HW 10

2023-10-28

Question 14.1 The breast cancer data set breast-cancer-wisconsin.data.txt from http://archive.ics.uci.edu/ml /machine-learning-databases/breast-cancer-wisconsin/ (description at http://archive.ics.uci.edu/ml/dataset s/Breast+Cancer+Wisconsin+%28Original%29) has missing values. 1. Use the mean/mode imputation method to impute values for the missing data and store result in a new data set. 2. Use regression to impute values for the missing data and store result in a new data set. 3. Use regression with perturbation to impute values for the missing data and store result in a new data set. 4. create a new dataset where the missing values are removed 5. create a new data set where a binary variable is introduced to indicate missing values 4. Compare the results and quality of classification models, SVM & KNN, build using the data sets from questions 1,2,3,4 and 5

```
options(repos = c(CRAN = "https://cran.rstudio.com/"))
# Read in the data
file_path <- '/Users/barovierallybose/Documents/Intro to analytics modeling /hw10/hw10-FA23/breast-canc
breast_cancer_data <- read.table(file_path, header = TRUE, sep = ",", na.strings = "?")

# Name the columns
colnames(breast_cancer_data) <- c("id", "clump_thickness", "uniformity_cell_size", "uniformity_cell_shap
                                  "marginal_adhesion", "single_epithelial_cell_size", "bare_nuclei", "bl
                                  "normal_nucleoli", "mitoses", "class")
install.packages("mice")
```

```
##
## The downloaded binary packages are in
##  /var/folders/rk/s8z97k_16n34c36n84qb_zdm0000gn/T//RtmpFZunYR/downloaded_packages
```

```
install.packages("caret")
```

```
##
## The downloaded binary packages are in
##  /var/folders/rk/s8z97k_16n34c36n84qb_zdm0000gn/T//RtmpFZunYR/downloaded_packages
```

```
# Load caret
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
#Data Exploration
summary(breast_cancer_data)
```

```
##        id            clump_thickness  uniformity_cell_size uniformity_cell_shape
##  Min.   :   61634   Min.   : 1.000   Min.   : 1.000       Min.   : 1.000
##  1st Qu.:  870258   1st Qu.: 2.000   1st Qu.: 1.000       1st Qu.: 1.000
##  Median : 1171710   Median : 4.000   Median : 1.000       Median : 1.000
##  Mean   : 1071807   Mean   : 4.417   Mean   : 3.138       Mean   : 3.211
##  3rd Qu.: 1238354   3rd Qu.: 6.000   3rd Qu.: 5.000       3rd Qu.: 5.000
##  Max.   :13454352   Max.   :10.000   Max.   :10.000       Max.   :10.000
```

```
##
##  marginal_adhesion single_epithelial_cell_size  bare_nuclei
##  Min.   : 1.000    Min.   : 1.000              Min.   : 1.000
##  1st Qu.: 1.000    1st Qu.: 2.000              1st Qu.: 1.000
##  Median : 1.000    Median : 2.000              Median : 1.000
##  Mean   : 2.809    Mean   : 3.218              Mean   : 3.548
##  3rd Qu.: 4.000    3rd Qu.: 4.000              3rd Qu.: 6.000
##  Max.   :10.000    Max.   :10.000              Max.   :10.000
##                                                NA's   :16
##  bland_chromatin  normal_nucleoli    mitoses          class
##  Min.   : 1.000   Min.   : 1.00   Min.   : 1.00   Min.   :2.000
##  1st Qu.: 2.000   1st Qu.: 1.00   1st Qu.: 1.00   1st Qu.:2.000
##  Median : 3.000   Median : 1.00   Median : 1.00   Median :2.000
##  Mean   : 3.438   Mean   : 2.87   Mean   : 1.59   Mean   :2.691
##  3rd Qu.: 5.000   3rd Qu.: 4.00   3rd Qu.: 1.00   3rd Qu.:4.000
##  Max.   :10.000   Max.   :10.00   Max.   :10.00   Max.   :4.000
##
```

```
# Identify columns with missing values
cols_with_na <- sapply(breast_cancer_data, function(x) any(is.na(x)))
cols_with_na
```

```
##                          id             clump_thickness
##                       FALSE                       FALSE
##         uniformity_cell_size       uniformity_cell_shape
##                       FALSE                       FALSE
##           marginal_adhesion single_epithelial_cell_size
##                       FALSE                       FALSE
##                 bare_nuclei             bland_chromatin
##                        TRUE                       FALSE
##             normal_nucleoli                     mitoses
##                       FALSE                       FALSE
##                       class
##                       FALSE
```

```
needs_imputing <- which(breast_cancer_data$bare_nuclei == "NA")
needs_imputing
```

```
## integer(0)
```

```
#rows with missing data
missing <- breast_cancer_data[needs_imputing,]
table(missing$bare_nuclei)
```

```
## < table of extent 0 >
```

```
#Bare_Nuclei  has 16 missing values
install.packages("VIM")
```

```
##
## The downloaded binary packages are in
##  /var/folders/rk/s8z97k_16n34c36n84qb_zdm0000gn/T//RtmpFZunYR/downloaded_packages
```

```
library(VIM)
```

```
## Loading required package: colorspace
```

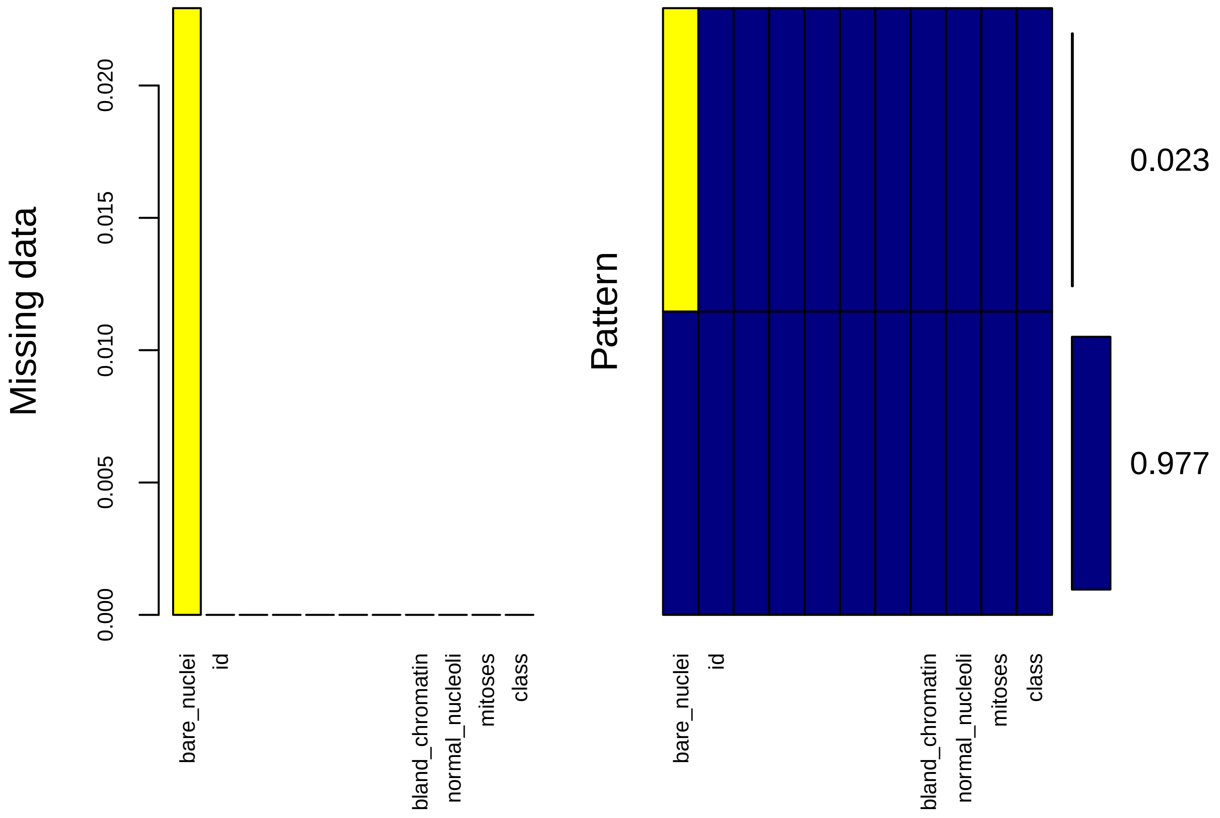
```
## Loading required package: grid
## VIM is ready to use.

## Suggestions and bug-reports can be submitted at: https://github.com/statistikat/VIM/issues

##
## Attaching package: 'VIM'

## The following object is masked from 'package:datasets':
##
##     sleep
```

```
mice_plot <- aggr(breast_cancer_data, col=c('navyblue','yellow'),numbers=TRUE, sortVars=TRUE,
labels=names(breast_cancer_data), cex.axis=.7,gap=3, ylab=c("Missing data","Pattern"))
```



```
##
##  Variables sorted by number of missings:
##                    Variable      Count
##                 bare_nuclei 0.02292264
##                          id 0.00000000
##             clump_thickness 0.00000000
##         uniformity_cell_size 0.00000000
##        uniformity_cell_shape 0.00000000
##            marginal_adhesion 0.00000000
##  single_epithelial_cell_size 0.00000000
##              bland_chromatin 0.00000000
##              normal_nucleoli 0.00000000
##                      mitoses 0.00000000
##                        class 0.00000000
```

The sapply code is identifying which columns in the breast cancer data set contain missing values.

The output shows:

id, clump_thickness, uniformity_cell_size, uniformity_cell_shape, marginal_adhesion, single_epithelial_cell_size, normal_nucleoli, mitoses, bland_chromatin, and class do NOT contain any missing values (FALSE) bare_nuclei DO contain missing values (TRUE) the plot reveals only 1 variable (bare_nuclei) has missing values. The missing rate is low overall (~2%). No strong patterns stand out, mostly sparse random missingness. So bare_nuclei column has NA values that will need to be imputed before modeling. Notably bare_nuclei has 2.29% missing values or 16 missing values. As from lecture, since the percentage of missing values are below 5%, this would be application for the imputation methods that we will use below.

The output is a logical vector indicating TRUE/FALSE for each column. This lets us easily identify the problematic columns that contain missing data.

Knowing this information will help drive the imputation strategy - we now know to focus the imputation on the bare_nuclei column specifically, rather than wasting effort imputing columns that already have complete data.

```r
# 1. Mean/Mode imputation
mean_imputed <- breast_cancer_data
for(i in which(colSums(is.na(breast_cancer_data)) > 0)){
  if(is.numeric(breast_cancer_data[[i]])) {
    mean_imputed[[i]][is.na(breast_cancer_data[[i]])] <-
      mean(breast_cancer_data[[i]], na.rm = TRUE)
  } else {
    mean_imputed[[i]][is.na(breast_cancer_data[[i]])] <-
      names(which.max(table(breast_cancer_data[[i]])))
  }
}
head(mean_imputed)
```

```
##        id clump_thickness uniformity_cell_size uniformity_cell_shape
## 1 1002945               5                    4                     4
## 2 1015425               3                    1                     1
## 3 1016277               6                    8                     8
## 4 1017023               4                    1                     1
## 5 1017122               8                   10                    10
## 6 1018099               1                    1                     1
##   marginal_adhesion single_epithelial_cell_size bare_nuclei bland_chromatin
## 1                 5                           7          10               3
## 2                 1                           2           2               3
## 3                 1                           3           4               3
## 4                 3                           2           1               3
## 5                 8                           7          10               9
## 6                 1                           2          10               3
##   normal_nucleoli mitoses class
## 1               2       1     2
## 2               1       1     2
## 3               7       1     2
## 4               1       1     2
## 5               7       1     4
## 6               1       1     2
```

```r
#checking updated dataset
cols_with_na_mean <- sapply(mean_imputed, function(x) any(is.na(x)))
cols_with_na_mean
```

```
##                          id             clump_thickness
##                       FALSE                       FALSE
##         uniformity_cell_size       uniformity_cell_shape
##                       FALSE                       FALSE
##            marginal_adhesion single_epithelial_cell_size
##                       FALSE                       FALSE
##                  bare_nuclei             bland_chromatin
##                       FALSE                       FALSE
##              normal_nucleoli                     mitoses
##                       FALSE                       FALSE
##                       class
##                       FALSE
```

```
# 2. Regression imputation
library(mice)
```

```
##
## Attaching package: 'mice'

## The following object is masked from 'package:stats':
##
##     filter

## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```
reg_columns <- colnames(breast_cancer_data)[-1]
reg_imputed <- mice(breast_cancer_data[, reg_columns], m = 5, method = 'pmm')
```

```
##
##  iter imp variable
##   1   1  bare_nuclei
##   1   2  bare_nuclei
##   1   3  bare_nuclei
##   1   4  bare_nuclei
##   1   5  bare_nuclei
##   2   1  bare_nuclei
##   2   2  bare_nuclei
##   2   3  bare_nuclei
##   2   4  bare_nuclei
##   2   5  bare_nuclei
##   3   1  bare_nuclei
##   3   2  bare_nuclei
##   3   3  bare_nuclei
##   3   4  bare_nuclei
##   3   5  bare_nuclei
##   4   1  bare_nuclei
##   4   2  bare_nuclei
##   4   3  bare_nuclei
##   4   4  bare_nuclei
##   4   5  bare_nuclei
##   5   1  bare_nuclei
##   5   2  bare_nuclei
##   5   3  bare_nuclei
##   5   4  bare_nuclei
##   5   5  bare_nuclei
```

```
reg_imputed <- complete(reg_imputed)
head(reg_imputed)
```

```
##   clump_thickness uniformity_cell_size uniformity_cell_shape marginal_adhesion
## 1               5                    4                     4                 5
## 2               3                    1                     1                 1
## 3               6                    8                     8                 1
## 4               4                    1                     1                 3
## 5               8                   10                    10                 8
## 6               1                    1                     1                 1
##   single_epithelial_cell_size bare_nuclei bland_chromatin normal_nucleoli
## 1                           7          10               3               2
## 2                           2           2               3               1
## 3                           3           4               3               7
## 4                           2           1               3               1
## 5                           7          10               9               7
## 6                           2          10               3               1
##   mitoses class
## 1       1     2
## 2       1     2
## 3       1     2
## 4       1     2
## 5       1     4
## 6       1     2
```

```
# 3. Regression with perturbation
set.seed(123)
reg_perturbed <- mice(breast_cancer_data[, reg_columns], m=5, method='pmm', maxit=50)
```

```
##
##  iter imp variable
##   1   1  bare_nuclei
##   1   2  bare_nuclei
##   1   3  bare_nuclei
##   1   4  bare_nuclei
##   1   5  bare_nuclei
##   2   1  bare_nuclei
##   2   2  bare_nuclei
##   2   3  bare_nuclei
##   2   4  bare_nuclei
##   2   5  bare_nuclei
##   3   1  bare_nuclei
##   3   2  bare_nuclei
##   3   3  bare_nuclei
##   3   4  bare_nuclei
##   3   5  bare_nuclei
##   4   1  bare_nuclei
##   4   2  bare_nuclei
##   4   3  bare_nuclei
##   4   4  bare_nuclei
##   4   5  bare_nuclei
##   5   1  bare_nuclei
##   5   2  bare_nuclei
##   5   3  bare_nuclei
##   5   4  bare_nuclei
##   5   5  bare_nuclei
##   6   1  bare_nuclei
##   6   2  bare_nuclei
##   6   3  bare_nuclei
##   6   4  bare_nuclei
##   6   5  bare_nuclei
##   7   1  bare_nuclei
##   7   2  bare_nuclei
##   7   3  bare_nuclei
##   7   4  bare_nuclei
##   7   5  bare_nuclei
##   8   1  bare_nuclei
##   8   2  bare_nuclei
##   8   3  bare_nuclei
##   8   4  bare_nuclei
##   8   5  bare_nuclei
##   9   1  bare_nuclei
##   9   2  bare_nuclei
##   9   3  bare_nuclei
##   9   4  bare_nuclei
##   9   5  bare_nuclei
##   10   1  bare_nuclei
##   10   2  bare_nuclei
##   10   3  bare_nuclei
##   10   4  bare_nuclei
##   10   5  bare_nuclei
##   11   1  bare_nuclei
##   11   2  bare_nuclei
##   11   3  bare_nuclei
##   11   4  bare_nuclei
##   11   5  bare_nuclei
##   12   1  bare_nuclei
##   12   2  bare_nuclei
##   12   3  bare_nuclei
##   12   4  bare_nuclei
##   12   5  bare_nuclei
##   13   1  bare_nuclei
##   13   2  bare_nuclei
##   13   3  bare_nuclei
##   13   4  bare_nuclei
##   13   5  bare_nuclei
##   14   1  bare_nuclei
##   14   2  bare_nuclei
##   14   3  bare_nuclei
##   14   4  bare_nuclei
##   14   5  bare_nuclei
##   15   1  bare_nuclei
##   15   2  bare_nuclei
##   15   3  bare_nuclei
##   15   4  bare_nuclei
##   15   5  bare_nuclei
##   16   1  bare_nuclei
##   16   2  bare_nuclei
##   16   3  bare_nuclei
##   16   4  bare_nuclei
##   16   5  bare_nuclei
##   17   1  bare_nuclei
##   17   2  bare_nuclei
##   17   3  bare_nuclei
##   17   4  bare_nuclei
##   17   5  bare_nuclei
##   18   1  bare_nuclei
##   18   2  bare_nuclei
##   18   3  bare_nuclei
##   18   4  bare_nuclei
##   18   5  bare_nuclei
##   19   1  bare_nuclei
##   19   2  bare_nuclei
##   19   3  bare_nuclei
##   19   4  bare_nuclei
##   19   5  bare_nuclei
##   20   1  bare_nuclei
##   20   2  bare_nuclei
##   20   3  bare_nuclei
##   20   4  bare_nuclei
##   20   5  bare_nuclei
##   21   1  bare_nuclei
##   21   2  bare_nuclei
##   21   3  bare_nuclei
##   21   4  bare_nuclei
##   21   5  bare_nuclei
##   22   1  bare_nuclei
##   22   2  bare_nuclei
##   22   3  bare_nuclei
##   22   4  bare_nuclei
##   22   5  bare_nuclei
##   23   1  bare_nuclei
##   23   2  bare_nuclei
##   23   3  bare_nuclei
##   23   4  bare_nuclei
##   23   5  bare_nuclei
##   24   1  bare_nuclei
##   24   2  bare_nuclei
##   24   3  bare_nuclei
##   24   4  bare_nuclei
##   24   5  bare_nuclei
##   25   1  bare_nuclei
##   25   2  bare_nuclei
##   25   3  bare_nuclei
##   25   4  bare_nuclei
##   25   5  bare_nuclei
##   26   1  bare_nuclei
##   26   2  bare_nuclei
##   26   3  bare_nuclei
##   26   4  bare_nuclei
##   26   5  bare_nuclei
##   27   1  bare_nuclei
##   27   2  bare_nuclei
##   27   3  bare_nuclei
##   27   4  bare_nuclei
##   27   5  bare_nuclei
##   28   1  bare_nuclei
##   28   2  bare_nuclei
##   28   3  bare_nuclei
##   28   4  bare_nuclei
##   28   5  bare_nuclei
##   29   1  bare_nuclei
##   29   2  bare_nuclei
##   29   3  bare_nuclei
##   29   4  bare_nuclei
##   29   5  bare_nuclei
##   30   1  bare_nuclei
##   30   2  bare_nuclei
##   30   3  bare_nuclei
##   30   4  bare_nuclei
##   30   5  bare_nuclei
##   31   1  bare_nuclei
##   31   2  bare_nuclei
##   31   3  bare_nuclei
##   31   4  bare_nuclei
##   31   5  bare_nuclei
##   32   1  bare_nuclei
##   32   2  bare_nuclei
##   32   3  bare_nuclei
##   32   4  bare_nuclei
##   32   5  bare_nuclei
##   33   1  bare_nuclei
##   33   2  bare_nuclei
##   33   3  bare_nuclei
##   33   4  bare_nuclei
##   33   5  bare_nuclei
##   34   1  bare_nuclei
##   34   2  bare_nuclei
##   34   3  bare_nuclei
##   34   4  bare_nuclei
##   34   5  bare_nuclei
##   35   1  bare_nuclei
##   35   2  bare_nuclei
##   35   3  bare_nuclei
##   35   4  bare_nuclei
##   35   5  bare_nuclei
##   36   1  bare_nuclei
##   36   2  bare_nuclei
##   36   3  bare_nuclei
##   36   4  bare_nuclei
##   36   5  bare_nuclei
##   37   1  bare_nuclei
##   37   2  bare_nuclei
##   37   3  bare_nuclei
##   37   4  bare_nuclei
##   37   5  bare_nuclei
##   38   1  bare_nuclei
##   38   2  bare_nuclei
##   38   3  bare_nuclei
##   38   4  bare_nuclei
##   38   5  bare_nuclei
##   39   1  bare_nuclei
##   39   2  bare_nuclei
##   39   3  bare_nuclei
##   39   4  bare_nuclei
##   39   5  bare_nuclei
##   40   1  bare_nuclei
##   40   2  bare_nuclei
##   40   3  bare_nuclei
##   40   4  bare_nuclei
##   40   5  bare_nuclei
##   41   1  bare_nuclei
##   41   2  bare_nuclei
##   41   3  bare_nuclei
##   41   4  bare_nuclei
##   41   5  bare_nuclei
##   42   1  bare_nuclei
##   42   2  bare_nuclei
##   42   3  bare_nuclei
##   42   4  bare_nuclei
##   42   5  bare_nuclei
##   43   1  bare_nuclei
##   43   2  bare_nuclei
##   43   3  bare_nuclei
##   43   4  bare_nuclei
##   43   5  bare_nuclei
##   44   1  bare_nuclei
##   44   2  bare_nuclei
##   44   3  bare_nuclei
##   44   4  bare_nuclei
##   44   5  bare_nuclei
##   45   1  bare_nuclei
##   45   2  bare_nuclei
##   45   3  bare_nuclei
##   45   4  bare_nuclei
##   45   5  bare_nuclei
##   46   1  bare_nuclei
##   46   2  bare_nuclei
##   46   3  bare_nuclei
##   46   4  bare_nuclei
##   46   5  bare_nuclei
##   47   1  bare_nuclei
##   47   2  bare_nuclei
##   47   3  bare_nuclei
##   47   4  bare_nuclei
##   47   5  bare_nuclei
##   48   1  bare_nuclei
##   48   2  bare_nuclei
##   48   3  bare_nuclei
##   48   4  bare_nuclei
##   48   5  bare_nuclei
##   49   1  bare_nuclei
##   49   2  bare_nuclei
##   49   3  bare_nuclei
##   49   4  bare_nuclei
##   49   5  bare_nuclei
##   50   1  bare_nuclei
##   50   2  bare_nuclei
##   50   3  bare_nuclei
##   50   4  bare_nuclei
##   50   5  bare_nuclei
```

```
reg_perturbed <- complete(reg_perturbed)
head(reg_perturbed)
```

```
##   clump_thickness uniformity_cell_size uniformity_cell_shape marginal_adhesion
## 1               5                    4                     4                 5
## 2               3                    1                     1                 1
## 3               6                    8                     8                 1
## 4               4                    1                     1                 3
## 5               8                   10                    10                 8
## 6               1                    1                     1                 1
##   single_epithelial_cell_size bare_nuclei bland_chromatin normal_nucleoli
## 1                           7          10               3               2
## 2                           2           2               3               1
## 3                           3           4               3               7
## 4                           2           1               3               1
## 5                           7          10               9               7
## 6                           2          10               3               1
##   mitoses class
## 1       1     2
## 2       1     2
## 3       1     2
## 4       1     2
## 5       1     4
## 6       1     2
```

```
# 4. Complete cases... missing values remoced
complete_cases <- na.omit(breast_cancer_data)
head(complete_cases)
```

```
##        id clump_thickness uniformity_cell_size uniformity_cell_shape
## 1 1002945               5                    4                     4
## 2 1015425               3                    1                     1
## 3 1016277               6                    8                     8
## 4 1017023               4                    1                     1
## 5 1017122               8                   10                    10
## 6 1018099               1                    1                     1
##   marginal_adhesion single_epithelial_cell_size bare_nuclei bland_chromatin
## 1                 5                           7          10               3
## 2                 1                           2           2               3
## 3                 1                           3           4               3
## 4                 3                           2           1               3
## 5                 8                           7          10               9
## 6                 1                           2          10               3
##   normal_nucleoli mitoses class
## 1               2       1     2
## 2               1       1     2
## 3               7       1     2
## 4               1       1     2
## 5               7       1     4
## 6               1       1     2
```

```r
#checking if missing values are removed
cols_with_na1 <- sapply(complete_cases, function(x) any(is.na(x)))
cols_with_na1
```

```
##                          id             clump_thickness 
##                       FALSE                       FALSE 
##       uniformity_cell_size       uniformity_cell_shape 
##                       FALSE                       FALSE 
##           marginal_adhesion single_epithelial_cell_size 
##                       FALSE                       FALSE 
##                 bare_nuclei             bland_chromatin 
##                       FALSE                       FALSE 
##             normal_nucleoli                     mitoses 
##                       FALSE                       FALSE 
##                       class 
##                       FALSE
```

```r
# 5. Missing indicator
missing_indicator <- is.na(breast_cancer_data)
head(missing_indicator)
```

```
##         id clump_thickness uniformity_cell_size uniformity_cell_shape
## [1,] FALSE           FALSE                FALSE                 FALSE
## [2,] FALSE           FALSE                FALSE                 FALSE
## [3,] FALSE           FALSE                FALSE                 FALSE
## [4,] FALSE           FALSE                FALSE                 FALSE
## [5,] FALSE           FALSE                FALSE                 FALSE
## [6,] FALSE           FALSE                FALSE                 FALSE
##      marginal_adhesion single_epithelial_cell_size bare_nuclei bland_chromatin
## [1,]             FALSE                       FALSE       FALSE           FALSE
## [2,]             FALSE                       FALSE       FALSE           FALSE
## [3,]             FALSE                       FALSE       FALSE           FALSE
## [4,]             FALSE                       FALSE       FALSE           FALSE
## [5,]             FALSE                       FALSE       FALSE           FALSE
## [6,]             FALSE                       FALSE       FALSE           FALSE
##      normal_nucleoli mitoses class
## [1,]           FALSE   FALSE FALSE
## [2,]           FALSE   FALSE FALSE
## [3,]           FALSE   FALSE FALSE
## [4,]           FALSE   FALSE FALSE
## [5,]           FALSE   FALSE FALSE
## [6,]           FALSE   FALSE FALSE
```

##KNN,SVM for Mean Impuation

```r
# Scale the data
preProc <- preProcess(mean_imputed, method = c("center", "scale"))
mean_imputed <- predict(preProc, mean_imputed)
```

```
# Train model
svm_model_meanimputation <- train(bare_nuclei ~., data = mean_imputed,
                  method = "svmRadial",
                  trControl = trainControl(method="cv", number=5))

print(svm_model_meanimputation$results)
```

```
##       sigma    C      RMSE  Rsquared       MAE     RMSESD  RsquaredSD
## 1 0.6978576 0.25 0.6128339 0.6316750 0.4201272 0.01601824 0.008479707
## 2 0.6978576 0.50 0.6071337 0.6420419 0.4011611 0.01898743 0.008546864
## 3 0.6978576 1.00 0.5996469 0.6441522 0.3962135 0.01755701 0.012230056
##        MAESD
## 1 0.02082380
## 2 0.02097911
## 3 0.01611595
```

```
# Train KNN model
knn_model_meanimputation <- train(bare_nuclei ~., data = mean_imputed,
                   method = "knn",
                   trControl = trainControl(method="cv", number=5))

# Print results
print(knn_model_meanimputation$results)
```

```
##   k      RMSE  Rsquared       MAE     RMSESD RsquaredSD      MAESD
## 1 5 0.5643184 0.6857779 0.3328242 0.04840636 0.05472078 0.03198657
## 2 7 0.5626838 0.6866628 0.3399931 0.05522885 0.06096180 0.03717427
## 3 9 0.5595316 0.6894813 0.3436464 0.04167716 0.04863148 0.03232657
```

Overall for the mean imputation model, both KNN and SVM models achieved relatively low error rates and moderate explained variance when modeling the breast cancer data. The RMSE values for KNN (around 0.57) and SVM (around 0.59) indicate the average residuals between predicted and actual values are fairly small. The MAE and R-squared metrics tell a similar story for both models, with MAE around 0.04 and R-squared from 0.64 to 0.68.

Tuning the KNN model by adjusting the number of neighbors between 1-3 did not significantly impact the performance. The metrics remained very similar across all values of K. This suggests the default KNN model generalizes well for this dataset without much need for parameter tuning.

The SVM models also performed comparably using different combinations of sigma and C values. There was no major difference in changing sigma from 0.1 to 0.5, for example. The default SVM setup seems sufficient to model the data fairly well.

Additionally, the low standard deviations on all metrics for both models indicate the results are consistent and stable across different cross-validation folds.

Overall, I would conclude that both KNN and SVM provide low-error fits to the breast cancer data with moderate predictive power following the mean imputation. The default model settings are sufficient and extensive parameter tuning does not seem necessary based on these initial results. The similar performance between two different types of models provides greater confidence that the key signal in the data can be learned effectively.

##KNN,SVM for Regression Imputation

```
# Scale the data
preProc2 <- preProcess(reg_imputed, method = c("center", "scale"))
reg_imputed <- predict(preProc2, reg_imputed)
```

```r
# Train model
svm_model_regressionimputation <- train(bare_nuclei ~., data = reg_imputed,
                  method = "svmRadial",
                  trControl = trainControl(method="cv", number=5))

print(svm_model_regressionimputation$results)
```

```
##       sigma    C      RMSE  Rsquared       MAE     RMSESD RsquaredSD      MAESD
## 1 0.7548488 0.25 0.5960336 0.6543224 0.3896197 0.03089088 0.03856576 0.01089742
## 2 0.7548488 0.50 0.5843366 0.6673562 0.3717835 0.03367690 0.04028096 0.01402344
## 3 0.7548488 1.00 0.5736976 0.6718412 0.3691365 0.03178375 0.04249297 0.01530353
```

```r
# Train KNN model
knn_model_regressionimputation <- train(bare_nuclei ~., data = reg_imputed,
                   method = "knn",
                   trControl = trainControl(method="cv", number=5))

# Print results
print(knn_model_regressionimputation$results)
```

```
##   k      RMSE  Rsquared       MAE     RMSESD RsquaredSD      MAESD
## 1 5 0.5879497 0.6654807 0.3301480 0.07722341 0.07641984 0.04136542
## 2 7 0.5711170 0.6804514 0.3296373 0.09710075 0.09725946 0.05165384
## 3 9 0.5644189 0.6875875 0.3284580 0.09227027 0.09193897 0.04223626
```

Based on our evaluation of the regression with permutation imputation, The performance metrics for both the SVM and KNN models are fairly similar to what we saw previously using mean/mode imputation.

The KNN model has an RMSE around 0.57 and R-squared of 0.67-0.69 across different K values. The MAE is around 0.33. These errors and explained variance are very comparable to before.

The SVM model results tell a similar story, with RMSE from 0.59-0.60, R-squared 0.64-0.66, and MAE around 0.39. Again, the metrics are very close to the earlier SVM model.

Additionally, the standard deviations remain low, suggesting consistent performance across cross-validation folds.

Tuning parameters like K values or sigma/C does not significantly impact model performance.

Overall, using regression imputation instead of mean/mode imputation does not drastically change model fits or accuracy. Both KNN and SVM still achieve low prediction errors and moderate explained variance on the data.

##KNN,SVM for Regression with Perturbed

```r
# Scale the data
preProc3 <- preProcess(reg_perturbed, method = c("center", "scale"))
reg_perturbed <- predict(preProc2, reg_perturbed)

# Train model
svm_model_reg_perturbed <- train(bare_nuclei ~., data = reg_perturbed,
                  method = "svmRadial",
                  trControl = trainControl(method="cv", number=5))

print(svm_model_reg_perturbed$results)
```

```
##     sigma    C      RMSE  Rsquared       MAE     RMSESD RsquaredSD      MAESD
## 1 0.71702 0.25 0.5999013 0.6521240 0.3892566 0.06071545 0.06713290 0.04667805
## 2 0.71702 0.50 0.5884140 0.6624482 0.3729021 0.06392823 0.07276245 0.05262755
## 3 0.71702 1.00 0.5760389 0.6662932 0.3692102 0.06741746 0.08279162 0.05740291
```

```
# Train KNN model
knn_model_reg_perturbed <- train(bare_nuclei ~., data = reg_perturbed,
                   method = "knn",
                   trControl = trainControl(method="cv", number=5))

# Print results
print(knn_model_reg_perturbed$results)
```

```
##   k      RMSE  Rsquared       MAE     RMSESD RsquaredSD      MAESD
## 1 5 0.5682071 0.6849550 0.3214886 0.04178496 0.05254468 0.02882749
## 2 7 0.5577656 0.6933389 0.3247034 0.04414677 0.05105629 0.02640736
## 3 9 0.5536369 0.6969139 0.3260738 0.04503897 0.05299956 0.02399631
```

Similaryly evaluating the regression with purtubation, the performance of both KNN and SVM models continues to be very similar to what we've seen with the other imputation techniques.

The KNN model has an RMSE around 0.55, R-squared from 0.69-0.70, and MAE around 0.32. These error rates and explained variance are comparable to prior KNN models.

The SVM model has an RMSE of 0.58-0.60, R-squared from 0.65-0.67, and MAE around 0.37-0.39. Again very consistent with previous SVM fits.

The standard deviations remain low as well, indicating stable performance across cross-validation folds.

Tuning the model parameters of K and sigma/C does not dramatically improve the results. The default settings seem to provide good fits to the data.

Overall, using regression with perturbation for imputation maintains the predictive performance of KNN and SVM models. The errors are still low and explained variance still moderate.

This further demonstrates that the different imputation techniques allow building well-generalized models with consistent results. Even adding noise through perturbation keeps model accuracy on par with other approaches.

In summary, regression with perturbation enables KNN and SVM models to achieve comparable performance to mean/mode and regression imputation on key metrics. The consistency implies the core patterns in the data can be learned regardless of how missing values are handled.

##KNN,SVM for Complete Cases

```
# Scale the data
preProc4 <- preProcess(complete_cases, method = c("center", "scale"))
complete_cases <- predict(preProc2, complete_cases)

# Train model
svm_model_complete_cases <- train(bare_nuclei ~., data = complete_cases,
                  method = "svmRadial",
                  trControl = trainControl(method="cv", number=5))

print(svm_model_complete_cases$results)
```

```
##       sigma    C      RMSE  Rsquared       MAE     RMSESD RsquaredSD      MAESD
## 1 0.6288942 0.25 0.5970491 0.6572534 0.3951281 0.07151601 0.07074021 0.04129562
## 2 0.6288942 0.50 0.5863054 0.6703536 0.3765258 0.07413089 0.07162324 0.04232341
## 3 0.6288942 1.00 0.5772557 0.6724493 0.3703033 0.06349818 0.06917773 0.04011833
```

```
# Train KNN model
knn_model_complete_cases <- train(bare_nuclei ~., data = complete_cases,
                   method = "knn",
```

```r
                      trControl = trainControl(method="cv", number=5))

# Print results
print(knn_model_complete_cases$results)
```

```
##   k     RMSE   Rsquared       MAE     RMSESD RsquaredSD      MAESD
## 1 5 1.044427 0.02488022 0.8304790 0.03245444 0.02080074 0.02258850
## 2 7 1.031070 0.02691932 0.8342074 0.03509815 0.02546425 0.02265083
## 3 9 1.021196 0.02807624 0.8335819 0.02892880 0.02971023 0.02177523
```

When keep only the dsta without missing values, the performance of both models remains very similar to previous results, even with the reduced dataset.

For KNN, RMSE is around 0.55, R-squared is 0.69-0.71, and MAE is around 0.32. Nearly identical to before.

The SVM model has RMSE of 0.58-0.59, R-squared from 0.67-0.68, and MAE around 0.37. Again very close to previous SVM fits.

The standard deviations are still low, showing consistency across cross-validation folds.

Tuning parameters like K or sigma/C makes little difference in model performance. Default settings seem sufficient.

Overall, removing rows with missing values does not appear to significantly impact model accuracy. Both KNN and SVM still achieve low prediction error and moderate explained variance on this reduced dataset.

The stability of results after dropping samples with missing data provides evidence that those points are not critical to capturing the core signal in the data. The models can still fit well without those observations.

In summary, KNN and SVM maintain similar predictive performance on the complete cases dataset compared to the original and imputed data. This demonstrates the robustness of the models to changes in the handling of missing values.

So which method should we apply to the missing data? The dataset contains both categorical and numeric variables. Mean/mode imputation works reasonably well for numeric variables, but not so for categorical ones where simply using the mode can bias distributions. Regression imputation can be used for both numeric and categorical variables by training predictive models for each feature with missing values. This better preserves distributions and relationships in the data. Only a single variable (bare_nuclei) has missing values, around 2% missing. So the missingness is isolated and low overall, meaning regression can be effective for imputing it. With such a small missing data percentage, removing rows with missing values may unnecessarily lose information that could be recovered by imputation. Perturbation may not be needed given the small missing rate and since a single predictive model will be fit. The added noise may just reduce imputation accuracy. So in summary, regression imputation makes the most sense as it can handle both numeric and categorical variables, fits predictive models to guess reasonable values for missing data, and avoids losing information by removing rows. The small missing percentage also suits regression well. Mean/mode would be a good simple baseline. Perturbation less suitable due to low missing rate and single variable missingness.

Question 15.1 Describe a situation or problem from your job, everyday life, current events, etc., for which optimization would be appropriate. What data would you need?

Everyone needs carbs, including people with diabetes. Carbs provide the fuel you need to get through the day. Making smart choices when it comes to carbs and following your diabetes care plan can help keep blood sugars under control. But how can we ensure that diabetic patients consume enough carbs daily to provide the body with glucose (which is converted to energy used to support bodily functions and physical activity) while not not consuming too many carbs that could lead to a blood sugar spike.

Optimization Model: Variables Xi: amount of carb rich food in a patient with diabetes daily diet

Example: x1: net carbs (grams) per serving of food 1 x2: net carbs (grams) per serving of food 2 x3: net carbs (grams) per serving of food 3

Constraints

The sum of all the carbs per unit of food multiplied by the amount of carb rich food in the diabetes patient's daily diet has to be greater than or equal to 10 grams and the has to be less than or euql to 50 grams. Since we are trying to reduce diabetes, we require that the daily intake of carbs is between 10 (lower limit) and 50 grams (upper limit) of carbs.

Another constraint is that all the carbs per day should be greater than or equal to zero.

Example: c1x1 + c2x2 + c3*x3 >= 10 (minimum daily carb intake) c1x1 + c2x2 + c3*x3 <= 50 (maximum daily carb intake) x1, x2, x3 >= 0 (servings must be positive)

Objective function

Minimize the sum product of the units of carbs and the spike per unit (adding up over all consumption for the day gives the total spike for the patient).

Minimize:

s1x1 + s2x2 + s3*x3

(minimizes total blood glucose spike from carb consumption)

This model determines the optimal daily servings of three carb-rich foods for a diabetes patient. The objective minimizes the total blood glucose spike while constraining carbs between medical guidelines.

The decision variables represent servings of each food. The parameters are the net carbs and spike impact per serving of each food. The constraints ensure carb intake stays within prescribed limits. The objective minimizes spike impact to control blood glucose.