(MATH 9810 Final Project)

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 Dennsity 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 $x_1, \cdots, x_N$ 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.

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

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).

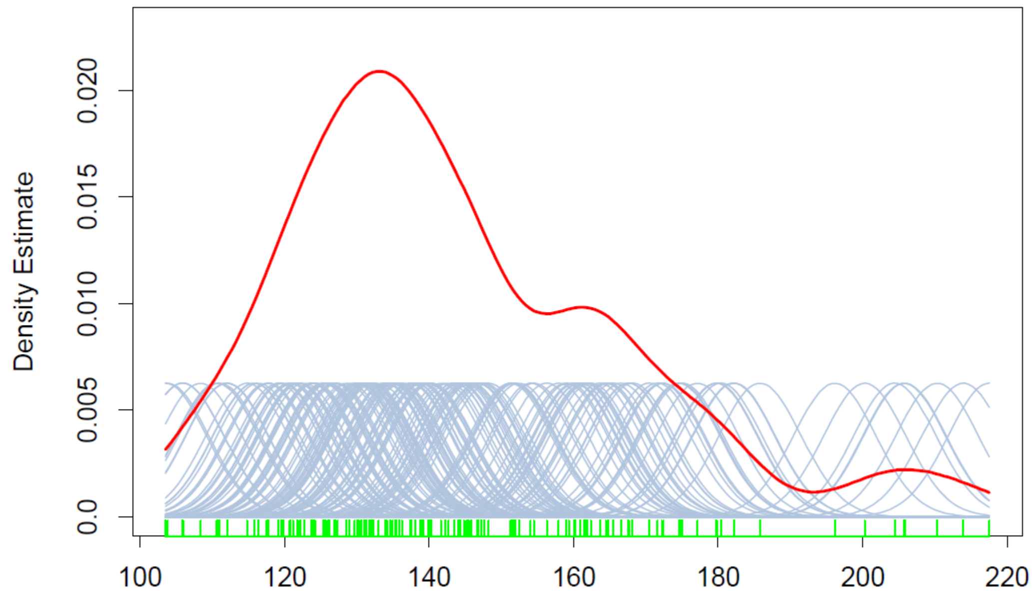


Figure 1: A kernel density estimate

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

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

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

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

where $\hat{f}_k(x_0)$ is estimated density at x_0 based on a kernel density fit involving only observations from kth class and usually, $\hat{\pi}_j$ is sample proportion falling into jth 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,

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

* where $\hat{f}_{li}$ is an estimate of the density of the ith variable in lth 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).

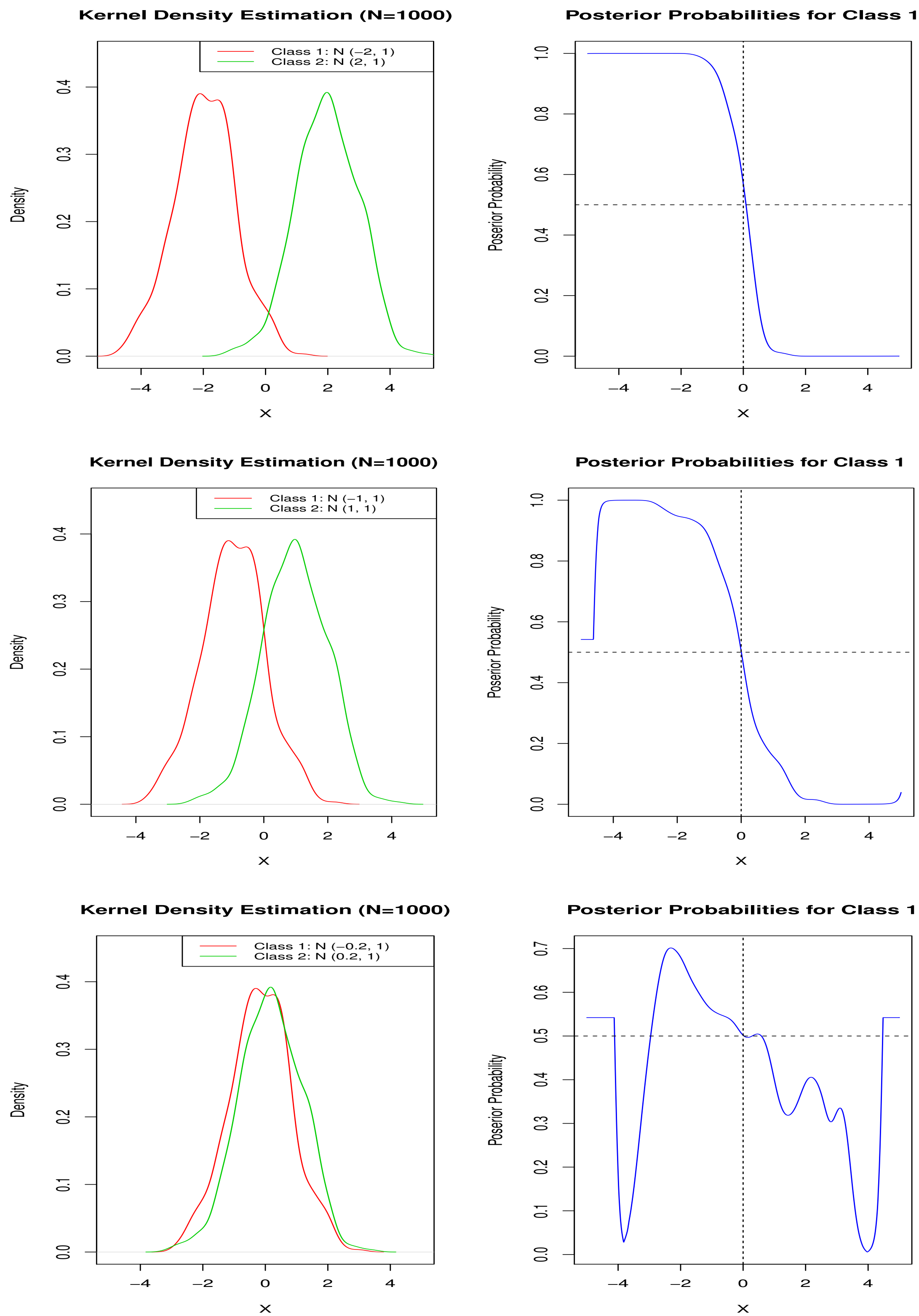

Kernel Density Estimation (N=1000)
Class 1: N (−2, 1)
Class 2: N (2, 1)
Density
x
Posterior Probabilities for Class 1
Poserior Probability
x
Kernel Density Estimation (N=1000)
Class 1: N (−1, 1)
Class 2: N (1, 1)
Density
x
Posterior Probabilities for Class 1
Poserior Probability
x
Kernel Density Estimation (N=1000)
Class 1: N (−0.2, 1)
Class 2: N (0.2, 1)
Density
x
Posterior Probabilities for Class 1
Poserior Probability
x

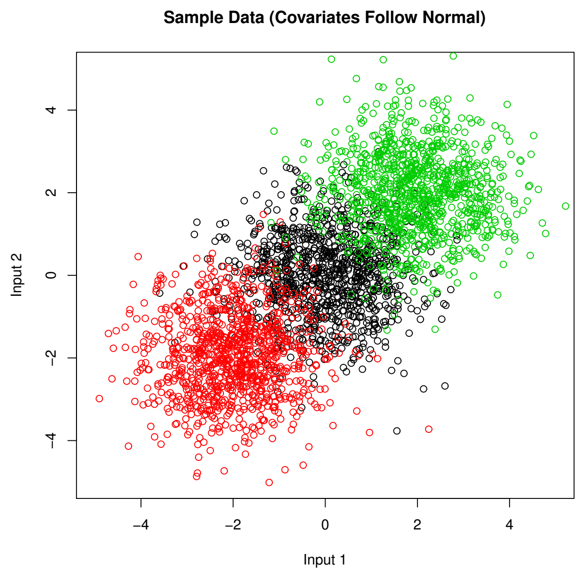
Figure 2:

2 Simulation

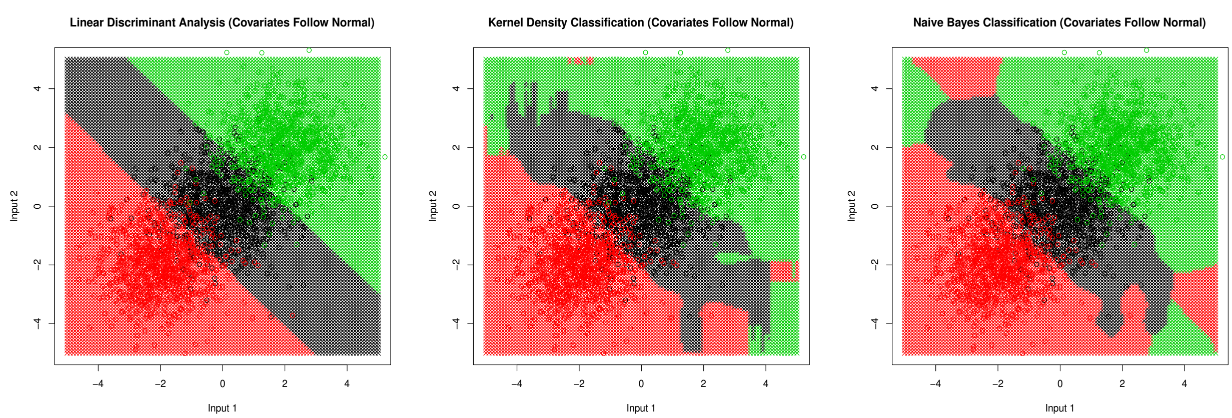
To see the difference between the Linear Discriminant Analysis (LDA), the Kernel Density Classification (KDC) and the Naive Bayes Classifier (NBC), we generate 3 classes of Y with 2 covariates X_1 and X_2 as like in class. We are considering the following two cases, 1. X_1 and X_2 are following normal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

Case 1. X_1 and X_2 are following normal distribution.

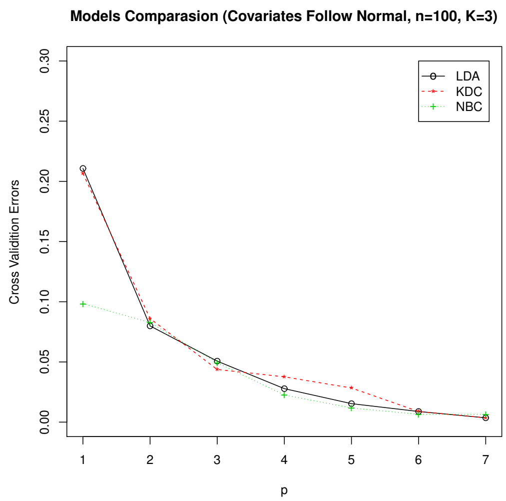
- **The distribution of sample dataset.**



- **Applying LDA, KDC and NBC.**



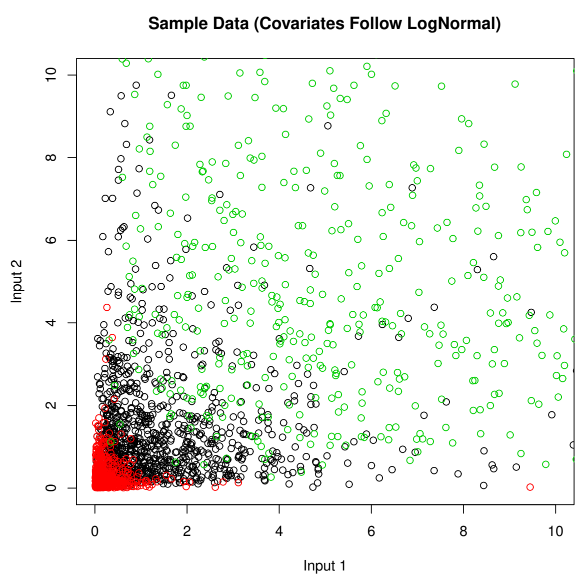
- **Results**
  To see the results more clearly, we've done the cross-validation as following.



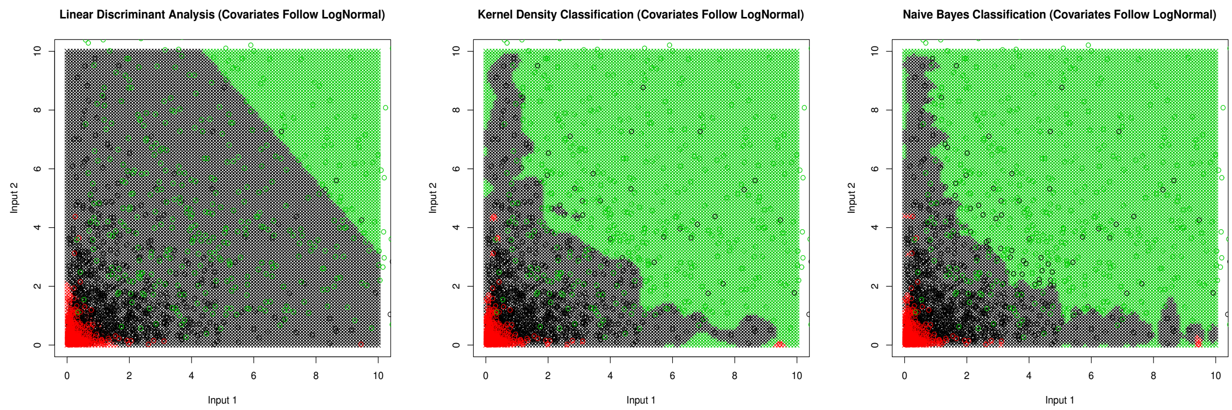
Overall, NBC is performing better than LDA and KDC since we are assuming the 2 covariates are independent when we generate the dataset for simulation. Also, as p is increased, that is as the number of covariates is increased, LDA is showing better cross validation error than KDC. It is because in LDA 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.**

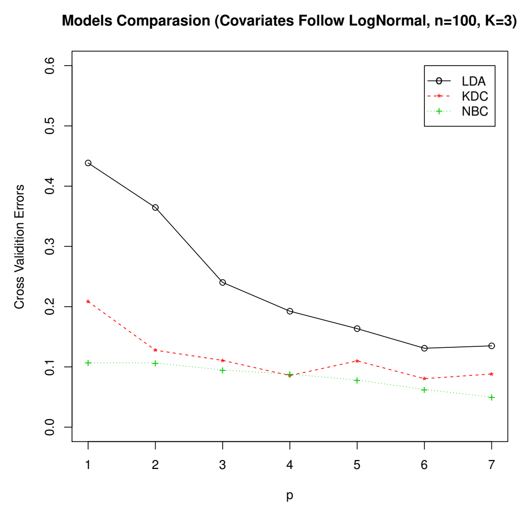
- **The distribution of sample dataset.**



- **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 densiity 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*.

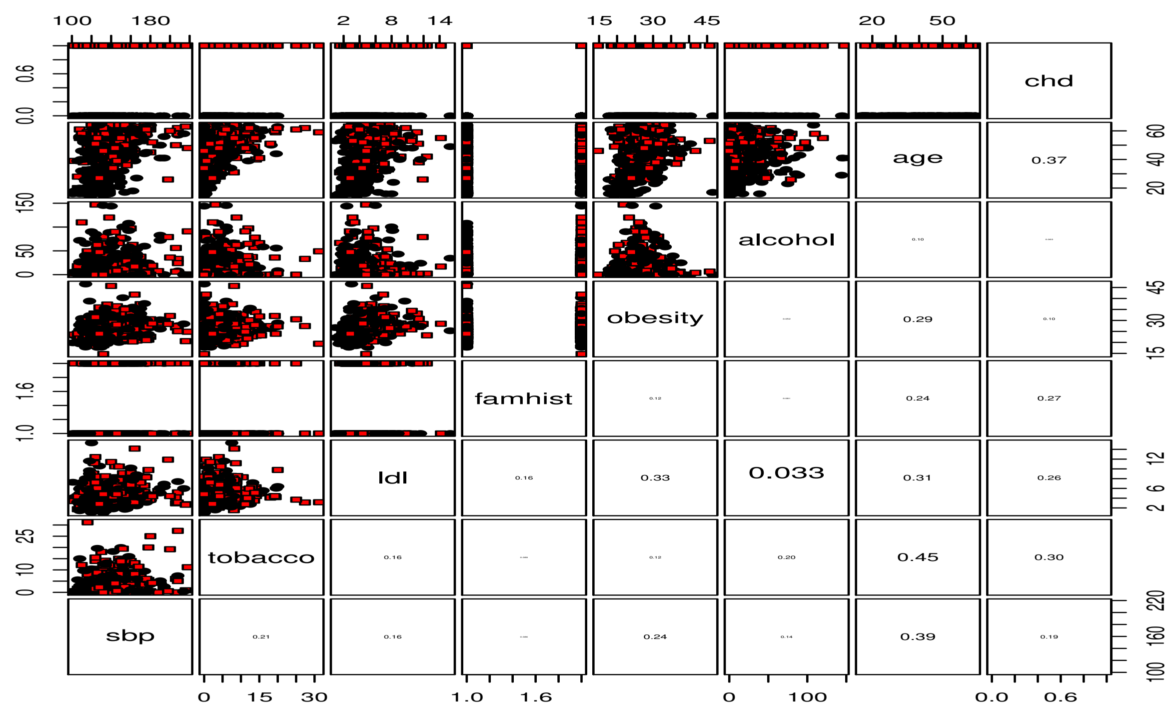


Figure 3: Scattor plot matrix of data

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.

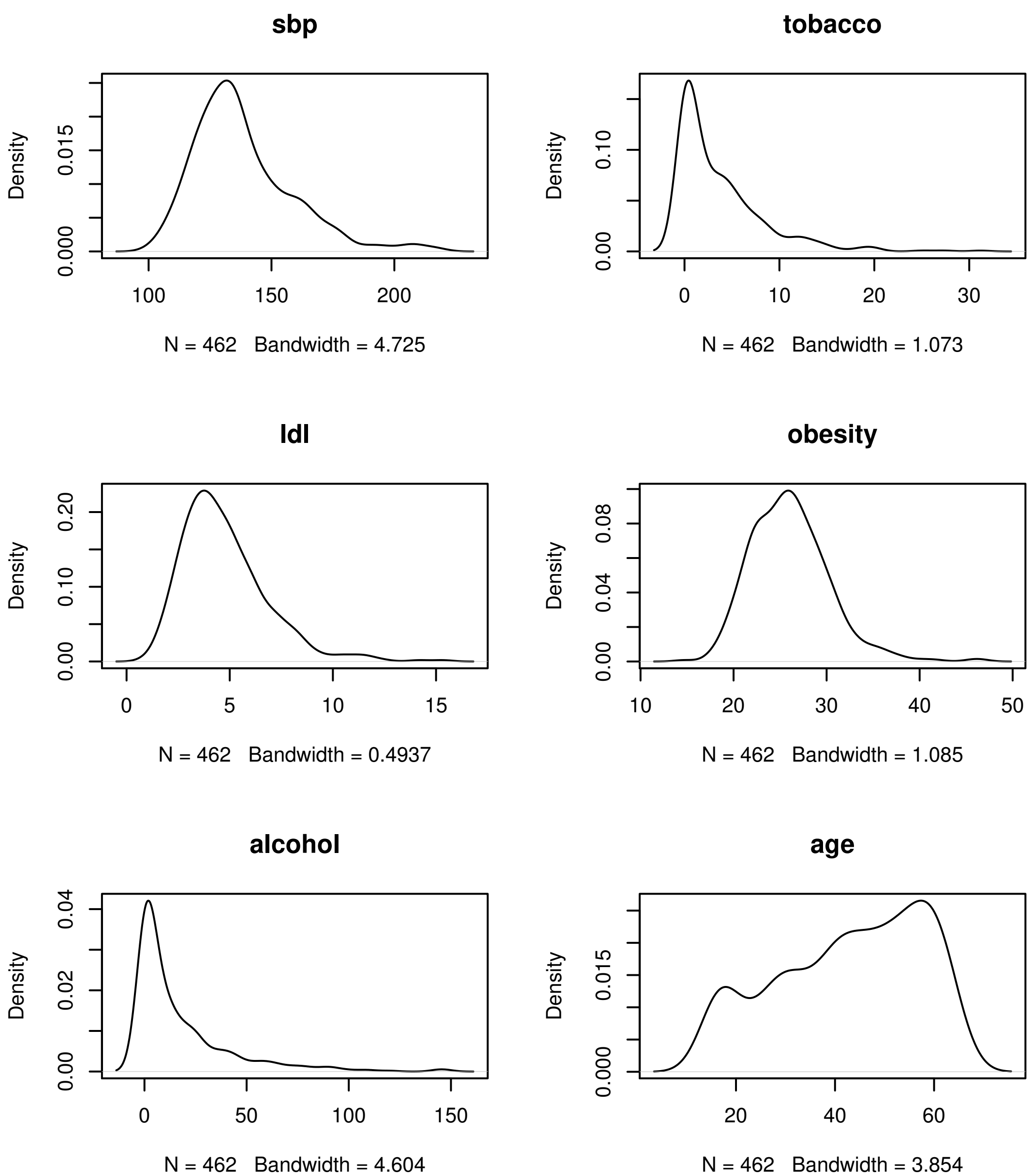


Figure 4: Kernel density of covariates

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.

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

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

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.

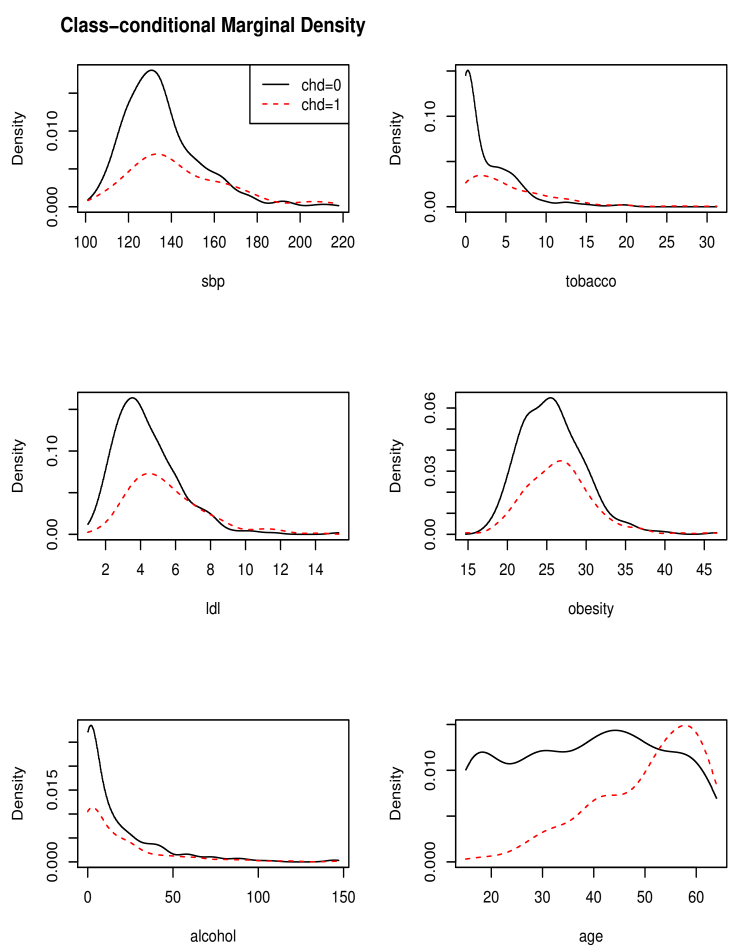


Figure 5: Naive Bayes Marginal Density

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

```

# R Code for Simulation

## Scatter plots of simulation (sampling from normal distribution)

library(MASS)

set.seed(123456)

n<-1000


X1<-mvrnorm(n,mu=c(0,0),Sigma=diag(c(1,1)))
X2<-mvrnorm(n,mu=c(-2,-2),Sigma=diag(c(1,1)))
X3<-mvrnorm(n,mu=c(2,2),Sigma=diag(c(1,1)))


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")
points(X2[,1],X2[,2],col=2)
points(X3[,1],X3[,2],col=3)


Y<-c(rep(1,n),rep(2,n),rep(3,n))

X<-rbind(X1,X2,X3)

# #############################################################################################
### Linear discriminant Analysis

# Note: This function asssumes that the KK classes are encoded as 1:KK
LDA.func<-function(xp,Y,Xt){
  KK<-length(unique(Y))
  N<-length(Y)
  p<-length(xp)
  muk<-matrix(-99,nrow=KK,ncol=p)
  pik<-rep(-99,KK)
  Xtc<-Xt

  for(k in 1:KK){
    muk[k,]<-apply(Xt[Y==k,],2,mean)
    pik[k]<-mean(Y==k)
    Xtc[Y==k,]<-t(t(Xt[Y==k,])-muk[k,])
  }
  Sig<-t(Xtc)%*%Xtc/(N-KK)
  Sigi<-solve(Sig)
  dk<-rep(-99,KK)
  for(k in 1:KK){
     dk[k]<-t(xp)%*%Sigi%*%muk[k,]    -t(muk[k,])%*%Sigi%*%muk[k,]/2 +log(pik[k])
  }
  return(order(dk)[KK])
}


grid<-expand.grid(seq(-5,5,.1),seq(-5,5,.1))
part<-apply(grid,1,LDA.func,Y=Y,Xt=X)

### Plot
plot(grid[,1],grid[,2],col=part,pch=4, xlab="Input 1", ylab="Input 2",main="Linear Discriminant Analysis (Covariates Follow Normal)",xlim=c(-5,5),ylim=c(-5,5))
points(X1[,1],X1[,2])
points(X2[,1],X2[,2],col=2)
points(X3[,1],X3[,2],col=3)
```

```
################################################################################
### kernel density classification

library(ks)
library(lattice)

grid<-expand.grid(seq(-5,5,.1),seq(-5,5,.1))

# KK is the number of classes
KK<-length(unique(Y))
pik<-rep(-99,KK)

for(k in 1:KK){
    pik[k]<-mean(Y==k)
}

fhat1 <- kde(x=X1,eval.points=grid)$estimate
fhat2 <- kde(x=X2,eval.points=grid)$estimate
fhat3 <- kde(x=X3,eval.points=grid)$estimate

fhat<-cbind(pik[1]*fhat1, pik[2]*fhat2, pik[3]*fhat3)

part<-rep(NA,length(grid[,1]))

for (k in 1:length(grid[,1])){

fhatk=fhat[k,]

part[k]<-order(fhatk)[3]

}


## Plot
plot(grid[,1],grid[,2],col=part,pch=4, xlab="Input 1", ylab="Input 2",main="Kernel Density
    Classification (Covariates Follow Normal)",xlim=c(-5,5),ylim=c(-5,5))
points(X1[,1],X1[,2])
points(X2[,1],X2[,2],col=2)
points(X3[,1],X3[,2],col=3)






################################################################################
### Naive Bayes Classification

library(ks)
library(lattice)

grid<-expand.grid(seq(-5,5,.1),seq(-5,5,.1))

# K is the number of classes
KK<-length(unique(Y))
pik<-rep(-99,KK)

for(k in 1:KK){
    pik[k]<-mean(Y==k)
}

fhat11 <- kde(x=X1[,1],eval.points=grid[,1])$estimate
fhat12 <- kde(x=X1[,2],eval.points=grid[,2])$estimate
fhat21 <- kde(x=X2[,1],eval.points=grid[,1])$estimate
fhat22 <- kde(x=X2[,2],eval.points=grid[,2])$estimate
fhat31 <- kde(x=X3[,1],eval.points=grid[,1])$estimate
```

```
fhat32 <- kde(x=X3[,2],eval.points=grid[,2])$estimate

fhat1 <- fhat11*fhat12
fhat2 <- fhat21*fhat22
fhat3 <- fhat31*fhat32

fhat<-cbind(pik[1]*fhat1,pik[2]*fhat2, pik[3]*fhat3)

part<-rep(NA,length(grid[,1]))

for (k in 1:length(grid[,1])){

fhatk=fhat[k,]

part[k]<-order(fhatk)[3]

}


### Plot
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))
points(X1[,1],X1[,2])
points(X2[,1],X2[,2],col=2)
points(X3[,1],X3[,2],col=3)


## Scatter plots of simulation (sampling from log-normal distribution)

 {r}
########################################################################################
### Simulate 1000 obs for each class and Estimate Prediction Error Using Cross Validation
### Assume the covariates are generated through Log Normal Distribution
### In this case, LDA is misspecified, and we can expect that KDC and NBC
### Should have a better performance in terms of prediction error than LDA
### Objective: Plot the graph for all these three methods, and have a basic impression about
### the difference of these three models
########################################################################################
library(MASS)

set.seed(123456)

n<-1000


X1<-mvrnorm(n,mu=c(0,0),Sigma=diag(c(1,1)))
X2<-mvrnorm(n,mu=c(-2,-2),Sigma=diag(c(1,1)))
X3<-mvrnorm(n,mu=c(2,2),Sigma=diag(c(1,1)))

X1<-exp(X1)
X2<-exp(X2)
X3<-exp(X3)


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")
points(X2[,1],X2[,2],col=2)
points(X3[,1],X3[,2],col=3)


Y<-c(rep(1,n),rep(2,n),rep(3,n))

X<-rbind(X1,X2,X3)


#
    ########################################################################################
```

```
### Linear discriminant Analysis

# Note: This function asssumes that the KK classes are encoded as 1:KK
LDA.func<-function(xp,Y,Xt){
  KK<-length(unique(Y))
  N<-length(Y)
  p<-length(xp)
  muk<-matrix(-99,nrow=KK,ncol=p)
  pik<-rep(-99,KK)
  Xtc<-Xt

  for(k in 1:KK){
    muk[k,]<-apply(Xt[Y==k,],2,mean)
    pik[k]<-mean(Y==k)
    Xtc[Y==k,]<-t(t(Xt[Y==k,])-muk[k,])
  }
  Sig<-t(Xtc)%*%Xtc/(N-KK)
  Sigi<-solve(Sig)
  dk<-rep(-99,KK)
  for(k in 1:KK){
     dk[k]<-t(xp)%*%Sigi%*%muk[k,]    -t(muk[k,])%*%Sigi%*%muk[k,]/2 +log(pik[k])
  }
  return(order(dk)[KK])
}


grid<-expand.grid(seq(0,10,.1),seq(0,10,.1))
part<-apply(grid,1,LDA.func,Y=Y,Xt=X)

### Plot
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))
points(X1[,1],X1[,2])
points(X2[,1],X2[,2],col=2)
points(X3[,1],X3[,2],col=3)




##########################################################################################
### kernel density classification

library(ks)
library(lattice)

grid<-expand.grid(seq(0,10,.1),seq(0,10,.1))

# KK is the number of classes
KK<-length(unique(Y))
pik<-rep(-99,KK)

for(k in 1:KK){
    pik[k]<-mean(Y==k)
}

fhat1 <- kde(x=X1,eval.points=grid)$estimate
fhat2 <- kde(x=X2,eval.points=grid)$estimate
fhat3 <- kde(x=X3,eval.points=grid)$estimate

fhat<-cbind(pik[1]*fhat1, pik[2]*fhat2, pik[3]*fhat3)

part<-rep(NA,length(grid[,1]))

for (k in 1:length(grid[,1])){

fhatk=fhat[k,]

part[k]<-order(fhatk)[3]
```

```
}


## Plot
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))
points(X1[,1],X1[,2])
points(X2[,1],X2[,2],col=2)
points(X3[,1],X3[,2],col=3)




#################################################################################
### Naive Bayes Classification

library(ks)
library(lattice)

grid<-expand.grid(seq(0,10,.1),seq(0,10,.1))

# KK is the number of classes
KK<-length(unique(Y))
pik<-rep(-99,KK)

for(k in 1:KK){
    pik[k]<-mean(Y==k)
}

fhat11 <- kde(x=X1[,1],eval.points=grid[,1])$estimate
fhat12 <- kde(x=X1[,2],eval.points=grid[,2])$estimate
fhat21 <- kde(x=X2[,1],eval.points=grid[,1])$estimate
fhat22 <- kde(x=X2[,2],eval.points=grid[,2])$estimate
fhat31 <- kde(x=X3[,1],eval.points=grid[,1])$estimate
fhat32 <- kde(x=X3[,2],eval.points=grid[,2])$estimate

fhat1 <- fhat11*fhat12
fhat2 <- fhat21*fhat22
fhat3 <- fhat31*fhat32

fhat<-cbind(pik[1]*fhat1,pik[2]*fhat2, pik[3]*fhat3)

part<-rep(NA,length(grid[,1]))

for (k in 1:length(grid[,1])){

fhatk=fhat[k,]

part[k]<-order(fhatk)[3]

}


### Plot
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))
points(X1[,1],X1[,2])
points(X2[,1],X2[,2],col=2)
points(X3[,1],X3[,2],col=3)


## Cross-validation of the normal sample across the number of predictors


###################################################################################
### Here we will generate p-covariables for each class
```

```
### Simulate observations for each class and Estimate Prediction Error Using Cross Validation
### Assume the covariates are generated through Normal Distribution
### In this case, LDA is correctly specified, and we can expect that KDC and NBC
### Should have a worse performance in terms of prediction error than LDA
#########################################################################################
library(MASS)

set.seed(123456)

n<-100
p<-7

X1<-mvrnorm(n,mu=rep(0,p),Sigma=diag(rep(1,p)))
X2<-mvrnorm(n,mu=rep(-2,p),Sigma=diag(rep(1,p)))
X3<-mvrnorm(n,mu=rep(2,p),Sigma=diag(rep(1,p)))



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")
points(X2[,1],X2[,2],col=2)
points(X3[,1],X3[,2],col=3)


Y<-c(rep(1,n),rep(2,n),rep(3,n))

X<-rbind(X1,X2,X3)

# K-folds
K <- 10
folds <- sample(1:K, length(X[,1]), replace = TRUE)


# #########################################################################################
### Linear discriminant Analysis

# Note: This function asssumes that the KK classes are encoded as 1:KK
LDA.func<-function(xp,Y,Xt){
  KK<-length(unique(Y))
  N<-length(Y)
  p<-length(xp)
  muk<-matrix(-99,nrow=KK,ncol=p)
  pik<-rep(-99,KK)
  Xtc<-Xt

  for(k in 1:KK){
    muk[k,]<-apply(matrix(Xt[Y==k,],ncol=p),2,mean)
    pik[k]<-mean(Y==k)
    Xtc[Y==k,]<-t(t(Xt[Y==k,])-muk[k,])
  }
  Sig<-t(Xtc)%*%Xtc/(N-KK)
  Sigi<-solve(Sig)
  dk<-rep(-99,KK)
  for(k in 1:KK){
     dk[k]<-t(xp)%*%Sigi%*%muk[k,]    -t(muk[k,])%*%Sigi%*%muk[k,]/2 +log(pik[k])
  }
  return(order(dk)[KK])
}


######################################
# cross validation

# LDA.cv.errors[i,j] saves the cross validation error for the first i covariates and jth fold
```

```
LDA.cv.errors <- matrix(NA, nrow=p, ncol=K)

for (pp in 1:p){

for (j in 1:K){
     ii <- which(folds==j)
     part.test<-apply(matrix(X[ii,1:pp],ncol=pp),1,LDA.func,Y=Y[-ii],Xt=matrix(X[-ii,1:pp],
         ncol=pp))
     jj<-which(Y[ii]!=part.test)
     LDA.cv.errors[pp,j] <- length(jj)/(length(ii))
}
}

apply(LDA.cv.errors,1,mean)




#########################################################################################
### kernel density classification

library(ks)

#######################################
# cross validation

KDC.cv.errors <- matrix(NA, nrow=p, ncol=K)

for (pp in 1:p){

for (i in 1:K){
     ii <- which(folds==i)
     Ytest <- Y[ii]
     Xtest <- X[ii,]
     Ytrain <- Y[-ii]
     Xtrain <- X[-ii,]

     ii1 <- which(Ytrain==1)
     ii2 <- which(Ytrain==2)
     ii3 <- which(Ytrain==3)

     # KK is the number of classes
     KK<-length(unique(Ytrain))
     pik<-rep(-99,KK)

     for(k in 1:KK){
     pik[k]<-mean(Ytrain==k)
     }

     fhat1.t <- kde(x=Xtrain[ii1,1:pp],eval.points=Xtest[,1:pp])$estimate
     fhat2.t <- kde(x=Xtrain[ii2,1:pp],eval.points=Xtest[,1:pp])$estimate
     fhat3.t <- kde(x=Xtrain[ii3,1:pp],eval.points=Xtest[,1:pp])$estimate

     fhat.t<-cbind(pik[1]*fhat1.t,pik[2]*fhat2.t,pik[3]*fhat3.t)
     part.test<-rep(NA,length(ii))

     for (j in 1:length(ii)){
                fhatj.t=fhat.t[j,]
                part.test[j]<-order(fhatj.t)[3]
     }

     jj<-which(Ytest!=part.test)
     KDC.cv.errors[pp,i] <- length(jj)/(length(ii))
}
}

apply(KDC.cv.errors,1,mean)

```

```
########################################################################################
### Naive Bayes Classification

library(ks)



######################################
# cross validation

NBC.cv.errors <- matrix(NA, nrow=p, ncol=K)

for (pp in 1:p){

for (i in 1:K){

     ii <- which(folds==i)
     Ytest <- Y[ii]
     Xtest <- X[ii,]
     Ytrain <- Y[-ii]
     Xtrain <- X[-ii,]

     ii1 <- which(Ytrain==1)
     ii2 <- which(Ytrain==2)
     ii3 <- which(Ytrain==3)

     # KK is the number of classes
     KK<-length(unique(Ytrain))
     pik<-rep(-99,KK)

     for(k in 1:KK){
     pik[k]<-mean(Ytrain==k)
     }

     fhat1.t <- kde(x=Xtrain[ii1,1],eval.points=Xtest[,1])$estimate
     fhat2.t <- kde(x=Xtrain[ii2,1],eval.points=Xtest[,1])$estimate
     fhat3.t <- kde(x=Xtrain[ii3,1],eval.points=Xtest[,1])$estimate

     for (k in 2:pp){

     fhat1k <- kde(x=Xtrain[ii1,k],eval.points=Xtest[,k])$estimate
     fhat2k <- kde(x=Xtrain[ii2,k],eval.points=Xtest[,k])$estimate
     fhat3k <- kde(x=Xtrain[ii3,k],eval.points=Xtest[,k])$estimate

     fhat1.t <- fhat1.t*fhat1k
     fhat2.t <- fhat2.t*fhat2k
     fhat3.t <- fhat3.t*fhat3k

     }

     fhat.t<-cbind(pik[1]*fhat1.t,pik[2]*fhat2.t,pik[3]*fhat3.t)
     part.test<-rep(NA,length(ii))

     for (j in 1:length(ii)){
                fhatj.t=fhat.t[j,]
                part.test[j]<-order(fhatj.t)[3]
     }

     jj<-which(Ytest!=part.test)
     NBC.cv.errors[pp,i] <- length(jj)/(length(ii))
}
}

apply(NBC.cv.errors,1,mean)

```

```
##############################################################

# check the outcome
print(apply(LDA.cv.errors,1,mean))
print(apply(KDC.cv.errors,1,mean))
print(apply(NBC.cv.errors,1,mean))


## Plot
LDA<-apply(LDA.cv.errors,1,mean)
KDC<-apply(KDC.cv.errors,1,mean)
NBC<-apply(NBC.cv.errors,1,mean)

number<-seq(1,p,by=1)


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))
points(number, KDC, pch="*", col=2)
lines(number, KDC, lty=2, col=2)
points(number, NBC, pch="+", col=3)
lines(number, NBC, lty=3, col=3)

legend(6,0.3,legend=c("LDA", "KDC", "NBC"),col=1:3, pch=c("o","*","+"), lty=c(1,2,3), ncol=1)


## Cross-validation of the log-normal sample across the number of predictors

# ##############################################################################################
### Here we will generate p-covariables for each class
### Simulate observations for each class and Estimate Prediction Error Using Cross Validation
### Assume the covariates are generated through Log Normal Distribution
### In this case, LDA is misspecified, and we can expect that KDC and NBC
### Should have a better performance in terms of prediction error than LDA
# ##############################################################################################
library(MASS)

set.seed(123456)

# number of observations for each class
n<-100
# number of covariates
p<-7

X1<-mvrnorm(n,mu=rep(0,p),Sigma=diag(rep(1,p)))
X2<-mvrnorm(n,mu=rep(-2,p),Sigma=diag(rep(1,p)))
X3<-mvrnorm(n,mu=rep(2,p),Sigma=diag(rep(1,p)))

# exponential covariates, so that all the covariates follow Log Normal Distribution
X1<-exp(X1)
X2<-exp(X2)
X3<-exp(X3)


plot(X1[,1],X1[,2],main="Sample Data",xlim=c(0,10),ylim=c(0,10),xlab="Input 1", ylab="Input 2")
points(X2[,1],X2[,2],col=2)
points(X3[,1],X3[,2],col=3)


Y<-c(rep(1,n),rep(2,n),rep(3,n))

```

```
X<-rbind(X1,X2,X3)

# K-folds, for CV
K <- 10
folds <- sample(1:K, length(X[,1]), replace = TRUE)


# ##########################################################################################
### Linear discriminant Analysis

# Note: This function asssumes that the KK classes are encoded as 1:KK
LDA.func<-function(xp,Y,Xt){
  KK<-length(unique(Y))
  N<-length(Y)
  p<-length(xp)
  muk<-matrix(-99,nrow=KK,ncol=p)
  pik<-rep(-99,KK)
  Xtc<-Xt

  for(k in 1:KK){
    muk[k,]<-apply(matrix(Xt[Y==k,],ncol=p),2,mean)
    pik[k]<-mean(Y==k)
    Xtc[Y==k,]<-t(t(Xt[Y==k,])-muk[k,])
  }
  Sig<-t(Xtc)%*%Xtc/(N-KK)
  Sigi<-solve(Sig)
  dk<-rep(-99,KK)
  for(k in 1:KK){
     dk[k]<-t(xp)%*%Sigi%*%muk[k,]    -t(muk[k,])%*%Sigi%*%muk[k,]/2 +log(pik[k])
  }
  return(order(dk)[KK])
}


#######################################
# cross validation

# LDA.cv.errors[i,j] saves the cross validation error for the first i covariates and jth
    fold
LDA.cv.errors <- matrix(NA, nrow=p, ncol=K)


for (pp in 1:p){

for (j in 1:K){
     ii <- which(folds==j)
     part.test<-apply(matrix(X[ii,1:pp],ncol=pp),1,LDA.func,Y=Y[-ii],Xt=matrix(X[-ii,1:pp],
         ncol=pp))
     jj<-which(Y[ii]!=part.test)
     LDA.cv.errors[pp,j] <- length(jj)/(length(ii))
}
}

apply(LDA.cv.errors,1,mean)




#########################################################################################
### kernel density classification

library(ks)


#######################################
# cross validation
```

```
KDC.cv.errors <- matrix(NA, nrow=p, ncol=K)

for (pp in 1:p){

for (i in 1:K){
     ii <- which(folds==i)
     Ytest <- Y[ii]
     Xtest <- X[ii,]
     Ytrain <- Y[-ii]
     Xtrain <- X[-ii,]

     ii1 <- which(Ytrain==1)
     ii2 <- which(Ytrain==2)
     ii3 <- which(Ytrain==3)

     # KK is the number of classes
     KK<-length(unique(Ytrain))
     pik<-rep(-99,KK)

     for(k in 1:KK){
     pik[k]<-mean(Ytrain==k)
     }

     fhat1.t <- kde(x=Xtrain[ii1,1:pp],eval.points=Xtest[,1:pp])$estimate
     fhat2.t <- kde(x=Xtrain[ii2,1:pp],eval.points=Xtest[,1:pp])$estimate
     fhat3.t <- kde(x=Xtrain[ii3,1:pp],eval.points=Xtest[,1:pp])$estimate

     fhat.t<-cbind(pik[1]*fhat1.t,pik[2]*fhat2.t,pik[3]*fhat3.t)
     part.test<-rep(NA,length(ii))

     for (j in 1:length(ii)){
                fhatj.t=fhat.t[j,]
                part.test[j]<-order(fhatj.t)[3]
     }

     jj<-which(Ytest!=part.test)
     KDC.cv.errors[pp,i] <- length(jj)/(length(ii))
}
}

apply(KDC.cv.errors,1,mean)




########################################################################################
### Naive Bayes Classification

library(ks)



######################################
# cross validation

NBC.cv.errors <- matrix(NA, nrow=p, ncol=K)

for (pp in 1:p){

for (i in 1:K){

     ii <- which(folds==i)
     Ytest <- Y[ii]
     Xtest <- X[ii,]
     Ytrain <- Y[-ii]
     Xtrain <- X[-ii,]

```

```
      ii1 <- which(Ytrain==1)
      ii2 <- which(Ytrain==2)
      ii3 <- which(Ytrain==3)

      # KK is the number of classes
      KK<-length(unique(Ytrain))
      pik<-rep(-99,KK)

      for(k in 1:KK){
      pik[k]<-mean(Ytrain==k)
      }

      fhat1.t <- kde(x=Xtrain[ii1,1],eval.points=Xtest[,1])$estimate
      fhat2.t <- kde(x=Xtrain[ii2,1],eval.points=Xtest[,1])$estimate
      fhat3.t <- kde(x=Xtrain[ii3,1],eval.points=Xtest[,1])$estimate

      for (k in 2:pp){

      fhat1k <- kde(x=Xtrain[ii1,k],eval.points=Xtest[,k])$estimate
      fhat2k <- kde(x=Xtrain[ii2,k],eval.points=Xtest[,k])$estimate
      fhat3k <- kde(x=Xtrain[ii3,k],eval.points=Xtest[,k])$estimate

      fhat1.t <- fhat1.t*fhat1k
      fhat2.t <- fhat2.t*fhat2k
      fhat3.t <- fhat3.t*fhat3k

      }

      fhat.t<-cbind(pik[1]*fhat1.t,pik[2]*fhat2.t,pik[3]*fhat3.t)
      part.test<-rep(NA,length(ii))

      for (j in 1:length(ii)){
                 fhatj.t=fhat.t[j,]
                 part.test[j]<-order(fhatj.t)[3]
      }

      jj<-which(Ytest!=part.test)
      NBC.cv.errors[pp,i] <- length(jj)/(length(ii))
}
}

apply(NBC.cv.errors,1,mean)

################################################################

# check the outcome
print(apply(LDA.cv.errors,1,mean))
print(apply(KDC.cv.errors,1,mean))
print(apply(NBC.cv.errors,1,mean))

## Plot
LDA<-apply(LDA.cv.errors,1,mean)
KDC<-apply(KDC.cv.errors,1,mean)
NBC<-apply(NBC.cv.errors,1,mean)

number<-seq(1,p,by=1)


plot(number, LDA, type="o", lty=1, col=1,main="Models Comparasion (Covariates Follow
    LogNormal, n=100, K=3)", xlab="p", ylab="Cross Validition Errors", ylim=c(0,0.6))
points(number, KDC, pch="*", col=2)
lines(number, KDC, lty=2, col=2)
points(number, NBC, pch="+", col=3)
lines(number, NBC, lty=3, col=3)

legend(6,0.6,legend=c("LDA", "KDC", "NBC"),col=1:3, pch=c("o","*","+"), lty=c(1,2,3), ncol
    =1)

```

```
# R Code for Data Application

#  --- (0) Data -----------------------------------------------------------------

data <- read.table(choose.files(), sep=",", header=TRUE)
  # Heart data
head(data)

panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...)
{
  usr <- par("usr"); on.exit(par(usr))
  par(usr = c(0, 1, 0, 1))
  r <- abs(cor(x, y))
  txt <- format(c(r, 0.123456789), digits = digits)[1]
  txt <- paste0(prefix, txt)
  if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
  text(0.5, 0.5, txt, cex = cex.cor * r)
}
pairs(data[,c(2:4,6,8:11)], pch=c(21,22)[unclass(factor(data$chd))], bg = c("black","red")[
    unclass(factor(data$chd))], lower.panel = panel.cor,
      gap=0.25, row1attop=FALSE)

par(mfrow=c(3,2))
for ( i in c(2:4,8:10)){
  plot(density(data[,i]), main = paste0(colnames(data)[i]))
}



data[,11] <- data[,11] + 1 # 0->1; 1->2

X <- cbind(data[,2:4],data[,6]=="Present",data[,8:10])
X <- as.matrix(X)
Y <- as.numeric(data[,11])

G <- unique(Y)
K <- 6 # 6-folds
n <- dim(X)[1]
p <- dim(X)[2]



# --- (1) LDA ------------------------------------------------------------------

LDA.func<-function(xp,Y,Xt){
  KK<-length(unique(Y))
  N<-length(Y)
  p<-length(xp)
  muk<-matrix(-99,nrow=KK,ncol=p)
  pik<-rep(-99,KK)
  Xtc<-Xt

  for(k in 1:KK){
    muk[k,]<-apply(matrix(Xt[Y==k,],ncol=p),2,mean)
    pik[k]<-mean(Y==k)
    Xtc[Y==k,]<-t(t(Xt[Y==k,])-muk[k,])
  }
  Sig<-t(Xtc)%*%Xtc/(N-KK)
  Sigi<-solve(Sig)
  dk<-rep(-99,KK)
  for(k in 1:KK){
    dk[k]<-t(xp)%*%Sigi%*%muk[k,]    -t(muk[k,])%*%Sigi%*%muk[k,]/2 +log(pik[k])
  }
  return(order(dk)[KK])
}
```

```r
LDA.test.error <- rep(-99, K) # K number of test error
set.seed(12345)
test.id <- split(sample(1:n),1:K) # A list of indices to set aside test data
for (k in 1:K){
  X.test <- X[test.id[[k]],]
  X.train <- X[-test.id[[k]],]
  Y.test <- Y[test.id[[k]]]
  Y.train <- Y[-test.id[[k]]]

  Y.test.hat <- rep(-99, length(Y.test)) # Record for a vector of log-likelihood fits
  for (r in 1:length(Y.test)){
    temp <- LDA.func(X.test[r,], Y=Y.train, Xt=X.train)
    Y.test.hat[r] <- temp
  }
  LDA.test.error[k] <- sum(Y.test!=Y.test.hat)/length(Y.test)
}

LDA.CV.score <- mean(LDA.test.error)


# --- (2) KDC ---------------------------------------------------------------

library(ks)

KDC.cv.errors <- rep(-99,K)

for (i in 1:K){
  ii <- test.id[[i]]
  Ytest <- Y[ii]
  Xtest <- X[ii,]
  Ytrain <- Y[-ii]
  Xtrain <- X[-ii,]

  ii1 <- which(Ytrain==1)
  ii2 <- which(Ytrain==2)

  # KK is the number of classes
  KK<-length(G)
  pik<-rep(-99,KK)

  for(k in 1:KK){
    pik[k]<-mean(Ytrain==k)
  }

  fhat1.t <- kde(x=Xtrain[ii1,],eval.points=Xtest)$estimate
  fhat2.t <- kde(x=Xtrain[ii2,],eval.points=Xtest)$estimate

  fhat.t<-cbind(pik[1]*fhat1.t, pik[2]*fhat2.t)
  part.test<-rep(NA,length(ii))

  for (j in 1:length(ii)){
    fhatj.t=fhat.t[j,]
    part.test[j]<-order(fhatj.t)[2]
  }

  jj<-which(Ytest!=part.test)
  KDC.cv.errors[i] <- length(jj)/(length(ii))
}

KDC.CV.score <- mean(KDC.cv.errors)



# --- (3) NBC ---------------------------------------------------------------

library(klaR)
#library(stats) # density(data$sbp,kernel = "epanechnikov")

```

```
NBC.test.error <- rep(-99, K) # K number of test error
for (k in 1:K){
  X.test <- X[test.id[[k]],]
  X.train <- X[-test.id[[k]],]
  Y.test <- Y[test.id[[k]]]
  Y.train <- Y[-test.id[[k]]]

  Y.train <- as.factor(Y.train)
  Y.test <- as.factor(Y.test)

  NBC.model <- NaiveBayes(X.train, Y.train, usekernel=T)
  Y.test.hat <- predict(NBC.model, X.test, threshold = 1e-4 )

  NBC.test.error[k] <- sum(Y.test!=Y.test.hat$class)/length(Y.test)
}

NBC.CV.score <- mean(NBC.test.error)


c(LDA.CV.score, KDC.CV.score, NBC.CV.score)

X <- cbind(data[,2:4],factor(data[,6]),data[,8:10])
Y <- as.factor(data[,11])

NBC.fit <- NaiveBayes(X,Y,usekernel=T)

par(mfrow=c(3,2))
  plot(NBC.fit,ylab = "Density",vars = "sbp", main = "Class-conditional Marginal Density",
      col = c("black","red"), legendplot = F)
  legend("topright", legend = c("chd=0","chd=1"), col = c("black","red"), lty = 1:2)
for (p in colnames(data[,c(3:4,6,8:10)])){
  plot(NBC.fit,ylab = "Density",vars = p, main = "", col = c("black","red"), legendplot = F)
}
```