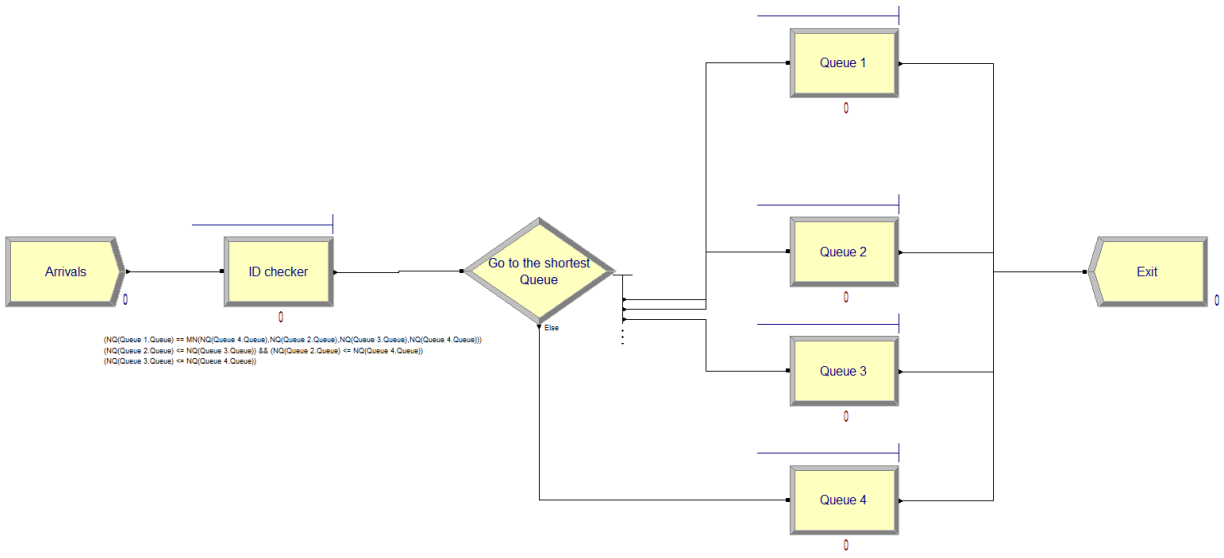
Homework 6

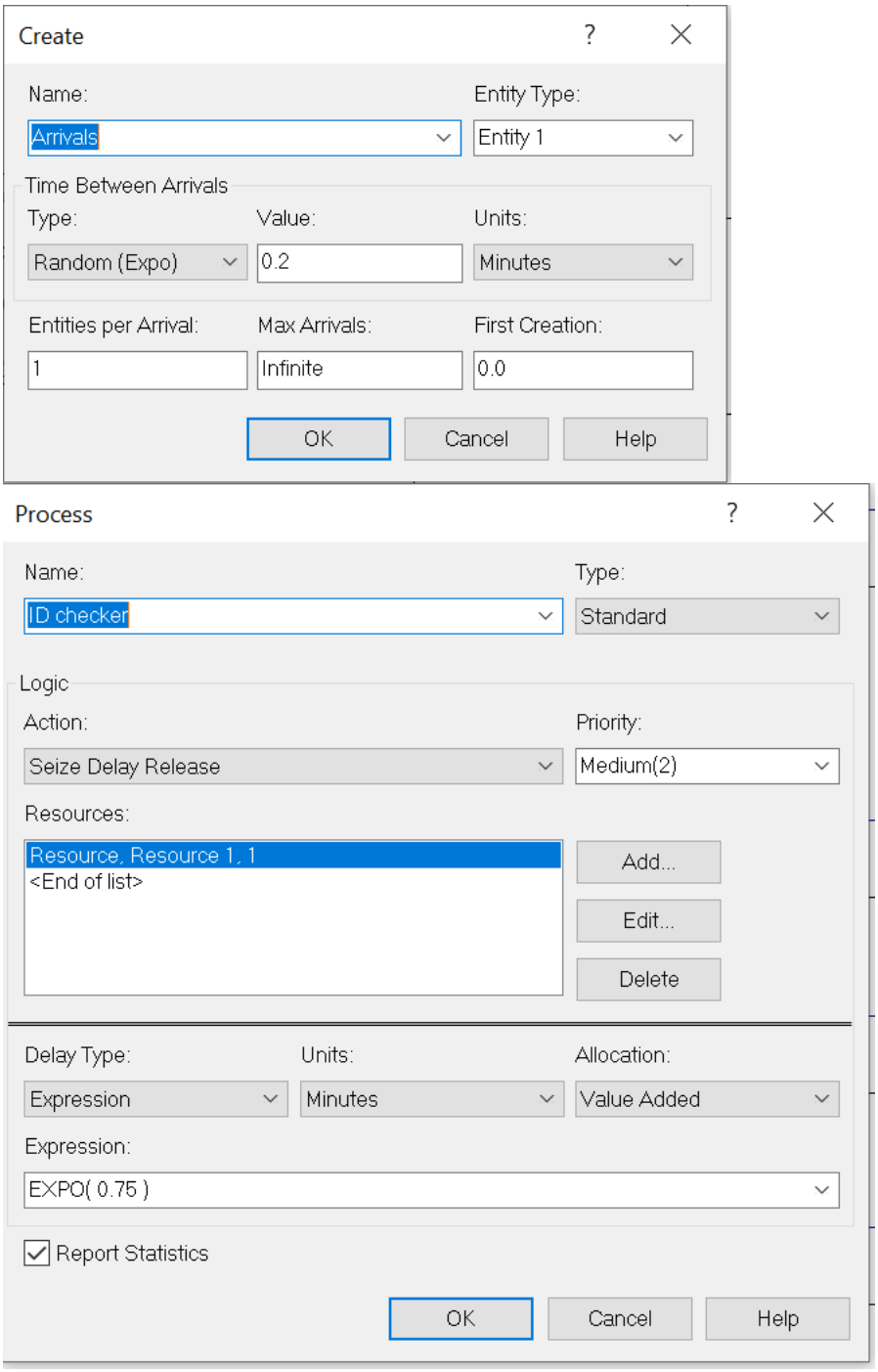
Question 13.2

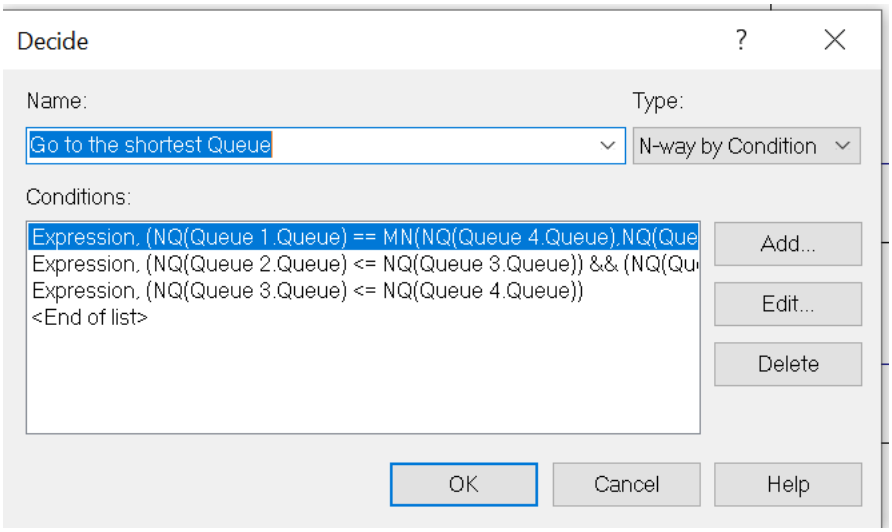
In this problem you, can simulate a simplified airport security system at a busy airport. Passengers arrive according to a Poisson distribution with λ1 = 5 per minute (i.e., mean inter-arrival rate μ1 = 0.2 minutes) to the ID/boarding-pass check queue, where there are several servers who each have exponential service time with mean rate μ2 = 0.75 minutes. [Hint: model them as one block that has more than one resource.] After that, the passengers are assigned to the shortest of the several personal-check queues, where they go through the personal scanner (time is uniformly distributed between 0.5 minutes and 1 minute).

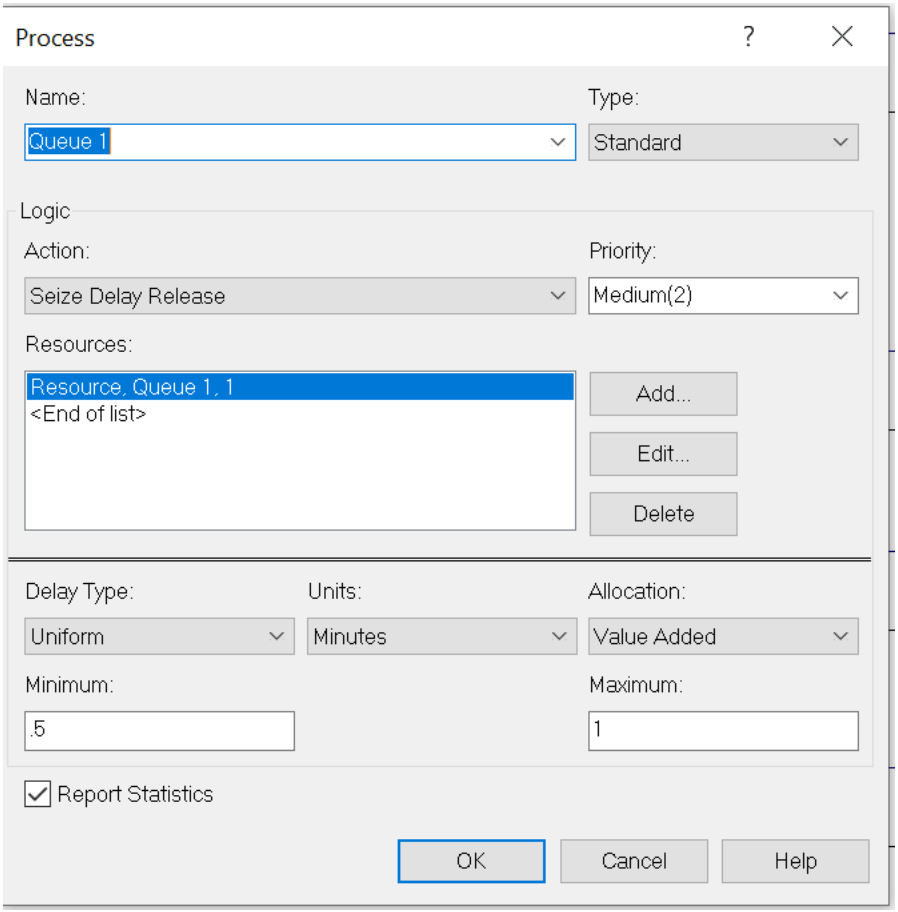
Answer:

I used Arena to solve this problem. The report generated by Arena was attached. Below are the screen shot of my setup:




Create
Name:
Arrivals
Entity Type:
Entity 1
Time Between Arrivals
Type:
Value:
Units:
Random (Expo)
0.2
Minutes
Entities per Arrival:
Max Arrivals:
First Creation:
1
Infinite
0.0
OK
Cancel
Help
Process
Name:
ID checker
Type:
Standard
Logic
Action:
Priority:
Seize Delay Release
Medium(2)
Resources:
Resource, Resource 1, 1
<End of list>
Add...
Edit...
Delete
Delay Type:
Units:
Allocation:
Expression
Minutes
Value Added
Expression:
EXPO(0.75)
Report Statistics
OK
Cancel
Help





The key of this assignment is to find the right capacity of the resource in the ID Checker, and the right number of queues after the ID check. My solution is to set capacity to 4, and added 4 queues to the program.

Question 14.1

The breast cancer data set breast-cancer-wisconsin.data.txt 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.

```
library(nnet)
library(MASS)
library(kknn)
cancer<- read.table("http://archive.ics.uci.edu/ml/machine-learning-databases
/breast-cancer-wisconsin/breast-cancer-wisconsin.data",sep = ",", stringsAsFa
ctors = FALSE, header=F)
head(cancer)
```

```
##        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
```

check for the missings

```
str(cancer)
```

```
## 'data.frame':    699 obs. of  11 variables:
##  $ V1 : int  1000025 1002945 1015425 1016277 1017023 1017122 1018099 10185
61 1033078 1033078 ...
##  $ V2 : int  5 5 3 6 4 8 1 2 2 4 ...
##  $ V3 : int  1 4 1 8 1 10 1 1 1 2 ...
##  $ V4 : int  1 4 1 8 1 10 1 2 1 1 ...
##  $ V5 : int  1 5 1 1 3 8 1 1 1 1 ...
##  $ V6 : int  2 7 2 3 2 7 2 2 2 2 ...
##  $ V7 : chr  "1" "10" "2" "4" ...
##  $ V8 : int  3 3 3 3 3 9 3 3 1 2 ...
##  $ V9 : int  1 2 1 7 1 7 1 1 1 1 ...
##  $ V10: int  1 1 1 1 1 1 1 1 5 1 ...
##  $ V11: int  2 2 2 2 2 4 2 2 2 2 ...
```

We noticed that V1-V11 are all integer values, except V7. I will take a closer look at V7.

```
table(cancer$V7)
```

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

V7 has missing value, which was marked as "?". Now check how many data are missing in V7.

```
mis<-subset(cancer,cancer$V7=="?")
nrow(mis)
```

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

16 obs were missing, which account for 16/699=2.29% of the total data. We can go ahead and impute values for the missings.

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

Find the mode value

```
v1<-nrow(subset(cancer,cancer$V7==1))
v2<-nrow(subset(cancer,cancer$V7==2))
v3<-nrow(subset(cancer,cancer$V7==3))
v4<-nrow(subset(cancer,cancer$V7==4))
v5<-nrow(subset(cancer,cancer$V7==5))
v6<-nrow(subset(cancer,cancer$V7==6))
v7<-nrow(subset(cancer,cancer$V7==7))
v8<-nrow(subset(cancer,cancer$V7==8))
v9<-nrow(subset(cancer,cancer$V7==9))
v10<-nrow(subset(cancer,cancer$V7==10))
v<-c( v1 , v2 , v3 , v4  , v5 ,  v6 , v7 , v8 , v9 , v10 )

mode<-which.max(v)
mode
```

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

1 is the mode

Assign mode value to the missings

```
cancer1<-cancer
cancer1$V7[cancer1$V7=="?"]<-mode
sum(cancer1$V7=="?")#We have sucessfully changed "?" to 1
```

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

```
cancer1$V7<-as.integer(cancer1$V7)
str(cancer1$V7)
```

```
##  int [1:699] 1 10 2 4 1 10 10 1 1 1 ...
```

2. Use regression to impute values for the missing data.

Leave out the response variables and V1 which is ID, and use stepwise method to predict the V7 with all the other variables

```
cancer2<-cancer[cancer$V7!="?",2:10]
cancer2$V7 <- as.integer(cancer2$V7)
```

70% for training

```
mask_train<-sample(nrow(cancer2), size = floor(nrow(cancer2) * 0.7))
```

training data set

```
train<-cancer2[mask_train,]
```

Using the remaining data for test

```
test<-cancer2[-mask_train, ]  # all rows except training
```

Fit the model

```
reg<- multinom(V7 ~ ., data = train)
```

```
## # weights:  100 (81 variable)
## initial  value 1100.635674
## iter  10 value 738.506697
## iter  20 value 546.160554
## iter  30 value 479.243308
## iter  40 value 432.340956
## iter  50 value 424.368253
```

```
## iter  60 value 423.160896
## iter  70 value 422.578859
## iter  80 value 422.074623
## iter  90 value 421.298496
## iter 100 value 420.270150
## final  value 420.270150
## stopped after 100 iterations
```

```
summary(reg)
```

```
## Call:
## multinom(formula = V7 ~ ., data = train)
##
## Coefficients:
##    (Intercept)          V2          V3          V4          V5          V6
## 2    -4.717949  0.15570995  0.12979546 -0.30938051 -0.04399826  0.05644216
## 3    -3.800999 -0.03723532  0.21116166  0.05390461  0.21550949 -0.05180473
## 4    -5.753214  0.21279724  0.37648965  0.02305562 -0.05610244 -0.12010129
## 5    -4.944981  0.18300893 -0.03782338  0.27806962  0.26608868  0.05526566
## 6   -24.646647 -1.64196772  1.02195343  3.75165874 -1.81272892 -4.18796189
## 7    -8.215568  0.63027705 -0.75256975  0.72682523  0.32657153 -0.92634166
## 8    -7.309866  0.05760286  0.06672619  0.32941415  0.30333228  0.34938933
## 9    -7.060588 -0.10358529  0.12739754  0.27452567 -0.02637578 -0.33538970
## 10   -6.454512  0.31794210 -0.09753105  0.31544033  0.40304005  0.11732450
##             V8          V9         V10
## 2   0.41234919  0.048187886  0.40044583
## 3  -0.01566152  0.077776199  0.27493870
## 4   0.11368783  0.149248626  0.12758427
## 5   0.14886235  0.136433546 -0.62391550
## 6  -7.58742416  3.527511651  3.83488941
## 7   0.15229462  0.370226768  0.22533082
## 8   0.25222710 -0.007207503 -0.37493148
## 9   0.44430271  0.211937012  0.53460981
## 10  0.30522100  0.012094298  0.07435297
##
## Std. Errors:
##    (Intercept)          V2        V3         V4          V5         V6
## 2    0.6055716  0.11705757 0.2051448  0.2145342  0.16917444  0.1931160
## 3    0.5790774  0.11598304 0.1920415  0.1910212  0.12829261  0.1973117
## 4    0.9164978  0.15165487 0.2078790  0.2036061  0.18925469  0.2208576
## 5    0.7776761  0.11516890 0.1957008  0.2069221  0.12151056  0.1704099
## 6   16.5881518 11.53426761 9.8912871 21.1457943 32.39631601 13.2292612
## 7    2.2080246  0.27591292 0.4714759  0.4266245  0.21225069  0.6549271
## 8    1.0864427  0.13921935 0.2154042  0.2247189  0.13152735  0.1712012
## 9    1.4563364  0.23652054 0.3367735  0.3410405  0.24003063  0.3755600
## 10   0.6161176  0.08704563 0.1469826  0.1519413  0.09454802  0.1287049
##            V8          V9       V10
## 2   0.1596219  0.13066645 0.1631515
## 3   0.1710745  0.12445152 0.1682904
## 4   0.1904857  0.12947967 0.2313830
```

```
## 5   0.1566109  0.11040643 0.4188275
## 6  28.5317700 11.00520741 7.5821159
## 7   0.3002198  0.23097065 0.2869629
## 8   0.1718295  0.11825923 0.3041753
## 9   0.2768113  0.21178338 0.2254439
## 10  0.1175911  0.08551794 0.1546946
##
## Residual Deviance: 840.5403
## AIC: 1002.54
```

Use stepwise method to re-fit the model with all the predictors

```
stp<-stepAIC(reg, direction="both")
```

```
stp$anova
```

```
## Stepwise Model Path
## Analysis of Deviance Table
##
## Initial Model:
## V7 ~ V2 + V3 + V4 + V5 + V6 + V8 + V9 + V10
##
## Final Model:
## V7 ~ V2 + V4 + V5 + V10
##
##
##   Step Df  Deviance Resid. Df Resid. Dev       AIC
## 1                         397   840.5403 1002.5403
## 2 - V9  9  6.562125       406   847.1024  991.1024
## 3 - V6  9 10.261035       415   857.3635  983.3635
## 4 - V8  9 14.064150       424   871.4276  979.4276
## 5 - V3  9 15.657716       433   887.0853  977.0853
```

```
summary(stp)
```

```
## Call:
## multinom(formula = V7 ~ V2 + V4 + V5 + V10, data = train)
##
## Coefficients:
##    (Intercept)            V2          V4            V5        V10
## 2    -4.244079   0.233314972 -0.01831701    0.09842886  0.4461068
## 3    -3.977251  -0.002954217  0.23971446    0.26597426  0.2680801
## 4    -5.984342   0.311416141  0.39004871    0.07554026  0.1362541
```

```
## 5    -4.784467   0.200656304  0.40964542    0.32030620 -0.5156343
## 6  -174.618945 -26.217078440 51.51823310 -138.74236122 24.7517131
## 7    -8.738836   0.568360901  0.22925155    0.21899372  0.2849337
## 8    -6.659441   0.142579032  0.63234144    0.36065952 -0.2551220
## 9    -6.861861   0.006896686  0.52630341    0.09042632  0.5020227
## 10   -6.004218   0.346808200  0.42398836    0.46102001  0.1049665
##
## Std. Errors:
##    (Intercept)          V2          V4         V5       V10
## 2    0.4985172  0.11167151  0.15751771 0.14376253 0.1480218
## 3    0.4758144  0.11000987  0.12846841 0.11967032 0.1655908
## 4    0.7961750  0.13979062  0.14234663 0.16011529 0.2147479
## 5    0.6899172  0.10958731  0.12103548 0.11680544 0.3917847
## 6    2.3055036 20.49769251 22.95502299 2.30818043 2.5236261
## 7    1.6504876  0.22498564  0.20329670 0.18762045 0.2174785
## 8    0.9550593  0.12773761  0.13886534 0.12713083 0.2792393
## 9    1.1871724  0.20899273  0.20195468 0.21076758 0.2147189
## 10   0.5434100  0.08321057  0.09448628 0.09089512 0.1463541
##
## Residual Deviance: 887.0853
## AIC: 977.0853
```

Generate the model from stepwise method

```
model<- lm(V7~V2+V4+V5+V8, cancer2)
summary(model)
```

```
##
## Call:
## lm(formula = V7 ~ V2 + V4 + V5 + V8, data = cancer2)
##
## 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
```

Use test dataset to validate

```
pred<-round(predict(model,test))
acc<-sum(pred == test$V7) / nrow(test)
acc
```

```
## [1] 0.3268293
```

0.356 accuracy rate is not good.But this is the only model we got, so I will go ahead and use this model to impute the missings

Get the subset of the data with the missings, and the subset with all the valid data points

```
mis2<-subset(cancer,V7=="?")
ok<-subset(cancer,V7!="?")
```

Assign the imputed values to V7

```
mis2$V7<-round(predict(model,mis2))
```

Put these data back to the cancer dataset

```
cancer2final<-rbind(ok,mis2)
cancer2final$V7<-as.integer(cancer2final$V7)
```

make the values outside of the orignal range back to [1,10]

```
cancer2final$V7[cancer2final$V7 > 10] <- 10
cancer2final$V7[cancer2final$V7 < 1] <- 1
```

3. Use regression with perturbation to impute values for the missing data.

```
set.seed(123)

v7<-round(predict(model,mis2))

mis3<-subset(cancer,V7=="?")

v7new<-round(rnorm(nrow(mis3),v7,sd(v7)))
```

make the values outside of the orignal range back to [1,10]

```
mis3$V7<-v7new

mis3$V7[mis3$V7 > 10] <- 10
mis3$V7[mis3$V7 < 1] <- 1

cancer3<-rbind(ok,mis3)
cancer3$V7<-as.integer(cancer3$V7)
```

4.Compare the results and quality of classification models (e.g., SVM, KNN) build using

(1) the data sets from questions 1,2,3;

70% for training

```
mask_train1<-sample(nrow(cancer1), size = floor(nrow(cancer1) * 0.7))
```

4.1.1, with mode imputation

```
acc<-rep(0,25)
```

training data set

```
train1<-cancer1[mask_train1,]
```

Using the remaining data for test

```
test1<-cancer1[-mask_train1, ]

for (k in 1:5){
knn4.1.1 <- kknn(V11~V2+V3+ V4+ V5+ V6+ V7+ V8+ V9+ V10,train1,test1,k=k)
pred1 <- as.integer(fitted(knn4.1.1)+0.5)
acc[k]<-sum(pred1 == test1$V11) / nrow(test1)
}
```

4.1.2, with regression imputation

```
train2<-cancer2final[mask_train1,]
test2<-cancer2final[-mask_train1, ]

for (k in 1:5){
        knn4.1.2 <- kknn(V11~V2+V3+ V4+ V5+ V6+ V7+ V8+ V9+ V10,train2,test2,
k=k)
```

```
        pred2 <- as.integer(fitted(knn4.1.2)+0.5)
        acc[k+5]<-sum(pred2 == test2$V11) / nrow(test2)
}
```

4.1.3, with regression imputation

```
train3<-cancer3[mask_train1,]
test3<-cancer3[-mask_train1, ]

for (k in 1:5){
        knn4.1.3 <- kknn(V11~V2+V3+ V4+ V5+ V6+ V7+ V8+ V9+ V10,train3,test3,
k=k)
        pred3<- as.integer(fitted(knn4.1.3)+0.5)
        acc[k+10]<-sum(pred3 == test3$V11) / nrow(test3)
}
```

4.2, the data that remains after data points with missing values are removed;

```
cancer4<-subset(cancer,V7!="?")
cancer4$V7<-as.integer(cancer4$V7)
train4<-cancer4[mask_train1,]
test4<-cancer4[-mask_train1, ]

for (k in 1:5){
        knn4.2 <- kknn(V11~V2+V3+ V4+ V5+ V6+ V7+ V8+ V9+ V10,train4,test4,k=
k)
        pred4<- as.integer(fitted(knn4.2)+0.5)
        acc[k+15]<-sum(pred4 == test4$V11) / nrow(test4)
}
```

4.3, the data set when a binary variable is introduced to indicate missing values

Add a binary variable to the original data to indicate if V7 is missing or not. 0=missing,1= not missing

```
cancer5 <- cancer
cancer5$V12[cancer5$V7 == "?"] <- 0
cancer5$V12[cancer5$V7 != "?"] <- 1
```

Create interaction factor for V7 and V12.

```
cancer5$V13[cancer5$V7 == "?"] <- 0
cancer5$V13[cancer5$V7 != "?"] <- as.integer(ok$V7)
```

```
train5<-cancer5[mask_train1,]
test5<-cancer5[-mask_train1, ]
```

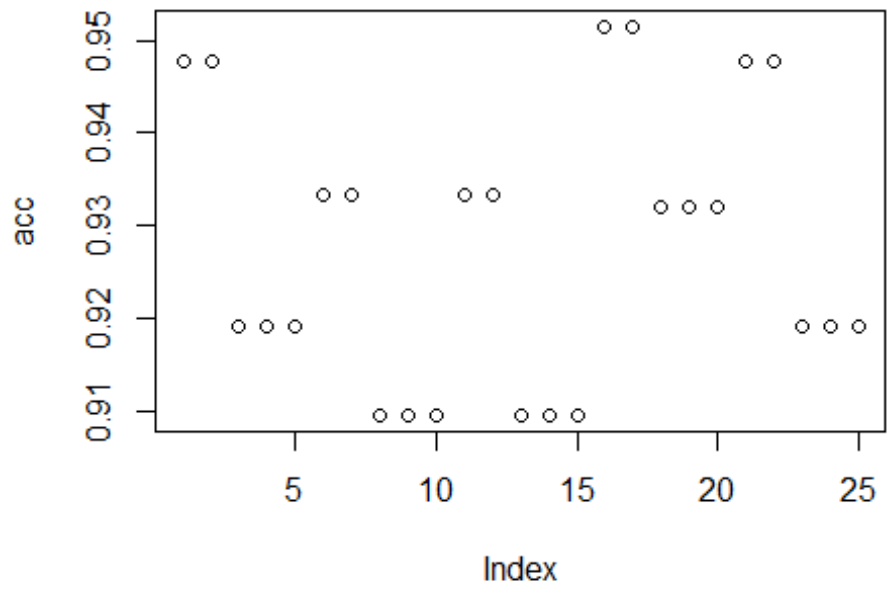
Use the interaction factor in the modeling.

```
for (k in 1:5){
        knn4.3 <- kknn(V11~V2+V3+ V4+ V5+ V6+ V8+ V9+ V10+V13,train5,test5,k=
k)
        pred5<- as.integer(fitted(knn4.3)+0.5)
        acc[k+20]<-sum(pred5 == test5$V11) / nrow(test5)
}

acc
```

```
##  [1] 0.9476190 0.9476190 0.9190476 0.9190476 0.9190476 0.9333333 0.9333333
##  [8] 0.9095238 0.9095238 0.9095238 0.9333333 0.9333333 0.9095238 0.9095238
## [15] 0.9095238 0.9514563 0.9514563 0.9320388 0.9320388 0.9320388 0.9476190
## [22] 0.9476190 0.9190476 0.9190476 0.9190476
```

```
plot(acc)
```




```
which.max(acc)
```

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

There isn't much differences between the differenct methods to deal with the missing data (the accuracy rate are all withn 90%-95%).

However, removing the missing values, generated a slightly higher predictive accuracy at k=1, for the knn model.

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:

Graduate students may want to decide which courses to choose in each semester, in order to maximize the GPA when graduating.

Data needed:

1. Workload of each courses and the time needed per week

2. Personal schedules and estimated time that can be used for study

3. Study plan that indicates which courses must be taken (based on school requirements, personal interests, and career goals)

4. The order of the coursers (take introduction courses before the ones that require deeper understanding)

5. Total credits taken each semester should meet school requirements

6. The amount paid should be within the education budget