ISYE 6501 Intro Analytics Modeling – HW10

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.

The first step is to read in the dataset and do some simple data summary. From the code below, I get to know this dataset contains 699 **data points**, 10 **predictors**, and one binary (2/4) **response**. Only one variable has 16 missing values. I replaced these missing values with the average V7 value:3.22.

```
> df[df=="?"]<-NA
> sapply(df, function(x) sum(is.na(x)))
 V1  V2  V3  V4  V5  V6  V7  V8  V9 V10 V11 
  0   0   0   0   0   0  16   0   0   0   0 
```

```
# ISYE 6501 Intro Analytics Modeling - HW10
# IP breast-cancer-wisconsin.data.txt
library(mice)
# 1. Loading and examining data
df<-read.delim("breast-cancer-wisconsin.data.txt", header = FALSE, sep = ",")
#### Display Head Lines ####
head(df,2)
#### Show summary, number of row of last column##
nrow(df)
ncol(df)
df[df=="?"]<-NA
sapply(df, function(x) sum(is.na(x)))
df$V7<-as.numeric(df$V7)
df$Dep_Var<-(df$V11-2)/2
df<- df[order(df$V1),]
```

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

Then I built a linear regression model to predict missing V7 using V2-V10, the predicted value is from 2.30 to 5.46 with an average 3.05 which is very close to the average of non-missing V7.

```
Call:  glm(formula = V7 ~ V2 + V3 + V4 + V5 + V6 + V8 + V9 + V10, family = gaussian, 
    data = df_rg_t)
Coefficients:
(Intercept)           V2           V3           V4           V5           V6           V8  
   1.862817     0.068118     0.087939     0.110046    -0.076950     0.043216     0.044536  
         V9          V10  
   0.119422     0.001405  
Degrees of Freedom: 682 Total (i.e. Null);  674 Residual
Null Deviance:	    3156 
Residual Deviance: 2422 	AIC: 2823
> predict(rg,df_rg_p,type="response")
      24       41      140      146      159      165      236      250      276      293 
3.990654 4.294718 2.305086 2.568395 2.457029 2.665311 2.859213 2.529074 2.704631 5.463968 
     295      298      316      322      412      618 
2.348302 3.343528 4.308317 2.529074 2.305086 2.260550 
```

```
#2. Use the mean/mode imputation method to impute values for the missing data
df_mean<-df
for(i in 1:ncol(df_mean)){
  df_mean[is.na(df_mean[,i]), i] <- mean(df_mean[,i], na.rm = TRUE)
}
df_mean <- df_mean[order(df_mean$V1),]

#2.Use regression to impute values for the missing data. .#
df_rg_t<-df[! is.na(df$V7),]
df_rg_p<-df[is.na(df$V7),]

rg<-glm(V7~V2+V3+V4+V5+V6+V8+V9+V10,family = gaussian,df_rg_t)
rg
MSE_train<-mean(rg$residuals^2)
df_rg_p$V7<-predict(rg,df_rg_p,type="response")

df_rg<-rbind(df_rg_t,df_rg_p)
df_rg <- df_rg[order(df_rg$V1),]
```

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

I used the r package *mice* to replace missing value by regression with perturbation, I get the below value with average value 3.17, which is even closer to the non-missing V7 value, but the standard deviation is obviously bigger than regression, the value is from 2 to 11.

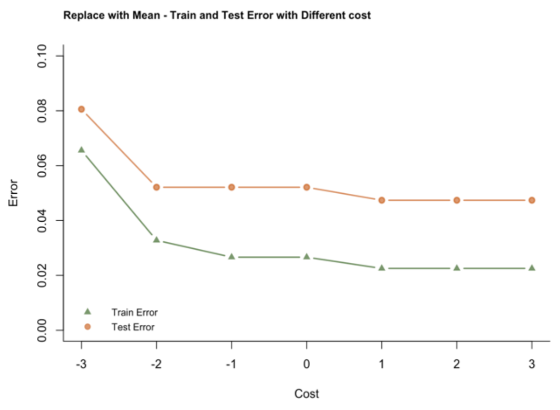
```
       24        41       140       146       159       165       236       250       276 
 2.000000  2.704631  2.000000  2.000000  2.000000  2.000000  2.000000  3.000000  4.000000 
      293       295       298       316       322       412       618 
11.000000  2.000000  3.000000  6.000000  2.000000  3.000000  2.000000 
```

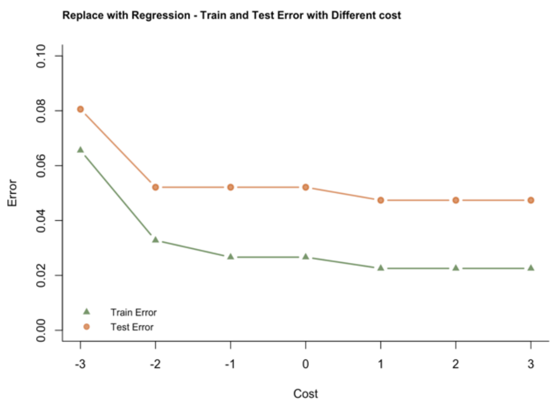
```
#3.Use regression with perturbation to impute values for the missing data. #
imp <- mice(df[,c(2:10)], method="norm.predict")
df_rgp<-complete(imp)
df_rgp$V1<-df$V1
df_rgp$Dep_Var<-df$Dep_Var
df_rgp <- df_rgp[order(df_rgp$V1),]
```

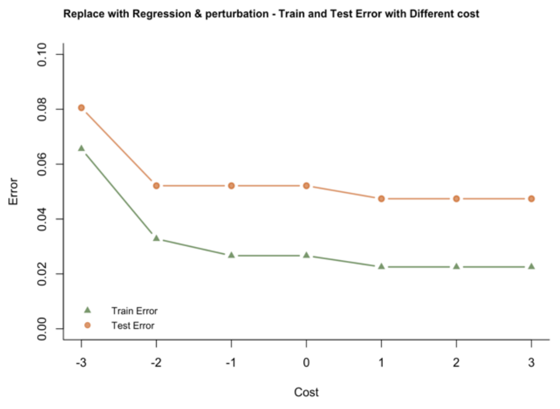
4. (Optional) Compare the results and quality of classification models.

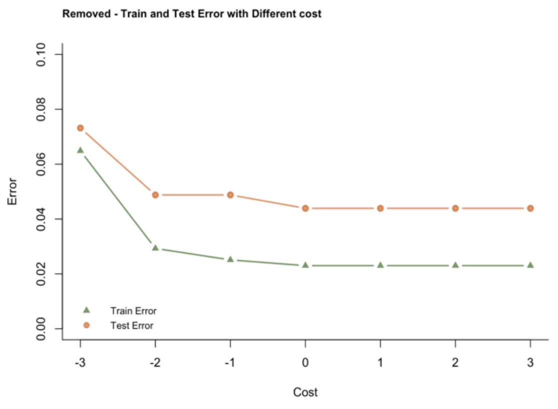
Then I created two more datasets, one with all missing values removed. Another dataset has all missing values replaced with 0, and a flag variable is created to indicates if V7 variable is missing or not. I used SVM for each dataset and tried different Cost Value.

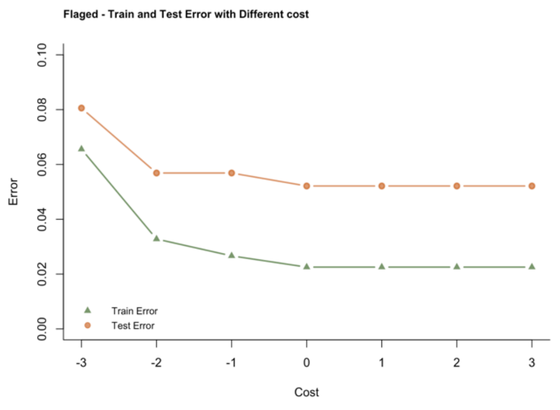
Comparing the 5 different methods, removing missing value gives us minimal training and testing error, and all other models give very similar testing and training errors. I think the reason might be: 1. The number of missing data is small; 2. The missing value might due to error, quality for those data points might not be good, including those might bring bias.

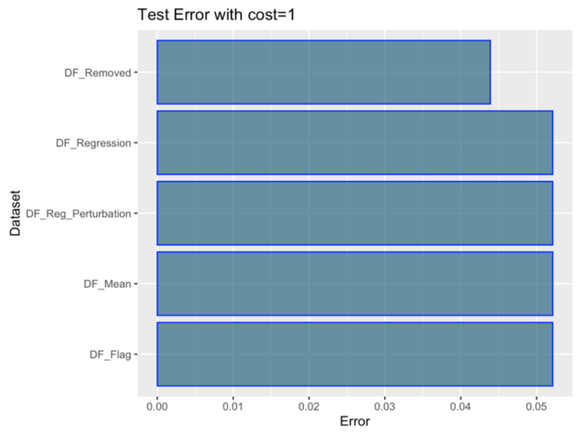












```r
#4. (Optional) Compare the results and quality of classification models
set.seed(123)
g <- sample(1:2,size=nrow(df),replace=TRUE,prob=c(0.7,0.3))
df_mean_t <- df_mean[g==1,]
df_mean_v <- df_mean[g==2,]

library(kernlab)
cost<-c(10^(-3:3)) ## Create a table to record models results
summary_df_mean <- data.frame(cost= c(-3:3))
```

```
indep_var<-c('V2','V3','V4','V5','V6','V8','V9','V10','V7')
for (c in cost){
  ## Create classifier with different C value
  svm <- ksvm(
    as.matrix(df_mean_t[,indep_var]), as.factor(df_mean_t[,'Dep_Var']), type=
"C-svc", kernel="vanilladot", C=c,
    scaled=TRUE )
  for (x in 1:10){
    summary_df_mean[cost==c,"Dataset"]="DF_Mean"
    summary_df_mean[cost==c,"Error_Train"]=as.numeric(svm@error)
    summary_df_mean[cost==c,"Error_Test"]= sum((df_mean_v$Dep_Var-as.numeric(
predict(svm,df_mean_v[,indep_var]))+1)^2)/nrow(df_mean_v)
    cf_mx<-table(df_mean_v$Dep_Var, predict(svm,df_mean_v[,indep_var]))
    summary_df_mean[cost==c,"TPR"]=cf_mx[4]/(cf_mx[4]+cf_mx[2]) ## True posit
ive rate
    summary_df_mean[cost==c,"TNR"]=cf_mx[1] /(cf_mx[1]+cf_mx[3]) ## Ture nega
tive rate
  }
}

df_rg_t<- df_rg[g==1,]
df_rg_v <- df_rg[g==2,]

summary_df_rg <- data.frame(cost= c(-3:3))
indep_var<-c('V2','V3','V4','V5','V6','V8','V9','V10','V7')
for (c in cost){
  ## Create classifier with different C value
  svm <- ksvm(
    as.matrix(df_rg_t[,indep_var]), as.factor(df_rg_t[,'Dep_Var']), type="C-s
vc", kernel="vanilladot", C=c,
    scaled=TRUE )
  for (x in 1:10){
    summary_df_rg[cost==c,"Dataset"]="DF_Regression"
    summary_df_rg[cost==c,"Error_Train"]=as.numeric(svm@error)
    summary_df_rg[cost==c,"Error_Test"]= sum((df_rg_v$Dep_Var-as.numeric(pred
ict(svm,df_rg_v[,indep_var]))+1)^2)/nrow(df_rg_v)
    cf_mx<-table(df_rg_v$Dep_Var, predict(svm,df_rg_v[,indep_var]))
    summary_df_rg[cost==c,"TPR"]=cf_mx[4]/(cf_mx[4]+cf_mx[2]) ## True positiv
e rate
    summary_df_rg[cost==c,"TNR"]=cf_mx[1] /(cf_mx[1]+cf_mx[3]) ## Ture negati
ve rate
  }
}

df_rgp_t <- df_rgp[g==1,]
df_rpg_v <- df_rgp[g==2,]

summary_df_rgp <- data.frame(cost= c(-3:3))
indep_var<-c('V2','V3','V4','V5','V6','V8','V9','V10','V7')
```

```
for (c in cost){
  ## Create classifier with different C value
  svm <- ksvm(
    as.matrix(df_rgp_t[,indep_var]), as.factor(df_rgp_t[,'Dep_Var']), type="C
-svc", kernel="vanilladot", C=c,
    scaled=TRUE )
  for (x in 1:10){
    summary_df_rgp[cost==c,"Dataset"]="DF_Reg_Perturbation"
    summary_df_rgp[cost==c,"Error_Train"]=as.numeric(svm@error)
    summary_df_rgp[cost==c,"Error_Test"]= sum((df_rpg_v$Dep_Var-as.numeric(pr
edict(svm,df_rpg_v[,indep_var]))+1)^2)/nrow(df_rpg_v)
    cf_mx<-table(df_rpg_v$Dep_Var, predict(svm,df_rpg_v[,indep_var]))
    summary_df_rgp[cost==c,"TPR"]=cf_mx[4]/(cf_mx[4]+cf_mx[2]) ## True positi
ve rate
    summary_df_rgp[cost==c,"TNR"]=cf_mx[1] /(cf_mx[1]+cf_mx[3]) ## Ture negat
ive rate
  }
}

df_rm_t <- na.omit(df[g==1,])
df_rm_v <- na.omit(df[g==2,])

summary_df_rm <- data.frame(cost= c(-3:3))
indep_var<-c('V2','V3','V4','V5','V6','V8','V9','V10','V7')
for (c in cost){
  ## Create classifier with different C value
  svm <- ksvm(
    as.matrix(df_rm_t[,indep_var]), as.factor(df_rm_t[,'Dep_Var']), type="C-s
vc", kernel="vanilladot", C=c,
    scaled=TRUE )
  for (x in 1:10){
    summary_df_rm[cost==c,"Dataset"]="DF_Removed"
    summary_df_rm[cost==c,"Error_Train"]=as.numeric(svm@error)
    summary_df_rm[cost==c,"Error_Test"]= sum((df_rm_v$Dep_Var-as.numeric(pred
ict(svm,df_rm_v[,indep_var]))+1)^2)/nrow(df_rm_v)
    cf_mx<-table(df_rm_v$Dep_Var, predict(svm,df_rm_v[,indep_var]))
    summary_df_rm[cost==c,"TPR"]=cf_mx[4]/(cf_mx[4]+cf_mx[2]) ## True positiv
e rate
    summary_df_rm[cost==c,"TNR"]=cf_mx[1] /(cf_mx[1]+cf_mx[3]) ## Ture negati
ve rate
  }
}

df_flag<-df
df_flag$V7_miss_flag <- as.numeric(is.na(df_flag$V7))
df_flag[is.na(df_flag)]<-0

df_flag_t <- df_flag[g==1,]
df_flag_v <- df_flag[g==2,]
```

```
summary_df_flag <- data.frame(cost= c(-3:3))
indep_var<-c('V2','V3','V4','V5','V6','V8','V9','V10','V7','V7_miss_flag')
for (c in cost){
  ## Create classifier with different C value
  svm <- ksvm(
    as.matrix(df_flag_t[,indep_var]), as.factor(df_flag_t[,'Dep_Var']), type=
"C-svc", kernel="vanilladot", C=c,
    scaled=TRUE )
  for (x in 1:10){
    summary_df_flag[cost==c,"Dataset"]="DF_Flag"
    summary_df_flag[cost==c,"Error_Train"]=as.numeric(svm@error)
    summary_df_flag[cost==c,"Error_Test"]= sum((df_flag_v$Dep_Var-as.numeric(
predict(svm,df_flag_v[,indep_var]))+1)^2)/nrow(df_flag_v)
    cf_mx<-table(df_flag_v$Dep_Var, predict(svm,df_flag_v[,indep_var]))
    summary_df_flag[cost==c,"TPR"]=cf_mx[4]/(cf_mx[4]+cf_mx[2]) ## True posit
ive rate
    summary_df_flag[cost==c,"TNR"]=cf_mx[1] /(cf_mx[1]+cf_mx[3]) ## Ture nega
tive rate
  }
}

summary<-rbind(summary_df_mean,summary_df_rg,summary_df_rgp,summary_df_rm,sum
mary_df_flag)
```

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?

Design a road trip visiting all capitals.
Goal: Minimize cost
Variables:

X_i= Total days spent in City$_i$

Y_i= Number of visit to City$_i$

C_i= Cost spent in City$_i$ per day (average hotel + parking + dinning price)

D_i= Driving distance from the last City

Constraints:

1. Travel for 3 months
2. No more than 10 hours driving/24 hour
3. Spend at least one day in each City

Date I need: a list of all capitals and their avg downtown hotel price, parking price and dining price; a matrix of driving distance of all capitals.