Data Challenge Summary

Summary

The main goal of this project was to construct a model that could accurately predict whether a hospital patient would recover or not from their hospital visit by using a set of laboratory measures that were recorded throughout their stay. One of the main challenges of the project was dealing with unstructured patient data. In order to make the data compatible with the classification models all the data had to be processed and stored in a rectangular data frame, with features that summarized the measures across a patient's entire visit. Using this data frame containing all patient data, we began our model selection process, fitting our data on a logistic regression, AdaBoost, SVM, and Neural Network model structure. Using a 70%-30% training split, and 3-fold cross-validation to tune parameters, we ended up choosing an AdaBoost model given that it produced the best balanced error rate and AUC score during our validation process.

Data Processing and Feature Engineering

There were two main obstacles we needed to overcome in the data processing stage: giving structure to all the patient data files (text files) and handling the large amount of "missing" values. We decided to put all text files into a rectangular data frame such that each row represents one patient and each column represents a patient health metric. Initially, we had a column for each combination of time and health metric, but found that the columns were very sparse. We would have had to impute the missing values or run a model compatible with missing values. Imputing the majority of values based off of the data from a minority of patients did not seem like a good plan. It would have also made the legitimate values for that patient to be less influential. This also didn't paint a practical picture of what the data was telling us.

Because of these reasons, we decided that summaries of metrics for each patient were the most informative for a model. When processing each patient's file, we took the mean, median and standard deviation of a particular metric. Additionally, for each metric of a patient, we ran a linear regression which was regressed against time. We then used the coefficients from these regressions as a proxy of how the metric was trending. If the coefficient was near 0, it was overall steady, alternatively if the magnitude was large, it would show that the metric is going up or down. If there was no data for a metric, the coefficient was 0.

Because patients didn't have at least one recording for every metric, for those models that did not handle missing values, we had to impute those values. We imputed by column, filling in the missing values with the columns median in order to be more robust to outliers.

Model Training and Evaluation

Given the high number of “missing” values in the patient data, we decided to try out models that could handle NA’s: a tree based model (AdaBoost) and a neural network model (Multilayer Perceptron). We also tried Logistic Regression and SVM, using median imputation on the missing values. Each person on our team tackled a particular model structure, and we all followed the same training and evaluation approach.

To train our model, we first divided the entire dataset into a “training dataset” (those patients with an “outcome” label) and a “test set” (those patients without an “outcome” label). Given that we had no separate test set that contained “outcome” values, we decided to first randomly split the training dataset into a “validation training set” (70% of the training data) and “validation test set” (30% of the training data). The purpose of this was to use the validation training set to find the “best” parameters for our model, and evaluate that model (with the “best” parameters) on the validation test set.

We determined what our best parameters were by trying out different values for our parameters on our model, doing 3-fold cross validation on each combination of parameters, and deciding the best parameters by which model had the lowest balanced error rate. Once we had decided on which parameters were the best to use for our respective model, we then evaluated this best model by using the remaining 30% of the training data (the validation test set), and once again calculating the balanced error rate and auc. The balanced error rate was used throughout our validation process because of the imbalance of class 0 and class 1 labels in the training set (where BER is the average of the positive class error rate and the negative class error rate). Ultimately, we decided on our final model based on the balanced error rate and AUC score from the validation test set.

Model Selection

We exhausted as many possible combinations of variables to use (removed some, added some, etc.) and parameter values, and in the end, the model that produced the best BER and AUC score was an AdaBoost model with 1000 trees, a shrinkage of 0.01, depth of 4, and a probability cutoff value of 0.70. Other models tended to produce a BER score of 0.32 or higher, while AdaBoost consistently produced an error rate around .30 and an AUC score of around 0.76 during our validation process. The variables that proved the most effective in predicting non-recovery were the linear regression coefficients, the median of each measure, and the standard deviation of each measure, alongside Age, Admission Type, and Gender.

Overall, we chose AdaBoost because it was the model that gave the best BER and AUC score in the validation process, and it was the model that was able to handle such a large number of “missing” values in our data (without needing to do imputation).