Supervised Learning Competition

Software Specifications and Usage

- Knime
  - How to use
    - Download the free trial from https://www.knime.com/
    - Open the exported project in the zip file.
    - Click the green play button at the top to execute the workflow. Results can be seen from the scorer node.
  - What was done
    - First step was to implement a file reader that can import the gathered .csv files.
    - Binners, scorers, normalizers, categorizers, practitioners etc. were used to pre process the data so it could be used effectively in the predictor.
    - Neural network predictor was implemented with a training and testing phase
    - Visualization
      - Interactive Table View
      - Linear Correlation node
      - Scorer Node for Confusion matrix

Exploration

The first step in solving this data prediction problem was to examine the data in its raw form to get a better understanding of it. This step was done in excel and consisted of us viewing the test and training data along with the data key. Right away it was apparent that the training data was nowhere near perfect and that there was a significant amount of data missing. On an patient to patient (row) basis the amount of missing data was seemingly random but when looking at a column to column basis, trends started to rise (i.e. some columns had no data at all). To get a better understanding of what affected a patient's re-admission to the hospital, a linear correlation was performed on the training data. This indicated that a patient's re-admittance was correlated highest to age and race. These were still relatively low correlations though as we now know that one attribute is not vastly more informative that the others. This is important to know as if there is an attribute with very high correlation its needs to be investigated to deem its validity rather than just be a dead giveaway for the predictor.

Preparation

Data preparation began with importing the data into Knime and feeding it through a column filter. This allowed us to remove columns that were not important to the prediction such as patient id and patient weight (very scarce data, not valid to fill with a missing value algorithm). Once we were left with a relevant dataset the next step was to convert all of the strings to numbers to be used in the predictor. This was done with the string to number and category to number node in Knime. These nodes take categories (i.e. small, medium, large) and

convert them into numbers (i.e. 0, 1, 2). This allows the predictor to recognize them as numerical values so they can be used in the prediction algorithm as you can't perform mathematical operations with words. The next step was to deal with the plentiful missing values. We tried to handle missing values as specifically as possible and our method to handle them was usually row specific. Generally, a rounded mean calculation was used. The thought behind this was that since these patients all have similar medical diagnosis and medical issues to begin with, the average could generally represent a valid value. Once the data has been converted to numerical values and missing values were taken car of, it was time to normalize it. This again was done in Knime and is a crucial step as it makes sure that all inputs are represented evenly, i.e. an age of 60 doesn't out weigh a drug dose of 0.1mg.

Modeling

Once the data was prepared it was fed through a neural network that was built in Knime. The net consists of an input layer with 46 nodes (number of inputs after preprocessing), a hidden layer with 15 nodes (rule of thumb is between number of inputs and number of outputs), and an output layer with three nodes (Three nodes aren't necessary but we wanted the bonus mark for the 3 classifications: >30, <30, No). The neural network was chosen as the dataset has a large amount of inputs and the data was relatively easy to classify in numerical form. As the test data does not have results, the training data had to be partitioned into test and training data to get an accuracy measure. The split was done 70% -30% with the latter for testing, and to not favor one section of the data, random sampling was done to ensure an even representation of the data. Once a consistent accuracy was obtained, the net was trained using the entire training dataset and was testing with the test dataset.

Other Approaches

Our first thought was to implement a decision tree learner but due to the lack of linear correlation a number of inputs, we ended up with a predictor that didn't function as intended. We also had issues getting the Neural net to work due to pre processing errors but this was solved with time and research.

Findings and Results

After modifying some settings (#layers/nodes) in the neural net, a consistent accuracy of ~60% was found. The confusion matrix for our 70/30 split on the training data can be seen to the right ->.

readmitted \ Prediction (readmitted)	NO	>30	<30
NO	12804	2005	15
>30	6343	2998	19
<30	1982	1036	28