Zihan Zheng

BIOST 546

Final Report

March 11th, 2022

**OBJECTIVE**

The objective of this project is training a classification model that predicts whether the heartbeat is normal or abnormal using 187 consecutive measurements of heartbeat Electrocardiogram (ECG) signals.

**DATA DESCRIPTION AND PRE-PROCESSING**

The dataset for analysis contains three R objects: X\_train, y\_train and X\_test. X\_train is a 12552\*187 matrix where every row contains the i-th heartbeat ECG signal; y\_train is a vector of length 12552 with associated diagnostic labels (0: normal heartbeat, 1 abnormal heartbeat); X\_test is a 2000\*187 matrix where every row contains a heartbeat ECG signal. We are expected to make predictions for each observation in X\_test.

We first combine the R objects of X\_train and y\_train into a single dataset (12552\*188). Then, we set a seed and randomly split the dataset into training set and test set with a ratio of 80: 20, which results in 10042 observations in the training set and 2510 observations in the test set. The training set is used to tune the hyperparameters for each statistical modeling technique we would apply (penalized logistic regression, K nearest neighbors, decision tree and random forest) using 10-fold cross-validation approach. The test set is used to evaluate the prediction accuracy of each cross-validated model with optimal hyperparameters. We set a seed before training each model so that the results can be reproducible.

**MODEL TRAINING AND TESTING**

Since our features consist of 187 heartbeat ECG signals measured at consecutive time points, we expect correlation among different features. Moreover, we suspect that some of the features may be uninformative for our outcome of interest. Hence, we train a logistic regression model with lasso penalty to shed light on the subset of features that contribute more to outcome prediction. Through 10-fold cross-validation, the model with lambda = 0.000355 is selected as having the best performance, where lambda is the tuning parameter that shrinks coefficients towards zero. *Figure 1* shows the binomial deviance across different values of log (lambda). The lambda value corresponding to the lowest binomial deviance is regarded as the optimal parameter. The cross-validated model gives non-zero coefficients to 125 out of 187 features. It has a training prediction accuracy of 0.835 and a test prediction accuracy of 0.834. We then restrict the candidate predictors to the 125 features with non-zero coefficients to train the rest of the candidate statistical models (K nearest neighbors, single classification tree and random forest).

Given that the test prediction accuracy of penalized logistic regression is not ideal, we expect some degree of non-linearity of the association between the predictors and the outcome. Therefore, we attempt K nearest neighbor (kNN) classification, which is the simplest nonparametric approach. The optimal model selected based on 10-fold cross-validation has k=1, where k is the number of nearest neighbors that are used to predict the outcome value. A plot of prediction accuracy across different k values is displayed in *Figure 2*. The kNN model with k=1 has a training prediction accuracy of 1 and a test prediction accuracy of 0.945. The performance is substantially better compared to penalized logistic regression. The fact that the optimal k value equals 1 is a little surprising because the model with k=1 was expected to have the lowest bias but the highest variance, which usually leads to overfitting.

A more complicated nonparametric approach that we would attempt is training a single classification tree, which divides the predictor space into distinct and non-overlapping regions (R1, R2, …, RJ) and use the mean of the response values for the training observations in region Rj as the predicted response for that region. We prune the overgrown tree and plot the subtree size against the cross-validated misclassification error rate obtained from the pruning procedure (*Figure 3*). It turns out that the prediction accuracy peaks when there are 9 or 11 terminal nodes included in the subtree, which yield a training accuracy of 0.832 and a test prediction accuracy 0.802. The performance of a single classification tree is obviously less ideal than kNN classification.

Given that a decision tree has high variance, we decide to apply random forest to improve model performance. Random forest builds many trees by bootstrapping from the original training data. Additionally, it randomly samples a fixed number of predictors each time building a bagged tree, which aims to “decorrelate” the trees and further reduce variance. Based on 10-fold cross-validation, the best model performance is achieved when “mtry” is 25, where “mtry” denotes the number of variables that are randomly sampled as candidates at each split. This optimal model produces a training accuracy of 1 and a test prediction accuracy 0.968. The performance is much more satisfactory compared to a single classification tree. *Figure 4* shows the test prediction accuracy across a range of “mtry” values.

**RESULT**

Comparing the four candidate statistical models selected via 10-fold cross-validation, random forest gives the highest test accuracy. Hence, the random forest model is used to make predictions for the 2000 observations in the object of X\_test. The prediction accuracy turns out to be 0.941. Although slightly lower than the prediction accuracy that we obtained from the test set with 2510 observations, the model performance is still appreciable.

**TABLES AND FIGURES**

**Chart

Description automatically generated**

*Figure 1: Log Lambda vs. Binomial Deviance in Penalized Logistic Regression*

**Chart, line chart

Description automatically generated**

*Figure 2: k vs. Prediction Accuracy in kNN*

**Chart, line chart

Description automatically generated**

*Figure 3: Tree Size vs. Misclassification Error Rate in Single Classification Tree*

**Chart, line chart

Description automatically generated**

*Figure 4: Number of Variables Sampled at Each Split vs. Test Prediction Accuracy in Random Forest*