# DA5030.P1.Parpattedar

*Shruti Parpattedar*

*February 8, 2019*

## Problem 1

### Question 1

Reading Glass Identification Database and assigning column names

```
data <- read.csv("GlassID.csv", header = FALSE, fileEncoding = "UTF-8-BOM")
names(data) = c("ID", "RI", "Na", "Mg", "Al", "Si", "K", "Ca", "Ba", "Fe", "Type")
```

### Question 2

Exploring the database

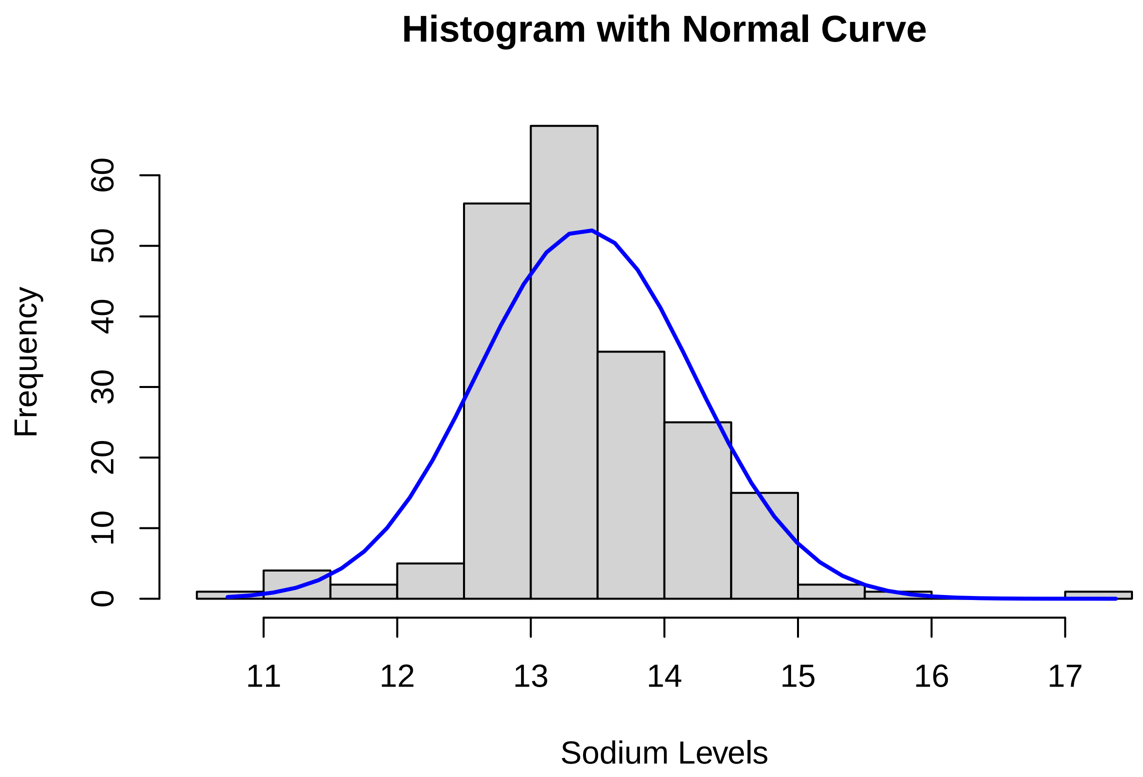
```
str(data)
```

```
## '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 ...
```

### Question 3

Histogram of Na column with an overlaying normal curve. Looking at the graph and curve, the data looks to be normally distributed.

```
na <- data$Na
na.hist <- hist(na, breaks = 10, xlab = "Sodium Levels", col = "lightgrey",
          main = "Histogram with Normal Curve")
xfit <- seq(min(na), max(na), length = 40)
yfit <- dnorm(xfit, mean = mean(na), sd = sd(na))
yfit <- yfit * diff(na.hist$mids[1:2]) * length(na)
lines(xfit, yfit, col="blue", lwd=2)
```



### Question 4

The kNN algorithm is a non-parametric method, which means that it is a simple, easy to use algorithm that makes no assumptions about the data. So, it does not need the data to be normally distributed. But, the calculations will prove to be meaningless if there are substantial differences in the scale of the dimensions.

### Question 5

Removing the first (index) column and normalizing all the remaining columns except the last(Type) column

```r
data <- data[,-1]
norm_data <- cbind(scale(data[,1:9], center = TRUE, scale = TRUE), Type = data$Type)
```

### Question 6

Dividing the dataset randomly into training and validation datasets.

```r
sample <- sample.int(n = nrow(norm_data), size = 0.5*nrow(norm_data), replace = FALSE)
validn <- norm_data[sample,]
train <- norm_data[-sample,]
```

### Question 7

Building a kNN prediction function to predict the type for the two given cases.

```r
getMode <- function(x) {
  uniq <- unique(x)
  uniq[which.max(tabulate(match(x, uniq)))]
}

knn_pred <- function(df, unk, k)
{
  n <- nrow(df)
  d <- numeric(n)
  for (i in 1:n)
  {
    d[i] <- sqrt(sum((df[i,1:9] - unk[1:9])^2))
  }
  o <- order(d)
  getMode(df[o[1:k], 10])
}

eg1 <- round(c(1.51721, 12.53, 3.48, 1.39, 73.39, 0.60, 8.55, 0.00, 0.08), digits = 5)
eg2 <- round(c(1.4897, 12.71, 1.85, 1.81, 72.69, 0.52, 10.01, 0.00, 0.02), digits = 5)

#eg2 <- scale(eg2, center = TRUE, scale = TRUE)

for(i in 1:9)
{
  eg1[i] <- round((eg1[i] - mean(data[,i])) / sd(data[,i]), digits = 5)
  eg2[i] <- round((eg2[i] - mean(data[,i])) / sd(data[, i]), digits = 5)
}

paste("Prediction for case 1 - ", knn_pred(norm_data, eg1, 10))
```

```
## [1] "Prediction for case 1 -  1"
```

```
paste("Prediction for case 2 - ", knn_pred(norm_data, eg2, 10))
```

```
## [1] "Prediction for case 2 -  2"
```

## Question 8

Using the knn function from the class package to predict the type of the two given cases.

```
library(class)
paste("Prediction for case 1 - ", knn(norm_data[,1:9], eg1, cl = norm_data[,10], 14))
```

```
## [1] "Prediction for case 1 -  1"
```

```
paste("Prediction for case 2 - ", knn(norm_data[,1:9], eg2, cl = norm_data[,10], 14))
```

```
## [1] "Prediction for case 2 -  2"
```

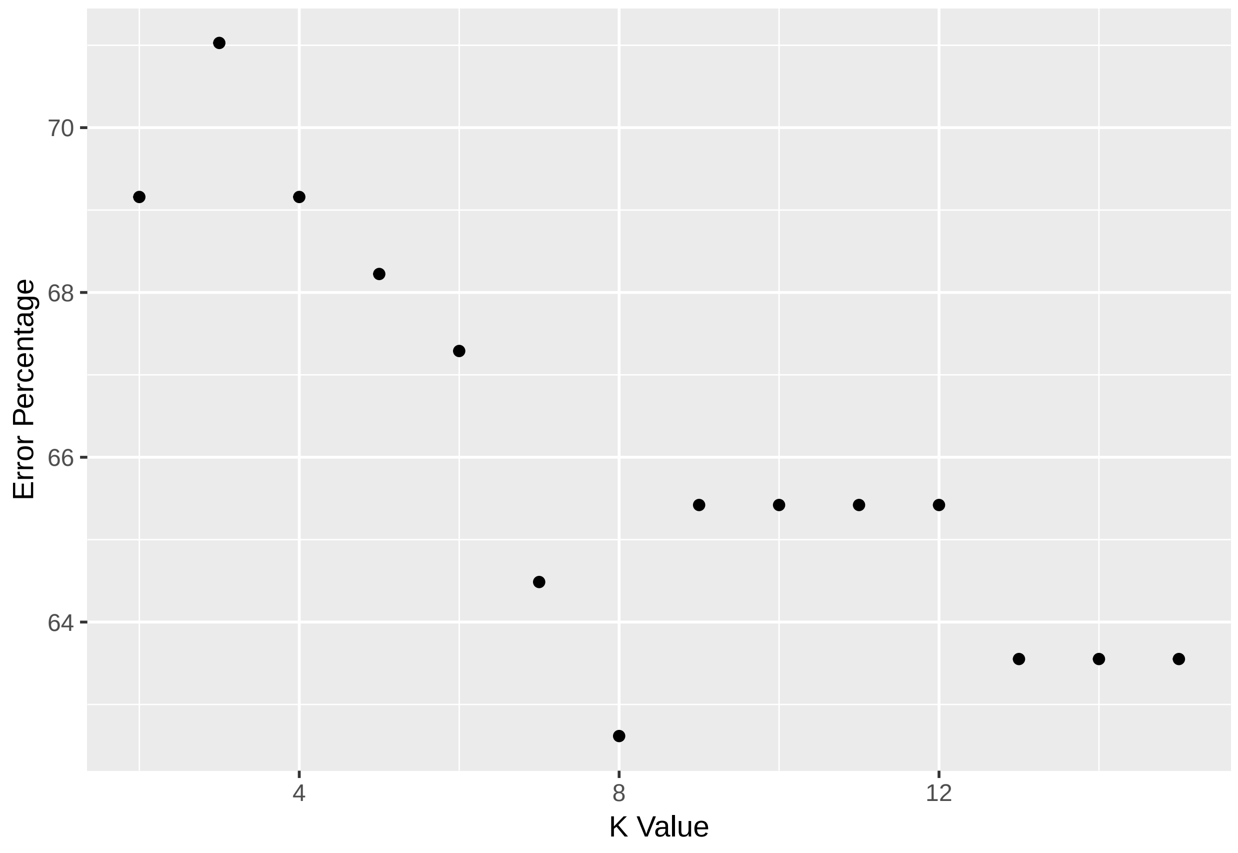
## Question 9

Plotting k values from 2 to 15 against the percentage of error using ggplot.

```
library(ggplot2)
train_d <- train[,1:9]
test_d <- validn[,1:9]
train_tar <- train[,10]
train_pred <- rep(0, nrow(train))
incorr_cl <- data.frame(n = c(2:15), perc_error = rep(NA, 14))
```

```r
for(n in 2:15)
{
  train_pred <- knn(train = train_d, test = test_d, train_tar, n)
  incorr_cl[n-1,2] <- sum(train_tar != train_pred)/nrow(train)*100
}

ggplot(incorr_cl, aes(x = incorr_cl[,1], y = incorr_cl[,2])) + geom_point() +
  xlab("K Value") + ylab("Error Percentage")
```



## Question 10

Cross-table confusion matrix showing the accuracy of the classification using knn from the class package with k = 5.

```r
library(gmodels)
train_pred <- knn(train = train[,1:9], test = validn[,1:9], cl = train[,10], k = 5)
CrossTable(train_tar, train_pred, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  107
##
##
##              | train_pred
##    train_tar |         1 |         2 |         3 |         5 |         6 |         7 | Row Total |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
##            1 |        14 |        14 |         2 |         3 |         1 |         3 |        37 |
##              |     0.378 |     0.378 |     0.054 |     0.081 |     0.027 |     0.081 |     0.346 |
##              |     0.368 |     0.318 |     0.667 |     0.600 |     0.500 |     0.200 |           |
##              |     0.131 |     0.131 |     0.019 |     0.028 |     0.009 |     0.028 |           |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
##            2 |        15 |        17 |         1 |         1 |         0 |         8 |        42 |
##              |     0.357 |     0.405 |     0.024 |     0.024 |     0.000 |     0.190 |     0.393 |
##              |     0.395 |     0.386 |     0.333 |     0.200 |     0.000 |     0.533 |           |
##              |     0.140 |     0.159 |     0.009 |     0.009 |     0.000 |     0.075 |           |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
##            3 |         1 |         5 |         0 |         0 |         1 |         0 |         7 |
##              |     0.143 |     0.714 |     0.000 |     0.000 |     0.143 |     0.000 |     0.065 |
##              |     0.026 |     0.114 |     0.000 |     0.000 |     0.500 |     0.000 |           |
##              |     0.009 |     0.047 |     0.000 |     0.000 |     0.009 |     0.000 |           |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
##            5 |         1 |         3 |         0 |         0 |         0 |         0 |         4 |
##              |     0.250 |     0.750 |     0.000 |     0.000 |     0.000 |     0.000 |     0.037 |
##              |     0.026 |     0.068 |     0.000 |     0.000 |     0.000 |     0.000 |           |
##              |     0.009 |     0.028 |     0.000 |     0.000 |     0.000 |     0.000 |           |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
##            6 |         2 |         2 |         0 |         0 |         0 |         0 |         4 |
##              |     0.500 |     0.500 |     0.000 |     0.000 |     0.000 |     0.000 |     0.037 |
##              |     0.053 |     0.045 |     0.000 |     0.000 |     0.000 |     0.000 |           |
##              |     0.019 |     0.019 |     0.000 |     0.000 |     0.000 |     0.000 |           |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
##            7 |         5 |         3 |         0 |         1 |         0 |         4 |        13 |
##              |     0.385 |     0.231 |     0.000 |     0.077 |     0.000 |     0.308 |     0.121 |
##              |     0.132 |     0.068 |     0.000 |     0.200 |     0.000 |     0.267 |           |
##              |     0.047 |     0.028 |     0.000 |     0.009 |     0.000 |     0.037 |           |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
## Column Total |        38 |        44 |         3 |         5 |         2 |        15 |       107 |
##              |     0.355 |     0.411 |     0.028 |     0.047 |     0.019 |     0.140 |           |
## -------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
##
##
```

## Problem 2

### Question 1

Exploring the home prices in King County (USA) dataset.

```r
data2 <- read.csv("kc_house_data.csv", header = TRUE, stringsAsFactors = FALSE)
str(data2)
```

```
## '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 ...
```

## Question 2

Creating the target and training datasets

```r
target_data <- data2$price
train_data <- data2[,4:15]
```

## Question 3

Normalizing the training data using min-max normalization

```r
list <- c(1,2,3,4,5,8,9,10,11,12)
for(i in list)
{
  train_data[,i] <- (train_data[,i] - min(train_data[,i])) / (max(train_data[,i]) -
                                                              min(train_data[,i]))
}
```

## Question 4

Building a knn regression function.

```r
kNN.reg <- function(new_data, target_data, train_data, k)
{
  n <- nrow(train_data)
  d <- rep(0,n)
  for (i in 1:n)
  {
    d[i] <- sqrt(sum((train_data[i,1:12] - new_data[1:12])^2))
  }
  o <- order(d)
  m <- mean(target_data[o[1:k]])
  return(m)
}
```

## Question 5

Using the knn.reg function to predict the price values for the given set of values.

```r
new_data <- c(4, 3, 4852, 9812, 3, 0, 1, 3, 11, 1860, 820, 1962)
for(i in list)
{
  new_data[i] <- (new_data[i] - min(train_data[,i])) / (max(train_data[,i]) -
                                                         min(train_data[,i]))
}
kNN.reg(new_data, target_data, train_data, 4)
```

```
## [1] 1358750
```