Project II: Optimization and the Kernel Trick

Quaye, George Ekow

Contents

1)

1 Bringing in the data

1. Bring the data into R (or Python)

```
#Reading the data
data <- read.table(file = "Shill Bidding Dataset.csv",sep=",", header = T, na.strings = c("NA", "", " ")
                   stringsAsFactors = T)
dim(data)
```

```
## [1] 6321   13
```

```
head(data)
```

```
##   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
```

The data set has 6321 observations with 13 variables.

1.1 Taking out the first three columns

```
data<-data[-c(1:3)]
head(data)
```

```
##   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
```

```r
dim(data)
```

```
## [1] 6321   10
```

The first three columns are removed since they are just ID's, leaving the data set with a dimension of 6321 observations and 10 columns.

1.2 Changing the value 0 to -1

```r
data$Class[data$Class==0]<--1
head(data)
```

```
##   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    -1
## 2              0.9935928  0.0131226852     0.9444444                5    -1
## 3              0.9935928  0.0030416667     1.0000000                5    -1
## 4              0.9935928  0.0974768519     1.0000000                5    -1
## 5              0.0000000  0.0012417328     0.5000000                7    -1
## 6              0.0000000  0.0168435847     0.8000000                7    -1
```

For the purpose of a logistics modeling the zero value of the response has been changed to -1.

2 Exploratory Data Analysis (EDA)

2.1 Distinct values of variable

```r
str(data)
```

```
## '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 ...
```

With the exception of Auction_Duration which is an integer value by structure, all the other 9 variables are numeric.

```r
sapply(data, 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
```

The numerical variables Class and Successive_outbidding has few distinct values.

2.2 Missing data and imputation

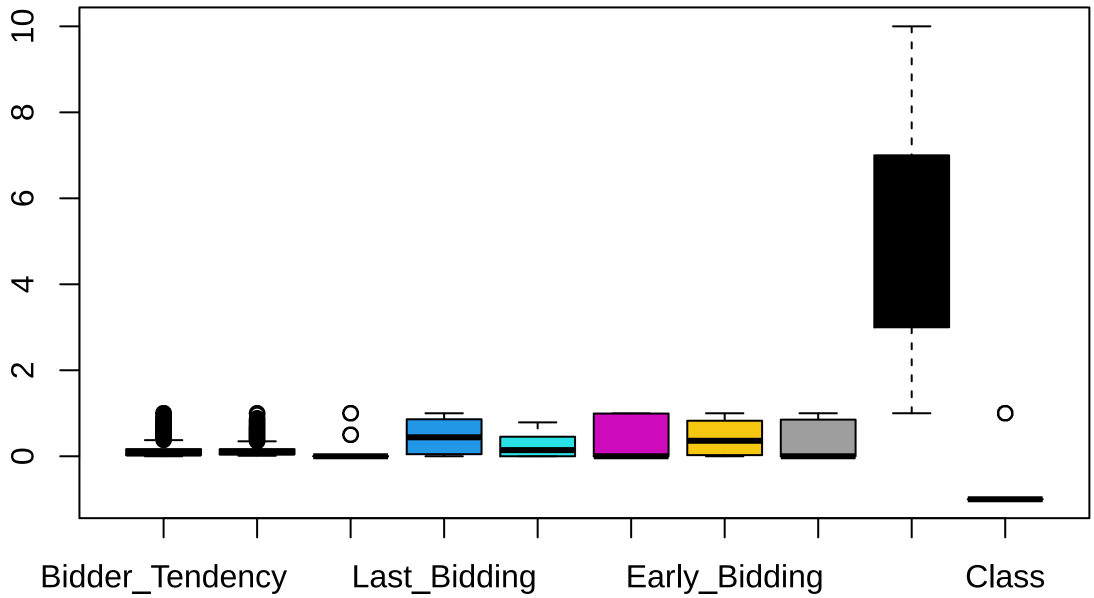
```r
# MISSING PERCENTAGES FOR ALL COLUMNS (OR VARIABLES)
colMeans(is.na(data))
```

```
##        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
```

There appears to be no missing values in the data indicated by the output above.

2.3 Parallel boxplot of predictors

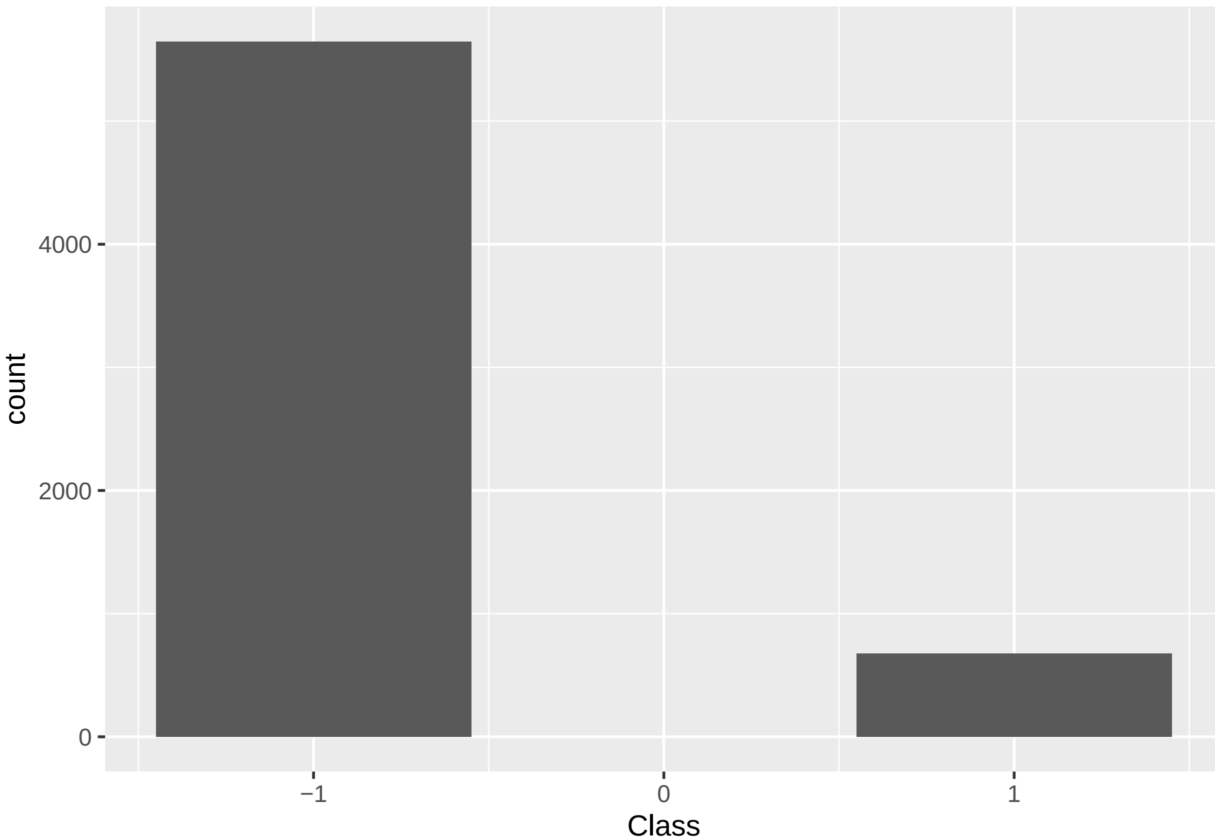
```
boxplot(data, col = c(1:9), horizontal = F)
```



Given the boxplot there appears to be unequal variations between the predictors variable and unequal range noticeably is the Auction_Duration and Successive_outbidding hence scaling is necessary for some particular modelings.

2.4 Bar plot of the Binary response

```
library(ggplot2)
c<-ggplot(data,aes(Class)) + geom_bar()
c
```



A bar plot is drawn to check the distribution of the target variable 'Class' and to ascertain whether or not there exist a balance or unbalanced classification, given the output above there appears to be an unbalanced classification with -1 having a by far higher percentage then 1 shown by the bars from the plot.

Note: This unbalanced classifiers can be balanced. The main objective of balancing classes is to either increasing the frequency of the minority class or decreasing the frequency of the majority class. This is done in order to obtain approximately the same number of instances for both the classes. Noticeable methods are Random Under-Sampling, Random Over-Sampling, Cluster-Based Over Sampling, Bagging Based techniques for imbalanced data etc.

3 Data Partitioning

```
#Partitioning of Data
set.seed(125)
n <- NROW(data)
id.split <- sample(x=1:3,size = n, replace = TRUE, prob = c(0.5,0.25,0.25))
TrainData <- data[id.split==1, ]
ValidData <- data[id.split==2, ]
TestData  <-  data[id.split==3, ]
dim(TrainData); dim(ValidData); dim(TestData)
```

```
## [1] 3171   10

## [1] 1596   10

## [1] 1554   10
```

The data set is partitioned for the purposed of training, validating and testing. with the TrainData ,ValidData and TestData having 3171,1596,1554 observations respectively.

4 Logistic Regression - Optimization

```
new_data<-rbind(TrainData,ValidData)
head(new_data)
```

```
##    Bidder_Tendency Bidding_Ratio Successive_Outbidding Last_Bidding
## 2       0.02439024    0.20000000                     0  0.013122685
## 3       0.14285714    0.20000000                     0  0.003041667
## 4       0.10000000    0.20000000                     0  0.097476852
## 8       0.13793103    0.44444444                     1  0.768043981
## 11      0.60000000    0.56250000                     1  0.457630622
## 13      0.01724138    0.05263158                     0  0.057655423
##    Auction_Bids Starting_Price_Average Early_Bidding Winning_Ratio
## 2    0.00000000              0.9935928   0.013122685     0.9444444
## 3    0.00000000              0.9935928   0.003041667     1.0000000
## 4    0.00000000              0.9935928   0.097476852     1.0000000
## 8    0.00000000              0.0000000   0.016311177     1.0000000
## 11   0.00000000              0.0000000   0.457473545     0.6000000
## 13   0.05263158              0.0000000   0.057655423     0.0000000
##    Auction_Duration Class
## 2                 5    -1
## 3                 5    -1
## 4                 5    -1
## 8                 7     1
## 11                7     1
## 13                7    -1
```

```
dim(new_data)
```

```
## [1] 4767   10
```

The combined Train and Validation is labeled new_data with 4767 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 <- new_data$Class
X <- as.matrix(new_data[, c(1:9)])
p <- NCOL(X) +1
fit <- optim(par=rep(0,p), fn=nloglik, method="BFGS",hessian=T, X=X, y=y)
estimate <- fit$par; estimate
```

```
##  [1] -10.10785135   1.05384204   1.25123047  10.49377567   0.93247462
##  [6]   0.63452561   0.12535724  -0.64309290   4.77652987   0.05764265
```

The output above indicates the Beta estimates for each predictor variables.

```
D<-solve(fit$hessian) # Obtaining the inverse of the covariance matrix
SE<-sqrt(diag(D)) # Standard errors
```

```
tval<-fit$par/SE # testing for each attributes .
pval<-2*(1-pt(abs(tval),nrow(X)-ncol(X))) # P_values for each betas
results<-cbind(fit$par,SE,tval,pval)
colnames(results)<-c("Beta","SE","t_value","P_value")
rownames(results)<-c("beta0","beta1","beta2","beta3","beta4","beta5","beta6","beta7","beta8","beta9")
print(results,digits=3)
```

```
##          Beta     SE t_value  P_value
## beta0 -10.1079 0.7708 -13.113 0.00e+00
## beta1   1.0538 0.5310   1.985 4.72e-02
## beta2   1.2512 0.9695   1.291 1.97e-01
## beta3  10.4938 0.6497  16.152 0.00e+00
## beta4   0.9325 0.7750   1.203 2.29e-01
## beta5   0.6345 0.7316   0.867 3.86e-01
## beta6   0.1254 0.3314   0.378 7.05e-01
## beta7  -0.6431 0.7740  -0.831 4.06e-01
## beta8   4.7765 0.6318   7.560 4.80e-14
## beta9   0.0576 0.0502   1.149 2.51e-01
```

Given $alpha$=0.05 and the p_values from test it is noticed that Bidder_Tendency,Successive_Outbidding,Winning_Ratio are statistically significant with $p_values < 0.05$. The optimization method used here is the 'BFGS'.

```
fit$convergence
```

```
## [1] 0
```

There is a successive convergence in the algorithm given that it converges to *zero*.

4.1 Comparing to Standard R function glm()

```
dat<-new_data[,-c(10)]
dat$y <- ifelse(new_data$Class ==-1,0,1)
fit.logit <- glm(y~., data=dat, family=binomial(link = "logit"))
data.frame(fit.logit$coef)
```

```
##                        fit.logit.coef
## (Intercept)              -10.10785348
## Bidder_Tendency            1.05386572
## Bidding_Ratio              1.25123937
## Successive_Outbidding     10.49376814
```

```
## Last_Bidding              0.93246937
## Auction_Bids              0.63453077
## Starting_Price_Average    0.12535664
## Early_Bidding            -0.64308500
## Winning_Ratio             4.77652387
## Auction_Duration          0.05764265
```

By comparing the coefficients obtained in glm() to that of 4(a) there appears to be no differences in the values.

```
fit.logit$converged
```

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

Also the algorithm in the glm() model converges.

4.2 Predicting with Test Data (D3)

```
my_sigmoid<-function(z){
  1/(1+exp(-z))
}

t_tdata=TestData
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, 1,0)

accuracy=sum(t_tdata$Class==t_tdata$fitted_result_class)/(nrow(t_tdata))
accuracy
```

```
## [1] 0.09073359
```

The prediction accuracy with a threshold of 0.5 is 9.07%. This small prediction accuracy might be as a result of the unbalanced classification of the response variable.

5 Primitive LDA – The Kernel Trick

5.1 Scaling Required Data

```
X1<-as.matrix(TrainData[-c(10)])
X2<-as.matrix(ValidData[-c(10)])
X3<-as.matrix(TestData[-c(10)])
```

```
scaledX1<-scale(X1, center = T, scale = T)
```

```
#attributes(scaledX1)
```

```
mu0<-attributes(scaledX1)$`scaled:center`
sd0<-attributes(scaledX1)$`scaled:scale`

scaledX2<-scale(X2, center = mu0, scale = sd0)
```

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

```
y<- TrainData[,c(10)]
x11<-cbind(scaledX1,y)
```

b)

5.2 Obtaining prediction accuracy with Laplace Kernel Family

```
library(kernlab)
```

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

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

```
sigma <- 1:20
pred_acc<-rep(0, length(sigma))
for (i in 1: length(sigma)){
  s=sigma[i]
  kern<- laplacedot(sigma = s)

  w.z<-  colMeans(kernelMatrix(kernel = kern,x=x11[y==1, ], y=(cbind(scaledX2, ValidData[,c(10)]))))-
    colMeans(kernelMatrix(kernel = kern,x=x11[y==-1, ],y=(cbind(scaledX2, ValidData[,c(10)]))))

  b<-0.5*(mean(kernelMatrix(kernel = kern,x=x11[y==-1, ], y=(cbind(scaledX2, ValidData[,c(10)])))) -
    mean(colMeans(kernelMatrix(kernel = kern,x=x11[y==1, ], y=(cbind(scaledX2, ValidData[,c(10)])))))
  tab<- table(sign(w.z+b),ValidData[,c(10)]);tab
  pred_accuracy<- sum(diag(tab))/sum(tab)
  pred_acc[i]<-pred_accuracy
  cat("The prediction accuracy is \n", pred_accuracy, "\n")
}
```

```
## The prediction accuracy is
##  0.9360902
## The prediction accuracy is
##  0.8176692
## The prediction accuracy is
##  0.7418546
## The prediction accuracy is
```

```
## 0.6860902
## The prediction accuracy is
## 0.6422306
## The prediction accuracy is
## 0.6058897
## The prediction accuracy is
## 0.5802005
## The prediction accuracy is
## 0.547619
## The prediction accuracy is
## 0.5300752
## The prediction accuracy is
## 0.5068922
## The prediction accuracy is
## 0.4931078
## The prediction accuracy is
## 0.4711779
## The prediction accuracy is
## 0.4561404
## The prediction accuracy is
## 0.4392231
## The prediction accuracy is
## 0.4285714
## The prediction accuracy is
## 0.4191729
## The prediction accuracy is
## 0.4078947
## The prediction accuracy is
## 0.3997494
## The prediction accuracy is
## 0.3878446
## The prediction accuracy is
## 0.3784461
```

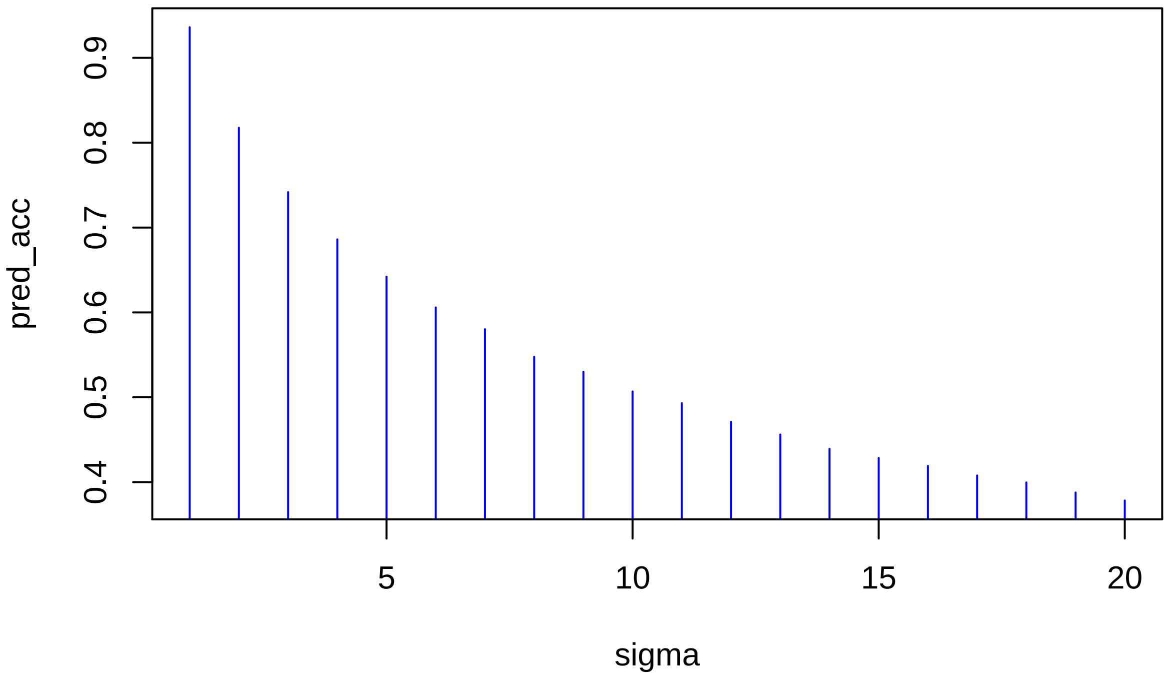
```r
laplacedot()
```

```
## Laplace kernel function.
##  Hyperparameter : sigma =  1
```

A laplace kernel family was used, the parameter *sigma* was set between $1:20$. The hyper parameter obtained from this algorithm was *sigma*=1 with a prediction accuracy of 93.61%.

5.3 Plot of the prediction accuracy values versus the candidate parameter values.

```r
plot(sigma,pred_acc,type="h",col="blue")
```



From the plot above its confirms that the best laplace parameter to be used is *sigma*=1.

5.4 Applying the best Kernel to Training and Validation data

```
Dprime<-rbind(TrainData,ValidData)
Xprime<-Dprime[,-c(10)]
scaled_Xprime<-scale(Xprime, center = T, scale=T)

mu1<-attributes(scaled_Xprime)$`scaled:center`
sd1<-attributes(scaled_Xprime)$`scaled:scale`

scaledX3<-scale(X3, center = mu1, scale = sd1)
```

```
  kern<- laplacedot(sigma = 1)

  w.z<-  colMeans(kernelMatrix(kernel = kern,x=x11[y==1, ], y=(cbind(scaledX3, TestData[,c(10)]))))-
    colMeans(kernelMatrix(kernel = kern,x=x11[y==-1, ],y=(cbind(scaledX3, TestData[,c(10)]))))

  b<-0.5*(mean(kernelMatrix(kernel = kern,x=x11[y==-1, ], y=(cbind(scaledX3, TestData[,c(10)])))) -
    mean(colMeans(kernelMatrix(kernel = kern,x=x11[y==1, ], y=(cbind(scaledX3, TestData[,c(10)])))))
  tab<- table(sign(w.z+b),TestData[,c(10)]);#tab
  pred.accuracy<- sum(diag(tab))/sum(tab)
 # pred.accuracy
  cat("The prediction accuracy is \n", pred.accuracy, "\n")
```

```
## The prediction accuracy is
##  0.9414414
```

After applying the best kernel laplace parameter (sigma=1) to the combined TrainData and ValidData to form D', the prediction accuracy obtained was 94.14% which is 85.34% more than the prediction accuracy obtained in 4(c) which was 9.07%. Hence the kernel family gives more prediction accuracy.