Assignment

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?

Answer 15.1

As a production engineer I have to make sure that each well production is optimized based on the current state of things. For a simple example we need to maintain the productivity of a well by making sure that it is economic to flow. For this we do optimization. One way to do this is to plan how the fluids will be lifted during the well's lifetime. Here the variables are tubing sizes, reservoir pressure, flow rate etc. The constraints will be at what rate and pressure we want to use with the current size of the tubing. The optimization model for this scenario will be to maximize the productivity of the well by keeping cost as low by avoiding workovers.

Question 14.1

The breast cancer data set breast-cancer-wisconsin.data.txt from 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; (3) the data set when a binary variable is introduced to indicate missing values.

```
#Getting the data
data<-read.table('breast-cancer-wisconsin.data.txt', sep = ',', header = TRUE,
                 stringsAsFactors = FALSE)

#Renaming response variable
colnames(data)[colnames(data) == 'X2.1'] <- 'R'


#Finding columns with missing data
apply(data, 2, function(x) which(x == "?"))
```

```
## $X1000025
## integer(0)
##
## $X5
## integer(0)
##
## $X1
## integer(0)
```

```
##
## $X1.1
## integer(0)
##
## $X1.2
## integer(0)
##
## $X2
## integer(0)
##
## $X1.3
##  [1]  23  40 139 145 158 164 235 249 275 292 294 297 315 321 411 617
##
## $X3
## integer(0)
##
## $X1.4
## integer(0)
##
## $X1.5
## integer(0)
##
## $R
## integer(0)
```

```
#From the above code we can see that only one column X1.3 has missing
#values
```

1. Use Mean/Mode Imputaion

For this question we can see that the most of data is in the form of a factor. So logically it seems appropriate to do a mode imputation. But we will do both imputations here.

```
#Using Mean imputation
df<-data
df<-as.data.frame((apply(df,2,as.numeric))) #Converting data type to Numeric
df[is.na(df[,7]),7]<-round(mean(df$X1.3, na.rm = TRUE),digits = 2)

#Using Mode imputation
df1<-data

#Creating a function to calculate mode
Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}
#Mode for the column Bare Nuclei
d<-Mode(df$X1.3)

#Now replacing the '?' in the column with the mode value
df1$X1.3[df1$X1.3=='?']<-d
```

1. Use regression to impute values for the missing data

```
library(MASS)
library(glmnet)
library(plyr)
```

```
#Getting the data
cancer<-read.table('breast-cancer-wisconsin.data.txt', sep = ',', header = TRUE,
                   stringsAsFactors = FALSE)

#Separating the missing value rows and creating two data frames
idx<-which(cancer$X1.3=='?')
miss_data<-cancer[idx,]

#Modeling without the missing data
new_data<-cancer[-idx,]

#Building the model
model<-lm(X1.3~X5+X1+X1.1+X1.2+X3+X1.4+X1.5+X2.1, data=new_data)

#Model Summary
summary(model)
```

```
##
## Call:
## lm(formula = X1.3 ~ X5 + X1 + X1.1 + X1.2 + X3 + X1.4 + X1.5 +
##     X2.1, data = new_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.5991 -0.4273 -0.2192  0.9055  8.6287
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.23728    0.30746 -13.782  < 2e-16 ***
## X5           0.01867    0.03965   0.471  0.63797
## X1          -0.15678    0.06547  -2.395  0.01691 *
## X1.1         0.18566    0.06538   2.840  0.00465 **
## X1.2         0.22135    0.04123   5.369 1.09e-07 ***
## X3           0.15213    0.05330   2.854  0.00445 **
## X1.4        -0.08616    0.03953  -2.180  0.02962 *
## X1.5        -0.05942    0.05117  -1.161  0.24591
## X2.1         2.51447    0.17745  14.170  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2 on 673 degrees of freedom
## Multiple R-squared:  0.7024, Adjusted R-squared:  0.6989
## F-statistic: 198.6 on 8 and 673 DF,  p-value: < 2.2e-16
```

```r
#The model is only approximately 70% accurate. Applying the variable selection
#method we learned for better prediction

##Using Step-wise regression and evaluating the results
ste<-stepAIC(model, scope = list(lower = X1.3~1, upper = X1.3~X5+X1+X1.1+X1.2+
                                   X3+X1.4+X1.5+X2.1),
             direction = 'both',trace = 1)
```

```
## Start:  AIC=954.54
## X1.3 ~ X5 + X1 + X1.1 + X1.2 + X3 + X1.4 + X1.5 + X2.1
##
##        Df Sum of Sq    RSS     AIC
## - X5    1      0.89 2693.5  952.76
## - X1.5  1      5.40 2698.0  953.90
## <none>              2692.6  954.54
## - X1.4  1     19.01 2711.6  957.34
## - X1    1     22.94 2715.5  958.32
## - X1.1  1     32.26 2724.8  960.66
## - X3    1     32.59 2725.2  960.74
## - X1.2  1    115.31 2807.9  981.14
## - X2.1  1    803.37 3495.9 1130.61
##
## Step:  AIC=952.76
## X1.3 ~ X1 + X1.1 + X1.2 + X3 + X1.4 + X1.5 + X2.1
##
##        Df Sum of Sq    RSS     AIC
## - X1.5  1      5.17 2698.6  952.07
## <none>              2693.5  952.76
## + X5    1      0.89 2692.6  954.54
## - X1.4  1     19.13 2712.6  955.59
## - X1    1     22.59 2716.1  956.46
## - X3    1     32.47 2725.9  958.94
## - X1.1  1     33.69 2727.1  959.24
## - X1.2  1    114.43 2807.9  979.14
## - X2.1  1    962.68 3656.1 1159.17
##
## Step:  AIC=952.07
## X1.3 ~ X1 + X1.1 + X1.2 + X3 + X1.4 + X2.1
##
##        Df Sum of Sq    RSS     AIC
## <none>              2698.6  952.07
## + X1.5  1      5.17 2693.5  952.76
## + X5    1      0.66 2698.0  953.90
## - X1.4  1     22.56 2721.2  955.75
## - X1    1     25.13 2723.8  956.39
## - X1.1  1     33.28 2731.9  958.43
## - X3    1     35.12 2733.7  958.89
## - X1.2  1    110.08 2808.7  977.34
## - X2.1  1    959.79 3658.4 1157.59
```

```r
summary(ste)
```

```
##
```

```
## Call:
## lm(formula = X1.3 ~ X1 + X1.1 + X1.2 + X3 + X1.4 + X2.1, data = new_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.9671 -0.4143 -0.2570  0.8576  8.5857
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.28598    0.30453 -14.074  < 2e-16 ***
## X1          -0.16305    0.06504  -2.507  0.01241 *
## X1.1         0.18760    0.06502   2.885  0.00403 **
## X1.2         0.21361    0.04071   5.247 2.07e-07 ***
## X3           0.15728    0.05307   2.964  0.00315 **
## X1.4        -0.09287    0.03910  -2.375  0.01782 *
## X2.1         2.54159    0.16404  15.494  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.999 on 675 degrees of freedom
## Multiple R-squared:  0.7018, Adjusted R-squared:  0.6991
## F-statistic: 264.7 on 6 and 675 DF,  p-value: < 2.2e-16
```

```r
#We can see that the step model predicts 6 predictors for the best model.
#Now we will build a new regression model using just the 6 predictors and
#evaluate the model

#New model
new<-lm(X1.3 ~ X1 + X1.1 + X1.2 + X3 + X1.4 + X2.1, data = new_data)
#New model summary
summary(new)
```

```
##
## Call:
## lm(formula = X1.3 ~ X1 + X1.1 + X1.2 + X3 + X1.4 + X2.1, data = new_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.9671 -0.4143 -0.2570  0.8576  8.5857
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.28598    0.30453 -14.074  < 2e-16 ***
## X1          -0.16305    0.06504  -2.507  0.01241 *
## X1.1         0.18760    0.06502   2.885  0.00403 **
## X1.2         0.21361    0.04071   5.247 2.07e-07 ***
## X3           0.15728    0.05307   2.964  0.00315 **
## X1.4        -0.09287    0.03910  -2.375  0.01782 *
## X2.1         2.54159    0.16404  15.494  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.999 on 675 degrees of freedom
## Multiple R-squared:  0.7018, Adjusted R-squared:  0.6991
```

```
## F-statistic: 264.7 on 6 and 675 DF,  p-value: < 2.2e-16
```

```
#Testing data
test<-miss_data[,c('X1','X1.1','X1.2','X3','X1.4','X2.1')]

#Predicting
pred<-predict(new,test)

#Binding the predicted data together
total<-data
total<-replace(data,data=='?',round(pred,digits = 0))
```

Use regression with perturbation to impute values for the missing

```
set.seed(42)
d<-data
imp <- rnorm(nrow(miss_data), pred, sd(pred))

#Data also contains negative values using perturbation, thus it requires futher
#investigation

#Binding the predicted data together
purt<-data
purt<-replace(data,data=='?',round(imp,digits = 0))
```

So, we imputed the data using Mean, Mode, Regression and Perturbation. Next step is to see if these imputations have helped in classification. I will use only KNN for this work.

Compare the results and quality of classification models

Conclusion

The code is also attached with this assignment for reference. After completing this part of the exercise the conclusions are as follows:

We can see from the accuracy calculated with and without imputed data to be in the same range and thus does not give good evidence that imputation of any kind was helpful for this example.

```{r message=TRUE, warning=FALSE}

library(kknn)

#Building the Mean Imputed data
df_mean<-df[,2:11]
df_index_mean<-sample(1:nrow(df_mean), size = round(0.7*nrow(df_mean)), replace=FALSE)

#Training data
train_mean<-df_mean[df_index_mean,]
#Testing data
test_mean<-df_mean[-df_index_mean,]

acc_mean<-list()
for (j in 1:10){ #Loop to test different k between 1-10

  model_mean = kknn(R~.,
                    train=train_mean,
                    test=test_mean,
                    k = j,
                    scale = TRUE)
  pred_mean = fitted(model_mean)
  pre_mean<-ifelse(pred_mean>2,4,2)
  acc_mean[j]=sum(pre_mean == test_mean[,10])/nrow(test_mean)
}
```

Figure 1: Image 1.

```{r cache=TRUE, include=FALSE}
#Building the Mode Imputed data
df_mode<-df1[,2:11]
df_index_mode<-sample(1:nrow(df_mode), size = round(0.7*nrow(df_mode)), replace=FALSE)

#Training data
train_mode<-df_mode[df_index_mode,]
#Testing data
test_mode<-df_mode[-df_index_mode,]

acc_mode<-list()
for (a in 1:10){ #Loop to test different k between 1-10

  model_mode = kknn(R~.,
                    train=train_mode,
                    test=test_mode,
                    k = a,
                    scale = TRUE)
  pred_mode = fitted(model_mode)
  pre_mode<-ifelse(pred_mode>2,4,2)
  acc_mode[a]=sum(pre_mode == test_mode[,10])/nrow(test_mode)
}
```

Figure 2: Image 2.

```{r eval=FALSE, cache=TRUE, include=FALSE}
#Building the Regression Imputed data
df_reg<-total[,2:11]
df_index_reg<-sample(1:nrow(df_reg), size = round(0.7*nrow(df_reg)), replace=FALSE)

#Training data
train_reg<-df_reg[df_index_reg,]
#Testing data
test_reg<-df_reg[-df_index_reg,]

acc_reg<-list()
for (b in 1:10){ #Loop to test different k between 1-10

  model_reg = kknn(R~.,
                   train=train_reg,
                   test=test_reg,
                   k = b,
                   scale = TRUE)
  pred_reg = fitted(model_reg)
  pre_reg<-ifelse(pred_reg>2,4,2)
  acc_reg[b]=sum(pre_reg == test_reg[,10])/nrow(test_reg)
}
```

Figure 3: Image 3.

```{r cache=TRUE, include=FALSE}
#Building the Perturbation Imputed data
df_purt<-purt[,2:11]
df_index_purt<-sample(1:nrow(df_purt), size = round(0.7*nrow(df_purt)), replace=FALSE)

#Training data
train_purt<-df_purt[df_index_purt,]
#Testing data
test_purt<-df_purt[-df_index_purt,]

acc_purt<-list()
for (c in 1:10){ #Loop to test different k between 1-10

  model_purt = kknn(R~.,
                    train=train_purt,
                    test=test_purt,
                    k = c,
                    scale = TRUE)
  pred_purt = fitted(model_purt)
  pre_purt<-ifelse(pred_purt>2,4,2)
  acc_purt[c]=sum(pre_purt == test_purt[,10])/nrow(test_purt)
}
```

Figure 4: Image 4.

```r
#Building without imputation
data_without<-data[-idx,]
df_without<-data_without[,2:11]
df_index_without<-sample(1:nrow(df_without), size = round(0.7*nrow(df_without)), replace=FALSE)

#Training data
train_without<-df_without[df_index_without,]
#Testing data
test_without<-df_without[-df_index_without,]

#Building the model using Mean Imputation
acc_without<-list()
for (i in 1:10){ #Loop to test different k between 1-10

  model_knn = kknn(R~.,
                   train=train_without,
                   test=test_without,
                   k = i,
                   scale = TRUE)
  pred_knn = fitted(model_knn)
  pre<-ifelse(pred_knn>2,4,2)
  acc_without[i]=sum(pre == test_without[,10])/nrow(test_without)
}

#Printing Accuracy for Mean Imputed Model
print(acc_mean)
#Printing Accuracy for Mode Imputed Model
print(acc_mode)
#Printing Accuracy for Regression Imputed Model
print(acc_reg)
#Printing Accuracy for Perturbation Imputed Model
print(acc_purt)
#Printing Accuracy for Mean Imputed Model
print(acc_without)
```

Figure 5: Table 5.