Intro to Analytics Modeling - Homework 9

Kunle Lawal, Ali Lakdawala, Anu Rana, Mihir Tulpule

1.

Describe a situation or problem from your job, everyday life, current events, etc., for which a design of experiments approach would be appropriate.

One situation in which we would use a design of experiments approach would be optimizing a website to increase click rates and online web traffic (in the context of e-commerce). With the ever-increasing prevalence of Internet-based marketing, increasing click through rates is an excellent way to boost revenues as well as search engine optimization success. Here, one can influence the probability of a website visit/link click through various factors: the text/color of font, the presence/placement/type of advertisements (banner ads, sidebar ads, etc.), variation of content, and so forth.

In such cases, randomly selecting factors to change would likely yield a low rate of success and expend excessive amounts of time/energy (severely skewing the exploration vs. exploitation ratio of the endeavor). A full factorial design would also be generally infeasible given the sheer number of potential factors that can be changed on a single website. Therefore, testing a subset of said factors through a fractional factorial design would likely be optimal, allowing the website owner to not only begin reaping the benefits of the adjustments as soon as possible, but also allowing the website to be optimized as per the desires of the online market/community in as efficient and effective a manner as possible.

2

To determine the value of 10 different yes/no features to the market value of a house (large yard, solar roof, etc.), a real estate agent plans to survey 50 potential buyers, showing a fictitious house with different combinations of features. To reduce the survey size, the agent wants to show just 16 fictitious houses. Use R's FrF2 function (in the FrF2 package) to find a fractional factorial design for this experiment: what set of features should each of the 16 fictitious houses have? Note: the output of FrF2 is "1" (include) or "-1" (don't include) for each feature.

```
#install.packages("FrF2")
```

```
library(FrF2)

features = c(
  "greaterthan2000sqft",
  "bigBasement",
  "modernkitchen",
  "recentlyrenovated",
  "largeRooms",
  "antique",
  "parks",
  "multiplefloors",
  "goodschool",
  "kidfriendlynbhood"
)

FrF2(16, factor.names = features)
```

```
##    greaterthan2000sqft bigBasement modernkitchen recentlyrenovated
## 1                    1          -1            -1                 1
## 2                   -1           1            -1                 1
## 3                   -1           1             1                -1
## 4                   -1          -1             1                 1
## 5                    1           1            -1                -1
## 6                   -1           1            -1                -1
## 7                    1          -1             1                -1
## 8                   -1           1             1                 1
## 9                   -1          -1             1                -1
## 10                  -1          -1            -1                -1
## 11                   1          -1            -1                -1
## 12                   1           1            -1                 1
## 13                   1           1             1                -1
## 14                   1          -1             1                 1
## 15                   1           1             1                 1
## 16                  -1          -1            -1                 1
##    largeRooms antique parks multiplefloors goodschool kidfriendlynbhood
## 1          -1      -1     1              1          1                 1
## 2          -1       1    -1             -1         -1                 1
## 3          -1      -1     1              1         -1                 1
## 4           1      -1    -1             -1         -1                 1
## 5           1      -1    -1             -1          1                 1
## 6          -1       1    -1              1          1                -1
## 7          -1       1    -1             -1          1                 1
## 8          -1      -1     1             -1          1                -1
## 9           1      -1    -1              1          1                -1
## 10          1       1     1              1         -1                 1
## 11         -1      -1     1             -1         -1                -1
## 12          1      -1    -1              1         -1                -1
## 13          1       1     1             -1         -1                -1
## 14         -1       1    -1              1         -1                -1
## 15          1       1     1              1          1                 1
## 16          1       1     1             -1          1                -1
## class=design, type= FrF2
```

3

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.

3 Part 1

For part one, we first replaced the ? in column 7 with NA to make the data easier to work with. Next we removed the NA from column 7 to compute the mean and the mode. Next we replaced NA with the value we

got for the mean and mode.

```
rm(list = ls())
library(data.table)
setwd("/Users/alimujtaba/Google Drive/isye6501modelling/isye6501homeworks/hw9")
data = read.table("breast-cancer-wisconsin.data.txt",sep = ",", header = FALSE)


data1 <- copy(data)

data1[data1 == "?"] <- NA

#dataset with NA's removed to be used later with optional parts.
data2 <- na.omit(data1)
#locate the NA's
#is.na(data1)
#how many missings per variable?
colSums(is.na(data1))
```

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

```
#data2[,7]

# We have 16 NA's we can use similar methodology for mode and mean
meandata = mean(as.numeric(data2[,7]))
data1[,7] <- sapply(data1[, 7], as.numeric)
data1[is.na(data1[, 7]), 7] <- meandata
#check
#data1[,7] <- commented out for readability of PDF


data3 <- copy(data)

getmode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}

mode <- getmode(data3[,7])
data3[data3 == "?"] <- NA
data3[,7] <- sapply(data3[, 7], as.numeric)
data3[is.na(data3[, 7]), 7] <- mode
#data3[,7] <- commented out for readability of PDF
```

3 Part 2

For part two, we first replaced the ? in column 7 with NA to make the data easier to work with. Next we removed the NA from column 7 in order to use the data in a regression. Next we replaced NA with the value for our regression.

```
# Using regression for imputation
#install.packages("mice")
library(mice)
```

```r
data4 <- copy(data)
data4[data4 == "?"] <- NA
data4[,7] <- sapply(data4[, 7], as.numeric)


# Deterministic regression imputation via mice
imp <- mice(data4, method = "norm.predict", m = 1)
```

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

```r
# Store data
data_imp <- complete(imp)
#data_imp[,7] <- commented out for readability of PDF
```

3 Part 3

For part three, we first replaced the ? in column 7 with NA to make the data easier to work with. Next we removed the NA from column 7 in order to use the data in a regression with perturbation. Next we replaced NA with the value for our regression with perturbation.

```r
#Using regression with perturbation for imputation
data5 <- copy(data)
data5[data5 == "?"] <- NA
data5[,7] <- sapply(data5[, 7], as.numeric)


# Deterministic regression imputation via mice with pertubation
imp1 <- mice(data5, method = "norm.nob", m = 1)
```

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

```r
# Store data
data_imp1 <- complete(imp1)
#data_imp1[,7] <- commented out for readability of PDF
```

3 Part 4

```r
library(kknn)
```

```r
knn_process <- function(data1){



kmax <- 30

# use train.kknn for leave-one-out cross-validation up to k=kmax

model <- train.kknn(V11~.,data1,kmax=kmax,scale=TRUE)

# create array of prediction qualities

accuracy <- rep(0,kmax)

# calculate prediction qualities

for (k in 1:kmax) {
  predicted <- as.integer(fitted(model)[[k]][1:nrow(data1)] + 0.5) # round off to 0 or 1
  accuracy[k] <- mean(predicted == data$V11)
}

# show accuracies

print(accuracy)

}
#Mean
knn_process(data1)
```

```
##  [1] 0.9499285 0.9499285 0.9155937 0.9155937 0.9141631 0.9298999 0.9298999
##  [8] 0.9256080 0.9213162 0.9198856 0.9184549 0.9313305 0.9313305 0.9313305
## [15] 0.9313305 0.9313305 0.9327611 0.9327611 0.9327611 0.9356223 0.9341917
## [22] 0.9341917 0.9341917 0.9370529 0.9384835 0.9370529 0.9384835 0.9384835
## [29] 0.9399142 0.9399142
```

```r
#Removed
knn_process(data2)
```

```
##  [1] 0.5879828 0.5879828 0.5679542 0.5679542 0.5665236 0.5722461 0.5722461
##  [8] 0.5722461 0.5693848 0.5693848 0.5679542 0.5765379 0.5736767 0.5722461
## [15] 0.5722461 0.5736767 0.5736767 0.5736767 0.5736767 0.5722461 0.5722461
## [22] 0.5722461 0.5722461 0.5736767 0.5736767 0.5736767 0.5722461 0.5736767
## [29] 0.5722461 0.5736767
```

```r
#Mode
knn_process(data3)
```

```
##  [1] 0.9484979 0.9484979 0.9170243 0.9170243 0.9155937 0.9313305 0.9313305
##  [8] 0.9256080 0.9213162 0.9213162 0.9198856 0.9327611 0.9327611 0.9327611
## [15] 0.9341917 0.9341917 0.9341917 0.9341917 0.9341917 0.9356223 0.9327611
## [22] 0.9341917 0.9356223 0.9356223 0.9384835 0.9384835 0.9384835 0.9399142
## [29] 0.9399142 0.9399142
```

```r
#Regression
knn_process(data_imp)
```

```
##  [1] 0.9456366 0.9456366 0.9141631 0.9141631 0.9127325 0.9284692 0.9284692
##  [8] 0.9227468 0.9184549 0.9184549 0.9170243 0.9313305 0.9313305 0.9313305
## [15] 0.9327611 0.9327611 0.9327611 0.9341917 0.9341917 0.9356223 0.9327611
## [22] 0.9356223 0.9356223 0.9356223 0.9384835 0.9370529 0.9384835 0.9399142
## [29] 0.9399142 0.9413448
```

```
#Regression with perturbation
knn_process(data_imp1)
```

```
##  [1] 0.9456366 0.9456366 0.9198856 0.9198856 0.9184549 0.9298999 0.9284692
##  [8] 0.9227468 0.9198856 0.9198856 0.9184549 0.9298999 0.9298999 0.9298999
## [15] 0.9313305 0.9313305 0.9327611 0.9341917 0.9356223 0.9370529 0.9356223
## [22] 0.9356223 0.9370529 0.9370529 0.9370529 0.9370529 0.9384835 0.9384835
## [29] 0.9384835 0.9413448
```

From the KNN Analysis above. We can see that the accuracy is most affected when the data points are removed versus all other scenarios. We can infer that it is better to impute the data point that to remove it in this particular data set.