4. Reset the seed to set.seed(9910314, kind="Mersenne-Twister"). Tune the KNN

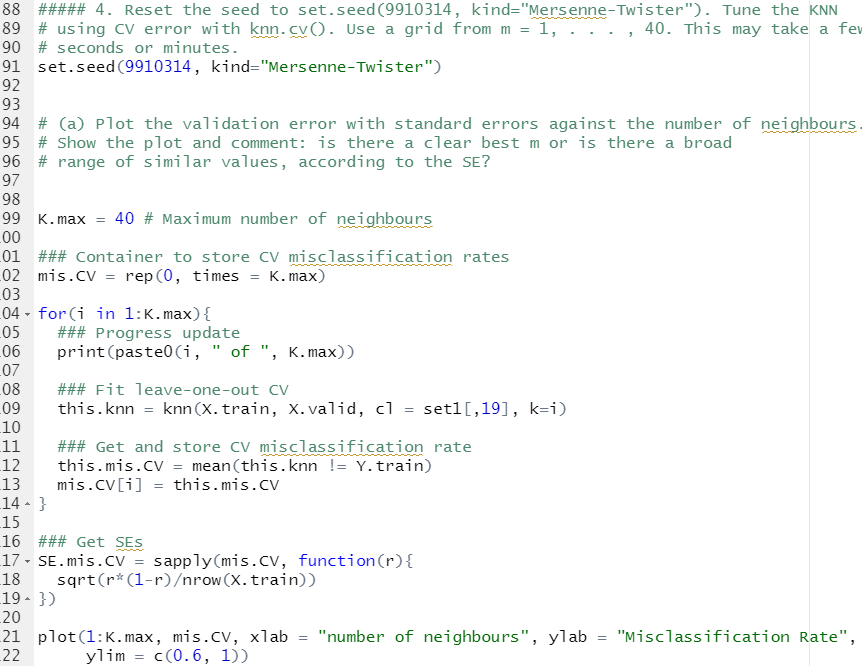
using CV error with knn.cv(). Use a grid from *m* = 1*, . . . ,* 40. This may take a few

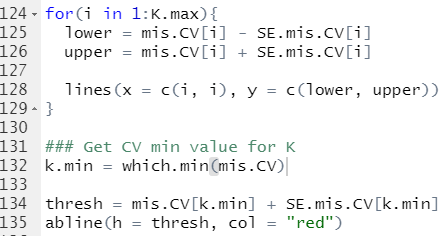
seconds or minutes.

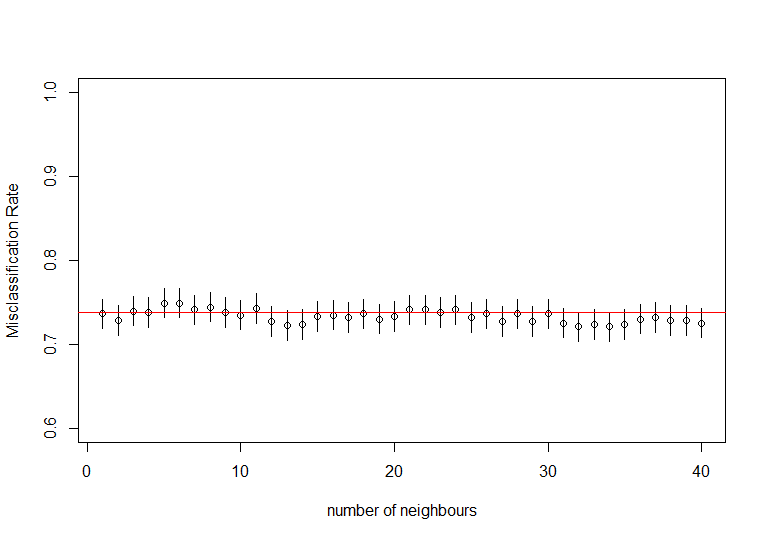
(a) Plot the validation error with standard errors against the number of neighbours.

**Show the plot and comment: is there a clear best** *m* **or is there a broad**

**range of similar values, according to the SE?**







* The error is the lowest at k=32 but there is no clear difference.

(b) Report the value of *m* with lowest error, as well as the one selected by the 1SE

rule.

M=32





(c) Compute the test misclassification rate with both of these parameter values. **Report**

**both error rates, using only one more digit than the first digit in**

**their standard errors.** (For example, if the SE is 0.00375, the first digit in the

SE 3 after the decimal, so report error to 4 after the decimal.)

Keep these error rates handy. You will use them for comparing all classification

methods.

It would be better to use a more rigourous method like multiple reps of CV

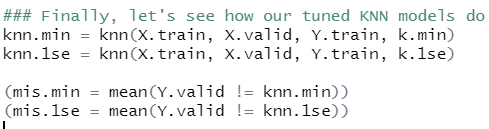
for computing error rates to make an “arena” for comparing methods. Classification

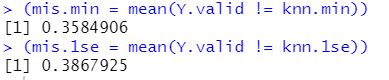
is no different from regression in this way. However, this process is time

consuming, and you have already practiced doing this on regression. I want to

keep the classification assignments lighter, because they will accumulate rather

quickly.





* 2 after the decimal