ISYE_6501_WK1_HW

Anon Skywalker

8/26/2020

Question 2.2.1

Coming from a job as a Clinical Data Analyst, one of the areas in healthcare that tends to spike up the cost of a patient's ER visit is the near mandatory ECG that is performed when any number of risk factors align upon arrival.

According to current guidelines from the American College of Cardiology and the American Heart Association, "ECG should be performed within 10 minutes of ED arrival for all patients with chest discomfort or other symptoms suggestive of STEMI".

By using a classification model for this problem, we can see how many of the patients that arrived needed an ECG and how many did not need extra test added to their bill. Some attributes this model would have might utilize the patient's: *BMI, Age, Heredity, Chest Pain on arrival (Y/N), and Gender*.

Question 2.2.2

For this problem, the initial approach was to use the supplied code and attempt plugging in a few different extreme numbers for lambda (or C as used in the KSVM R equation).

Using values such as 1000, 100, 50, 1, and .001 idled around the same accuracy of about 86.39% and this percentage decreased as the number dropped below 1. Realizing this would take a more incremental approach to find the right number, I utilized a while loop function that would take in the data set, several integers such as a goal accuracy for the function to attempt to achieve, the count to start at, the interval the function would count up by, and a string for the kernel the model would use.

By using the "Vanilladot" Kernel, I arrived at a value of C that was around 0.00157778 yielding its highest accuracy of around 88%. To improve upon this level of accuracy, I switched to one of the more popular kernels, "rbfdot", and the function was able to achieve an accuracy of 98.16% using a C value of 508.6. However, the higher the C value, the higher the accuracy, which leaves me to believe that this is not the best kernel for this situation. Exploring additional kernels may possibly represent the data set more accurately and would need to be explored before making a choice in which one works best.

1. "Final count was: 508.6. Highest accuracy was: 96.94%. Using kernel: rbfdot."
2. "Final count was: 0.0015. Highest accuracy was: 87.00%. Using kernel: vanilladot."
3. "Final count was: 158. Highest accuracy was: 91.13%. Using kernel: anovadot."
4. "Final count was: 1.16. Highest accuracy was: 97.25%. Using kernel: splinedot."
5. "Final count was: 22. Highest accuracy was: 99.23%. Using kernel: laplacedot."
6. "Final count was: 57. Highest accuracy was: 92.05%. Using kernel: besseldot."

Assessment:

Using a variety of kernels, it would appear that laplacedot is the winner with a C value of 22 and a accuracy of 99.25%.

```
#Takes in a data set, decimal point representing accuracy, starting number
#for counting from, interval for incrementing the count, and kernel choice.
ksvm.function <- function(data, goal.accuracy, count, interval, k.choice) {
   accuracy <- 0 #initialize accuracy
   while (accuracy < goal.accuracy)

   {
      count = count + interval
      ksvm.model <- ksvm(as.matrix(data[,1:10]),as.factor(data[,11]),type= "C-svc" ,kernel = k.c
hoice, C = count, scaled=TRUE)
      pred <- predict(ksvm.model,data[,1:10])
      accuracy = sum(pred == data[,11]) / nrow(data)
      print(count) #uncomment for count progress while running
      print(accuracy) #uncomment for accuracy progress while running
   }
   FinalList = paste("Final count was: ", round(count, digits = 4), ".",
                     " Highest accuracy was: ", round(accuracy * 100, digits = 4), "%. ",
                      "Using kernel: ", k.choice, ".", sep = "")
   print(FinalList)
   return(FinalList)
}
#Uncomment to run lines below to utilize the above function with parameters of your choice.
ksvm.result <- ksvm.function(data, 0.99, 20, 1,"laplacedot" )
```

Question 2.2.3

For the K-Nearest-Neighbors (KNN) model, the idea behind the code is to test a new value of lambda (K for KNN model) for every row in the data set (each representing a data point). By using a set of for loops, I was able to step through each row of the credit data, using a different row each time as the test point and the remaining rows as the train data set. For each value of “i”, that row would be the selected row for testing purposes, utilizing column R1 to verify that the model predicted correctly. This process would be repeated for each value of K that we want to test, which can be any value between 1 neighbor to 654 neighbors within the data set. However, once you pass a certain threshold of neighbors, the prediction of the test point becomes diluted by the distant points that vary greatly. After running through the range of K as a initial test, the result was a optimal value of K being under 20. Because of this, the code below will only run through a subset of the data from 1:100 rows in order to speed up the discovery of the best fitting K value.

To elaborate on extreme values; when running through the entirety (nrow(data)) set of rows, I quickly ran into a issue where the for loop would test 654 rows against 653 possible points in the data set due to the [-i,] train data having only 653 rows. To negate this, I used (nrow(data) - 1) initially in the first for loop to prevent comparisons that were impossible to be made.

After using the kernal “optimal”, I ran through a few additional kernels and the results are posted below:

1. “The best K value is: 12. The highest accuracy is: 85.32%. Using Kernel: optimal”
2. “The best K value is: 6. The highest accuracy is: 85.17%. Using Kernel: rectangular.”
3. “The best K value is: 6. The highest accuracy is: 85.17%. Using Kernel: rank.”
4. “The best K value is: 8. The highest accuracy is: 85.02%. Using Kernel: gaussian.”
5. “The best K value is: 10. The highest accuracy is: 85.17%. Using Kernel: triangular.”
6. “The best K value is: 10. The highest accuracy is: 85.17%. Using Kernel: epanechnikov.”
7. “The best K value is: 15. The highest accuracy is: 84.71%. Using Kernel: biweight.”

8. “The best K value is: 10. The highest accuracy is: 85.02%. Using Kernel: cos.”
9. “The best K value is: 42. The highest accuracy is: 85.02%. Using Kernel: inv.”

Assessment: The above attempts with different kernels show that “optimal” at a K value of *12* is the best result. One method to potentially find a higher value besides just using different kernels would be to use the train.kknn() model as a crossvalidation method in order to find the best value of K. This would involve a similar “leave one out method” to reach the same destination but could hold potentially better results by using sample sizes instead of single points for testing. For instance, 2/3 of the data could be sampled out using the sample(1:nrows(data), size = round(nrow(data))/ 3), replace = false) function for test data and 1/3 could be sampled out for train data via the -i method.

```
#Takes in a range for lambda (K) to traverse and a kernel choice.
knn.function <- function(k.range,k.choice)
   {
   #an empty vector for storing values avg accuracy across all points
   #determined in the innermost loop.
   avgAccuracyk.list <- c()
   #the outermost loop steps through values of k that the user sets the range for.
     for(kcount in 1:k.range)
   {
      accuracyK <- 0 #resets accuracyk value
      #the innermost loop steps through each row of the data.
      for (i in 1:nrow(Data))
      {
         #The KNN model, utilizing the results column, train and test data, and a K value that i
ncrements.
         KNN.Model <- kknn(R1~., Data[-i, ] , Data[i, ] , k = kcount, kernel = k.choice,  scale=
TRUE)
         #The predicted value. if > 1, round to 1. If < 1, round to 0.
         KNN.pred <- as.integer(predict(KNN.Model) + 0.5)
         #compare that value with the result column from the data for ith point.
         KNN.Success <- KNN.pred == Data[i,11]
         #sum the knn.success values up for later comparison
         accuracyK  <- accuracyK + KNN.Success

      }

      avgAccuracyk.list[kcount] = accuracyK / nrow(Data)

   }
   max.accuracy <- max(avgAccuracyk.list) # max accuracy
   best.K <- which.max(avgAccuracyk.list)
   result <- paste("The best K value is:  ", best.K, ". ",
                   "The highest accuracy is: ",  round(max.accuracy * 100, digits = 2), "%. ",
                    "Using Kernel: ", k.choice, ".", sep = "")
   return(result)
}

knn.result <- knn.function(100, "optimal")
```