TUTORIAL 7

Aim:

In this tutorial we will cover:

1. **Reading data files into R**
2. **Preparing and transforming data**
3. **Splitting data into train and test components**
4. **Fitting kNN classifiers in R**
5. **Evaluating the performance of kNN classifiers**

Mode of study:

You can work through this tutorial in your own time, or in one of the assigned venues during the tutorial slot. No in-person help is available on Friday 29 September 2023, as the university is following a Monday timetable on that day. You should therefore work through this tutorial on your own. You can ask for help on the module's Discussion Forum if you require assistance; alternatively, during the tutorial session on Friday 6 October you can ask for help with Tutorial 7 and/or Tutorial 8.

Solutions will be made available in Week 10.

Before you start

If you have not worked through Practical 7 yet, you should do so before starting this tutorial. If you struggle with importing the data into RStudio, you should refer to Practical 4 and Tutorial 4.

DATASET

This week we will again use a dataset from the UCI Machine Learning Repository (recall that the mushroom dataset we used in Tutorial 5 also came from this repository).

The dataset we will be using comes from research done at the University of Wisconsin in the USA. This is an image classification problem in the medical domain. The original dataset consists of digitised images of fine-needle aspirate of a breast mass, and the purpose of the analysis is to aid in diagnosing breast cancer. The images have been processed and features have been extracted, relating to different characteristics of the digitised cell nuclei. The first variable in the dataset is an identification number, the next is the cancer diagnosis ("M" = malignant and "B" = benign). This is followed by 30 numeric measurements comprising the mean, standard error and worst (i.e. largest) value for the following 10 characteristics of the digitised cell nuclei:

- Radius
- Texture
- Perimeter
- Area
- Smoothness
- Compactness
- Concavity
- Concave points
- Symmetry
- Fractal dimension

You can read more about this dataset at
https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic).

Links to numerous journal articles using the dataset is also provided.

A CSV version of the dataset was uploaded onto SUNLearn and is called `wisconsin.csv`.

QUESTIONS

1. **Load the libraries `caret` and `class`.**

2. **Import the dataset `wisconsin.csv` into RStudio and store it in an object called `mydata`.**

3. **View the structure of the dataset.**

   You might find that the target variable (`Diagnosis`) is of type character instead of factor.

   If this is the case, you can fix this by using the `factor()` function and specifying the levels of the factor in the following way:

```
mydata$Diagnosis <- factor(mydata$Diagnosis, levels = c("B", "M"), labels = c("Benign", "Malignant")))
```

   You will also see that the dataset includes an identifier column (`id`). This should not be included in your model **so delete this column from your dataset**.

4. Recall from the practical this week that working with a very imbalanced dataset can make classification more difficult. For instance, good accuracy can be achieved by simply predicting the majority class.
Therefore, **examine the distribution of the target variable `Diagnosis` by constructing a table showing the proportion of observations in each class.**

5. From this week's theoretical lecture you will recall that the distance calculations required for kNN are very dependent on the measurement scale of the attributes.
Calculate the summary statistics for the attributes `mean_radius`, `mean_area` and `mean_smoothness` and take note of the fact that the scale of measurement for these three attributes differs dramatically. It will therefore be necessary to scale the attributes in the dataset.

6. In the practical this week, we used *z-score standardisation* to rescale the attributes. In the theory lectures, we also mentioned that *min-max normalisation* is another way of rescaling the data. This rescales the data to the interval between 0 and 1: the minimum value gets transformed into a 0 and the maximum value into a 1, with every other value to a decimal between 0 and 1. For instance, if the minimum value for a feature was 10 and the maximum value was 30, then a value of 20 would be transformed to 0.5, since it is halfway between 10 and 30.
You should rescale the attributes in the `mydata` dataset using min-max normalisation. To do this, you should use the `preProcess()` function in the `caret` package in R. You should then also **create a dataframe called `myattr` which consists of only the attributes, not the target variable.**
Also check that the min-max normalisation worked correctly by looking at summary statistics for the `mean_area` attribute.

You should therefore enter the following code:

```
process <- preprocess(as.data.frame(mydata), method = c("range"))
myattr <- predict(process, as.data.frame(mydata))
myattr <- myattr[,-1]
summary(myattr$mean_area)
```

7. Create a label vector which contains only the target variable. Call this object `mylabs`.

8. Split the data into a training and test component. You should have the first 469 observations in the dataset in your training data, and the remaining observations in your test data. Call your training attributes `train.attr` and your training labels `train.lab`. Call your test attributes `test.attr` and your test labels `test.lab`. Set a seed of 7 before fitting the models, so that your results are reproducible.

   Note that the records in the original dataset were already randomly ordered. For this reason, we can simply extract consecutive observations for our training and test sets. If there was any ordering of the data, this would not be a suitable approach, and a random sample would have to be taken instead.

9. Fit the following three models:

a) 1-nearest neighbour

b) 5-nearest neighbour

c) 25-nearest neighbour

10. **Evaluate the performance of the three models fit in Question 9 by calculating the overall accuracy, but also the number of false positives and the number of false negatives.**
Note that a *false positive* will be when a mass is classified as malignant when it was in fact benign, and a *false negative* will be when the predicted value was benign but it was in fact a malignant tumour.
Based on these results, consider which of the models you think is preferable.

After completing this week's practical and tutorial you should be able to fit basic kNN models in R on a training dataset, and evaluate the performance of the models based on a test dataset.