k-Nearest Neighbor: An Introductory Example

Overview

Researchers in the social sciences often have multivariate data, and want to make predictions or groupings based on certain aspects of their data.

This tutorial will provide code to conduct k-nearest neighbors (https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm) (k-NN) for both classification and regression problems using a data set from the University of California - Irvine's machine learning respository. Specifically, we are using a cross-sectional dataset measuring student math achievement (http://archive.ics.uci.edu/ml/datasets/student+performance) in two Portuguese secondary schools from N = 395 students. The data include student grades, demographic, social, and school related features, and were collected using school reports and questionnaires. We will use k-NN classification to predict mother's job and we will use k-NN regression to predict students' absences. Both examples will use all of the other variables in the data set as predictors; however, variables should be selected based upon theory. In this case, we utilize all variables to demonstrate how to work with different types of variables and discuss issues of dimensionality.

Outline

1. Read in data and needed packages.
2. k-NN classification.
    - Prepare data set for k-NN.
    - k-NN classification using `class` package.
    - k-NN classification using `caret` package.
3. k-NN regression.
    - Prepare data set for k-NN.
    - k-NN regression using `FNN` package.
4. Cautions.
5. Conclusions.

Introduction to k-Nearest Neighbors

k-Nearest Neighbors (k-NN) is an algorithm that is useful for making classifications/predictions when there are potential non-linear boundaries separating classes or values of interest. Conceptually, k-NN examines the classes/values of the points around it (i.e., its neighbors) to determine the value of the point of interest. The majority or average value will be assigned to the point of interest.

Note: We use k-NN classification when predicting a **categorical** outcome, and k-NN regression when predicting a **continuous** outcome.

Step 1: Read in data and needed packages.

```
# Read in the data.

#set filepath for data file
filepath <- "https://quantdev.ssri.psu.edu/sites/qdev/files/student-mat.csv"
#read in the .csv file using the url() function
data <- read.table(file=url(filepath),sep=";",header=TRUE)

#change all variable names to lowercase
var.names.data <-tolower(colnames(data))
colnames(data) <- var.names.data
head(data)
```

```
##   school sex age address famsize pstatus medu fedu     mjob     fjob
## 1     GP   F  18       U     GT3       A    4    4  at_home  teacher
## 2     GP   F  17       U     GT3       T    1    1  at_home    other
## 3     GP   F  15       U     LE3       T    1    1  at_home    other
## 4     GP   F  15       U     GT3       T    4    2   health services
## 5     GP   F  16       U     GT3       T    3    3    other    other
## 6     GP   M  16       U     LE3       T    4    3 services    other
##       reason guardian traveltime studytime failures schoolsup famsup paid
## 1     course   mother          2         2        0       yes     no   no
## 2     course   father          1         2        0        no    yes   no
## 3      other   mother          1         2        3       yes     no  yes
## 4       home   mother          1         3        0        no    yes  yes
## 5       home   father          1         2        0        no    yes  yes
## 6 reputation   mother          1         2        0        no    yes  yes
##   activities nursery higher internet romantic famrel freetime goout dalc
## 1         no     yes    yes       no       no      4        3     4    1
## 2         no      no    yes      yes       no      5        3     3    1
## 3         no     yes    yes      yes       no      4        3     2    2
## 4        yes     yes    yes      yes      yes      3        2     2    1
## 5         no     yes    yes       no       no      4        3     2    1
## 6        yes     yes    yes      yes       no      5        4     2    1
##   walc health absences g1 g2 g3
## 1    1      3        6  5  6  6
## 2    1      3        4  5  5  6
## 3    3      3       10  7  8 10
## 4    1      5        2 15 14 15
## 5    2      5        4  6 10 10
## 6    2      5       10 15 15 15
```

```
#Libraries needed
library(caret)
library(class)
library(dplyr)
library(e1071)
library(FNN)
library(gmodels)
library(psych)
```

Step 2. k-NN classification.

For k-NN classification, we are going to predict the categorical variable mother's job ("mjob") using all the other variables within the data set.

Data preparation.

We will make a copy of our data set so that we can prepare it for our k-NN classification.

```
data_class <- data
```

Next, we will put our outcome variable, mother's job ("mjob"), into its own object and remove it from the data set.

```
# put outcome in its own object
mjob_outcome <- data_class %>% select(mjob)

# remove original variable from the data set
data_class <- data_class %>% select(-mjob)
```

Note that because k-NN involves calculating *distances* between datapoints, we must use numeric variables only. This only applies to the predictor variables. The outcome variable for k-NN classification should remain a factor variable.

First, we scale the data just in case our features are on different metrics. For example, if we had "income" as a variable, it would be on a much larger scale than "age", which could be problematic given the k-NN relies on distances. Note that we are using the 'scale' function here, which means we are scaling to a z-score metric.

Determine which variables are integers.

```
str(data_class)
```

```
## 'data.frame':    395 obs. of  32 variables:
##  $ school    : Factor w/ 2 levels "GP","MS": 1 1 1 1 1 1 1 1 1 1 ...
##  $ sex       : Factor w/ 2 levels "F","M": 1 1 1 1 1 2 2 1 2 2 ...
##  $ age       : int  18 17 15 15 16 16 16 17 15 15 ...
##  $ address   : Factor w/ 2 levels "R","U": 2 2 2 2 2 2 2 2 2 2 ...
##  $ famsize   : Factor w/ 2 levels "GT3","LE3": 1 1 2 1 1 2 2 1 2 1 ...
##  $ pstatus   : Factor w/ 2 levels "A","T": 1 2 2 2 2 2 2 1 1 2 ...
##  $ medu      : int  4 1 1 4 3 4 2 4 3 3 ...
##  $ fedu      : int  4 1 1 2 3 3 2 4 2 4 ...
##  $ fjob      : Factor w/ 5 levels "at_home","health",..: 5 3 3 4 3 3 3 5 3 3 ...
##  $ reason    : Factor w/ 4 levels "course","home",..: 1 1 3 2 2 4 2 2 2 2 ...
##  $ guardian  : Factor w/ 3 levels "father","mother",..: 2 1 2 2 1 2 2 2 2 2 ...
##  $ traveltime: int  2 1 1 1 1 1 1 2 1 1 ...
##  $ studytime : int  2 2 2 3 2 2 2 2 2 2 ...
##  $ failures  : int  0 0 3 0 0 0 0 0 0 0 ...
##  $ schoolsup : Factor w/ 2 levels "no","yes": 2 1 2 1 1 1 1 2 1 1 ...
##  $ famsup    : Factor w/ 2 levels "no","yes": 1 2 1 2 2 2 1 2 2 2 ...
##  $ paid      : Factor w/ 2 levels "no","yes": 1 1 2 2 2 2 1 1 2 2 ...
##  $ activities: Factor w/ 2 levels "no","yes": 1 1 1 2 1 2 1 1 1 2 ...
##  $ nursery   : Factor w/ 2 levels "no","yes": 2 1 2 2 2 2 2 2 2 2 ...
##  $ higher    : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ internet  : Factor w/ 2 levels "no","yes": 1 2 2 2 1 2 2 1 2 2 ...
##  $ romantic  : Factor w/ 2 levels "no","yes": 1 1 1 2 1 1 1 1 1 1 ...
##  $ famrel    : int  4 5 4 3 4 5 4 4 4 5 ...
##  $ freetime  : int  3 3 3 2 3 4 4 1 2 5 ...
##  $ goout     : int  4 3 2 2 2 2 4 4 2 1 ...
##  $ dalc      : int  1 1 2 1 1 1 1 1 1 1 ...
##  $ walc      : int  1 1 3 1 2 2 1 1 1 1 ...
##  $ health    : int  3 3 3 5 5 5 3 1 1 5 ...
##  $ absences  : int  6 4 10 2 4 10 0 6 0 0 ...
##  $ g1        : int  5 5 7 15 6 15 12 6 16 14 ...
##  $ g2        : int  6 5 8 14 10 15 12 5 18 15 ...
##  $ g3        : int  6 6 10 15 10 15 11 6 19 15 ...
```

We see that the variables “age,” “medu,” “fedu,” “traveltime,” “studytime,” “failures,” “famrel,” “freetime,” “goout,” “dalc,” “walc,” “health,” “absences,” “g1,” “g2,” and “g3” are interger variables, which means they can be scaled.

```
data_class[, c("age", "medu", "fedu", "traveltime", "studytime", "failures", "famrel", "freetime", "goout", "dalc", "walc", "health", "absences", "g1", "g2", "g3")] <- scale(data_class[, c("age", "medu", "fedu", "traveltime", "studytime", "failures", "famrel", "freetime", "goout", "dalc", "walc", "health", "absences", "g1", "g2", "g3")])

head(data_class)
```

```
##   school sex        age address famsize pstatus       medu       fedu
## 1     GP   F  1.0217506       U     GT3       A  1.1424068  1.3586476
## 2     GP   F  0.2380778       U     GT3       T -1.5979820 -1.3981972
## 3     GP   F -1.3292678       U     LE3       T -1.5979820 -1.3981972
## 4     GP   F -1.3292678       U     GT3       T  1.1424068 -0.4792490
## 5     GP   F -0.5455950       U     GT3       T  0.2289439  0.4396993
## 6     GP   M -0.5455950       U     LE3       T  1.1424068  0.4396993
##       fjob     reason guardian traveltime   studytime   failures schoolsup
## 1  teacher     course   mother  0.7912473 -0.04223229 -0.4493737       yes
## 2    other     course   father -0.6424347 -0.04223229 -0.4493737        no
## 3    other      other   mother -0.6424347 -0.04223229  3.5847768       yes
## 4 services       home   mother -0.6424347  1.14932149 -0.4493737        no
## 5    other       home   father -0.6424347 -0.04223229 -0.4493737        no
## 6    other reputation   mother -0.6424347 -0.04223229 -0.4493737        no
##   famsup paid activities nursery higher internet romantic      famrel
## 1     no   no         no     yes    yes       no       no  0.06211528
## 2    yes   no         no      no    yes      yes       no  1.17736694
## 3     no  yes         no     yes    yes      yes       no  0.06211528
## 4    yes  yes        yes     yes    yes      yes      yes -1.05313638
## 5    yes  yes         no     yes    yes       no       no  0.06211528
## 6    yes  yes        yes     yes    yes      yes       no  1.17736694
##     freetime       goout       dalc       walc     health    absences
## 1 -0.2357113  0.80046413 -0.5400138 -1.0025178 -0.3987837  0.03637833
## 2 -0.2357113 -0.09778397 -0.5400138 -1.0025178 -0.3987837 -0.21352497
## 3 -0.2357113 -0.99603207  0.5826465  0.5504019 -0.3987837  0.53618492
## 4 -1.2368505 -0.99603207 -0.5400138 -1.0025178  1.0397512 -0.46342827
## 5 -0.2357113 -0.99603207 -0.5400138 -0.2260579  1.0397512 -0.21352497
## 6  0.7654280 -0.99603207 -0.5400138 -0.2260579  1.0397512  0.53618492
##          g1         g2          g3
## 1 -1.780209 -1.2532017 -0.96371171
## 2 -1.780209 -1.5190528 -0.96371171
## 3 -1.177653 -0.7214996 -0.09062427
## 4  1.232570  0.8736068  1.00073503
## 5 -1.478931 -0.1897975 -0.09062427
## 6  1.232570  1.1394578  1.00073503
```

Second, we need to dummy code any factor or categorical variables.

Examine the structure of the data to determine which variables need to be dummy coded.

```
str(data_class)
```

```
## 'data.frame':    395 obs. of  32 variables:
##  $ school    : Factor w/ 2 levels "GP","MS": 1 1 1 1 1 1 1 1 1 1 ...
##  $ sex       : Factor w/ 2 levels "F","M": 1 1 1 1 1 2 2 1 2 2 ...
##  $ age       : num  1.022 0.238 -1.329 -1.329 -0.546 ...
##  $ address   : Factor w/ 2 levels "R","U": 2 2 2 2 2 2 2 2 2 2 ...
##  $ famsize   : Factor w/ 2 levels "GT3","LE3": 1 1 2 1 1 2 2 1 2 1 ...
##  $ pstatus   : Factor w/ 2 levels "A","T": 1 2 2 2 2 2 2 1 1 2 ...
##  $ medu      : num  1.142 -1.598 -1.598 1.142 0.229 ...
##  $ fedu      : num  1.359 -1.398 -1.398 -0.479 0.44 ...
##  $ fjob      : Factor w/ 5 levels "at_home","health",..: 5 3 3 4 3 3 3 5 3 3 ...
##  $ reason    : Factor w/ 4 levels "course","home",..: 1 1 3 2 2 4 2 2 2 2 ...
##  $ guardian  : Factor w/ 3 levels "father","mother",..: 2 1 2 2 1 2 2 2 2 2 ...
##  $ traveltime: num  0.791 -0.642 -0.642 -0.642 -0.642 ...
##  $ studytime : num  -0.0422 -0.0422 -0.0422 1.1493 -0.0422 ...
##  $ failures  : num  -0.449 -0.449 3.585 -0.449 -0.449 ...
##  $ schoolsup : Factor w/ 2 levels "no","yes": 2 1 2 1 1 1 1 2 1 1 ...
##  $ famsup    : Factor w/ 2 levels "no","yes": 1 2 1 2 2 2 1 2 2 2 ...
##  $ paid      : Factor w/ 2 levels "no","yes": 1 1 2 2 2 2 1 1 2 2 ...
##  $ activities: Factor w/ 2 levels "no","yes": 1 1 1 2 1 2 1 1 1 2 ...
##  $ nursery   : Factor w/ 2 levels "no","yes": 2 1 2 2 2 2 2 2 2 2 ...
##  $ higher    : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ internet  : Factor w/ 2 levels "no","yes": 1 2 2 2 1 2 2 1 2 2 ...
##  $ romantic  : Factor w/ 2 levels "no","yes": 1 1 1 2 1 1 1 1 1 1 ...
##  $ famrel    : num  0.0621 1.1774 0.0621 -1.0531 0.0621 ...
##  $ freetime  : num  -0.236 -0.236 -0.236 -1.237 -0.236 ...
##  $ goout     : num  0.8005 -0.0978 -0.996 -0.996 -0.996 ...
##  $ dalc      : num  -0.54 -0.54 0.583 -0.54 -0.54 ...
##  $ walc      : num  -1.003 -1.003 0.55 -1.003 -0.226 ...
##  $ health    : num  -0.399 -0.399 -0.399 1.04 1.04 ...
##  $ absences  : num  0.0364 -0.2135 0.5362 -0.4634 -0.2135 ...
##  $ g1        : num  -1.78 -1.78 -1.18 1.23 -1.48 ...
##  $ g2        : num  -1.253 -1.519 -0.721 0.874 -0.19 ...
##  $ g3        : num  -0.9637 -0.9637 -0.0906 1.0007 -0.0906 ...
```

We can see that the variables “fjob,” “reason,” and “guardian” are factor variables that have three or more levels, and that the variables “school,” “sex,” “address,” “famsize,” “pstatus,” “schoolsup,” “famsup,” “paid,” “activities,” “nursery,” “higher,” “internet,” and “romantic” are factor variables that have only two levels.

We now dummy code variables that have just two levels and are coded 1/0.

```
data_class$schoolsup <- ifelse(data_class$schoolsup == "yes", 1, 0)
data_class$famsup <- ifelse(data_class$famsup == "yes", 1, 0)
data_class$paid <- ifelse(data_class$paid == "yes", 1, 0)
data_class$activities <- ifelse(data_class$activities == "yes", 1, 0)
data_class$nursery <- ifelse(data_class$nursery == "yes", 1, 0)
data_class$higher <- ifelse(data_class$higher == "yes", 1, 0)
data_class$internet <- ifelse(data_class$internet == "yes", 1, 0)
data_class$romantic <- ifelse(data_class$romantic == "yes", 1, 0)
```

Then dummy code variables that have two levels, but are not numeric.

```
data_class$school <- dummy.code(data_class$school)
data_class$sex <- dummy.code(data_class$sex)
data_class$address <- dummy.code(data_class$address)
data_class$famsize <- dummy.code(data_class$famsize)
data_class$pstatus <- dummy.code(data_class$pstatus)
```

Next we dummy code variables that have three or more levels.

```
fjob <- as.data.frame(dummy.code(data_class$fjob))
reason <- as.data.frame(dummy.code(data_class$reason))
guardian <- as.data.frame(dummy.code(data_class$guardian))
```

Rename “other” columns in “fjob”, “reason,” and “guardian,” and rename “health” in “fjob” (so we don’t have duplicate columns later).

```
fjob <- rename(fjob, other_fjob = other)
fjob <- rename(fjob, health_fjob = health)

reason <- rename(reason, other_reason = other)

guardian <- rename(guardian, other_guardian = other)
```

Combine new dummy variables with original data set.

```
data_class <- cbind(data_class, fjob, guardian, reason)
```

Remove original variables that had to be dummy coded.

```
data_class <- data_class %>% select(-one_of(c("fjob", "guardian", "reason")))

head(data_class)
```

```
##   school.GP school.MS sex.F sex.M        age address.R address.U
## 1         1         0     1     0  1.0217506         0         1
## 2         1         0     1     0  0.2380778         0         1
## 3         1         0     1     0 -1.3292678         0         1
## 4         1         0     1     0 -1.3292678         0         1
## 5         1         0     1     0 -0.5455950         0         1
## 6         1         0     0     1 -0.5455950         0         1
##   famsize.GT3 famsize.LE3 pstatus.A pstatus.T       medu       fedu
## 1           1           0         1         0  1.1424068  1.3586476
## 2           1           0         0         1 -1.5979820 -1.3981972
## 3           0           1         0         1 -1.5979820 -1.3981972
## 4           1           0         0         1  1.1424068 -0.4792490
## 5           1           0         0         1  0.2289439  0.4396993
## 6           0           1         0         1  1.1424068  0.4396993
##   traveltime   studytime   failures schoolsup famsup paid activities
## 1  0.7912473 -0.04223229 -0.4493737         1      0    0          0
## 2 -0.6424347 -0.04223229 -0.4493737         0      1    0          0
## 3 -0.6424347 -0.04223229  3.5847768         1      0    1          0
## 4 -0.6424347  1.14932149 -0.4493737         0      1    1          1
## 5 -0.6424347 -0.04223229 -0.4493737         0      1    1          0
## 6 -0.6424347 -0.04223229 -0.4493737         0      1    1          1
##   nursery higher internet romantic      famrel   freetime       goout
## 1       1      1        0        0  0.06211528 -0.2357113  0.80046413
## 2       0      1        1        0  1.17736694 -0.2357113 -0.09778397
## 3       1      1        1        0  0.06211528 -0.2357113 -0.99603207
## 4       1      1        1        1 -1.05313638 -1.2368505 -0.99603207
## 5       1      1        0        0  0.06211528 -0.2357113 -0.99603207
## 6       1      1        1        0  1.17736694  0.7654280 -0.99603207
##         dalc       walc     health    absences        g1         g2
## 1 -0.5400138 -1.0025178 -0.3987837  0.03637833 -1.780209 -1.2532017
## 2 -0.5400138 -1.0025178 -0.3987837 -0.21352497 -1.780209 -1.5190528
## 3  0.5826465  0.5504019 -0.3987837  0.53618492 -1.177653 -0.7214996
## 4 -0.5400138 -1.0025178  1.0397512 -0.46342827  1.232570  0.8736068
## 5 -0.5400138 -0.2260579  1.0397512 -0.21352497 -1.478931 -0.1897975
## 6 -0.5400138 -0.2260579  1.0397512  0.53618492  1.232570  1.1394578
##            g3 at_home health_fjob other_fjob services teacher father
## 1 -0.96371171       0           0          0        0       1      0
## 2 -0.96371171       0           0          1        0       0      1
## 3 -0.09062427       0           0          1        0       0      0
## 4  1.00073503       0           0          0        1       0      0
```

```
## 5 -0.09062427       0           0          1        0       0      1
## 6  1.00073503       0           0          1        0       0      0
##   mother other_guardian course home other_reason reputation
## 1      1              0      1    0            0          0
## 2      0              0      1    0            0          0
## 3      1              0      0    0            1          0
## 4      1              0      0    1            0          0
## 5      0              0      0    1            0          0
## 6      1              0      0    0            0          1
```

Now we're ready for k-NN classification!

We split the data into training and test sets. We partition 75% of the data into the training set and the remaining 25% into the test set.

```
set.seed(1234) # set the seed to make the partition reproducible

# 75% of the sample size
smp_size <- floor(0.75 * nrow(data_class))

train_ind <- sample(seq_len(nrow(data_class)), size = smp_size)

# creating test and training sets that contain all of the predictors
class_pred_train <- data_class[train_ind, ]
class_pred_test <- data_class[-train_ind, ]
```

Split outcome variable into training and test sets using the same partition as above.

```
mjob_outcome_train <- mjob_outcome[train_ind, ]
mjob_outcome_test <- mjob_outcome[-train_ind, ]
```

Use `class` package. Run k-NN classification.

We have to decide on the number of neighbors (k). There are several rules of thumb, one being the square root of the number of observations in the training set. In this case, we select 17 as the number of neighbors, which is approximately the square root of our sample size *N* = 296.

```
mjob_pred_knn <- knn(train = class_pred_train, test = class_pred_test, cl = mjob_outcome_train, k=17)
```

Model evaluation.

```
# put "mjob_outcome_test" in a data frame
mjob_outcome_test <- data.frame(mjob_outcome_test)

# merge "mjob_pred_knn" and "mjob_outcome_test"
class_comparison <- data.frame(mjob_pred_knn, mjob_outcome_test)

# specify column names for "class_comparison"
names(class_comparison) <- c("PredictedMjob", "ObservedMjob")

# inspect "class_comparison"
head(class_comparison)
```

```
##   PredictedMjob ObservedMjob
## 1         other      at_home
## 2         other     services
## 3       teacher       health
## 4      services     services
## 5         other      teacher
## 6      services       health
```

```
# create table examining model accuracy
CrossTable(x = class_comparison$ObservedMjob, y = class_comparison$PredictedMjob, prop.chisq=FALSE, prop.c = FALSE, prop.r =
FALSE, prop.t = FALSE)
```

```
##
##
##   Cell Contents
## |-------------------------|
## |                       N |
## |-------------------------|
##
##
## Total Observations in Table:  99
##
##
##                              | class_comparison$PredictedMjob
## class_comparison$ObservedMjob |   at_home |    health |     other |  services |   teacher | Row Total |
## ------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
##                       at_home |         2 |         0 |        12 |         2 |         0 |        16 |
## ------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
##                        health |         0 |         0 |         5 |         3 |         2 |        10 |
## ------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
##                         other |         1 |         2 |        27 |         4 |         0 |        34 |
## ------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
##                      services |         0 |         0 |        16 |         3 |         3 |        22 |
## ------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
##                       teacher |         0 |         1 |         5 |         3 |         8 |        17 |
## ------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
##                  Column Total |         3 |         3 |        65 |        15 |        13 |        99 |
## ------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
##
##
```

The results of the Cross Table indicate that our model did not predict mother's job very well. To read the Cross Table, we begin by examining the top-left to bottom-right diagonal of the matrix. The diagonal of the matrix represents the number of cases that were correctly classified for each category. If the model correctly classified all cases, the matrix would have zeros everywhere but the diagonal. In this case, we see that the numbers are quite high in the off-diagonals, indicating that our model did not successfully classify our outcome based on our predictors.

To examine the success of the classification given a certain category, one reads across the rows of the matrix. For example, when reading the first row, we see that the model classified 2 of 16 "at home" cases correctly, 12 of 16 "at home" cases as "other," and 2 of 16 "at home" cases as "services."

Use `caret` package. Run k-NN classification.

In this package, the function picks the optimal number of neighbors (k) for you.

```
mjob_pred_caret <- train(class_pred_train, mjob_outcome_train, method = "knn", preProcess = c("center","scale"))
```

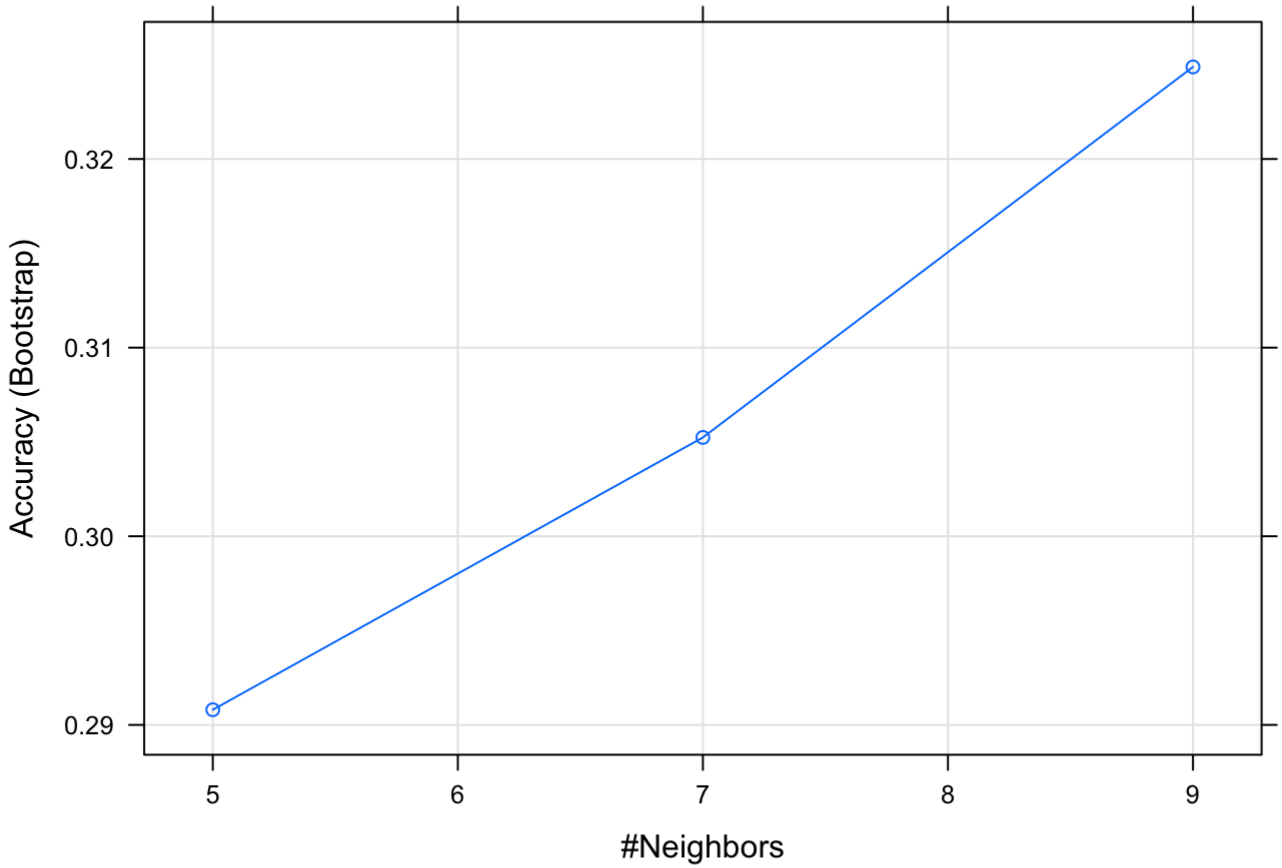
Looking at the output of the `caret` package k-NN model, we can see that it chose k = 9, given that this was the number at which accuracy and kappa peaked.

```
mjob_pred_caret
```

```
## k-Nearest Neighbors
##
## 296 samples
##  41 predictor
##   5 classes: 'at_home', 'health', 'other', 'services', 'teacher'
##
## Pre-processing: centered (36), scaled (36), ignore (5)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 296, 296, 296, 296, 296, 296, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.2908024  0.05318353
##   7  0.3052401  0.06644765
##   9  0.3248805  0.08444429
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```

This plot also shows accuracy peaking at 9.

```
plot(mjob_pred_caret)
```



Next, we compare our predicted values of mjob to our actual values. The confusion matrix gives an indication of how well our model predicted the actual values.

The confusion matrix output also shows overall model statistics and statistics by class

```
knnPredict <- predict(mjob_pred_caret, newdata = class_pred_test)

confusionMatrix(knnPredict, mjob_outcome_test$mjob_outcome_test)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction at_home health other services teacher
##   at_home        3      0     0        1       0
##   health         0      2     0        2       1
##   other         11      2    27       13       3
##   services       1      4     4        2       5
##   teacher        1      2     3        4       8
## 
## Overall Statistics
## 
##                Accuracy : 0.4242          
##                  95% CI : (0.3255, 0.5277)
##     No Information Rate : 0.3434          
##     P-Value [Acc > NIR] : 0.05781         
## 
##                   Kappa : 0.208           
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: at_home Class: health Class: other
## Sensitivity                  0.1875       0.20000       0.7941
## Specificity                  0.9880       0.96629       0.5538
## Pos Pred Value               0.7500       0.40000       0.4821
## Neg Pred Value               0.8632       0.91489       0.8372
## Prevalence                   0.1616       0.10101       0.3434
## Detection Rate               0.0303       0.02020       0.2727
## Detection Prevalence         0.0404       0.05051       0.5657
## Balanced Accuracy            0.5877       0.58315       0.6740
##                      Class: services Class: teacher
## Sensitivity                  0.09091        0.47059
## Specificity                  0.81818        0.87805
## Pos Pred Value               0.12500        0.44444
## Neg Pred Value               0.75904        0.88889
## Prevalence                   0.22222        0.17172
## Detection Rate               0.02020        0.08081
## Detection Prevalence         0.16162        0.18182
## Balanced Accuracy            0.45455        0.67432
```

The model did not perform well - it only successfully classified 42% of the cases correctly. The success of the model can also be evaluated with a variety of other metrics (e.g., sensitivity, specificity, etc.) included here. The interpretation of some of these metrics is a bit difficult given the number of classes (these metrics are best suited for two classes), but you can read more about these metrics here (https://onlinecourses.science.psu.edu/stat507/node/71).

k-NN classification summary

To summarize, we utilized two different packages (`class` and `caret`) to perform k-NN classification, predicting mother's job. Our models may not have accurately predicted our outcome variable for a number of reasons. A large number of our predictor variables were binary or dummy-coded categorical variables, which are not necessarily the most suited for k-NN (to be discussed further in the Cautions section of the tutorial).

Step 3. k-NN regression.

For k-NN regression, we are going to predict students' absences ("absences") using all variables within the data set.

Data preparation.

We will make a copy of our data set so that we can prepare it for our k-NN regression.

```
data_reg <- data
```

Next, we will put our outcome variable, students' absences ("absences"), into its own object and remove it from the data set.

```
# put outcome in its own object
absences_outcome <- data_reg %>% select(absences)

# remove original from the data set
data_reg <- data_reg %>% select(-absences)
```

Note that because k-NN involves calculating *distances* between datapoints, we must use numeric variables only. This only applies to the predictor variables. The outcome variable for k-NN regression should already be a numeric variable.

First, we scale the data just in case our features are on different metrics. For example, if we had "income" as a variable, it would be on a much larger scale than "age", which could be problematic given the k-NN relies on distances. Note that we are using the 'scale' function here, which means we are scaling to a z-score metric.

Determine which variables are integers.

```
str(data_reg)
```

```
## 'data.frame':    395 obs. of  32 variables:
##  $ school    : Factor w/ 2 levels "GP","MS": 1 1 1 1 1 1 1 1 1 1 ...
##  $ sex       : Factor w/ 2 levels "F","M": 1 1 1 1 1 2 2 1 2 2 ...
##  $ age       : int  18 17 15 15 16 16 16 17 15 15 ...
##  $ address   : Factor w/ 2 levels "R","U": 2 2 2 2 2 2 2 2 2 2 ...
##  $ famsize   : Factor w/ 2 levels "GT3","LE3": 1 1 2 1 1 2 2 1 2 1 ...
##  $ pstatus   : Factor w/ 2 levels "A","T": 1 2 2 2 2 2 2 1 1 2 ...
##  $ medu      : int  4 1 1 4 3 4 2 4 3 3 ...
##  $ fedu      : int  4 1 1 2 3 3 2 4 2 4 ...
##  $ mjob      : Factor w/ 5 levels "at_home","health",..: 1 1 1 2 3 4 3 3 4 3 ...
##  $ fjob      : Factor w/ 5 levels "at_home","health",..: 5 3 3 4 3 3 3 5 3 3 ...
##  $ reason    : Factor w/ 4 levels "course","home",..: 1 1 3 2 2 4 2 2 2 2 ...
##  $ guardian  : Factor w/ 3 levels "father","mother",..: 2 1 2 2 1 2 2 2 2 2 ...
##  $ traveltime: int  2 1 1 1 1 1 1 2 1 1 ...
##  $ studytime : int  2 2 2 3 2 2 2 2 2 2 ...
##  $ failures  : int  0 0 3 0 0 0 0 0 0 0 ...
##  $ schoolsup : Factor w/ 2 levels "no","yes": 2 1 2 1 1 1 1 2 1 1 ...
##  $ famsup    : Factor w/ 2 levels "no","yes": 1 2 1 2 2 2 1 2 2 2 ...
##  $ paid      : Factor w/ 2 levels "no","yes": 1 1 2 2 2 2 1 1 2 2 ...
##  $ activities: Factor w/ 2 levels "no","yes": 1 1 1 2 1 2 1 1 1 2 ...
##  $ nursery   : Factor w/ 2 levels "no","yes": 2 1 2 2 2 2 2 2 2 2 ...
##  $ higher    : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ internet  : Factor w/ 2 levels "no","yes": 1 2 2 2 1 2 2 1 2 2 ...
##  $ romantic  : Factor w/ 2 levels "no","yes": 1 1 1 2 1 1 1 1 1 1 ...
##  $ famrel    : int  4 5 4 3 4 5 4 4 4 5 ...
##  $ freetime  : int  3 3 3 2 3 4 4 1 2 5 ...
##  $ goout     : int  4 3 2 2 2 2 4 4 2 1 ...
##  $ dalc      : int  1 1 2 1 1 1 1 1 1 1 ...
##  $ walc      : int  1 1 3 1 2 2 1 1 1 1 ...
##  $ health    : int  3 3 3 5 5 5 3 1 1 5 ...
##  $ g1        : int  5 5 7 15 6 15 12 6 16 14 ...
##  $ g2        : int  6 5 8 14 10 15 12 5 18 15 ...
##  $ g3        : int  6 6 10 15 10 15 11 6 19 15 ...
```

We see that the variables “age,” “medu,” “fedu,” “traveltime,” “studytime,” “failures,” “famrel,” “freetime,” “goout,” “dalc,” “walc,” “health,” “g1,” “g2,” and “g3” are interger variables, which means they can be scaled.

```
data_reg[, c("age", "medu", "fedu", "traveltime", "studytime", "failures", "famrel", "freetime", "goout", "dalc", "walc", "h
ealth", "g1", "g2", "g3")] <- scale(data_reg[, c("age", "medu", "fedu", "traveltime", "studytime", "failures", "famrel", "fr
eetime", "goout", "dalc", "walc", "health", "g1", "g2", "g3")])

head(data_reg)
```

```
##   school sex        age address famsize pstatus       medu       fedu
## 1     GP   F  1.0217506       U     GT3       A  1.1424068  1.3586476
## 2     GP   F  0.2380778       U     GT3       T -1.5979820 -1.3981972
## 3     GP   F -1.3292678       U     LE3       T -1.5979820 -1.3981972
## 4     GP   F -1.3292678       U     GT3       T  1.1424068 -0.4792490
## 5     GP   F -0.5455950       U     GT3       T  0.2289439  0.4396993
## 6     GP   M -0.5455950       U     LE3       T  1.1424068  0.4396993
##       mjob     fjob     reason guardian traveltime   studytime   failures
## 1  at_home  teacher     course   mother  0.7912473 -0.04223229 -0.4493737
## 2  at_home    other     course   father -0.6424347 -0.04223229 -0.4493737
## 3  at_home    other      other   mother -0.6424347 -0.04223229  3.5847768
## 4   health services       home   mother -0.6424347  1.14932149 -0.4493737
## 5    other    other       home   father -0.6424347 -0.04223229 -0.4493737
## 6 services    other reputation   mother -0.6424347 -0.04223229 -0.4493737
##   schoolsup famsup paid activities nursery higher internet romantic
## 1       yes     no   no         no     yes    yes       no       no
## 2        no    yes   no         no      no    yes      yes       no
## 3       yes     no  yes         no     yes    yes      yes       no
## 4        no    yes  yes        yes     yes    yes      yes      yes
## 5        no    yes  yes         no     yes    yes       no       no
## 6        no    yes  yes        yes     yes    yes      yes       no
##        famrel   freetime       goout       dalc       walc     health
## 1  0.06211528 -0.2357113  0.80046413 -0.5400138 -1.0025178 -0.3987837
## 2  1.17736694 -0.2357113 -0.09778397 -0.5400138 -1.0025178 -0.3987837
## 3  0.06211528 -0.2357113 -0.99603207  0.5826465  0.5504019 -0.3987837
## 4 -1.05313638 -1.2368505 -0.99603207 -0.5400138 -1.0025178  1.0397512
## 5  0.06211528 -0.2357113 -0.99603207 -0.5400138 -0.2260579  1.0397512
## 6  1.17736694  0.7654280 -0.99603207 -0.5400138 -0.2260579  1.0397512
##          g1         g2          g3
## 1 -1.780209 -1.2532017 -0.96371171
## 2 -1.780209 -1.5190528 -0.96371171
## 3 -1.177653 -0.7214996 -0.09062427
## 4  1.232570  0.8736068  1.00073503
## 5 -1.478931 -0.1897975 -0.09062427
## 6  1.232570  1.1394578  1.00073503
```

Second, we need to dummy code any factor or categorical variables.

Examine the structure of the data to determine which variables need to be dummy coded.

```
str(data_reg)
```

```
## 'data.frame':    395 obs. of  32 variables:
##  $ school    : Factor w/ 2 levels "GP","MS": 1 1 1 1 1 1 1 1 1 1 ...
##  $ sex       : Factor w/ 2 levels "F","M": 1 1 1 1 1 2 2 1 2 2 ...
##  $ age       : num  1.022 0.238 -1.329 -1.329 -0.546 ...
##  $ address   : Factor w/ 2 levels "R","U": 2 2 2 2 2 2 2 2 2 2 ...
##  $ famsize   : Factor w/ 2 levels "GT3","LE3": 1 1 2 1 1 2 2 1 2 1 ...
##  $ pstatus   : Factor w/ 2 levels "A","T": 1 2 2 2 2 2 2 1 1 2 ...
##  $ medu      : num  1.142 -1.598 -1.598 1.142 0.229 ...
##  $ fedu      : num  1.359 -1.398 -1.398 -0.479 0.44 ...
##  $ mjob      : Factor w/ 5 levels "at_home","health",..: 1 1 1 2 3 4 3 3 4 3 ...
##  $ fjob      : Factor w/ 5 levels "at_home","health",..: 5 3 3 4 3 3 3 5 3 3 ...
##  $ reason    : Factor w/ 4 levels "course","home",..: 1 1 3 2 2 4 2 2 2 2 ...
##  $ guardian  : Factor w/ 3 levels "father","mother",..: 2 1 2 2 1 2 2 2 2 2 ...
##  $ traveltime: num  0.791 -0.642 -0.642 -0.642 -0.642 ...
##  $ studytime : num  -0.0422 -0.0422 -0.0422 1.1493 -0.0422 ...
##  $ failures  : num  -0.449 -0.449 3.585 -0.449 -0.449 ...
##  $ schoolsup : Factor w/ 2 levels "no","yes": 2 1 2 1 1 1 1 2 1 1 ...
##  $ famsup    : Factor w/ 2 levels "no","yes": 1 2 1 2 2 2 1 2 2 2 ...
##  $ paid      : Factor w/ 2 levels "no","yes": 1 1 2 2 2 2 1 1 2 2 ...
##  $ activities: Factor w/ 2 levels "no","yes": 1 1 1 2 1 2 1 1 1 2 ...
##  $ nursery   : Factor w/ 2 levels "no","yes": 2 1 2 2 2 2 2 2 2 2 ...
##  $ higher    : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ internet  : Factor w/ 2 levels "no","yes": 1 2 2 2 1 2 2 1 2 2 ...
##  $ romantic  : Factor w/ 2 levels "no","yes": 1 1 1 2 1 1 1 1 1 1 ...
##  $ famrel    : num  0.0621 1.1774 0.0621 -1.0531 0.0621 ...
##  $ freetime  : num  -0.236 -0.236 -0.236 -1.237 -0.236 ...
##  $ goout     : num  0.8005 -0.0978 -0.996 -0.996 -0.996 ...
##  $ dalc      : num  -0.54 -0.54 0.583 -0.54 -0.54 ...
##  $ walc      : num  -1.003 -1.003 0.55 -1.003 -0.226 ...
##  $ health    : num  -0.399 -0.399 -0.399 1.04 1.04 ...
##  $ g1        : num  -1.78 -1.78 -1.18 1.23 -1.48 ...
##  $ g2        : num  -1.253 -1.519 -0.721 0.874 -0.19 ...
##  $ g3        : num  -0.9637 -0.9637 -0.0906 1.0007 -0.0906 ...
```

We can see that the variables “mjob”, “fjob,” “reason,” and “guardian” are factor variables that have three or more levels, and that the variables “school,” “sex,” “address,” “famsize,” “pstatus,” “schoolsup,” “famsup,” “paid,” “activities,” “nursery,” “higher,” “internet,” and “romantic” are factor variables that have only two levels.

We now dummy code variables that have just two levels and are coded 1/0.

```
data_reg$schoolsup <- ifelse(data_reg$schoolsup == "yes", 1, 0)
data_reg$famsup <- ifelse(data_reg$famsup == "yes", 1, 0)
data_reg$paid <- ifelse(data_reg$paid == "yes", 1, 0)
data_reg$activities <- ifelse(data_reg$activities == "yes", 1, 0)
data_reg$nursery <- ifelse(data_reg$nursery == "yes", 1, 0)
data_reg$higher <- ifelse(data_reg$higher == "yes", 1, 0)
data_reg$internet <- ifelse(data_reg$internet == "yes", 1, 0)
data_reg$romantic <- ifelse(data_reg$romantic == "yes", 1, 0)
```

Then dummy code variables that have two levels, but are not numeric.

```
data_reg$school <- dummy.code(data_reg$school)
data_reg$sex <- dummy.code(data_reg$sex)
data_reg$address <- dummy.code(data_reg$address)
data_reg$famsize <- dummy.code(data_reg$famsize)
data_reg$pstatus <- dummy.code(data_reg$pstatus)
```

Next we dummy code variables that have three or more levels.

```
mjob <- as.data.frame(dummy.code(data_reg$mjob))
fjob <- as.data.frame(dummy.code(data_reg$fjob))
reason <- as.data.frame(dummy.code(data_reg$reason))
guardian <- as.data.frame(dummy.code(data_reg$guardian))
```

Rename “other” columns in “mjob,” “fjob,” “reason,” and “guardian,” and rename “health,” “at_home,” “services,” and “teacher” in “mjob” and “fjob” (so we don’t have duplicate columns later).

```
mjob <- rename(mjob, health_mjob = health)
mjob <- rename(mjob, at_home_mjob = at_home)
mjob <- rename(mjob, services_mjob = services)
mjob <- rename(mjob, teacher_mjob = teacher)
mjob <- rename(mjob, other_mjob = other)

fjob <- rename(fjob, health_fjob = health)
fjob <- rename(fjob, at_home_fjob = at_home)
fjob <- rename(fjob, services_fjob = services)
fjob <- rename(fjob, teacher_fjob = teacher)
fjob <- rename(fjob, other_fjob = other)

reason <- rename(reason, other_reason = other)

guardian <- rename(guardian, other_guardian = other)
```

Combine new dummy variables with original data set.

```
data_reg <- cbind(data_reg, mjob, fjob, guardian, reason)
```

Remove original variables that had to be dummy coded.

```
data_reg <- data_reg %>% select(-one_of(c("mjob", "fjob", "guardian", "reason")))

head(data_reg)
```

```
##   school.GP school.MS sex.F sex.M        age address.R address.U
## 1         1         0     1     0  1.0217506         0         1
## 2         1         0     1     0  0.2380778         0         1
## 3         1         0     1     0 -1.3292678         0         1
## 4         1         0     1     0 -1.3292678         0         1
## 5         1         0     1     0 -0.5455950         0         1
## 6         1         0     0     1 -0.5455950         0         1
##   famsize.GT3 famsize.LE3 pstatus.A pstatus.T       medu       fedu
## 1           1           0         1         0  1.1424068  1.3586476
## 2           1           0         0         1 -1.5979820 -1.3981972
## 3           0           1         0         1 -1.5979820 -1.3981972
## 4           1           0         0         1  1.1424068 -0.4792490
## 5           1           0         0         1  0.2289439  0.4396993
## 6           0           1         0         1  1.1424068  0.4396993
##   traveltime   studytime   failures schoolsup famsup paid activities
## 1  0.7912473 -0.04223229 -0.4493737         1      0    0          0
## 2 -0.6424347 -0.04223229 -0.4493737         0      1    0          0
## 3 -0.6424347 -0.04223229  3.5847768         1      0    1          0
## 4 -0.6424347  1.14932149 -0.4493737         0      1    1          1
## 5 -0.6424347 -0.04223229 -0.4493737         0      1    1          0
## 6 -0.6424347 -0.04223229 -0.4493737         0      1    1          1
##   nursery higher internet romantic      famrel   freetime       goout
## 1       1      1        0        0  0.06211528 -0.2357113  0.80046413
## 2       0      1        1        0  1.17736694 -0.2357113 -0.09778397
## 3       1      1        1        0  0.06211528 -0.2357113 -0.99603207
## 4       1      1        1        1 -1.05313638 -1.2368505 -0.99603207
## 5       1      1        0        0  0.06211528 -0.2357113 -0.99603207
## 6       1      1        1        0  1.17736694  0.7654280 -0.99603207
##         dalc       walc     health        g1         g2          g3
## 1 -0.5400138 -1.0025178 -0.3987837 -1.780209 -1.2532017 -0.96371171
## 2 -0.5400138 -1.0025178 -0.3987837 -1.780209 -1.5190528 -0.96371171
## 3  0.5826465  0.5504019 -0.3987837 -1.177653 -0.7214996 -0.09062427
## 4 -0.5400138 -1.0025178  1.0397512  1.232570  0.8736068  1.00073503
## 5 -0.5400138 -0.2260579  1.0397512 -1.478931 -0.1897975 -0.09062427
## 6 -0.5400138 -0.2260579  1.0397512  1.232570  1.1394578  1.00073503
##   at_home_mjob health_mjob other_mjob services_mjob teacher_mjob
## 1            1           0          0             0            0
## 2            1           0          0             0            0
## 3            1           0          0             0            0
## 4            0           1          0             0            0
```

```
## 5            0           0          1             0            0
## 6            0           0          0             1            0
##   at_home_fjob health_fjob other_fjob services_fjob teacher_fjob father
## 1            0           0          0             0            1      0
## 2            0           0          1             0            0      1
## 3            0           0          1             0            0      0
## 4            0           0          0             1            0      0
## 5            0           0          1             0            0      1
## 6            0           0          1             0            0      0
##   mother other_guardian course home other_reason reputation
## 1      1              0      1    0            0          0
## 2      0              0      1    0            0          0
## 3      1              0      0    0            1          0
## 4      1              0      0    1            0          0
## 5      0              0      0    1            0          0
## 6      1              0      0    0            0          1
```

Now we're ready for k-NN regression!

We split the data into training and test sets. We partition 75% of the data into the training set and the remaining 25% into the test set.

```
set.seed(1234) # set the seed to make the partition reproducible

# 75% of the sample size
smp_size <- floor(0.75 * nrow(data_reg))

train_ind <- sample(seq_len(nrow(data_reg)), size = smp_size)

# creating test and training sets that contain all of the predictors
reg_pred_train <- data_reg[train_ind, ]
reg_pred_test <- data_reg[-train_ind, ]
```

Split outcome variable into training and test sets using the same partition as above.

```
abs_outcome_train <- absences_outcome[train_ind, ]
abs_outcome_test <- absences_outcome[-train_ind, ]
```

Using `FNN` package. Run kNN regression.

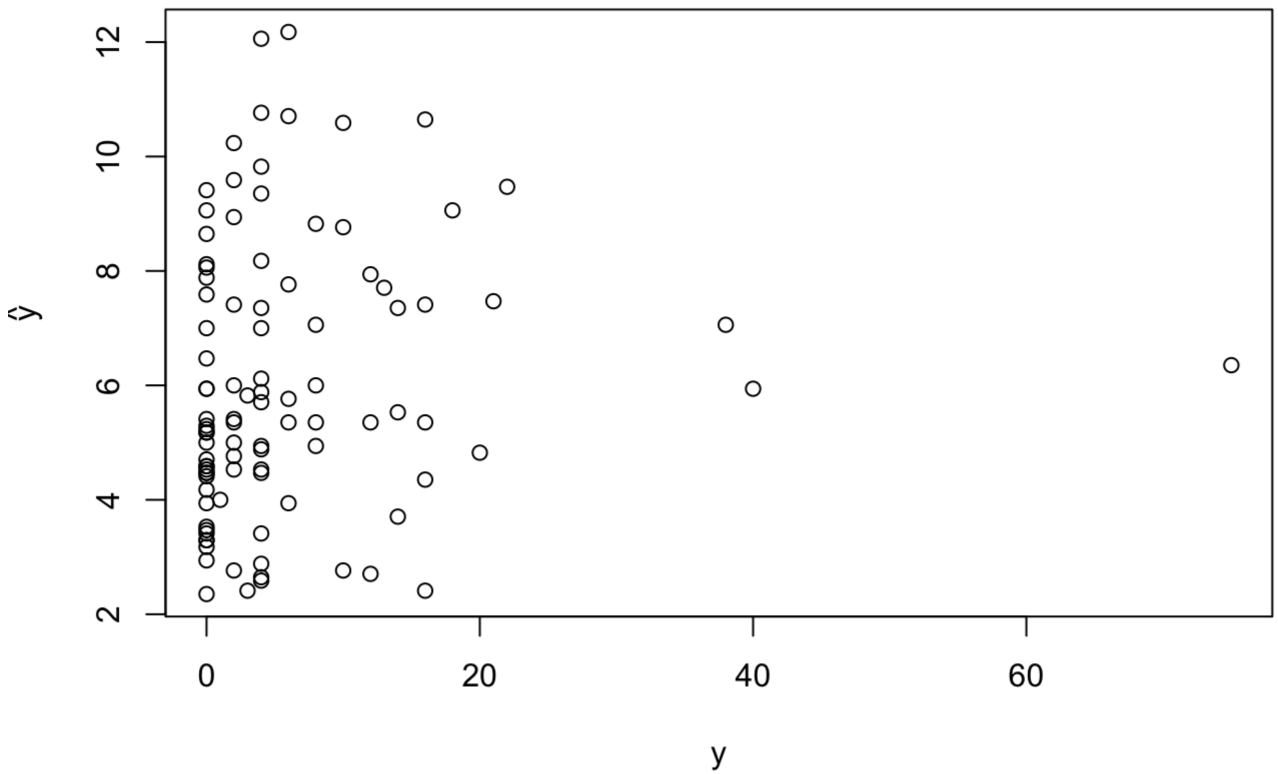
We have to decide on the number of neighbors (k). There are several rules of thumb, one being the square root of the number of observations in the training set. In this case, we select 17 as the number of neighbors, which is approximately the square root of our sample size *N* = 296.

```
reg_results <- knn.reg(reg_pred_train, reg_pred_test, abs_outcome_train, k = 17)
print(reg_results)
```

```
## Prediction:
##  [1]  3.411765  3.176471  6.000000  5.352941  3.294118  2.352941  7.000000
##  [8]  5.882353 10.647059  3.411765  7.058824  6.000000  2.647059  8.058824
## [15] 12.176471  7.352941  9.588235  9.823529  2.705882  7.000000 10.764706
## [22]  5.352941  7.588235  5.000000  4.529412  5.352941  3.529412  9.058824
## [29]  8.647059  4.941176  3.941176  5.294118  4.764706  2.941176  4.411765
## [36]  5.764706  5.411765  6.470588  7.764706  7.411765  2.411765  5.176471
## [43]  6.117647  9.352941  8.176471  8.764706  4.529412  7.941176  2.764706
## [50] 10.705882  2.882353  2.764706  5.176471  3.294118  7.882353  8.941176
## [57]  3.941176  4.470588  9.470588  7.705882  4.529412  3.705882  2.588235
## [64]  5.352941  4.470588  7.411765  7.470588  5.000000 10.235294  6.352941
## [71]  8.823529  4.000000  5.705882  4.882353  5.352941 10.588235  3.470588
## [78]  4.823529  7.058824  5.941176  7.352941  5.941176  9.411765  4.470588
## [85]  4.588235  4.352941  9.058824 12.058824  4.588235  5.941176  8.117647
## [92]  4.941176  2.411765  5.529412  4.705882  5.235294  4.176471  5.411765
## [99]  5.823529
```

Plot the predicted results printed above against the actual values of the outcome variable test set.

```
plot(abs_outcome_test, reg_results$pred, xlab="y", ylab=expression(hat(y)))
```



If the values were perfectly predicted, we would expect to see points along the y = x line (the lower-left to upper-right diagonal if the scales on each axis of the plot are the same). In this case, the prediction does not look great, but let's quantify its accuracy using the mean square error (MSE) and mean absolute error (MAE).

```
#mean square prediction error
mean((abs_outcome_test - reg_results$pred) ^ 2)
```

```
## [1] 102.5213
```

```
#mean absolute error
mean(abs(abs_outcome_test - reg_results$pred))
```

```
## [1] 6.092692
```

The MSE and MAE by themselves are difficult to interpret. They are most useful when comparing model results - the model with the lowest values is a better fitting model.

Cautions.

In this example, the model did not classify or predict our outcome variables of interest very well. Potential issues could be (1) the dimensionality of the data (i.e., the relatively high number of predictors –> 40+ dimensional space), which can create problems given that k-NN relies on a distance metric, and (2) the large number of dummy-coded cateogorical variables used (again, because k-NN is a distance metric, binary variables will result in values that are on opposite sides of that values spectrum, and thus a point's nearest neighbors will be more influenced by those on the same side of the spectrum - see this Cross Validated discussion (https://stats.stackexchange.com/questions/271043/k-nearest-neighbour-with-continuous-and-binary-variables)). Finally, k-NN can be computationally expensive and may not be the best fit for especially large data sets.

Conclusion.

In this tutorial, we have demonstrated how to prepare data for k-NN as well as conduct k-NN classification and k-NN regression.