Ajay Iyer

DAT 402

Project 2: Breast Cancer Prediction Model

The next method I used was Logistic regression to predict whether the person had breast cancer based on the features that would predict it. This dataset was also accessed from Kaggle, and the author is Buddhini Waidyawansa. This dataset can also be found under the UCI Machine Learning Repository and the link is given along with the other links at the end of the report. The first thing I did after reading in the dataset in R was to get rid of the “id” column because I knew that is only to keep track of each person and not a feature/predictor for my model. Before reading in the dataset there was a blank column at the end so I named it ‘Bruh” so I could delete it later when I went into R. The diagnosis column stands for whether cancer is present with “M” standing for Yes and “B” standing for No. This will be changed later to make it easier for anyone to understand. Then, I wanted to find if there were any null/NA values so I could delete those columns even though I looked at the dataset beforehand, I wanted to check once again and there were none to be found. This made it a little easier when it came to doing Exploratory Data Analysis (EDA). Next, I wanted to change the values of the “diagnosis” column so it would be easier to follow along. If the value has an “M” then I will change it to “Yes” and if it is “B” then it would be “No”. Next thing I did was the train-test split. I made the train set 70% of the data and obviously the rest (30%) will be for the test set. For a sanity check, I checked the dimensions of the train and test to make sure they were good. Next, I split the dependent variable which was diagnosis, and my independent variables were all the features or everything except for diagnosis in the data. After that, I converted the data into a matrix so I could use the cv. glmnet function to do regularization methods for Logistic Regression. The first method that was produced was the Ridge Regularization. Then, I used the cv. glmnet function to get a plot the Binomial Deviance vs log(lambda). In the code we made alpha=0 because it is for ridge regularization, and this does change when needed to do for lasso. Next, I did another plot but this type it was MSE or Mean Squared Error. The cv. glmnet function was used once again except the ‘type.measure’ was set to “mse” rather than deviance. Then, I used ‘lambda.min’ to get the value of lambda to give us the minimum mean-crossed error through mean squared error and the value I got was 0.005 which is small so that was a good sign. Then I used the lambda.1se to get the value of lambda to give us the most regularized model when the mean squared error is within the standard error and the value, I got was 0.02. Then, I used the ‘coef()’ and the “predict()” functions to get the coefficients so we can compare if we are getting the same equation which turns out to be true. Now, I am ready to predict the test data. First thing I did was create a test y variable which was equal to what we are predicting, and the rest of the columns/features are predictors which will be in the test x variable. Next, I created a ‘pihat’ variable which will store the test x data which will be used later. Then, I converted the test x data to a matrix so I could do measure the performance. Then, I used the ‘roc()’ function to get the curve when the diagnosis is equal to “Yes” when the predictor is our ‘pihat’. Also, I used the auc() function to show what our performance would be with the data. Then, I plotted sensitivity vs specificity which had specificity was decreasing. Next, I printed the value of ‘AUC’ which turned out to be 0.9979 which shows this is a good model since there is a good measure of separability. Then I created a ‘yhat’ variable which will help us create a table for when ‘pihat>0.5’. Then, I created a table using ‘yhat’ with ‘testY’ to show the distribution with yes and no. I used the table to calculate the accuracy and the false negative rate. The accuracy turned out to be 0.9824561 and the false negative rate is 0.03448276. Then I plotted the beat coefficients which gave will show us the various models/flexibilities. This is the end of Ridge Regularization and now I wanted to see how different it would be if I used Lasso Regularization. I fit the data in the lasso model using cross-validation with 10 folds and alpha was set to 1 for lasso in the cvfit() function. Most steps were repeated once again and then I wanted to find the accuracy and false negative rate which done the same way as well. The accuracy turned out to be 0.9707602 and the false negative rate is 0.05172414. After comparing both models, the Ridge Regularization turned out to be better but there was not that much of a difference. Both models basically had a high accuracy but Ridge beat Lasso with accuracy by 1% and the false negative rate by 2%.

<https://www.kaggle.com/code/buddhiniw/breast-cancer-prediction>

<https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29>