hw6\_r\_code.R

aaa

Wed Jun 27 21:33:05 2018

set.seed(5)  
databc <- read.table("breastcancer.txt", stringsAsFactors = FALSE, header = F, sep=",")  
head(databc)

## 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

#show where the missing data is  
for (i in 2:11) {  
 print (paste0("v",i))  
 print(table(databc[,i]))  
}

## [1] "v2"  
##   
## 1 2 3 4 5 6 7 8 9 10   
## 145 50 108 80 130 34 23 46 14 69   
## [1] "v3"  
##   
## 1 2 3 4 5 6 7 8 9 10   
## 384 45 52 40 30 27 19 29 6 67   
## [1] "v4"  
##   
## 1 2 3 4 5 6 7 8 9 10   
## 353 59 56 44 34 30 30 28 7 58   
## [1] "v5"  
##   
## 1 2 3 4 5 6 7 8 9 10   
## 407 58 58 33 23 22 13 25 5 55   
## [1] "v6"  
##   
## 1 2 3 4 5 6 7 8 9 10   
## 47 386 72 48 39 41 12 21 2 31   
## [1] "v7"  
##   
## ? 1 10 2 3 4 5 6 7 8 9   
## 16 402 132 30 28 19 30 4 8 21 9   
## [1] "v8"  
##   
## 1 2 3 4 5 6 7 8 9 10   
## 152 166 165 40 34 10 73 28 11 20   
## [1] "v9"  
##   
## 1 2 3 4 5 6 7 8 9 10   
## 443 36 44 18 19 22 16 24 16 61   
## [1] "v10"  
##   
## 1 2 3 4 5 6 7 8 10   
## 579 35 33 12 6 3 9 8 14   
## [1] "v11"  
##   
## 2 4   
## 458 241

#show observations with missing data  
databc[which(databc$V7=="?"),]

## V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11  
## 24 1057013 8 4 5 1 2 ? 7 3 1 4  
## 41 1096800 6 6 6 9 6 ? 7 8 1 2  
## 140 1183246 1 1 1 1 1 ? 2 1 1 2  
## 146 1184840 1 1 3 1 2 ? 2 1 1 2  
## 159 1193683 1 1 2 1 3 ? 1 1 1 2  
## 165 1197510 5 1 1 1 2 ? 3 1 1 2  
## 236 1241232 3 1 4 1 2 ? 3 1 1 2  
## 250 169356 3 1 1 1 2 ? 3 1 1 2  
## 276 432809 3 1 3 1 2 ? 2 1 1 2  
## 293 563649 8 8 8 1 2 ? 6 10 1 4  
## 295 606140 1 1 1 1 2 ? 2 1 1 2  
## 298 61634 5 4 3 1 2 ? 2 3 1 2  
## 316 704168 4 6 5 6 7 ? 4 9 1 2  
## 322 733639 3 1 1 1 2 ? 3 1 1 2  
## 412 1238464 1 1 1 1 1 ? 2 1 1 2  
## 618 1057067 1 1 1 1 1 ? 1 1 1 2

#nrow(databc[which(databc$V7=="?"),])/nrow(databc)  
  
#store the indices of the missing v7 data  
(missing=which(databc$V7=="?",arr.ind=T))

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

#Mean/mode Imputation  
# creat a function to find the mode of a vector  
getmode=function(v){  
 uniqv=unique(v)  
 uniqv[which.max(tabulate(match(v,uniqv)))]  
}  
#find the mode of v7  
(mode\_v7=as.numeric(getmode(databc[-missing,"V7"])))

## [1] 1

#mean of v7  
(mean\_v7=mean(as.numeric(getmode(databc[-missing,"V7"]))))

## [1] 1

#impute v7 for observations with missing data for v7 to mode\_v7.  
modeimp=databc  
modeimp[missing,]$V7=mode\_v7  
modeimp$V7=as.integer(modeimp$V7)  
#check if there's still missing data in V7  
modeimp[which(modeimp$V7=="?"),]

## [1] V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11  
## <0 rows> (or 0-length row.names)

head(modeimp)

## 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

#regressiong imputation  
datare=databc[-missing,2:10]  
datare$V7=as.integer(datare$V7)  
#generate linear model using all other variables as predictors  
modere=lm(V7~., data=datare)  
summary(modere)

##   
## Call:  
## lm(formula = V7 ~ ., data = datare)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -9.7316 -0.9426 -0.3002 0.6725 8.6998   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -0.616652 0.194975 -3.163 0.00163 \*\*   
## V2 0.230156 0.041691 5.521 4.83e-08 \*\*\*  
## V3 -0.067980 0.076170 -0.892 0.37246   
## V4 0.340442 0.073420 4.637 4.25e-06 \*\*\*  
## V5 0.339705 0.045919 7.398 4.13e-13 \*\*\*  
## V6 0.090392 0.062541 1.445 0.14883   
## V8 0.320577 0.059047 5.429 7.91e-08 \*\*\*  
## V9 0.007293 0.044486 0.164 0.86983   
## V10 -0.075230 0.059331 -1.268 0.20524   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2.274 on 674 degrees of freedom  
## Multiple R-squared: 0.615, Adjusted R-squared: 0.6104   
## F-statistic: 134.6 on 8 and 674 DF, p-value: < 2.2e-16

#use step () function to find a better reduced model  
step(modere)

## Start: AIC=1131.43  
## V7 ~ V2 + V3 + V4 + V5 + V6 + V8 + V9 + V10  
##   
## Df Sum of Sq RSS AIC  
## - V9 1 0.139 3486.8 1129.5  
## - V3 1 4.120 3490.8 1130.2  
## - V10 1 8.317 3495.0 1131.0  
## <none> 3486.6 1131.4  
## - V6 1 10.806 3497.5 1131.5  
## - V4 1 111.227 3597.9 1150.9  
## - V8 1 152.482 3639.1 1158.7  
## - V2 1 157.657 3644.3 1159.6  
## - V5 1 283.119 3769.8 1182.8  
##   
## Step: AIC=1129.45  
## V7 ~ V2 + V3 + V4 + V5 + V6 + V8 + V10  
##   
## Df Sum of Sq RSS AIC  
## - V3 1 4.028 3490.8 1128.2  
## - V10 1 8.179 3495.0 1129.0  
## <none> 3486.8 1129.5  
## - V6 1 11.211 3498.0 1129.7  
## - V4 1 114.768 3601.6 1149.6  
## - V2 1 158.696 3645.5 1157.8  
## - V8 1 160.776 3647.6 1158.2  
## - V5 1 285.902 3772.7 1181.3  
##   
## Step: AIC=1128.24  
## V7 ~ V2 + V4 + V5 + V6 + V8 + V10  
##   
## Df Sum of Sq RSS AIC  
## - V6 1 8.606 3499.4 1127.9  
## - V10 1 8.889 3499.7 1128.0  
## <none> 3490.8 1128.2  
## - V4 1 153.078 3643.9 1155.6  
## - V2 1 155.308 3646.1 1156.0  
## - V8 1 157.123 3647.9 1156.3  
## - V5 1 282.133 3772.9 1179.3  
##   
## Step: AIC=1127.92  
## V7 ~ V2 + V4 + V5 + V8 + V10  
##   
## Df Sum of Sq RSS AIC  
## - V10 1 5.562 3505.0 1127.0  
## <none> 3499.4 1127.9  
## - V2 1 159.594 3659.0 1156.4  
## - V8 1 169.954 3669.4 1158.3  
## - V4 1 206.785 3706.2 1165.1  
## - V5 1 295.807 3795.2 1181.3  
##   
## Step: AIC=1127.01  
## V7 ~ V2 + V4 + V5 + V8  
##   
## Df Sum of Sq RSS AIC  
## <none> 3505.0 1127.0  
## - V2 1 155.70 3660.7 1154.7  
## - V8 1 172.42 3677.4 1157.8  
## - V4 1 201.22 3706.2 1163.1  
## - V5 1 290.68 3795.7 1179.4

##   
## Call:  
## lm(formula = V7 ~ V2 + V4 + V5 + V8, data = datare)  
##   
## Coefficients:  
## (Intercept) V2 V4 V5 V8   
## -0.5360 0.2262 0.3173 0.3323 0.3238

moderef=lm(formula = V7 ~ V2 + V4 + V5 + V8, data = datare)  
summary(moderef)

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

#get prediction for missing v7 values  
(V7hat=predict(moderef,newdata=databc[missing,]))

## 24 41 140 146 159 165 236   
## 5.4585352 7.9816106 0.9872832 1.6218560 0.9807851 2.2157441 2.7152652   
## 250 276 293 295 298 316 322   
## 1.7634059 2.0741942 6.0866099 0.9872832 2.5265324 5.2438347 1.7634059   
## 412 618   
## 0.9872832 0.6634986

#impute v7 for the observations with missing data for v7  
#values with this linear model,#round the v7hat since the orinial are all interger  
datareimp=databc  
datareimp[missing,]$V7=round(V7hat)  
datareimp$V7=as.integer(datareimp$V7)  
datareimp[missing,]$V7

## [1] 5 8 1 2 1 2 3 2 2 6 1 3 5 2 1 1

datareimp[which(datareimp$V7=="?"),]

## [1] V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11  
## <0 rows> (or 0-length row.names)

head(datareimp)

## 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

#make sue no v7 values are out of the original range  
datareimp$V7[datareimp$V7>10]=10  
datareimp$V7[datareimp$V7<1]=1  
datareimp$V7

## [1] 1 10 2 4 1 10 10 1 1 1 1 1 3 3 9 1 1 1 10 1 10 7 1  
## [24] 5 1 7 1 1 1 1 1 1 5 1 1 1 1 1 10 7 8 3 10 1 1 1  
## [47] 9 1 1 8 3 4 5 8 8 5 6 1 10 2 3 2 8 2 1 2 1 10 9  
## [70] 1 1 2 1 10 4 2 1 1 3 1 1 1 1 2 9 4 8 10 1 1 1 1  
## [93] 1 1 1 1 1 1 6 10 5 5 1 3 1 3 10 10 1 9 2 9 10 8 3  
## [116] 5 2 10 3 2 1 2 10 10 7 1 10 1 10 1 1 1 10 1 1 2 1 1  
## [139] 1 1 1 1 5 5 1 2 8 2 1 10 1 10 5 3 1 10 1 1 1 10 10  
## [162] 1 1 3 2 2 10 1 1 1 1 1 1 10 10 10 1 1 1 10 1 1 1 10  
## [185] 10 1 8 10 8 1 8 10 1 1 1 1 7 1 1 1 10 10 1 1 1 10 5  
## [208] 1 1 1 10 8 1 10 10 5 1 1 4 1 1 10 5 8 10 1 10 5 1 10  
## [231] 7 8 1 10 1 3 10 2 9 10 2 1 1 5 1 2 10 9 1 2 1 10 10  
## [254] 10 8 10 1 1 1 8 10 10 10 10 3 1 10 10 4 1 10 1 10 4 1 2  
## [277] 1 1 1 7 1 1 10 10 10 10 10 1 5 10 1 1 6 10 1 10 5 3 1  
## [300] 10 4 1 10 1 10 10 1 1 3 5 1 1 1 1 1 5 10 8 1 5 10 2  
## [323] 1 10 1 1 10 1 4 10 8 1 1 10 10 1 10 1 1 10 10 1 1 1 10  
## [346] 1 1 1 1 8 1 1 3 10 1 1 3 10 4 7 10 10 3 3 1 1 10 10  
## [369] 1 1 1 1 1 1 1 1 1 1 1 1 1 10 1 1 1 1 10 1 1 2 1  
## [392] 10 1 1 1 1 1 1 1 1 9 1 1 4 1 1 1 1 2 1 1 1 4 1  
## [415] 10 3 10 1 2 1 3 10 1 1 1 10 1 2 1 1 1 1 1 1 8 10 1  
## [438] 1 1 1 10 4 3 2 1 1 1 1 1 10 1 1 1 10 1 6 10 3 1 1  
## [461] 1 5 1 1 1 4 10 10 1 1 1 1 1 1 1 1 1 1 1 10 1 1 5  
## [484] 10 1 3 1 10 3 4 1 10 1 10 5 1 1 1 1 1 1 1 1 1 1 1  
## [507] 5 4 1 1 1 1 1 1 10 10 1 1 1 10 1 1 5 10 1 1 1 1 1  
## [530] 1 10 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 10 1 1 5 1 1  
## [553] 1 5 1 1 1 1 1 1 1 1 1 1 1 10 1 3 10 5 10 10 1 1 2  
## [576] 1 1 1 1 1 1 10 10 1 1 1 10 1 3 1 1 10 10 1 10 1 1 1  
## [599] 1 1 1 1 1 1 10 8 1 1 10 1 10 2 10 1 1 1 1 1 1 1 1  
## [622] 2 1 1 1 4 6 5 1 1 1 1 1 3 1 1 1 2 1 1 1 1 1 1  
## [645] 1 1 1 1 2 1 4 1 1 1 1 1 1 1 10 1 1 1 1 1 1 1 1  
## [668] 1 1 5 8 1 1 1 1 1 1 1 1 1 10 10 1 1 1 1 1 1 1 1  
## [691] 1 5 1 1 2 1 3 4 5

head(datareimp)

## 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

#regression with perturbation imputation (add error term to the predicted value)  
(v7pert=rnorm(nrow(databc[missing,]),V7hat,sd(V7hat)))

## [1] 3.605143 11.032979 -1.780039 1.776463 4.753098 0.886830 1.674528  
## [8] 0.362937 1.444299 6.391024 3.693194 0.759271 2.862462 1.416173  
## [15] -1.375062 0.357149

#negative value  
#values with this perturbation imputation,round the v7pert since the orinial are all interger  
datapertimp=databc  
datapertimp[missing,]$V7=round(v7pert)  
datapertimp$V7=as.integer(datapertimp$V7)  
datapertimp[missing,]$V7

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

#make sue no v7 values are out of the original range  
datapertimp$V7[datapertimp$V7>10]=10  
datapertimp$V7[datapertimp$V7<1]=1  
datapertimp$V7

## [1] 1 10 2 4 1 10 10 1 1 1 1 1 3 3 9 1 1 1 10 1 10 7 1  
## [24] 4 1 7 1 1 1 1 1 1 5 1 1 1 1 1 10 7 10 3 10 1 1 1  
## [47] 9 1 1 8 3 4 5 8 8 5 6 1 10 2 3 2 8 2 1 2 1 10 9  
## [70] 1 1 2 1 10 4 2 1 1 3 1 1 1 1 2 9 4 8 10 1 1 1 1  
## [93] 1 1 1 1 1 1 6 10 5 5 1 3 1 3 10 10 1 9 2 9 10 8 3  
## [116] 5 2 10 3 2 1 2 10 10 7 1 10 1 10 1 1 1 10 1 1 2 1 1  
## [139] 1 1 1 1 5 5 1 2 8 2 1 10 1 10 5 3 1 10 1 1 5 10 10  
## [162] 1 1 3 1 2 10 1 1 1 1 1 1 10 10 10 1 1 1 10 1 1 1 10  
## [185] 10 1 8 10 8 1 8 10 1 1 1 1 7 1 1 1 10 10 1 1 1 10 5  
## [208] 1 1 1 10 8 1 10 10 5 1 1 4 1 1 10 5 8 10 1 10 5 1 10  
## [231] 7 8 1 10 1 2 10 2 9 10 2 1 1 5 1 2 10 9 1 1 1 10 10  
## [254] 10 8 10 1 1 1 8 10 10 10 10 3 1 10 10 4 1 10 1 10 4 1 1  
## [277] 1 1 1 7 1 1 10 10 10 10 10 1 5 10 1 1 6 10 4 10 5 1 1  
## [300] 10 4 1 10 1 10 10 1 1 3 5 1 1 1 1 1 3 10 8 1 5 10 1  
## [323] 1 10 1 1 10 1 4 10 8 1 1 10 10 1 10 1 1 10 10 1 1 1 10  
## [346] 1 1 1 1 8 1 1 3 10 1 1 3 10 4 7 10 10 3 3 1 1 10 10  
## [369] 1 1 1 1 1 1 1 1 1 1 1 1 1 10 1 1 1 1 10 1 1 2 1  
## [392] 10 1 1 1 1 1 1 1 1 9 1 1 4 1 1 1 1 2 1 1 1 4 1  
## [415] 10 3 10 1 2 1 3 10 1 1 1 10 1 2 1 1 1 1 1 1 8 10 1  
## [438] 1 1 1 10 4 3 2 1 1 1 1 1 10 1 1 1 10 1 6 10 3 1 1  
## [461] 1 5 1 1 1 4 10 10 1 1 1 1 1 1 1 1 1 1 1 10 1 1 5  
## [484] 10 1 3 1 10 3 4 1 10 1 10 5 1 1 1 1 1 1 1 1 1 1 1  
## [507] 5 4 1 1 1 1 1 1 10 10 1 1 1 10 1 1 5 10 1 1 1 1 1  
## [530] 1 10 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 10 1 1 5 1 1  
## [553] 1 5 1 1 1 1 1 1 1 1 1 1 1 10 1 3 10 5 10 10 1 1 2  
## [576] 1 1 1 1 1 1 10 10 1 1 1 10 1 3 1 1 10 10 1 10 1 1 1  
## [599] 1 1 1 1 1 1 10 8 1 1 10 1 10 2 10 1 1 1 1 1 1 1 1  
## [622] 2 1 1 1 4 6 5 1 1 1 1 1 3 1 1 1 2 1 1 1 1 1 1  
## [645] 1 1 1 1 2 1 4 1 1 1 1 1 1 1 10 1 1 1 1 1 1 1 1  
## [668] 1 1 5 8 1 1 1 1 1 1 1 1 1 10 10 1 1 1 1 1 1 1 1  
## [691] 1 5 1 1 2 1 3 4 5

head(datapertimp)

## 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 if the missing data has been replaced successfully  
datapertimp[which(datapertimp$V7=="?"),]

## [1] V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11  
## <0 rows> (or 0-length row.names)

# svm on Mean/mode Imputation data :  
head(modeimp)

## 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

library(kernlab)

## Warning: package 'kernlab' was built under R version 3.4.4

databcm=modeimp  
CCtrain\_ind=sample(nrow(databcm),size=floor(nrow(databcm)\*0.7))  
CCtrain=databcm[CCtrain\_ind,]  
CCother=databcm[-CCtrain\_ind,]  
CCtv\_ind=sample(nrow(CCother),size=floor(nrow(CCother)/2))  
CCtest=CCother[CCtv\_ind,]  
CCvalid=CCother[-CCtv\_ind,]  
nrow(CCvalid)

## [1] 105

nrow(CCtest)

## [1] 105

nrow(CCtrain)

## [1] 489

acc=rep(0,9)   
amounts=c(0.00001,0.0001,0.001,0.01,0.1,1,10,100,1000)  
for(i in 1:9){  
 modelsvm=ksvm(V11 ~ V2+V3+V4+V5+V6+V7+V8+V9+V10,   
 data=CCtrain,   
 type="C-svc",   
 kernel= "vanilladot",  
 C=amounts[i],   
 scaled =T)  
 predsvm <- predict(modelsvm,CCvalid[,2:10])  
 acc[i]=sum(predsvm == CCvalid[,11]) / nrow(CCvalid)  
}

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

acc[1:9]

## [1] 0.6190476 0.6190476 0.9523810 0.9619048 0.9523810 0.9333333 0.9333333  
## [8] 0.9333333 0.9333333

## find best svm model  
cat("Best SVM model is number", which.max(acc[1:9]),"\n")

## Best SVM model is number 4

cat("Best C value is", amounts[which.max(acc[1:9])],"\n")

## Best C value is 0.01

cat("Best validation set accuracy is", max(acc[1:9]),"\n")

## Best validation set accuracy is 0.9619048

##restrain the best model  
modelsvmb=ksvm(V11 ~ V2+V3+V4+V5+V6+V7+V8+V9+V10,   
 data=CCtrain,   
 type="C-svc",   
 kernel= "vanilladot",  
 C= 1,   
 scaled =T)

## Setting default kernel parameters

#test model in test dataset  
cat("accuracy on test data =",sum(predict(modelsvmb,CCtest[,1:10])== CCtest[,11]) / nrow(CCtest),"\n")

## accuracy on test data = 1

# svm on regressing Imputation data :  
databcm=datareimp  
CCtrain\_ind=sample(nrow(databcm),size=floor(nrow(databcm)\*0.7))  
CCtrain=databcm[CCtrain\_ind,]  
CCother=databcm[-CCtrain\_ind,]  
CCtv\_ind=sample(nrow(CCother),size=floor(nrow(CCother)/2))  
CCtest=CCother[CCtv\_ind,]  
CCvalid=CCother[-CCtv\_ind,]  
nrow(CCvalid)

## [1] 105

nrow(CCtest)

## [1] 105

nrow(CCtrain)

## [1] 489

acc=rep(0,9)   
amounts=c(0.00001,0.0001,0.001,0.01,0.1,1,10,100,1000)  
for(i in 1:9){  
 modelsvm=ksvm(V11 ~ V2+V3+V4+V5+V6+V7+V8+V9+V10,   
 data=CCtrain,   
 type="C-svc",   
 kernel= "vanilladot",  
 C=amounts[i],   
 scaled =T)  
 predsvm <- predict(modelsvm,CCvalid[,2:10])  
 acc[i]=sum(predsvm == CCvalid[,11]) / nrow(CCvalid)  
}

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

acc[1:9]

## [1] 0.6190476 0.6190476 0.9523810 0.9714286 0.9619048 0.9714286 0.9714286  
## [8] 0.9714286 0.9714286

## find best svm model  
cat("Best SVM model is number", which.max(acc[1:9]),"\n")

## Best SVM model is number 4

cat("Best C value is", amounts[which.max(acc[1:9])],"\n")

## Best C value is 0.01

cat("Best validation set accuracy is", max(acc[1:9]),"\n")

## Best validation set accuracy is 0.9714286

##restrain the best model  
modelsvmb=ksvm(V11 ~ V2+V3+V4+V5+V6+V7+V8+V9+V10,   
 data=CCtrain,   
 type="C-svc",   
 kernel= "vanilladot",  
 C= 0.01,   
 scaled =T)

## Setting default kernel parameters

#test model in test dataset  
cat("accuracy on test data =",sum(predict(modelsvmb,CCtest[,1:10])== CCtest[,11]) / nrow(CCtest),"\n")

## accuracy on test data = 0.9619048

# svm on regression with perturbation imputation data :  
  
databcm=datapertimp  
CCtrain\_ind=sample(nrow(databcm),size=floor(nrow(databcm)\*0.7))  
CCtrain=databcm[CCtrain\_ind,]  
CCother=databcm[-CCtrain\_ind,]  
CCtv\_ind=sample(nrow(CCother),size=floor(nrow(CCother)/2))  
CCtest=CCother[CCtv\_ind,]  
CCvalid=CCother[-CCtv\_ind,]  
nrow(CCvalid)

## [1] 105

nrow(CCtest)

## [1] 105

nrow(CCtrain)

## [1] 489

acc=rep(0,9)   
amounts=c(0.00001,0.0001,0.001,0.01,0.1,1,10,100,1000)  
for(i in 1:9){  
 modelsvm=ksvm(V11 ~ V2+V3+V4+V5+V6+V7+V8+V9+V10,   
 data=CCtrain,   
 type="C-svc",   
 kernel= "vanilladot",  
 C=amounts[i],   
 scaled =T)  
 predsvm <- predict(modelsvm,CCvalid[,2:10])  
 acc[i]=sum(predsvm == CCvalid[,11]) / nrow(CCvalid)  
}

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

acc[1:9]

## [1] 0.6285714 0.6285714 0.9523810 0.9619048 0.9714286 0.9714286 0.9714286  
## [8] 0.9714286 0.9714286

## find best svm model  
cat("Best SVM model is number", which.max(acc[1:9]),"\n")

## Best SVM model is number 5

cat("Best C value is", amounts[which.max(acc[1:9])],"\n")

## Best C value is 0.1

cat("Best validation set accuracy is", max(acc[1:9]),"\n")

## Best validation set accuracy is 0.9714286

##restrain the best model  
modelsvmb=ksvm(V11 ~ V2+V3+V4+V5+V6+V7+V8+V9+V10,   
 data=CCtrain,   
 type="C-svc",   
 kernel= "vanilladot",  
 C= 0.01,   
 scaled =T)

## Setting default kernel parameters

#test model in test dataset  
cat("accuracy on test data =",sum(predict(modelsvmb,CCtest[,1:10])== CCtest[,11]) / nrow(CCtest),"\n")

## accuracy on test data = 0.9809524

# remove missing data :  
datan=as.data.frame(sapply(databc,sub,pattern='\\?',replacement=NA))  
datana <- na.omit(datan)  
databcm=datana  
head(databcm)

## 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

CCtrain\_ind=sample(nrow(databcm),size=floor(nrow(databcm)\*0.7))  
CCtrain=databcm[CCtrain\_ind,]  
CCother=databcm[-CCtrain\_ind,]  
CCtv\_ind=sample(nrow(CCother),size=floor(nrow(CCother)/2))  
CCtest=CCother[CCtv\_ind,]  
CCvalid=CCother[-CCtv\_ind,]  
nrow(CCvalid)

## [1] 103

head(CCvalid)

## V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11  
## 5 1017023 4 1 1 3 2 1 3 1 1 2  
## 12 1036172 2 1 1 1 2 1 2 1 1 2  
## 20 1050718 6 1 1 1 2 1 3 1 1 2  
## 22 1054593 10 5 5 3 6 7 7 10 1 4  
## 27 1066373 3 2 1 1 1 1 2 1 1 2  
## 39 1084584 5 4 4 9 2 10 5 6 1 4

nrow(CCtest)

## [1] 102

head(CCtest)

## V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11  
## 64 1116132 6 3 4 1 5 2 3 9 1 4  
## 374 521441 5 1 1 2 2 1 2 1 1 2  
## 267 1198641 10 10 6 3 3 10 4 3 2 4  
## 570 1343374 10 10 8 10 6 5 10 3 1 4  
## 283 488173 1 4 3 10 4 10 5 6 1 4  
## 685 466906 1 1 1 1 2 1 1 1 1 2

nrow(CCtrain)

## [1] 478

head(CCtrain)

## V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11  
## 525 749653 3 1 1 1 2 1 2 1 1 2  
## 33 1072179 10 7 7 3 8 5 7 4 3 4  
## 494 1142706 5 10 10 10 6 10 6 5 2 4  
## 212 1219859 8 10 8 8 4 8 7 7 1 4  
## 663 13454352 1 1 3 1 2 1 2 1 1 2  
## 516 1313325 4 10 4 7 3 10 9 10 1 4

acc=rep(0,9)   
amounts=c(0.00001,0.0001,0.001,0.01,0.1,1,10,100,1000)  
for(i in 1:9){  
 modelsvm=ksvm(V11 ~ V2+V3+V4+V5+V6+V7+V8+V9+V10,   
 data=CCtrain,   
 type="C-svc",   
 kernel= "vanilladot",  
 C=amounts[i],   
 scaled =T)  
 predsvm <- predict(modelsvm,CCvalid[,2:10])  
 acc[i]=sum(predsvm == CCvalid[,11]) / nrow(CCvalid)  
}

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

acc[1:9]

## [1] 0.6116505 0.6116505 0.6116505 0.9514563 0.9611650 0.9611650 0.9514563  
## [8] 0.9514563 0.9514563

## find best svm model  
cat("Best SVM model is number", which.max(acc[1:9]),"\n")

## Best SVM model is number 5

cat("Best C value is", amounts[which.max(acc[1:9])],"\n")

## Best C value is 0.1

cat("Best validation set accuracy is", max(acc[1:9]),"\n")

## Best validation set accuracy is 0.961165

##restrain the best model  
modelsvmb=ksvm(V11 ~ V2+V3+V4+V5+V6+V7+V8+V9+V10,   
 data=CCtrain,   
 type="C-svc",   
 kernel= "vanilladot",  
 C= 1,   
 scaled =T)

## Setting default kernel parameters

#test model in test dataset  
cat("accuracy on test data =",sum(predict(modelsvmb,CCtest[,1:10])== CCtest[,11]) / nrow(CCtest),"\n")

## accuracy on test data = 0.9411765

#binary variables  
library(tidyverse)

## Warning: package 'tidyverse' was built under R version 3.4.4

## -- Attaching packages ------------------------------------------------------------- tidyverse 1.2.1 --

## v ggplot2 2.2.1 v purrr 0.2.5  
## v tibble 1.4.2 v dplyr 0.7.5  
## v tidyr 0.8.1 v stringr 1.3.1  
## v readr 1.1.1 v forcats 0.3.0

## Warning: package 'ggplot2' was built under R version 3.4.4

## Warning: package 'tibble' was built under R version 3.4.4

## Warning: package 'tidyr' was built under R version 3.4.4

## Warning: package 'readr' was built under R version 3.4.4

## Warning: package 'purrr' was built under R version 3.4.4

## Warning: package 'dplyr' was built under R version 3.4.4

## Warning: package 'stringr' was built under R version 3.4.4

## Warning: package 'forcats' was built under R version 3.4.4

## -- Conflicts ---------------------------------------------------------------- tidyverse\_conflicts() --  
## x ggplot2::alpha() masks kernlab::alpha()  
## x purrr::cross() masks kernlab::cross()  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

databinn <- databc %>%   
 mutate(bv7 = ifelse(databc$V7 == "?",0,1))

## Warning: package 'bindrcpp' was built under R version 3.4.4

head(databinn)

## V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 bv7  
## 1 1000025 5 1 1 1 2 1 3 1 1 2 1  
## 2 1002945 5 4 4 5 7 10 3 2 1 2 1  
## 3 1015425 3 1 1 1 2 2 3 1 1 2 1  
## 4 1016277 6 8 8 1 3 4 3 7 1 2 1  
## 5 1017023 4 1 1 3 2 1 3 1 1 2 1  
## 6 1017122 8 10 10 8 7 10 9 7 1 4 1

databinn[which(databinn$bv7==0),]

## V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 bv7  
## 24 1057013 8 4 5 1 2 ? 7 3 1 4 0  
## 41 1096800 6 6 6 9 6 ? 7 8 1 2 0  
## 140 1183246 1 1 1 1 1 ? 2 1 1 2 0  
## 146 1184840 1 1 3 1 2 ? 2 1 1 2 0  
## 159 1193683 1 1 2 1 3 ? 1 1 1 2 0  
## 165 1197510 5 1 1 1 2 ? 3 1 1 2 0  
## 236 1241232 3 1 4 1 2 ? 3 1 1 2 0  
## 250 169356 3 1 1 1 2 ? 3 1 1 2 0  
## 276 432809 3 1 3 1 2 ? 2 1 1 2 0  
## 293 563649 8 8 8 1 2 ? 6 10 1 4 0  
## 295 606140 1 1 1 1 2 ? 2 1 1 2 0  
## 298 61634 5 4 3 1 2 ? 2 3 1 2 0  
## 316 704168 4 6 5 6 7 ? 4 9 1 2 0  
## 322 733639 3 1 1 1 2 ? 3 1 1 2 0  
## 412 1238464 1 1 1 1 1 ? 2 1 1 2 0  
## 618 1057067 1 1 1 1 1 ? 1 1 1 2 0

databin1=databinn[,-7]  
databin2=databin1[,-1]  
head(databin1)

## V1 V2 V3 V4 V5 V6 V8 V9 V10 V11 bv7  
## 1 1000025 5 1 1 1 2 3 1 1 2 1  
## 2 1002945 5 4 4 5 7 3 2 1 2 1  
## 3 1015425 3 1 1 1 2 3 1 1 2 1  
## 4 1016277 6 8 8 1 3 3 7 1 2 1  
## 5 1017023 4 1 1 3 2 3 1 1 2 1  
## 6 1017122 8 10 10 8 7 9 7 1 4 1

databin=databin1[c(1,2,3,4,5,6,7,8,9,11,10)]  
head(databin)

## V1 V2 V3 V4 V5 V6 V8 V9 V10 bv7 V11  
## 1 1000025 5 1 1 1 2 3 1 1 1 2  
## 2 1002945 5 4 4 5 7 3 2 1 1 2  
## 3 1015425 3 1 1 1 2 3 1 1 1 2  
## 4 1016277 6 8 8 1 3 3 7 1 1 2  
## 5 1017023 4 1 1 3 2 3 1 1 1 2  
## 6 1017122 8 10 10 8 7 9 7 1 1 4

databcm=databin  
CCtrain\_ind=sample(nrow(databcm),size=floor(nrow(databcm)\*0.7))  
CCtrain=databcm[CCtrain\_ind,]  
CCother=databcm[-CCtrain\_ind,]  
CCtv\_ind=sample(nrow(CCother),size=floor(nrow(CCother)/2))  
CCtest=CCother[CCtv\_ind,]  
CCvalid=CCother[-CCtv\_ind,]  
nrow(CCvalid)

## [1] 105

head(CCvalid)

## V1 V2 V3 V4 V5 V6 V8 V9 V10 bv7 V11  
## 3 1015425 3 1 1 1 2 3 1 1 1 2  
## 16 1047630 7 4 6 4 6 4 3 1 1 4  
## 21 1054590 7 3 2 10 5 5 4 4 1 4  
## 23 1056784 3 1 1 1 2 2 1 1 1 2  
## 32 1071760 2 1 1 1 2 3 1 1 1 2  
## 39 1084584 5 4 4 9 2 5 6 1 1 4

nrow(CCtest)

## [1] 105

head(CCtest)

## V1 V2 V3 V4 V5 V6 V8 V9 V10 bv7 V11  
## 227 1227481 10 5 7 4 4 8 9 1 1 4  
## 181 1203096 1 1 1 1 1 3 1 1 1 2  
## 192 1211202 7 5 10 10 10 4 10 3 1 4  
## 389 1114570 2 1 1 1 2 2 2 1 1 2  
## 580 824249 1 1 1 1 2 3 1 1 1 2  
## 316 704168 4 6 5 6 7 4 9 1 0 2

nrow(CCtrain)

## [1] 489

head(CCtrain)

## V1 V2 V3 V4 V5 V6 V8 V9 V10 bv7 V11  
## 67 1117152 4 1 1 1 2 3 1 1 1 2  
## 48 1105524 1 1 1 1 2 2 1 1 1 2  
## 492 1119189 5 8 9 4 3 7 1 1 1 4  
## 417 1239347 8 7 8 5 10 7 2 1 1 4  
## 216 1222936 8 7 8 7 5 5 10 2 1 4  
## 621 1083817 3 1 1 1 2 2 1 1 1 2

library(kernlab)  
## svm  
acc=rep(0,9)   
## svm training  
amounts=c(0.00001,0.0001,0.001,0.01,0.1,1,10,100,1000)  
for(i in 1:9){  
 modelsvm=ksvm(V11 ~ V2+V3+V4+V5+V6+V8+V9+V10+bv7,   
 data=CCtrain,   
 type="C-svc",   
 kernel= "vanilladot",  
 C=amounts[i],   
 scaled =T)  
 predsvm <- predict(modelsvm,CCvalid[,2:10])  
 acc[i]=sum(predsvm == CCvalid[,11]) / nrow(CCvalid)  
}

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

acc[1:9]

## [1] 0.6476190 0.6476190 0.9523810 0.9714286 0.9809524 0.9714286 0.9714286  
## [8] 0.9714286 0.9714286

## find best svm model  
cat("Best SVM model is number", which.max(acc[1:9]),"\n")

## Best SVM model is number 5

cat("Best C value is", amounts[which.max(acc[1:9])],"\n")

## Best C value is 0.1

cat("Best validation set accuracy is", max(acc[1:9]),"\n")

## Best validation set accuracy is 0.9809524

##restrain the best model  
modelsvmb=ksvm(V11 ~ V2+V3+V4+V5+V6+V8+V9+V10+bv7,   
 data=CCtrain,   
 type="C-svc",   
 kernel= "vanilladot",  
 C= 0.01,   
 scaled =T)

## Setting default kernel parameters

#test model in test dataset  
cat("accuracy on test data =",sum(predict(modelsvmb,CCtest[,1:10])== CCtest[,11]) / nrow(CCtest),"\n")

## accuracy on test data = 0.9333333