# leave-one-out cross validation  
# This function receives a lambda sequence, predictors and response  
# returns a lambda selected by cross-validation  
# also plots accuracy vs lambda  
naive.bayes.cv <- function(lambda.seq, predictors, response){  
 miscl <- rep(NA, length(lambda.seq))  
 for (k in 1:length(lambda.seq)) {  
 lambda <- lambda.seq[k]  
 n <- length(response)  
 naiveBayes.raw <- rep(NA, n)  
   
 # iteratively choose one as test data every time  
 for (i in 1:n) {  
 data.tr.predictors <- predictors[-i,]  
 data.tr.response <- response[-i]  
 data.test.predictors <- predictors[i,]  
 data.test.response <- response[i]  
   
 # select ones and zeros  
 ones <- which(data.tr.response %in% c(1))  
 response.ones <- which(response %in% c(1))  
 len.ones <- length(response.ones)  
 len.zeros <- n - len.ones  
   
 # KDE  
 density <- dnorm(abs(data.test.predictors - data.tr.predictors)/lambda)  
 density.ones <- density[ones,]  
 density.zeros <- density[-ones,]  
   
 # prior estimated probability  
 prior.ones <- len.ones/n  
 prior.zeros <- len.zeros/n  
   
 # KDE for different classes  
 density.one <- prod(apply(density.ones, 2, sum)/len.ones)  
 density.zero <- prod(apply(density.zeros, 2, sum)/len.zeros)  
   
 # posterior for class one  
 prob.one <- prior.ones \* density.one / (prior.ones \* density.one + prior.zeros \* density.zero)  
 naiveBayes.raw[i] <- prob.one   
 }  
 # assign the class based on raw scores  
 cl <- sapply(naiveBayes.raw, function(x) x>0.5)  
 miscl[k] <- 1-mean(cl == response)  
 }  
 plot(lambda.seq, miscl, ylim=c(0.25, 0.5), ylab="misclassification rate", xlab="lambda", type="l")  
 abline(v=lambda.seq[which.min(miscl)], col="red", lty=3)  
 cat("Best lambda is :", lambda.seq[which.min(miscl)],"\n")  
}

# naive bayes classifier  
# this function receives a certain value of lambda, predictors and   
# the train predictors that are used to estimate density  
# it returns the raw scores of NBC  
naive.bayes.cl <- function(lambda, predictors, trainPredictors) {  
 n <- dim(predictors)[1]  
 naiveBayes.raw <- rep(NA, n)  
 for (i in 1:n) {  
 # data processing  
 data.test.predictors <- predictors[i,]  
 ones <- which(trainResponse %in% c(1))  
 response.ones <- which(trainResponse %in% c(1))  
 len.ones <- length(response.ones)  
 len.zeros <- length(trainResponse) - len.ones  
   
 # KDE  
 density <- dnorm(abs(data.test.predictors - trainPredictors)/lambda, mean = 0, sd=1)  
 density.ones <- density[ones,]  
 density.zeros <- density[-ones,]  
 density.one <- prod(apply(density.ones, 2, sum)/len.ones)  
 density.zero <- prod(apply(density.zeros, 2, sum)/len.zeros)  
   
 # prior estimated probability  
 prior.ones <- len.ones/length(trainResponse)  
 prior.zeros <- len.zeros/length(trainResponse)  
   
 # posterior probability  
 prob.one <- prior.ones \* density.one / (prior.ones \* density.one + prior.zeros \* density.zero)  
 naiveBayes.raw[i] <- prob.one   
 }  
 return(naiveBayes.raw)  
}