DataClassifier classifies a column in the database (i.e., it checks if a specific column belongs to a specific category). The Data Sensitivity Analysis tool contains a library of classifiers. The library includes the following classifiers:

- CityColumnClassifier – City classifier
- DateColumnClassifier – Date classifier
- EmailColumnClassifier – Email Classifier
- IpColumnClassifier – Ip Classifier
- BirthColumnClassifier – Birth Classifier
- IsrealSocialIdColumnClassifier – IsrealSocialId classifier
- ItalianSocialIdColumnClassifier – ItalianSocialId classifier
- UsSsnColumnClassifier – SocialSecurityNumber classifier
- BirthDateColumnClassifier – Birth Date classifier. A column is classified as Birth Date if whether both Birth and Date.
- PersonallInfoColumnClassifier – Personal Info classifier. A column is classified as Personl info whether the column is classified as either Birth Date, Email, Ip, or City.
- PrimaryKeyClassifier – Primary Key classifier. A Column is classified as a PrimaryKey if it is part of its table’s primary key. A Primary Key column may contain identification data.
• UniquenessClassifier – Uniqueness classifier. A column belongs to the Uniqueness category whether all its instances are different from each other. A Uniqueness column may contain identification data.

2.2.2.2 ColumnClassifierFactory

The ColumnClassifierFactory class eases the process of identifying and creating a new DataClassifier and prevents generation of multiple instances of the same DataClassifier. It contains a list of get methods, each returns a specific DataClassifier. For example, the method getBirthDateColumnClassifier returns an instance of the BirthDateColumnClassifier.

2.2.2.3 CompositionClassifier

CompositionClassifier is a classifier which composes other classifiers to decide if a column belongs to a specific category or not. This classifier contains category, Boolean expression, and confidence score expression. If the Boolean expression is evaluated to true, then the column belongs to the classifier’s category. The Boolean expression contains Boolean flags and Boolean operators. The Boolean flags represents categories. A category flag is evaluated to true if the column under test belongs to the category. For example, in Equation 1 the category flags are IsraelSocialId, ItalianSocialId and SocialSecurityNumber. IsraelSocialId flag is true if the column is classified as IsraelSocialId. ItalianSocialId flag is true if the column is classified as ItalianSocialId, and SocialSecurityNumber flag is true if the column is classified as SocialSecurityNumber. If one of these flags is true, then the corresponding column is classified as SocialId.

\[ \text{IsraelSocialId or ItalianSocialId or SocialSecurityNumber} \rightarrow \text{SocialId} \]

Equation 1: SocialId Expression

When the Boolean expression is evaluated to true, the classifier evaluates the confidence score expression. The score expression variables are category names. Each variable is a double number between 0.0 to 1.0 which represents the category’s confidence score. For instance, in the above example the confidence score expression is:

\[ \text{max(IsraelSocialId, ItalianSocialId, SocialSecurityNumber)} \]

Constructor

ClassificationRule(String category,  
Set<DataClassifier> classifiers,  
String expr,  
String confidenceExpr)

Parameters:

- category – The name of the resulting classification.
- classifiers – List of DataClassifier the classifier depends on.
- expr – The Boolean expression. The expression is defined using the Java Expression Language (JEXL), and contains Boolean flags and operators. The Boolean flags are category names.
- confidenceExpr – The confidence score expression. The expression is defined using the Java Expression Language (JEXL). In addition, it supports the methods in org.apache.commons.lang3.math.NumberUtils using the namespace NumberUtils.
To ease the process of creating new CompositionClassifier, the tool provides a set of primitive CompositionClassifier. Currently, the tool supports two primitives: OrClassifier and AndClassifier as described in Table 1.

### Table 1 - CompositionClassifier library

<table>
<thead>
<tr>
<th>Primitive CompositionClassifier</th>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrClassifier</td>
<td><code>OrClassifier( String category, Set&lt;DataClassifier&gt; classifiers)</code></td>
<td>Execute underline classifiers (classifiers parameter). Returns the specified category (category parameter) with the highest score produced by the underline classifiers. score.</td>
</tr>
<tr>
<td>AndClassifier</td>
<td><code>AndClassifier( String category, Set&lt;DataClassifier&gt; classifiers)</code></td>
<td>Execute underline classifiers (classifiers parameter). Returns the specified category (category parameter) if all classifier returned scores are above 0. The returned score is the average of the underline classifier scores.</td>
</tr>
</tbody>
</table>

#### 2.2.2.4 DataStoreClassifier

The DataStoreClassifier classifies a data in a database.

**Constructor**

`DataStoreClassifier(Set<DataClassifier> classifiers)`

**Parameters:**

- Classifiers – A list of classifiers to run on the data. For additional information on DataClassifier see Section 2.2.2.1.

**classify**

`DataStoreClassification classify(DataStore ds)`

Classifies the data in a DataStore.

**Parameters:**

- ds – The DataStore to classify.

**Returns:** A classified DataStore in DataStoreClassification object. For additional information on DataStoreClassification, see Section 2.2.2.5.

#### 2.2.2.5 DataStoreClassification

DataStoreClassification contains the classification of a DataStore. It supports the following methods:
filter

Filters the data store classification.

DataStoreClassification filter(double threshold)

Removes categories with confidence score below the threshold value.

**Parameters:**

- threshold – The threshold value. A double number between 0.0 to 1.0.

**Returns:** A filtered DataStoreClassification. Any category that its confidence score is below the threshold is removed.

**toString**

String toString()

**Returns:** A string representing the DataStoreClassification. The string provides for each column in the database its categories and the corresponding confidence score.
2.2.3 Creating a PostgreSQL Database from csv file

The Data Sensitivity Analysis tool provides a utility to load a csv file into PostgreSQL database.

Figure 6 demonstrates how to load a csv file into the PostgreSQL database. In lines 1 to 6, it creates a new CsvInfo object. The CsvInfo object contains the information required to create the relational database table. In lines 8 to 18, the code defines a new PostgreSqlDatabase. Lines 9 to 16 declare the database connection information. In line 21, the table is created and uploaded. For additional information on each of these classes, see sections 2.2.3.1, 2.2.3.2, and 2.2.3.3.

```java
1 // Defining the CSV
2 CsvInfo csvInfo = new CsvInfo("example_csv", "examples/example_csv.csv",
3     new ColumnInfo[]{
4         new ColumnInfo("Name", "VARCHAR(255)"),
5         new ColumnInfo("Id", "INTEGER", true),
6         new ColumnInfo("Grade", "INTEGER")});
7
8 // Defining the database to connection
9 String host = "localhost";
10 int port = 5432;
11 String name = "exampleDb";
12
13 DatabaseId dbId = new DatabaseId("PostgreSQL", host, port, name);
14 String userName = "admin";
15 String password = "admin";
16 Credentials credentials = new Credentials(userName, password);
17
18 PostgreSqlDatabase myDatabase = new PostgreSqlDatabase(credentials, dbId);
19
20 // create the table, and upload the csv
21 myDatabase.uploadCSV("simple", csvInfo, true, true);
```

Figure 6: Loading csv file code example

2.2.3.1 ColumnInfo

ColumnInfo represents a single column in the database.

Constructors

ColumnInfo(String name, String type)

Creates a new Column Info. The column is not part of the table’s primary key.

Parameters:

- name – a column name.
- type – a column type. The possible values are INTEGER, VARCHAR(max-length), CHAR(length).

ColumnInfo(String name, String type, boolean isPrimaryKey)
Parameters:

- name – a column name.
- type – a column type. Possible values are INTEGER, VARCHAR(max-length), CHAR(length).
- isPrimaryKey – if true, the column is part of the table’s primary key. Otherwise, it is not.

2.2.3.2 CsvInfo

CsvInfo object represents a csv table.

Constructor

CsvInfo(String name, String path, ColumnInfo[] columns)

Parameters:

- name – a table name.
- path – a path to the csv file
- columns – an array that contains the information on all the table columns.

2.2.3.3 PostgreSqlDatabase

PostgreSqlDatabase represents a PostgreSql database. It enables loading a csv file into a new PostgreSQL table.

Constructor

PostgreSqlDatabase(Credentials credentials, DatabaseId databaseId)

Parameters:

- credentials – a database user and password. For additional information, see Section 2.2.1.2.
- databaseId – database connection information. For additional information, see Section 2.2.1.1.

Methods

String uploadCSV(String tableName, CsvInfo csvInfo, boolean header, boolean isTemporary)

Creates and uploads a csv file into PostgreSQL database.

Parameters:

- tableName – a name of the table to create.
- csvInfo – information on the csv file to load.
- header – true if the csv file contains a header line.
- isTemporary – true if the table should be considered as a temporary table.

Returns: The table name. If temporary is true, the created table name is tmp_tableName.
APPENDIX A: Tool Design Changes

During the prototype development, we have made changes in the original design. In this section we describe the major changes between the original design as we described in D5.2 and the current design. For each such change, we include the reasoning behind the change.

1. In the original design, the user defines the list of classifiers that are relevant to the problem. Based on this list, the tool classifies each column and decides whether the column is sensitive or not. As part of the prototype development, we found that it is impractical to declare automatically whether a column is sensitive. This is because sensitivity is domain specific. Therefore, in the new design, sensitivity is defined by the user who adds a corresponding sensitivity classifier. This new understanding caused changes in the classification problem definitions, and the classification result. The new APIs are described in Section 2.2.2.

2. Changes in the DataStore package:
   a. Renamed `RelationalDBDataIdentifier` class to `Column`. The `RelationalDBDataIdentifier` represented a column in relational database table. `Column` is a more meaningful name.
   b. Moved functionality related to specific column from `RelationalDatabase` to the `Column` class. For example, the method which checks if a specific column is part of the table’s primary key was moved to `Column`. 