HW10

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.

```
#Input the data
data<-read.csv('breast-cancer-wisconsin.data.txt',header=F,stringsAsFactors = F)
head(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
```

```
#Find the missing data
miss<-which(data$V7== '?')
miss
```

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

```
impute_me<-data[-miss,]
```

Mean method

```
mean<-mean(as.numeric(impute_me$V7))
mean
```

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

```
mean_mod<-round(mean)
mean_mod
```

```
## [1] 4
```

From above, mean method indicates that missing value should be 4.

##Mode method

```
mode<-which.max(tabulate(as.numeric(impute_me$V7)))
mode
```

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

With mode method, it ends up with 1 as the imputation value for Colume 7.

##Use regression to impute

```
impute_mod<-impute_me[,-1]  # Exclude the Column ID
impute_mod2<-impute_mod[,-10] # Exclude V11, which is our response for further analysis
impute_mod2<-cbind(impute_mod$V7,impute_mod[,1:5],impute_mod[,7:9])
colnames(impute_mod2)[1]<- 'V7'
impute_mod2$V7<-as.integer(impute_mod2$V7)
```

Since Column 1 denotes ID number, it should be deleted.Also, in our analysis, it's better to exclude categorical response(Column 11). Including it in the imputation analysis will cause overfitting.

```
#Use stepwise method to select the most related variables.

library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
ctrl<-trainControl(method='repeatedcv',number=5,repeats=5)
impute_mod<-data.frame(impute_mod)
lm_step<-train(V7~.,data=impute_mod2,"lmStepAIC",scope=list(lower=V7~1, upper=V7~.),direction='backward
```

Step: AIC=933.64 .outcome ~ V2 + V4 + V5 + V6 + V8 + V9 + V10

```
   Df Sum of Sq    RSS     AIC
```

2617.7 933.64 - V6 1 13.13 2630.8 935.06 - V10 1 17.66 2635.3 936.23 - V9 1 26.73 2644.4 938.58 - V2 1 171.36 2789.0 974.95 - V4 1 248.24 2865.9 993.52 - V8 1 263.20 2880.9 997.08 - V5 1 348.67 2966.3 1017.05

```
# Execute the linear regression with only the optimal factors devrived from stepwise method.
impute_reg<-lm(V7~ V2 + V4 + V5 + V6 + V8 + V9 + V10, data=impute_mod2)
summary(impute_reg)
```

```
##
## Call:
## lm(formula = V7 ~ V2 + V4 + V5 + V6 + V8 + V9 + V10, data = impute_mod2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -9.6884 -0.9129 -0.2961  0.7161  8.7039
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.576177   0.189599  -3.039  0.00247 **
## V2           0.226760   0.041510   5.463 6.60e-08 ***
## V4           0.299210   0.057053   5.244 2.10e-07 ***
## V5           0.333912   0.045451   7.347 5.88e-13 ***
## V6           0.076768   0.060640   1.266  0.20596
## V8           0.310819   0.058017   5.357 1.16e-07 ***
## V9           0.004206   0.044345   0.095  0.92446
## V10         -0.077772   0.059253  -1.313  0.18979
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.274 on 675 degrees of freedom
## Multiple R-squared:  0.6145, Adjusted R-squared:  0.6105
## F-statistic: 153.7 on 7 and 675 DF,  p-value: < 2.2e-16
```

Linear regression output shows that V6, V9, V10 is not strongly related with the response(V7).

We exclude them.

```
impute_reg2<-lm(V7~ V2 + V4 + V5 + V8, data=impute_mod2)
summary(impute_reg2)
```

```
##
## Call:
## lm(formula = V7 ~ V2 + V4 + V5 + V8, data = impute_mod2)
##
## 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
```

Adjusted R^2 is 0.6107 from above regression model and let's see how it perform under LOOCV.

```
#Cross-validate the model.

SStot<- sum((impute_mod2$V7-mean(impute_mod2$V7))^2)
```

```r
totsse<-0
for (i in 1:nrow(impute_mod2)){
  mod_step_i= lm(V7 ~ V2 + V4 + V5 + V8,data=impute_mod2[-i,])
  predict<- predict(mod_step_i,impute_mod2[i,])
  totsse<-totsse+((predict-impute_mod[i,1])^2)
}
R2_mod_step<-1- totsse/SStot
R2_mod_step
```

```
##         1
## 0.6445699
```

The R^2 from LOOCV is even higher than the one from regression, which infers that the model is reliable.

```r
#Restructure the data.
missing_mod2<-data[miss,]
missing_mod2<-missing_mod2[,-1]
missing_mod2<-missing_mod2[,-10]
missing_mod2$V7[missing_mod2$V7 == '?']<-NA
missing_mod2<-cbind(missing_mod2$V7,missing_mod2[,1:5],missing_mod2[,7:9])
head(missing_mod2)
```

```
##     missing_mod2$V7 V2 V3 V4 V5 V6 V8 V9 V10
## 24             <NA>  8  4  5  1  2  7  3   1
## 41             <NA>  6  6  6  9  6  7  8   1
## 140            <NA>  1  1  1  1  1  2  1   1
## 146            <NA>  1  1  3  1  2  2  1   1
## 159            <NA>  1  1  2  1  3  1  1   1
## 165            <NA>  5  1  1  1  2  3  1   1
```

```r
#get the predicted results from regression
predicted_reg<-round(predict(impute_reg2,missing_mod2))
predicted_reg
```

```
##  24  41 140 146 159 165 236 250 276 293 295 298 316 322 412 618
##   5   8   1   2   1   2   3   2   2   6   1   3   5   2   1   1
```

##Impute with regression and perturbation

I use rnorm to factor the variabilities into the prediction. The standard deviation i used here is the residual standard error from the linear regression since it can reflect the variabilities more broadly.

```r
perturbation<-rnorm(length(predicted_reg), mean=predicted_reg, sd=summary(impute_reg2)$sigma)
perturbation
```

```
##  [1]  7.0977446 11.3907361  4.8159242  3.3075931  0.6277916  3.9939778
##  [7]  5.9528859  0.2063312  2.9213984  8.6376872  2.9464637  3.2049316
## [13]  8.7712699  0.3411606 -2.3327318 -0.6335187
```

```r
perturb_mod<-round(perturbation)
perturb_mod
```

```
##  [1]  7 11  5  3  1  4  6  0  3  9  3  3  9  0 -2 -1
```

```r
#The output is on a scale of 1-10
perturb_mod[perturb_mod<1]<-1
perturb_mod[perturb_mod>10]<-10
perturb_mod
```

```
##  [1]  7 10  5  3  1  4  6  1  3  9  3  3  9  1  1  1
```

Optional Question

(Optional) Compare the results and quality of classification models (e.g., SVM, KNN) build using (1) the data sets from questions 1,2,3;

```r
#Prepare the data
set.seed(1)
data_miss<-data[miss,]
data_mean<-data_miss
data_mean$V7[data_mean$V7=='?']<-mean_mod
data_mean_full<-rbind(impute_me,data_mean)
data_mean_full<-data_mean_full[,-1]
index<-runif(nrow(data_mean_full))
data_mean_full<-data_mean_full[order(index),]
head(data_mean_full)
```

```
##     V2 V3 V4 V5 V6 V7 V8 V9 V10 V11
## 503  4  1  1  2  2  1  2  1   1   2
## 478  4  1  1  1  2  1  1  1   1   2
## 528  4  1  1  1  2  1  3  1   1   2
## 568  4  1  1  1  2  3  2  1   1   2
## 424  5  1  3  1  2  1  2  1   1   2
## 165  5  1  1  1  2  4  3  1   1   2
```

##Experiment with SVM model:

##Mean method:

```r
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
library(ggplot2)

ksvm_mean<- ksvm(V11~.,data=data_mean_full,type="C-svc",kernel="vanilladot",C=100,scaled=TRUE)
```

```
##  Setting default kernel parameters
```

```
pred_mean<-predict(ksvm_mean,data_mean_full[,1:9])

accuracy_mean<-sum(pred_mean == data_mean_full$V11)/nrow(data_mean_full)
accuracy_mean
```

```
## [1] 0.9713877
```

```
ksvm_mean
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 100
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 61
##
## Objective Function Value : -4773.306
## Training error : 0.028612
```

##Mode method:

```
data_mode<-data_miss
data_mode$V7[data_mode$V7=='?']<-mode
data_mode_full<-rbind(impute_me,data_mode)
data_mode_full<-data_mode_full[,-1]
index<-runif(nrow(data_mode_full))
data_mode_full<-data_mode_full[order(index),]

ksvm_mode<- ksvm(V11~.,data=data_mode_full,type="C-svc",kernel="vanilladot",C=100,scaled=TRUE)
```

```
##  Setting default kernel parameters
```

```
pred_mode<-predict(ksvm_mode,data_mode_full[,1:9])

accuracy_mod<-sum(pred_mode == data_mode_full$V11)/nrow(data_mode_full)
accuracy_mod
```

```
## [1] 0.9728183
```

```
ksvm_mode
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 100
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 58
##
## Objective Function Value : -4572.789
## Training error : 0.027182
```

##Regression method:

```
data_reg<-data_miss
for (i in 1:length(predicted_reg)){
  data_reg$V7[i]<-predicted_reg[i]
}
data_reg_full<-rbind(impute_me,data_reg)
data_reg_full<-data_reg_full[,-1]

ksvm_reg<- ksvm(V11~.,data=data_reg_full,type="C-svc",kernel="vanilladot",C=100,scaled=TRUE)
```

```
##  Setting default kernel parameters
```

```
pred_reg<-predict(ksvm_reg,data_reg_full[,1:9])

accuracy_reg<-sum(pred_reg == data_reg_full$V11)/nrow(data_reg_full)
accuracy_reg
```

```
## [1] 0.9728183
```

```
ksvm_reg
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 100
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 59
##
## Objective Function Value : -4691.052
## Training error : 0.027182
```

##Perturbation method:

```
data_per<-data_miss
for (i in 1:length(perturb_mod)){
  data_per$V7[i]<-perturb_mod[i]
```

```r
}
data_per_full<-rbind(impute_me,data_per)
data_per_full<-data_per_full[,-1]

ksvm_per<- ksvm(V11~.,data=data_per_full,type="C-svc",kernel="vanilladot",C=100,scaled=TRUE)
```

```
##  Setting default kernel parameters
```

```r
pred_per<-predict(ksvm_per,data_per_full[,1:9])

accuracy_per<-sum(pred_per == data_per_full$V11)/nrow(data_per_full)
accuracy_per
```

```
## [1] 0.9685265
```

```r
ksvm_per
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 100
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 61
##
## Objective Function Value : -5011.227
## Training error : 0.031474
```

##Remove missing data:

```r
data_rem<-impute_me[,-1]

ksvm_rem<- ksvm(V11~.,data=data_rem,type="C-svc",kernel="vanilladot",C=100,scaled=TRUE)
```

```
##  Setting default kernel parameters
```

```r
pred_rem<-predict(ksvm_rem,data_rem[,1:9])

accuracy_rem<-sum(pred_rem == data_rem$V11)/nrow(data_rem)
accuracy_rem
```

```
## [1] 0.9751098
```

```r
ksvm_rem #Check model output
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
```

```
##  parameter : cost C = 100
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 52
##
## Objective Function Value : -4120.225
## Training error : 0.02489
```

##Compare model accuracy

The model that generates the highest SVM accuracy(0.9751098) is the one derived from data which removes all the missing data. For this standpoint, when dealing with large dataset, removing missing data could be a better option.

##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?

If i were the campaign manager for US presidential election, i would utilize the optimization method to allocate funds to different campaign channels to maximize the number of ballots.

variables:

x= total labor cost spent in state i. y= social media commercials cost in state i. z= the budget for state i a= 1 if the fund spent in state i is over z, 0 if not. ....