Homework 10 - Advanced Data Prep & Optimizations

Omer Farooq (EDx ID: mfarooq4)

03/24/2020

Table of Contents

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/datasets/Breast+Cancer+Wisconsin+%28Original%29) has missing values.

- **1. Use the mean/mode imputation method to impute values for the missing data.**
- **2. Use regression to impute values for the missing data.**
- **3. Use regression with perturbation to impute values for the missing data.**
- **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**

First of all, I pulled the data in.

```r
#setting the seed so that results are the same at every run
set.seed(101)

#loading data
cancer_data <- read.table("data_14.1/breast-cancer-wisconsin.data.txt",
stringsAsFactors = FALSE, header = FALSE, sep = ",")


#quick glance at the data
head(cancer_data)
```

```
##        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
```

I did some basic checks to see where missing data was and how much data is missing

```
#checking how many values are missing and where
colSums(cancer_data == '?')

##  V1  V2  V3  V4  V5  V6  V7  V8  V9 V10 V11
##   0   0   0   0   0   0  16   0   0   0   0

#showing the actual missing value rows
cancer_data[cancer_data$V7=='?',]

##          V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11
## 24  1057013  8  4  5  1  2  ?  7  3   1   4
## 41  1096800  6  6  6  9  6  ?  7  8   1   2
## 140 1183246  1  1  1  1  1  ?  2  1   1   2
## 146 1184840  1  1  3  1  2  ?  2  1   1   2
## 159 1193683  1  1  2  1  3  ?  1  1   1   2
## 165 1197510  5  1  1  1  2  ?  3  1   1   2
## 236 1241232  3  1  4  1  2  ?  3  1   1   2
## 250  169356  3  1  1  1  2  ?  3  1   1   2
## 276  432809  3  1  3  1  2  ?  2  1   1   2
## 293  563649  8  8  8  1  2  ?  6 10   1   4
## 295  606140  1  1  1  1  2  ?  2  1   1   2
## 298   61634  5  4  3  1  2  ?  2  3   1   2
## 316  704168  4  6  5  6  7  ?  4  9   1   2
## 322  733639  3  1  1  1  2  ?  3  1   1   2
## 412 1238464  1  1  1  1  1  ?  2  1   1   2
## 618 1057067  1  1  1  1  1  ?  1  1   1   2

#percentage of data with missinf values
100*nrow(cancer_data[cancer_data$V7=='?',])/nrow(cancer_data)

## [1] 2.288984
```

Above analysis shows that column V7 has 16 missing values (indicated by ?) and they make up 2.28% of the total data (which is less than 5%) making it Ok to apply missing data techniques.

##Imputation using Mode and Mean

The mode of v7 column came out to be 1 and mean was 3.5.I replaced the missing values with these. It was interesting that these two very different values replaced the missing values.

```
#storing indices of missing values
indices <- which(cancer_data$V7 == '?', arr.ind = T)
indices

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

#figure out the range of rest of the V7 data
max(cancer_data[-indices,]$V7)

## [1] "9"

min(cancer_data[-indices,]$V7)

## [1] "1"

#calcuting mode of v7 column

v7 <- cancer_data[-indices,'V7']
uniq_v7<- unique(v7)
mode <- as.numeric(uniq_v7[which.max(tabulate(match(v7, uniq_v7)))])
mode

## [1] 1

#imputation using mode
data_mode <- cancer_data
data_mode[indices, 'V7'] <- mode

#check
colSums(data_mode == '?')

##  V1  V2  V3  V4  V5  V6  V7  V8  V9 V10 V11
##   0   0   0   0   0   0   0   0   0   0   0

data_mode[indices,]

##          V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11
## 24  1057013  8  4  5  1  2  1  7  3   1   4
## 41  1096800  6  6  6  9  6  1  7  8   1   2
## 140 1183246  1  1  1  1  1  1  2  1   1   2
## 146 1184840  1  1  3  1  2  1  2  1   1   2
## 159 1193683  1  1  2  1  3  1  1  1   1   2
## 165 1197510  5  1  1  1  2  1  3  1   1   2
## 236 1241232  3  1  4  1  2  1  3  1   1   2
## 250  169356  3  1  1  1  2  1  3  1   1   2
## 276  432809  3  1  3  1  2  1  2  1   1   2
## 293  563649  8  8  8  1  2  1  6 10   1   4
## 295  606140  1  1  1  1  2  1  2  1   1   2
## 298   61634  5  4  3  1  2  1  2  3   1   2
## 316  704168  4  6  5  6  7  1  4  9   1   2
## 322  733639  3  1  1  1  2  1  3  1   1   2
## 412 1238464  1  1  1  1  1  1  2  1   1   2
## 618 1057067  1  1  1  1  1  1  1  1   1   2
```

```
#calculating mean of v7 column
mean <- mean(as.integer(cancer_data[-indices, 'V7']))
mean
```

```
## [1] 3.544656
```

```
#imputation using mean
data_mean <- cancer_data
data_mean[indices, 'V7'] <- as.integer(mean)

#check
colSums(data_mean == '?')
```

```
##  V1  V2  V3  V4  V5  V6  V7  V8  V9 V10 V11 
##   0   0   0   0   0   0   0   0   0   0   0
```

```
data_mean[indices,]
```

```
##          V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11
## 24  1057013  8  4  5  1  2  3  7  3   1   4
## 41  1096800  6  6  6  9  6  3  7  8   1   2
## 140 1183246  1  1  1  1  1  3  2  1   1   2
## 146 1184840  1  1  3  1  2  3  2  1   1   2
## 159 1193683  1  1  2  1  3  3  1  1   1   2
## 165 1197510  5  1  1  1  2  3  3  1   1   2
## 236 1241232  3  1  4  1  2  3  3  1   1   2
## 250  169356  3  1  1  1  2  3  3  1   1   2
## 276  432809  3  1  3  1  2  3  2  1   1   2
## 293  563649  8  8  8  1  2  3  6 10   1   4
## 295  606140  1  1  1  1  2  3  2  1   1   2
## 298   61634  5  4  3  1  2  3  2  3   1   2
## 316  704168  4  6  5  6  7  3  4  9   1   2
## 322  733639  3  1  1  1  2  3  3  1   1   2
## 412 1238464  1  1  1  1  1  3  2  1   1   2
## 618 1057067  1  1  1  1  1  3  1  1   1   2
```

##Imputation using Regression

I first built a basic regression model to predict V7 column values and trained it on data without missing value indices (those are the ones we would predict later). I then used stepwise regression to ensure I use the best variables in my model. Predictions from this model became the imputations for the missing values.

```
#truncated data without missing values indices
data_reg <- cancer_data[-indices, 2:10]

#regression model
```

```r
model <- lm(V7 ~., data = data_reg)
summary(model)

##
## Call:
## lm(formula = V7 ~ ., data = data_reg)
##
## 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 **
## V2           0.230156   0.041691   5.521 4.83e-08 ***
## V3          -0.067980   0.076170  -0.892  0.37246
## V4           0.340442   0.073420   4.637 4.25e-06 ***
## V5           0.339705   0.045919   7.398 4.13e-13 ***
## V6           0.090392   0.062541   1.445  0.14883
## V8           0.320577   0.059047   5.429 7.91e-08 ***
## V9           0.007293   0.044486   0.164  0.86983
## V10         -0.075230   0.059331  -1.268  0.20524
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## 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

#stepwise regression to find model with best variables
step(model)

## Start:  AIC=1131.43
## V7 ~ V2 + V3 + V4 + V5 + V6 + V8 + V9 + V10
##
##        Df Sum of Sq    RSS    AIC
## - V9    1     0.139 3486.8 1129.5
## - V3    1     4.120 3490.8 1130.2
## - V10   1     8.317 3495.0 1131.0
## <none>              3486.6 1131.4
## - V6    1    10.806 3497.5 1131.5
## - V4    1   111.227 3597.9 1150.9
## - V8    1   152.482 3639.1 1158.7
## - V2    1   157.657 3644.3 1159.6
## - V5    1   283.119 3769.8 1182.8
##
## Step:  AIC=1129.45
## V7 ~ V2 + V3 + V4 + V5 + V6 + V8 + V10
##
##        Df Sum of Sq    RSS    AIC
## - V3    1     4.028 3490.8 1128.2
## - V10   1     8.179 3495.0 1129.0
## <none>              3486.8 1129.5
## - V6    1    11.211 3498.0 1129.7
## - V4    1   114.768 3601.6 1149.6
## - V2    1   158.696 3645.5 1157.8
## - V8    1   160.776 3647.6 1158.2
## - V5    1   285.902 3772.7 1181.3
##
## Step:  AIC=1128.24
## V7 ~ V2 + V4 + V5 + V6 + V8 + V10
##
##        Df Sum of Sq    RSS    AIC
## - V6    1     8.606 3499.4 1127.9
## - V10   1     8.889 3499.7 1128.0
## <none>              3490.8 1128.2
## - V4    1   153.078 3643.9 1155.6
## - V2    1   155.308 3646.1 1156.0
## - V8    1   157.123 3647.9 1156.3
## - V5    1   282.133 3772.9 1179.3
##
## Step:  AIC=1127.92
## V7 ~ V2 + V4 + V5 + V8 + V10
##
##        Df Sum of Sq    RSS    AIC
## - V10   1     5.562 3505.0 1127.0
## <none>              3499.4 1127.9
## - V2    1   159.594 3659.0 1156.4
## - V8    1   169.954 3669.4 1158.3
## - V4    1   206.785 3706.2 1165.1
## - V5    1   295.807 3795.2 1181.3
##
## Step:  AIC=1127.01
## V7 ~ V2 + V4 + V5 + V8
##
##        Df Sum of Sq    RSS    AIC
## <none>              3505.0 1127.0
## - V2    1    155.70 3660.7 1154.7
## - V8    1    172.42 3677.4 1157.8
## - V4    1    201.22 3706.2 1163.1
## - V5    1    290.68 3795.7 1179.4

##
## Call:
## lm(formula = V7 ~ V2 + V4 + V5 + V8, data = data_reg)
##
## Coefficients:
## (Intercept)           V2           V4           V5           V8
##     -0.5360       0.2262       0.3173       0.3323       0.3238
```

```
new_model <- lm(V7 ~ V2 + V4 + V5 + V8, data = data_reg)
summary(new_model)

##
## Call:
## lm(formula = V7 ~ V2 + V4 + V5 + V8, data = data_reg)
##
## 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 **
## V2           0.22617    0.04121   5.488 5.75e-08 ***
## V4           0.31729    0.05086   6.239 7.76e-10 ***
## V5           0.33227    0.04431   7.499 2.03e-13 ***
## V8           0.32378    0.05606   5.775 1.17e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## 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

#regression predictions for missing value indices
regress <- predict(new_model, cancer_data[indices,])
regress

##        24        41       140       146       159       165       236
250
## 5.4585352 7.9816106 0.9872832 1.6218560 0.9807851 2.2157441 2.7152652
1.7634059
##       276       293       295       298       316       322       412
618
## 2.0741942 6.0866099 0.9872832 2.5265324 5.2438347 1.7634059 0.9872832
0.6634986

#imputation using regression
data_regres <- cancer_data
data_regres[indices, 'V7'] <- regress
data_regres$V7 <- as.integer(data_regres$V7)

#making sure values stay within orginal range
data_regres$V7[data_regres$V7 > 10] <- 10
data_regres$V7[data_regres$V7 < 1] <- 1

#check
colSums(data_regres == '?')
```

```
##  V1  V2  V3  V4  V5  V6  V7  V8  V9 V10 V11
##   0   0   0   0   0   0   0   0   0   0   0
```

```
data_regres[indices,]
```

```
##          V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11
## 24  1057013  8  4  5  1  2  5  7  3   1   4
## 41  1096800  6  6  6  9  6  7  7  8   1   2
## 140 1183246  1  1  1  1  1  1  2  1   1   2
## 146 1184840  1  1  3  1  2  1  2  1   1   2
## 159 1193683  1  1  2  1  3  1  1  1   1   2
## 165 1197510  5  1  1  1  2  2  3  1   1   2
## 236 1241232  3  1  4  1  2  2  3  1   1   2
## 250  169356  3  1  1  1  2  1  3  1   1   2
## 276  432809  3  1  3  1  2  2  2  1   1   2
## 293  563649  8  8  8  1  2  6  6 10   1   4
## 295  606140  1  1  1  1  2  1  2  1   1   2
## 298   61634  5  4  3  1  2  2  2  3   1   2
## 316  704168  4  6  5  6  7  5  4  9   1   2
## 322  733639  3  1  1  1  2  1  3  1   1   2
## 412 1238464  1  1  1  1  1  1  2  1   1   2
## 618 1057067  1  1  1  1  1  1  1  1   1   2
```

##Imputation using Regression Perturbation

I used the rnorm function (https://www.rdocumentation.org/packages/compositions/versions/1.40-3/topics/rnorm) to createa normal distribution vector using the predicted values from regression model to come up with

```
set.seed(101)
data_reg_pert <- cancer_data

#using normal distribution variation
data_reg_pert[indices, 'V7'] <- rnorm(length(regress), regress, sd(regress))
data_reg_pert$V7 <- as.integer(data_reg_pert$V7)

#keeping data in the range
data_reg_pert$V7[data_reg_pert$V7 > 10] <- 10
data_reg_pert$V7[data_reg_pert$V7 < 1] <- 1

#check
colSums(data_reg_pert == '?')
```

```
##  V1  V2  V3  V4  V5  V6  V7  V8  V9 V10 V11
##   0   0   0   0   0   0   0   0   0   0   0
```

```
data_reg_pert[indices,]
```

```
##          V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11
## 24  1057013  8  4  5  1  2  4  7  3   1   4
## 41  1096800  6  6  6  9  6  9  7  8   1   2
## 140 1183246  1  1  1  1  1  1  2  1   1   2
## 146 1184840  1  1  3  1  2  2  2  1   1   2
## 159 1193683  1  1  2  1  3  1  1  1   1   2
## 165 1197510  5  1  1  1  2  4  3  1   1   2
## 236 1241232  3  1  4  1  2  4  3  1   1   2
## 250  169356  3  1  1  1  2  1  3  1   1   2
## 276  432809  3  1  3  1  2  4  2  1   1   2
## 293  563649  8  8  8  1  2  5  6 10   1   4
## 295  606140  1  1  1  1  2  2  2  1   1   2
## 298   61634  5  4  3  1  2  1  2  3   1   2
## 316  704168  4  6  5  6  7  8  4  9   1   2
## 322  733639  3  1  1  1  2  1  3  1   1   2
## 412 1238464  1  1  1  1  1  1  2  1   1   2
## 618 1057067  1  1  1  1  1  1  1  1   1   2
```

##Classification Model (KKNN)

Next, I used the KKNN model to check which of these datasets performed better. I started off with the data imputed with mode.

```
library(kknn)

#setting seed for consistent results
set.seed(101)

#creating training & test indices
training <- sample(nrow(cancer_data), size = floor(nrow(cancer_data) * 0.75))
test <- setdiff(1:nrow(cancer_data), training)

data_mode$V7 <- as.integer(data_mode$V7)

iter = 10

KKNNmode <- matrix(NA, nrow=iter, ncol=2)
colnames(KKNNmode) <- c("K-Value","Accuracy")

#looping Ks on the model and documenting accuracy percentage
for(i in 1:iter){
    model_kknn <- kknn( V11~V2+V3+V4+V5+V6+V7+V8+V9+V10,
                    data_mode[training,],
                    data_mode[test,],
                    k=i,
                    distance = 2,
                    kernel = "optimal",
                    scale= TRUE
                    )
    predictions <- predict(model_kknn,newdata = data_mode[test,2:10])
    predictions01 <- as.integer(predictions+0.5)
    KKNNmode[i,] <- c(i, sum(predictions01 == data_mode[test,11]) /
nrow(data_mode[test,]))
}

KKNNmode

##       K-Value  Accuracy
##  [1,]       1 0.9600000
##  [2,]       2 0.9600000
##  [3,]       3 0.9257143
##  [4,]       4 0.9257143
##  [5,]       5 0.9257143
##  [6,]       6 0.9428571
##  [7,]       7 0.9428571
##  [8,]       8 0.9428571
##  [9,]       9 0.9371429
## [10,]      10 0.9371429
```

Next, I tried the model with data imputated with mean.

```
#setting seed for consistent results
set.seed(101)

data_mean$V7 <- as.integer(data_mean$V7)

iter = 10

KKNNmean <- matrix(NA, nrow=iter, ncol=2)
colnames(KKNNmean) <- c("K-Value","Accuracy")

#looping Ks on the model and documenting accuracy percentage
for(i in 1:iter){
    model_kknn <- kknn( V11~V2+V3+V4+V5+V6+V7+V8+V9+V10,
                        data_mean[training,],
                        data_mean[test,],
                        k=i,
                        distance = 2,
                        kernel = "optimal",
                        scale= TRUE
                        )
    predictions <- predict(model_kknn,newdata = data_mean[test,2:10])
    predictions01 <- as.integer(predictions+0.5)
    KKNNmean[i,] <- c(i, sum(predictions01 == data_mean[test,11]) /
nrow(data_mean[test,]))
}

KKNNmean

##       K-Value  Accuracy
##  [1,]       1 0.9600000
##  [2,]       2 0.9600000
##  [3,]       3 0.9257143
##  [4,]       4 0.9257143
##  [5,]       5 0.9257143
##  [6,]       6 0.9428571
##  [7,]       7 0.9428571
##  [8,]       8 0.9428571
##  [9,]       9 0.9371429
## [10,]      10 0.9371429
```

The results for mean and mode imputated data were exactly the same. Next, i tried regression imputated data.

```
#setting seed for consistent results
set.seed(101)

iter = 10

KKNNreg <- matrix(NA, nrow=iter, ncol=2)
colnames(KKNNreg) <- c("K-Value","Accuracy")

#looping Ks on the model and documenting accuracy percentage
for(i in 1:iter){
    model_kknn <- kknn( V11~V2+V3+V4+V5+V6+V7+V8+V9+V10,
                        data_regres[training,],
                        data_regres[test,],
                        k=i,
                        distance = 2,
                        kernel = "optimal",
                        scale= TRUE
                        )
    predictions <- predict(model_kknn,newdata = data_regres[test,2:10])
    predictions01 <- as.integer(predictions+0.5)
    KKNNreg[i,] <- c(i, sum(predictions01 == data_regres[test,11]) /
nrow(data_regres[test,]))
}

KKNNreg

##       K-Value  Accuracy
##  [1,]       1 0.9600000
##  [2,]       2 0.9600000
##  [3,]       3 0.9200000
##  [4,]       4 0.9200000
##  [5,]       5 0.9200000
##  [6,]       6 0.9428571
##  [7,]       7 0.9428571
##  [8,]       8 0.9428571
##  [9,]       9 0.9371429
## [10,]      10 0.9371429
```

The accuracies changed a little but not significantly to call regression imputation better for this dataset. Next,I tried regression perturbation imputated data.

```
#setting seed for consistent results
set.seed(101)

iter = 10

KKNNregpert <- matrix(NA, nrow=iter, ncol=2)
colnames(KKNNregpert) <- c("K-Value","Accuracy")

#looping Ks on the model and documenting accuracy percentage
for(i in 1:iter){
    model_kknn <- kknn( V11~V2+V3+V4+V5+V6+V7+V8+V9+V10,
                    data_reg_pert[training,],
                    data_reg_pert[test,],
                    k=i,
                    distance = 2,
                    kernel = "optimal",
                    scale= TRUE
                    )
    predictions <- predict(model_kknn,newdata = data_reg_pert[test,2:10])
    predictions01 <- as.integer(predictions+0.5)
    KKNNregpert[i,] <- c(i, sum(predictions01 == data_reg_pert[test,11]) /
nrow(data_reg_pert[test,]))
}

KKNNregpert

##       K-Value  Accuracy
##  [1,]       1 0.9600000
##  [2,]       2 0.9600000
##  [3,]       3 0.9200000
##  [4,]       4 0.9200000
##  [5,]       5 0.9200000
##  [6,]       6 0.9428571
##  [7,]       7 0.9428571
##  [8,]       8 0.9428571
##  [9,]       9 0.9371429
## [10,]      10 0.9371429
```

Results did not change for perturbation data as well. I tried dropping missing values next.

```
#setting seed for consistent results
set.seed(101)

cancer_data$V7 <- as.integer(cancer_data$V7)

cancer_data_drop <- cancer_data[-indices,]
```

```
#creating training & test indices
training_drop <- sample(nrow(cancer_data_drop), size =
floor(nrow(cancer_data_drop) * 0.75))
test_drop <- setdiff(1:nrow(cancer_data_drop), training)

iter = 10

KKNNdrop <- matrix(NA, nrow=iter, ncol=2)
colnames(KKNNdrop) <- c("K-Value","Accuracy")

#looping Ks on the model and documenting accuracy percentage
for(i in 1:iter){
    model_kknn <- kknn( V11~V2+V3+V4+V5+V6+V7+V8+V9+V10,
                    cancer_data_drop[training_drop,],
                    cancer_data_drop[test_drop,],
                    k=i,
                    distance = 2,
                    kernel = "optimal",
                    scale= TRUE
                    )
    predictions <- predict(model_kknn,newdata =
cancer_data_drop[test_drop,2:10])
    predictions01 <- as.integer(predictions+0.5)
    KKNNdrop[i,] <- c(i, sum(predictions01 == cancer_data_drop[test_drop,11])
/ nrow(cancer_data_drop[test_drop,]))
}

KKNNdrop

##       K-Value  Accuracy
##  [1,]       1 0.9763314
##  [2,]       2 0.9763314
##  [3,]       3 0.9467456
##  [4,]       4 0.9467456
##  [5,]       5 0.9467456
##  [6,]       6 0.9644970
##  [7,]       7 0.9644970
##  [8,]       8 0.9644970
##  [9,]       9 0.9585799
## [10,]      10 0.9585799
```

The accuracy went up a little but on dataset after dropping missin values. This could be due to overfitting or very small amount of missing values. But the improvement is not very large nonetheless. Lastly, I tried data set with binary variable introduced.

```
#setting seed for consistent results
set.seed(101)

bin_data <- read.table("data_14.1/breast-cancer-wisconsin.data.txt",
stringsAsFactors = FALSE, header = FALSE, sep = ",")
```

```r
#creating binary variable
bin_data$V12[bin_data$V7 == "?"] <- 0
bin_data$V12[bin_data$V7 != "?"] <- 1

#creating new variable to accommodate for missing values
bin_data$V13[bin_data$V7 == "?"] <- 0
bin_data$V13[bin_data$V7 != "?"] <- as.integer(bin_data[-indices,]$V7)

iter = 10

KKNNbin <- matrix(NA, nrow=iter, ncol=2)
colnames(KKNNbin) <- c("K-Value","Accuracy")

#looping Ks on the model and documenting accuracy percentage
for(i in 1:iter){
    model_kknn <- kknn( V11~V2+V3+V4+V5+V6+V8+V9+V10+V12+V13,
                        bin_data[training,],
                        bin_data[test,],
                        k=i,
                        distance = 2,
                        kernel = "optimal",
                        scale= TRUE
                        )
    predictions <- predict(model_kknn,newdata = bin_data[test,2:13])
    predictions01 <- as.integer(predictions+0.5)
    KKNNbin[i,] <- c(i, sum(predictions01 == bin_data[test,11]) /
nrow(bin_data[test,]))
}

KKNNbin
```

```
##       K-Value  Accuracy
##  [1,]       1 0.9542857
##  [2,]       2 0.9542857
##  [3,]       3 0.9314286
##  [4,]       4 0.9314286
##  [5,]       5 0.9314286
##  [6,]       6 0.9371429
##  [7,]       7 0.9371429
##  [8,]       8 0.9371429
##  [9,]       9 0.9314286
## [10,]      10 0.9314286
```

Binary variable did not affect the accruacy much. As noted above, highest acccuracy waas obtained when missing values were dropped.

QUESTION 12.2

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

Given the current world situation with the Covid-19 pandemic, I believe it's a good optimization problem to solve. The virus has a certain Ro value and as long as it stays above 1, virus continues to spread. We shut down the valve by social distancing and lock down and the number of cases drop. The number of patients needing hospitalization is a key data point to track b/c that determines supply and demand balance of hospital ICU beds and respirators needed for the incoming patients. If more patients keep coming in b/c the valve is not closed enough, hospitals would max their capacity and death rate would worsen.

Data we would need to track would be total confirmed positive cases, % of total positive cases needing hospitalization and ICU beds, mortality rate, Ro, and number of respirators in stockpile.

- Variables would be total confirmed cases, % needing hospitalization, total hospital capacity in terms of respirators and total distance traveled by a person on average (as a measure of social distancing)
- constraints would be total capacity not exceeding current respirators on hand and respirators in stock, total confirmed cases not exceeding total population, number of positive cases needing hospitalization not exceeding hospital capacity (key constraint), total distance not being negative.
- We would have to come up with an objective function that minimize the number of patients needing hospitalization.