Dealing with Missing Data

Using the breast cancer dataset from UCI, compute the missing values using mean, mode, regression, and regression with perturbation. After the missing values have been imputed, ran a KNN model on the different datasets and compare the qualuty of the classfication.

Packages & Data

```
library(tidyverse)
library(mice)
library(caret)

rm(list = ls())

set.seed(123)

#setwd("/Users/Ryan/Desktop/DS")

bcw = read.table("breast-cancer-wisconsin.txt.",
                         sep=",",
                         fill=FALSE,
                         strip.white=TRUE,
                         header = FALSE,
                         stringsAsFactors = FALSE)

bcw2 = read.csv("breast-cancer-wisconsin.txt.",
                header = FALSE,
                na.strings = '?')
```

I loaded in the data two different ways: one treating it as a text file and the other as a csv file. I originally loaded in the data as a text file because that is how the file is saved as. However, I ended up wanting the data values as integers and it is easier to declare the missing values.

Explore the Data

```
head(bcw)
```

```
##        V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11
## 1 1000025  5  1  1  1  2  1  3  1   1   2
## 2 1002945  5  4  4  5  7 10  3  2   1   2
## 3 1015425  3  1  1  1  2  2  3  1   1   2
## 4 1016277  6  8  8  1  3  4  3  7   1   2
## 5 1017023  4  1  1  3  2  1  3  1   1   2
## 6 1017122  8 10 10  8  7 10  9  7   1   4
```

```
summary(bcw)
```

```
##        V1                  V2               V3              V4
##  Min.   :   61634   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.:  870688   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000
##  Median : 1171710   Median : 4.000   Median : 1.000   Median : 1.000
##  Mean   : 1071704   Mean   : 4.418   Mean   : 3.134   Mean   : 3.207
##  3rd Qu.: 1238298   3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000
##  Max.   :13454352   Max.   :10.000   Max.   :10.000   Max.   :10.000
##        V5               V6              V7                V8
##  Min.   : 1.000   Min.   : 1.000   Length:699         Min.   : 1.000
##  1st Qu.: 1.000   1st Qu.: 2.000   Class :character   1st Qu.: 2.000
##  Median : 1.000   Median : 2.000   Mode  :character   Median : 3.000
##  Mean   : 2.807   Mean   : 3.216                      Mean   : 3.438
##  3rd Qu.: 4.000   3rd Qu.: 4.000                      3rd Qu.: 5.000
##  Max.   :10.000   Max.   :10.000                      Max.   :10.000
##        V9              V10              V11
##  Min.   : 1.000   Min.   : 1.000   Min.   :2.00
##  1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.:2.00
##  Median : 1.000   Median : 1.000   Median :2.00
##  Mean   : 2.867   Mean   : 1.589   Mean   :2.69
##  3rd Qu.: 4.000   3rd Qu.: 1.000   3rd Qu.:4.00
##  Max.   :10.000   Max.   :10.000   Max.   :4.00
```

```
#index's of the missing data
impute_me <- which(bcw$V7 == "?")
impute_me
```

```
##  [1]  24  41 140 146 159 165 236 250 276 293 295 298 316 322 412 618
```

```
#How much are we missing?
length(impute_me)/length(bcw$V7) # ~2%
```

```
## [1] 0.02288984
```

```
#Check out V11
bcw_clean <- bcw[-impute_me,]
bcw_missing <- bcw[impute_me,]
table(bcw$V11)
```

```
##
##   2   4
## 458 241
```

```
table(bcw_clean$V11)
```

```
##
##   2   4
## 444 239
```

```r
table(bcw_missing$V11) #proportion looks extreme
```

```
##
##  2  4
## 14  2
```

```r
sum(bcw$V11 == 2)/nrow(bcw)
```

```
## [1] 0.6552217
```

```r
sum(bcw_clean$V11 == 2)/nrow(bcw_clean)
```

```
## [1] 0.6500732
```

```r
sum(bcw_missing$V11 == 2)/nrow(bcw_missing)
```

```
## [1] 0.875
```

```r
#missing data is biased

#Data when loaded as csv and missing values converted to NA
table(bcw2$V7)
```

```
##
##   1   2   3   4   5   6   7   8   9  10
## 402  30  28  19  30   4   8  21   9 132
```

```r
sum(is.na(bcw2$V7))
```

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

I started off by looking at the data and exploring what we are working with. From the summary we see that values are missing in the column V7. Next, I computed the proportion of missing values in that column which came out to about 2%. This is pretty low amount so we should be okay imputing the missing values. After that I created two different datasets: one with no missing data points and the other solely consisting of missing values. In the missing values dataset, we see that there is a bias in the data so dont use that for imputation. The clean dataset has a consistant proportion of values compared to the original.

Mean Method

```r
avg1 <- mean(bcw_clean$V7)
```

```
## Warning in mean.default(bcw_clean$V7): argument is not numeric or logical:
## returning NA
```

```r
avg <- mean(bcw2$V7, na.rm = TRUE) #compute average excluding missing values
avg1 == avg
```

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

```r
bcw_mean_replace <- bcw2 %>%
  mutate(V7  = ifelse(is.na(V7), avg, V7))

table(bcw_mean_replace$V7)
```

```
##
##                 1                 2                 3 3.54465592972182
##               402                30                28                16
##                 4                 5                 6                 7
##                19                30                 4                 8
##                 8                 9                10
##                21                 9               132
```

I took the mean of column V7 excluding the missing values, which came out to be around 3.54. Then I mutated the column V7 with the mean if that value is missing. If that column already has a value, then it is skipped over until there is a missing value cell. I printed out the table so you could see the new distribution of values in the column.

Mode Method

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

mdv7 <- the_mode(bcw2[-impute_me,"V7"])
mdv7
```

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

```r
bcw_mode <- bcw2
bcw_mode[impute_me,]$V7 <- mdv7

table(bcw_mode$V7)
```

```
##
##   1   2   3   4   5   6   7   8   9  10
## 418  30  28  19  30   4   8  21   9 132
```

Mode was a little harder since there is no built in function but there are a lot of examples on Stack Overflow. You could also just look at the table of the column V7 and choose it that way. We replaced the values at the index's in impute_me with the mode value, which happens to be 1. The table shows 418 which is 16 higher than the original count so we have done it correctly.

Imputation using regression

```
impute <- mice(bcw2, method = 'norm.predict')
```

```
##
##  iter imp variable
##   1   1  V7
##   1   2  V7
##   1   3  V7
##   1   4  V7
##   1   5  V7
##   2   1  V7
##   2   2  V7
##   2   3  V7
##   2   4  V7
##   2   5  V7
##   3   1  V7
##   3   2  V7
##   3   3  V7
##   3   4  V7
##   3   5  V7
##   4   1  V7
##   4   2  V7
##   4   3  V7
##   4   4  V7
##   4   5  V7
##   5   1  V7
##   5   2  V7
##   5   3  V7
##   5   4  V7
##   5   5  V7
```

```
#norm.predict is linear regression, predicted values
bcw_regression <- complete(impute)

table(bcw_regression$V7)
```

```
##
##                1 1.05937027181892 1.17980589341214 1.18620544316557
##              402                1                1                1
## 1.18895098884158 1.26045314611238 1.30325306504751 1.42879706989917
##                1                1                1                1
## 1.46911881469832  1.5625742438013 1.57993610120741 1.74098962937797
##                1                1                1                1
## 1.94384154096917                2 2.08333403420357                3
##                1               30                1               28
## 3.41920848887475                4                5                6
##                1               19               30                4
## 6.43288386965875                7 7.19123667219868                8
##                1                8                1               21
##                9               10
##                9              132
```

```
sum(is.na(bcw_regression$V7))
```

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

You can do this the long way by running a lm model and predicting the missing values, which I did previously. But when I was researching, I found a function that does the linear regression for you and it's made for imputing missing values. All you have to do is specify the data and which method to use. There is a bunch of different methods but norm.predict does linear regression and prediction. As you can see from the table all the values aren't the same as the previous methods. It has a lot more range which will see later if that helps us.

Imputation using Regression with perturbation

```
impute2 <- mice(bcw2, method = 'norm.nob')
```

```
##
##  iter imp variable
##   1   1  V7
##   1   2  V7
##   1   3  V7
##   1   4  V7
##   1   5  V7
##   2   1  V7
##   2   2  V7
##   2   3  V7
##   2   4  V7
##   2   5  V7
##   3   1  V7
##   3   2  V7
##   3   3  V7
##   3   4  V7
##   3   5  V7
##   4   1  V7
##   4   2  V7
##   4   3  V7
##   4   4  V7
##   4   5  V7
##   5   1  V7
##   5   2  V7
##   5   3  V7
##   5   4  V7
##   5   5  V7
```

```
bcw_regression_perturb <- complete(impute2)

table(bcw_regression_perturb$V7)
```

```
##
## -3.89420142575849 -1.00624199699465 0.489109562504009 0.592388919972116
```

```
##                 1                 1                 1                 1
## 0.687579585236096                 1                 2  2.05971748441523
##                 1               402                30                 1
##  2.20232321005571                 3  3.01060278096757  3.04756076969181
##                 1                28                 1                 1
##  3.09520036601409  3.12056621824417  3.47991555427408                 4
##                 1                 1                 1                19
##  4.16487678429881                 5  5.51291965625589  5.84244038479296
##                 1                30                 1                 1
##                 6                 7                 8                 9
##                 4                 8                21                 9
##                10  10.0526611053766
##               132                 1
```

```
sum(is.na(bcw_regression_perturb$V7))
```

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

This code is very similar to the previous one but this time we choose a different method. If you look at the list of methods norm.nob using linear regression with perturbation.

Quality of Classification Models:

```
# Dataset by removing the missing values
bcw_removed <- na.omit(bcw2)
table(bcw_removed$V7)
```

```
##
##   1   2   3   4   5   6   7   8   9  10
## 402  30  28  19  30   4   8  21   9 132
```

```
length(bcw_removed$V7)
```

```
## [1] 683
```

One of the optional questions said to run a classification model on a dataset with the missing values removed. One nice thing about using the bcw2 dataset is that when it was imported it set the values "?" to NA so we can use functions that relate to NA. Thus, we just omit the rows that have values of NA.

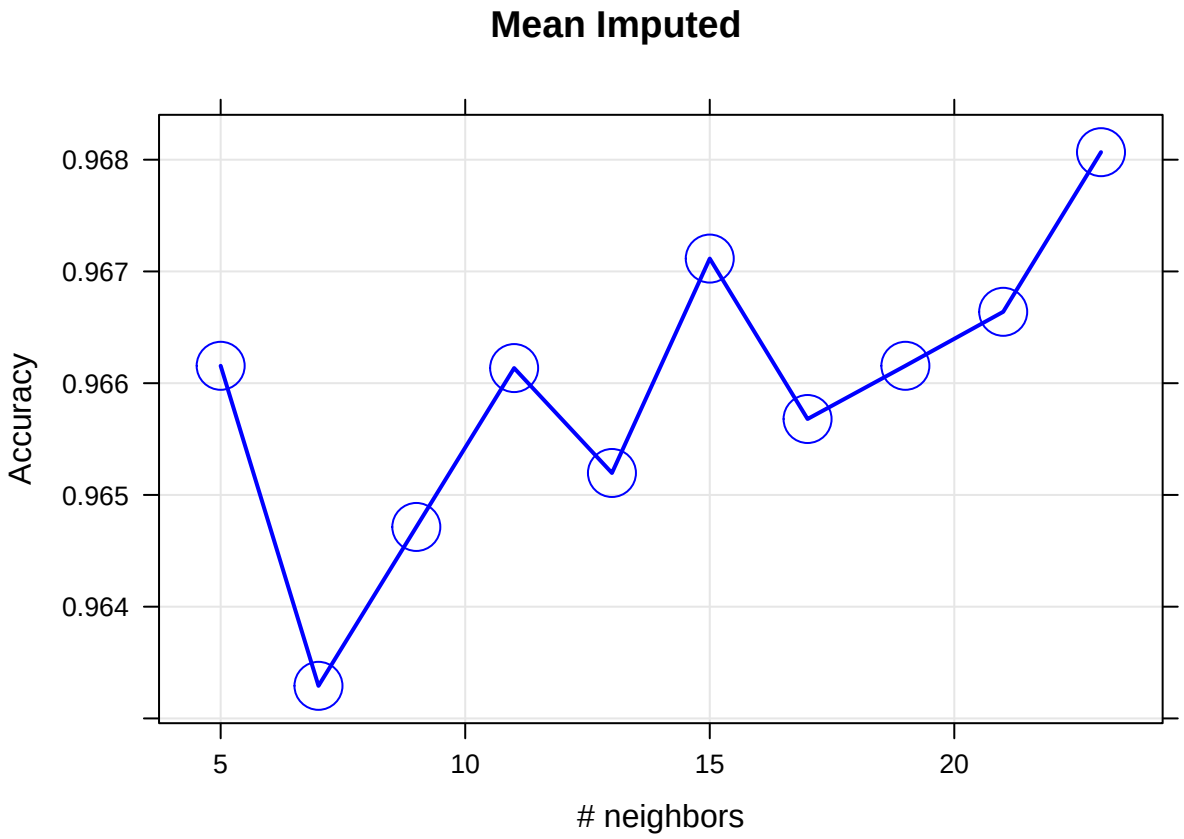
Train with Knn

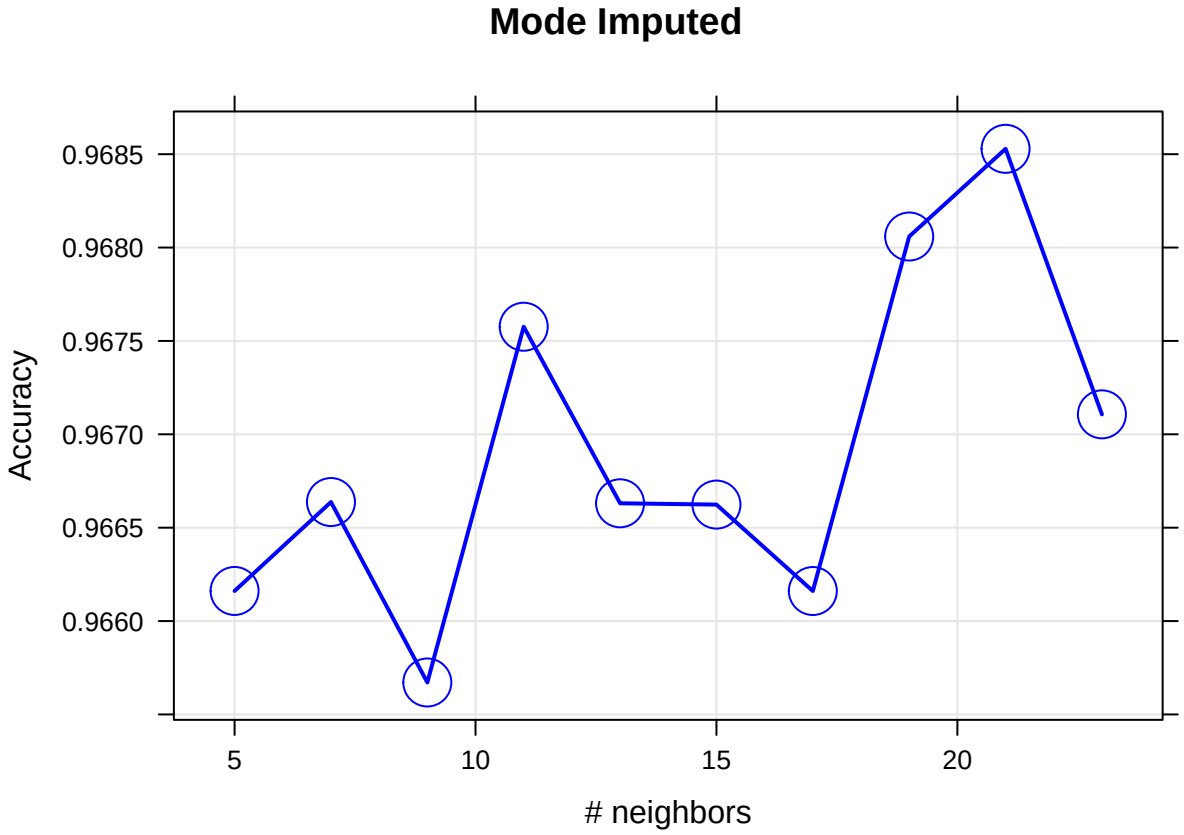
```
ctrl <- trainControl(method="repeatedcv", #cross validation
                     number=10,
                     repeats = 3) # repeats the process 3 times

#knn with missing values computed with mean
knn.mean <- train(bcw_mean_replace[,1:10],
```

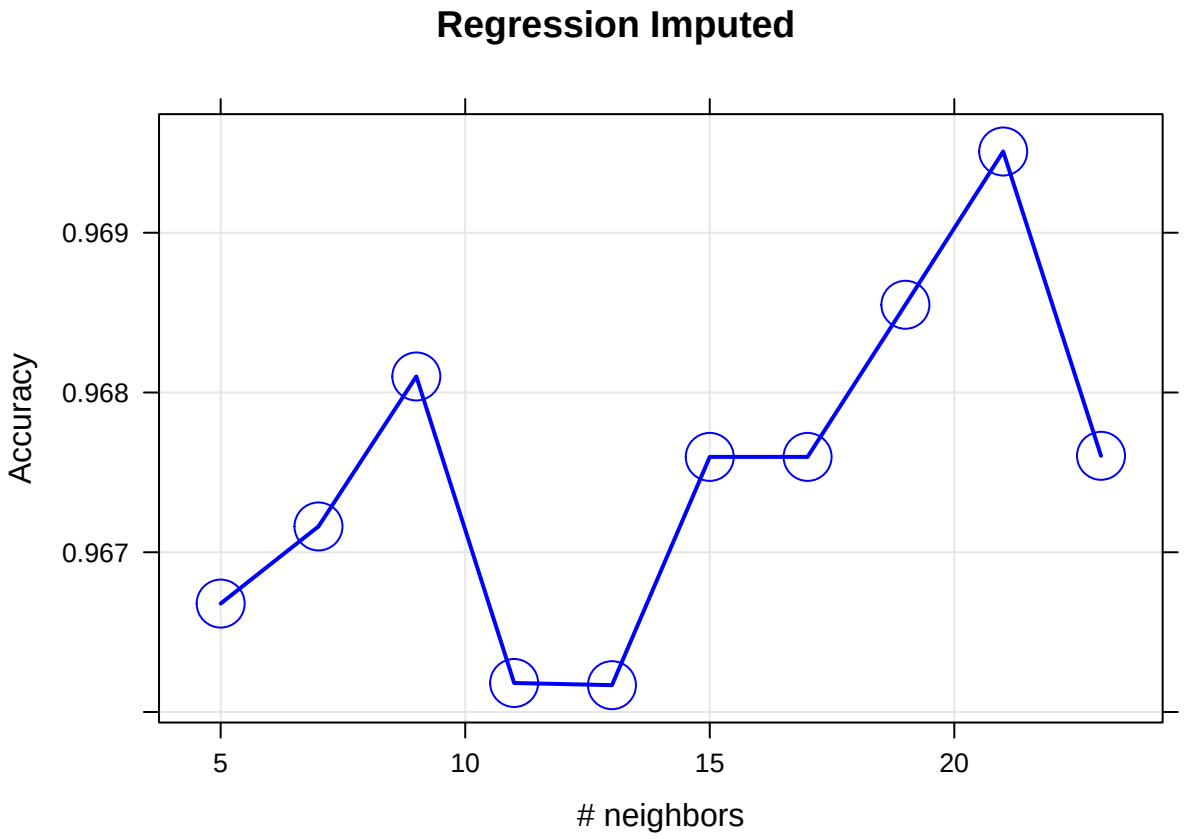
```
                  as.factor(bcw_mean_replace[,11]),
                  method = "knn",
                  trControl = ctrl,
                  preProcess = c("center","scale"),
                  tuneLength = 10)

#knn with missing values computed with mode
knn.mode <- train(bcw_mode[,1:10],
                  as.factor(bcw_mode[,11]),
                  method = "knn",
                  trControl = ctrl,
                  preProcess = c("center","scale"),
                  tuneLength = 10)

#knn with missing values computed with regression
knn.reg <- train(bcw_regression[,1:10],
                  as.factor(bcw_regression[,11]),
                  method = "knn",
                  trControl = ctrl,
                  preProcess = c("center","scale"),
                  tuneLength = 10)

#knn with missing values computed using regression with perturbation
knn.reg.pert <- train(bcw_regression_perturb[,1:10],
                      as.factor(bcw_regression_perturb[,11]),
                      method = "knn",
                      trControl = ctrl,
                      preProcess = c("center","scale"),
                      tuneLength = 10)

#knn with missing values removed
knn.remove <- train(bcw_removed[,1:10],
                    as.factor(bcw_removed[,11]),
                    method = "knn",
                    trControl = ctrl,
                    preProcess = c("center","scale"),
                    tuneLength = 10)
```

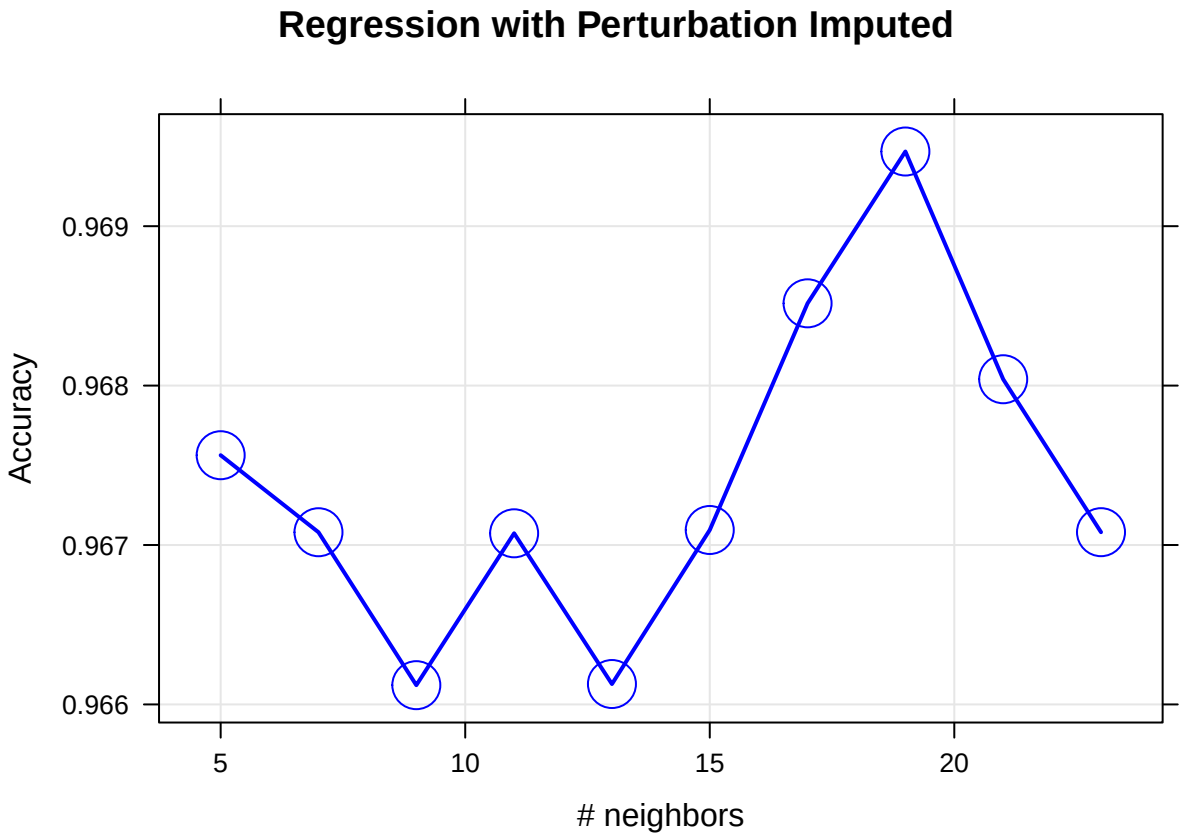
The ctrl part is what the train function uses to cross validate instead of having to run two different functions. We are trying to predict the column V11, so we convert it into a factor since there are only two possibilities. We want to center and scale the function so we can specifiy it in the function. The only thing that is different between the models is the input data, but all the tuning parameters are the same.

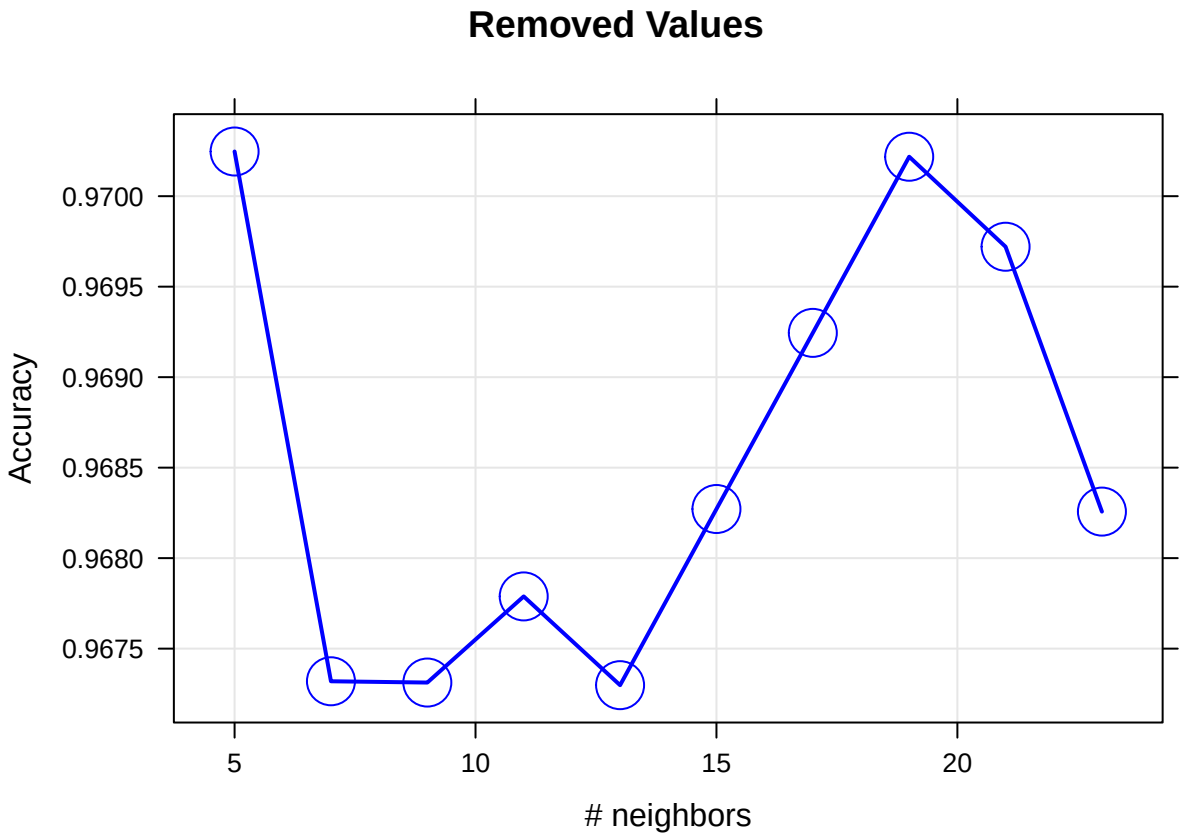
Plot Values


Mean Imputed
Accuracy
0.968
0.967
0.966
0.965
0.964
5
10
15
20
neighbors


Mode Imputed
Accuracy
0.9685
0.9680
0.9675
0.9670
0.9665
0.9660
5
10
15
20
neighbors


Regression Imputed
Accuracy
0.969
0.968
0.967
5
10
15
20
neighbors


Regression with Perturbation Imputed
Accuracy
0.969
0.968
0.967
0.966
5
10
15
20
neighbors



I plotted all the models so that you can see the accuracy of the model compared to the number of neighbors. The removed value had the best prediction compared to the rest which means, or imputed values might not have been the best prediction. The amount of neighbors seems to be the most accurate above 17.

Best k value & Errors

```
knn.mean$bestTune
```

```
##     k
## 10 23
```

```
knn.reg$bestTune
```

```
##    k
## 9 21
```

```
knn.reg.pert$bestTune
```

```
##    k
## 8 19
```

```
knn.remove$bestTune
```

```
##   k
## 1 5
```

```
100*(1-knn.mean$results$Accuracy)
```

```
##  [1] 3.384442 3.670827 3.528660 3.386473 3.480351 3.288494 3.432061 3.384442
##  [9] 3.336133 3.193276
```

```
100*(1-knn.mode$results$Accuracy)
```

```
##  [1] 3.383888 3.336269 3.432887 3.242392 3.336940 3.337630 3.383907 3.194082
##  [9] 3.147134 3.289320
```

```
100*(1-knn.reg$results$Accuracy)
```

```
##  [1] 3.332128 3.283858 3.189980 3.381837 3.383198 3.240321 3.240321 3.145083
##  [9] 3.049155 3.239631
```

```
100*(1-knn.reg.pert$results$Accuracy)
```

```
##  [1] 3.243675 3.291984 3.387912 3.292654 3.387202 3.290584 3.148398 3.053160
##  [9] 3.196017 3.291945
```

```
100*(1-knn.remove$results$Accuracy)
```

```
##  [1] 2.975319 3.268037 3.268748 3.221149 3.270190 3.172861 3.075532 2.978204
##  [9] 3.027934 3.174282
```

To no surprise the models have different number of neighbors that maximize the accuracy of the model. The last things that are printed out are the errors in the models. You noticed that there are multiple error values and that is because we cross validated the models. None of the error values was over 4% so we are pretty accurate with the imputed values. Though we did not split our data, so we are definitely overfitting with the models.