DEMO 10 - ACCOUNTING FOR MISSING DATA

ANSWER 1 - HANDLING MISSING DATA

1.0 - Prerequisite Check for Missing Data

The dataset given to us did not have predictor names but on this linke those attributes were available - http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29

Hence, after updating the

```
            ID Clump_Thickness Uniform_Cell_Size Uniform_Cell_Shape Marg_Adhesion
24  1057013               8                 4                  5             1
41  1096800               6                 6                  6             9
140 1183246               1                 1                  1             1
146 1184840               1                 1                  3             1
159 1193683               1                 1                  2             1
165 1197510               5                 1                  1             1
236 1241232               3                 1                  4             1
250  169356               3                 1                  1             1
276  432809               3                 1                  3             1
293  563649               8                 8                  8             1
295  606140               1                 1                  1             1
298   61634               5                 4                  3             1
316  704168               4                 6                  5             6
322  733639               3                 1                  1             1
412 1238464               1                 1                  1             1
618 1057067               1                 1                  1             1

    Single_Epith_Cell_Size Bare_Nuclei Bland_Chromatin Normal_Nucleoli Mitoses
24                       2          NA               7               3       1
41                       6          NA               7               8       1
140                      1          NA               2               1       1
146                      2          NA               2               1       1
159                      3          NA               1               1       1
165                      2          NA               3               1       1
236                      2          NA               3               1       1
250                      2          NA               3               1       1
276                      2          NA               2               1       1
293                      2          NA               6              10       1
295                      2          NA               2               1       1
298                      2          NA               2               3       1
316                      7          NA               4               9       1
322                      2          NA               3               1       1
412                      1          NA               2               1       1
618                      1          NA               1               1       1

    Class
24      1
41      0
140     0
146     0
159     0
165     0
236     0
250     0
276     0
293     1
295     0
298     0
316     0
322     0
412     0
618     0
```

Percentage of missing data: 2.289%

Please notice that 'Bare Nuclei' had all values assigned to NA which is below 5% threshold and justifies imputation.

In [ ]:
```
#CODE 1.0

require(ggthemes)
library(tidyverse)
library(magrittr)
library(TTR)
library(tidyr)
library(dplyr)
library(ggplot2)
library(plotly)
library(fpp2)
library(caTools)
library(reshape2)
library(psych)
require(graphics)
library(fBasics)
library(caret)
library(gridExtra)
library(DAAG)
library(rpart)
library(randomForest)
library(data.table)
library(mice)
library(MASS)
library(kknn)

# Read the file an name the columns
df<-read.table("breast-cancer-wisconsin.data.txt", stringsAsFactor = FALSE, header = F, sep = ",", na.strings="?'
colnames(df) <- c("ID", "Clump_Thickness", "Uniform_Cell_Size", "Uniform_Cell_Shape",
                  "Marg_Adhesion", "Single_Epith_Cell_Size", "Bare_Nuclei", "Bland_Chromatin",
                  "Normal_Nucleoli", "Mitoses", "Class")
df$Class <- as.factor(df$Class)
levels(df$Class) <- c(0, 1)

#which column contains the missing data
df[is.na(df$Bare_Nuclei),]
#check for % of missing observation (threshold < 5%)
print(sprintf("Percent of missing observation = %0.3f", 16/nrow(df)*100))
```

1.1 - Mean Imputation

The Bare Nuclei column is updated with mean values and first 24 values are displayed -

```
 1.000000 10.000000  2.000000  4.000000  1.000000 10.000000 10.000000  1.000000
 1.000000  1.000000  1.000000  1.000000  3.000000  3.000000  9.000000  1.000000
 1.000000  1.000000 10.000000  1.000000 10.000000  7.000000  1.000000  3.544656
```

To do a sanity check, we also take a look at the mean of the column -

Average Value: 3.544656

In [ ]:
```
#CODE - 1.1

#mean imputation
df.mean<-df
df.mean<-df.mean %>% mutate_at(vars(Bare_Nuclei),~ifelse(is.na(.x), mean(.x, na.rm = TRUE), .x))
#check it was imputed correctly
head(df.mean$Bare_Nuclei,24)

#check if mean was calculated correctly
mean(df.mean$Bare_Nuclei)
```

1.2 - Mode Imputation

The Bare Nuclei column is updated with mode values and first 24 values are displayed -

```
 1 10  2  4  1 10 10  1  1  1  1  1  3  3  9  1  1  1 10  1 10  7  1  1
```

It doesn't tell us very much but the mode value of this column is - 1.

In [ ]:
```
#CODE - 1.2

#found this mode function in the internet
getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}
df.mode<-df
mode.result <- getmode(df.mode$Bare_Nuclei)

#fill NA with Mode of 1s
df.mode$Bare_Nuclei[is.na(df.mode$Bare_Nuclei)] <- mode.result
#check it was imputed correctly
head(df.mode$Bare_Nuclei,24)
print(mode.result)
```

1.3 - Regression Imputation

1.3 - Regression Imputation

1 - First, we take a full model regression prediction of Bare Nuclei using all the other predictors -

```
Call:
lm(formula = Bare_Nuclei ~ Clump_Thickness + Uniform_Cell_Size + 
    Uniform_Cell_Shape + Marg_Adhesion + Single_Epith_Cell_Size + 
    Bland_Chromatin + Normal_Nucleoli + Mitoses, data = newdata.1)

Residuals:
    Min      1Q  Median      3Q     Max 
-9.7316 -0.9426 -0.3002  0.6725  8.6998 

Coefficients:
                        Estimate Std. Error t value Pr(>|t|)    
(Intercept)            -0.616652   0.194975  -3.163  0.00163 ** 
Clump_Thickness         0.230156   0.041691   5.521 4.83e-08 ***
Uniform_Cell_Size      -0.067980   0.076170  -0.892  0.37246    
Uniform_Cell_Shape      0.340442   0.073420   4.637 4.25e-06 ***
Marg_Adhesion           0.339705   0.045919   7.398 4.13e-13 ***
Single_Epith_Cell_Size  0.090392   0.062541   1.445  0.14883    
Bland_Chromatin         0.320577   0.059047   5.429 7.91e-08 ***
Normal_Nucleoli         0.007293   0.044486   0.164  0.86983    
Mitoses                -0.075230   0.059331  -1.268  0.20524    

Residual standard error: 2.274 on 674 degrees of freedom
Multiple R-squared:  0.615,  Adjusted R-squared:  0.6104 
F-statistic: 134.6 on 8 and 674 DF,  p-value: < 2.2e-16
```

2 - Then we perform stepwise variable selection to get -

```
Call:
lm(formula = Bare_Nuclei ~ Clump_Thickness + Uniform_Cell_Shape + 
    Marg_Adhesion + Bland_Chromatin, data = newdata.1)

Residuals:
    Min      1Q  Median      3Q     Max 
-9.8115 -0.9531 -0.3111  0.6678  8.6889 

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)    
(Intercept)        -0.53601    0.17514  -3.060   0.0023 ** 
Clump_Thickness     0.22617    0.04121   5.488 5.75e-08 ***
Uniform_Cell_Shape  0.31729    0.05086   6.239 7.76e-10 ***
Marg_Adhesion       0.33227    0.04431   7.499 2.03e-13 ***
Bland_Chromatin     0.32378    0.05606   5.775 1.17e-08 ***

Residual standard error: 2.274 on 678 degrees of freedom
Multiple R-squared:  0.6129, Adjusted R-squared:  0.6107 
F-statistic: 268.4 on 4 and 678 DF,  p-value: < 2.2e-16
```

3 - Due to insignificant predictors we used stepwise forwards and backwards mode to build a more accurate model for Bare Nuclei.

```
  nvmax     RMSE  Rsquared      MAE    RMSESD RsquaredSD     MAESD
1     1 2.548924 0.5228085 1.765516 0.2208983 0.08742394 0.1577356
2     2 2.432898 0.5624336 1.642851 0.2839852 0.10115619 0.1857652
3     3 2.362419 0.5900746 1.565549 0.2804584 0.09463731 0.1827845
4     4 2.278984 0.6185545 1.534310 0.2734543 0.08969728 0.1880005
```

Best model - nvmax : 4

After cross-validation, we infer that the model with these predictors -

1. Clump_Thickness
2. Uniform_Cell_Shape
3. Marg_Adhesion
4. Bland_Chromatin

have the **lowest RMSE value of 2.278 and highest R-Square of 0.618**!

4 - After applying the new regression model, the Bare Nuclei column is updated with regressed values and first 24 values are displayed -

```
1 10 2 4 1 10 10 1 1 1 1 1 3 3 9 1 1 1 10 1 10 7 1 5
```

In [ ]:
```
#CODE - 1.3

set.seed(123)

newdata<-df
missing.index<-which(is.na(newdata$Bare_Nuclei), arr.ind=TRUE)
newdata.1 <- newdata[-missing.index,2:10]
#all other predictors data points except for the missing value and response variable

#Linear Model
model <- lm(Bare_Nuclei~Clump_Thickness+
            Uniform_Cell_Size+
            Uniform_Cell_Shape+
            Marg_Adhesion+
            Single_Epith_Cell_Size+
            Bland_Chromatin+
            Normal_Nucleoli+
            Mitoses,data=newdata.1 )
summary(model)

# Fit the full model
full.model <- model
# Stepwise regression model
step.model <- stepAIC(full.model, direction = "both", trace = FALSE)
summary(step.model)

# Set seed for reproducibility
set.seed(123)
# Set up repeated k-fold cross-validation
train.control <- trainControl(method = "cv", number = 10)
# Train the model
step.model <- train(Bare_Nuclei ~., data = newdata.1 ,
                    method = "leapBackward",
                    tuneGrid = data.frame(nvmax = 1:4),
                    trControl = train.control
                    )
step.model$results
predicted.missing<-predict(step.model,newdata=df[missing.index,])
df.final.regression<-df
df.final.regression[missing.index,]$Bare_Nuclei<-as.integer(predicted.missing)#make predicted values integers
#final data with imputed regressed values
head(df.final.regression$Bare_Nuclei,24)
```

1.4 - Regression and Perturbation Imputation

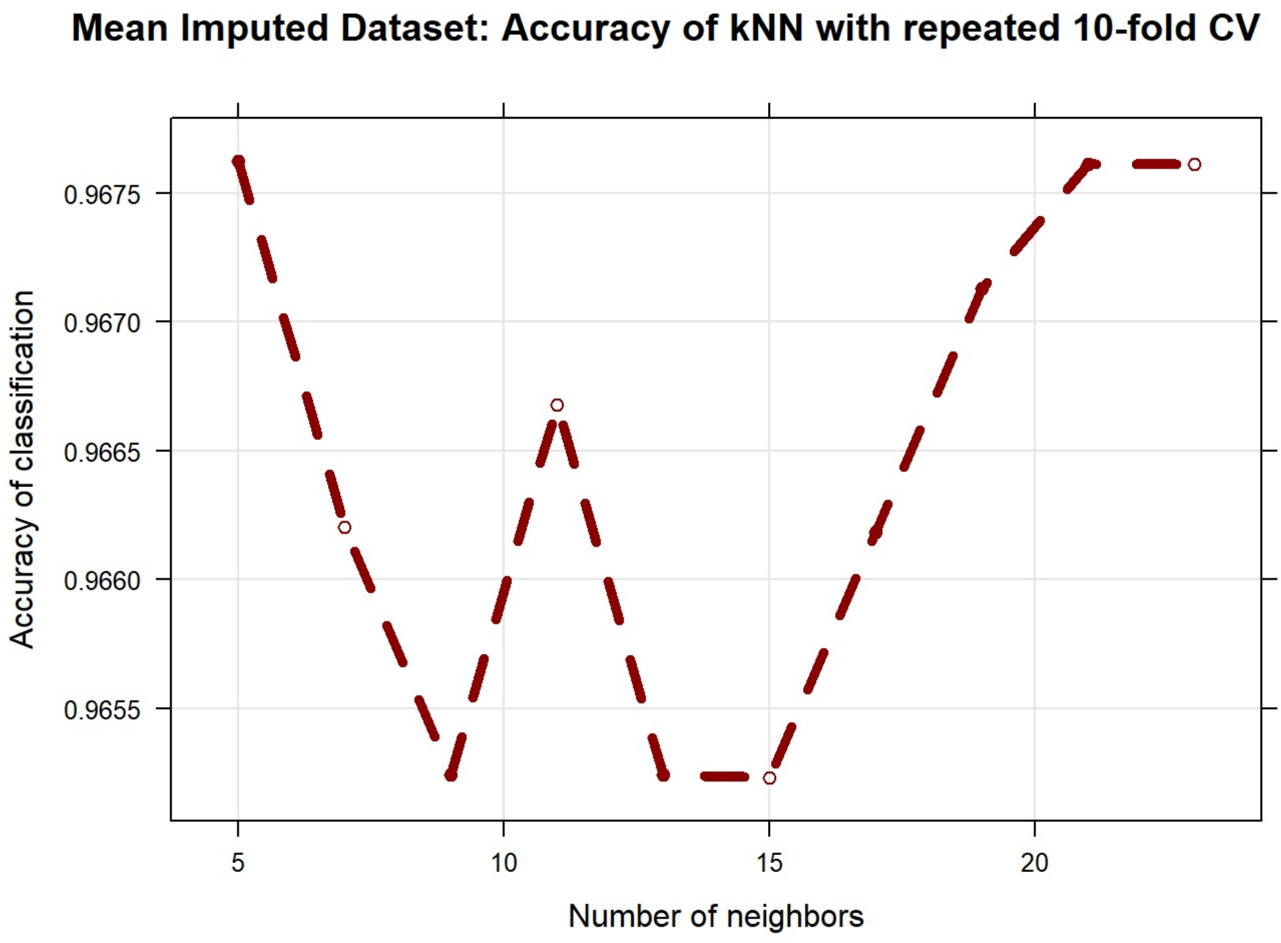
1 - This warrants for the regression using random values for missing data, and this is final Bare Nuclei values plugged in -

```
1 10 2 4 1 10 10 1 1 1 1 1 3 3 9 1 1 1 10 1 10 7 1 5
```

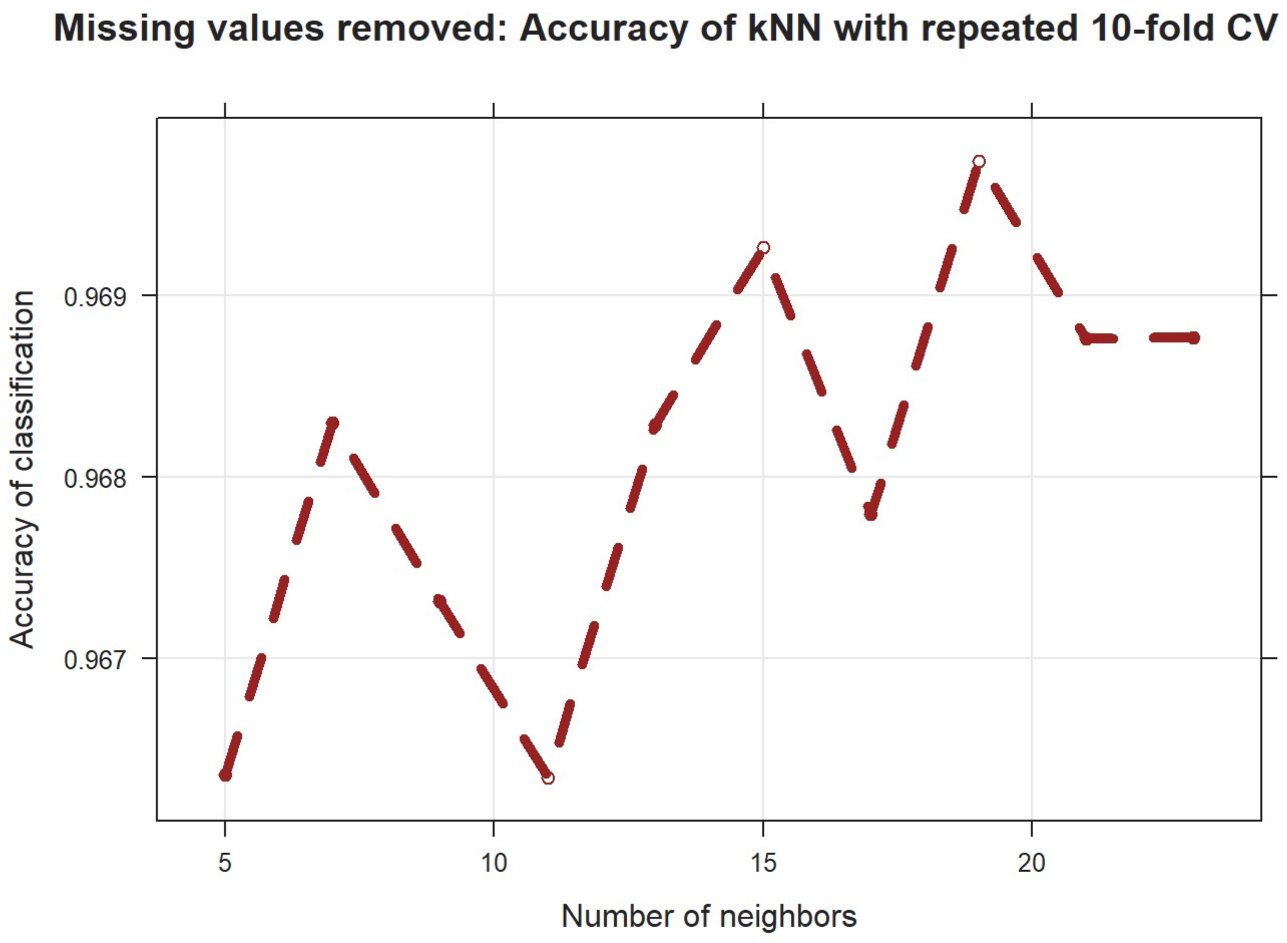
In [ ]:
```
#CODE - 1.4

set.seed(123)
#generate 16 random numbers based off missing predicted values
n <- rnorm(16, mean = predicted.missing, sd = sd(predicted.missing))
#bounding the negative numbers to positive only
abs(n)
df.final.regression.pertubed<-df
df.final.regression.pertubed[missing.index,]$Bare_Nuclei<-as.integer(abs(n))#make predicted values integers
#final data with imputed  perturbed regressed values
head(df.final.regression.pertubed$Bare_Nuclei,24)
```

1.5 - Comparisons

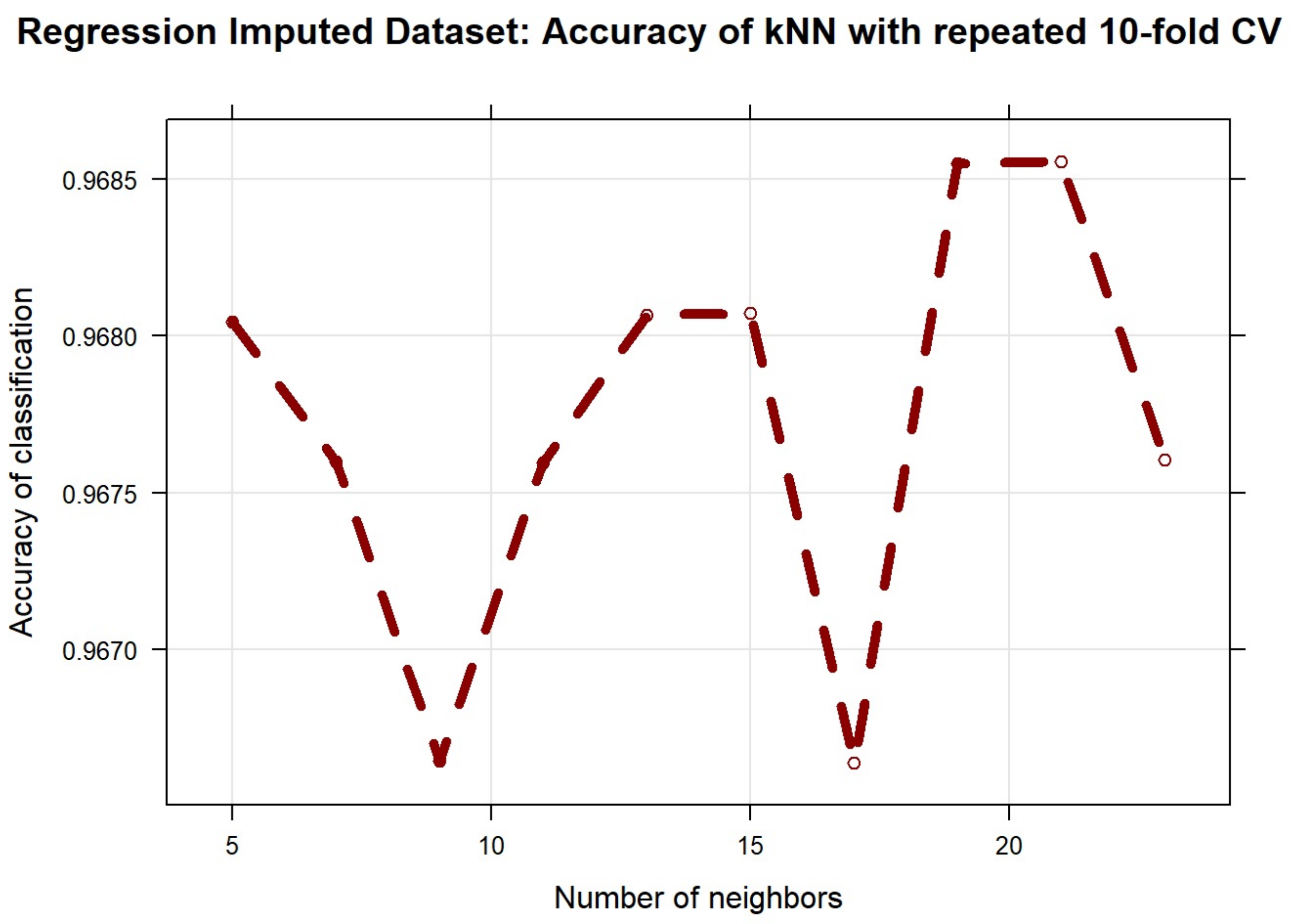
1 - KNN versus Mean Imputation



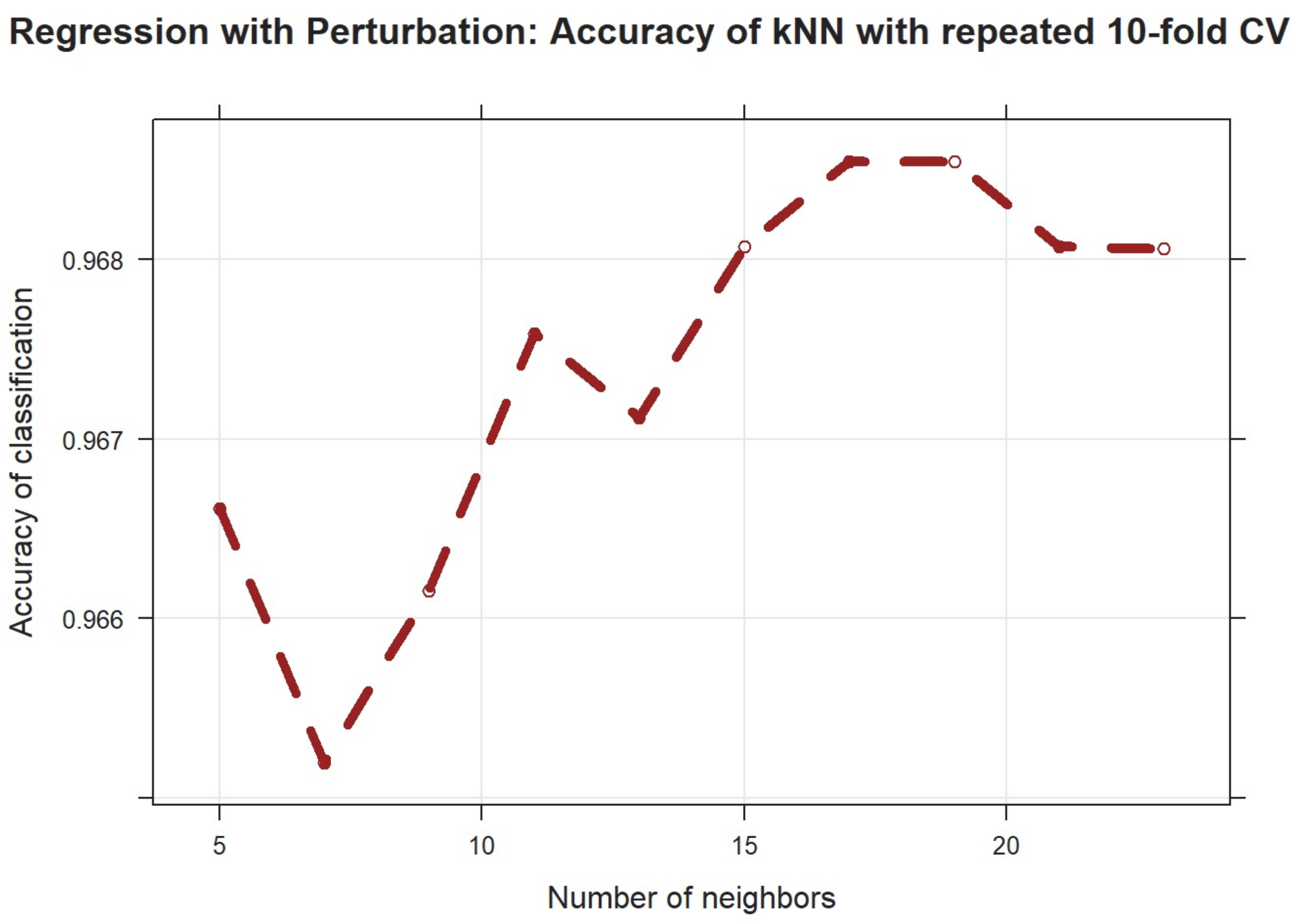
2 - KNN versus Removed Missing Points



3 - KNN versus Regular Regression



4 - KNN versus Perturbed Regression



5 - Conclusion

With 10-fold CVV KNN accuracy was quite high - over 90% - but with differing k values across the techniques -

1. Mean approach K: 5

2. Missing approach K: 19

3. Perturbed regression approach K: 19

4. Regular regression approach K: 20

In [ ]:
```
#CODE - 1.5

df1<-read.table("breast-cancer-wisconsin.data.txt",
                stringsAsFactor = FALSE, header = F, sep = ",", na.strings="?")
ctrl <- trainControl(method="repeatedcv",number=10,repeats = 3)
knn.mean <- train(x=df.mean[,1:10],y=as.factor(df.mean[,11]), method = "knn", trControl = ctrl,
                preProcess = c("center","scale"), tuneLength = 10)
#Plot 1
plot(knn.mean,col = "dark red",lwd=5,lty=2,cex.lab=1.25,cex.main=1.5,
     main="Mean Imputed Dataset: Accuracy of kNN with repeated 10-fold CV",
     xlab="Number of neighbors",
     ylab="Accuracy of classification")

knn.regressed <- train(x=df.final.regression[,1:10],y=as.factor(df.final.regression[,11]), method = "knn",
                       trControl = ctrl,
                       preProcess = c("center","scale"), tuneLength = 10)
#plot 2
plot(knn.regressed,col = "dark red",lwd=5,lty=2,cex.lab=1.25,cex.main=1.5,
     main="Regression Imputed Dataset: Accuracy of kNN with repeated 10-fold CV",
     xlab="Number of neighbors",
     ylab="Accuracy of classification")

knn.perturbreg <- train(x=df.final.regression.pertubed[,1:10],
                        y=as.factor(df.final.regression.pertubed[,11]), method = "knn", trControl = ctrl,
                        preProcess = c("center","scale"), tuneLength = 10)
#Plot 3
plot(x=knn.perturbreg,col = "dark red",lwd=5,lty=2,cex.lab=1.25,cex.main=1.5,
     main="Regression with Perturbation: Accuracy of kNN with repeated 10-fold CV",
     xlab="Number of neighbors",
     ylab="Accuracy of classification")

df1.removed<-na.omit(df1)
knn.removed <- train(x=df1.removed[,1:10],y=as.factor(df1.removed[,11]), method = "knn", trControl = ctrl,
                        preProcess = c("center","scale"), tuneLength = 10)
#Plot 4
plot(knn.removed,col = "dark red",lwd=5,lty=2,cex.lab=1.25,cex.main=1.5,
     main="Missing values removed: Accuracy of kNN with repeated 10-fold CV",
     xlab="Number of neighbors",
     ylab="Accuracy of classification")
```

ANSWER 2 - REAL-LIFE OPTIMIZATION USE-CASE

On a past operational strategy endeavour at a non-profit in natural catastrophe management, a system of UAV vehicles needed to be set up along with their stationary charging banks. Then upon an incoming request for emergency delivery, a UAV would need to be selected and assigned to pickup commodity from source to destination and land at nearest chraging bank to recharge to full fuel. The following describes the system's optimized work plan -

Decision Variables for drone 'n' upon service request 'r' -

1. Total Distance Covered = Distance from drone 'n' current location to commodity source location + Distance from commodity source location to delivery location + Distance between delivery location and nearest power charge + Distance covered in idle time to get back to original location + Deviation distance in between due to recharge

2. Fuel usage for drone 'n' for service request 'r' based on Total Distance Covered

3. Emergency level in measure of time of service request 'r'.

This is calculated across the entire system of drones and the constraints are as follows -

1. No service waits longer than time 't' if a drone is available.

2. Total fuel usage across 'k' drones having overlapping work schedules within period 'p' does not exceed f(k,p).

3. Recharges are kept to a minimum of 'm' recharges across f(k,p).

And for the objective function (IN THE FOLLOWING ORDER OF IMPORTANCE):

1. Minimize average delivery time.

2. Minimize total fuel usage.

3. Minimize recharge trips.

THE END---