PRACTICAL
Regression Methods
Thursday, 16 January 2025

Introduction

Regression analysis refers to methods which seek to establish the relationships between dependent (outcome) variables and independent (predictor) variables and construct a model capable of predicting outcome values. In most basic scenarios, a single outcome variable is predicted based on one or more predictor variables.

This is similar to classification methods, which seek to predict the *class* of an observation based on predictors. The difference is that instead of selecting from a finite group of classes, regression produces a numeric outcome, which is generally continuous and can take an infinite number of values.

Dataset

The datasets used today are based on the ones you previously used on Tuesday (*Ensemble-based support vector machine classifiers as an efficient tool for quality assessment of beef fillets from electronic nose data*, Mohareb et al*., Analytical Methods*, 2016, **8**, 3711). However, instead of associating e-nose data with classes assigned to beef samples by a sensory panel, the predictor data is associated with the counts of foodborne bacteria sampled from the meat and grown on a generic medium. As bacterial growth is a primary cause of meat spoilage, the ability to accurately estimate and predict bacterial counts based on automated measurements of volatile compounds produced by foodborne bacteria is highly desirable.

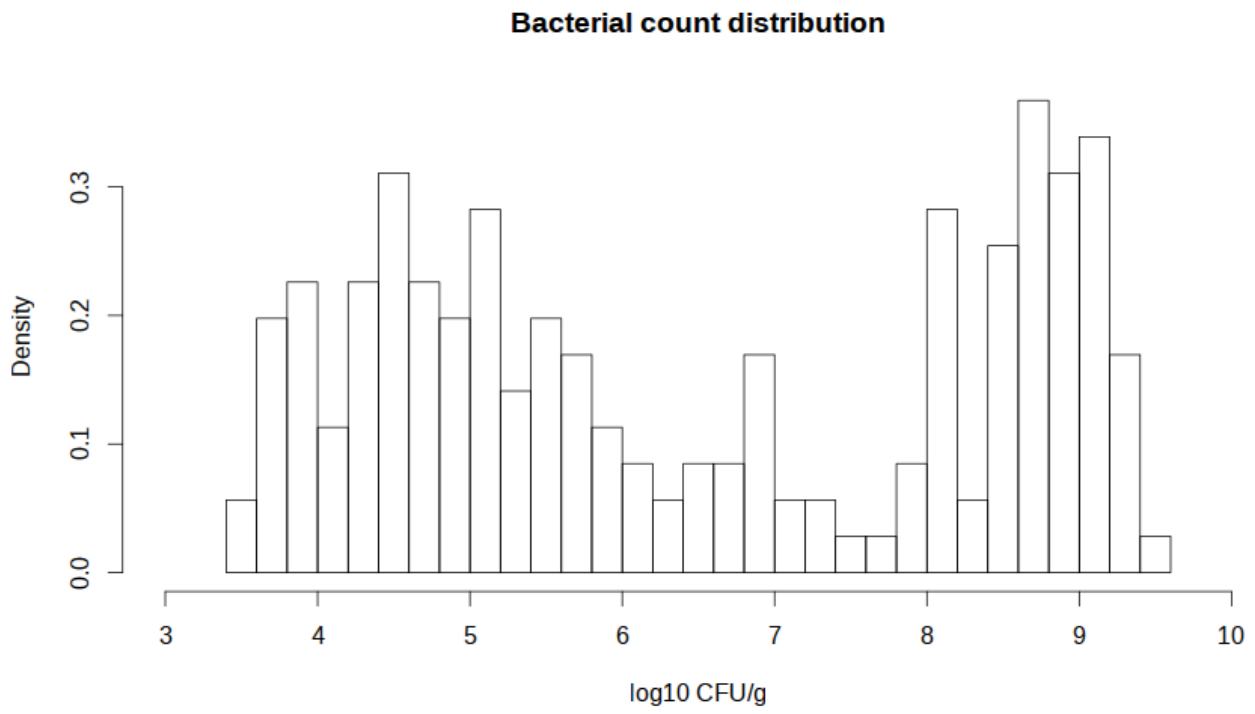
Two files (*EnoseAllSamples.csv* and *MicroCounts.csv*) containing the independent variable dataset (e-nose measurements) and the associated bacterial counts for each sample have been made available for you on Canvas. The bacterial counts refer to the number of colony forming units (CFUs) of Pseudomonads per gram as estimated by growing them on a CFC medium.

Loading the data

As in previous practicals, you should be able to import the data using the `read.table()` function and merge the independent (e-nose) and dependent (bacterial count) variables using the `merge()` function. Note that the bacterial counts are not factors (which you used previously to represent classes) but *numeric* variables, so you should avoid converting them the way you did for classification.

Quick look at the data

One part of the data set used should be quite familiar to you, but the other is new. When confronted with a new data set, it is good to try to visualise it to get a feeling for what you are dealing with. One simple approach is visualising the frequency distribution, which can be done in multiple ways, e.g. using histograms (try the `hist()` function) or box plots (try the `boxplot()` function).



As bacterial growth is (for certain parameters) exponential, the measured CFU values vary widely. It is often preferable to express such values in a logarithmic scale. You can convert the bacterial counts to a logarithmic scale using the `log10()` function:

```
all.data$CFC <- log10(all.data$CFC)
```

Splitting the data

We will continue working with supervised learning algorithms, so you will need to split the dataset into a randomly selected training and test dataset. This can be accomplished using the `createDataPartition()` function from the caret package. A 70% / 30% split as in the previous practical should work fine.

Multiple linear regression

The most basic commonly used regression technique is multiple linear regression, which attempts to fit a linear equation to the provided data, with one coefficient per variable

You can use the `lm()` function to fit a linear model to your training data. For all variables this can be done using "`CFC ~ .`" as the formula, but you can be more selective (e.g. "`CFC ~ DF1 + DF3`" to use only the DF1 and DF3 variables).

```
model.fit <- lm(CFC ~ ., data=data.train)
```

Note that depending on whether or not you have converted the bacterial counts to a logarithmic scale, your results will be different. You can also do this conversion directly in the formula, e.g. "`log10(CFC) ~ .` ":

```
model.fit <- lm(log10(CFC) ~ ., data=data.train)
```

You can view a summary of your fit by passing the output of the `lm()` function to the `summary()` function. Try summarizing both the results of fitting the model to raw bacterial counts (as they appear in the original CSV file) and to bacterial counts on a logarithmic scale. Is there any difference? What does that tell you about the possible relationships between your variables?

```
Residuals:
    Min      1Q  Median      3Q     Max
-3.3326 -1.1254 -0.1626  1.1311  3.4658

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 23.126882   3.681358   6.282 6.03e-09 ***
DF1         -0.008652   0.006883  -1.257  0.21124
DF1.1       -0.032441   0.008024  -4.043 9.53e-05 ***
DF3          0.002169   0.007325   0.296  0.76771
DF4          0.024765   0.008207   3.018  0.00313 **
DF5          0.020710   0.006563   3.156  0.00204 **
```

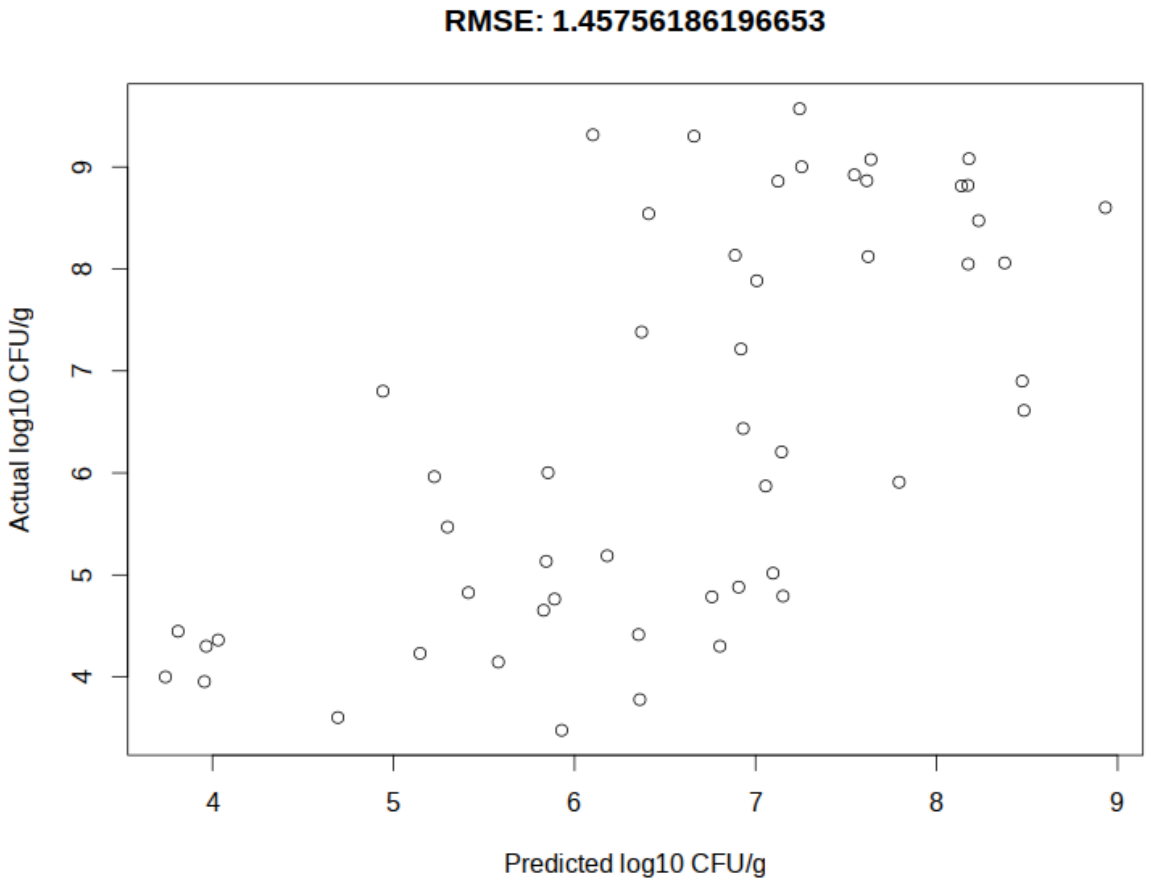
```
DF6          -0.046215   0.007920  -5.835 4.96e-08 ***
DF7           0.015478   0.005916   2.616  0.01007 *
DF8          -0.037807   0.013700  -2.760  0.00673 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.562 on 116 degrees of freedom
Multiple R-squared:  0.3802,     Adjusted R-squared:  0.3374
F-statistic: 8.893 on 8 and 116 DF,  p-value: 1.928e-09
```

Variable importance

'Variable importance' is a commonly used metric used to asses the relative importance of independent variables to your model. You can use the **`varImp()`** function from the caret package to find the variable importance for a model. What value in fact constitutes the importance varies by model – for multiple linear regression it is the absolute value of the *t*-statistic.

Assessing fit quality

You can use the **`predict()`** function to produce predicted bacterial count values for your test data. You can then visualise the result of fitting by creating a scatter plot of predicted vs actual values:



A standard metric (the lower, the better) for assessing fit quality is RMSE, the root mean square error, which you can generate using the `RMSE()` function.

Can you improve the model by using a subset of independent variables rather than all of them? You will need to manually specify the variables used in the formula.

Eliminating variables in multiple linear regression can improve the fit. In particular, it is often good to eliminate (by selecting only one) groups of highly collinear, correlated variables, as one of the primary assumptions of multiple linear regression is the lack of such relationships. However, for our current data that is unlikely to yield much improvement. Why might that be?

Model tuning

The caret package contains the `train()` function, which can serve as a single interface for automating the tuning of both classification and regression models. For example, an approach similar to the manual variable selection you attempted in the previous step can be carried out automatically by specifying the `lmStepAIC` method in the `train()` function. Note that this method requires the MASS package.

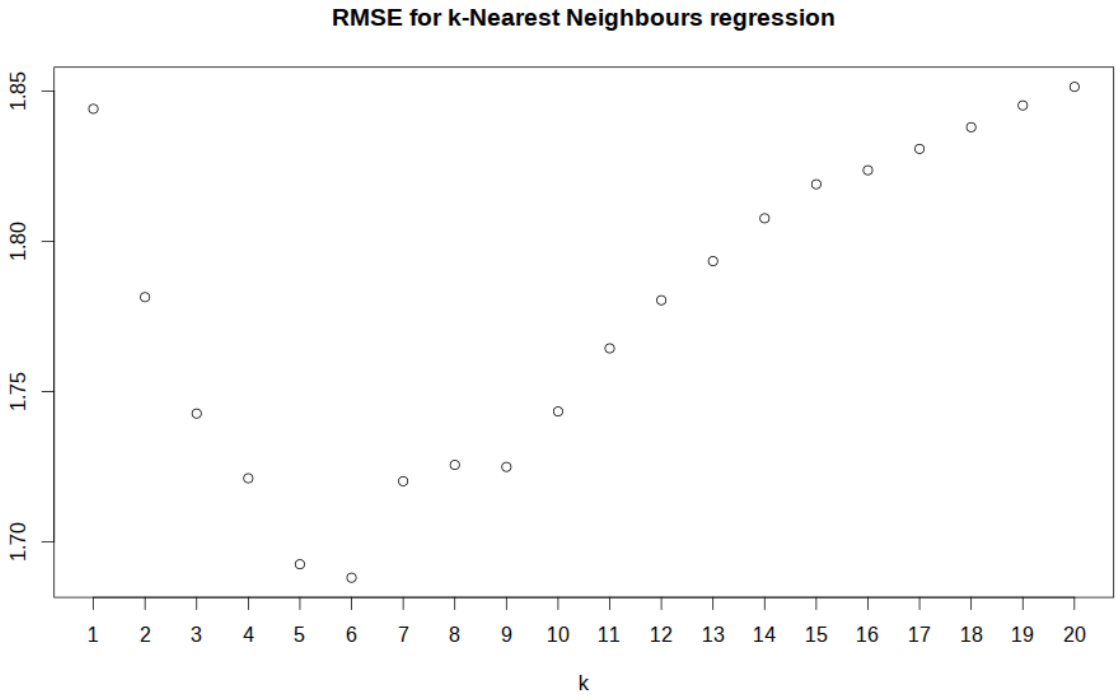
```
model.fit <- train(CFC ~ ., method='lmStepAIC', data=data.train)
```

The `train()` function attempts to tune models by testing all possible combinations of their tuning parameters and minimizing some fit quality metric (such as RMSE). Each method generally has a set of default values their tuning parameters can take, but you can specify your own using the `tuneGrid` parameter (e.g. by setting it to `expand.grid(k=1:20)` for a k-nearest-neighbours model):

```
model.fit <- train(CFC ~ ., method='knn', data=data.train,
                   tuneGrid=expand.grid(k=1:20))
```

Try finding the optimal regression parameters for the methods you are already familiar with (e.g. the k parameter for k-Nearest Neighbours, as shown in the plot on the next page).

An exhaustive list of models available for regression/classification tuning (with their associated libraries and tuning parameters) is available on https://topepo.github.io/caret/available-models.html. If you have time, try some models we have not tested yet, such as svmLinear3 or lars (a variant of LASSO), and see how they perform.



Optional: *Try to merge the sample classes from the Tuesday practical and plot the frequency distribution for each class separately. This should give you an interesting view of the relationship between bacterial growth and spoilage. Could one be used as a proxy for the other?*