CS156 Predicting Mortality

Li-Lian Ang

Minerva Schools at KGI

Problem Definition

Given data obtained within the first 24 hours of a patient's admittance into intensive care, the task is to predict whether the patient will die. The data set was compiled from Intensive Care Units (ICU) from a wide range of hospitals globally. The challenge posted on Kaggle gives a labelled training data set with 91,713 entries where 8.3% of entries are for class label 1 (patients who died).

Solution Specification

Cleaning Data

Certain columns which have nothing or little to do with mortality were removed such as all columns that provide id information and ethnicity. Height and weight were removed because there is already a column for BMI which relates both of these variables and outputs a more meaningful metric.

Columns with more than 60% of missing data were dropped since interpolating from it would yield more guesswork than real values. All other columns with missing data were interpolated with data within the same class. For example, missing data for the 'bmi' column was stratified according to 'hospital_death' and filled with the mean 'bmi' value of the same class. For categorical values, the missing entries were interpolated with the mode of the column. This makes the missing data more similar to its class which gives the model more information regarding similar entries rather than simply throwing out data or filling it with an overall average, which since most of the entries are for class 0 would skew the model.

To level out the class imbalance, I upsampled the entries for class label 1 by randomly selecting entries from class label 1 with replacement to make up for the difference in classes. Then, I added a small amount of noise to perturb the new entries with the assumption that patients with very similar entries will have the same outcome while giving the model different data points for better classification.

All non-numeric values were converted into numbers. Then, the data was split into a training and test set using stratification to preserve the percentage of classes in both. All data was scaled to prevent skew from arising due to high magnitude numbers.

Finally, I will use principal component analysis to reduce the number of features from 120 to 20. This step prevents noise from possibly unrelated features from dulling the signal given by relevant features. As shown by the scree plot below, these 20 features explain about 80% of the classification.

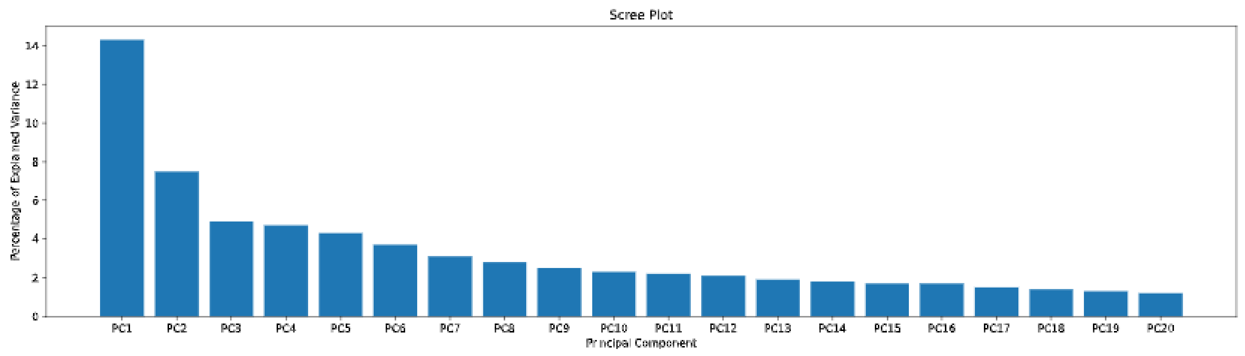

Figure 1. Scree plot of the percentage explained variance for each principal component.

Random Forest

The random forest model is a type of soft decision tree where rules are applied that steer data points down the nodes of the tree until it reaches a node with a specific classification. The random forest model selects samples from the data set with replacement and randomly chooses which columns to use to classify the entries. It will build multiple random forests and use the cumulative scores from all the trees to give the final classification of each data point.

A random forest is an appropriate choice given the kind of data since it is non-parametric, meaning that the data doesn't have to be linearly separable and is not perturbed by outliers. The separation between patients who will or will not die may not be linear since the pattern underlying it is extremely complex. It is safer to allow the random forest to decide on how each feature affects the outcome through rules, thereby making the model more specific. Another advantage is that random forests are much faster than exploration done classifying with Support Vector Machines and K-Nearest Neighbours

Testing and Analysis

To prevent overfitting, the several parameters were cross-validated to control the maximum depth of each tree. If a random forest is allowed to grow to any size or, there is a possibility that it will be too biased to the data as it becomes increasingly complex. By tuning the maximum depth, we can select the appropriate parameter that will maximise the testing recall.

We are primarily interested in predicting which patients will die. Misclassifying a patient who might have died but survived has lesser repercussions than vice versa. Therefore, we used recall to judge the performance of the model. Recall is calculated as the fraction of patients who were correctly classified in class 1 out of the total number of true patients who are in class 1.

The training recall score for the random forest was 0.956 while the test recall score was 0.526 with a 100 maximum depth of the tree. The huge difference between training and testing recall scores is because the tree has high bias, meaning that it fits the training set data well but is not able to extrapolate its findings accurately onto unseen data. This may be because the testing data includes many duplicated entries which were slightly perturbed which makes it more accurate in training but also more biased.

References

WiDS Datathon 2020. (n.d.). Retrieved from

https://www.kaggle.com/c/widsdatathon2020/overview