HW6

Question 14.1

The breast cancer data set breast-cancer-wisconsin.data.txtfromhttp://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/ (description at http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29 (http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29)) has missing values.

1.

Use the mean/mode imputation method to impute values for the missing data.

```
brc_data <- read.table("/Users/chintan/Downloads/6501/breast-cancer-wisconsin.data.txt",
                      stringsAsFactors = FALSE, sep = ',',header = TRUE)
summary(brc_data)
```

```
##     X1000025             X5               X1              X1.1             X1.2
X2
##  Min.   :   61634   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
Min.   : 1.000
##  1st Qu.:  870258   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.: 1.000
1st Qu.: 2.000
##  Median : 1171710   Median : 4.000   Median : 1.000   Median : 1.000   Median : 1.000
Median : 2.000
##  Mean   : 1071807   Mean   : 4.417   Mean   : 3.138   Mean   : 3.211   Mean   : 2.809
Mean   : 3.218
##  3rd Qu.: 1238354   3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000   3rd Qu.: 4.000
3rd Qu.: 4.000
##  Max.   :13454352   Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000
Max.   :10.000
##      X1.3               X3              X1.4            X1.5            X2.1
##  Length:698         Min.   : 1.000   Min.   : 1.00   Min.   : 1.00   Min.   :2.000
##  Class :character   1st Qu.: 2.000   1st Qu.: 1.00   1st Qu.: 1.00   1st Qu.:2.000
##  Mode  :character   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
```

```
head(brc_data)
```

```
##   X1000025 X5 X1 X1.1 X1.2 X2 X1.3 X3 X1.4 X1.5 X2.1
## 1  1002945  5  4    4    5  7   10  3    2    1    2
## 2  1015425  3  1    1    1  2    2  3    1    1    2
## 3  1016277  6  8    8    1  3    4  3    7    1    2
## 4  1017023  4  1    1    3  2    1  3    1    1    2
## 5  1017122  8 10   10    8  7   10  9    7    1    4
## 6  1018099  1  1    1    1  2   10  3    1    1    2
```

When we look at the above summary of all columns we can notice that the column X1.3 has a class called character and there is no min, max or mean value provided. And if we compare that column’s value with other columns’ value there is no difference all are digits. So the question is why that column data type is Character? There must be something wrong with it. Let’s focus on that particular column.. I just ran the column X1.3 for 30 rows and found “?” special character.

```
head((brc_data$X1.3),30)
```

```
##  [1] "10" "2"  "4"  "1"  "10" "10" "1"  "1"  "1"  "1"  "1"  "3"  "3"  "9"  "1"  "1"
"1"  "10" "1"  "10" "7"
## [22] "1"  "?"  "1"  "7"  "1"  "1"  "1"  "1"  "1"
```

We can also check other columns for missing values. Normally they are mentioned as NA in the table

```
sum(is.na(brc_data))
```

```
## [1] 0
```

There is no missing (NA) values in any of the columns..

```
c7missing <-  which(brc_data$X1.3 == "?")
length(c7missing)
```

```
## [1] 16
```

```
16*100/nrow(brc_data)
```

```
## [1] 2.292264
```

Missing data is only 2.2% of the total data, hence we can replace it with mean or mode.

```
c7 <- as.numeric((brc_data[-c7missing,"X1.3"]))
meanc7 <- round(mean(c7),2)
meanc7
```

```
## [1] 3.55
```

```
getmode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}
```

```
modec7 <- as.numeric(getmode(brc_data[-c7missing,"X1.3"]))
modec7
```

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

Let’s impute first mean for the missing data for column X1.3

```
mean_impute <- brc_data
mean_impute[c7missing,]$X1.3 <- meanc7
head(mean_impute$X1.3,30)
```

```
##  [1] "10"   "2"    "4"    "1"    "10"   "10"   "1"    "1"    "1"    "1"    "1"    "3"   
"3"    "9"    "1"   
## [16] "1"    "1"    "10"   "1"    "10"   "7"    "1"    "3.55" "1"    "7"    "1"    "1"   
"1"    "1"    "1"
```

As you can see above (roe #23), “?” is replaced with mean value of 3.55

Let’s impute mode value....

```
mode_impute <- brc_data
mode_impute[c7missing,]$X1.3 <- modec7
head(mode_impute$X1.3,30)
```

```
##  [1] "10" "2"  "4"  "1"  "10" "10" "1"  "1"  "1"  "1"  "1"  "3"  "3"  "9"  "1"  "1" 
"1"  "10" "1"  "10" "7" 
## [22] "1"  "1"  "1"  "7"  "1"  "1"  "1"  "1"  "1"
```

Here also row#23’s value is replaced with “1” .

2)

Use regression to impute values for the missing data.

```
brc_reg_data <- brc_data[-c7missing, 2:10]
brc_reg_data$X1.3 <- as.integer(brc_reg_data$X1.3)
```

Now, let's built the linear regression model..

```
model_lm <- lm(X1.3~.,data = brc_reg_data)
summary(model_lm)
```

```
##
## Call:
## lm(formula = X1.3 ~ ., data = brc_reg_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.7308 -0.9397 -0.3023  0.6739  8.6977
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.615998   0.195082  -3.158  0.00166 **
## X5           0.230916   0.041737   5.533 4.52e-08 ***
## X1          -0.068338   0.076213  -0.897  0.37022
## X1.1         0.339795   0.073468   4.625 4.49e-06 ***
## X1.2         0.339413   0.045946   7.387 4.45e-13 ***
## X2           0.090353   0.062574   1.444  0.14922
## X3           0.321324   0.059094   5.437 7.57e-08 ***
## X1.4         0.007045   0.044512   0.158  0.87429
## X1.5        -0.075191   0.059362  -1.267  0.20572
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.276 on 673 degrees of freedom
## Multiple R-squared:  0.6149, Adjusted R-squared:  0.6103
## F-statistic: 134.3 on 8 and 673 DF,  p-value: < 2.2e-16
```

Let's only use the variables that are important (with $P < 0.05$)...

```
new_lmmodel <- lm(X1.3~X5+X1.1+X1.2+X3, data = brc_reg_data)
summary(new_lmmodel)
```

```
##
## Call:
## lm(formula = X1.3 ~ X5 + X1.1 + X1.2 + X3, data = brc_reg_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.8105 -0.9436 -0.3130  0.6869  8.6870
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.53519    0.17524  -3.054  0.00235 **
## X5           0.22688    0.04125   5.500 5.40e-08 ***
## X1.1         0.31628    0.05092   6.212 9.15e-10 ***
## X1.2         0.33190    0.04434   7.485 2.23e-13 ***
## X3           0.32438    0.05611   5.782 1.13e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.275 on 677 degrees of freedom
## Multiple R-squared:  0.6128, Adjusted R-squared:  0.6105
## F-statistic: 267.9 on 4 and 677 DF,  p-value: < 2.2e-16
```

Now, let's predict the value for X1.3 column....

```
c7predict <- predict(new_lmmodel,newdata = brc_data[c7missing,])
c7predict <- round(c7predict,2)
c7predict
```

```
##   23   40  139  145  158  164  235  249  275  292  294  297  315  321  411  617
## 5.46 7.98 0.99 1.62 0.98 2.22 2.72 1.77 2.07 6.09 0.99 2.53 5.24 1.77 0.99 0.66
```

Let's imput the predicted data to X1.3

```
reg_impute <- brc_data
reg_impute[c7missing,]$X1.3 <- c7predict
reg_impute$X1.3 <- as.integer(as.numeric(reg_impute$X1.3))
head(reg_impute$X1.3,30)
```

```
##  [1] 10  2  4  1 10 10  1  1  1  1  1  3  3  9  1  1  1 10  1 10  7  1  5  1  7  1  1
1  1  1
```

The row #23's value has been replaced with "5". I have also converted the values to be integer as all other values in this columns are integer...

3)

Regression with Perturbation Imputation

```
ptb_model <- rnorm(nrow(brc_data[c7missing,]),c7predict, sd(c7predict))
ptb_impute <- brc_data
ptb_impute[c7missing,]$X1.3 <-ptb_model
ptb_impute$X1.3 <- as.integer(as.numeric(ptb_impute$X1.3))
ptb_model
```

```
##  [1] 10.1480148  8.4408252  2.9866378 -0.5981380  3.7797119  4.1267299  2.3000540  0.9
945822  1.5372035
## [10]  4.7150914  1.3197462  0.1934421  4.4202113 -0.3641186 -3.6187708 -0.8150633
```

Hmm, there are some negative values. We need make sure these random values are within the range of column's Min(1)and MAx(10) values.

```
ptb_impute$X1.3[ptb_impute$X1.3 > 10] <- 10
ptb_impute$X1.3[ptb_impute$X1.3 < 1] <- 1
```

4) Optional

Compare the results and quality of classification models (e.g., SVM, KNN) build using (1) the data sets from questions 1,2,3; (2) the data that remains after data points with missing values are removed; and (3) the data set when a binary variable is introduced to indicate missing values.

We are going build the KKN model with above created imputed data sets and compare the results.... Let's divide the data into training and test...

```
n <- nrow(brc_data)
train.rows <- sample(1:n,.70*n)
test.rows <- setdiff(1:nrow(brc_data), train.rows)
```

```
library(kknn)
k_accurcy <- rep(0,9)
k_accurcy
```

```
## [1] 0 0 0 0 0 0 0 0 0
```

Let's try first with mode impute data....

```
for (k in 1:3) {
  model_knn <- kknn(X2.1~., mode_impute[train.rows,], mode_impute[test.rows,],k=k)
  mode_pred <- as.integer(fitted(model_knn)+0.5)
  k_accuracy[k]= sum(mode_pred == mode_impute[test.rows,]$X2.1)/nrow(mode_impute[test.row
  s,])
}
```

Now let's try with mean impute data....

```
for (k in 1:3) {
  model_knn2 <- kknn(X2.1~., mean_impute[train.rows,], mean_impute[test.rows,],k=k)
  mean_pred <- as.integer(fitted(model_knn2)+0.5)
  k_accuracy[k+3]= sum(mean_pred == mean_impute[test.rows,]$X2.1)/nrow(mean_impute[test.r
  ows,])
}
```

Now let's try with regression impute data....

```
for (k in 1:3) {
  model_knn3 <- kknn(X2.1~., reg_impute[train.rows,], reg_impute[test.rows,],k=k)
  reg_pred <- as.integer(fitted(model_knn3)+0.5)
  k_accuracy[k+6]= sum(reg_pred == reg_impute[test.rows,]$X2.1)/nrow(reg_impute[test.row
  s,])
}
```

Now let's try with perturb impute data....

```
for (k in 1:3) {
  model_knn4 <- kknn(X2.1~., ptb_impute[train.rows,], ptb_impute[test.rows,],k=k)
  ptb_pred <- as.integer(fitted(model_knn4)+0.5)
  k_accuracy[k+9]= sum(ptb_pred == ptb_impute[test.rows,]$X2.1)/nrow(ptb_impute[test.row
  s,])
}
```

```
k_accuracy
```

```
##  [1] 0.9714286 0.9714286 0.9428571 0.9666667 0.9666667 0.9476190 0.9714286 0.9714286
0.9523810 0.9714286
## [11] 0.9714286 0.9380952
```

The first 1 to 3 results are of mode data set, next 4 to 6 are of mean data set, next 7 to 9 are of reg data set and last three are of perturb dataset. For K= 3, we can see that there is no significant difference in the results if we replace missing values with mean or mode. However there is very slight decline in the accuracy if we replace the missing value with regression...

We can do similar kind of analysis using SVM model as well.

Let's just try one more analysis by removing the rows with missing data....

```
new_brcdata <-brc_data[-c7missing,]
m <- nrow(new_brcdata)
new_trainrows <- sample(1:m,.70*m)
new_testrows <- setdiff(1:nrow(new_brcdata), new_trainrows)
```

```
k_acc <- rep(0,3)
for (k in 1:3) {
  model_knn5 <- kknn(X2.1~., new_brcdata[new_trainrows,], new_brcdata[new_testrows,],k=k)
  new_pred <- as.integer(fitted(model_knn5)+0.5)
  k_acc= sum(new_pred == new_brcdata[new_testrows,]$X2.1)/nrow(new_brcdata[new_testrow
  s,])
}
k_acc
```

```
## [1] 0.9414634
```

We remove the rows with missing values and tested the KNN model and it looks like removing the rows from the data is not making any big impact to the accuracy. Maybe because the missing data is only 2.2% of the total data.

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?

Let's say if a builder wants to optimize the use of his crew so he can finish fixing the roof of all the builings in a community. He has a crew of 10 people and each can work 5 hrs max at any day.

h = hours a person can work in a day n= numbe of crew members

Variables

x = number of roofs completed in one day

Contrains: h <= 5 n <= 10

Objective is to Max h*n which will max the X- the number of roofs completed