Project02

Isaiah Thompson Ocansey

2022-09-26

(1) Bring in the data. Remove the first three columns, which are ID variables. Change the value 0 to -1 for Class since we will experiment with a logistic model with ± 1 valued responses

```
df<- read.csv("Shill _Bidding_Dataset.csv")
head(df);dim(df)
```

```
  Record_ID Auction_ID Bidder_ID Bidder_Tendency Bidding_Ratio
1         1        732     _***i      0.20000000     0.4000000
2         2        732     g***r      0.02439024     0.2000000
3         3        732     t***p      0.14285714     0.2000000
4         4        732     7***n      0.10000000     0.2000000
5         5        900     z***z      0.05128205     0.2222222
6         8        900     i***e      0.03846154     0.1111111
  Successive_Outbidding Last_Bidding Auction_Bids Starting_Price_Average
1                     0 0.0000277778            0              0.9935928
2                     0 0.0131226852            0              0.9935928
3                     0 0.0030416667            0              0.9935928
4                     0 0.0974768519            0              0.9935928
5                     0 0.0013177910            0              0.0000000
6                     0 0.0168435847            0              0.0000000
  Early_Bidding Winning_Ratio Auction_Duration Class
1  0.0000277778     0.6666667                5     0
2  0.0131226852     0.9444444                5     0
3  0.0030416667     1.0000000                5     0
4  0.0974768519     1.0000000                5     0
5  0.0012417328     0.5000000                7     0
6  0.0168435847     0.8000000                7     0
```

```
[1] 6321   13
```

The data has 6321 observations and 13 variables

```
dat<-df[,-c(1:3)]
head(dat);dim(dat)
```

```
  Bidder_Tendency Bidding_Ratio Successive_Outbidding Last_Bidding Auction_Bids
1      0.20000000     0.4000000                     0 0.0000277778            0
2      0.02439024     0.2000000                     0 0.0131226852            0
3      0.14285714     0.2000000                     0 0.0030416667            0
4      0.10000000     0.2000000                     0 0.0974768519            0
```

```
5      0.05128205     0.2222222                    0 0.0013177910            0
6      0.03846154     0.1111111                    0 0.0168435847            0
  Starting_Price_Average Early_Bidding Winning_Ratio Auction_Duration Class
1              0.9935928  0.0000277778     0.6666667                5     0
2              0.9935928  0.0131226852     0.9444444                5     0
3              0.9935928  0.0030416667     1.0000000                5     0
4              0.9935928  0.0974768519     1.0000000                5     0
5              0.0000000  0.0012417328     0.5000000                7     0
6              0.0000000  0.0168435847     0.8000000                7     0
```

```
[1] 6321   10
```

After removing the first 3 ID variables, the data set has 6321 observations and 10 variables.

```r
dat$Class[dat$Class==0]<--1
```

The code above changes all the 0 values of the Class variable to -1

EXPLORATOTY DATA ANALYSIS (EDA)

(2a) Compute the number of distinct levels or values for each variable. Are there any categorical variable or numerical variable that has only a few distinct values

```r
str(dat)
```

```
'data.frame':   6321 obs. of  10 variables:
 $ Bidder_Tendency       : num  0.2 0.0244 0.1429 0.1 0.0513 ...
 $ Bidding_Ratio         : num  0.4 0.2 0.2 0.2 0.222 ...
 $ Successive_Outbidding : num  0 0 0 0 0 0 0 1 1 0.5 ...
 $ Last_Bidding          : num  2.78e-05 1.31e-02 3.04e-03 9.75e-02 1.32e-03 ...
 $ Auction_Bids          : num  0 0 0 0 0 ...
 $ Starting_Price_Average: num  0.994 0.994 0.994 0.994 0 ...
 $ Early_Bidding         : num  2.78e-05 1.31e-02 3.04e-03 9.75e-02 1.24e-03 ...
 $ Winning_Ratio         : num  0.667 0.944 1 1 0.5 ...
 $ Auction_Duration      : int  5 5 5 5 7 7 7 7 7 7 ...
 $ Class                 : num  -1 -1 -1 -1 -1 -1 -1 1 1 1 ...
```

From the above output,it can be observed that all the variables are numeric except Auction Duration which is an integer

```r
sapply(dat, function(x) length(unique(x)))
```

```
       Bidder_Tendency          Bidding_Ratio  Successive_Outbidding
                   489                    400                      3
          Last_Bidding           Auction_Bids Starting_Price_Average
                  5807                     49                     22
         Early_Bidding          Winning_Ratio       Auction_Duration
                  5690                     72                      5
                 Class
                     2
```

From the above output, the variables; class,successive outbidding and Auction duration have few distinct levels

**HANDLING MISSING VALUES*

(1b) Are there any missing data? If so, deal with them with an imputation or listwise deletion accordingly. Document your steps carefully.

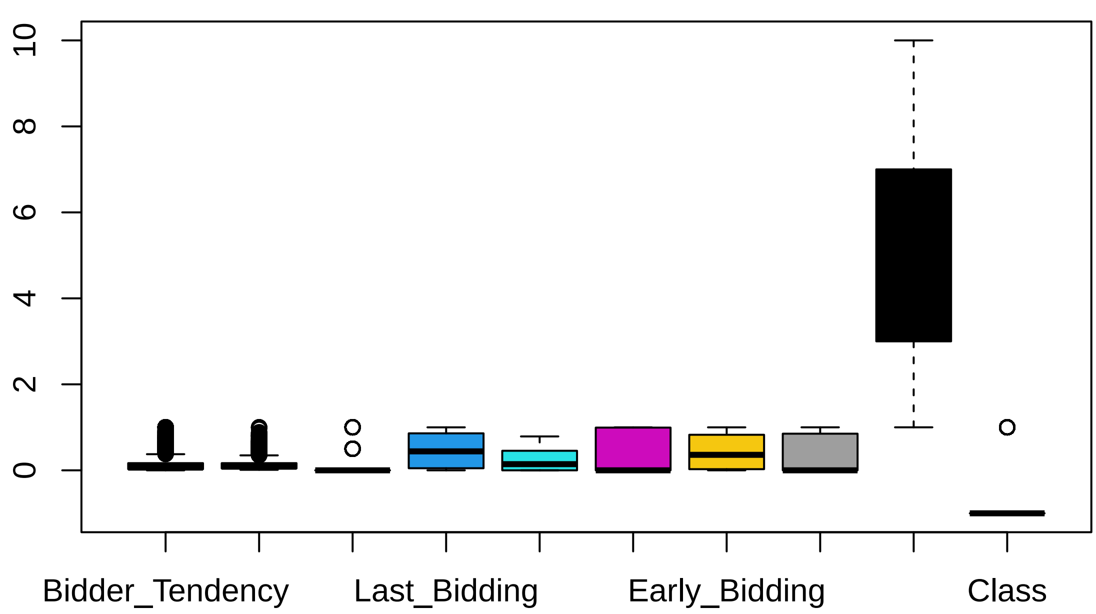
```
colMeans(is.na(dat))
```

```
##       Bidder_Tendency         Bidding_Ratio  Successive_Outbidding
##                     0                     0                      0
##          Last_Bidding          Auction_Bids Starting_Price_Average
##                     0                     0                      0
##         Early_Bidding         Winning_Ratio       Auction_Duration
##                     0                     0                      0
##                 Class
##                     0
```

From the above output, it can be clearly seen that there are no missing values in the data set

(1c) Make a parallel boxplot of the data to view the predictors or attributes in the data. Inspect whether they have the same range and variation. This helps us to determine whether scaling is necessary for some modeling approaches.

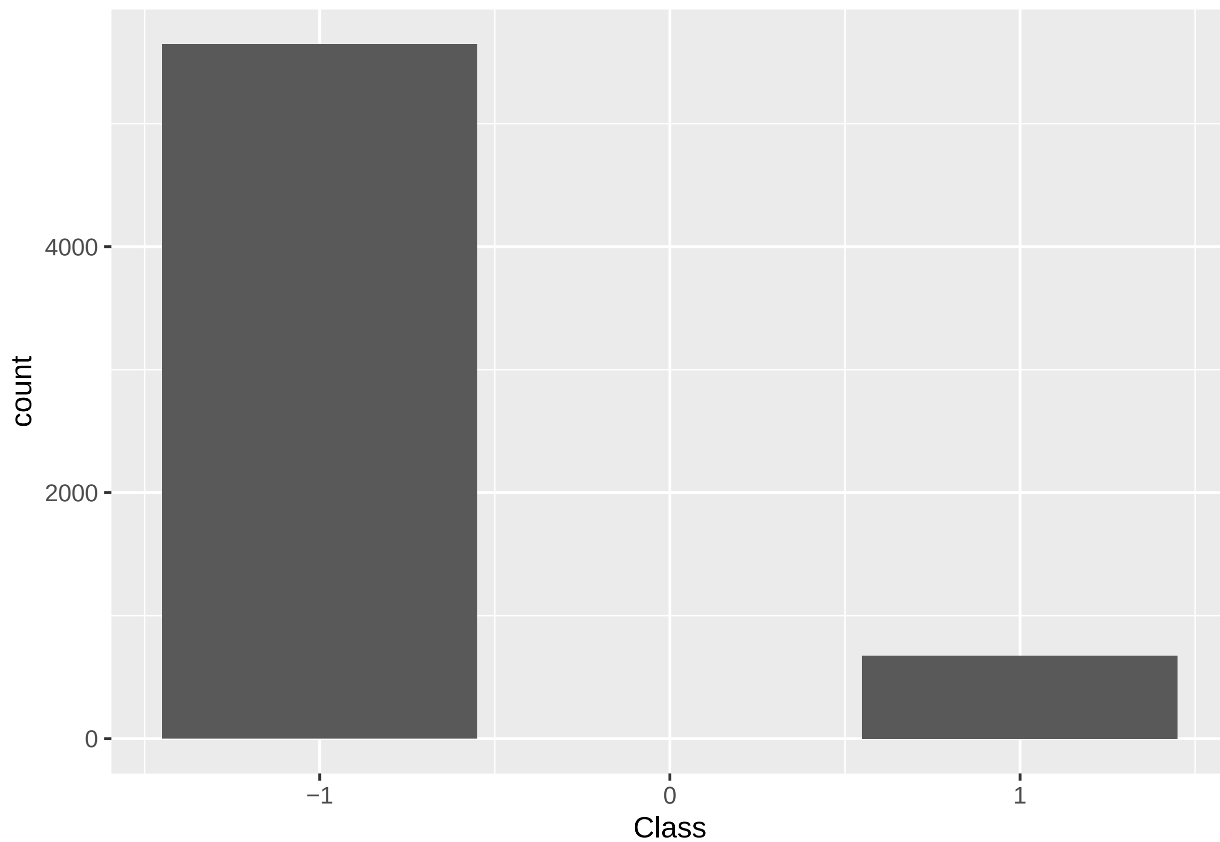
```
boxplot(dat,col = c(1:10))
```



From the boxplots above, there seems to be an unequal variables among the predictors and thus, we need scaling for a good modeling

(1d) Make a bar plot of the binary response Class. Do we seem to have an unbalanced classification problem?

```
library(ggplot2)
ggplot(dat,aes(Class)) + geom_bar()
```



From the bar plot above, the classification problem appears to be unbalanced however, this is negligible.

```
**DATA PARTITION**
```

3) Partition the data D into three sets: the training data D1, the validation data D2, and the test data D3 with a ratio approximately of 2 : 1 : 1.

```
set.seed(123)
n <- NROW(dat)
id.group <- sample(x=1:3,size = n, replace = TRUE, prob = c(2,1,1)/4)
D1 <- dat[id.group==1, ]
D2 <- dat[id.group==2, ]
D3 <- dat[id.group==3, ]
dim(D1); dim(D2); dim(D3)
```

```
[1] 3196   10
```

```
[1] 1552   10
```

```
[1] 1573   10
```

The above code partitions the data into Training set, Validation set and Testing set in the ratio 2:1:1 respectively

LOGISTIC REGRESSION (OPTIMIZATION)

(4a) Pool the training data and the validation data together into D' = D1   D2. Based on D' obtain the maximum likelihood estimates (MLE) b = ˆj of regression parameters and their standard errors from the resultant Hessian matrix. Test the significance of each attribute and obtain the corresponding p-values. Tabulate the results. Also, specify the optimization method that you use in R function optim(), e.g., BFGS. Check to make sure that the algorithm converges by looking at the "convergence" value in the output, which should be 0 if success.

```
df_new<-rbind(D1,D2)
head(df_new);dim(df_new)
```

```
   Bidder_Tendency Bidding_Ratio Successive_Outbidding Last_Bidding
1       0.20000000    0.40000000                   0.0 0.0000277778
3       0.14285714    0.20000000                   0.0 0.0030416667
6       0.03846154    0.11111111                   0.0 0.0168435847
10      0.15517241    0.34615385                   0.5 0.5709937169
12      0.50000000    0.10526316                   0.0 0.0286921296
15      0.14285714    0.04166667                   0.0 0.3873478836
   Auction_Bids Starting_Price_Average Early_Bidding Winning_Ratio
1    0.00000000              0.9935928  0.0000277778     0.6666667
3    0.00000000              0.9935928  0.0030416667     1.0000000
6    0.00000000              0.0000000  0.0168435847     0.8000000
10   0.30769231              0.9935928  0.4137880291     0.6111111
12   0.05263158              0.0000000  0.0286541005     0.6666667
15   0.25000000              0.0000000  0.3873478836     0.0000000
   Auction_Duration Class
1                 5    -1
3                 5    -1
6                 7    -1
10                7     1
12                7    -1
15                7    -1
```

```
[1] 4748   10
```

The above code puts the training data and the validation data together into D'= D1   D2. The new data set D' formed as a result of the union has 4748 observations and 10 variables

THE NEGATIVE LOGLIKEHOOD FUNCTION FOR Y=+1/-1

```
nloglik <- function(beta, X, y){
if (length(unique(y)) !=2) stop("Are you sure you've got Binary Target?")
X <- cbind(1, X)
nloglik <- sum(log(1+ exp(-y*X%*%beta)))
return(nloglik)
}
y <- df_new$Class
X <- as.matrix(df_new[, c(1:9)])
```

```
p <- NCOL(X) +1
fit <- optim(par=rep(0,p), fn=nloglik, method="BFGS",hessian=T, X=X, y=y)
estimate <- fit$par; estimate
```

```
 [1] -12.5668925   0.4687752   2.2361310  11.2729636   1.6467437   0.2098433
 [7]   0.2394736  -1.2255131   6.4092912   0.1884254
```

The above output shows the beta estimates for the negative log-likelihood function using the BFGS optimization method

```
beta.hat<-fit$par
VCOV.est<--fit$hessian
se<-sqrt(abs(diag(VCOV.est)))
z.wald<-beta.hat/se
pvalue<-pchisq(z.wald^2,df=1,lower.tail = FALSE)
result<-data.frame(beta.hat,se, z.wald,pvalue)
round(result,digits = 4)
```

```
   beta.hat      se  z.wald pvalue
1  -12.5669  7.8233 -1.6063 0.1082
2    0.4688  2.5756  0.1820 0.8556
3    2.2361  2.2958  0.9740 0.3300
4   11.2730  3.9037  2.8878 0.0039
5    1.6467  4.8457  0.3398 0.7340
6    0.2098  2.6065  0.0805 0.9358
7    0.2395  5.3127  0.0451 0.9640
8   -1.2255  4.4300 -0.2766 0.7821
9    6.4093  6.6474  0.9642 0.3350
10   0.1884 41.5641  0.0045 0.9964
```

The above output shows the beta estimates, the standard error , the Z-values and their corresponding p-values.For a given alpha=5%, Last_Bidding is the only statistically significant variable. The optimization method used is BFGS.

CONVERGENCE

```
fit$convergence
```

```
[1] 0
```

From the above,the algorithm converges to 0 which is good

COMPARING USING THE STANDARD GLM()

(4b) Compare your results in 4(a) with the fitting results from standard R function glm().

```
dat0<-df_new[,-c(10)]
dat0$y <- ifelse(df_new$Class ==-1,0,1)
fit.logit <- glm(y~., data=dat0, family=binomial(link = "logit"))
summary(fit.logit)
```

```
Call:
glm(formula = y ~ ., family = binomial(link = "logit"), data = dat0)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.0559  -0.0695  -0.0065  -0.0044   2.5626

Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)            -12.56688    0.95061 -13.220  < 2e-16 ***
Bidder_Tendency          0.46877    0.56691   0.827 0.408300
Bidding_Ratio            2.23614    1.05908   2.111 0.034738 *
Successive_Outbidding   11.27297    0.72742  15.497  < 2e-16 ***
Last_Bidding             1.64673    0.82655   1.992 0.046337 *
Auction_Bids             0.20984    0.74867   0.280 0.779255
Starting_Price_Average   0.23947    0.34836   0.687 0.491812
Early_Bidding           -1.22551    0.81131  -1.511 0.130906
Winning_Ratio            6.40928    0.71632   8.947  < 2e-16 ***
Auction_Duration         0.18842    0.05419   3.477 0.000507 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3301.87  on 4747  degrees of freedom
Residual deviance:  424.14  on 4738  degrees of freedom
AIC: 444.14

Number of Fisher Scoring iterations: 10
```

From the above output,it can be observed that the beta estimates of the glm() in R and beta estimates in (4a) is almost the same

```
fit.logit$converged
```

```
[1] TRUE
```

The above output shows that the glm() algorithm converges as well

PREDICTING WITH TEST DATA D3

(4c) Apply your trained logistic model in 4(a) to predict the response in the test data D3. Specifically, let X0 denote the design matrix from the test data; don't forget to add the first column of all 1's. with the default threshold 0.5. Compute the prediction accuracy

```
my_sigmoid<-function(z){
1/(1+exp(-z))
}
t_tdata=D3
G<-as.matrix(cbind(1,t_tdata[,-c(10)]))
t_tdata$fitted_result=my_sigmoid(G%*%fit$par)
t_tdata$fitted_result_class=ifelse(t_tdata$fitted_result>=0.5, 1,-1)
accuracy=sum(t_tdata$Class==t_tdata$fitted_result_class)/(nrow(t_tdata))
accuracy
```

```
[1] 0.9764781
```

the prediction accuracy is 97.6% when we use the trained logistic model in 4(a) to predict the response in the test data which is pretty good

PRIMITIVE LINEAR DISCRIMINANT ANALYSIS (LDA)

(5a) Let X1, X2, and X3 denote the matrix of all predictors or attributes in the training data D1, the validation data D2 and the test data D3, respectively. Scale or standardize X1. Then scale X2 according to the column means and SDs computed from X1. Check out the R function scale() to find out how this step can be done conveniently.

```
X1<-as.matrix(scale(D1[-c(10)]))
X2<-as.matrix(D2[-c(10)])
X3<-as.matrix(D3[-c(10)])
```

X1 is scaled using the scale function

OPTIMIZATION AND THE KERNEL TRICK

```
X1<-scale(as.matrix(dat[,-NCOL(dat)],center=TRUE,scale=TRUE))
mu0<-attributes(X1)$`scaled:center`
sd0<-attributes(X1)$`scaled:scale`
X2<-scale(X2, center = mu0, scale = sd0)
```

X2 is scaled with the mean and standard deviation of X1.

```
y<- D3[,c(10)]
X0<-cbind(X1,y)
```

PREDICTION WITH POLYNOMIAL KERNEL FAMILY

```
library(kernlab)
```

```

Attaching package: 'kernlab'

The following object is masked from 'package:ggplot2':

    alpha
```

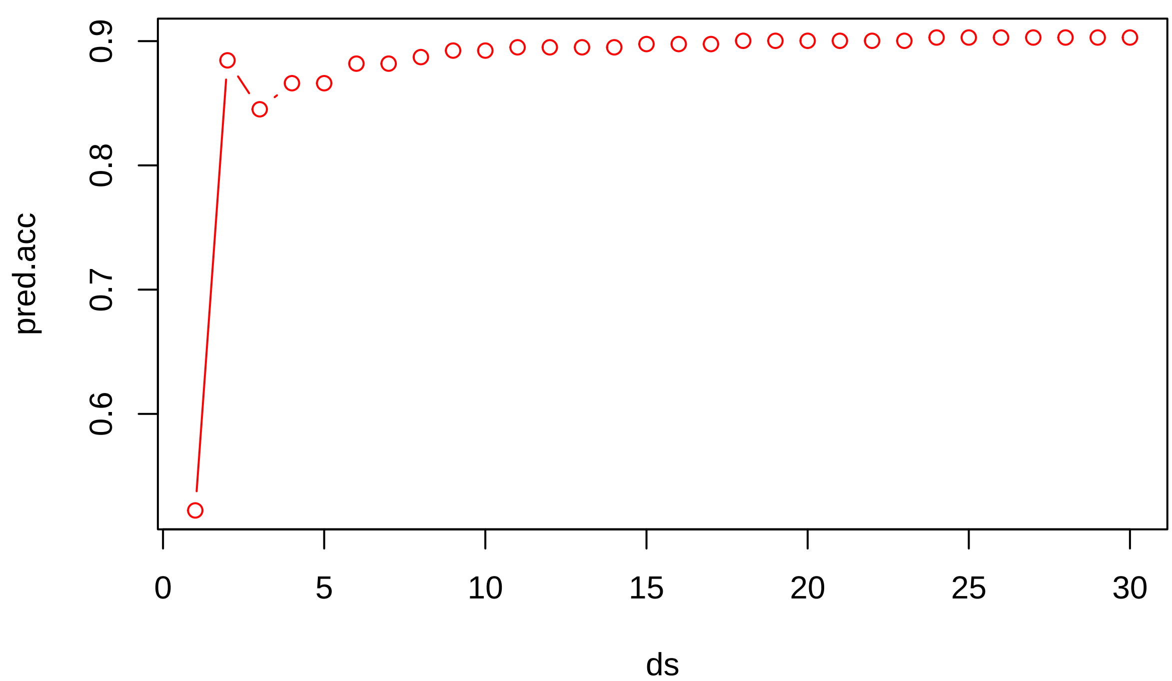
```
a<-1
c<-0
group<-2
ds<-1:30
pred.acc<-rep(0,length(ds))
for (i in 1:length(ds)){
  d<-ds[i]
  kern <- polydot(degree=d, scale=a,offset=c)
w.z<- colMeans(kernelMatrix(kernel = kern,x=X1[id.group==1 & y==1, ], y=as.matrix(X1[id.group==group,]))
colMeans(kernelMatrix(kernel = kern,x=X1[y==-1, ],y=as.matrix(X1[id.group==group,])))
b <-0.5*(mean (kernelMatrix(kernel=kern, x=X1[id.group==1 & y==-1,])))-
  mean(kernelMatrix(kernel=kern, x= X1[id.group==1 & y==+1,], y=as.matrix(X1[id.group==1& y==+1,])))
```

```
  tab<- table(sign (w.z+b),y[id.group==group]) ; tab # CLASSIFICATION TABLE
pred.accuracy <- sum (diag (tab))/sum (tab) # PREDICTION ACCURACY
pred.acc[i]<-pred.accuracy
cat ("The prediction accuracy is  n", pred.accuracy, "\n"  )
}
```

```
The prediction accuracy is
 0.5223097
The prediction accuracy is
 0.8845144
The prediction accuracy is
 0.8451444
The prediction accuracy is
 0.8661417
The prediction accuracy is
 0.8661417
The prediction accuracy is
 0.8818898
The prediction accuracy is
 0.8818898
The prediction accuracy is
 0.8871391
The prediction accuracy is
 0.8923885
The prediction accuracy is
 0.8923885
The prediction accuracy is
 0.8950131
The prediction accuracy is
 0.8950131
The prediction accuracy is
 0.8950131
The prediction accuracy is
 0.8950131
The prediction accuracy is
 0.8976378
The prediction accuracy is
 0.8976378
The prediction accuracy is
 0.8976378
The prediction accuracy is
 0.9002625
The prediction accuracy is
 0.9002625
The prediction accuracy is
 0.9002625
The prediction accuracy is
 0.9002625
The prediction accuracy is
 0.9002625
The prediction accuracy is
 0.9002625
The prediction accuracy is
```

```
 0.9028871
The prediction accuracy is
 0.9028871
The prediction accuracy is
 0.9028871
The prediction accuracy is
 0.9028871
The prediction accuracy is
 0.9028871
The prediction accuracy is
 0.9028871
The prediction accuracy is
 0.9028871
```

```
plot(ds,pred.acc,type="b",col="red")
```



From the above plot, d=15 appears to produce the best prediction accuracy

```
D<-rbind(D1,D2)
X<-D[,-c(10)]
scaledX<-scale(X, center = T, scale=T)
mu1<-attributes(scaledX)$`scaled:center`
sd1<-attributes(scaledX)$`scaled:scale`
X3<-scale(X3, center = mu1, scale = sd1)
```

From the above, X3 is scaled according to the column means and SDs computed from X0

```
a<-1;c<-0
d<-15
group<-3
kern <- polydot(degree=d, scale=a,offset=c)
w.z<- colMeans(kernelMatrix(kernel = kern,x=X1[id.group==1 & y==1, ], y=as.matrix(X1[id.group==group,]))
colMeans(kernelMatrix(kernel = kern,x=X1[y==-1, ],y=as.matrix(X1[id.group==group,])))
b <-0.5*(mean (kernelMatrix(kernel=kern, x=X1[id.group==1 & y==-1,])))-
  mean(kernelMatrix(kernel=kern, x= X1[id.group==1 & y==+1,], y=as.matrix(X1[id.group==1& y==+1,])))

  tab<- table(sign (w.z+b),y[id.group==group]) ; tab # CLASSIFICATION TABLE
```

```
      -1   1
  -1 354  36
  1    1   0
```

```
pred.accuracy <- sum (diag (tab))/sum (tab) # PREDICTION ACCURACY
cat ("The prediction accuracy is  n", pred.accuracy, "\n"  )
```

```
The prediction accuracy is
 0.9053708
```

When I use the best parameter d=15 on the testing set, the prediction accuracy is 90.5% which is pretty good.It appears however that the prediction accuracy in 4(c) is higher than the prediction accuracy here