Agustin Ortiz

CSE 5243

Homework 2 Report

**Introduction**

For this analysis, we have included 2 datasets partitioned into “Train” and “Test” subsets for data processing, data calculations, and analysis of the output. The first dataset, named “Iris”, consists of 150 records of flower lengths and widths. The second set, named “Income”, is much larger at a sample of records consisting of various attributes describing a person, their job and the amount of income. The chosen language and processing of the data is R. All the code is written inside R studio to give the developer the best environment for debugging and will be contained in a \*.R file. The parent directory of this report includes a read me file containing the procedure in which to run the code.

**Abstract**

The goal of the script is to locate its nearest neighbors by using the KNN algorithm, and then using this method to predict its class. There is a resulting output which will include the actual class of the record, a predictive class and a Posterior probability. The training set is a larger set in which our model uses as the base to compare the test. The test set runs through the model, computes distances, and produces an output of predicted classes.

**Pre-Processing**

The pre-processing of the data is required to get a retrieve a reliable analysis. Pre-processing includes data transformations, excluding outliers, normalization, and finding missing values. The algorithm includes all data processing before our distances are calculated.

Normalizing all the data is required to get the distances. There is a simple normalize function included in both scripts to change the values to lie between 0 and 1. This allows the data distance to calculated with a better proximity to other attributes.

We must do this to both the training and the test set. All data must be consistent at every point in the process.

**Code**

As stated above all code for the script is written in R. The script runs through the pre-processing of each attribute individually. After all data is filled in, outliers removed, and have been normalized, each record is passed to a Euclidean distance formula. The training set is used for each record of the test set to be compared against. Each row in the test set gets a distance to include every value in the training set. The vector returned is then indexed and sorted. The output format forces the developer to use a for loop and place each neighbor into a new data frame per their index.

The nearest neighbor was calculated using the indexes of the training set and then gathering the highest frequency of a class to determine the predictive class.

The Posterior probability is run through a method after the predictive analysis is previously gathered. Each output record gets a probability calculated from the probability of the predicted class happening, given the probability of the actual class. The is the last vector added into our matrix, the converted to a data frame for output.

**Output**

The output includes a 3 columns, actual class, predicted class, and a probability. The iris data takes less than 10 seconds to compute and the income takes around 7-8 minutes to compute. There were multiple computations performed in order to see different patterns and anomalies in our data.

**Analysis**

The developer was able to gain the correct output and ran the algorithm though many different tests. The K value was programed to be easily changed. There were two separate tests run to see what the difference of K would have. A value of 1 seems to have less accurate predictive classes while a higher value like 5 had more correct classes predictions. There was also a bug discovered when an even number of K was used. The compiler would report a runtime error because there was no max value to be chosen for the predictive class. For example, if k = 4, and the 4 nearest neighbor’s classes were equally split by 2, there was no way to decide which predictive class to assign. You can look at the table below to find confirm the findings for the iris set:

|  |  |
| --- | --- |
| K Value | # of Correct Predictive Classes |
| K = 1 | 49 |
| K = 5 | 55 |
| K = 9 | 58 |

The combination of the distance formula and the predictive class method provides the final result. I put the result into a confusion matrix to see how well these two performed under a K value of 5. The table below shows a very high percentage of True-Positives at a rate of 76% of the test sample set.

|  |  |  |
| --- | --- | --- |
|  | Predicted |  |
| Actual | <=50K | >50K |
| <=50K | 200 | 21 |
| >50K | 48 | 19 |

Lastly, the developer lowered the sample size of the the training set to try and identify any anomalies in the current analysis. After decreasing the size of the training set, the KNN algorithm seemed to be as efficient it terms of the predictive analysis. This is to be expected because the less data there is to work with, the more likely noise can play a role in our nearest neighbor algorithm.

In conclusion, the KNN algorithm can be a good fundamental tool for predictive analysis. Our data yielded high accuracy percentage rates on the predicted class when there was a healthy amount of data to compare the test set.

**Future Work**

The developer was focused on getting correct outputs, and thus didn’t allow for enhancements to be made in other parts of the program. The code could be refactored into a more efficient algorithm from a computation perspective. It took close to 10 minutes to run the income data sets. Also, faster analysis can be done with other classes to help with graphical analysis. At no point did the developer put any of the data to any graphical representation.