Implementing kNN algorithm Jaccard Similarities

by:

Darius Kharazi

September 29, 2017

The Ohio State University

1. Introductory Program Output Description

The provided program is able to output useful information that is extracted from input training data. Once training data is read, the program outputs each observation's "ID," along with each observation's "class," which is the income class of the given "ID." After analysis and program completion, the predicted "class" and posterior probability is also provided with the previously stated "ID" and "class" variables. The posterior probability is based on the *k* nearest neighbors algorithm and jaccard similarity calculations, and the predicted "class" variable is based on this posterior probability of the particular class.

2. Model Training and Testing

The training data was used in the first report, which effectively discovered predictive relationships between its observations. First, exploratory analysis was conducted on the training data to understand whether the data should be transformed or not. After performing any necessary transformations and binarizing the data, jaccard similarities were calculated between each observation. The k-most similar observations were shown for each observation, which had been determined by the highest jaccard similarity values.

This report required applications from the previous report, along with an implementation of a kNN algorithm based on matching our testing data to our previously applied training data. Essentially, posterior probabilities were computed by calculating jaccard similarities from the testing data, rather than the training data. These posterior probabilities would be assigned a value depending on its degree of similarity, and provided a predicted value for each observation's "class" variable. In order to evaluate the performance of the general classification method, these predicted "class" variables were then compared to the true "class" variables. Since the predicted "class" values were only preliminary predictions and were not perfectly comparable to the actual "class," the true positive, false positive, true negative, and false negative values were accumulated for each posterior probability across the total amount of observations, in order to understand some of the error. Additionally, a confusion matrix was constructed in order to understand the general accumulation of true positive, false positive, true negative, and false negative values. These true positive, false positive, true negative, and false negative values were useful for calculating True Positive rates, False Positive rates, True Negative rates, False Negative rates, Recall, Precision, and any plotting of the ROC curve.

3. Performance Evaluation of the Classifiers

Using the testing data was important for determining the performance of the classifiers. Moreover, adjusting the two parameters of the kNN algorithm is as

important for determining the performance of the classifiers. For example, the number of neighbors *k* and the proximity measure impacted the performance and values given by the confusion matrix.

Given that *k* is equal to 5, the confusion matrix for the total training data has fairly large False Negative and False Positive values (Figure 1). The kNN algorithm used for distinguishing the confusion matrix values in Figure 1 is not an off-the-shelf kNN implementation, and instead a personal implementation. Additionally, the error rate for the training dataset 74.3%. Although the off-the-shelf kNN implementation is common in R, the off-the-shelf kNN implementation has comparable confusion matrix values and error rates to the personal implementation. Specifically, the off-the-shelf kNN implementation produces a similar, but worse error rate of 72.6%.

	Predicted Success	Predicted Failure
Actual Success	3	64
Actual Failure	10	211

Figure 1: Confusion Matrix for k=5

Given that *k* is equal to 20, the confusion matrix for the total training data has a larger False Negative, a lower False Positive, a lower True Positive, and a larger True Negative (Figure 2). The error rate for the personal implementation is 76.7%, while the error rate for the off-the-shelf kNN implementation produces a 76.4% error rate. These error rates are nearly equivalent to each other, and essentially equal to the corresponding error rates for *k* equals 5. However, the values a part of the confusion matrix are differently distributed, i.e. True Positive, True Negative, False Positive, and False Negative values.

	Predicted Success	Predicted Failure
Actual Success	0	67
Actual Failure	0	221

Figure 2: Confusion Matrix for k=20

Given that *k* is equal to 10, the confusion matrix for the total training data has a lower True Positive, larger True Negative, larger False Negative, and lower False Positive, compared to the confusion matrix for *k* equal to 5. The error rate for the personal implementation is 75.3%, while the error rate for the off-the-shelf kNN implementation produces a 76.7% error rate. In this case, the error rates are nearly equivalent to each other, but the error rate for the off-the-shelf implementation is the better of the two. Again, the values a part of the confusion matrix are differently distributed, which indicates the user may find each confusion matrix useful for different purposes, i.e. the lowest False Negative or lowest False Positive.

	Predicted Success	Predicted Failure
Actual Success	0	67
Actual Failure	4	217

Figure 3: Confusion Matrix for k=10

5. Additional Measurements

By including a different proximity measure during the implementation of the kNN algorithm, the prediction results should not change significantly. The two proximity measure that are adjustable in the program are the cosine and jaccard similarity functions. Although the relative values differed due to the differing algorithms, the final results were very similar. For *k* equal to 10, the ROC curve seems good, but looks incomplete because of the lack of data at each threshold and the large amount of True Negative values in the training dataset. It is however difficult to construct a proper ROC curve, since there are not any True Negative values for each threshold after using the kNN algorithm, which can be seen in the "output.roc.df" output file or the confusion matrices shown above. As a result the precision, recall, and f-score values are skewed, since the False Negative and True Negative values essentially equate to zero. However, the True Positive and False Positive values have high amounts, which seems to be an effect of a lack of False Negative and True Negative values from the actual data, rather than an issue with the kNN algorithm. Moreover, the True Positive Rate of the total training data is 0%, since the amount of True Positive values is 0; whereas the True Negative Rate of the total training data is 1%, since the total amount of True Negative values were correctly predicted. Additionally, the True Negative Rate of the total training data is nearly 76%; whereas the False Negative Rate is about 24%.

6. Analysis of the Results

After further analysis, there seems to be more reason to establish a decision boundary using the kNN algorithm for the "class" variable. According to Figure 4, there seem to be a greater proportion of more educated individuals who are of a greater class, which may seem obvious, but a notable trend regardless. Additionally, the proportion of married individuals from the class greater than 50k is nearly 78%; whereas the proportion of married individuals from the class less than 50k is only 41%, which is nearly half of the former figure. Interestingly, nearly 40% of individuals who are a part of the "<=50K class" have never been married; whereas only about 7% of individuals from the class greater than 50k have never been married. Lastly, the proportion of female individuals from the class greater than 50k is only about 18%; whereas the proportion of female individuals from the class less than 50k is nearly 29%, which is nearly a 10% increase from the former figure.

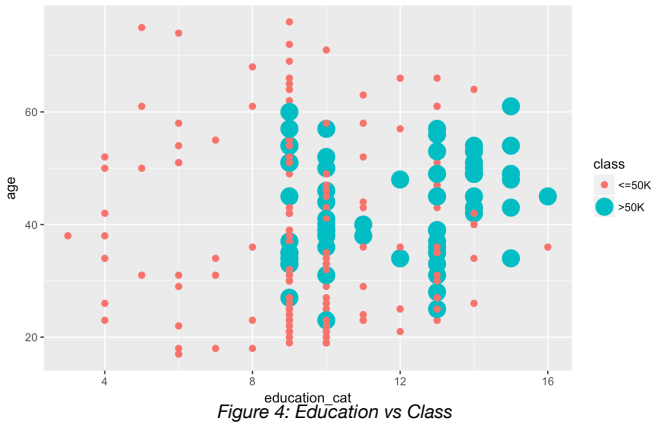

Figure 4: Education vs Class