Practicum 1

Smit Patil and Harsh Janyani

6/3/2020

--- Question 1 ---

Problem 1:

Download the data set Glass Identification Database along with its explanation. Note that the data file does not contain header names; you may wish to add those. The description of each column can be found in the data set explanation. This assignment must be completed within an R Markdown Notebook.

```r
#Importing Data and adding column names using colnames function
library(data.table)
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------------------------------------------ t

## v tibble  3.0.1     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## v purrr   0.3.4

## -- Conflicts ---------------------------------------------------------------------------------- tidyvers
## x dplyr::between()   masks data.table::between()
## x dplyr::filter()    masks stats::filter()
## x dplyr::first()     masks data.table::first()
## x dplyr::lag()       masks stats::lag()
## x dplyr::last()      masks data.table::last()
## x purrr::transpose() masks data.table::transpose()
```

```r
library(dplyr)

#Importing CSV file
glass.data <- read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases/glass/glass.data", hea

#Renaming column names i.e headers
col_names <- c("Id","RI","Na","Mg","Al","Si","K","Ca","Ba","Fe","Type")
colnames(glass.data) <- col_names
```

Problem 2: Explore the data set as you see fit and that allows you to get a sense of the data and get comfortable with it.

```r
#Exploring dataset
head(glass.data)
```

```
##   Id      RI    Na   Mg   Al    Si    K   Ca Ba   Fe Type
## 1  1 1.52101 13.64 4.49 1.10 71.78 0.06 8.75  0 0.00    1
## 2  2 1.51761 13.89 3.60 1.36 72.73 0.48 7.83  0 0.00    1
## 3  3 1.51618 13.53 3.55 1.54 72.99 0.39 7.78  0 0.00    1
```

```
## 4  4 1.51766 13.21 3.69 1.29 72.61 0.57 8.22  0 0.00    1
## 5  5 1.51742 13.27 3.62 1.24 73.08 0.55 8.07  0 0.00    1
## 6  6 1.51596 12.79 3.61 1.62 72.97 0.64 8.07  0 0.26    1
```

```r
#Studying the data types used in the glass data and checking for NA's by using summary
str(glass.data)
```

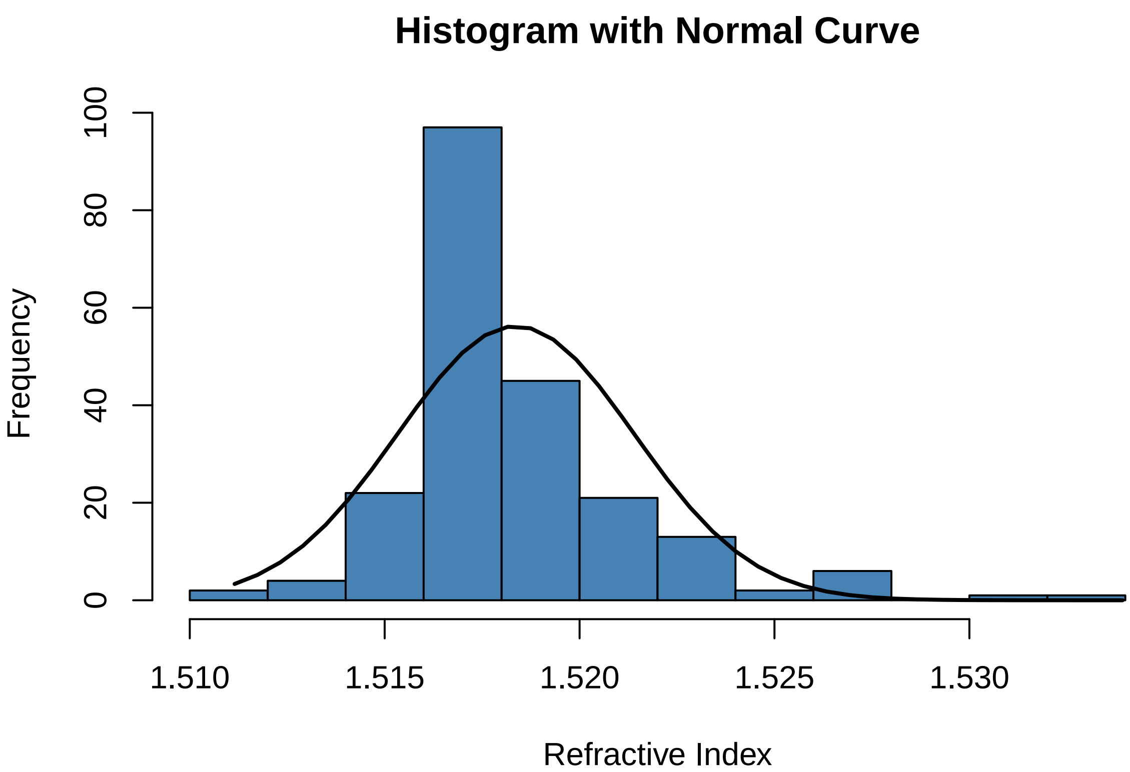
```
## 'data.frame':    214 obs. of  11 variables:
##  $ Id  : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ RI  : num  1.52 1.52 1.52 1.52 1.52 ...
##  $ Na  : num  13.6 13.9 13.5 13.2 13.3 ...
##  $ Mg  : num  4.49 3.6 3.55 3.69 3.62 3.61 3.6 3.61 3.58 3.6 ...
##  $ Al  : num  1.1 1.36 1.54 1.29 1.24 1.62 1.14 1.05 1.37 1.36 ...
##  $ Si  : num  71.8 72.7 73 72.6 73.1 ...
##  $ K   : num  0.06 0.48 0.39 0.57 0.55 0.64 0.58 0.57 0.56 0.57 ...
##  $ Ca  : num  8.75 7.83 7.78 8.22 8.07 8.07 8.17 8.24 8.3 8.4 ...
##  $ Ba  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Fe  : num  0 0 0 0 0 0.26 0 0 0 0.11 ...
##  $ Type: int  1 1 1 1 1 1 1 1 1 1 ...
```

```r
summary(glass.data)
```

```
##        Id               RI              Na              Mg
##  Min.   :  1.00   Min.   :1.511   Min.   :10.73   Min.   :0.000
##  1st Qu.: 54.25   1st Qu.:1.517   1st Qu.:12.91   1st Qu.:2.115
##  Median :107.50   Median :1.518   Median :13.30   Median :3.480
##  Mean   :107.50   Mean   :1.518   Mean   :13.41   Mean   :2.685
##  3rd Qu.:160.75   3rd Qu.:1.519   3rd Qu.:13.82   3rd Qu.:3.600
##  Max.   :214.00   Max.   :1.534   Max.   :17.38   Max.   :4.490
##        Al              Si              K                Ca
##  Min.   :0.290   Min.   :69.81   Min.   :0.0000   Min.   : 5.430
##  1st Qu.:1.190   1st Qu.:72.28   1st Qu.:0.1225   1st Qu.: 8.240
##  Median :1.360   Median :72.79   Median :0.5550   Median : 8.600
##  Mean   :1.445   Mean   :72.65   Mean   :0.4971   Mean   : 8.957
##  3rd Qu.:1.630   3rd Qu.:73.09   3rd Qu.:0.6100   3rd Qu.: 9.172
##  Max.   :3.500   Max.   :75.41   Max.   :6.2100   Max.   :16.190
##        Ba              Fe               Type
##  Min.   :0.000   Min.   :0.00000   Min.   :1.00
##  1st Qu.:0.000   1st Qu.:0.00000   1st Qu.:1.00
##  Median :0.000   Median :0.00000   Median :2.00
##  Mean   :0.175   Mean   :0.05701   Mean   :2.78
##  3rd Qu.:0.000   3rd Qu.:0.10000   3rd Qu.:3.00
##  Max.   :3.150   Max.   :0.51000   Max.   :7.00
```

Problem 3: Create a histogram of column 2 (refractive index) and overlay a normal curve; visually determine whether the data is normally distributed. You may use the code from this tutorial.

```r
#Plotting Histogram of Refractive index with an overlaying normal curve. We make use of hist() function
hist <- hist(glass.data$RI, breaks=10, col="steelBlue", xlab="Refractive Index", main="Histogram with No
xfit <- seq(min(glass.data$RI), max(glass.data$RI), length=40)
yfit <- dnorm(xfit, mea =mean(glass.data$RI), sd=sd(glass.data$RI))

#Using lines() function we plot the normal curve
yfit <- yfit*diff(hist$mids[1:2])*length(glass.data$RI)
lines(xfit, yfit, col="black", lwd=2)
```




```
#Based on visual understanding, we can see that the data is normally distributed.
```

Problem 4: Does the k-NN algorithm require normally distributed data or is it a non-parametric method? Comment on your findings. Answer this in a code block as a comment only.

```
#k-NN algorithm is a non-parametric method, it does not make any assumptioms on the underlying data dis
```

Problem 5: Identify any outliers for the columns using a z-score deviation approach, i.e., consider any values that are more than 2 standard deviations from the mean as outliers. Which are your outliers for each column? What would you do? Do not remove them the outliers.

```
#We have used the z-score deviation approach to calculated the outliers. We calculate for each column b

data <- glass.data
for (i in 2:10)
  {
    mean_data <- mean(data[,i])
    sd_data <- sd(data[,i])
    zscore <- (data[,i]-mean_data)/sd_data
    data[which(!(zscore>2)),i] = NA
    print(col_names[i])
    print(which(is.na(data[,i])==FALSE))
    }
```

```
## [1] "RI"
## [1]  48 104 106 107 108 111 112 113 132
## [1] "Na"
## [1] 185 190 201
## [1] "Mg"
## integer(0)
## [1] "Al"
```

```
##  [1] 164 172 173 193 196 197 199 200 209 210
## [1] "Si"
## [1] 110 181 185 202
## [1] "K"
## [1] 172 173 202
## [1] "Ca"
## [1] 106 107 108 111 112 113 132 171 174
## [1] "Ba"
##  [1] 107 164 186 187 190 194 195 204 206 207 208 211 212 213 214
## [1] "Fe"
##  [1]   6  45  57  72 106 107 119 136 146 163 175 176
```

```
data
```

```
##       Id      RI    Na Mg   Al    Si    K    Ca   Ba   Fe Type
## 1      1      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 2      2      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 3      3      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 4      4      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 5      5      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 6      6      NA    NA NA   NA    NA   NA    NA   NA 0.26    1
## 7      7      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 8      8      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 9      9      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 10    10      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 11    11      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 12    12      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 13    13      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 14    14      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 15    15      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 16    16      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 17    17      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 18    18      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 19    19      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 20    20      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 21    21      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 22    22      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 23    23      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 24    24      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 25    25      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 26    26      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 27    27      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 28    28      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 29    29      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 30    30      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 31    31      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 32    32      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 33    33      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 34    34      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 35    35      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 36    36      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 37    37      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 38    38      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 39    39      NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 40    40      NA    NA NA   NA    NA   NA    NA   NA   NA    1
```

```
## 41   41       NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 42   42       NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 43   43       NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 44   44       NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 45   45       NA    NA NA   NA    NA   NA    NA   NA 0.30    1
## 46   46       NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 47   47       NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 48   48 1.52667    NA NA   NA    NA   NA    NA   NA   NA    1
## 49   49       NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 50   50       NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 51   51       NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 52   52       NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 53   53       NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 54   54       NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 55   55       NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 56   56       NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 57   57       NA    NA NA   NA    NA   NA    NA   NA 0.31    1
## 58   58       NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 59   59       NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 60   60       NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 61   61       NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 62   62       NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 63   63       NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 64   64       NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 65   65       NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 66   66       NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 67   67       NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 68   68       NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 69   69       NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 70   70       NA    NA NA   NA    NA   NA    NA   NA   NA    1
## 71   71       NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 72   72       NA    NA NA   NA    NA   NA    NA   NA 0.32    2
## 73   73       NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 74   74       NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 75   75       NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 76   76       NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 77   77       NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 78   78       NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 79   79       NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 80   80       NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 81   81       NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 82   82       NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 83   83       NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 84   84       NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 85   85       NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 86   86       NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 87   87       NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 88   88       NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 89   89       NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 90   90       NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 91   91       NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 92   92       NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 93   93       NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 94   94       NA    NA NA   NA    NA   NA    NA   NA   NA    2
```

```
## 95   95      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 96   96      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 97   97      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 98   98      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 99   99      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 100 100      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 101 101      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 102 102      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 103 103      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 104 104 1.52725    NA NA   NA    NA   NA    NA   NA   NA    2
## 105 105      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 106 106 1.52475    NA NA   NA    NA   NA 13.24   NA 0.34    2
## 107 107 1.53125    NA NA   NA    NA   NA 13.30 3.15 0.28    2
## 108 108 1.53393    NA NA   NA    NA   NA 16.19   NA   NA    2
## 109 109      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 110 110      NA    NA NA   NA 74.45   NA    NA   NA   NA    2
## 111 111 1.52664    NA NA   NA    NA   NA 14.68   NA   NA    2
## 112 112 1.52739    NA NA   NA    NA   NA 14.96   NA   NA    2
## 113 113 1.52777    NA NA   NA    NA   NA 14.40   NA   NA    2
## 114 114      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 115 115      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 116 116      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 117 117      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 118 118      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 119 119      NA    NA NA   NA    NA   NA    NA   NA 0.29    2
## 120 120      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 121 121      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 122 122      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 123 123      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 124 124      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 125 125      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 126 126      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 127 127      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 128 128      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 129 129      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 130 130      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 131 131      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 132 132 1.52614    NA NA   NA    NA   NA 13.44   NA   NA    2
## 133 133      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 134 134      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 135 135      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 136 136      NA    NA NA   NA    NA   NA    NA   NA 0.28    2
## 137 137      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 138 138      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 139 139      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 140 140      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 141 141      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 142 142      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 143 143      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 144 144      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 145 145      NA    NA NA   NA    NA   NA    NA   NA   NA    2
## 146 146      NA    NA NA   NA    NA   NA    NA   NA 0.35    2
## 147 147      NA    NA NA   NA    NA   NA    NA   NA   NA    3
## 148 148      NA    NA NA   NA    NA   NA    NA   NA   NA    3
```

```
## 149 149      NA    NA NA   NA    NA   NA    NA   NA   NA    3
## 150 150      NA    NA NA   NA    NA   NA    NA   NA   NA    3
## 151 151      NA    NA NA   NA    NA   NA    NA   NA   NA    3
## 152 152      NA    NA NA   NA    NA   NA    NA   NA   NA    3
## 153 153      NA    NA NA   NA    NA   NA    NA   NA   NA    3
## 154 154      NA    NA NA   NA    NA   NA    NA   NA   NA    3
## 155 155      NA    NA NA   NA    NA   NA    NA   NA   NA    3
## 156 156      NA    NA NA   NA    NA   NA    NA   NA   NA    3
## 157 157      NA    NA NA   NA    NA   NA    NA   NA   NA    3
## 158 158      NA    NA NA   NA    NA   NA    NA   NA   NA    3
## 159 159      NA    NA NA   NA    NA   NA    NA   NA   NA    3
## 160 160      NA    NA NA   NA    NA   NA    NA   NA   NA    3
## 161 161      NA    NA NA   NA    NA   NA    NA   NA   NA    3
## 162 162      NA    NA NA   NA    NA   NA    NA   NA   NA    3
## 163 163      NA    NA NA   NA    NA   NA    NA   NA 0.37    3
## 164 164      NA    NA NA 3.50    NA   NA    NA 2.20   NA    5
## 165 165      NA    NA NA   NA    NA   NA    NA   NA   NA    5
## 166 166      NA    NA NA   NA    NA   NA    NA   NA   NA    5
## 167 167      NA    NA NA   NA    NA   NA    NA   NA   NA    5
## 168 168      NA    NA NA   NA    NA   NA    NA   NA   NA    5
## 169 169      NA    NA NA   NA    NA   NA    NA   NA   NA    5
## 170 170      NA    NA NA   NA    NA   NA    NA   NA   NA    5
## 171 171      NA    NA NA   NA    NA   NA 12.24   NA   NA    5
## 172 172      NA    NA NA 3.04    NA 6.21    NA   NA   NA    5
## 173 173      NA    NA NA 3.02    NA 6.21    NA   NA   NA    5
## 174 174      NA    NA NA   NA    NA   NA 12.50   NA   NA    5
## 175 175      NA    NA NA   NA    NA   NA    NA   NA 0.51    5
## 176 176      NA    NA NA   NA    NA   NA    NA   NA 0.28    5
## 177 177      NA    NA NA   NA    NA   NA    NA   NA   NA    6
## 178 178      NA    NA NA   NA    NA   NA    NA   NA   NA    6
## 179 179      NA    NA NA   NA    NA   NA    NA   NA   NA    6
## 180 180      NA    NA NA   NA    NA   NA    NA   NA   NA    6
## 181 181      NA    NA NA   NA 74.55   NA    NA   NA   NA    6
## 182 182      NA    NA NA   NA    NA   NA    NA   NA   NA    6
## 183 183      NA    NA NA   NA    NA   NA    NA   NA   NA    6
## 184 184      NA    NA NA   NA    NA   NA    NA   NA   NA    6
## 185 185      NA 17.38 NA   NA 75.41   NA    NA   NA   NA    6
## 186 186      NA    NA NA   NA    NA   NA    NA 1.19   NA    7
## 187 187      NA    NA NA   NA    NA   NA    NA 1.63   NA    7
## 188 188      NA    NA NA   NA    NA   NA    NA   NA   NA    7
## 189 189      NA    NA NA   NA    NA   NA    NA   NA   NA    7
## 190 190      NA 15.79 NA   NA    NA   NA    NA 1.68   NA    7
## 191 191      NA    NA NA   NA    NA   NA    NA   NA   NA    7
## 192 192      NA    NA NA   NA    NA   NA    NA   NA   NA    7
## 193 193      NA    NA NA 2.79    NA   NA    NA   NA   NA    7
## 194 194      NA    NA NA   NA    NA   NA    NA 1.59   NA    7
## 195 195      NA    NA NA   NA    NA   NA    NA 1.57   NA    7
## 196 196      NA    NA NA 2.68    NA   NA    NA   NA   NA    7
## 197 197      NA    NA NA 2.54    NA   NA    NA   NA   NA    7
## 198 198      NA    NA NA   NA    NA   NA    NA   NA   NA    7
## 199 199      NA    NA NA 2.66    NA   NA    NA   NA   NA    7
## 200 200      NA    NA NA 2.51    NA   NA    NA   NA   NA    7
## 201 201      NA 15.15 NA   NA    NA   NA    NA   NA   NA    7
## 202 202      NA    NA NA   NA 75.18 2.70    NA   NA   NA    7
```

```
## 203 203      NA     NA NA    NA     NA    NA     NA    NA    NA     7
## 204 204      NA     NA NA    NA     NA    NA     NA  1.71    NA     7
## 205 205      NA     NA NA    NA     NA    NA     NA    NA    NA     7
## 206 206      NA     NA NA    NA     NA    NA     NA  1.55    NA     7
## 207 207      NA     NA NA    NA     NA    NA     NA  1.38    NA     7
## 208 208      NA     NA NA    NA     NA    NA     NA  2.88    NA     7
## 209 209      NA     NA NA  2.74     NA    NA     NA    NA    NA     7
## 210 210      NA     NA NA  2.88     NA    NA     NA    NA    NA     7
## 211 211      NA     NA NA    NA     NA    NA     NA  1.59    NA     7
## 212 212      NA     NA NA    NA     NA    NA     NA  1.64    NA     7
## 213 213      NA     NA NA    NA     NA    NA     NA  1.57    NA     7
## 214 214      NA     NA NA    NA     NA    NA     NA  1.67    NA     7
```

```r
# We have printed the column names and the rows which are outliers for the same. Apart from that, we ha
```

Problem 6: After removing the ID column (column 1), normalize the numeric columns, except the last one (the glass type), using z-score standardization. The last column is the glass type and so it is excluded.

```r
#Normalization of whole data excluding first and last column which are Id and Type column. We put it in

normalize <- function(x)
  {
    zscore <- (x - mean(x))/sd(x)
  }
glass.norm <- as.data.frame(lapply(glass.data[,2:10], normalize))

#We summarize the data and observe whether the data has been normalized or not.
summary(glass.norm)
```

```
##        RI                 Na                 Mg                 Al
##  Min.   :-2.3759   Min.   :-3.2793   Min.   :-1.8611   Min.   :-2.3132
##  1st Qu.:-0.6069   1st Qu.:-0.6127   1st Qu.:-0.3948   1st Qu.:-0.5106
##  Median :-0.2257   Median :-0.1321   Median : 0.5515   Median :-0.1701
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.2608   3rd Qu.: 0.5108   3rd Qu.: 0.6347   3rd Qu.: 0.3707
##  Max.   : 5.1252   Max.   : 4.8642   Max.   : 1.2517   Max.   : 4.1162
##        Si                 K                   Ca                 Ba
##  Min.   :-3.6679   Min.   :-0.76213   Min.   :-2.4783   Min.   :-0.3521
##  1st Qu.:-0.4789   1st Qu.:-0.57430   1st Qu.:-0.5038   1st Qu.:-0.3521
##  Median : 0.1795   Median : 0.08884   Median :-0.2508   Median :-0.3521
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.5636   3rd Qu.: 0.17318   3rd Qu.: 0.1515   3rd Qu.:-0.3521
##  Max.   : 3.5622   Max.   : 8.75961   Max.   : 5.0824   Max.   : 5.9832
##        Fe
##  Min.   :-0.5851
##  1st Qu.:-0.5851
##  Median :-0.5851
##  Mean   : 0.0000
##  3rd Qu.: 0.4412
##  Max.   : 4.6490
```

```r
#Adding the last column back to the normalized data.
glass.norm$Type <- glass.data$Type

#Exploring normalized data
head(glass.norm)
```

```
##           RI         Na        Mg         Al          Si           K         Ca
## 1  0.8708258  0.2842867 1.2517037 -0.6908222 -1.12444556 -0.67013422 -0.1454254
## 2 -0.2487502  0.5904328 0.6346799 -0.1700615  0.10207972 -0.02615193 -0.7918771
## 3 -0.7196308  0.1495824 0.6000157  0.1904651  0.43776033 -0.16414813 -0.8270103
## 4 -0.2322859 -0.2422846 0.6970756 -0.3102663 -0.05284979  0.11184428 -0.5178378
## 5 -0.3113148 -0.1688095 0.6485456 -0.4104126  0.55395746  0.08117845 -0.6232375
## 6 -0.7920739 -0.7566101 0.6416128  0.3506992  0.41193874  0.21917466 -0.6232375
##           Ba         Fe Type
## 1 -0.3520514 -0.5850791    1
## 2 -0.3520514 -0.5850791    1
## 3 -0.3520514 -0.5850791    1
## 4 -0.3520514 -0.5850791    1
## 5 -0.3520514 -0.5850791    1
## 6 -0.3520514  2.0832652    1
```

Problem 7: The data set is sorted, so creating a validation data set requires random selection of elements. Create a stratified sample where you randomly select 20% of each of the cases for each glass type to be part of the validation data set. The remaining cases will form the training data set.

```
#Setting the seed value to 1 for random allocation of data
set.seed(1)

library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(dplyr)

#Checking total number of rows for each case of type
glass.norm %>% group_by(Type) %>% summarise(rows = length(Type))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 6 x 2
##    Type  rows
##   <int> <int>
## 1     1    70
## 2     2    76
## 3     3    17
## 4     5    13
## 5     6     9
## 6     7    29
```

```
#Splitting data using createDataPartition function with a probabilty of 20% for validation data.
index <- createDataPartition(glass.norm$Type, p = 0.2, list = FALSE)
training_data <- glass.norm[-index,]
validation_data <- glass.norm[index,]

#Verifying validation data
validation_data %>% group_by(Type) %>% summarise(rows = length(Type))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 6 x 2
##    Type  rows
##   <int> <int>
## 1     1    15
## 2     2    15
## 3     3     4
## 4     5     2
## 5     6     1
## 6     7     8
```

```
#Creating label vector for training and validation data
training_data_label <- training_data[,10]
validation_data_label <- validation_data[,10]
```

Problem 8: Implement the k-NN algorithm in R (do not use an implementation of k-NN from a package) and use your algorithm with a k=5 to predict the glass type for the following two cases: RI = 1.51621 | 12.53 | 3.48 | 1.39 | 73.39 | 0.60 | 8.55 | 0.00 | Fe = 0.08 RI = 1.5893 | 12.71 | 1.85 | 1.82 | 72.62 | 0.52 | 10.51 | 0.00 | Fe = 0.05 Use the whole normalized data set for this; not just the training data set. Note that you need to normalize the values of the new cases the same way as you normalized the original data.

```
#Implementation of kNN algorithm
data.knn <- glass.norm
data <- glass.data[-1]

#Creating unknown case variables and assigning the values to it
unknown1 <- as.data.frame(cbind(1.51621,12.53,3.48,1.39,73.39,0.6,8.55,0.00,0.08))
unknown2 <- as.data.frame(cbind(1.5893,12.71,1.85,1.82,72.62,0.52,10.51,0.00,0.05))

#Normalizing the new test cases using z-score method which was used earlier
for (i in 1:9)
  {
    unknown1[i] <- (unknown1[i]-mean(data[,i]))/sd(data[,i])
    unknown2[i] <- (unknown2[i]-mean(data[,i]))/sd(data[,i])
  }

#Creating a distance function to calculate the euclidean distance
distance <- function(x,y)
  {
    d <- 0
    for (i in 1:length(x))
      {
        d <- d + (x[i]-y[i])^2
      }
    return(sqrt(d))
  }

#Neighbours function to a list of neighbours
neighbours <- function(train,unknown)
  {
    m <- nrow(train)
    k <- nrow(unknown)
    ds <- numeric(m)
    for (i in 1:m) {
      p <- train[i,1:9]
```

```r
      q <- unknown[c(1:9)]
      ds[i] <- distance(p,q)
    }
    return(unlist(ds))
  }

#k.closest function is used to select k nearest neighbours by using order function
k.closest <- function(neighbours,k)
  {
    ordered.neighbours <- order(neighbours)
    return(ordered.neighbours[1:k])
  }

#Mode function used to calculate mode
Mode <- function(x)
  {
    ux <- unique(x)
    ux[which.max(tabulate(match(x,ux)))]
  }

#kNN Implementation
knn_new <- function(train,unknown,k)
  {
    neighbour <- neighbours(train, unknown)
    closest_neighbours <- k.closest(neighbour, k)
    return(Mode(data.knn$Type[closest_neighbours]))
  }

#Predicting values for both test cases with K=5 value
test1 <- knn_new(data.knn,unknown1,5)
sprintf("Predicted glass type for unknown case one is %s", test1)
```

```
## [1] "Predicted glass type for unknown case one is 1"
```

```r
test2 <- knn_new(data.knn,unknown2,5)
sprintf("Predicted glass type for unknown case two is %s", test2)
```

```
## [1] "Predicted glass type for unknown case two is 2"
```

Problem 9: Apply the knn function from the class package with k=5 and redo the cases from Question (8). Compare your answers.

```r
#Implementation of kNN algorithm using Class package
library(class)

test_class1 <- knn(train = data.knn[,1:9],test = unknown1,cl = data.knn[,10],k=5)
sprintf("Predicted glass type for unknown case one using class package is %s", test_class1)
```

```
## [1] "Predicted glass type for unknown case one using class package is 1"
```

```r
test_class2 <- knn(train = data.knn[,1:9],test = unknown2,cl = data.knn[,10],k=5)
sprintf("Predicted glass type for unknown case two using class package is %s", test_class2)
```

```
## [1] "Predicted glass type for unknown case two using class package is 2"
```

```r
#After predicting both cases with two implementation we get same result with both implementation i.e Pr
```

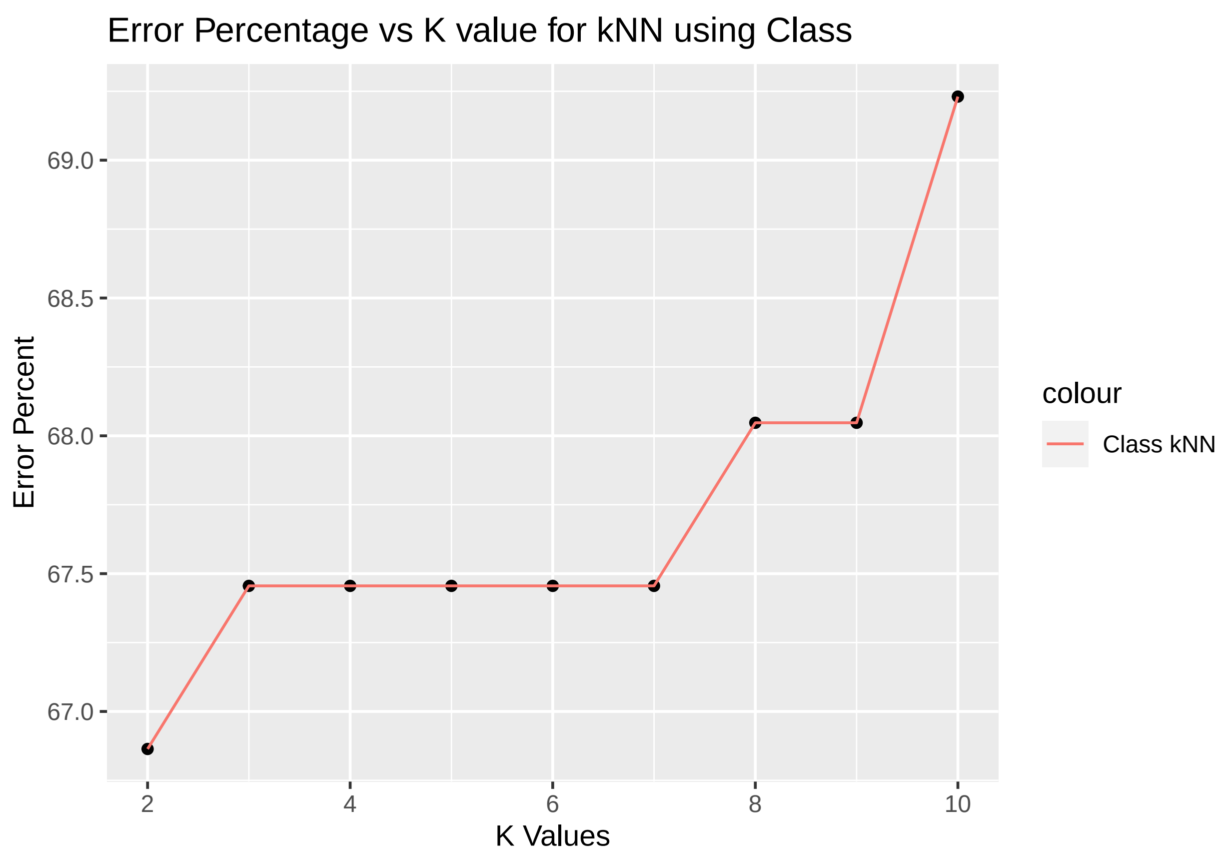
Problem 10: Using your own implementation as well as the class package implementation of kNN, create a plot of k (x-axis) from 2 to 10 versus error rate (percentage of incorrect classifications) for both algorithms using ggplot.

```r
#Error plot
#install.packages("ggplot2")
library(ggplot2)

#Calculating error percentage Using Class implementation
train_error_pred <- rep(0,nrow(training_data))
error <- data.frame(n=c(2:10),perc = rep(NA,9))

for (i in 2:10) {
  train_error_pred <- knn(training_data, validation_data, training_data_label, i)
  error[i-1,2] <- (sum(training_data_label != train_error_pred)/nrow(training_data))*100
}
```

```
## Warning in `!=.default`(training_data_label, train_error_pred): longer object
## length is not a multiple of shorter object length

## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length

## Warning in `!=.default`(training_data_label, train_error_pred): longer object
## length is not a multiple of shorter object length

## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length

## Warning in `!=.default`(training_data_label, train_error_pred): longer object
## length is not a multiple of shorter object length

## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length

## Warning in `!=.default`(training_data_label, train_error_pred): longer object
## length is not a multiple of shorter object length

## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length

## Warning in `!=.default`(training_data_label, train_error_pred): longer object
## length is not a multiple of shorter object length

## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length

## Warning in `!=.default`(training_data_label, train_error_pred): longer object
## length is not a multiple of shorter object length

## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length

## Warning in `!=.default`(training_data_label, train_error_pred): longer object
## length is not a multiple of shorter object length

## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length

## Warning in `!=.default`(training_data_label, train_error_pred): longer object
## length is not a multiple of shorter object length
```

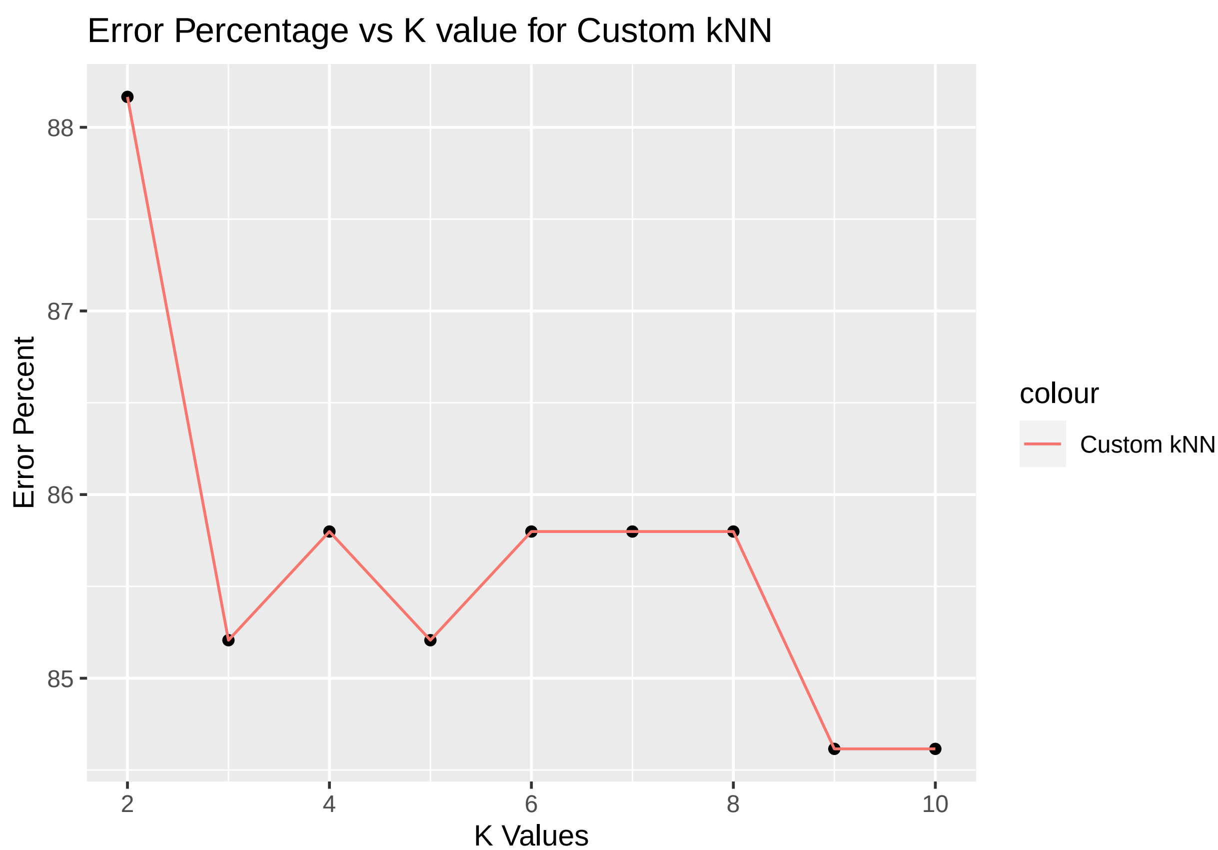
```
## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length

## Warning in `!=.default`(training_data_label, train_error_pred): longer object
## length is not a multiple of shorter object length

## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length
```

```r
#Plotting Error percentage vs k-Values using ggplot function for class package
ggplot(error, aes(x = error[,1], y = error[,2]))+geom_point()+geom_line(aes(color = "Class kNN"))+xlab(
```




```r
#Error using kNN.reg Implementation
train_error_pred <- rep(0,nrow(training_data))
error_new <- data.frame(n=c(2:10),perc = rep(NA,9))

#Defining kNN.reg function as newKnn
newKnn <- function(train,x,k)
{
  dist <- rep(NA, nrow(train))
for (i in 1:nrow(train))
  {
  # Calculating euclidean distance
  dist[i] <- sqrt(sum((train[i,1:9] - glass.norm[x, 1:9])^2))
  }
  o <- order(dist)
  closest <- glass.norm$Type[o[1:k]]
  return(Mode(closest))
}
```

```
val_rows <- as.integer(rownames(validation_data))
for (i in 2:10){
  for (j in 1:nrow(validation_data)){
    train_error_pred[j] <- newKnn(training_data,val_rows[j],i)
  }
error_new[i-1,2] <- sum(training_data_label != train_error_pred) / nrow(training_data) * 100
}

#Plotting Error percentage vs k-Values using ggplot function for kNN.reg implementation
ggplot(error_new, aes(x = error_new[,1], y = error_new[,2]))+geom_point()+geom_line(aes(color = "Custom
```



Problem 11: Produce a cross-table confusion matrix showing the accuracy of the classification using knn from the class package with k = 5.

```
#Calling gmodels library to use crosstable function

#install.packages("gmodels")
#install.packages("caret")
#install.packages("e1071")
library(class)
library(caret)
library(e1071)
library(gmodels)

#Running knn function from class package for training and validation data with k = 5
train_pred <- knn(train = training_data, test = validation_data, cl = training_data_label, k = 5)
```

```r
#Producing CrossTable and Confusion matrix for the classification
CrossTable(x = validation_data_label, y = train_pred, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  45
##
##
##                       | train_pred
## validation_data_label |          1 |          2 |          3 |          5 |          6 |          7 | Row T
## ----------------------|------------|------------|------------|------------|------------|------------|------
##                     1 |         15 |          0 |          0 |          0 |          0 |          0 |
##                       |      1.000 |      0.000 |      0.000 |      0.000 |      0.000 |      0.000 |      0
##                       |      0.833 |      0.000 |      0.000 |      0.000 |      0.000 |      0.000 |
##                       |      0.333 |      0.000 |      0.000 |      0.000 |      0.000 |      0.000 |
## ----------------------|------------|------------|------------|------------|------------|------------|------
##                     2 |          1 |         14 |          0 |          0 |          0 |          0 |
##                       |      0.067 |      0.933 |      0.000 |      0.000 |      0.000 |      0.000 |      0
##                       |      0.056 |      1.000 |      0.000 |      0.000 |      0.000 |      0.000 |
##                       |      0.022 |      0.311 |      0.000 |      0.000 |      0.000 |      0.000 |
## ----------------------|------------|------------|------------|------------|------------|------------|------
##                     3 |          2 |          0 |          2 |          0 |          0 |          0 |
##                       |      0.500 |      0.000 |      0.500 |      0.000 |      0.000 |      0.000 |      0
##                       |      0.111 |      0.000 |      1.000 |      0.000 |      0.000 |      0.000 |
##                       |      0.044 |      0.000 |      0.044 |      0.000 |      0.000 |      0.000 |
## ----------------------|------------|------------|------------|------------|------------|------------|------
##                     5 |          0 |          0 |          0 |          2 |          0 |          0 |
##                       |      0.000 |      0.000 |      0.000 |      1.000 |      0.000 |      0.000 |      0
##                       |      0.000 |      0.000 |      0.000 |      0.667 |      0.000 |      0.000 |
##                       |      0.000 |      0.000 |      0.000 |      0.044 |      0.000 |      0.000 |
## ----------------------|------------|------------|------------|------------|------------|------------|------
##                     6 |          0 |          0 |          0 |          0 |          1 |          0 |
##                       |      0.000 |      0.000 |      0.000 |      0.000 |      1.000 |      0.000 |      0
##                       |      0.000 |      0.000 |      0.000 |      0.000 |      0.500 |      0.000 |
##                       |      0.000 |      0.000 |      0.000 |      0.000 |      0.022 |      0.000 |
## ----------------------|------------|------------|------------|------------|------------|------------|------
##                     7 |          0 |          0 |          0 |          1 |          1 |          6 |
##                       |      0.000 |      0.000 |      0.000 |      0.125 |      0.125 |      0.750 |      0
##                       |      0.000 |      0.000 |      0.000 |      0.333 |      0.500 |      1.000 |
##                       |      0.000 |      0.000 |      0.000 |      0.022 |      0.022 |      0.133 |
## ----------------------|------------|------------|------------|------------|------------|------------|------
##          Column Total |         18 |         14 |          2 |          3 |          2 |          6 |
##                       |      0.400 |      0.311 |      0.044 |      0.067 |      0.044 |      0.133 |
## ----------------------|------------|------------|------------|------------|------------|------------|------
##
```

```
##
confusionMatrix(train_pred,as.factor(validation_data_label))

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3  5  6  7
##          1 15  1  2  0  0  0
##          2  0 14  0  0  0  0
##          3  0  0  2  0  0  0
##          5  0  0  0  2  0  1
##          6  0  0  0  0  1  1
##          7  0  0  0  0  0  6
##
## Overall Statistics
##
##                Accuracy : 0.8889
##                  95% CI : (0.7595, 0.9629)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 1.408e-14
##
##                   Kappa : 0.8481
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 5 Class: 6 Class: 7
## Sensitivity            1.0000   0.9333  0.50000  1.00000  1.00000   0.7500
## Specificity            0.9000   1.0000  1.00000  0.97674  0.97727   1.0000
## Pos Pred Value         0.8333   1.0000  1.00000  0.66667  0.50000   1.0000
## Neg Pred Value         1.0000   0.9677  0.95349  1.00000  1.00000   0.9487
## Prevalence             0.3333   0.3333  0.08889  0.04444  0.02222   0.1778
## Detection Rate         0.3333   0.3111  0.04444  0.04444  0.02222   0.1333
## Detection Prevalence   0.4000   0.3111  0.04444  0.06667  0.04444   0.1333
## Balanced Accuracy      0.9500   0.9667  0.75000  0.98837  0.98864   0.8750

#We observe the accurary for model in the confusion matrix. Accuracy varies for different seed value. F
```

Problem 12: Download this (modified) version of the Glass data set containing missing values in column 4. Identify the missing values. Impute the missing values using your version of kNN from Problem 2 below using the other columns are predictor features.

```
#Importing new modified data using read.csv function

missing_glass_data <- read.csv("https://da5030.weebly.com/uploads/8/6/5/9/8659576/da5030.glass.data_wit

#Assigning header to the data
colnames(missing_glass_data) <- col_names

#Checking for NA's in the data. We observe that "Mg" column has 9 NA Values
summary(missing_glass_data)

##        Id               RI               Na               Mg
##  Min.   :  1.00   Min.   :1.511   Min.   :10.73   Min.   :0.000
```

```
##  1st Qu.: 54.25   1st Qu.:1.517   1st Qu.:12.91   1st Qu.:2.240
##  Median :107.50   Median :1.518   Median :13.30   Median :3.480
##  Mean   :107.50   Mean   :1.518   Mean   :13.41   Mean   :2.733
##  3rd Qu.:160.75   3rd Qu.:1.519   3rd Qu.:13.82   3rd Qu.:3.610
##  Max.   :214.00   Max.   :1.534   Max.   :17.38   Max.   :4.490
##                                                   NA's   :9
##        Al              Si              K                Ca
##  Min.   :0.290   Min.   :69.81   Min.   :0.0000   Min.   : 5.430
##  1st Qu.:1.190   1st Qu.:72.28   1st Qu.:0.1225   1st Qu.: 8.240
##  Median :1.360   Median :72.79   Median :0.5550   Median : 8.600
##  Mean   :1.445   Mean   :72.65   Mean   :0.4971   Mean   : 8.957
##  3rd Qu.:1.630   3rd Qu.:73.09   3rd Qu.:0.6100   3rd Qu.: 9.172
##  Max.   :3.500   Max.   :75.41   Max.   :6.2100   Max.   :16.190
##
##        Ba              Fe              Type
##  Min.   :0.000   Min.   :0.00000   Min.   :1.00
##  1st Qu.:0.000   1st Qu.:0.00000   1st Qu.:1.00
##  Median :0.000   Median :0.00000   Median :2.00
##  Mean   :0.175   Mean   :0.05701   Mean   :2.78
##  3rd Qu.:0.000   3rd Qu.:0.10000   3rd Qu.:3.00
##  Max.   :3.150   Max.   :0.51000   Max.   :7.00
##
```

```r
#Removing Id column
missing_glass_data <- missing_glass_data[-1]

col_list <- c(1,2,4,5,6,7,8,9)

normalized_data <- missing_glass_data

#Normalizing the new data with min-max normalization method
for (i in col_list)
  {
    normalized_data[,i] <- (missing_glass_data[,i]-min(missing_glass_data[,i]))/(max(missing_glass_data
  }

#We create new forecast variable for each NA row. Since we have 9 NA rows we create 9 vectors
forecast_data.1 <- normalized_data[20,c(1,2,4,5,6,7,8,9,10)]
forecast_data.2 <- normalized_data[30,c(1,2,4,5,6,7,8,9,10)]
forecast_data.3 <- normalized_data[95,c(1,2,4,5,6,7,8,9,10)]
forecast_data.4 <- normalized_data[163,c(1,2,4,5,6,7,8,9,10)]
forecast_data.5 <- normalized_data[169,c(1,2,4,5,6,7,8,9,10)]
forecast_data.6 <- normalized_data[184,c(1,2,4,5,6,7,8,9,10)]
forecast_data.7 <- normalized_data[194,c(1,2,4,5,6,7,8,9,10)]
forecast_data.8 <- normalized_data[200,c(1,2,4,5,6,7,8,9,10)]
forecast_data.9 <- normalized_data[208,c(1,2,4,5,6,7,8,9,10)]

missing_data <- normalized_data[-c(20,30,95,163,169,184,194,200,208),]

#Creating training and target data of the new modified data
train_data_new <- missing_data[,c(1,2,4,5,6,7,8,9,10)]
target_data_new <- missing_data[,3]

#kNN.reg function from problem 2
```

```r
kNN.reg <- function(new_data, target_data, train_data, k){
  weights <- numeric(k)
  n <- nrow(train_data)
  d <- rep(0,n)
  for (i in 1:n)
  {
    d[i] <- sqrt(sum((train_data[i,] - new_data)^2))
  }
  o <- order(d)
  values <- target_data[o[1:k]]
  weights <- c(3,2)
  for (i in 3:k)
  {
    weights[i] <- 1
  }
  sw <- values * weights
  return(sum(sw)/sum(weights))
}

#Calling kNN.reg function from problem 2 to get predicted values
impute_forecast.1 <- kNN.reg(forecast_data.1, target_data_new, train_data_new, 4)
impute_forecast.2 <- kNN.reg(forecast_data.2, target_data_new, train_data_new, 4)
impute_forecast.3 <- kNN.reg(forecast_data.3, target_data_new, train_data_new, 4)
impute_forecast.4 <- kNN.reg(forecast_data.4, target_data_new, train_data_new, 4)
impute_forecast.5 <- kNN.reg(forecast_data.5, target_data_new, train_data_new, 4)
impute_forecast.6 <- kNN.reg(forecast_data.6, target_data_new, train_data_new, 4)
impute_forecast.7 <- kNN.reg(forecast_data.7, target_data_new, train_data_new, 4)
impute_forecast.8 <- kNN.reg(forecast_data.8, target_data_new, train_data_new, 4)
impute_forecast.9 <- kNN.reg(forecast_data.9, target_data_new, train_data_new, 4)

impute_forecast <- rbind(impute_forecast.1,impute_forecast.2,impute_forecast.3,impute_forecast.4,impute_

#We can observe the imputed values in a dataframe for the Mg column
rownames(impute_forecast) <- c(20,30,95,163,169,184,194,200,208)
colnames(impute_forecast) <- "Mg"
print("Imputed Values for the Mg Column:")
```

```
## [1] "Imputed Values for the Mg Column:"
```

```r
impute_forecast
```

```
##            Mg
## 20  3.3385714
## 30  3.4842857
## 95  3.5114286
## 163 3.4928571
## 169 0.7971429
## 184 2.3485714
## 194 0.0000000
## 200 0.0000000
## 208 0.4657143
```

```r
#Appending the imputed values to main dataset which is missing_glass_data
missing_glass_data[c(20,30,95,163,169,184,194,200,208),3] <- c(impute_forecast.1,impute_forecast.2,impu
```

```r
#Verifying whether NA values are imputed or not
summary(missing_glass_data)
```

```
##        RI              Na              Mg              Al       
##  Min.   :1.511   Min.   :10.73   Min.   :0.000   Min.   :0.290  
##  1st Qu.:1.517   1st Qu.:12.91   1st Qu.:2.192   1st Qu.:1.190  
##  Median :1.518   Median :13.30   Median :3.480   Median :1.360  
##  Mean   :1.518   Mean   :13.41   Mean   :2.700   Mean   :1.445  
##  3rd Qu.:1.519   3rd Qu.:13.82   3rd Qu.:3.600   3rd Qu.:1.630  
##  Max.   :1.534   Max.   :17.38   Max.   :4.490   Max.   :3.500  
##        Si              K                Ca               Ba       
##  Min.   :69.81   Min.   :0.0000   Min.   : 5.430   Min.   :0.000  
##  1st Qu.:72.28   1st Qu.:0.1225   1st Qu.: 8.240   1st Qu.:0.000  
##  Median :72.79   Median :0.5550   Median : 8.600   Median :0.000  
##  Mean   :72.65   Mean   :0.4971   Mean   : 8.957   Mean   :0.175  
##  3rd Qu.:73.09   3rd Qu.:0.6100   3rd Qu.: 9.172   3rd Qu.:0.000  
##  Max.   :75.41   Max.   :6.2100   Max.   :16.190   Max.   :3.150  
##        Fe               Type     
##  Min.   :0.00000   Min.   :1.00  
##  1st Qu.:0.00000   1st Qu.:1.00  
##  Median :0.00000   Median :2.00  
##  Mean   :0.05701   Mean   :2.78  
##  3rd Qu.:0.10000   3rd Qu.:3.00  
##  Max.   :0.51000   Max.   :7.00  
```

--- Question 2 ---

Problem 1: Investigate this data set of home prices in King County (USA).

```r
#Importing house data using read.csv
house.data <- read.csv("kc_house_data.csv", stringsAsFactors = F, header = T)

#Exploring house data and studying the data types present in it
head(house.data)
```

```
##           id            date   price bedrooms bathrooms sqft_living sqft_lot
## 1 7129300520 20141013T000000  221900        3      1.00        1180     5650
## 2 6414100192 20141209T000000  538000        3      2.25        2570     7242
## 3 5631500400 20150225T000000  180000        2      1.00         770    10000
## 4 2487200875 20141209T000000  604000        4      3.00        1960     5000
## 5 1954400510 20150218T000000  510000        3      2.00        1680     8080
## 6 7237550310 20140512T000000 1225000        4      4.50        5420   101930
##   floors waterfront view condition grade sqft_above sqft_basement yr_built
## 1      1          0    0         3     7       1180             0     1955
## 2      2          0    0         3     7       2170           400     1951
## 3      1          0    0         3     6        770             0     1933
## 4      1          0    0         5     7       1050           910     1965
## 5      1          0    0         3     8       1680             0     1987
## 6      1          0    0         3    11       3890          1530     2001
##   yr_renovated zipcode     lat     long sqft_living15 sqft_lot15
## 1            0   98178 47.5112 -122.257          1340       5650
## 2         1991   98125 47.7210 -122.319          1690       7639
## 3            0   98028 47.7379 -122.233          2720       8062
## 4            0   98136 47.5208 -122.393          1360       5000
## 5            0   98074 47.6168 -122.045          1800       7503
## 6            0   98053 47.6561 -122.005          4760     101930
```

```r
str(house.data)
```

```
## 'data.frame':    21613 obs. of  21 variables:
##  $ id           : num  7.13e+09 6.41e+09 5.63e+09 2.49e+09 1.95e+09 ...
##  $ date         : chr  "20141013T000000" "20141209T000000" "20150225T000000" "20141209T000000" ...
##  $ price        : num  221900 538000 180000 604000 510000 ...
##  $ bedrooms     : int  3 3 2 4 3 4 3 3 3 3 ...
##  $ bathrooms    : num  1 2.25 1 3 2 4.5 2.25 1.5 1 2.5 ...
##  $ sqft_living  : int  1180 2570 770 1960 1680 5420 1715 1060 1780 1890 ...
##  $ sqft_lot     : int  5650 7242 10000 5000 8080 101930 6819 9711 7470 6560 ...
##  $ floors       : num  1 2 1 1 1 1 2 1 1 2 ...
##  $ waterfront   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ view         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ condition    : int  3 3 3 5 3 3 3 3 3 3 ...
##  $ grade        : int  7 7 6 7 8 11 7 7 7 7 ...
##  $ sqft_above   : int  1180 2170 770 1050 1680 3890 1715 1060 1050 1890 ...
##  $ sqft_basement: int  0 400 0 910 0 1530 0 0 730 0 ...
##  $ yr_built     : int  1955 1951 1933 1965 1987 2001 1995 1963 1960 2003 ...
##  $ yr_renovated : int  0 1991 0 0 0 0 0 0 0 0 ...
##  $ zipcode      : int  98178 98125 98028 98136 98074 98053 98003 98198 98146 98038 ...
##  $ lat          : num  47.5 47.7 47.7 47.5 47.6 ...
##  $ long         : num  -122 -122 -122 -122 -122 ...
##  $ sqft_living15: int  1340 1690 2720 1360 1800 4760 2238 1650 1780 2390 ...
##  $ sqft_lot15   : int  5650 7639 8062 5000 7503 101930 6819 9711 8113 7570 ...
```

```r
summary(house.data)
```

```
##        id                date               price            bedrooms     
##  Min.   :1.000e+06   Length:21613       Min.   :  75000   Min.   : 0.000  
##  1st Qu.:2.123e+09   Class :character   1st Qu.: 321950   1st Qu.: 3.000  
##  Median :3.905e+09   Mode  :character   Median : 450000   Median : 3.000  
##  Mean   :4.580e+09                      Mean   : 540088   Mean   : 3.371  
##  3rd Qu.:7.309e+09                      3rd Qu.: 645000   3rd Qu.: 4.000  
##  Max.   :9.900e+09                      Max.   :7700000   Max.   :33.000  
##    bathrooms      sqft_living       sqft_lot           floors     
##  Min.   :0.000   Min.   :  290   Min.   :    520   Min.   :1.000  
##  1st Qu.:1.750   1st Qu.: 1427   1st Qu.:   5040   1st Qu.:1.000  
##  Median :2.250   Median : 1910   Median :   7618   Median :1.500  
##  Mean   :2.115   Mean   : 2080   Mean   :  15107   Mean   :1.494  
##  3rd Qu.:2.500   3rd Qu.: 2550   3rd Qu.:  10688   3rd Qu.:2.000  
##  Max.   :8.000   Max.   :13540   Max.   :1651359   Max.   :3.500  
##    waterfront             view          condition         grade       
##  Min.   :0.000000   Min.   :0.0000   Min.   :1.000   Min.   : 1.000  
##  1st Qu.:0.000000   1st Qu.:0.0000   1st Qu.:3.000   1st Qu.: 7.000  
##  Median :0.000000   Median :0.0000   Median :3.000   Median : 7.000  
##  Mean   :0.007542   Mean   :0.2343   Mean   :3.409   Mean   : 7.657  
##  3rd Qu.:0.000000   3rd Qu.:0.0000   3rd Qu.:4.000   3rd Qu.: 8.000  
##  Max.   :1.000000   Max.   :4.0000   Max.   :5.000   Max.   :13.000  
##    sqft_above   sqft_basement       yr_built     yr_renovated   
##  Min.   : 290   Min.   :   0.0   Min.   :1900   Min.   :   0.0  
##  1st Qu.:1190   1st Qu.:   0.0   1st Qu.:1951   1st Qu.:   0.0  
##  Median :1560   Median :   0.0   Median :1975   Median :   0.0  
##  Mean   :1788   Mean   : 291.5   Mean   :1971   Mean   :  84.4  
##  3rd Qu.:2210   3rd Qu.: 560.0   3rd Qu.:1997   3rd Qu.:   0.0  
```

```
##  Max.   :9410   Max.   :4820.0   Max.   :2015   Max.   :2015.0
##     zipcode          lat             long        sqft_living15
##  Min.   :98001   Min.   :47.16   Min.   :-122.5   Min.   : 399
##  1st Qu.:98033   1st Qu.:47.47   1st Qu.:-122.3   1st Qu.:1490
##  Median :98065   Median :47.57   Median :-122.2   Median :1840
##  Mean   :98078   Mean   :47.56   Mean   :-122.2   Mean   :1987
##  3rd Qu.:98118   3rd Qu.:47.68   3rd Qu.:-122.1   3rd Qu.:2360
##  Max.   :98199   Max.   :47.78   Max.   :-121.3   Max.   :6210
##    sqft_lot15
##  Min.   :   651
##  1st Qu.:  5100
##  Median :  7620
##  Mean   : 12768
##  3rd Qu.: 10083
##  Max.   :871200
```

Problem 2: Save the price column in a separate vector/dataframe called target_data. Move all of the columns except the ID, date, price, yr_renovated, zipcode, lat, long, sqft_living15, and sqft_lot15 columns into a new data frame called train_data.

```r
#Create Target Data which contains price column
target_data <- house.data$price
head(target_data)
```

```
## [1]  221900  538000  180000  604000  510000 1225000
```

```r
#Moving other columns to train_data which are used for training the model
train_data <- house.data[,4:15]
head(train_data)
```

```
##   bedrooms bathrooms sqft_living sqft_lot floors waterfront view condition
## 1        3      1.00        1180     5650      1          0    0         3
## 2        3      2.25        2570     7242      2          0    0         3
## 3        2      1.00         770    10000      1          0    0         3
## 4        4      3.00        1960     5000      1          0    0         5
## 5        3      2.00        1680     8080      1          0    0         3
## 6        4      4.50        5420   101930      1          0    0         3
##   grade sqft_above sqft_basement yr_built
## 1     7       1180             0     1955
## 2     7       2170           400     1951
## 3     6        770             0     1933
## 4     7       1050           910     1965
## 5     8       1680             0     1987
## 6    11       3890          1530     2001
```

Problem 3: Normalize all of the columns (except the boolean columns waterfront and view) using min-max normalization.

```r
#Normalization of Data using min-max normalization method. Here we exclude waterfront and view columns
col_list <- c(1,2,3,4,5,8,9,10,11,12)

train_data_normalized <- train_data

for (i in col_list)
  {
    train_data_normalized[,i] <- (train_data[,i]-min(train_data[,i]))/(max(train_data[,i])-min(train_da
  }
```

```r
#Verifying whether data is normalized or not
head(train_data_normalized)
```

```
##     bedrooms bathrooms sqft_living    sqft_lot floors waterfront view condition
## 1 0.09090909   0.12500  0.06716981 0.003107511    0.0          0    0       0.5
## 2 0.09090909   0.28125  0.17207547 0.004071869    0.4          0    0       0.5
## 3 0.06060606   0.12500  0.03622642 0.005742535    0.0          0    0       0.5
## 4 0.12121212   0.37500  0.12603774 0.002713772    0.0          0    0       1.0
## 5 0.09090909   0.25000  0.10490566 0.004579490    0.0          0    0       0.5
## 6 0.12121212   0.56250  0.38716981 0.061429370    0.0          0    0       0.5
##       grade sqft_above sqft_basement  yr_built
## 1 0.5000000 0.09758772    0.00000000 0.4782609
## 2 0.5000000 0.20614035    0.08298755 0.4434783
## 3 0.4166667 0.05263158    0.00000000 0.2869565
## 4 0.5000000 0.08333333    0.18879668 0.5652174
## 5 0.5833333 0.15241228    0.00000000 0.7565217
## 6 0.8333333 0.39473684    0.31742739 0.8782609
```

```r
summary(train_data_normalized)
```

```
##     bedrooms          bathrooms        sqft_living          sqft_lot
##  Min.   :0.00000   Min.   :0.0000   Min.   :0.00000   Min.   :0.000000
##  1st Qu.:0.09091   1st Qu.:0.2188   1st Qu.:0.08581   1st Qu.:0.002738
##  Median :0.09091   Median :0.2812   Median :0.12226   Median :0.004300
##  Mean   :0.10215   Mean   :0.2643   Mean   :0.13509   Mean   :0.008836
##  3rd Qu.:0.12121   3rd Qu.:0.3125   3rd Qu.:0.17057   3rd Qu.:0.006159
##  Max.   :1.00000   Max.   :1.0000   Max.   :1.00000   Max.   :1.000000
##      floors          waterfront            view          condition
##  Min.   :0.0000   Min.   :0.000000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.000000   1st Qu.:0.0000   1st Qu.:0.5000
##  Median :0.2000   Median :0.000000   Median :0.0000   Median :0.5000
##  Mean   :0.1977   Mean   :0.007542   Mean   :0.2343   Mean   :0.6024
##  3rd Qu.:0.4000   3rd Qu.:0.000000   3rd Qu.:0.0000   3rd Qu.:0.7500
##  Max.   :1.0000   Max.   :1.000000   Max.   :4.0000   Max.   :1.0000
##      grade          sqft_above      sqft_basement        yr_built
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.5000   1st Qu.:0.09868   1st Qu.:0.00000   1st Qu.:0.4435
##  Median :0.5000   Median :0.13925   Median :0.00000   Median :0.6522
##  Mean   :0.5547   Mean   :0.16430   Mean   :0.06048   Mean   :0.6174
##  3rd Qu.:0.5833   3rd Qu.:0.21053   3rd Qu.:0.11618   3rd Qu.:0.8435
##  Max.   :1.0000   Max.   :1.00000   Max.   :1.00000   Max.   :1.0000
```

Problem 4: Build a function called knn.reg that implements a regression version of kNN that averages the prices of the k nearest neighbors using a weighted average where the weight is 3 for the closest neighbor, 2 for the second closest and 1 for the remaining neighbors (recall that a weighted average requires that you divide the sum product of the weight and values by the sum of the weights).

It must use the following signature:

knn.reg (new_data, target_data, train_data, k)

where new_data is a data frame with new cases, target_data is a data frame with a single column of prices from (2), train_data is a data frame with the features from (2) that correspond to a price in target_data, and k is the number of nearest neighbors to consider. It must return the predicted price.

```r
#Building kNN.reg Function
kNN.reg <- function(new_data, target_data, train_data, k){
  weights <- numeric(k)
  n <- nrow(train_data)
  d <- rep(0,n)

  #Calculating euclidean distance
  for (i in 1:n)
  {
    d[i] <- sqrt(sum((train_data[i,] - new_data)^2))
  }

  #Ordering neighbours
  o <- order(d)

  #Adding k nearest neighbours to values
  values <- target_data[o[1:k]]

  #Creating a weights variable which contains values 3,2,1 for 1st nearest, 2nd nearest, and other neig
  weights <- c(3,2)
  for (i in 3:k)
  {
    weights[i] <- 1
  }

  #Calculating weighted average forecast and returning the value
  sw <- values * weights
  return(sum(sw)/sum(weights))
}
```

Problem 5: Forecast the price of this new home using your regression kNN using k = 4: bedrooms = 4 | bathrooms = 3 | sqft_living = 4852 | sqft_lot = 10244 | floors = 3 | waterfront = 0 | view = 1 | condition = 3 | grade = 11 sqft_above = 1960 | sqft_basement = 820 | yr_built = 1978

```r
#Storing new test case in new_data vector variable

new_data <- as.data.frame(cbind(4,3,4852,10244,3,0,1,3,11,1960,820,1978))

#Normalizing new_data using min-max normalization method
for (i in 1:ncol(new_data))
  {
    new_data[i] <- (new_data[i]-min(train_data[,i]))/(max(train_data[,i])-min(train_data[,i]))
  }

#Calling the kNN.reg function to forecast the new_data and printing the value
forecast <- kNN.reg(new_data,target_data,train_data_normalized,4)
sprintf("Forecast Price for new test case is %s", forecast)
```

```
## [1] "Forecast Price for new test case is 954515.714285714"
```