Forrest Fallon  
11/22/21  
HW6 Naïve Bayes

This week’s exercise is a first-dive into the world of naïve bayes, and compare results to our decision tree process as learned last week. We will be aiming this program at a dataset containing images of handwriting samples of the numbers zero through nine. We want our decision tree and naïve bayes programs to be able to read the data, then predict the data on its own with minimal supervision. After using both programs, we will compare each and see which is best for the data on hand.

**Part 1: Data Prep**

As the assignment says in its instructions, this dataset standalone is very large. Weka has a hard time with the size, and even R takes a while to run the steps necessary for our algorithms. We will cut the data down to 10% and evaluate from there.

Text

Description automatically generated

As seen above, we needed to make sure all classification variables were structured as such, which luckily was only the label column. After turning that into a factor, we split the datasets into their respective slices, and subset those. These steps are only necessary for the decision tree algorithm, as the naïve bayes algorithm requires less data cleaning to run properly.

**Part 2: Decision Tree**



The above parameters are a result of many different tries to create any differences in accuracy from the resulting confusion matrix (shown below in a few steps). The CP, minsplit, and maxdepth values all have minimal effect on the accuracy found within the prediction model, I believe this to be a result of the wide range of values within the data, a result of the nature of differences within handwriting. Pixel-data is a detailed mess of a field of measurement, and more data wrangling than R is capable of would likely result in a more fine-tunable result. Below are my results for a confusion matrix.

Text

Description automatically generated  
Text

Description automatically generated with low confidence  
 Upon first inspection, this is a solid prediction! 85% accuracy with a stellar p-value, we can place confidence in the decision tree’s ability to predict the correct number based on the data provided. From looking at the table we can see that the most incorrect number prediction is the number nine when presented with the number four. This makes sense as the two numbers share a lot of characteristics, a machine would likely have a hard time with this and would need large amounts of data in order to fine-tune this on its own.

**Part 3: Naïve Bayes**

Below are the commands necessary for building a Naïve Bayes model with R. It is worth noting that there is no need to create a data frame, merge columns between the data frame and original data subset, and so on. Naïve Bayes comes ready to take the data, compare it to a given dataset, and throw caution to the wind and give you predictions based on its complex computations going on behind the scenes. In the right circumstances, a naïve bayes model will save you time and potential error debugging when compared to a decision tree model. Anyways, here is the code:  
  
Text

Description automatically generated

Text

Description automatically generated with medium confidence  
 The first noticeable difference here would be the steep decline in accuracy. 55% is nothing to write home about, and the program could not generate a p-value. This data is clearly not fit for the naïve bayes model.

In the spirit of fine-tuning, however, I decided to run the entire dataset through the naïve bayes model to see if anything different was found:  
Text, calendar

Description automatically generated with medium confidence  
 These findings are unfortunately even worse than the smaller dataset, which is to be expected. The backbone of the naïve bayes model is Bayes Theorem, which measures the probability of A given B. If A happens, how likely is it that B will happen as well? In our case, there are so many variables and options that the naïve bayes model simply cannot make the right decision most of the time. This is particularly unfortunate given the fact that this second model took almost 5 entire minutes, and made my computer sound like a jet engine while it was calculating.

**Part 4: Comparison**

The obvious findings of these models when compared is that when given a dataset with a very high number different outcomes, you will often find better accuracy from a decision tree model. The ability to subset data, split it meaningfully, and make decisions based off of those splits creates much more accuracy in the findings when compared to naïve bayes. If a dataset has a few possible outcomes, a decision tree will not only be redundant, it will also likely take more time to fine-tune and create meaningful inference from. The next set of differences comes from the computation time needed to run each of these models.   
 When running the following code for each model, we can see the exact difference in time taken to run the model:   
Text

Description automatically generated  
  
Text

Description automatically generated  


The difference is clear, even when running the data trimming necessary for the decision tree, the run time comes in almost 40 seconds less than the naïve bayes model. Keep in mind, this was only for a dataset 10% the size of what it should be, and this dataset is not nearly as large as more meaningful ones in existence.

In conclusion, the differences between the two models are on a need-to-know basis for any data scientist. Being able to recognize when a naïve bayes model can be used is important for not only saving time, but also saving accuracy. When in doubt, try the decision tree model first; if those findings are not satisfactory, give the naïve bayes model a try as well. The main quantifier for when to use each model would be the number of outcomes, if there are too many options then you will likely end up using the decision tree.