**Project 1 – Data Preprocessing**

**CS539 Machine Learning – Fall 2014**

**Prof. Carolina Ruiz**

**Student’s Name:** Daniel Fitzgerald

**I. Data Exploration**

**Three observations of good things about the dataset (include visualizations) [6 Points]:**

* **The data has many attributes.** There are 24 dimensions plus 24 features for medication. In general, the larger the dataset, the more certain we can be that any trends we observe in it are accurate predictive models.
* **The dataset is large.** There are 82599 entries, collected over 10 years. This was a very large study. In addition, there is a large variation of data. Almost every combination of attributes appears. This is good because if some combination doesn’t appear, we can know its probability is very low, but we can’t know how low beyond some point. If it appears at-least once, then we can estimate its probability as we do all other combinations.
* **The data contains many nominal attributes.** Nominal attributes are good for making predictive models because they lend themselves to classification algorithms such as decision trees and clustering. Continuous numerical values must be discretized for such categorical methods to apply.

**Three observations of bad things about the dataset (include visualizations) [6 Points]:**

* **The data is sparse.** Many of the values are missing and it was collected from disparate unstandardized sources. Some of the numeric and nominal values are based on somewhat arbitrary thresholds determined by the scientists. For example, the 30 day readmission limit was “was chosen based on criteria often used by funding agencies.” 30 days has no special significance in the context of diabetes diagnosis, testing, or treatment.
* **The data is noisy.** As with all real-world data, there are not only missing values, but there are likely mistakes or random variations in some of the data. For example, “race” is not a clear-cut category. For example, some patients could be Asian-Hispanic and they would have had to choose one of those races or the “other” option. Some doctors may have multiple medical specialties, or experience in areas outside their formal specialty.
* **The data contains irrelevant features.** Payer code, for example, has no bearing on any medical procedures, diagnosis, etc. This is not a problem in practice because irrelevant data can easily be identified and ignored by a human interpreter (data scientist)

**List of all attributes that you would remove from the dataset right away and why [3 Points]**

* **Weight:** “considered to be too sparse and it was not included in further analysis” (Strack *et al.*)
* **Payer Code**: “had a high percentage of missing values and it was not considered relevant to the outcome” (Strack *et al.*)
* **Patient Number**: Like payer code, these are arbitrary numbers that have no relationship to medical conditions, diagnoses, treatment, etc.
* **Encounter ID**: Like payer code, these are arbitrary numbers that have no relationship to medical conditions, diagnoses, treatment, etc.

**II.1. Data Preprocessing: Discretization**

1. **Using Weka:**
2. **Description of the supervised discretization results [2 Points]**

Applying the supervised discretization filter separates the *num\_lab\_procedures* attribute in to three bins of three ranges (-inf, 24.5], (24.5,40.5], and (40.5,inf) using the default parameters. Using the Kononenko criterion produces the same ranges. If *­makeBinary*  is *True* then *num\_lab\_procedures­*  will be replaced by two binary attributes, *­num\_lab\_procedures\_*1 and *num\_lab\_procedures\_*2 and all other numeric attributes will be likewise replaced by numbered copies. If *useBinNumbers* is set to *True* the ranges will be replaced by named bins of the format ”B*i*of*N*” where *I* is the ordered bin number and *N* is the total number of bins.

**Description of the Java code implementing the supervised discretization filter [5 Points]**

The java code for this filter contains implementations of the algorithms described in *Multi-interval Discretization of Continuous-Valued Attributes for Classification Learning* (Fayyad, 1993) and *On Biases in Estimating Multi-Valued Attributes* (Kononenko, 1995). The user can choose which of these two criterion to use for discretization. The default Fayyad algorithm applies a heuristic for Minimum Descriptive Length (MDL) for minimizing the entropy (information gain) of the data in intervals it produces. The Kononenjo *RELIEFF* function is similar but includes a bias. Both algorithms choose “split” points for the intervals in a very similar way to how the branching attributes of binary decision trees are chosen – they are chosen such that the most the most additional information can be inferred with the minimum additional information known. The filter function takes a list of data columns to discretize as input, as well as several flags specifying how to treat the data, and produces a list of splitpoints (doubles) specifying the intervals to discretize the data in to.

1. **Description of the unsupervised discretization results [2 Points]**

Applying the supervised discretization filter separates *num\_lab\_procedures* into ten equally wide bins, each of width 13.1, starting with the first (-inf, 14.1] and the last bin (118.9, inf). Changing the number of bins makes the size of the bins proportionally smaller because they are equal-width. If *­findNumBins* is *True*, the number of bins is automatically optimized to 9. If *useEqualFrequency* is *True* then the bins will have the same size (number of data that fall in each bin) but different (integer) widths. *­findNumBins* cannot be used in conjunction with *useEqualFrequency*.

**Description of the Java code implementing the unsupervised discretization filter [3 Points]**

The java code for this filter essentially divides the range of the given data column into *N* equal parts or “bins” by default. If *useEqualFrequency* is True then it will divide the total samples in the column by N and increment the size of each sequential bin until it contains that many (non-missing) samples. Like the supervised filter, the unsupervised filter function takes a list of data columns to discretize as input, as well as several flags specifying how to treat the data, and produces a list of splitpoints (doubles) specifying the intervals to discretize the data in to (bins).

1. **Using R:**
2. **Description of the R functions used for discretization [5 Points]**

To convert a numeric attribute to a nominal attribute (“factor” in R), one approach is

>myFactors <- factor(levels=myNumericData, nmax=N, ordered=TRUE)

Where the argument myNumericData is a list of numeric values and *N* is the maximum number of bins to use. Missing values are excluded by default.

Another function is cut(x, …) which divides the range of x into intervals and assigns corresponding codes (bin numbers) to the values. Because cut can also take a list of breakpoints, it can also be used to approximate equal-size binning by using percentiles as break/splitpoints as follows

>datafact <- cut(mydata, quantile(data, (0:N)/N)

The above functions are suitable for unsupervised binning (constant width or size bins). No functions for supervised discretization, such as the MDL or RELIEFF algorithm implementations in WEKA, were found in the standard R language. However, several packages are available, including the deprecated dprep (http://cran.r-project.org/web/packages/dprep), discretization (http://cran.r-project.org/web/packages/discretization/), and infotheo (http://cran.r-project.org/web/packages/infotheo/index.html).

1. **Description of the results obtained with these discretization functions [3 Points]**

Discretizing the num\_lab\_procedures column into 10 equal-width bins using the cut function:

>equalBinWidth <- cut(diabetes\_data$num\_lab\_procedures, breaks = 10, ordered\_result=TRUE)

This produces a factor with 10 levels, named after the intervals which they correspond to: (0.869,14.1]" "(14.1,27.2]" "(27.2,40.3]" "(40.3,53.4]" "(53.4,66.5]" "(66.5,79.6]" "(79.6,92.7]" "(92.7,106]" "(106,119]" "(119,132]"

Discretizing num\_lab\_procedures into 10 equally-sized bins using the cut function:

>equalBinSize <- cut(diabetes\_data$num\_lab\_procedures, quantile(diabetes\_data$num\_lab\_procedures, (0:10)/10))

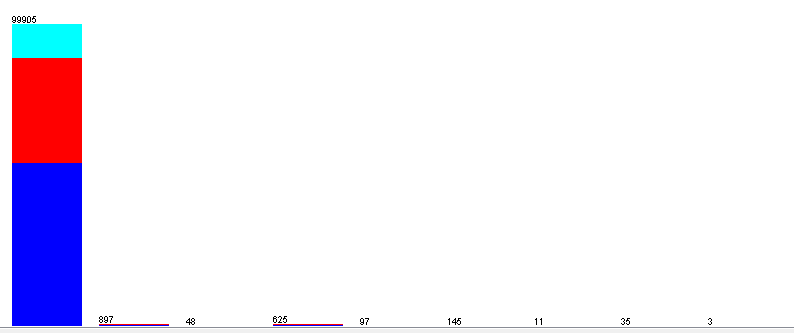
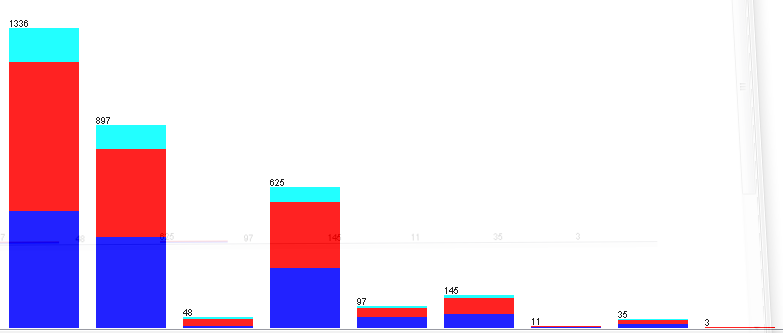
This results in a factor of ten levels, where each label is the interval it corresponded to in the numeric data: (1,14], 14,27], (27,35], (35,40] (40,44] (44,49] (49,54] (54,60] ... (67,132]

**II. 2. Data Preprocessing: Missing Values.**

1. **Using Weka: Description of the results of replacing missing values [3 Points]**

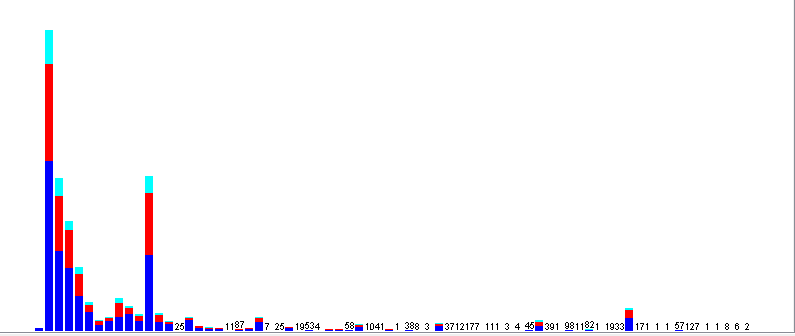
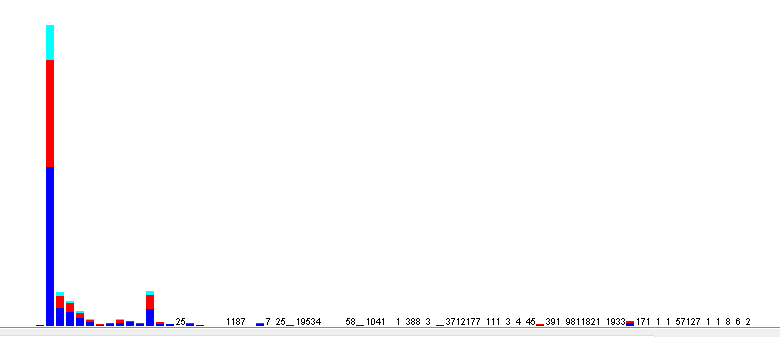
Applying the *ReplaceMissingValues* filter to a numeric attribute replaces all missing values with the mean of that attribute. Applying it to a nominal attribute replaces missing values with the mode. For example, the mode of the **weight** attribute is “[75, 100)”, so applying the filter causes all missing values (which is most values for this attribute) to be replaced with “[75, 100)”, causing this value to dominate.

Weights before filter Weight attribute after filter



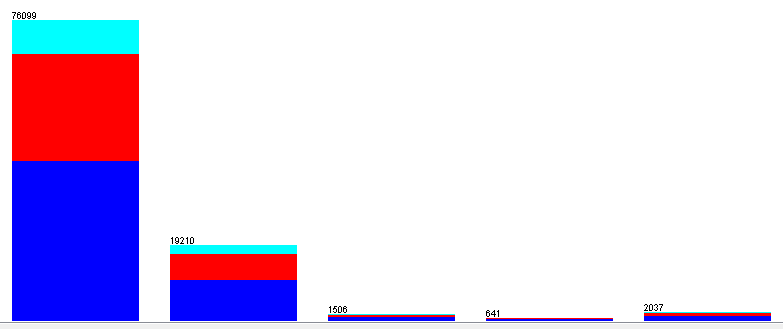
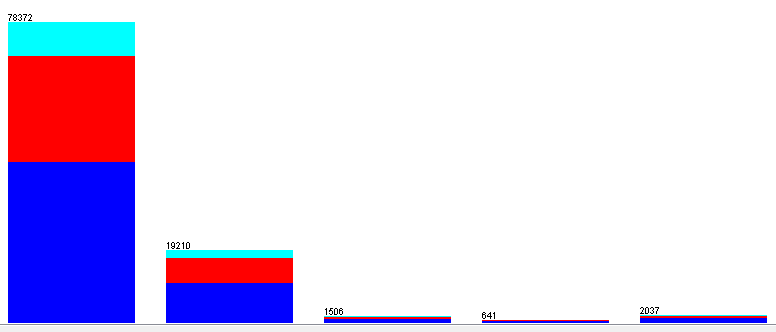
Applying the filter to the **medical\_specialty** attribute has a similar effect by dramatically increasing the frequency of “InternalMedicine”:

Medical Specialty before filter Medical Specialty after filter

Applying the filter to **race** does not have as much of an effect because the **race** attribute was not as sparse as the others.

Race before filter Race after filter

1. **Using R:**
2. **Description of the R functions used for replacing missing values [4 Points]**

The same effects can be achieved in R by:

Finding the non-missing factor attributes

> notMissing <- diabetes\_data$myAttribute[(diabetes\_data$ myAttribute != "?")]

Calculating the mode of those levels

> attributeMode <- names(sort(-table(notMissing)))[1] # find the first element in a table of the levels sorted by their descending frequency (the most common level is the mode.)

Replacing undefined levels with that mode

> diabetes\_data$myAttribute[diabetes\_data$myAttribute == "?"] <- attributeMode

Where myAttribute is the attribute with missing values to replace.

1. **Description of the results obtained with these functions [3 Points]**

This method was applied to the **weight**, **medical\_specialty**, and **race** attributes with the same results as in WEKA, as expected.

Replacing missing **weight** values is not advised. This attribute is too sparse to be statistically useful. There are so many missing values that when they are all replaced with the mode of the few existing values, the resulting distribution is unrealistic and does not likely represent the actual weight ranges of the patients in the data. Similarly, replacing missing **mdeicalSpecialty** values results in an unrealistic proportion of InternalMedacine doctors. Replacing missing **race** entries, however, may be acceptable, since there are only a few missing values and the distribution is wide enough that replacing these missing values with the mode does not affect the resulting distribution as dramatically.

**II.3 Data Preprocessing: Attribute/Feature Selection**

1. **Using R.**
2. **Include either the Correlation Matrix, a visualization of it, or both. [4 Points]**

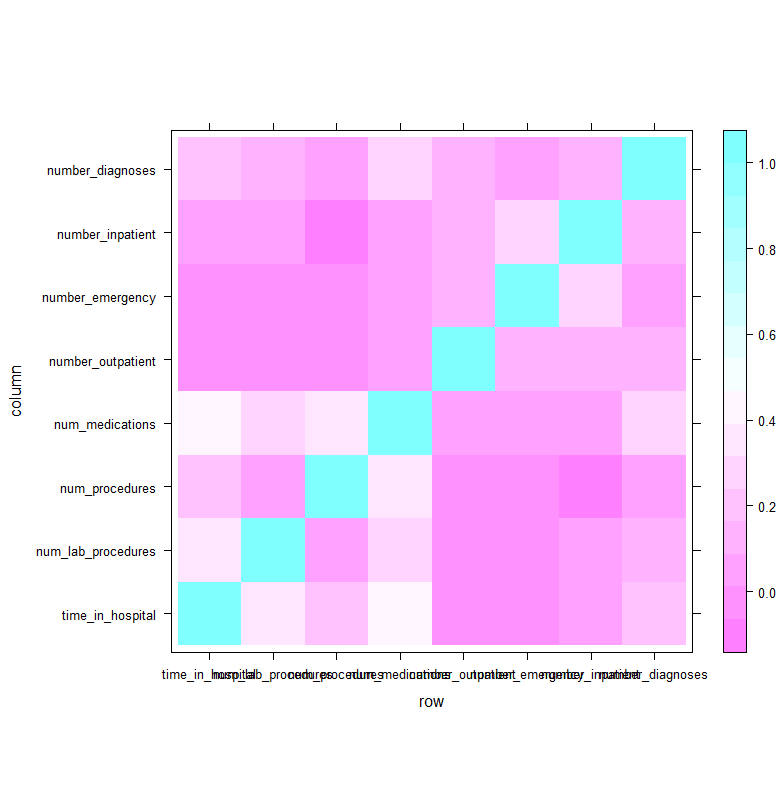
After ensuring that each attribute is treated as the correct type (numeric, nominal), the correlation matrix for the continuous attributes is calculated as:

> cormatrix <- cor(diabetes\_data[sapply(diabetes\_data, is.numeric)])

This is visualized using the *lattice* library

> library(lattice)

> levelplot(cormatrix)



1. **Which 3 attributes would you remove based on the correlation matrix and why? [1 Points]**

In *Table 1: List of features and their descriptions*, the attributes **encounter ID**, and **Patient number** are listed as numeric. Although they are numbers, they are arbitrary, and should have no correlation to any other data. These were converted to nominal attributes before generating the above correlation matrix. Based on the matrix, the numerical attribute **number\_diagnoses** because it has little correlation (lightest rows/columns) with most other attributes, and has no very strong correlation with any attribute.

1. **Using Weka. Correlation Based Feature Selection.**
   1. **Result of applying CfsSubsetEval. [1 Points]**

WEKA version 3.7’s *CfsSubsetEval* evaluator with default options is not applicable to the raw data because the **diag\_1**, **diag\_2**, and **diag3** attributes are of type string, which the evaluator does not handle. These three attributes were removed before CFS feature extraction was applied. Using the full training set, the evaluator selected the attributes **patient\_nbr**, **discharge\_disposition\_id**, **number\_emergency**, and **number\_inpatient.**

**Explain in your own words what property this subset of attributes satisfies. [3 Points]**

The selected attributes are those that the evaluator identified at the most predictive of the data overall. That is, knowledge of them will be more informative and increase the predictive accuracy of a statistical model more than knowledge of any other attributes.

* 1. **Comparison of results with your answers to part (a) above. [1 Points]**

Two of the selected features, **patient\_nbr**, **discharge\_disposition\_id**. It is odd that these two attributes were selected, since it is known that both are unordered arbitrary nominal values. They already been ignored from the computation of the correlation matrix because they should not have any actual correlation to the rest of the data. The other two selected attributes, **number\_emergency**, and **number\_inpatient** had strong correlations to multiple other attributes.

1. **Using R. Description of Feature Selection functions in R [6 Points]**

**and results of using them on the dataset [4 Points] (may continue on next page)**

**II.4. Data Preprocessing: Attribute/Feature Extraction. AT MOST 2 PAGES FOR II.3 AND II.4 COMBINED**

1. **Using Weka. Principal Components Analysis (PCA) Results and Discussion [4 Points]**

PCA in WEKA produces the five principle components:

0.396change+0.356diabetesMed-0.283insulin=No-0.229metformin=No+0.21 metformin=Steady

**Using R. Principal Components Analysis (PCA) Results and Discussion [4 Points]**

**Specify what functions you used in R. [4 Points]**

The PCA in R using the *princomp* function applied to the numeric attributes of the data results in 8 variables with standard deviations of:

Comp.1 Comp.2 Comp.3 Comp.4

1.4252548 1.170689 1 0.9919185 0.9755235

Comp.5 Comp.6 Comp.7 Comp.8

0.9052142 0.8562160 0.8001353 0.6854577

1. **Comparison of the Weka and the R results [3 Points]**

**III. Model Construction.**

Summarize the experiments you ran in the table below. Add more table rows as needed. [24 Points]

* **What code/functions did you used to run ZeroR experiments in R? [1 Points]**

None available

* **What code/functions did you used to run OneR experiments in R? [5 Points]**

None available

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tool** | **ML**  **technique** | **Pre-**  **processing** | **Testing**  **method** | **Resulting**  **model** | **Evaluation:**  **accuracy,**  **conf. matrix** | **Observations and Analysis of your results** |
| Weka?  R? | ZeroR?  OneR? | none? |  |  |  |  |
| WEKA | ZeroR | none | Training set |  | 53.91% | inaccurate |
| WEKA | ZeroR | none | 66% split |  | 54.03% | Still inaccurate |
| WEKA | ZeroR | none | 10-fold cross validation |  | 53.91% | Still inaccurate |
| WEKA | OneR | None | Training set |  | 64.29% | Better than ZeroR |
| WEKA | OneR | None | 66% split |  | 53.04% | Worse than ZeroR |
| WEKA | OneR | None | 10-fold |  | 53.379 | About the same as ZeroR |
| WEKA | OneR | Some ignored\* | Training set |  | 56.512% | Worse than no preprocessing |
| WEKA | OneR | Some ignored\* | 66% split |  | 56.52% | Better with preprocessing |
| WEKA | OneR | Some ignored\* | 10-fold |  | 56.48% | Better with preprocessing |

**\***The **Encounter ID**, **Patient number**, **weight**, and **payer code** were ignored.

**Optional additional page to include any interesting work that you did in the project that you want to show, but didn’t have enough space to include on the previous pages.**

It is interesting that ignoring some attributes, such as patient ID, decreased the accuracy of some classifiers using certain training methods, but increased the accuracy of others or the same using another training method. Is was expected that the elimination of extraneous data would make all models more accurate. In general, the OneR classifier was slightly more accurate than ZeroR.