

Kernel Density Estimation and Classification

Boyoung Hur, Jaejeong Shin, Shirong Zhao

1 Introduction

Kernel density estimation is a non-parametric estimation of the probability density function of random variable. Usually, we can use the histogram for a non-parametric estimation if the data we have is discrete. However, if the data is continuous, then we need to consider the Kernel density function. It is also an unsupervised learning and it can be used for classification, which is Kernel density classification. We will introduce these non parametric estimations and compare these to linear discriminant classifier we have learned in class.

1.1 Kernel Density Estimation

Kernel Density Estimation (KDE) is a useful statistical tool. It is a technique to create a smooth curve given a set of data. Suppose, given a random sample x_1, \dots, x_N drawn from a probability density function $f_x(x)$, we want to estimate f_x at a point x_0 where is not in sample. In this situation, kernels can be used to construct non-parametric estimate for f_x without any assumption for unknown parameters as we did in parametric estimation. We can consider the following form.

$$\hat{f}_X(x_0) = \frac{1}{N\lambda} \sum_{i=1}^N K_\lambda(x_0, x_i) \quad (1)$$

where λ is bandwidth parameter for kernel K . (That is, this λ determines the flatness of top in kernel function.) There are many kinds of kernel function such as Gaussian, Uniform that are symmetric on 0, non-negative, but the most common used K is the Gaussian kernel. Figure 1 [1] is showing how the density estimate is constructed. As we can see, based on each sample points, the kernel function is generated and then get a sum and divide into the number of samples as in the form (1).

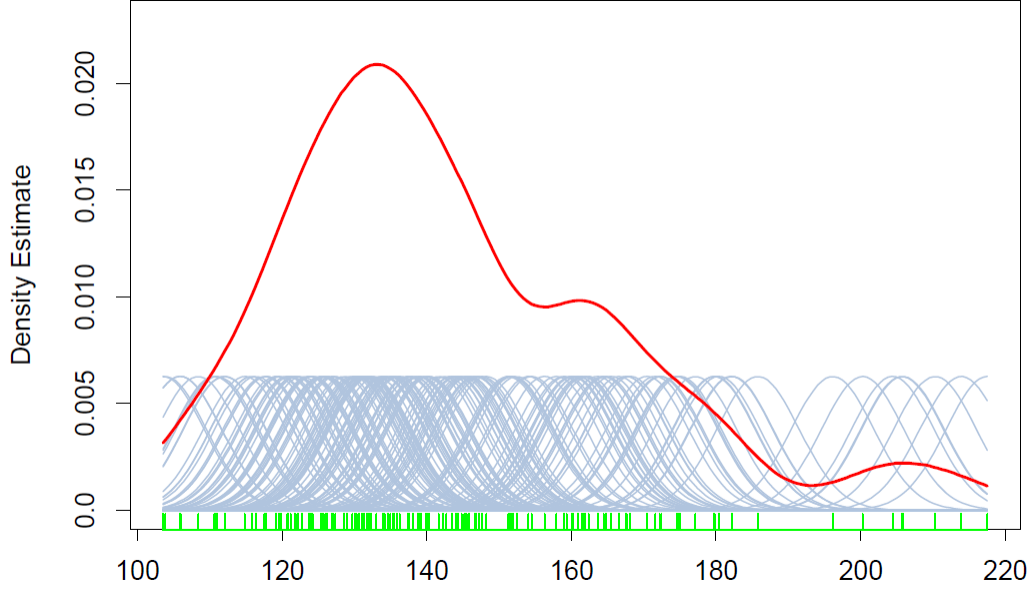


Figure 1: A kernel density estimate

1.2 Kernel Density Classification

We can use this non-parametric estimation for classification using Bayes' theorem. Suppose we have the response variable y with J different categories and remind linear discriminant analysis in Chapter 4. We have used Bayes's theorem in the form of

$$P(G = k | \mathbf{X} = \mathbf{x}) = \frac{f_k(x)\pi_k}{\sum_l f_l(x)\pi_l}.$$

Here, knowing $f_k(x)$ is almost equivalent to knowing the density function to predict the classification, that is, $P(G = k | \mathbf{X} = \mathbf{x})$. In the kernel density classification, we estimate $f_k(x)$ using kernel density function. That is,

$$\hat{P}(G = k | \mathbf{X} = \mathbf{x}) = \frac{\hat{f}_k(x)\hat{\pi}_k}{\sum_l \hat{f}_l(x)\hat{\pi}_l}$$

where $\hat{f}_k(x_0)$ is estimated density at x_0 based on a kernel density fit involving only observations from k th class and usually, $\hat{\pi}_j$ is sample proportion falling into j th category. ($\hat{\pi}_j$ is the estimate of the prior probability of class j). Unlike linear discriminant classifier, it is not restricted to a parametric specification. However, it has a curse of dimensionality problem that there are many regions with little data that makes unstable prediction. That is, higher x values ($x > 180$) in figure 1 has sparse data set, that is why the prediction on that region can have higher variance. To avoid this problem in high-dimensional analysis, the Naive Bayes Classifier can be used.

1.3 The Naive Bayes Classifier

The Naive Bayes Classifier is the popular technique over the years [1]. As mentioned above, it is better for the high dimensional analysis than kernel density classification. It assumes that given a class $G = k$, X_k are independent, thus,

$$\hat{f}_l(\mathbf{X}) = \prod_{i=1}^p \hat{f}_{li}(X_i) \quad (2)$$

* where \hat{f}_{li} is an estimate of the density of the i th variable in l th class. Actually, this assumption is generally not true but it can simplify the estimation dramatically. That is, even though it increased the bias, it lower the variance drastically. Also, it can alleviates the curse of high dimension by taking f_{li} to be estimated with one-dimensional kernel in each class. Thus, no matter if we have enough sample data set, this classifier has stable prediction. To summarize, if classification is the ultimate goal, then learning the separate class densities well may be unnecessary, and can in fact be misleading. In fact, if classification is the ultimate goal, then we need only to estimate the posterior well near the decision boundary (for two classes, this is the set $\{x | Pr(G = 1 | X = x) = \frac{1}{2}\}$). We can find this in Figure 2. If the individual density of each two covariates are far each other, so that it is easy to classify, then posterior density is also well-behaved, but if these covariates are distributed closely, and thus it is more difficult to classify properly, then the posterior density is not well-behaved. Nevertheless, the classification is well-determined with more than 0.5 probability near the decision boundary ($x = 0$ in this situation).

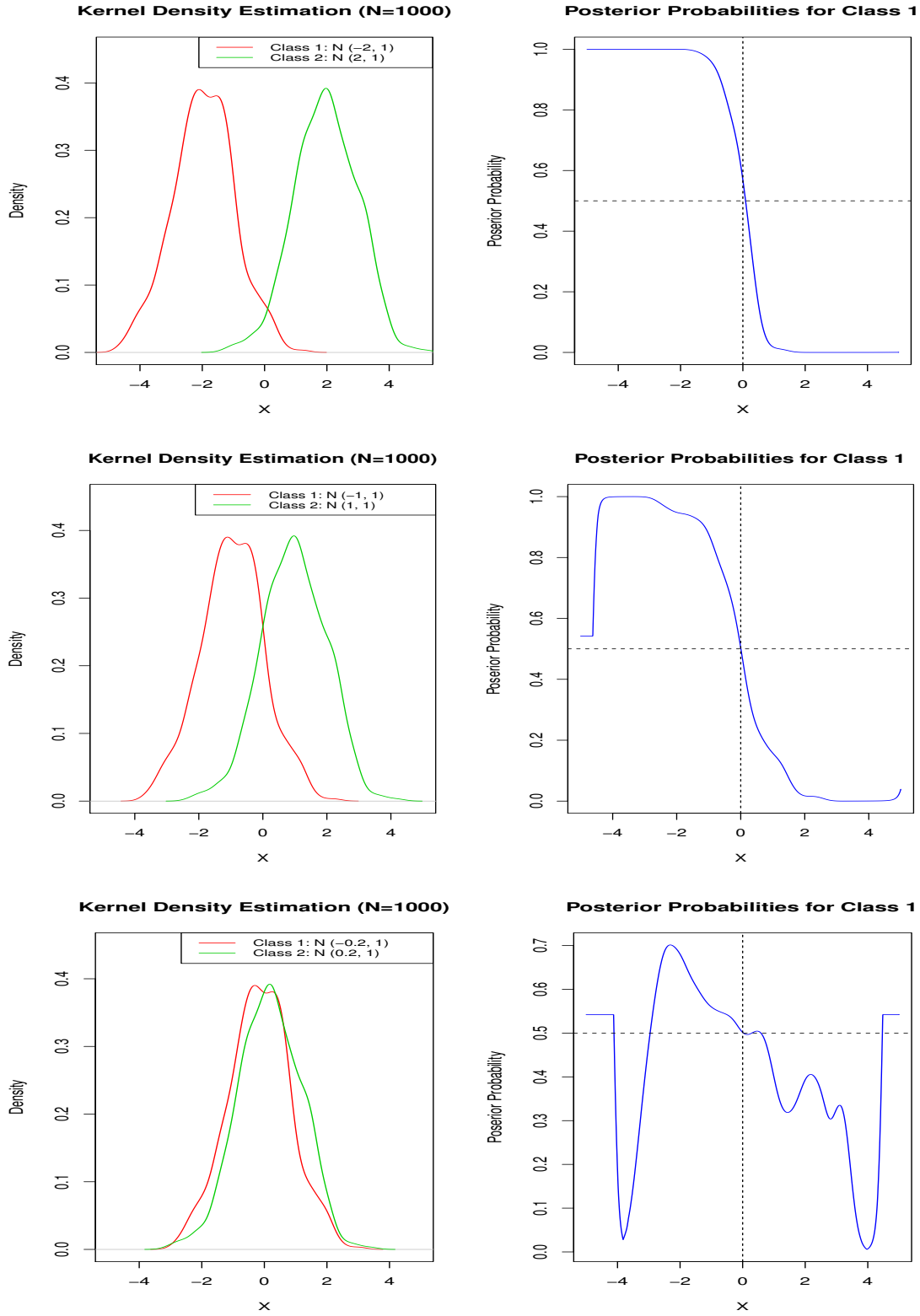


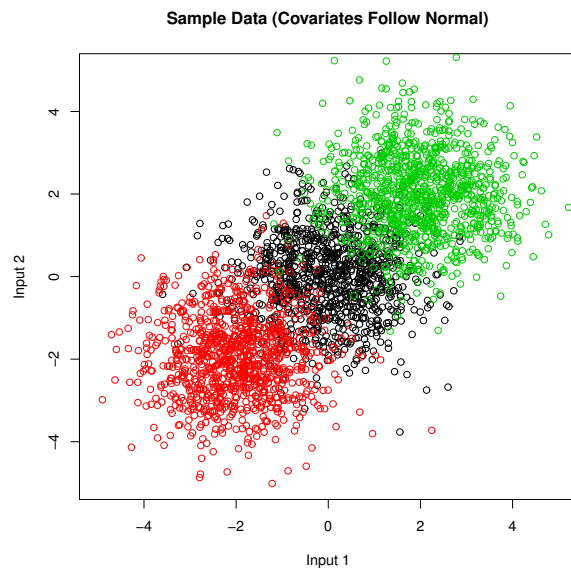
Figure 2:

2 Simulation

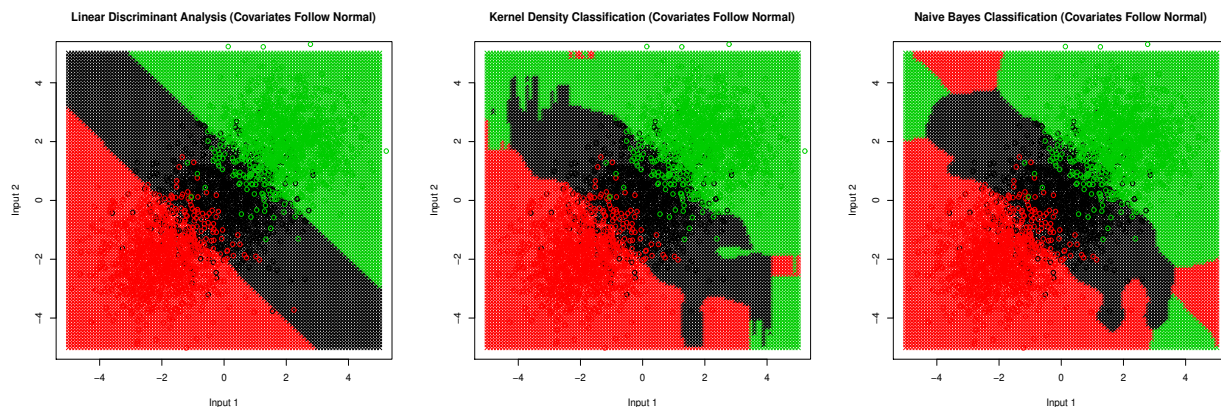
To see the difference between the Linear Discriminant Analysis (LDA), the Kernel Density Classification (KDC) and the Naive Bayes Classifier (NBC), we generate 3 classes of Y with 2 covariates X_1 and X_2 as like in class. We are considering the following two cases, 1. X_1 and X_2 are following normal distribution and 2. X_1 and X_2 are following log-normal distribution.

Case 1. X_1 and X_2 are following normal distribution.

- The distribution of sample dataset.

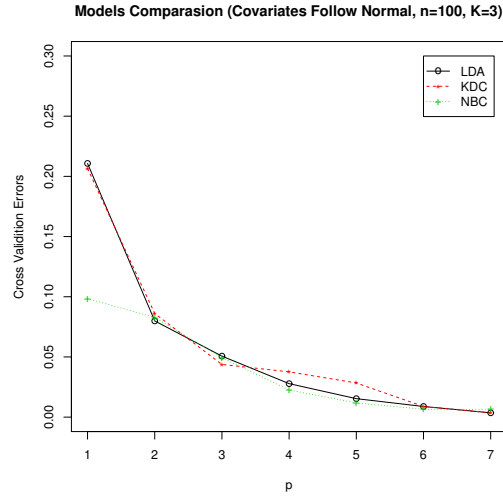


- Applying LDA, KDC and NBC.



- **Results**

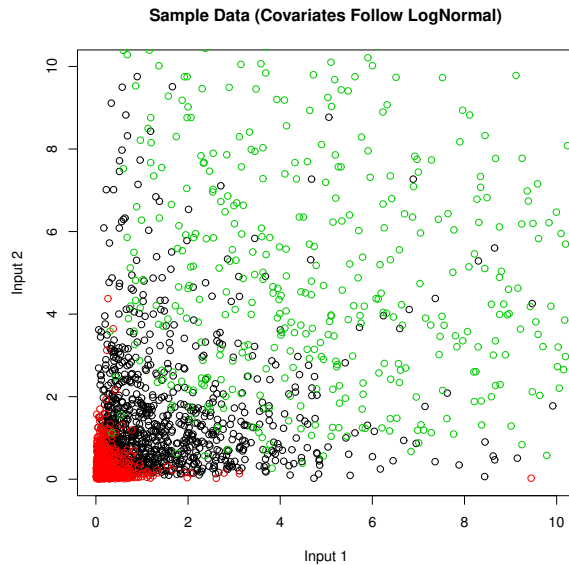
To see the results more clearly, we've done the cross-validation as following.



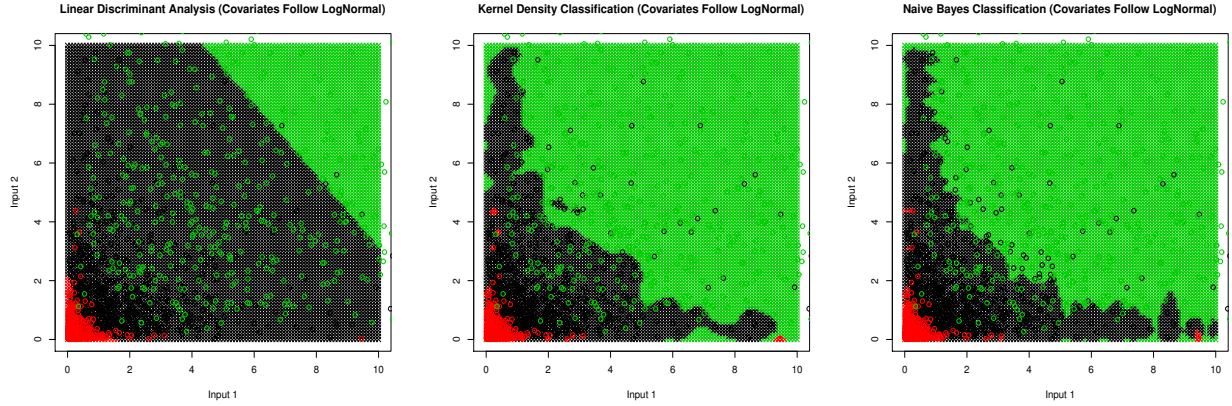
Overall, NBC is performing better than LDA and KDC since we are assuming the 2 covariates are independent when we generate the dataset for simulation. Also, as p is increased, that is as the number of covariates is increased, LDA is showing better cross validation error than KDC. It is because in LDA we assume the $f_k(x)$ as the normal distribution and also our covariates are from normal distribution as well.

Case 2. X_1 and X_2 are following log-normal distribution.

- The distribution of sample dataset.

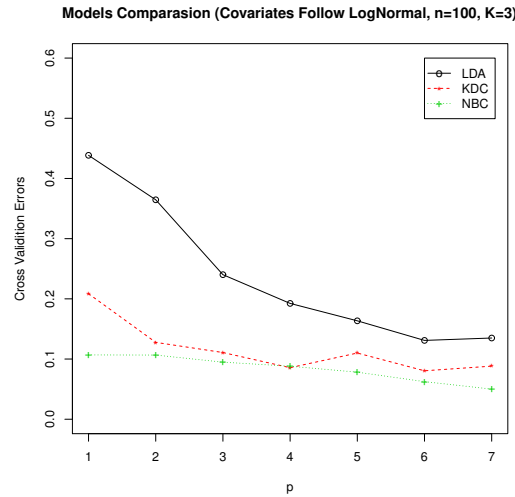


- Applying LDA, KDC and NBC.



• Results

To see the results more clearly, we've done the cross-validation as following.



Similar to case1 above, NBC is performing better than LDA and KDC since we are assuming the 2 covariates are independent when we generate the dataset for simulation. However, as p is increased, that is as the number of covariates is increased, KDC is showing better cross validation error than LDA. It is because this covariates is not following normal distribution anymore and thus KDC which is not restricted to linear is performing better than LDA.

3 Data application

To see if the result from simulation work well in real dataset, we've taken South African heart disease data which is from <https://web.stanford.edu/hastie/ElemStatLearn/>. The variables are defined as following.

- *chd*: response, coronary heart disease
- *sbp*: systolic blood pressure
- *tobacco*: cumulative tobacco (kg)
- *ldl*: low density lipoprotein cholesterol
- *famhist*: family history of heart disease (Present, Absent)
- *typea*: type-A behavior
- *alcohol*: current alcohol consumption
- *age*: age at onset

That is, we'd like to classify into the classes of *chd* which is the response variables based on covariates *sbp*, *tobacco*, *ldl*, *famhist*, *typea*, *alcohol* and *age*. Before we apply the classification, we can see the correlation between each variables. From this correlation plot, we can find that the covariates *tobacco* and *sbp* are highly correlated with *age* and also age is highly correlated with the response variable *chd*.

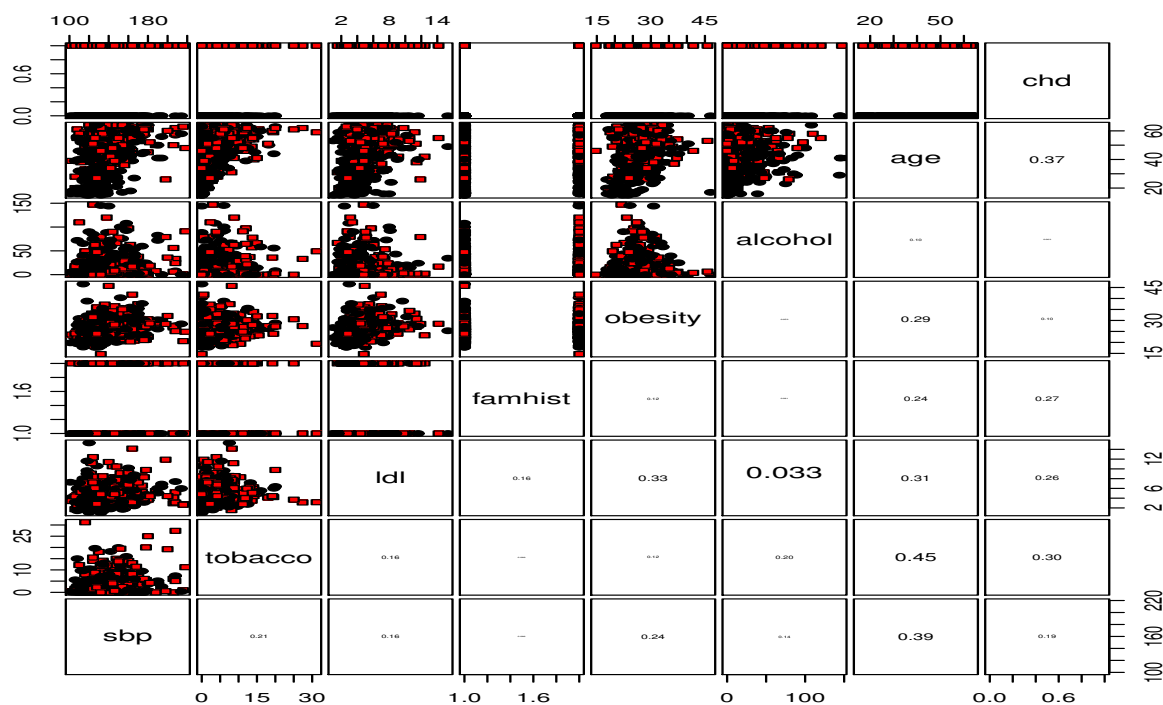


Figure 3: Scatter plot matrix of data

Also, to see if the covariates are satisfying the assumption such as normal distribution, we construct the kernel density function for each covariates in Figure 4. From this densities, we can find that each covariates are not exactly following normal but it closes to normal distribution.

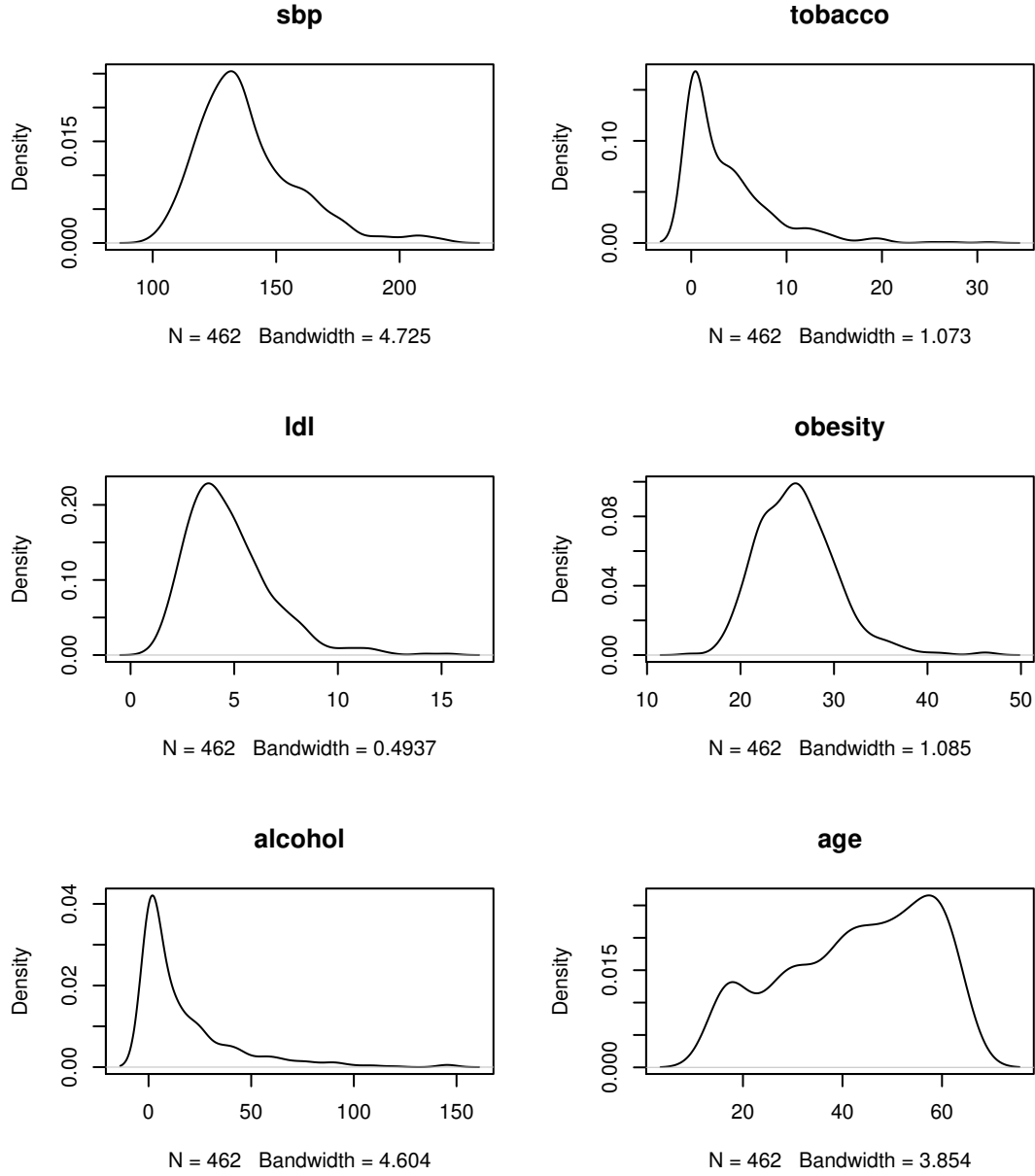


Figure 4: Kernel density of covariates

Based on these variables, we applied LDA, KDC and NBC and compared the cross-validation errors to compare the mis specification error. We are using 6-fold cross validation

estimate using the following form.

$$CV_k = \sum_{i \in \text{Test}_k} \frac{I(chd_i \neq \widehat{chd}_i)}{N_k}$$

$$CV = \sum_{k=1}^6 \frac{CV_k}{6}$$

Based on this CV error, we got the following results.

$$0.275(LDA) < 0.297(NBC) < 0.387(KDC)$$

In conclusion, since each covariates are close to normal distribution, the LDA which is using normal distribution for the density function has the lowest CV error. Also, since as we've seen in correlation plot, there are some covariates which is highly correlated each other, NBC has better (lower) CV error because NBC is assuming each covariates are independent so that each correlation does not affect to the result. Additionally, to see the Naive Bayese Classifier(NBC) more precisely, we have seen the marginal density for each class using the NBC as following.

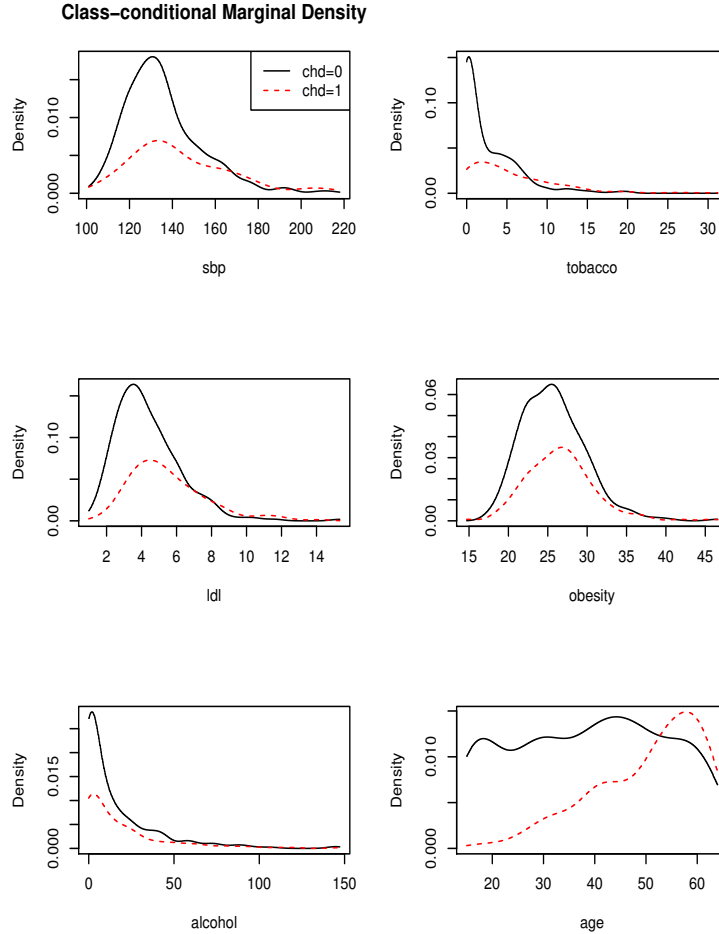


Figure 5: Naive Bayes Marginal Density

References

- [1] FRIEDMAN, J., HASTIE, T., AND TIBSHIRANI, R. *The elements of statistical learning*, vol. 1. Springer series in statistics New York, 2001.

Appendix: R Code

```
1
2 # R Code for Simulation
3
4 ## Scatter plots of simulation (sampling from normal distribution)
5
6 library(MASS)
7
8 set.seed(123456)
9
10 n<-1000
11
12
13 X1<-mvrnorm(n,mu=c(0,0),Sigma=diag(c(1,1)))
14 X2<-mvrnorm(n,mu=c(-2,-2),Sigma=diag(c(1,1)))
15 X3<-mvrnorm(n,mu=c(2,2),Sigma=diag(c(1,1)))
16
17
18 plot(X1[,1],X1[,2],main="Sample Data (Covariates Follow Normal)",xlim=c(-5,5),ylim=c(-5,5),
19       xlab="Input 1", ylab="Input 2")
19 points(X2[,1],X2[,2],col=2)
20 points(X3[,1],X3[,2],col=3)
21
22
23 Y<-c(rep(1,n),rep(2,n),rep(3,n))
24
25 X<-rbind(X1,X2,X3)
26
27 #
28 #####
29
30 ### Linear discriminant Analysis
31
32 # Note: This function assumes that the KK classes are encoded as 1:KK
33 LDA.func<-function(xp,Y,Xt){
34   KK<-length(unique(Y))
35   N<-length(Y)
36   p<-length(xp)
37   muk<-matrix(-99,nrow=KK,ncol=p)
38   pik<-rep(-99,KK)
39   Xtc<-Xt
40
41   for(k in 1:KK){
42     muk[k,]<-apply(Xt[Y==k,],2,mean)
43     pik[k]<-mean(Y==k)
44     Xtc[Y==k,]<-t(t(Xt[Y==k,])-muk[k,])
45   }
46   Sig<-t(Xtc)%*%Xtc/(N-KK)
47   Sigi<-solve(Sig)
48   dk<-rep(-99,KK)
49   for(k in 1:KK){
50     dk[k]<-t(xp)%*%Sigi%*%muk[k,]-t(muk[k,])%*%Sigi%*%muk[k,]/2 +log(pik[k])
51   }
52   return(order(dk)[KK])
53 }
54
55 grid<-expand.grid(seq(-5,5,.1),seq(-5,5,.1))
56 part<-apply(grid,1,LDA.func,Y=Y,Xt=X)
57
58 ### Plot
59 plot(grid[,1],grid[,2],col=part,pch=4, xlab="Input 1", ylab="Input 2",main="Linear
60 Discriminant Analysis (Covariates Follow Normal)",xlim=c(-5,5),ylim=c(-5,5))
61 points(X1[,1],X1[,2])
62 points(X2[,1],X2[,2],col=2)
63 points(X3[,1],X3[,2],col=3)
```

```

62 |
63 |
64 | #####
65 | ## kernel density classification
66 |
67 | library(ks)
68 | library(lattice)
69 |
70 | grid<-expand.grid(seq(-5,5,.1),seq(-5,5,.1))
71 |
72 | # KK is the number of classes
73 | KK<-length(unique(Y))
74 | pik<-rep(-99, KK)
75 |
76 | for(k in 1:KK){
77 |     pik[k]<-mean(Y==k)
78 | }
79 |
80 | fhat1 <- kde(x=X1,eval.points=grid)$estimate
81 | fhat2 <- kde(x=X2,eval.points=grid)$estimate
82 | fhat3 <- kde(x=X3,eval.points=grid)$estimate
83 |
84 | fhat<-cbind(pik[1]*fhat1, pik[2]*fhat2, pik[3]*fhat3)
85 |
86 | part<-rep(NA,length(grid[,1]))
87 |
88 | for (k in 1:length(grid[,1])){
89 |
90 |     fhatk=fhat[k,]
91 |
92 |     part[k]<-order(fhatk)[3]
93 |
94 | }
95 |
96 |
97 | ## Plot
98 | plot(grid[,1],grid[,2],col=part,pch=4, xlab="Input 1", ylab="Input 2",main="Kernel Density
99 |     Classification (Covariates Follow Normal)",xlim=c(-5,5),ylim=c(-5,5))
100 | points(X1[,1],X1[,2])
101 | points(X2[,1],X2[,2],col=2)
102 | points(X3[,1],X3[,2],col=3)
103 |
104 |
105 |
106 |
107 |
108 | #####
109 | ## Naive Bayes Classification
110 |
111 | library(ks)
112 | library(lattice)
113 |
114 | grid<-expand.grid(seq(-5,5,.1),seq(-5,5,.1))
115 |
116 | # K is the number of classes
117 | KK<-length(unique(Y))
118 | pik<-rep(-99, KK)
119 |
120 | for(k in 1:KK){
121 |     pik[k]<-mean(Y==k)
122 | }
123 |
124 | fhat11 <- kde(x=X1[,1],eval.points=grid[,1])$estimate
125 | fhat12 <- kde(x=X1[,2],eval.points=grid[,2])$estimate
126 | fhat21 <- kde(x=X2[,1],eval.points=grid[,1])$estimate
127 | fhat22 <- kde(x=X2[,2],eval.points=grid[,2])$estimate
128 | fhat31 <- kde(x=X3[,1],eval.points=grid[,1])$estimate

```

```

129 fhat32 <- kde(x=X3[,2],eval.points=grid[,2])$estimate
130
131 fhat1 <- fhat11*fhat12
132 fhat2 <- fhat21*fhat22
133 fhat3 <- fhat31*fhat32
134
135 fhat<-cbind(pik[1]*fhat1,pik[2]*fhat2, pik[3]*fhat3)
136
137 part<-rep(NA,length(grid[,1]))
138
139 for (k in 1:length(grid[,1])){
140
141   fhatk=fhat[k,]
142
143   part[k]<-order(fhatk)[3]
144
145 }
146
147
148 ### Plot
149 plot(grid[,1],grid[,2],col=part,pch=4, xlab="Input 1", ylab="Input 2",main="Naive Bayes
      Classification (Covariates Follow Normal)",xlim=c(-5,5),ylim=c(-5,5))
150 points(X1[,1],X1[,2])
151 points(X2[,1],X2[,2],col=2)
152 points(X3[,1],X3[,2],col=3)
153
154
155 ## Scatter plots of simulation (sampling from log-normal distribution)
156
157 {r}
158 #####
159 ### Simulate 1000 obs for each class and Estimate Prediction Error Using Cross Validation
160 ### Assume the covariates are generated through Log Normal Distribution
161 ### In this case, LDA is misspecified, and we can expect that KDC and NBC
162 ### Should have a better performance in terms of prediction error than LDA
163 ### Objective: Plot the graph for all these three methods, and have a basic impression about
164 ### the difference of these three models
165 #####
166 library(MASS)
167
168 set.seed(123456)
169
170 n<-1000
171
172
173 X1<-mvrnorm(n,mu=c(0,0),Sigma=diag(c(1,1)))
174 X2<-mvrnorm(n,mu=c(-2,-2),Sigma=diag(c(1,1)))
175 X3<-mvrnorm(n,mu=c(2,2),Sigma=diag(c(1,1)))
176
177 X1<-exp(X1)
178 X2<-exp(X2)
179 X3<-exp(X3)
180
181
182 plot(X1[,1],X1[,2],main="Sample Data (Covariates Follow LogNormal)",xlim=c(0,10),ylim=c
      (0,10),xlab="Input 1", ylab="Input 2")
183 points(X2[,1],X2[,2],col=2)
184 points(X3[,1],X3[,2],col=3)
185
186
187 Y<-c(rep(1,n),rep(2,n),rep(3,n))
188
189 X<-rbind(X1,X2,X3)
190
191
192 #
      #####

```

```

193 ### Linear discriminant Analysis
194
195 # Note: This function assumes that the KK classes are encoded as 1:KK
196 LDA.func<-function(xp,Y,Xt){
197   KK<-length(unique(Y))
198   N<-length(Y)
199   p<-length(xp)
200   muk<-matrix(-99,nrow=KK,ncol=p)
201   pik<-rep(-99,KK)
202   Xtc<-Xt
203
204   for(k in 1:KK){
205     muk[k,]<-apply(Xt[Y==k,],2,mean)
206     pik[k]<-mean(Y==k)
207     Xtc[Y==k,]<-t(t(Xt[Y==k,])-muk[k,])
208   }
209   Sig<-t(Xtc)%*%Xtc/(N-KK)
210   Sigi<-solve(Sig)
211   dk<-rep(-99,KK)
212   for(k in 1:KK){
213     dk[k]<-t(xp)%*%Sigi%*%muk[k,] -t(muk[k,])%*%Sigi%*%muk[k,]/2 +log(pik[k])
214   }
215   return(order(dk)[KK])
216 }
217
218
219 grid<-expand.grid(seq(0,10,.1),seq(0,10,.1))
220 part<-apply(grid,1,LDA.func,Y=Y,Xt=X)
221
222 ### Plot
223 plot(grid[,1],grid[,2],col=part,pch=4, xlab="Input 1", ylab="Input 2",main="Linear
  Discriminant Analysis (Covariates Follow LogNormal)",xlim=c(0,10),ylim=c(0,10))
224 points(X1[,1],X1[,2])
225 points(X2[,1],X2[,2],col=2)
226 points(X3[,1],X3[,2],col=3)
227
228
229
230
231 #####
232 ### kernel density classification
233
234 library(ks)
235 library(lattice)
236
237 grid<-expand.grid(seq(0,10,.1),seq(0,10,.1))
238
239 # KK is the number of classes
240 KK<-length(unique(Y))
241 pik<-rep(-99,KK)
242
243 for(k in 1:KK){
244   pik[k]<-mean(Y==k)
245 }
246
247 fhat1 <- kde(x=X1,eval.points=grid)$estimate
248 fhat2 <- kde(x=X2,eval.points=grid)$estimate
249 fhat3 <- kde(x=X3,eval.points=grid)$estimate
250
251 fhat<-cbind(pik[1]*fhat1, pik[2]*fhat2, pik[3]*fhat3)
252
253 part<-rep(NA,length(grid[,1]))
254
255 for (k in 1:length(grid[,1])){
256
257   fhatk=fhat[k,]
258
259   part[k]<-order(fhatk)[3]

```

```

260 }
261 }
262
263
264 ## Plot
265 plot(grid[,1],grid[,2],col=part,pch=4, xlab="Input 1", ylab="Input 2",main="Kernel Density
      Classification (Covariates Follow LogNormal)",xlim=c(0,10),ylim=c(0,10))
266 points(X1[,1],X1[,2])
267 points(X2[,1],X2[,2],col=2)
268 points(X3[,1],X3[,2],col=3)
269
270
271
272
273 #####
274 ### Naive Bayes Classification
275
276 library(ks)
277 library(lattice)
278
279 grid<-expand.grid(seq(0,10,.1),seq(0,10,.1))
280
281 # KK is the number of classes
282 KK<-length(unique(Y))
283 pik<-rep(-99,KK)
284
285 for(k in 1:KK){
286     pik[k]<-mean(Y==k)
287 }
288
289 fhat11 <- kde(x=X1[,1],eval.points=grid[,1])$estimate
290 fhat12 <- kde(x=X1[,2],eval.points=grid[,2])$estimate
291 fhat21 <- kde(x=X2[,1],eval.points=grid[,1])$estimate
292 fhat22 <- kde(x=X2[,2],eval.points=grid[,2])$estimate
293 fhat31 <- kde(x=X3[,1],eval.points=grid[,1])$estimate
294 fhat32 <- kde(x=X3[,2],eval.points=grid[,2])$estimate
295
296 fhat1 <- fhat11*fhat12
297 fhat2 <- fhat21*fhat22
298 fhat3 <- fhat31*fhat32
299
300 fhat<-cbind(pik[1]*fhat1,pik[2]*fhat2, pik[3]*fhat3)
301
302 part<-rep(NA,length(grid[,1]))
303
304 for (k in 1:length(grid[,1])){
305
306     fhatk=fhat[k,]
307
308     part[k]<-order(fhatk)[3]
309 }
310 }
311
312
313 ## Plot
314 plot(grid[,1],grid[,2],col=part,pch=4, xlab="Input 1", ylab="Input 2",main="Naive Bayes
      Classification (Covariates Follow LogNormal)",xlim=c(0,10),ylim=c(0,10))
315 points(X1[,1],X1[,2])
316 points(X2[,1],X2[,2],col=2)
317 points(X3[,1],X3[,2],col=3)
318
319
320 ## Cross-validation of the normal sample across the number of predictors
321
322
323 #####
324 ### Here we will generate p-covariables for each class

```



```

325 ### Simulate observations for each class and Estimate Prediction Error Using Cross
      Validation
326 ### Assume the covariates are generated through Normal Distribution
327 ### In this case, LDA is correctly specified, and we can expect that KDC and NBC
328 ### Should have a worse performance in terms of prediction error than LDA
329 #####
330 library(MASS)
331
332 set.seed(123456)
333
334 n<-100
335 p<-7
336
337 X1<-mvrnorm(n,mu=rep(0,p),Sigma=diag(rep(1,p)))
338 X2<-mvrnorm(n,mu=rep(-2,p),Sigma=diag(rep(1,p)))
339 X3<-mvrnorm(n,mu=rep(2,p),Sigma=diag(rep(1,p)))
340
341
342
343 plot(X1[,1],X1[,2],main="Sample Data (Covariates Follow Normal)",xlim=c(-5,5),ylim=c(-5,5),
      xlab="Input 1", ylab="Input 2")
344 points(X2[,1],X2[,2],col=2)
345 points(X3[,1],X3[,2],col=3)
346
347
348 Y<-c(rep(1,n),rep(2,n),rep(3,n))
349
350 X<-rbind(X1,X2,X3)
351
352 # K-folds
353 K <- 10
354 folds <- sample(1:K, length(X[,1]), replace = TRUE)
355
356
357 #
      #####
358 ### Linear discriminant Analysis
359
360 # Note: This function assumes that the KK classes are encoded as 1:KK
361 LDA.func<-function(xp,Y,Xt){
362   KK<-length(unique(Y))
363   N<-length(Y)
364   p<-length(xp)
365   muk<-matrix(-99,nrow=KK,ncol=p)
366   pik<-rep(-99,KK)
367   Xtc<-Xt
368
369   for(k in 1:KK){
370     muk[k,]<-apply(matrix(Xt[Y==k,],ncol=p),2,mean)
371     pik[k]<-mean(Y==k)
372     Xtc[Y==k,]<-t(t(Xt[Y==k,])-muk[k,])
373   }
374   Sig<-t(Xtc)%*%Xtc/(N-KK)
375   Sigi<-solve(Sig)
376   dk<-rep(-99,KK)
377   for(k in 1:KK){
378     dk[k]<-t(xp)%*%Sigi%*%muk[k,] -t(muk[k,])%*%Sigi%*%muk[k,]/2 +log(pik[k])
379   }
380   return(order(dk)[KK])
381 }
382
383
384 #####
385 # cross validation
386
387 # LDA.cv.errors[i,j] saves the cross validation error for the first i covariates and jth
      fold

```

```

388 LDA.cv.errors <- matrix(NA, nrow=p, ncol=K)
389
390 for (pp in 1:p){
391
392   for (j in 1:K){
393     ii <- which(folds==j)
394     part.test<-apply(matrix(X[ii,1:pp],ncol=pp),1,LDA.func,Y=Y[-ii],Xt=matrix(X[-ii,1:pp],
395                                   ncol=pp))
396     jj<-which(Y[ii]!=part.test)
397     LDA.cv.errors[pp,j] <- length(jj)/(length(ii))
398   }
399 }
400 apply(LDA.cv.errors,1,mean)
401
402
403
404
405 #####
406 ### kernel density classification
407
408 library(ks)
409
410 #####
411 # cross validation
412
413 KDC.cv.errors <- matrix(NA, nrow=p, ncol=K)
414
415 for (pp in 1:p){
416
417   for (i in 1:K){
418     ii <- which(folds==i)
419     Ytest <- Y[ii]
420     Xtest <- X[ii,]
421     Ytrain <- Y[-ii]
422     Xtrain <- X[-ii,]
423
424     ii1 <- which(Ytrain==1)
425     ii2 <- which(Ytrain==2)
426     ii3 <- which(Ytrain==3)
427
428     # KK is the number of classes
429     KK<-length(unique(Ytrain))
430     pik<-rep(-99, KK)
431
432     for(k in 1:KK){
433       pik[k]<-mean(Ytrain==k)
434     }
435
436     fhat1.t <- kde(x=Xtrain[ii1,1:pp],eval.points=Xtest[,1:pp])$estimate
437     fhat2.t <- kde(x=Xtrain[ii2,1:pp],eval.points=Xtest[,1:pp])$estimate
438     fhat3.t <- kde(x=Xtrain[ii3,1:pp],eval.points=Xtest[,1:pp])$estimate
439
440     fhat.t<-cbind(pik[1]*fhat1.t,pik[2]*fhat2.t,pik[3]*fhat3.t)
441     part.test<-rep(NA,length(ii))
442
443     for (j in 1:length(ii)){
444       fhatj.t=fhat.t[j,]
445       part.test[j]<-order(fhatj.t)[3]
446     }
447
448     jj<-which(Ytest!=part.test)
449     KDC.cv.errors[pp,i] <- length(jj)/(length(ii))
450   }
451 }
452
453 apply(KDC.cv.errors,1,mean)
454

```

```

455
456
457
458 #####
459 ### Naive Bayes Classification
460
461 library(ks)
462
463
464
465 #####
466 # cross validation
467
468 NBC.cv.errors <- matrix(NA, nrow=p, ncol=K)
469
470 for (pp in 1:p){
471
472   for (i in 1:K){
473
474     ii <- which(folds==i)
475     Ytest <- Y[ii]
476     Xtest <- X[ii,]
477     Ytrain <- Y[-ii]
478     Xtrain <- X[-ii,]
479
480     ii1 <- which(Ytrain==1)
481     ii2 <- which(Ytrain==2)
482     ii3 <- which(Ytrain==3)
483
484     # KK is the number of classes
485     KK<-length(unique(Ytrain))
486     pik<-rep(-99, KK)
487
488     for(k in 1:KK){
489       pik[k]<-mean(Ytrain==k)
490     }
491
492     fhat1.t <- kde(x=Xtrain[ii1,1], eval.points=Xtest[,1])$estimate
493     fhat2.t <- kde(x=Xtrain[ii2,1], eval.points=Xtest[,1])$estimate
494     fhat3.t <- kde(x=Xtrain[ii3,1], eval.points=Xtest[,1])$estimate
495
496     for (k in 2:pp){
497
498       fhat1k <- kde(x=Xtrain[ii1,k], eval.points=Xtest[,k])$estimate
499       fhat2k <- kde(x=Xtrain[ii2,k], eval.points=Xtest[,k])$estimate
500       fhat3k <- kde(x=Xtrain[ii3,k], eval.points=Xtest[,k])$estimate
501
502       fhat1.t <- fhat1.t*fhat1k
503       fhat2.t <- fhat2.t*fhat2k
504       fhat3.t <- fhat3.t*fhat3k
505
506     }
507
508     fhat.t<-cbind(pik[1]*fhat1.t,pik[2]*fhat2.t,pik[3]*fhat3.t)
509     part.test<-rep(NA,length(ii))
510
511     for (j in 1:length(ii)){
512       fhatj.t=fhat.t[j,]
513       part.test[j]<-order(fhatj.t)[3]
514     }
515
516     jj<-which(Ytest!=part.test)
517     NBC.cv.errors[pp,i] <- length(jj)/(length(ii))
518   }
519 }
520
521 apply(NBC.cv.errors,1,mean)
522

```

```

523 #####
524
525 # check the outcome
526 print(apply(LDA.cv.errors,1,mean))
527 print(apply(KDC.cv.errors,1,mean))
528 print(apply(NBC.cv.errors,1,mean))
529
530
531 ## Plot
532 LDA<-apply(LDA.cv.errors,1,mean)
533 KDC<-apply(KDC.cv.errors,1,mean)
534 NBC<-apply(NBC.cv.errors,1,mean)
535
536 number<-seq(1,p,by=1)
537
538
539 plot(number, LDA, type="o", lty=1, col=1,main="Models Comparasion (Covariates Follow Normal,
      n=100, K=3)", xlab="p", ylab="Cross Validition Errors", ylim=c(0,0.3))
540 points(number, KDC, pch="*", col=2)
541 lines(number, KDC, lty=2, col=2)
542 points(number, NBC, pch="+", col=3)
543 lines(number, NBC, lty=3, col=3)
544
545 legend(6,0.3,legend=c("LDA", "KDC", "NBC"),col=1:3, pch=c("o","*","+"), lty=c(1,2,3), ncol
      =1)
546
547
548 ## Cross-validation of the log-normal sample across the number of predictors
549
550 #
      #####
551
552 ### Here we will generate p-covariables for each class
553 ### Simulate observations for each class and Estimate Prediction Error Using Cross
      Validation
554 ### Assume the covariates are generated through Log Normal Distribution
555 ### In this case, LDA is misspecified, and we can expect that KDC and NBC
556 ### Should have a better performance in terms of prediction error than LDA
557 #
      #####
558
559 library(MASS)
560
561 set.seed(123456)
562
563 # number of observations for each class
564 n<-100
565 # number of covariates
566 p<-7
567
568 X1<-mvrnorm(n,mu=rep(0,p),Sigma=diag(rep(1,p)))
569 X2<-mvrnorm(n,mu=rep(-2,p),Sigma=diag(rep(1,p)))
570 X3<-mvrnorm(n,mu=rep(2,p),Sigma=diag(rep(1,p)))
571
572 # exponential covariates, so that all the covariates follow Log Normal Distribution
573 X1<-exp(X1)
574 X2<-exp(X2)
575 X3<-exp(X3)
576
577 plot(X1[,1],X1[,2],main="Sample Data",xlim=c(0,10),ylim=c(0,10),xlab="Input 1", ylab="Input
      2")
578 points(X2[,1],X2[,2],col=2)
579 points(X3[,1],X3[,2],col=3)
580
581 Y<-c(rep(1,n),rep(2,n),rep(3,n))
582

```

```

583 X<-rbind(X1,X2,X3)
584
585 # K-folds, for CV
586 K <- 10
587 folds <- sample(1:K, length(X[,1]), replace = TRUE)
588
589
590 #
#####

591 ### Linear discriminant Analysis
592
593 # Note: This function assumes that the KK classes are encoded as 1:KK
594 LDA.func<-function(xp,Y,Xt){
595   KK<-length(unique(Y))
596   N<-length(Y)
597   p<-length(xp)
598   muk<-matrix(-99,nrow=KK,ncol=p)
599   pik<-rep(-99,KK)
600   Xtc<-Xt
601
602   for(k in 1:KK){
603     muk[k,]<-apply(matrix(Xt[Y==k,],ncol=p),2,mean)
604     pik[k]<-mean(Y==k)
605     Xtc[Y==k,]<-t(t(Xt[Y==k,])-muk[k,])
606   }
607   Sig<-t(Xtc)%*%Xtc/(N-KK)
608   Sigi<-solve(Sig)
609   dk<-rep(-99,KK)
610   for(k in 1:KK){
611     dk[k]<-t(xp)%*%Sigi%*%muk[k,] -t(muk[k,])%*%Sigi%*%muk[k,]/2 +log(pik[k])
612   }
613   return(order(dk)[KK])
614 }
615
616
617 #####
618 # cross validation
619
620 # LDA.cv.errors[i,j] saves the cross validation error for the first i covariates and jth
621   fold
622 LDA.cv.errors <- matrix(NA, nrow=p, ncol=K)
623
624 for (pp in 1:p){
625
626   for (j in 1:K){
627     ii <- which(folds==j)
628     part.test<-apply(matrix(X[ii,1:pp],ncol=pp),1,LDA.func,Y=Y[-ii],Xt=matrix(X[-ii,1:pp],
629       ncol=pp))
630     jj<-which(Y[ii]!=part.test)
631     LDA.cv.errors[pp,j] <- length(jj)/(length(ii))
632   }
633 }
634 apply(LDA.cv.errors,1,mean)
635
636
637
638
639 #####
640 ### kernel density classification
641
642 library(ks)
643
644
645 #####
646 # cross validation

```

```

647
648 KDC.cv.errors <- matrix(NA, nrow=p, ncol=K)
649
650 for (pp in 1:p){
651
652   for (i in 1:K){
653     ii <- which(folds==i)
654     Ytest <- Y[ii]
655     Xtest <- X[ii,]
656     Ytrain <- Y[-ii]
657     Xtrain <- X[-ii,]
658
659     ii1 <- which(Ytrain==1)
660     ii2 <- which(Ytrain==2)
661     ii3 <- which(Ytrain==3)
662
663     # KK is the number of classes
664     KK<-length(unique(Ytrain))
665     pik<-rep(-99, KK)
666
667     for(k in 1:KK){
668       pik[k]<-mean(Ytrain==k)
669     }
670
671     fhat1.t <- kde(x=Xtrain[ii1,1:pp], eval.points=Xtest[,1:pp])$estimate
672     fhat2.t <- kde(x=Xtrain[ii2,1:pp], eval.points=Xtest[,1:pp])$estimate
673     fhat3.t <- kde(x=Xtrain[ii3,1:pp], eval.points=Xtest[,1:pp])$estimate
674
675     fhat.t<-cbind(pik[1]*fhat1.t, pik[2]*fhat2.t, pik[3]*fhat3.t)
676     part.test<-rep(NA, length(ii))
677
678     for (j in 1:length(ii)){
679       fhatj.t=fhat.t[j,]
680       part.test[j]<-order(fhatj.t)[3]
681     }
682
683     jj<-which(Ytest!=part.test)
684     KDC.cv.errors[pp,i] <- length(jj)/(length(ii))
685   }
686 }
687
688 apply(KDC.cv.errors, 1, mean)
689
690
691
692
693 #####
694 ### Naive Bayes Classification
695
696 library(ks)
697
698
699
700 #####
701 # cross validation
702
703 NBC.cv.errors <- matrix(NA, nrow=p, ncol=K)
704
705 for (pp in 1:p){
706
707   for (i in 1:K){
708
709     ii <- which(folds==i)
710     Ytest <- Y[ii]
711     Xtest <- X[ii,]
712     Ytrain <- Y[-ii]
713     Xtrain <- X[-ii,]
714

```

```

715     ii1 <- which(Ytrain==1)
716     ii2 <- which(Ytrain==2)
717     ii3 <- which(Ytrain==3)
718
719     # KK is the number of classes
720     KK<-length(unique(Ytrain))
721     pik<-rep(-99, KK)
722
723     for(k in 1:KK){
724       pik[k]<-mean(Ytrain==k)
725     }
726
727     fhat1.t <- kde(x=Xtrain[ii1,1], eval.points=Xtest[,1])$estimate
728     fhat2.t <- kde(x=Xtrain[ii2,1], eval.points=Xtest[,1])$estimate
729     fhat3.t <- kde(x=Xtrain[ii3,1], eval.points=Xtest[,1])$estimate
730
731     for (k in 2:pp){
732
733       fhat1k <- kde(x=Xtrain[ii1,k], eval.points=Xtest[,k])$estimate
734       fhat2k <- kde(x=Xtrain[ii2,k], eval.points=Xtest[,k])$estimate
735       fhat3k <- kde(x=Xtrain[ii3,k], eval.points=Xtest[,k])$estimate
736
737       fhat1.t <- fhat1.t*fhat1k
738       fhat2.t <- fhat2.t*fhat2k
739       fhat3.t <- fhat3.t*fhat3k
740
741     }
742
743     fhat.t<-cbind(pik[1]*fhat1.t,pik[2]*fhat2.t,pik[3]*fhat3.t)
744     part.test<-rep(NA,length(ii))
745
746     for (j in 1:length(ii)){
747       fhatj.t=fhat.t[j,]
748       part.test[j]<-order(fhatj.t)[3]
749     }
750
751     jj<-which(Ytest!=part.test)
752     NBC.cv.errors[pp,i] <- length(jj)/(length(ii))
753 }
754 }
755
756 apply(NBC.cv.errors,1,mean)
757
758 #####
759
760 # check the outcome
761 print(apply(LDA.cv.errors,1,mean))
762 print(apply(KDC.cv.errors,1,mean))
763 print(apply(NBC.cv.errors,1,mean))
764
765 ## Plot
766 LDA<-apply(LDA.cv.errors,1,mean)
767 KDC<-apply(KDC.cv.errors,1,mean)
768 NBC<-apply(NBC.cv.errors,1,mean)
769
770 number<-seq(1,p,by=1)
771
772
773 plot(number, LDA, type="o", lty=1, col=1,main="Models Comparasion (Covariates Follow
      LogNormal, n=100, K=3)", xlab="p", ylab="Cross Validation Errors", ylim=c(0,0.6))
774 points(number, KDC, pch="*", col=2)
775 lines(number, KDC, lty=2, col=2)
776 points(number, NBC, pch="+", col=3)
777 lines(number, NBC, lty=3, col=3)
778
779 legend(6,0.6,legend=c("LDA", "KDC", "NBC"),col=1:3, pch=c("o","*","+"), lty=c(1,2,3), ncol
      =1)
780

```

```

781 |
782 |
783 | # R Code for Data Application
784 |
785 | # --- (0) Data -----
786 |
787 | data <- read.table(choose.files(), sep=",", header=TRUE)
788 | # Heart data
789 | head(data)
790 |
791 | panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...)
792 | {
793 |   usr <- par("usr"); on.exit(par(usr))
794 |   par(usr = c(0, 1, 0, 1))
795 |   r <- abs(cor(x, y))
796 |   txt <- format(c(r, 0.123456789), digits = digits)[1]
797 |   txt <- paste0(prefix, txt)
798 |   if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
799 |   text(0.5, 0.5, txt, cex = cex.cor * r)
800 | }
801 | pairs(data[,c(2:4,6,8:11)], pch=c(21,22)[unclass(factor(data$chd))], bg = c("black","red")[
802 |   unclass(factor(data$chd))], lower.panel = panel.cor,
803 |   gap=0.25, rowlattop=FALSE)
804 |
805 | par(mfrow=c(3,2))
806 | for ( i in c(2:4,8:10)){
807 |   plot(density(data[,i]), main = paste0(colnames(data)[i]))
808 | }
809 |
810 |
811 | data[,11] <- data[,11] + 1 # 0->1; 1->2
812 |
813 | X <- cbind(data[,2:4],data[,6]=="Present",data[,8:10])
814 | X <- as.matrix(X)
815 | Y <- as.numeric(data[,11])
816 |
817 | G <- unique(Y)
818 | K <- 6 # 6-folds
819 | n <- dim(X)[1]
820 | p <- dim(X)[2]
821 |
822 |
823 |
824 | # --- (1) LDA -----
825 |
826 | LDA.func<-function(xp,Y,Xt){
827 |   KK<-length(unique(Y))
828 |   N<-length(Y)
829 |   p<-length(xp)
830 |   muk<-matrix(-99,nrow=KK,ncol=p)
831 |   pik<-rep(-99,KK)
832 |   Xtc<-Xt
833 |
834 |   for(k in 1:KK){
835 |     muk[k,]<-apply(matrix(Xt[Y==k,],ncol=p),2,mean)
836 |     pik[k]<-mean(Y==k)
837 |     Xtc[Y==k,]<-t(t(Xt[Y==k,])-muk[k,])
838 |   }
839 |   Sig<-t(Xtc)%*%Xtc/(N-KK)
840 |   Sigi<-solve(Sig)
841 |   dk<-rep(-99,KK)
842 |   for(k in 1:KK){
843 |     dk[k]<-t(xp)%*%Sigi%*%muk[k,] -t(muk[k,])%*%Sigi%*%muk[k,]/2 +log(pik[k])
844 |   }
845 |   return(order(dk)[KK])
846 | }
847 |

```



```

848 LDA.test.error <- rep(-99, K) # K number of test error
849 set.seed(12345)
850 test.id <- split(sample(1:n),1:K) # A list of indices to set aside test data
851 for (k in 1:K){
852   X.test <- X[test.id[[k]],]
853   X.train <- X[-test.id[[k]],]
854   Y.test <- Y[test.id[[k]]]
855   Y.train <- Y[-test.id[[k]]]
856
857   Y.test.hat <- rep(-99, length(Y.test)) # Record for a vector of log-likelihood fits
858   for (r in 1:length(Y.test)){
859     temp <- LDA.func(X.test[r,], Y=Y.train, Xt=X.train)
860     Y.test.hat[r] <- temp
861   }
862   LDA.test.error[k] <- sum(Y.test!=Y.test.hat)/length(Y.test)
863 }
864
865 LDA.CV.score <- mean(LDA.test.error)
866
867 # --- (2) KDC -----
868
869 library(ks)
870
871 KDC.cv.errors <- rep(-99,K)
872
873 for (i in 1:K){
874   ii <- test.id[[i]]
875   Ytest <- Y[ii]
876   Xtest <- X[ii,]
877   Ytrain <- Y[-ii]
878   Xtrain <- X[-ii,]
879
880   ii1 <- which(Ytrain==1)
881   ii2 <- which(Ytrain==2)
882
883   # KK is the number of classes
884   KK<-length(G)
885   pik<-rep(-99, KK)
886
887   for(k in 1:KK){
888     pik[k]<-mean(Ytrain==k)
889   }
890
891   fhat1.t <- kde(x=Xtrain[ii1,], eval.points=Xtest)$estimate
892   fhat2.t <- kde(x=Xtrain[ii2,], eval.points=Xtest)$estimate
893
894   fhat.t<-cbind(pik[1]*fhat1.t, pik[2]*fhat2.t)
895   part.test<-rep(NA, length(ii))
896
897   for (j in 1:length(ii)){
898     fhatj.t=fhat.t[j,]
899     part.test[j]<-order(fhatj.t)[2]
900   }
901
902   jj<-which(Ytest!=part.test)
903   KDC.cv.errors[i] <- length(jj)/(length(ii))
904 }
905
906 KDC.CV.score <- mean(KDC.cv.errors)
907
908 # --- (3) NBC -----
909
910 library(klaR)
911 #library(stats) # density(data$sbp, kernel = "epanechnikov")
912

```

```

916 NBC.test.error <- rep(-99, K) # K number of test error
917 for (k in 1:K){
918   X.test <- X[test.id[[k]],]
919   X.train <- X[-test.id[[k]],]
920   Y.test <- Y[test.id[[k]]]
921   Y.train <- Y[-test.id[[k]]]
922
923   Y.train <- as.factor(Y.train)
924   Y.test <- as.factor(Y.test)
925
926   NBC.model <- NaiveBayes(X.train, Y.train, usekernel=T)
927   Y.test.hat <- predict(NBC.model, X.test, threshold = 1e-4 )
928
929   NBC.test.error[k] <- sum(Y.test!=Y.test.hat$class)/length(Y.test)
930 }
931
932 NBC.CV.score <- mean(NBC.test.error)
933
934
935 c(LDA.CV.score, KDC.CV.score, NBC.CV.score)
936
937 X <- cbind(data[,2:4],factor(data[,6]),data[,8:10])
938 Y <- as.factor(data[,11])
939
940 NBC.fit <- NaiveBayes(X,Y,usekernel=T)
941
942 par(mfrow=c(3,2))
943 plot(NBC.fit,ylab = "Density",vars = "sbp", main = "Class-conditional Marginal Density",
944       col = c("black","red"), legendplot = F)
945 legend("topright", legend = c("chd=0","chd=1"), col = c("black","red"), lty = 1:2)
946 for (p in colnames(data[,c(3:4,6,8:10)])){
947   plot(NBC.fit,ylab = "Density",vars = p, main = "", col = c("black","red"), legendplot = F)
948 }

```