Cortana Manual

August 15, 2012

1 Before you start

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. This is a problem particularly relevant to the bioinformatics setting (see section 9.2), as some of the background sources used for enrichment are relatively large.

2 Preliminaries

Before giving a detailed description of the functionalities of *Cortana*, it is beneficial to first clarify some of the terms used in the rest of this manual.

2.1 Dataset

First there is the data that will be used as input. The data can often best be visualised as a table, or matrix, containing *columns* and *rows*. The first *row* of the data, the *header*, contains all the names of the columns. The remaining rows form the so-called *instances*, and describe, for each *instance*, the values for each *column*. Different authors use different names for the same concepts. To be clear, a *dataset* describes all data under investigation, and is referred to by some authors as *population*. The members that constitute a *dataset* are referred to as *instances* or *examples*. Columns will be referred to as *attributes*, and the *column name* is the *attribute name*.

Currently, two types of files can be loaded into *Cortana*, *arff* files and plain (comma separated) text files. Both file types provide the data in a matrix form. *arff* (*Attribute-Relation File Format*) files, however, additionally define the data type, or *attribute type*, of each *attribute* in a section preceding the actual data. *arff* files are very common in data mining. More on them can be found at `http://www.cs.waikato.ac.nz/~ml/weka/arff.html`.

2.2 Attribute Type

Each *attribute* has an *attribute type* that indicates what kind of data is associated with the *attribute*. Currently, *Cortana* distinguishes three *attribute types*: *nominal*, *numeric* and *binary*. This is less than the number of *attribute types* that can be declared in an *arff* file. Some of the valid '*arff* types' are mapped to a valid *Cortana attribute type*. For instance, the '*arff* types' *numeric*, *real* and *integer* are all handled by *Cortana* as the *numeric attribute type*. An *attribute* with the *nominal attribute type* is said to be a nominal *attribute*, and likewise for the other *attribute types*.

2.3 Subgroup

A *subgroup* is a selected number of *instances* from the *dataset*. As returned by *Cortana*, it will both be non-empty, and will not contain all *instances* in the *dataset*. Any *Subgroup Discovery* process will try to find *subgroups* that, on a characteristic of interest, show a statistically unusual deviation from the whole *dataset*.

2.4 Subgroup Discovery

Cortana is a generic *Subgroup Discovery* tool. However, it differs from other such tools in some respects. Traditionally, *Subgroup Discovey* could be described as a heuristic search process in which one tries to optimise for a single *target attribute*, or *target value*, using some *quality measure*. A *target attribute*, or *target* for short, should be understood as one of the *attributes* available in the *dataset*, and the *target value*, then, is one of the valid values for that *attribute*.

Note that for *nominal targets* the *target value* is contained in the data as one of its values. For a *numeric target* however, one does not set a *target value* in *Cortana*, and the underlying *Subgroup Discovery* algorithm will, within the range determined for that *target*, infer its own boundaries to create *subgroups*.

2.5 Condition

A *condition* is a description, or the intension, of a *subgroup*, while the actual *instances* that form a *subgroup* constitute its extension. The simplest, or atomic, *condition* consists of three parts: an *attribute*, an *operator* and a value. More complex *conditions* can be formed from conjunctions of atomic *conditions*. *Conjunctions* formed this way are sometimes also referred to as *condition lists*. An

example of an atomic *condition* would be: $age \leq 18$, a more complex one would be $age \leq 18 \wedge length \leq 1.8$.

2.6 Quality Measure

In *Subgroup Discovery*, one uses a *quality measure* to asses the quality of the *subgroups* found. Here, *quality* should be understood as a score obtained by appying the *qualitiy measure* to, the characteristics of, the *subgroup*. It depends on the actual *quality measure*, and/or the goal the researcher has in mind, whether such a *quality* should evaluate high or low. For example, if the very simple *quality measure* '*average*' is used, it depends on the domain specific goal whether one would try to obtain a very high or low score. *Cortana* features more than 40 *quality measures*, most based on statistical tests. This manual currently does not provide a description of them, and the user is referred to any statistical handbook, or relevant papers from the data mining community, e.g. [1, 2].

2.7 Target Attribute vs. Target Concept

Traditionally, *Subgroup Discovery* focussed only on optimising for a single *target*, or *target value*. Obviously, this is a setting that can still be used in *Cortana*. However, since *Cortana* can also be used in a novel setting referred to as *Exceptional Model Mining* (*EMM*) [3, 4], the object of optimisation need no longer be singular, and hence non-plural terms like *target* and *target value* are no longer applicable. Therefore the term *target concept* is introduced. In better describes what is the object of the optimisation effort used in the *Subgroup Discovery* process. There are a number of *target types* related to *target concepts*. Section *Target Type* 2.8 describes them in more detail, including their influence on the *Subgroup Discovery* algorithm used by *Cortana*.

2.8 Target Type

2.8.1 Single Nominal

The first *target type* is *single nominal*. It may very well be the best known, and most used, *target type* in the *Subgroup Discovery* discipline. In this setting, the *target concept* is simply one of the values of the *target attribute*, more commonly known as *target value*. For most *quality measures*, *Cortana* will optimise the quality of *subgroups* by considering the proportion of positive and negative examples in the *dataset*. *Instances* count as positive examples if their value for the *target attribute* is identical to the *target concept*, they are negative otherwise. Note that this yields a binary distinction, such that, for any *target attribute*, only that value that coincides with the *target concept* is considered positive, all other values, including *missing values*, are considered negative.

2.8.2 Single Numeric

The second *target type* is *single numeric*. This is another simple *target type*, however, it substantially differs from the *single nominal* type. Most importantly, one does set a single *target attribute*, but one does not set a *target value* when using this setting.

This *target type* can only be used for *numeric attributes*. All values of such *attributes* are numeric, and thus define a range. The goal of the *Subgroup Discovery* process will then, most often, be to find a certain threshold value, within that range, that best separates the whole *dataset* into a *subgroup* and its complement, such that the distribution, on the *target attribute*, of that *subgroup* differs most from the distribution of the whole *dataset*. Here the *quality measure* plays an important role, since it determines whether *Cortana* should try to find *subgroups* that have a much higher or lower average value for the *target attribute* than the average value for the whole *dataset*. The simple *quality measure* '*average*' would try to miximise this value, while '*inverse average*' would try to minimise it.

As alluded to above, in this *target type* setting, threshold values are inferred to form *subgroups*. For a *numeric attribute*, *Cortana* will test various threshold values to set apart *instances*, constituting a new *subgroup*, from the rest of the *dataset*. Instead of an (in)equality test, as used in the *single nominal* setting, *Cortana* uses can use, a combination of, different tests ('<=', '>=' and '=') to form *subgroups* in the *single numeric* setting. So, for a *numeric attribute*, a threshold value is selected, and then, for each *instance* in the *dataset*, it is determined whether this *instance* is to be regarded as a positive or a negative example, based on the value of that *instance* and the actual test performed. Eg., when using the '<=' test, an *instance* counts as a positive example if its value is less than, or equal to, the selected threshold, and negative otherwise. Obviously, in case of the '>=' test an *instance* is regarded a positive example if its value is greater than, or equal to, the selected threshold, and considered a negative example otherwise.

The final point to address for this *target type* is to clarifying how *Cortana* actually determines the thresholds it uses. Unfortunately, there is no short answer to this question, as it depends on the actual *search strategy* selected by the end user. However, for all three *best numeric search strategy* types, *Cortana* selects as threshold a value from the data range of the *numeric attribute* under consideration. For more details on *search strategy*, see section 4.4.

2.8.3 Double Correlation

This *target type* is used in one of *Cortana's* implementations of possible *Exceptional Model Mining* extentions of traditional *Subgroup Discovery*. More on *Exceptional Model Mining* in general, and some of its possible implementations, can be found in Leman, Feelders, Knobbe (2008) [3], and the ideas presented there will not be reiterated in this manual. However, as the purpose of this manual is to provide some useful, or necessary, information about, parts of, *Cortana*,

a little will be said about this *target type*, its use, and the way it influences the *Subgroup Discovery* process used by *Cortana* to form candidate *subgroups*.

2.8.4 Multi-label

3 Starting Cortana

After obtaining and unpacking a copy of *Cortana* from `http://datamining.liacs.nl/cortana.html`, navigate to the Cortana directory. There 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. This way, a lot more information about the *Subgroup Discovery* process is fed back 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 Section 1 *Before you start* if you do.

After starting *Cortana* one is asked to select the file that contains the dataset to be analysed, after which *Cortana*'s main screen is shown.

Currently, two types of files can be loaded into *Cortana*, *arff* files and plain (comma seperated) text files. The first is very common in data mining and describes the data and additionally defines the *attribute type* of each *attribute*. As such, *arff* files are preferred over plain text files. For text files, *Cortana* wil try to infer the correct attribute type of each attribute. Unfortunately, this may fail, so to check whether Cortana was able to infer the correct attribute type, or change the attribute type anyway, the *Meta Data...* button on the main screen gives access to the *Meta Data Window* (section 6).

4 Cortana - Main Screen

The main screen of *Cortana* is devided into four major panels, *Dataset*, *Target Concept*, *Search Conditions*, and *Search Strategy*. The following subsections will address them all.

4.1 Dataset

As can be expected from the name, this 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 example 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 data type. The type of an attribute determines the sort of mining

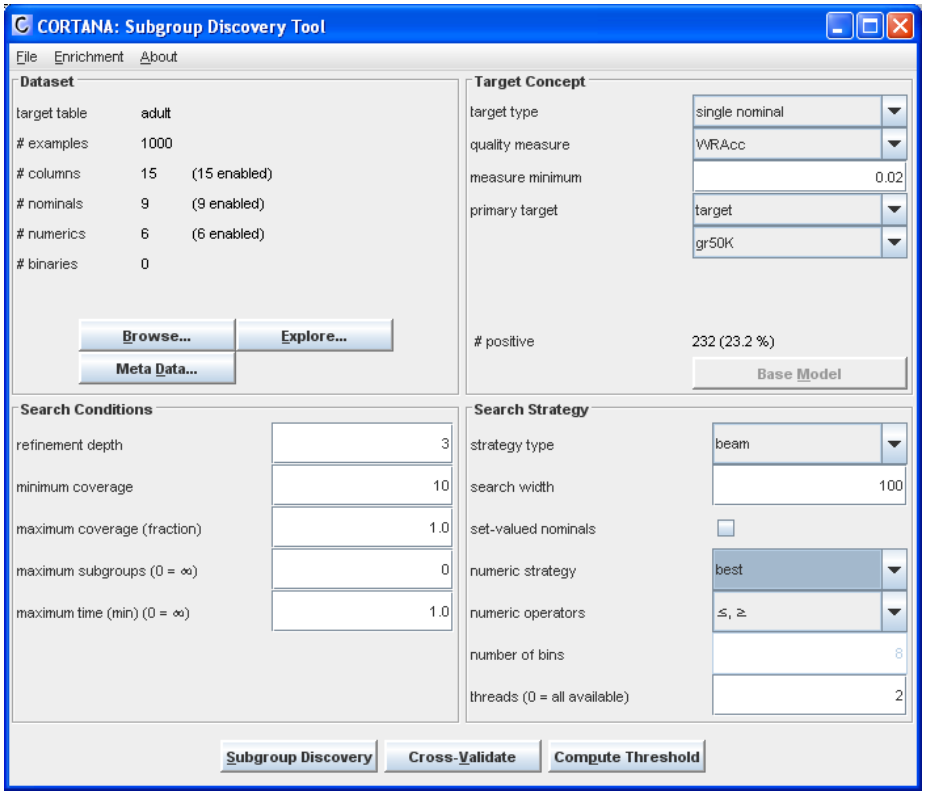

Figure 1: Cortana's main window.

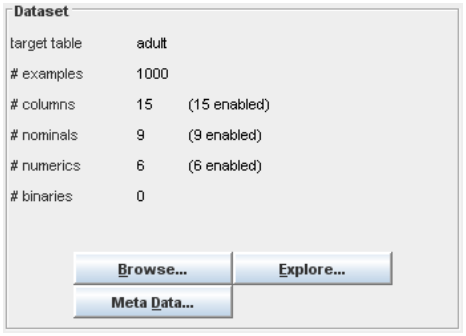

Figure 2: Dataset details in the main window.

algorithm or quality measure that is applicable to it. More about this can be found in section 2.2.

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

By clicking the **Browse...** button, a **Browse Window** (section 5) is presented, showing a table with the data in the state that it is currently in. Additionally, in the table header, it shows the number of distinct values for each attribute. Note that the data may not be in the same state as when it was loaded, as it can be modified using functionalities of the **Meta Data Window**, presented after pressing the **Meta Data...** button, which is also present on the **Dataset** panel. Section 6 describes the Meta Data Window, its components, and the data manipulation functionalities, in more detail. To be able to explain them more thoroughly though, it is benefical to describe the various fields of the **Target Concept** panel first.

Explore button.

4.2 Target Concept

The fields of this panel serve to manipulate the target concept-related search settings used during the Subgroup Discovery process. Traditionally, Subgroup Discovery focussed only on optimising for a single target or target value. Obviously, this is a setting that can still be used in Cortana. However, since Cortana can also be used in a novel setting referred to as *Exceptional Model Mining* (*EMM*) [3], the term target value is no longer applicable, hence the term *Target Concept*. Section 2.8 describes the various target types in more detail, including its relation to the type of search strategy (section 4.4) adopted by the Subgroup Discovery algorithm used by Cortana. In this section only a short description of the various fields will be given.

The first field is **target type**, that, obviously, is used to change the target type for which one wants to do the Subgroup Discovery. Selecting another target type from the drop down box will force Cortana to consider other attributes as

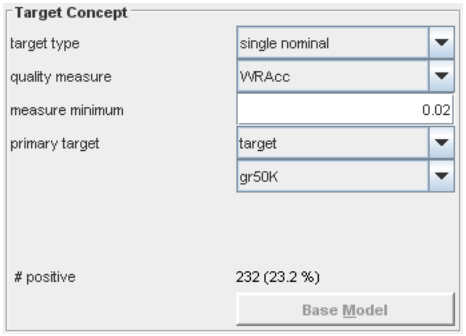

Figure 3: The target concept specification details in the main window.

target concept. For example, when changing the target type from the default single nominal to single numeric, the first attribute having the numeric attribute type is selected, and subsequently displayed in the **primary target** drop down box (see below).

Based on the target type selected, a number of *quality measures* is available. These are listed in the **quality measure** drop down box. After changing the *target type*, the *quality measures* listed in the *quility measure* drop down box are automatically updated, to fit the new *target type*.

The text field next to *measure minimum* will, by default, show a minimum threshold value for the selected *quality measure*, that, based on historical observations, is considered to be sensible. One might consider lowering this value if a *Subgroup Discovery* experiment, run with the default value, did not return any, or only few, *subgroups*. On the other hand, one can increase this value to force *Cortana* to only report *subgroups* that have a quality, as calculated by the selected *quality measure*, that lies above this threshold. Also, it is worth mentioning that if the search depth is greater than one, leading to conjunctions of *conditions* to be used to select *instances* from the *dataset* to form *subgroups*, the number of intermediate results, when going from one search depth to a higher one, can be lower when using a higher threshold. This, naturally, aleviates the computational burden of the *Subgroup Discovery* process. However, the recommended way of controling the amount of (intermediate) results, is through the *maximum subgroups* setting in *Search Conditions*. More on search depth, or *refinement depth*, and *maximum subgroups* can be found in section 4.3 below.

The items described in the remainder of this section will not be available in every *target type* setting, as the various *target types* are related to the *attribute type* of the *target concept*. For each item, it will be mentioned to which *attribute type* it applies.

primary target is available in the following *target type* settings: *single nominal*, *single numeric* and *double correlation*. The *primary target* drop down box lists all *attributes* available in the *dataset*.

target value is available only in the *single nominal* setting. The *target*

value drop down box lists all values of the *attribute* set as *primary target*, which can be of the *nominal* and *binary attribute type*.

secondary target is available only in the *double correlation* setting. In this setting only *attributes* of the *numeric attribute type* will be listed in both the *primary target* and *secondary target* drop down box.

targets and settings is available only in the *multi-label* setting. If this *target type* is selected, the *Targets and Settings* button will be enabled, and this gives access to the *Multi-label details*. The upper part of which shows some input fields related to *multi-label* search settings. The lower part lists all *attributes* of the *binary attribute type*. More on the *multi-label* can be found in section 7 and in [3, 4].

Then, there is an information field that shows information relevant to the selected *target type*. The number of positives, *# positives* is shown for the *single nominal target type*, *average* for *single numeric*, *correlation* for *double correlation* and *# binary targets* for *multi-label*.

The **Base Model** button will only be enabled if the *target type* is *double correlation* or *multi-label*. It gives access to a *Base Model Window*, showing a correlation plot for the selected *primary target* and *secondary target*, if the *target type* is *double correlation*. For the *multi-label target type*, it will show a *bayesian network*, connecting the various *binary attributes*.

4.3 Search Conditions

The fields on this panel allow setting the *search conditions* used in the *Subgroup Discovery* process. For all fields the default values are also given.

refinement depth controls the number of *conditions* that is used to create *subgroups*. A *refinement depth* of 1 would lead to *conditions* like $[x <= 9.11]$, while a *refinement depth* of 2 would allow for the creation of *conditions* like $[x <= 1.2 \wedge y >= 3.4]$. It is not recommended to set the *refinement depth* to very high values. In *Subgroup Discovery*, often a depth greater than 4 or 5 does not lead to significant improvements of the *measure score* any more, and just increases both the risk if overfitting, and the computational time needed to calculate the result. The default value is 1.

minimum coverage sets the lower bound for the size of the *subgroups* that should be reported by *Cortana*, meaning all reported *subgroups* have at least this size. The default value is set to 10% of the total *dataset* size, which is shown as *# examples*, see section 4.1 *Dataset* above.

coverage fraction sets the upper bound for the size of the *subgroups* that should be reported by *Cortana*, meaning all reported *subgroups* have at most this size. The default value is set to 100% of the total *dataset* size, which is shown as *# examples*, see section 4.1 *Dataset* above.

maximum subgroups is used to control the maximum number of *subgroups* in the result list generated by *Cortana*. First, this means that the *Result Window* (section 8) will show at most this number of *subgroups*.

But also, as alluded to in the *Dataset* section (4.1 on *measure minimum*, this number also controls the number of intermediate results retained by *Cortana*

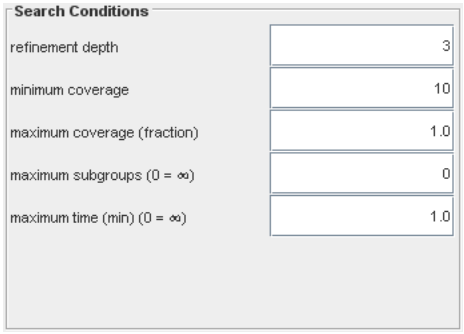

Figure 4: Specification of the searchconditions in the main window.

to form the pool out of which the generation for the next *refinement depth* is formed. As such, this *Search Condition* parameter has significant influence on the computational demands, as it directly controls the number of combinatorial candidates to be tested during *Cortana*'s *Subgroup Discovery* process. Without going into the actual algorithm, a small example probably suffices to make this clear. For a *dataset* consisting of only *nominal attributes*, assume n = *maximum subgroups*, and m = *number of attribute-value pairs*, then the number of candidates c to be tested by *Cortana* for the next *refinement depth* iteration is: $c = n \cdot m$. The default value is 50.

maximum time (min) will determine the maximum time, measured in minutes, *Cortana* is allowed to search for new *subgroups*. After this periode *Cortana* will abort the *Subgroup Discovery* process and report all *subgroups* found up until that point. This means that especially for the *search strategy* (section 4.4) *strategy type depth first* a large of the search space will not be explored at all, if the search proces is aborted. More precisely, only the first *attributes* will have been addressed at that time. The default value is 1.

4.4 Search Strategy

This is an intro. It describes *strategy type*, *best numeric* and *number of bins*.

strategy type *beam*, *cover-based beam selection*, *best first*, *depth first* and *breadth first*. The default value is *beam*.

search width This parameter can not be set for the *best first strategy type*. The default value is 100.

numeric operators The default value is <=, >=.

best numeric *bins*, *best* and *all*. The default value is *bins*.

number of bins Only available for the *best numeric* setting of *bins*. Default value is 8.

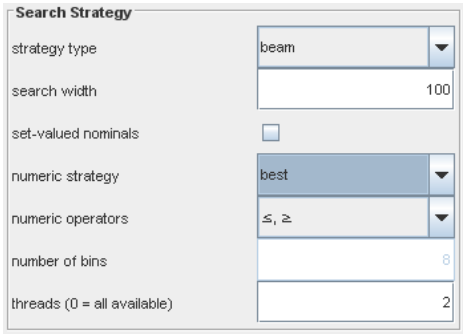

Figure 5: Search strategy specification in the main window.

5 Browse Window

6 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.

6.1 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*.

6.2 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.

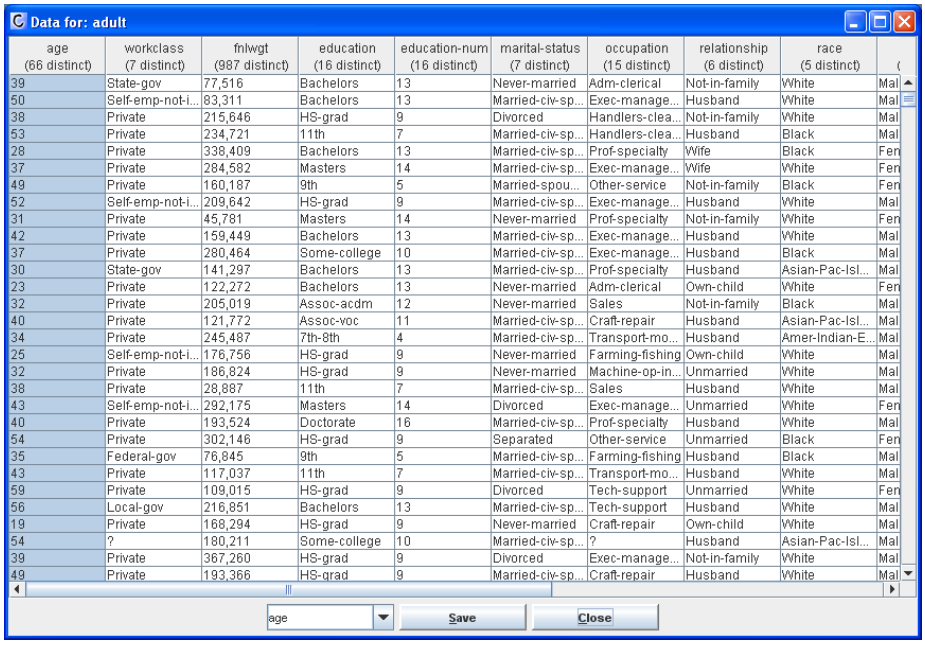

Figure 6: The browse window.

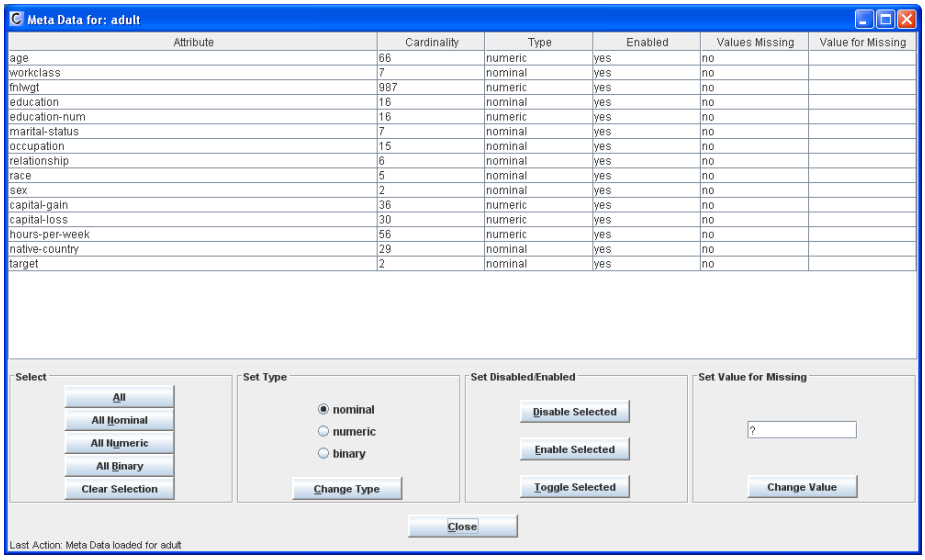

Figure 7: Meta data window.

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 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 $[doors\ equals\ `2' \vee doors\ equals\ `3' \ldots]$, 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 2.7 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,

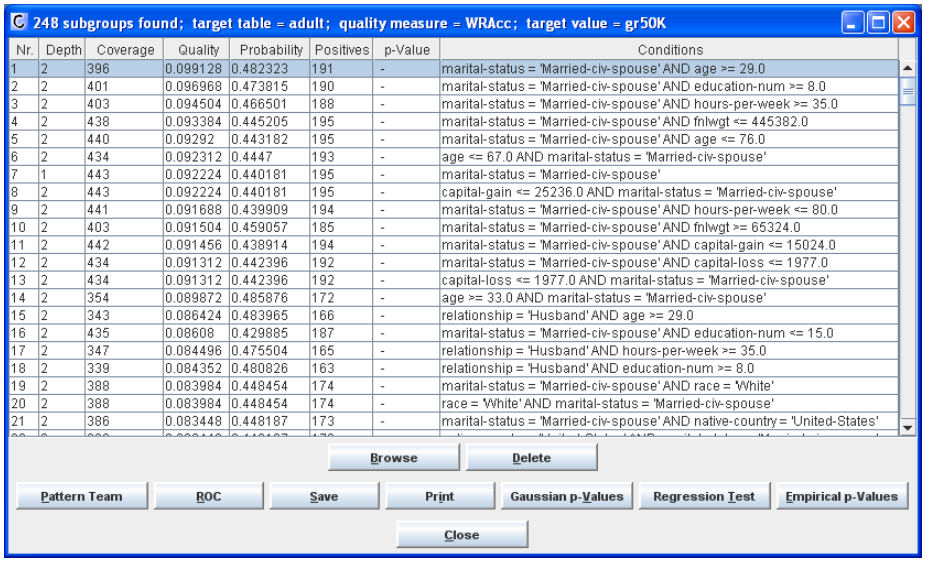

Figure 8: The result window.

this can be done analogously to the *nominal* case.

7 Multi-label

8 Result Window

When a mining run terminates, a *Result Window* is shown. The titlebar of this window shows some basic information about the search settings that were used to obtain the result displayed. The top half of a *Result Window* shows a table with the *subgroups* found, the lower half contains two rows of buttons. Some of the buttons on this *Button Panel* will be presented only in certain search settings. Also, a number of buttons is used for post-processing, others just give access to additional information. Finally, the buttons in the lower row will execute operations on the whole *Result Set*, that is all results shown in the *Result Window*, while those in the upper row operate only on the selected *subgroup(s)*.

The following subsections will go into detail about the use of the various components of the *Result Window*.

8.1 Result Set Table

The top half of the *Result Window* consists of a *Result Set Table*, showing all *subgroups* in the *result set*. Form left to right the columns in this table are:

Nr., *Depth*, *Coverage*, *Measure*, *p-Value* and *Conditions*, and they all reveil a characteristic of the *subgroup* found. This section will address all columns, and explains how to interpret the information they display.

Nr. shows the rank of the *subgroup*. Note, however, that *Cortana* does not use a partial ranking for ties, and, as a result, *subgroups* that attain the same *measure* score, will not receive equal ranks. Rather, if such an event occurs, *Cortana* assigns the value, displayed by *Nr.*, based on where in the *Subgroup Discovery* process the *subgroup* was found.

Depth shows the *refinement depth* at which the *subgroup* was found. As explained in the section on *Search Conditions* (4.3), the *refinement depth* controls the number of *conditions* that is used to create a *subgroup*. So, a *subgroup* that is formed by using only one *condition* is said to have been found at depth 1. The number displayed at *Depth* is the same as the number of conjuncts in the *Conditions* column of the *Result Set Table*.

Note that in *Subgroup Discovery* it often happens that one *condition* already selects a highly valuable, high scoring, *subgroup*. Adding extra *conditions* to this first one may improve the *measure* score, but only by a, possibly very, limited amount. This will lead to a *result set* containing a number of *subgroups* that all consist of the same 'base' *condition*, which is then refined with additional *conditions* using another *attribute*. If these extra *conditions* do not substantially increase the *measure* score, all of these variations will attain consecutive ranks in the final *result set*.

Coverage shows the number of *instances* in the *subgroup*. Depending on the *target type* used, this number can consist of both *true positives* and *false positives*, or just indicates the total size of the *subgroup*.

Measure shows the score of the *subgroup*, according to the *quality measure* used. Different *quality measures* have different ranges for the scores they asssign to the worst and best possible *subgroups*. As a result, *measure* scores resulting from applying different *quality measures* can not be compared directly in any sensible manner.

p-Value is initially blank. But it will show the p-value of the *subgroup* after additional user input. In *Subgroup Discovery*, *quality measures* are used that are based on, or insired by, well known, standard, statistical tests. However, *Cortana* does not allow calculation of p-values in a fashion that is customary in statistics. To be able to do so, the *dataset* under investigation should be considered a sample of the total population itself. But, as the *dataset* is not treated as such, the calculation of the p-value is done using a different paradigm. The major benefit of this paradigm is that it is agnostic to the *target type* used in the *Subgroup Discovery* process. Section 8.3.2 *Compute p-Value* below describes how the p-value is calculated in *Cortana*.

Conditions shows the *condition(s)* used to form a *subgroup*. As explained in section 2.5 *Condition*, a *condition* consists of an *attribute*, an operator, and a value. For depths greater than 1 *Conditions* shows a conjunction of such *conditions*.

8.2 Subgroup Buttons

The following subsections describe the various buttons that can appear on the upper row of the lower part of the *Result Window*. These buttons operate only on the *subgroup(s)* that are selected, or highlighted, in the *Result Set Table*. The selection can be changed using the mouse or pressing (Shift) and the Down/Up arrow of the keyboard. The *target type* setting used for the *Subgroup Discovery* experiment determines which buttons will be shown. For all buttons it will be indicated for what *target type* setting it is applicable.

8.2.1 Show Model

Show Model is available only for the *multi-label target type*. For all *subgroups* selected in the *Result Set Table*, it will show a new *Model Window*, depicting the bayesian network connecting the *binary attributes*, according to the model that was inferred through the *Subgroup Discovery* process.

8.2.2 Browse Subgroup

Browse Subgroup is available for all *target types* and will show a *Browse Window* as described in the *Browse Window* section (5). For all *subgroups* selected in the *Result Set Table*, it will show such a window, depicting all *instances* that constitude the *subgroup*. For the *single nominal target type* one additional button is available, the *True Positives* button. Clicking it will highlight only those *instances* that count as *true positives*.

8.2.3 Delete Subgroup

Delete Subgroup is available for all *target types* and deletes a *subgroup* from the *Result Set*. Note that this means that the *subgroup* is removed both from the visible *Result Set Table*, as from the underlying *result set*. This means that, after deletion, the *subgroup* will no longer show up in the results of the *Save* and *Print* actions described below. Also, in the *single nominal* setting, this point is no longer used for any ROC plot created after deletion. ROC plots created before deletion are unaffected. This is intended behaviour, as in this way it allows comparison of ROC plots of the same *result set* in a way that would not be possible if pre-existing ROC plots are update after any deletion.

8.3 Result Set Buttons

The following subsections describe the various buttons that can appear on the lower row of the lower part of the *Result Window*. These buttons operate on the whole *result set*, that is, on all *subgroups* in the *result set*. As such, the results of these operations are unaffected by, changes in, the selection of *subgroups* in the *Result Set Table*. The *target type* setting used for the *Subgroup Discovery* experiment determines which buttons will be shown. For all buttons it will be indicated for what *target type* setting it is applicable.

8.3.1 ROC

ROC is only available for the *single nominal target type*. All *subgroups* in the *Result Set Table* will be taken into account to show a *ROC Curve Window*. ROC (Receiver Operating Characteristic) curves have their origin in electrival engeneering, where they are used to show the performance of classifiers. In the *single nominal* setting some of the members of a *subgroup* are *true positives*, while others are *false positives*. In a ROC plot the *true positive rate* (TPR) of each *subgroup* is plotted against its *false positive rate* (FPR). When this is done, a line, the ROC curve, can be draw that forms a convex hull connecting the outer most points. A straight diagonal from the lower left corner to the upper right indicates complete randomness of the result, and the furher the line approaches the upper left corner the better the result. For a *subgroup* to reach the upper left corner, its TPR must be 1, and its FPR 0.

8.3.2 Compute p-Values

To calculate the p-values for the *subgroups* found, the end user clicks the *Compute p-Value* button on the *Button Panel*. In short the calculation of p-values is done by creating n random subgroups ...

8.3.3 Regression Test

TBI.

8.3.4 Empirical p-Values

This functionality may be removed.

8.3.5 Fold

TBI.

8.4 Save

Save is available for all *target types*. It allows saving of the whole *result set* to a file. The user will be asked for a name and location of this file. The file will contain the information as it is depicted in the *Result Set Table*, and the values for the various columns will be comma separated. Currently, there is no method of changing this separator.

8.4.1 Print

Print is available for all *target types*. It allows printing of the whole *Result Set Table*. Note that it does not have a maximum amount of result it prints, the *Result Set Table* will be printed as-is. Therefore, it is not recommended to use this functionality on large *Result Sets*/*Result Set Tables*.

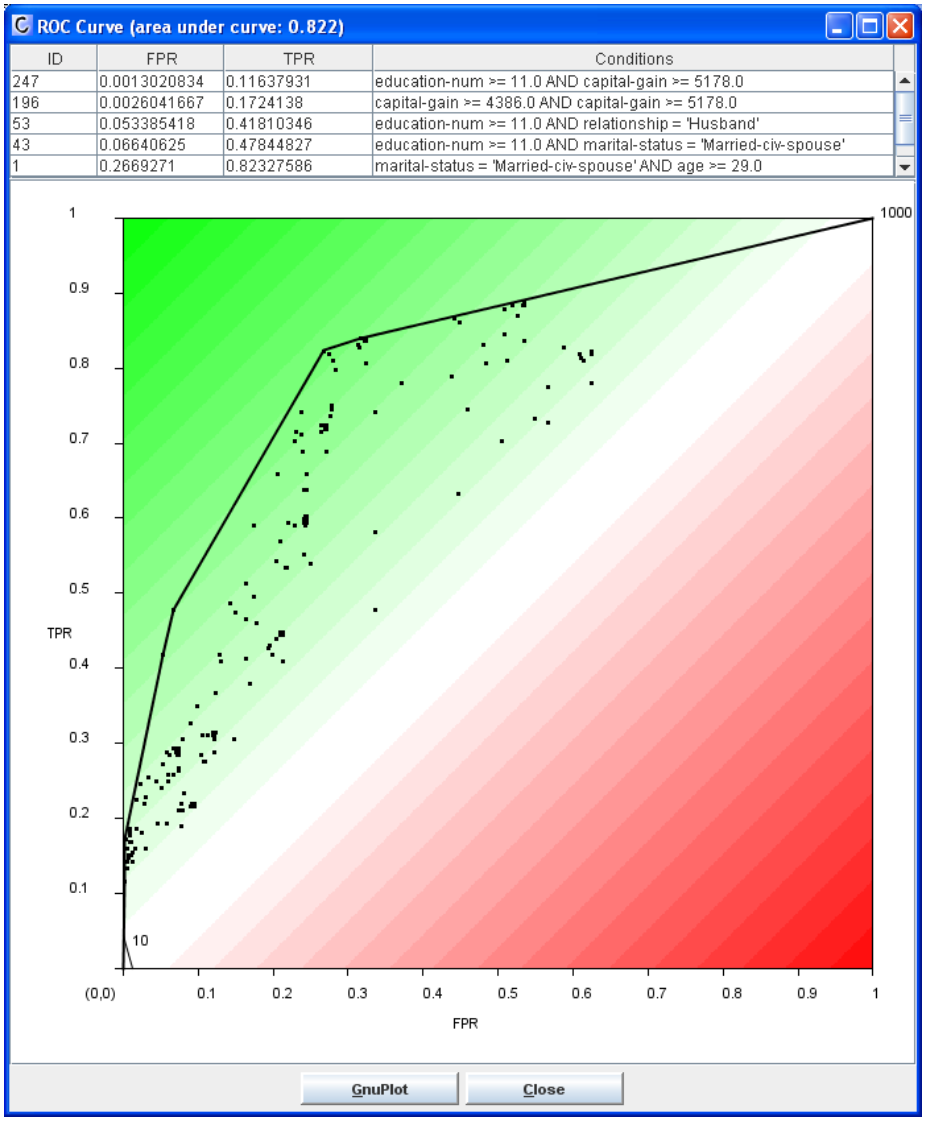

Figure 9: The ROC window.

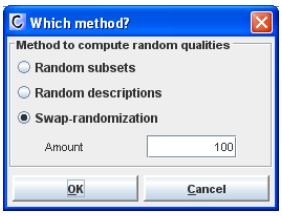

Figure 10: Choosing the type of randomization to be used for any significance testing.

8.4.2 Close

Close is available for all *target types*, and it will close the *Result Window*. Note that it can not be reopened. Therefore, all necessary actions like, saving or printing the results, should be done before closing the *Result Window*. All results will be lost, and the only way to get a new *Result Window* is to run the same *Subgroup Discovery* process again. As this may be a very lengthy process, be sure to finish all operations before closing the *Result Window*.

9 Usage

9.1 Orinary Subgroup Discovery

java -jar Cortana.jar

9.2 Bioinformatics Setting

10 Autorun

Cortana comes with the ability to run experiments in an unsupervised setting. This is a mode that is specifically created to allow so-called batch mining. This section will explain how *Cortana* accomplishes this, and how to set up an *autorun* or batch mining experiment.

In normal usage mode, the end user can set all search settings using the graphical user interface (GUI). After loading a data file and setting the search settings one usually clicks the *Subgroup Discovery* button, and the *Subgroup Discovery* process begins. However, *Cortana*'s menu bar on the top of the main window gives access to two options that allow saving the search parameters to a XML-file. This file can then be used to start the *Subgroup Discovery* process, using the defined search settings, at some other time.

The two autorun related options can be found under *File*, and are named *Create Autorun File* and *Add to Autorun File*. The first will ask the user for a name and location for the new file that will be created, containing all information, as shown in the main window, needed to do an unsupervised *Subgroup*

Discovery experiment. The second option allows to add all information currently presented by the main window to an already existing *autorun* file. By using this last option one can create a large file, containing multiple experiments. An example usage would be running several experiments on a single data file, where each experiment uses a different *quality measure*.

The paragraph above refers to information, and not solely search settings, as the XML-file used in the *autorun* setting also needs to know the name of the data file to use. That name is not part of the search settings, but it is important to state how *Cortana* looks for this data file. When loading the *autorun* XML-file, *Cortana* will remember the directory in which this *autorun* file resides. It will then parse the XML-file, and retrieve the name of the data file to use. *Cortana* will then try to load the data file from the directory in which the *autorun* resides, and that directory only. This means no relative paths can be used for the location of the data file, and the *autorun* file should always reside in the same directory as the data file(s) for which it describes experiments. Although this limits its use, it keeps it easily portable, as, in this way, one can simply exchange directories containing *autorun* file(s) and accompanying data file(s), without having to worry about setting paths correctly in the *autorun* file, or placing the directories in the correct relative location, before being able to run an experiment.

When *Cortana* runs an experiment defined in an *autorun* file, the *Subgroup Discovery* process is exactly the same as when the experiment would be started from the GUI. However, at the end of the *Subgroup Discovery* process, there are a few deviations from the normal GUI situation.

First of all, when the *Subgroup Discovery* process terminates, the *Result Set* will be automatically saved, as it would happen when the end used clicks the *Save* button in the *Result Window*. The result file will have a name indicative of the search settings used, and starts with the data file name. Also, it will contain a time stamp as part of its name, so that identical runs on, updated versions of, the data file will not overwrite older results. The result file will be saved to the directory in which the *autorun* file resides.

Second, a *Result Window* may or may not be shown. Whether it is, is controlled by a command line parameter, as shown at the end of this section. Omitting this command line parameter will cause *Cortana* to default to always showing a *Result Window* after an experiment finishes. Showing the *Result Window* is necessary if one wishes to perform additional actions on the *result set*, be it plotting *ROC Curve Windows*, calculating p-Values or any of the other actions described in section 8 *Result Window*.

Third, if the *autorun* file defined multiple experiments, the next experiment will be run automatically. This will go on until all the experiments in the *autorun* file are executed. In case of any errors, for example when *Cortana* can not locate a data file, some efforts are made to resume execution by discarding the erroneous experiment and continuing with the next.

A final, more technical, word about the *autorun* files. The preferred way to create *autorun* files is through the *Cortana*'s main GUI. In most cases this is the most convenient way, eliminating the chance of errors that could occur

when one were to craft an *autorun* file by hand. The XML-parser parsing the *autorun* file uses a very strict *autorun* definition file, and will throw errors when any of the XML tags are not present in the *autorun* file. This means that, even for XML-tags that define settings that are not relevant to the experiment for which they are defined, they should be present, although they will be so-called empty tags in such a case.

Note that some efforts were taken to allow *Cortana* to run in a headless environment, as is customary for the majority of servers. If all goes well, this should not be a problem. However, if an error occurs, it may cause an error message windows or dialogs to be shown. Obviously, showing this in a headless environment will fail, and may lead to additional errors. This is a known problem, but will not be addressed any time soon.

To use *Cortana* in *autorun* mode start it as shown below, where *file.xml* is the *autorun* file to use, and the optional *false* controls whether a *Result Window* should be shown after each experiment finishes. Remember that the default is *true*, and the second parameter can thus be omitted if one wishes to see the *Result Window*.

java -jar Cortana.jar file.xml [false]

References

[1] Author, Statistical Handbook .

[2] Lavrac, Flach, Zupan, Rule Evaluation Measures, a unifying View, .

[3] Leman, Feelders, Knobbe, *Exeptional Model Mining*, .

[4] Wouterd, Subgroup Discovery meets Bayesian Networks, an EMM approach .