Cortana for Dummies

Marvin Meeng, Wouter Duivesteijn

November 20, 2012

1 What is Cortana, and what does it do?

Cortana is a tool which, given a dataset, performs Supervised Descriptive Local Pattern Mining:

Local Pattern Mining: a run of Cortana strives to find *subsets* of the dataset where *something interesting* is going on. We try to pinpoint local exceptionalities in your dataset.

Descriptive: subsets delivered by Cortana are not just any subsets of the data, but *coherent* subsets, defined on attributes. Such subsets are called *subgroups.*

Supervised: the interestingness is measured with respect to a user-defined *target concept*: a subset of attributes that we are particularly interested in. Cortana finds subgroups for which the target concept is substantially different than the target concept over the entire dataset.

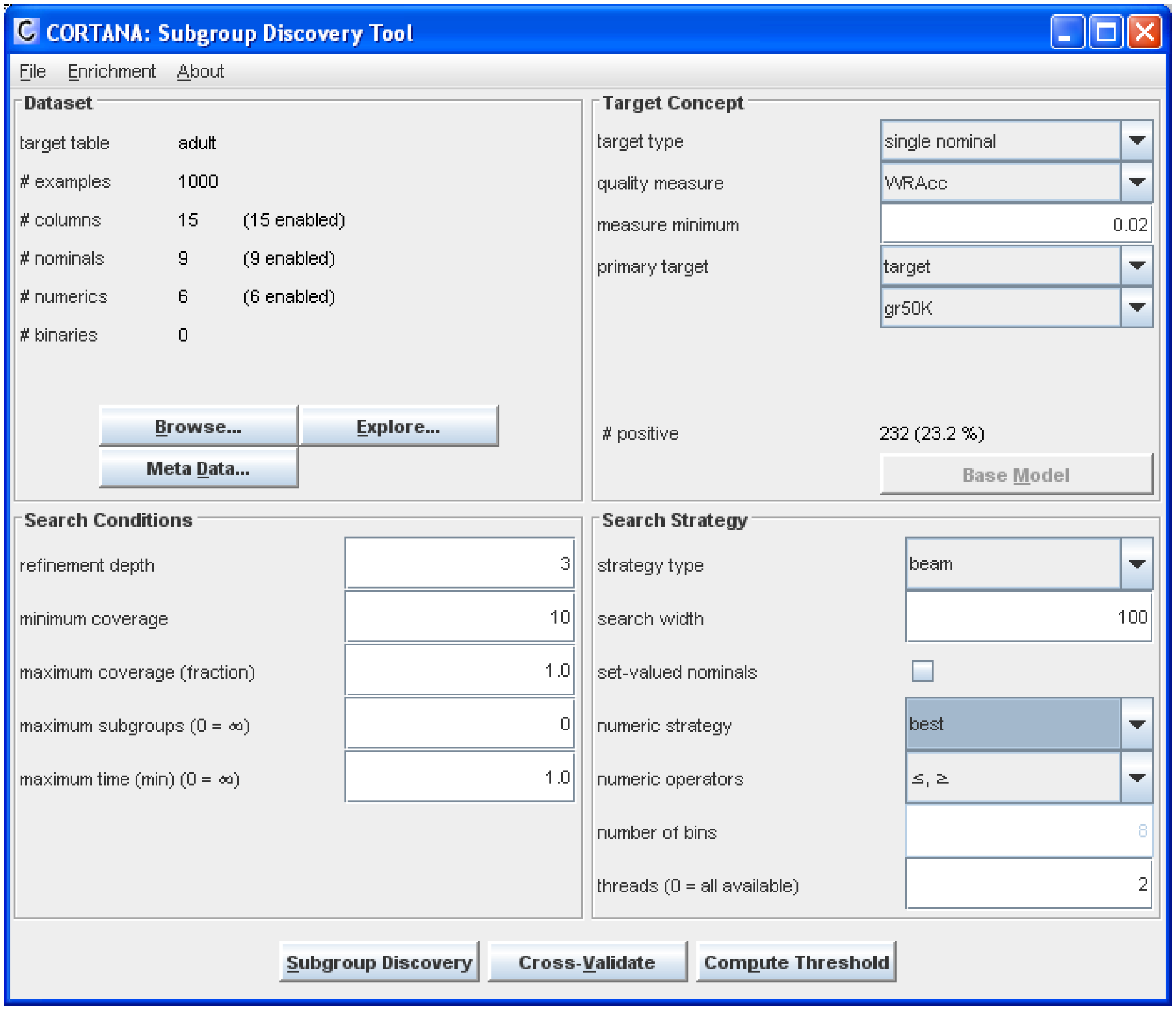
As an example, suppose a dataset about people, and let the target concept be the distribution of just one binary attribute: whether the person develops lung cancer or not. Cortana can find subgroups like *smoking* = *true* (the smokers, which have a substantially higher incidence of lung cancer), and *weekly walking kilometers* ≥ 50 (athletes, which have a substantially lower incidence of lung cancer). [TODO: might need a better example here]

2 Launching Cortana

After obtaining and unpacking a copy of Cortana from `http://datamining.liacs.nl/cortana.html`, navigate to the Cortana directory where you will find a cortana.jar file. One could start Cortana by double clicking the jar. However, it is recommended to start Cortana from the command line, since this will feed back information information about the Subgroup Discovery process to the user. To start Cortana this way, open a terminal or command window, navigate to the Cortana directory containing cortana.jar, and type: *java -jar cortana.jar*. Alternatively, one can use either cortana.bat (for Windows) or cortana.sh (Bash shell script). Be sure to read Appendix C if your computer displays memory issues while running Cortana.

After starting Cortana you are prompted for the file containing the dataset to be analysed, after which Cortana's main screen is shown.

Cortana can handle two kinds of files: ARFF files and plain (comma seperated) text files. An ARFF file combines the data with definitions of the type of each attribute. For text files, Cortana wil try to automatically infer these types. Unfortunately, this may not always deliver the desired outcome. We will discuss how to correct wrong assessments in Section 3.1.3, but in order to prevent the problem, we recommend the use of ARFF files when this option is available.



Dataset

target table	adult	
# examples	1000	
# columns	15	(15 enabled)
# nominals	9	(9 enabled)
# numerics	6	(6 enabled)
# binaries	0	

Target Concept

target type	single nominal
quality measure	WRAcc
measure minimum	0.02
primary target	target
	gr50K
# positive	232 (23.2 %)

Search Conditions

refinement depth	3
minimum coverage	10
maximum coverage (fraction)	1.0
maximum subgroups (0 = ∞)	0
maximum time (min) (0 = ∞)	1.0

Search Strategy

strategy type	beam
search width	100
set-valued nominals	☐
numeric strategy	best
numeric operators	≤, ≥
number of bins	8
threads (0 = all available)	2

Figure 1: Cortana's main window.

3 Main Screen

The main screen of Cortana is devided into four major panels, *Dataset*, *Target Concept*, *Search Conditions*, and *Search Strategy*.

3.1 Dataset Panel

The dataset panel gives some information about the dataset that is currently loaded:

target table shows the name of the data file used, which for text files is just the filename, and for ARFF files is the name defined in the '@relation' field;

# examples shows the number of examples in the dataset;

# columns shows the number of columns in the dataset. Remember that columns are also be referred to as attributes.

Finally, there is a number of fields that indicate the number of attributes from each type: nominal, numeric, and binary. Notice that Cortana cannot handle ordinal data; such attributes can be handled as either nominal or numeric, depending on what kind of subgroups you would want to define on the attribute.

In addition to the fields described above, there are three buttons present on the **Dataset** panel, **Browse. . .** , **Meta Data. . .** and **Explore. . .** .

3.1.1 Browse Window

Clicking the Browse button will open a Browse Window, displaying the data in its current state. You can sort the data by clicking on the column head of an attribute, and save the current state of the data to a file, which is useful if you have made modifications via the Meta Data Window (see Section 3.1.3).

3.1.2 Explore Window

Clicking the Explore button will open an Explore Window, which allows you to examine the distribution of single attributes, and cross-examine two attributes. This section of Cortana is independent of the Local Pattern Mining for which Cortana was developed; it exists merely to allow you to learn more about the makeup of your data.

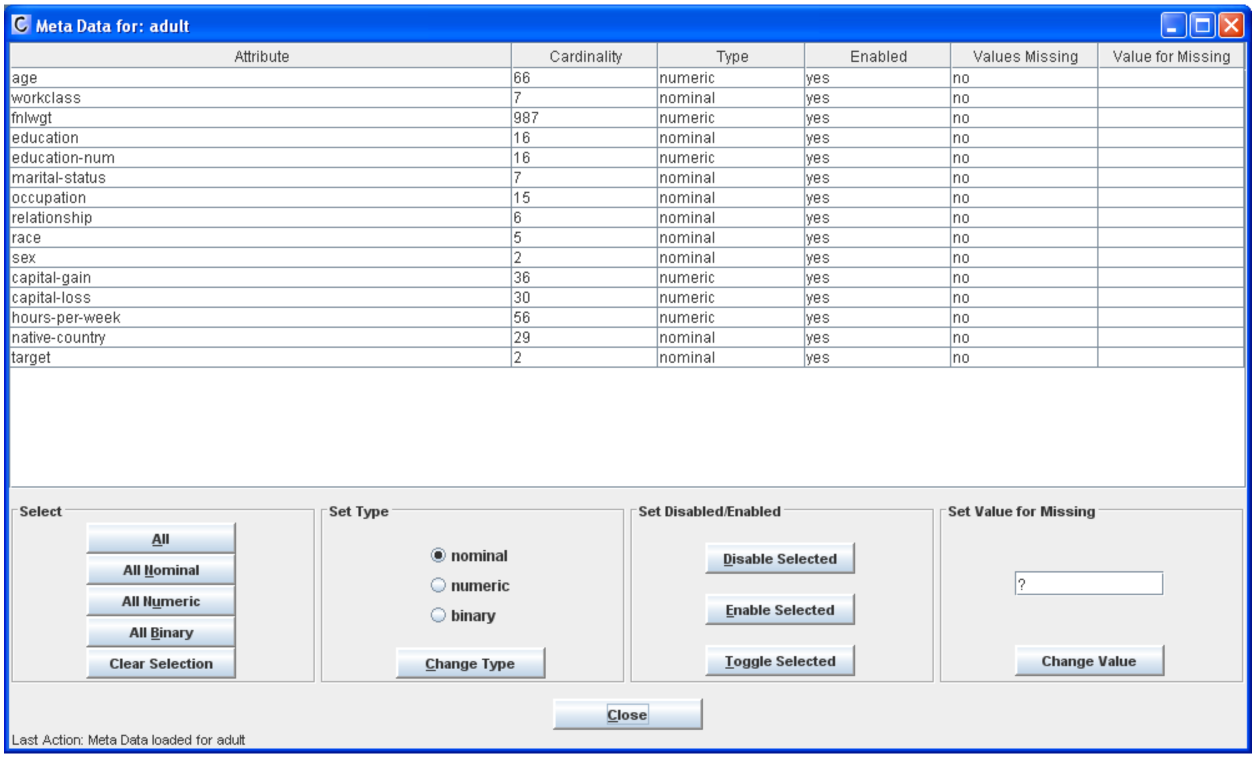
3.1.3 Meta Data Window

The **Meta Data...** button gives access to a new window that, next to displaying some additional information about the dataset loaded, also allows changing some of the (characteristics of) the data. The upper part of the *Meta Data Window* shows a table with six columns. The lower part contains a number of panels that allow modification of the data as it is in memory, note that no modifications are made to the original data file. First the properties shown in the table in the upper part will be described, the purpose of the various data manipulations will be explained after that.

Meta Data Table The first column in this table, *Attribute*, lists all *attribute names* of the attributes in the dataset. The remaining columns show some information for each of these attributes. *Cardiality* gives the number of distinct values for an attribute. *Type* shows its attribute type. *Enabled* indicates whether the attribute is *enabled* or *disabled*. *Values Missing* indicates whether the attribute contains missing values or not. And finally, *Missing Value* shows the value that is currently used for missing values in the data. Obviously, this field is blank if there are no values missing for the attribute.

Meta Data Functions The panels in the lower part of the *Meta Data Window* all allow selecting or changing the data. The first, *Select*, allows selecting all attributes of a certain type. This is a conveiniance method to be used in combination with the functionalities available in other panels.

Set Type allows changing the attribute type. This can be useful for various reasons. The first is that, after loading a plain text file, it is observed that Cortana was not capable to infer the correct type for an attribute. Related to this is the possiblity to change the type of an attribute to allow other quality measures to be used in the Subgroup Discovery process. An example of this would be an attribute that describes the number of doors in a car dataset. If this value is used as a target value, one might treat it as a *nominal* property, forcing the Subgroup Discovery process to only perform equality tests on this *attribute value* for the creation of the conditions used to form subgroups. *Instances* in the dataset are then either in the target



Attribute	Cardinality	Type	Enabled	Values Missing	Value for Missing
age	66	numeric	yes	no	
workclass	7	nominal	yes	no	
fnlwgt	987	numeric	yes	no	
education	16	nominal	yes	no	
education-num	16	numeric	yes	no	
marital-status	7	nominal	yes	no	
occupation	15	nominal	yes	no	
relationship	6	nominal	yes	no	
race	5	nominal	yes	no	
sex	2	nominal	yes	no	
capital-gain	36	numeric	yes	no	
capital-loss	30	numeric	yes	no	
hours-per-week	56	numeric	yes	no	
native-country	29	nominal	yes	no	
target	2	nominal	yes	no	

Set Type: [✓ nominal] numeric binary

Set Value for Missing: ?

Last Action: Meta Data loaded for adult

Figure 2: Meta data window.

set if they have the same value for the 'doors' attribute as the selected target value, or are in the complement of the set formed by those instances. If the 'doors' attribute is treated as *numeric*, any, combination, of the '<=', '>=' and '=' tests can used to create conditions to perform on the *attribute values*. This means that the size of the set of instances selected using an *attribute value* might be bigger than in the *nominal* case. A condition using $doors >= 2$ will select all cars having two or more doors. In the *nominal* case it would not be possible to select this group using only one condition (assuming the set of cars having more than two doors is not empty). Obviously, it would be possible to select the same group using a set of conditions like $doorsequals`2' \vee doorsequals`3' \ldots$, but, among other negative characteristics, creating such conditions would be computationally more demanding, and less intuitive. Note that if there are missing values for an attribute, the *missing value* value for this attribute might be automaticaly changed to a value that is relevant to the attribute type. See *Set Value for Missing* below for more on the *missing value* values for the different attribute types.

Set Disabled/Enabled allows to disable or enable an attribute. When an attribute is disabled, it will not be considered by the mining algorithm to form conditions with to create subgroups. Note that disabling an attribute does not affect the possiblity to select it as a target concept (see section **??** for more on target concepts).

Set Value for Missing can be used to change the value that is curently used for values that were missing in the data. The value that is used for missing values depends on the type of the attribute. If, in an ARFF file, values are declared missing, using the '*?*' directive, Cortana's file loader might replace this value with one that makes more sense in its Subgroup Discovery setting. For *nominal* types it will leave this value as is. This will result in '*?*' being one of the possible target values one can select for the corresponding attribute. However, one can assign a diffent value to the *missing values*. One than has two options, either assign the *missing values* a value that is an existing one for the attribute, or a non-exiting one. In the first case one effectively assigns all instances that have a missing value for the corresponding attribute to one of the other *attribute values*. In the latter case, one just changes the value. When changing the *missing value* value of an attribute, the *Cardinality* column is updated accordingly. For *numeric* and *binary* attribute types Cortana's file loader will replace '*?*' values with 0.0 and $false$, respectively. Again, if this is incorrect, or one wishes to assign the *missing values* another *attribute value*, either existing or non-existing, this can be done analogously to the *nominal* case.

3.2 Target Concept Panel

3.3 Search Conditions Panel

3.4 Search Strategy Panel

4 Result Window

Appendices

A Glossary

binary An attribute that can have only two distinct values. These values will be represented in Cortana with 0 and 1. Cortana will form subgroups attr=0 and attr=1..

nominal An attribute that can have any countable number of distinct values. No ordering is assumed between these values. Cortana will form subgroups attr=value, and attr$\in$ {value1,value2,...} when the set-valued nominals checkbox in the Search Strategy Panel is selected (see Section 3.4)..

numeric An attribute that can have values from a continuous range. Cortana will form subgroups attr$\leq$value and attr$\geq$value. You can control how many and which values will be selected in the Search Strategy Panel (see Section 3.4)..

ordinal An attribute that can have any countable number of distinct values. A natural ordering exists between these values. Cortana cannot handle this attribute type. Depending on which kind of subgroups you would want to consider, you can make Cortana handle such an attribute as either nominal or numeric..

type Determines the range of values an attribute can have. In Cortana, an attribute can have types binary, nominal, or numeric. Attributes in Cortana cannot be ordinal; for these attributes you can either select the nominal or numeric type, depending on what kind of subgroups you would like to see..

B Autorun

C Memory Issues

Although Cortana is written in Java, and therefore platform independent, it will behave slightly different on different operation systems and/or platforms. These differences arise from small variations in the Java Virtual Machines, used in different situations. The main issue is with 32-bit operating systems (OS). On such systems the maximum amount of memory the Java Virtual Machine (JVM) can use is around 1600 MegaBytes. However, the actual amount depends on the amount of RAM available. The cortana.bat and cortana.sh file included in the Cortana.zip set the maximum amount of memory the JVM can use to 1600 MegaBytes, through the -Xmx option. The value should, at most, be set to half the amount of available RAM, meaning eg. for a 2GB machine to *-Xmx1000m*. For 64-bit OSes no such limit exists, and it should be save to remove the -Xmx. Note that the above means that, especially for 32-bit OSes, not all datasets will fit into memory.