**CS 484 Data Mining**

**Fall 2015**

**Professor Jessica Lin**

**HW 3 – Due 10/15**

**Total: 130 points**

**Weka**

* Download and install the Weka data mining software from <http://www.cs.waikato.ac.nz/ml/weka/>
* We will be using the Explorer component of Weka for this project.  Tutorials and user manuals are available here: <http://www.cs.waikato.ac.nz/ml/weka/> (under Documentation)
* Note that Weka assumes by default that the class attribute is the last column.

**Task 0: Introduction to WEKA (0 point)**

* Familiarize yourself with Weka by going through this tutorial: <http://research.cs.queensu.ca/home/cisc333/tutorial/Weka.html>

**Task 1: Analyzing Heart Disease Dataset (60 points total)**

**(This assignment was borrowed from the TCSS 555A Data Mining Class taught at the University of Washington, Tacoma Branch.)**

The dataset studied is the ***heart disease*** dataset from ***UCI repository***. Two different datasets are provided: ***heart-h.arff*** (Hungarian data), and ***heart-c.arff*** (Cleveland data). These datasets describe factors of heart disease. Both these data sets are available to you on the assignment page.

The ***data mining project goal*** is to better understand the risk factors for heart disease, as represented in the 14th attribute: ***num*** (<50 means no disease, and values <50-1 to <50-4 represent increasing levels of heart disease).

The ***question*** on which this machine learning study concentrates is whether it is possible to predict heart disease from the other known data about a patient. The ***data mining*** task of choice to answer this question will be classification/prediction, and several different algorithms will be used to find which one provides the best predictive power. However this exercise focuses on the various aspects of the KDD process.

**1. Data preparation- integration (5 points)**

Integrate the two datasets into one single dataset, which will be used as a starting point for the next questions, and load it in the ***Explorer***. How many instances do you have? How many attributes? (You could do this using Excel or spreadsheet programs. First, save your individual files as “csv” files in weka, Open them in a spreadsheet viewing program. Copy the rows from one file to another after verifying that the attributes match. Save the merged file (csv).

Paste a screenshot of the ***Explorer*** window.

**2. Descriptive data summarization (21 points)**

Before preprocessing the data, an important step is to get acquainted with the data – also called ***data understanding***.

1. (4 points) Stay in the ***Preprocess*** tab for now. Study for example the ***age*** attribute. What is its ***mean***? Its ***standard deviation***? Its ***min*** and ***max***?
2. (3 points) Specify which attributes are numeric, which are ordinal, and which are categorical/nominal.
3. (4 points) Interpret the graphic showing in the lower right corner of the ***Explorer***. What do the red and blue colors mean (pay attention to the pop-up messages that appear when dragging the mouse over the graphic)? What does this graphic represent?
4. (2 points) Visualize all the attributes in graphic format. Paste a screenshot.
5. (2 points) Comment on what you learn from these graphics.
6. (3 points) Switch to the ***Visualize*** tab. By selecting the maximum jitter (click “Update” to update the plots), and looking at the ***num*** column – the last one – can you determine which attributes seem to be the most linked to heart disease? Justify your answer.
7. (3 points) Does any pair of different attributes seem correlated?

**3. Data preparation – selection (2 points)**

The datasets studied have already been processed by selecting a subset of attributes relevant for the data mining project.

1. (2 points) From the documentation provided in the dataset, how many attributes were originally in these datasets? The original dataset and the documentation can be found here: [https://archive.ics.uci.edu/ml/datasets/Heart+Disease](https://archive.ics.uci.edu/ml/datasets/Heart%2BDisease)

**4. Data preparation – cleaning (7 points)**

Data cleaning deals with such defaults of real-world data as incompleteness, noise, and inconsistencies. In ***Weka***, data cleaning can be accomplished by applying ***filters*** to the data in the ***Preprocess*** tab.

**(5 points) Missing values**. List the methods seen in class for dealing with missing values, and which ***Weka*** ***filters*** implement them – if available. To see what filters are available, click on “Choose” under Filter, and then click on “Filters…” Examine the checkboxes and make sure that only the appropriate ones are checked. You can read more about each filter and set different parameters by clicking on the rectangle box next to the “Choose” button. Remove or estimate the missing values with the method of your choice, explaining which filter you are using and why you make this choice.

If a filter is not available for your method of choice, you can do one of the two things for extra credit (up to 10 additional points):

1. Develop a new one that you add to the available filters as a Java class.
2. Do this outside of Weka – write your own program or use existing packages such as Matlab.

The amount of points you get will depend on the method you choose, e.g. whether it makes sense, and whether it affects the results adversely. So don’t choose a method because it’s the “easiest” to do unless you can justify that it’s the right choice.

1. (2 points) Save the cleaned dataset into ***heart-cleaned.arff***, and paste here a screenshot showing at least the first 10 rows of this dataset – with all the columns.

**5. Data preparation – transformation (17 points)**

Among the different data transformation techniques, explore those available through the ***Weka Filters***. Stay in the ***Preprocess*** tab for now. Study the following data transformation only:

1. (3 points) ***Attribute construction*** – for example adding an attribute representing the sum of two other ones. Which ***Weka filter*** permits to do this?
2. (9 points) ***Normalize*** an attribute. Which ***Weka filter*** permits to do this? Can this filter perform Min-max normalization? Z-score normalization? Provide detailed information about how to perform these in ***Weka***.
3. (3 points) ***Normalize*** all real attributes in the dataset using the method of your choice – state which one you choose.
4. (2 points) Save the normalized dataset into ***heart-normal.arff***, and paste here a screenshot showing at least the first 10 rows of this dataset – with all the columns.

**6. Data preparation- reduction (8 points)**

Often, data mining datasets are too large to process directly. Data reduction techniques are used to preprocess the data. Once the data mining project has been successful on these reduced data, the larger dataset can be processed too.

Stay in the ***Preprocess*** tab for now. Beside attribute selection, a reduction method is to select rows from a dataset. This is called sampling. How to perform sampling with ***Weka filters***? Can it perform the two main methods: ***Simple Random Sample Without Replacement***, and ***Simple Random Sample With Replacement***? Explain how (which filter, what parameter(s) to set, etc).

**Task 2: Spam Detection**

The amount of SPAM we receive by e-mail appears to increase every day. By SPAM we mean things like:

• advertising for products or web sites

• messages with schemes to make a lot of money fast

• chain letters

• porn

• etcetera

To avoid the ﬂooding of your mailbox by SPAM, you can make a so-called SPAM-ﬁlter, that attempts to distinguish SPAM from non-SPAM messages by looking at the contents of the message. In this task we are going to use a collection of hand-labeled data to construct a SPAM-ﬁlter.

**Available data**

We have a collection of 4601 e-mail messages of which 1813 (39.4%) are SPAM. The dataset is called spambase.arff.

For each message 57 attributes have been recorded, as well as an indicator whether a message is SPAM (1) or not (0).

Most attributes indicate whether a particular word or sign occurred frequently in the message. The run-length attributes (55-57) measure the length of uninterrupted sequences of capital letters. Below a description of the attributes is given:

• 48 numeric attributes of type *word\_freq\_WORD* = percentage of words in the message that is equal to WORD, that is 100 \* (number of times WORD occurs in message) / total number of words in message.

• 6 numeric attributes of type *char\_freq\_CHAR* = percentage of characters in message equal to CHAR, that is 100 \* (number of occurrences of CHAR) / total number of characters in message.

• 1 numeric attribute called *capital\_run\_length\_average* = mean length of uninterrupted sequences of capital letters.

• 1 numeric attribute called *capital\_run\_length\_longest* = length of longest uninterrupted sequence of capital letters.

• 1 numeric attribute called capital *run\_length\_total* = total number of capital letters in the message.

• 1 binary {0,1} class attribute named *spam* = indicates whether message is (1) or is not (0) SPAM.

**2.1: Classiﬁcation Trees (40 points)**

The classiﬁcation tree (decision tree) algorithm C4.5 is implemented in Weka as J48. Go to the Classify tab, and under the “Classifier” box, click on the “Choose” button. Choose Trees -> J48. This algorithm has two important parameters, denoted by C (confidenceFactor, default value: 0.25) and M (minNumObj, default value: 2). Click on the parameter values (shown to the right of Choose) to change the parameter settings. If you click on “More” under “About,” you can learn more about what these parameters mean.

Below is a table with diﬀerent combinations of values for these parameters.

1. (15 points) Report the error rate (incorrectly classified instances) on the *training set* of trees built with these different parameter settings. Also report the size of the trees in a separate table. **Explain your ﬁndings in relation to the meaning of the parameters.**

 Error rate:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| C\M | 2 | 10 | 50 | 100 |
| 0.40 |  |  |  |  |
| 0.25 |  |  |  |  |
| 0.05 |  |  |  |  |

 Size of tree:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| C\M | 2 | 10 | 50 | 100 |
| 0.40 |  |  |  |  |
| 0.25 |  |  |  |  |
| 0.05 |  |  |  |  |

1. (20 points) Perform the same analysis, but now using 10-fold cross-validation. Remember to construct tables like above, and explain your ﬁndings in relation to the meaning of the parameters. In general, how do the error rates compared to the ones you got from using all the training data? Why do you think this is the case?

(c) (5 points) To gain some insight, we construct a simple tree on the training sample by setting M=100 (with default C). To see what the tree looks like, right-click on the result in the Result list (bottom left corner). Report the resulting tree (copy & paste the decision tree here). Does it make sense? Explain.

**2.2: Naive Bayes (30 points)**

Naive Bayes is an example of a generative classiﬁer, that is, it models the probability distribution of the attributes within each class. Since all attributes are numeric, we will use the assumption that each attribute is normally distributed within each class. This algorithm is available under the name NaiveBayes in Weka.

(a) (5 points) Use 10-fold cross-validation to estimate the error of the Naive Bayes classiﬁer. How does it compare to the decision tree classifier?

(b) (10 points) Naive Bayes simply uses all attributes, which may lead to bad performance if there are too many redundant attributes. First select a subset of the attributes using the standard settings under the Select Attributes tab. This algorithm selects attributes with a high correlation to the class label, but low correlation among themselves, so each selected attribute contributes some information about the class label that is not already contained in the other attributes. Does it help if you only use the subset of attributes selected to construct the Naive Bayes classiﬁer?

(c) (5 points) Looking at the data, do you think the normality assumption is appropriate in this case?

(d) (10 points) Instead of assuming normality, we can also ﬁrst discretize the attributes, and then apply the Naive Bayes classiﬁer for nominal data. This option is available in Weka using NaiveBayes, and then setting the parameter use SupervisedDiscretization to True. This performs a discretization of numeric attributes similar to how a classiﬁcation tree computes a split on a numeric attribute. Binary splitting may be applied recursively, so that we may end up with more than two intervals.

* Does this improve performance compared to ﬁtting normal distributions?
* Look at the naive bayes parameter estimates produced by Weka. If you add up the counts in the output, we end up with a higher count than the total number of observations in the data. How is that possible?

**Report and Submission**

* Collect output/screenshots from your experiments.  **Submit the Weka output electronically on Blackboard.**
* Write a report addressing the above questions. To conserve paper, put just the relevant answers on the report, and only put a screenshot from Weka if asked to (the rest can go in the electronic copy). Turn in a hard copy of the **report** as well as submitting the report on Blackboard.