KNN MODEL METHODOLOGY & EVALUATION

LUKE HOUGHTON (Team Leader) lhoug001@gold.ac.uk
(IMPLEMENTATION AND REPORT)

KACPER WISNICKI kwisni001@gold.ac.uk
(IMPLEMENTATION AND REPORT)

THIS REPORT CONTAINS AN IN-DEPTH ANALYSIS AND EVALUATION OF THE KNN (K NEAREST NEIGHBOUR) CLASSIFICATION MODEL.

INTRODUCTION

In this report we shall review and explain the methodology and the analysis behind the K nearest neighbour algorithm (Denoted KNN) and the model implemented within Java. We shall also be going through the steps of the data analysis process, the results including the accuracy and efficiency of the model and the best value of K in terms of optimisation. We shall be using the results stored in the results.txt file to refer to these areas of analysis too.

There shall also be an in-depth analysis of the confusion matrix and the performances that were observed using the test set data provided at the start of the assignment.

METHODOLOGY

The methodology behind the model is essentially to find whether or not the potential candidate earns more than $50,000 per annum. The model then needed to take the best value (closest distance) of K (nearest neighbour) and evaluate multiple performance measures using the test data set provided such as: precision (accuracy), sensitivity, confusion matrix and the specificity in accordance of the class that is less than or equal to $50,000 (salary) per annum.

In order to do this, we needed to create a set of steps to analyse the data throughout the model and to also produce successful and accurate results using the test and training data sets provided. We also were required to produce results within our model by using five cross validation (5CV).

Below are the data analysis steps we had used within our implantation of our K nearest neighbour model.

These steps were:

1. First the data must be read, stripped of irrelevant data and the raw data is then stored.
2. The data must then be converted into numerical weighted values in order for the model to then process the data.
3. Any values (data) that is missing (in this case replaced by a question mark) must be given a value generated by calculating the average (mean) of the other existing data. For example, a record (e.g. age datum) may be missing. Using the existing age data, we must calculate an average (mean) in order to fill the missing records. This is vital as without doing this, the model will not be accurate and will skew the end results.
4. The training data and test data must then be prepared and separated into five separate folds. In this case there are 11,000 records, therefore the first fold shall contain 8800 training data records and 2200 test data records, and this continues onto the further folds. However, the ratio between the training and test data must be reduced on each fold to produce an accurate set of fivefold (five cross) validation

results. For example, the second fold shall contain 6,600 training data records and 4400 test data records

5. Cross validation then must occur in order to calculate the results of each of the five folds. The folds must then need to produce results based on their confusion matrix, precision, sensitivity and specificity. To calculate the nearest neighbours (nearest values) of the possible data point the Euclidean distance shall be used. By using this formula, all data points shall be given a distance. This distance is calculated using the attributes within each record, for example: Age, salary and nationality etc. Each data point is then given a distance (score).
6. The data points are then ordered from lowest distance (closest) to highest distance (furthest). We can then see the accuracy of the model itself by counting the results and stating whether they fall into the positive category or negative category. If they fall within the positive category, then we shall say the predicted point is a hit and else it shall fall under the miss category.
7. We can then calculate the accuracy by the number of hits (positive) values divided by the number of predicted values of each fold and the final fold.
8. To calculate the confusion matrix, we then use the processed data from each fold and calculate the true positive values, false negative values, false positive values and the true negative values that are produced from the KNN classification model.
9. To calculate the sensitivity, we take the true positive values and divide them by the true positive values plus the false negative values. And to calculate the specificity we take the true negative values and divide them by the true negative values plus the false positive values and thus are results are produced.

By using these steps, we were able to accurately and efficiently implement the process of data analysation within our model that would allow the other components of the model to produce accurate and precise results based on the datasets provided.

Process of Implementation

In order to fully implement the KNN algorithm along with optimisation testing we first needed to read the data provided from the training data set that consisted of 11,000 records including data such as income, nationality and age.

We first created a multi-dimensional array consisting of 11,000 by 17 to store the values of the data that was going to be collected as string datatypes to begin with. The buffered reader was then used to pass values from the csv file into the multi-dimensional array that would hold the data that would be used further on in the program. The data (columns of data) collected then needed to represent a numerical value along with also weighting those values in order to use within the KNN model. To do this a series of nested for loops were implemented to loop through the training data array we had originally made that would go

through each column and give a numerical value to the string values (data) collected and add them to an Arraylist. We did however notice several records had missing data, therefore to overcome this obstacle we initially implemented an if statement that would be triggered if any fields had missing data. The missing data would then be replaced by an average (mean) that would be calculated.

It was vital to replace the missing data as this could have had major implications on the KNN model itself and would have resulted in inaccurate results. The average (mean) values that replaced the missing data were calculated using the values within each column. Excluding the last three columns as there were no relevant missing data.

We then proceeded to split all the values (data) collected and prepare them for the five folds containing double datatypes (variables). In order to do this, we had several if statements that would determine what records to put into each of the five folds that we would need for the five-fold cross validation later on in the model. We then needed to do the same thing with the test data, therefore we proceeded to use the same process in order to prepare and load the data for the test data too.

We then needed to implement a function that would allow us to calculate the distance between the data points of the test and training data. We decided the most efficient and effective way was to implement the Euclidean distance. We implemented the formula ((distance((x, y), (a, b)) = $\sqrt{(x - a)^2 + (y - b)^2}$) , however as we relied on the column data i.e. race, salary , nationality etc. We needed to apply the formula to each column (field) by doing this we were able to efficiently implement the next phase of the model which was taking

Now we had both the five folds created for the training data and the test data we then needed to find the distance by cycling through both sets of data and counting the distance from small to large. We first created a multi-dimensional array that would store the distances we were about to collect this would store up to 2200 data points and 40 results (nearest neighbours), this would be necessary to estimate if a potential individual were to earn more than $50k or less than the salary amount.

We proceeded to take the values from each of the folds and apply the Euclidean distance function we had made to measure the distances between all of the data points. We then had an array containing all of the distances, however it was not automatically sorted by distance, thus we created a for loop that would essentially cycle through the distance data and sort it by lowest to highest distance. In order to then display the nearest 40 neighbours, we ran another for loop to store them into the test data results array, this would leave an efficient and organised array that we would later use to display the results.

Finally, we then needed to test if the nearest neighbours (data points) found were either negative (meaning less than $50k) or positive (meaning more than $50k). To do this we implemented a for loop that would cycle 40 times (as we needed 40 nearest neighbours) and on each cycle, it would check if the values of the index within the test data array were equal to 1. If so the positive variable would be incremented, however if not the negative variable would be implemented. These results would then be stored within an Arraylist which would then allow us to calculate the number of hits (correct estimates) required in order to calculate the accuracy of the KNN model that has been implemented.

We had set the value of the hits to zero, naturally. The for loop then would deal by cycling through the result Arraylist we had made previously and checking the value of each Arraylist index against the values within the training data array. If and only if the results equalled one another we would the number of successful hits become incremented.

And finally, we calculated the accuracy of each value of the neighbours (K) by dividing the number of successful hits by the number of data points, which in this case was 2200. With the model now functioning efficiently and appropriately we then wrote each value of each neighbour (K) and the accuracy to a separate results file (results.grid.txt) that can be found within this assignment submission.

RESULTS

The results that were produced by the implementation of our K nearest neighbour model were produced by implanting a series of tests. These included: Confusion matrix testing, precision (Accuracy) testing, Sensitivity testing and Specificity testing.

Using five-fold cross validation (11000 records) and the best K value (1 - 39) on the testing data (16281 records) we found:

- Confusion matrix:
  - 1262.0 | 2584.0
  - 1126.0 | 11309.0

- Accuracy: 0.7721270192248634 (77.2%).
- Sensitivity: 0.07751366623671764.
- Specificity: 0.6946133529881456.

These results indicate that the tests based upon the model implemented have successfully managed to calculate accurately and appropriately realistic and specific results in accordance with the data sets provided. We have therefore managed to design, implement and test a model that can be used to accurately make predictions based upon five-cross (five-folds) validation as well as testing the model itself to ensure that the model produces accurate results/predictions based upon the data sets provided.

CONCLUSION

The model that we have produced is accurate, efficient and successfully handles missing data/values that were provided within the datasets provided. We have also managed to successfully run several tests based upon the model itself to gain an understanding of the accuracy, specificity, sensitivity and precision. Within this assignment and report we have successfully managed to calculate the nearest 39 neighbours (39 K values) and have produced these values based upon the data sets provided. We have also overcome the challenges of missing values/data within the data sets given and have handled this by replacing the missing values with averages based upon the other records provided. Thus, we have been able to keep the results that the model produces accurate and non-misleading by doing so.

Overall the design, methodology and implementation of the K nearest neighbour model were successful as it manages to produce accurate results both in the tests that were conducted and the predictions it produced.