Jeremiah Herr, Prime Number Analysis 3, DATA 445

Abstract

The goal for this project was to build on the last one (Number Analysis 2) and try again to create a model (using the ones from Chapters 8 and 9) that can accurately classify which numbers in a given set are prime and not prime based on a given set of parameters. Once again, none of the tested models were able to consistently identify true primes.

Introduction

The data used for this project were all the integers 1 through 20,000. Using the Java programs I wrote for the last two projects (with more modifications), I generated these integers (along with some parameters of interest) and stored them in NumberSample1.txt and NumberSample2.txt. NumberSample1.txt was used as the training data set and contained all the integers 1 through 10,000, and NumberSample2.txt was used as the test data set and contained all the integers 10,001 through 20,000. Then I performed statistical analysis on the data using R (recorded in Nums2.R) and stored the important outputs in NumAnalysis2.txt.

Both data sets were organized (and labeled in R) as follows: the first column (“nums”) is the set of integers 1 through 10,000. The second column (“nth”) gives the numbers of primes less than or equal to each number. The third column (“order”) gives the order of magnitude of each number (in base 10). The fourth column (“prop”) gives the proportion of numbers that are prime at each order of magnitude (these first four parameters are the exact same ones from the previous project and were included for convenience). The fifth column (“gap”) gives the difference between each number and the largest prime less than itself (1 and 2 are the only exceptions; 1 was given a gap of -1 and 2 a gap of 0, as there are no positive primes less than either of these two numbers). The sixth column (“mod6”) gives the value of each number modulo 6 (since every prime greater than 3 is of the form 6k + 1 or 6k + 5). The seventh column (“mod2”) gives the value of each number modulo 2 (to determine whether each number is even or odd). The eighth column (“avgGap”) gives the approximate average gap between primes less than or equal to each number. The ninth column (“probPrime”) gives the approximate probability that each number is prime. The tenth column (“prime”) tells whether each number is prime, where “N” means a number is not prime and “Y” means a number is prime.

NOTE: both avgGap and probPrime rely on the Prime Number Theorem (source: https://en.wikipedia.org/wiki/Prime_number_theorem). See my comments in Nums2.R for more details.

Methods

Seven models were used in this project, namely two tree models (one pruned and the other un-pruned), bagging, a random forest, a support vector classifier, and two support vector machines (each using a different kernel: radial and polynomial)—in order to practice using each one and to compare their results and determine which (if any of them) could be used classify primes. The only concern in this project was the predictive accuracy of the models, hence their interpretability wasn't considered, and no subset selection methods were used. Since the number of parameters was very small relative to the sample size, dimension reduction methods were not used either. A boosting model was not used due R generating errors whenever I attempted to train one.

Each model was trained on the NumberSample1.txt data set and then tested on the NumberSample2.txt data set.

Each model generated a pair outputs: a table showing the number of accurately and inaccurately classified integers, and a test error estimate (given as the mean of the proportion of misclassified integers). Each table gives the number of true negatives (row N, column N), false positives (row N, column Y), false negatives (row Y, column N), and true positives (row Y, column Y).

Results

Both tree models, the random forest, support vector classifier, and the radial kernel support vector machine models all gave the identical result of classifying every integer as being not prime, and each had an estimated test error of 0.1033 (Outputs 1, 2, 4, 5, and 6; NumAnalysis2.txt). The bagging model gave an almost identical result, except it correctly identified 11 primes and misclassified 13 non-primes and had an estimated test error of 0.1035 (Output 3). The only model that gave a significantly different (albeit not useful) result was the support vector machine using a polynomial kernel, which identified 973 true primes and 3,851 non-primes and misclassified over half of the entire data set by falsely identifying 5,116 non-primes as being prime, resulting in a much larger estimated test error of 0.5176 (Output 7).

So, none of these models performed any better or worse than the Chapter 4 models. The more accurate models would classify all the integers as non-prime, while the model with the most true positives also had an even larger number of false positives.

Conclusions

As with the previous project, none of the tested models could reliably classify numbers as being prime or not prime. Most of the models would greatly underestimate the number of primes, and one model greatly overestimated the number of primes. These results are about the same as those found in the last project, though it may be necessary to try fitting the Chapter 4 models again using the two data sets from the current project to ensure a fair comparison (but I suspect the results would not change much if at all).