HW1

20989977 Zhang Mingtao

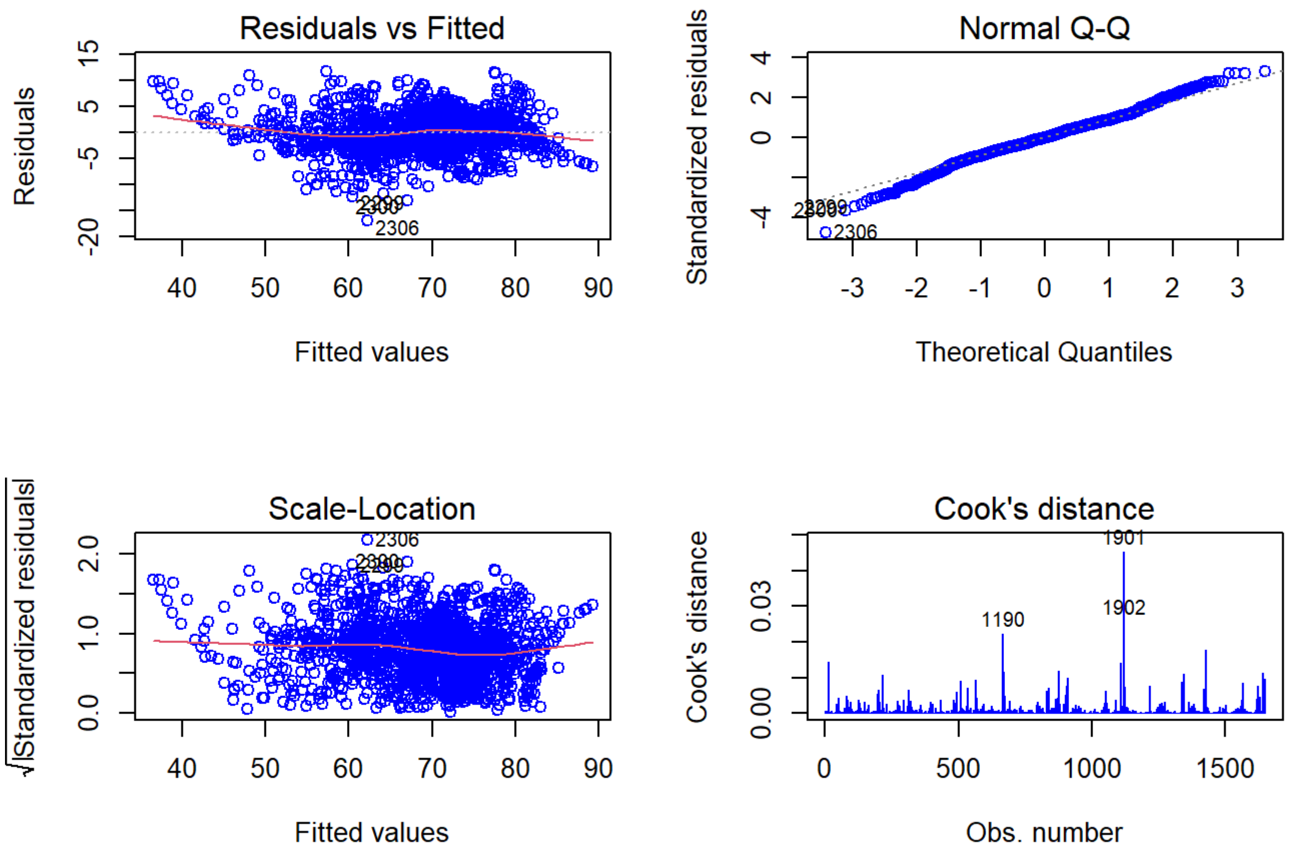
2023/9/19

1.

```
###1
#(1)
csv1=read.csv("C://Users//张铭韬//Desktop//学业//港科大//MSDM5054机器学习//作业//hw1//Life Expectancy Data.csv")
df1=csv1[,-c(1,2)]
df1=na.omit(df1)
mod1=lm(Life.expectancy~.,data=df1)
summary(mod1)
```

```
##
## Call:
## lm(formula = Life.expectancy ~ ., data = df1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.9597  -2.0621  -0.0147   2.2751  11.7115
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     5.445e+01  8.400e-01  64.822  < 2e-16 ***
## StatusDeveloping               -9.684e-01  3.379e-01  -2.865  0.00422 **
## Adult.Mortality                -1.663e-02  9.494e-04 -17.517  < 2e-16 ***
## infant.deaths                   9.350e-02  1.065e-02   8.777  < 2e-16 ***
## Alcohol                        -9.140e-02  3.316e-02  -2.756  0.00592 **
## percentage.expenditure          3.673e-04  1.801e-04   2.040  0.04156 *
## Hepatitis.B                    -6.525e-03  4.449e-03  -1.467  0.14265
## Measles                        -7.865e-06  1.079e-05  -0.729  0.46597
## BMI                             3.376e-02  5.998e-03   5.628 2.15e-08 ***
## under.five.deaths              -7.035e-02  7.711e-03  -9.123  < 2e-16 ***
## Polio                           7.935e-03  5.152e-03   1.540  0.12370
## Total.expenditure               7.586e-02  4.067e-02   1.865  0.06236 .
## Diphtheria                      1.490e-02  5.928e-03   2.513  0.01205 *
## HIV.AIDS                       -4.370e-01  1.784e-02 -24.490  < 2e-16 ***
## GDP                             8.738e-06  2.837e-05   0.308  0.75813
## Population                     -6.425e-10  1.749e-09  -0.367  0.71337
## thinness..1.19.years           -1.238e-02  5.300e-02  -0.234  0.81527
## thinness.5.9.years             -4.798e-02  5.231e-02  -0.917  0.35917
## Income.composition.of.resources  9.817e+00  8.321e-01  11.797  < 2e-16 ***
## Schooling                       8.665e-01  5.940e-02  14.587  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.588 on 1629 degrees of freedom
## Multiple R-squared:  0.8356, Adjusted R-squared:  0.8336
## F-statistic: 435.7 on 19 and 1629 DF,  p-value: < 2.2e-16
```

```
#Adult.Mortality, infant.deaths, under.five.deaths, BMI, HIV.AIDS, Income.composition.of.resources, Schooling may be the most
#important variables; next are: Status, Alcohol, Diphtheria and percentage.expenditure.

#总体检验
par(mfrow=c(2,2))
plot(mod1, which=1:4, col="blue")
#独立性
library(DescTools)
```



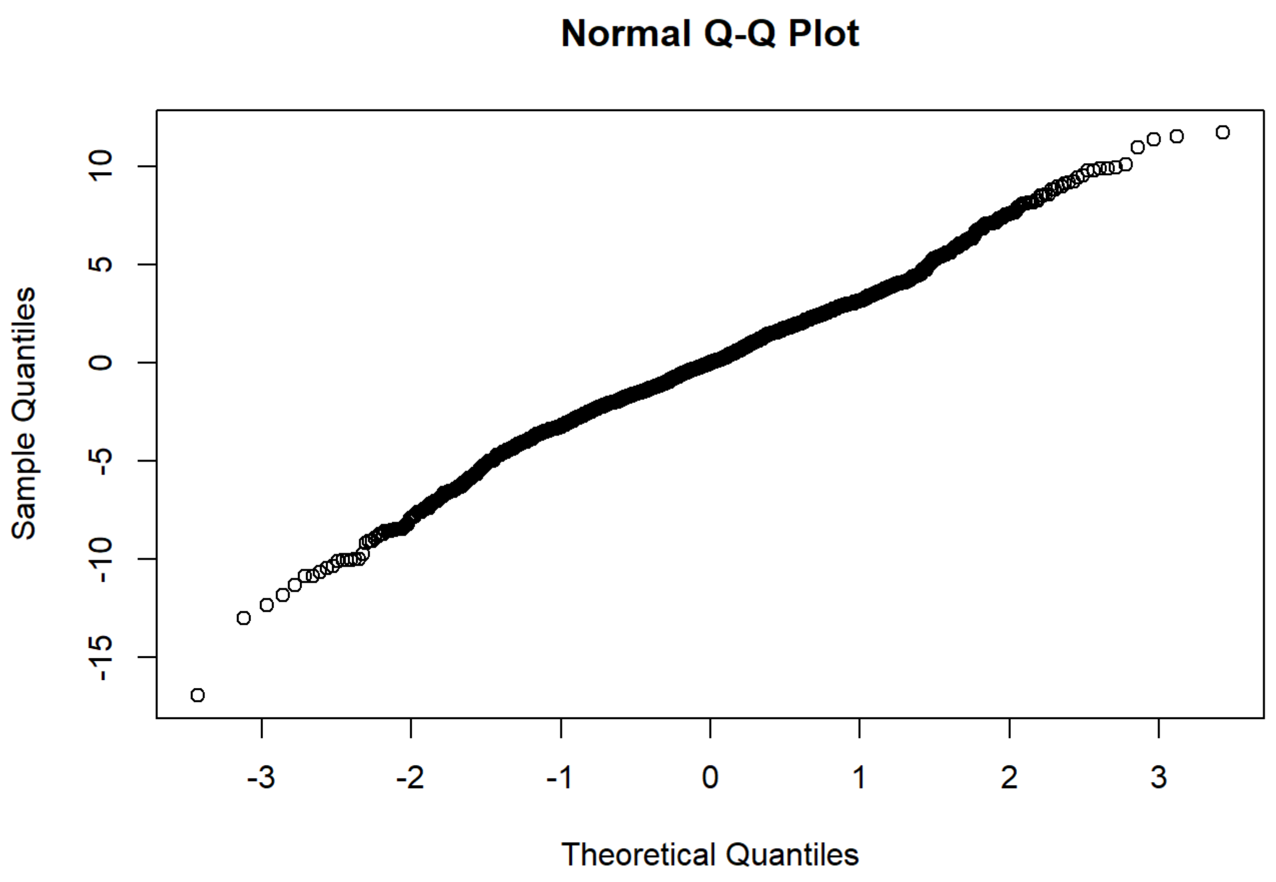
```
DurbinWatsonTest(mod1)
```

```
##
##  Durbin-Watson test
##
## data:  mod1
## DW = 0.70551, p-value < 2.2e-16
## alternative hypothesis: true autocorrelation is greater than 0
```

```
#残差正态性
lmres=residuals(mod1)
shapiro.test(lmres)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  lmres
## W = 0.99082, p-value = 1.159e-08
```

```
par(mfrow=c(1,1))
qqnorm(lmres)
```



```
#方差齐性
library(zoo)
```

```
##
## 载入程辑包：'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(lmtest)
bptest(mod1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  mod1
## BP = 188.24, df = 19, p-value < 2.2e-16
```

```
#(2)
confint(mod1,level=0.95)
```

```
##                                       2.5 %        97.5 %
## (Intercept)                    5.280349e+01  5.609873e+01
## StatusDeveloping              -1.631210e+00 -3.055239e-01
## Adult.Mortality               -1.849399e-02 -1.476948e-02
## infant.deaths                  7.260415e-02  1.143953e-01
## Alcohol                       -1.564424e-01 -2.634759e-02
## percentage.expenditure         1.406977e-05  7.206029e-04
## Hepatitis.B                   -1.525013e-02  2.200833e-03
## Measles                       -2.902123e-05  1.329036e-05
## BMI                            2.199085e-02  4.552044e-02
## under.five.deaths             -8.547344e-02 -5.522328e-02
## Polio                         -2.169947e-03  1.804045e-02
## Total.expenditure             -3.921521e-03  1.556380e-01
## Diphtheria                     3.272244e-03  2.652641e-02
## HIV.AIDS                      -4.719608e-01 -4.019673e-01
## GDP                           -4.690855e-05  6.438443e-05
## Population                    -4.072414e-09  2.787485e-09
## thinness..1.19.years          -1.163427e-01  9.157275e-02
## thinness.5.9.years            -1.505946e-01  5.462745e-02
## Income.composition.of.resources  8.184423e+00  1.144872e+01
## Schooling                      7.499887e-01  9.830178e-01
```

```
#Adult.Mortality: [-1.849399e-02,-1.476948e-02]
#       HIV.AIDS: [-4.719608e-01,-4.019673e-01]

#These 2 variables both pass the t-test(α=.001) so I'm confident they have negative impact on the life expectancy.

#(3)
confint(mod1,level=0.97)
```

```
##                                        1.5 %        98.5 %
## (Intercept)                      5.262661e+01  5.627561e+01
## StatusDeveloping                -1.702370e+00 -2.343631e-01
## Adult.Mortality                 -1.869392e-02 -1.456956e-02
## infant.deaths                    7.036087e-02  1.166385e-01
## Alcohol                         -1.634257e-01 -1.936430e-02
## percentage.expenditure          -2.385583e-05  7.585285e-04
## Hepatitis.B                     -1.618687e-02  3.137574e-03
## Measles                         -3.129245e-05  1.556158e-05
## BMI                              2.072782e-02  4.678347e-02
## under.five.deaths               -8.709722e-02 -5.359950e-02
## Polio                           -3.254810e-03  1.912532e-02
## Total.expenditure               -1.248643e-02  1.642029e-01
## Diphtheria                       2.023997e-03  2.777466e-02
## HIV.AIDS                        -4.757180e-01 -3.982101e-01
## GDP                             -5.288259e-05  7.035846e-05
## Population                      -4.440643e-09  3.155714e-09
## thinness..1.19.years            -1.275033e-01  1.027333e-01
## thinness.5.9.years              -1.616106e-01  6.564344e-02
## Income.composition.of.resources  8.009200e+00  1.162394e+01
## Schooling                        7.374800e-01  9.955264e-01
```

```
#Schooling:[7.374800e-01,9.955264e-01]----→The larger the number of years of Schooling is, the longer the Life Expectancy is.
#  Alcohol:[-1.634257e-01 -1.936430e-02]--→The more alcohol consumption recorded per capita (15+) is, the shorter the Life Exp
ectancy is.

#(4)
#Adult.Mortality:< 2e-16
#infant.deaths：< 2e-16
#under.five.deaths：< 2e-16
#BMI：2.15e-08
#HIV.AIDS：< 2e-16
#Income.composition.of.resources：< 2e-16
#Schooling：< 2e-16
mod2=lm(Life.expectancy~Adult.Mortality+infant.deaths+under.five.deaths+BMI+HIV.AIDS+Income.composition.of.resources+Schooling,
data=df1)
summary(mod2)
```

```
##
## Call:
## lm(formula = Life.expectancy ~ Adult.Mortality + infant.deaths +
##     under.five.deaths + BMI + HIV.AIDS + Income.composition.of.resources +
##     Schooling, data = df1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.604  -2.072  -0.023   2.200  12.308
##
## Coefficients:
##                                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     53.2080766  0.5361598  99.239  < 2e-16 ***
## Adult.Mortality                 -0.0178405  0.0009597 -18.589  < 2e-16 ***
## infant.deaths                    0.0910551  0.0098397   9.254  < 2e-16 ***
## under.five.deaths               -0.0699153  0.0073116  -9.562  < 2e-16 ***
## BMI                              0.0366104  0.0057061   6.416 1.83e-10 ***
## HIV.AIDS                        -0.4337578  0.0180613 -24.016  < 2e-16 ***
## Income.composition.of.resources 10.8860992  0.8247651  13.199  < 2e-16 ***
## Schooling                        0.9746212  0.0555178  17.555  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.683 on 1641 degrees of freedom
## Multiple R-squared:  0.8254, Adjusted R-squared:  0.8247
## F-statistic:  1108 on 7 and 1641 DF,  p-value: < 2.2e-16
```

```
#(5)
new_obs=data.frame(Adult.Mortality=125,infant.deaths=94,under.five.deaths=2,BMI=55,HIV.AIDS=0.5,Income.composition.of.resources
=0.9,Schooling=18)
# create the new observation
predict(mod2,newdata=new_obs,interval="confidence",level=0.99)
```

```
##        fit      lwr      upr
## 1 88.53472 86.15881 90.91064
```

```
#(6)
AIC(mod1)
```

```
## [1] 8914.947
```

```
AIC(mod2)
```

```
## [1] 8989.692
```

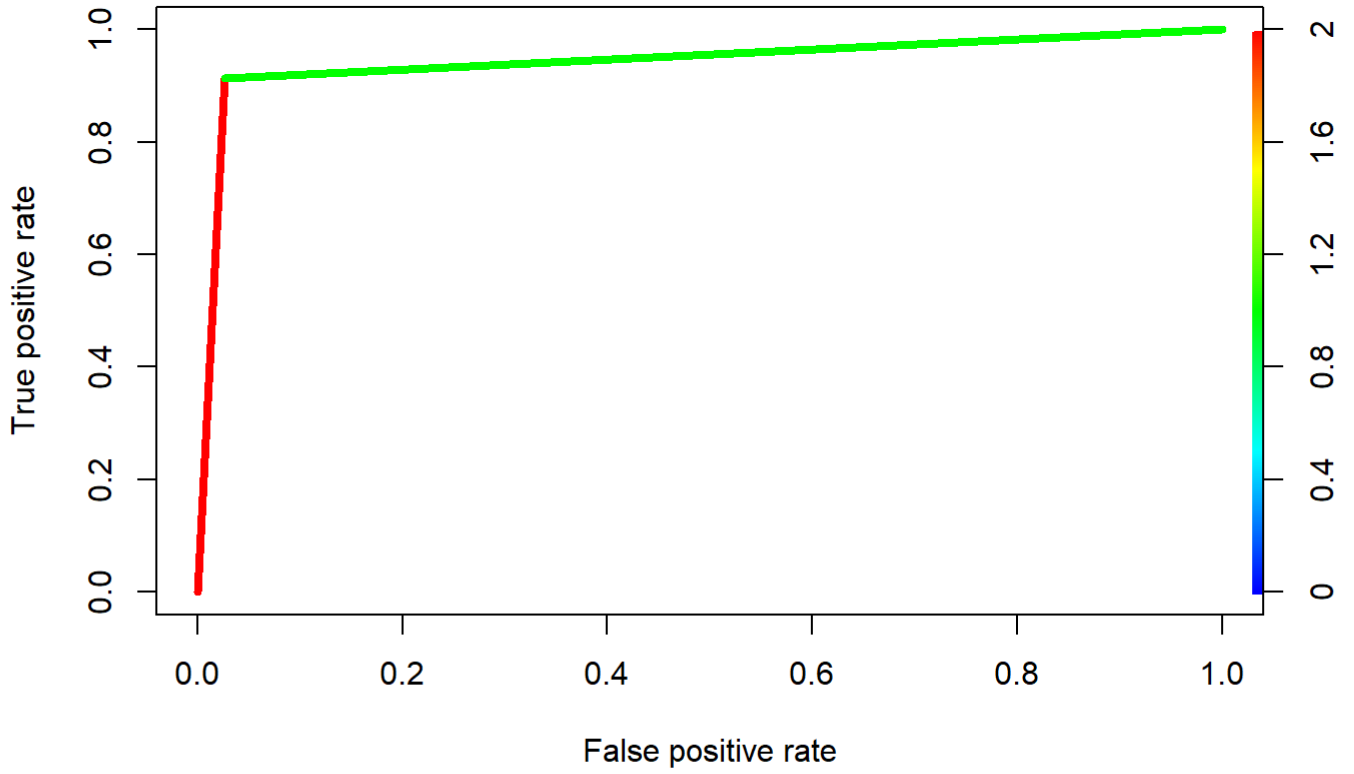
```
#The AIC of mod1(full model) is smaller than mod2(smaller model).
```

2.

```
###2
#(1)
train=read.csv("C://Users//张铭韬//Desktop//学业//港科大//MSDM5054机器学习//作业//hw1//BreastCancer_train.csv")
train=train[,-1]
test=read.csv("C://Users//张铭韬//Desktop//学业//港科大//MSDM5054机器学习//作业//hw1//BreastCancer_test.csv")
test=test[,-1]

train$type[which(train$Class== "benign")]=0 # benign编号为0
train$type[which(train$Class== "malignant")]=1 # malignant编号为1
test$type[which(test$Class== "benign")]=0 # benign编号为0
test$type[which(test$Class== "malignant")]=1 # malignant编号为1

train=train[,-10]
train=na.omit(train)
test=test[,-10]
test=na.omit(test)

train$type=factor(train$type)
test$type=factor(test$type)

glmod1=glm(type~., data=train,family=binomial)
summary(glmod1)
```

```
##
## Call:
## glm(formula = type ~ ., family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2871  -0.0863  -0.0542   0.0218   1.8764
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -10.5582     1.7131  -6.163 7.12e-10 ***
## Cl.thickness      0.3671     0.2154   1.704   0.0883 .
## Cell.size        -0.2865     0.2772  -1.034   0.3013
## Cell.shape        0.5608     0.3082   1.820   0.0688 .
## Marg.adhesion     0.3486     0.1411   2.470   0.0135 *
## Epith.c.size      0.4145     0.2656   1.561   0.1186
## Bare.nuclei       0.2939     0.1214   2.420   0.0155 *
## Bl.cromatin       0.6060     0.2613   2.319   0.0204 *
## Normal.nucleoli   0.1968     0.1635   1.203   0.2288
## Mitoses           0.3238     0.4110   0.788   0.4308
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 506.096  on 391  degrees of freedom
## Residual deviance:  53.745  on 382  degrees of freedom
## AIC: 73.745
##
## Number of Fisher Scoring iterations: 8
```

```
#Marg.adhesion, Bare.nuclei and Bl.cromatin are significant variables.
pred1=predict(glmod1,test,interval="prediction",type="response")
pred1=ifelse(pred1>0.5,1,0)
library(ROCR)
p1=prediction(pred1,test$type) #预测值和真实值
perf1=performance(p1,"tpr","fpr")
plot(perf1,colorize=TRUE,lwd=4)
```



```
#AUC:0.943
performance(p1, "auc")@y.values[[1]]
```

```
## [1] 0.9430128
```

```
#Another method:
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## 载入程辑包：'pROC'
```

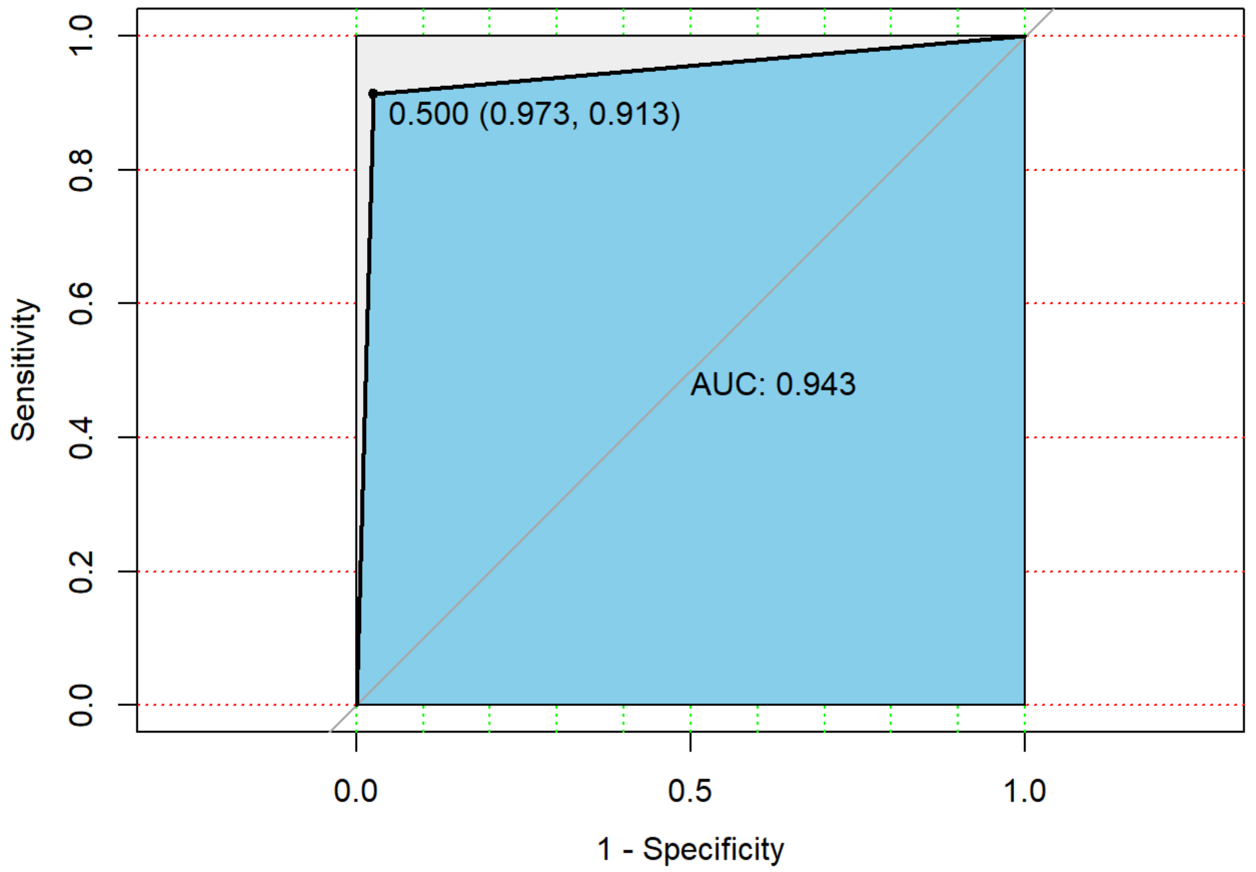
```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
modelroc1=roc(test$type,pred1) #真实值和预测值
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

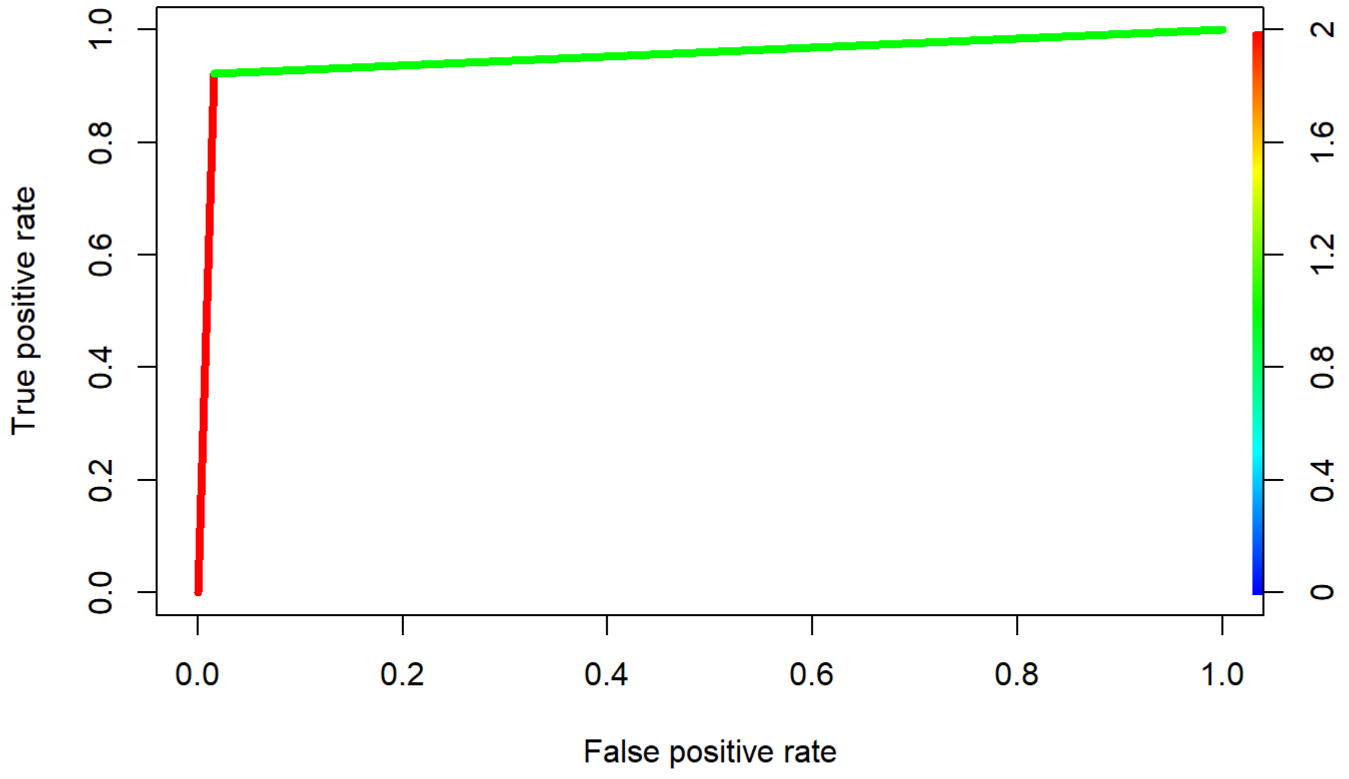
```
plot(modelroc1, print.auc=TRUE, auc.polygon=TRUE,legacy.axes=TRUE, grid=c(0.1, 0.2),
     grid.col=c("green", "red"), max.auc.polygon=TRUE,
     auc.polygon.col="skyblue", print.thres=TRUE)
```



```
#(2)
glmod2=glm(type~Cl.thickness+Cell.shape+Marg.adhesion+Bare.nuclei+Bl.cromatin, data=train,family=binomial)
summary(glmod2)
```

```
##
## Call:
## glm(formula = type ~ Cl.thickness + Cell.shape + Marg.adhesion +
##     Bare.nuclei + Bl.cromatin, family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2350  -0.1046  -0.0577   0.0239   1.9109
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -9.7502     1.4970  -6.513 7.35e-11 ***
## Cl.thickness    0.4866     0.1936   2.513  0.01198 *
## Cell.shape      0.4915     0.2002   2.455  0.01410 *
## Marg.adhesion   0.3515     0.1236   2.844  0.00446 **
## Bare.nuclei     0.2917     0.1078   2.707  0.00679 **
## Bl.cromatin     0.7599     0.2293   3.314  0.00092 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 506.096  on 391  degrees of freedom
## Residual deviance:  61.363  on 386  degrees of freedom
## AIC: 73.363
##
## Number of Fisher Scoring iterations: 8
```

```
pred2=predict(glmod2,test,interval="prediction",type="response")
pred2=ifelse(pred2>0.5,1,0)

p2=prediction(pred2,test$type) #预测值和真实值
perf2=performance(p2,"tpr","fpr")
plot(perf2,colorize=TRUE,lwd=4)
```



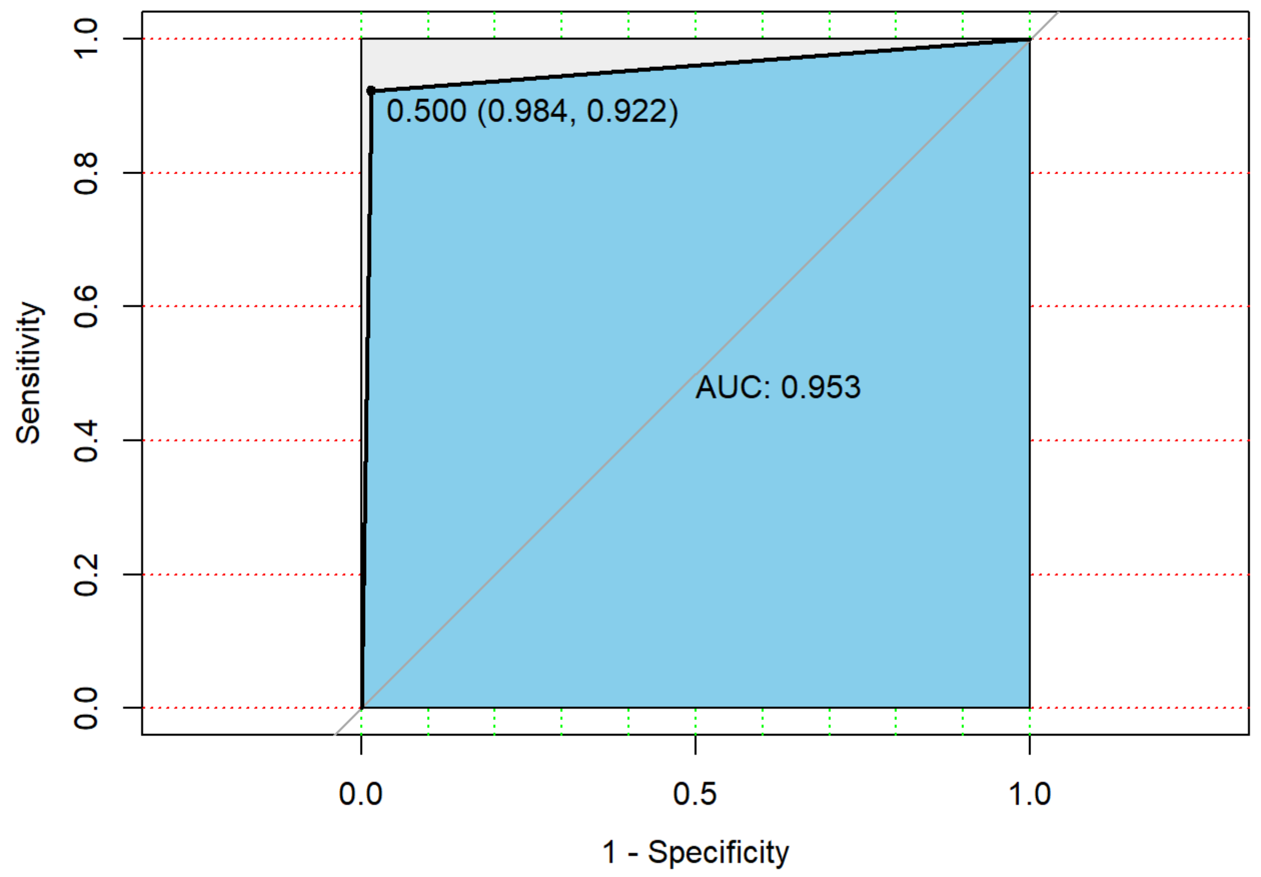
```
#AUC:0.953
performance(p2, "auc")@y.values[[1]]
```

```
## [1] 0.9531863
```

```
modelroc2=roc(test$type,pred2) #真实值和预测值
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
plot(modelroc2, print.auc=TRUE, auc.polygon=TRUE,legacy.axes=TRUE, grid=c(0.1, 0.2),
     grid.col=c("green", "red"), max.auc.polygon=TRUE,
     auc.polygon.col="skyblue", print.thres=TRUE)
```



```
#The AUC of glmod2 is larger than glmod1.



#(3)
library(MASS)
lda.fit=lda(type~.,data=train)
lda.fit  #Prior probabilities of groups先验概率; Coefficients of linear discriminants线性方程系数
```

```
## Call:
## lda(type ~ ., data = train)
##
## Prior probabilities of groups:
##         0         1
## 0.6530612 0.3469388
##
## Group means:
##   Cl.thickness Cell.size Cell.shape Marg.adhesion Epith.c.size Bare.nuclei
## 0     2.878906  1.281250   1.382812      1.304688     2.097656    1.339844
## 1     7.117647  6.669118   6.617647      5.786765     5.345588    7.757353
##   Bl.cromatin Normal.nucleoli  Mitoses
## 0    2.023438        1.238281 1.066406
## 1    6.264706        6.176471 2.786765
##
## Coefficients of linear discriminants:
##                         LD1
## Cl.thickness     0.15467544
## Cell.size        0.07723506
## Cell.shape       0.11382233
## Marg.adhesion    0.08656188
## Epith.c.size     0.13050211
## Bare.nuclei      0.24893576
## Bl.cromatin      0.12066142
## Normal.nucleoli  0.10230580
## Mitoses         -0.02736122
```

```
lda.pred=predict(lda.fit,test)
lda.pred$class  #预测的所属类的结果;后验概率为lda.pred$posterior
```

```
##   [1] 1 1 0 0 0 1 0 0 1 0 1 0 0 0 0 1 0 0 1 1 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0
##  [38] 0 0 0 1 1 0 0 0 1 0 1 0 1 1 0 0 1 0 0 1 0 0 1 0 0 0 1 1 0 1 1 0 0 1 0 0 1
##  [75] 1 0 1 0 1 0 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 0 1 0 1 1 0 0 1 1 0 1 0 1 0 0
## [112] 1 1 1 0 1 1 0 1 1 1 1 0 1 1 0 1 1 1 1 1 0 1 0 1 0 1 0 0 0 0 1 0 1 0 0 0 1
## [149] 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0
## [186] 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0
## [223] 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 1 0 0 0 1 1 1 0 0 1 0 1 0 1 0 0 0 0
## [260] 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0
## Levels: 0 1
```

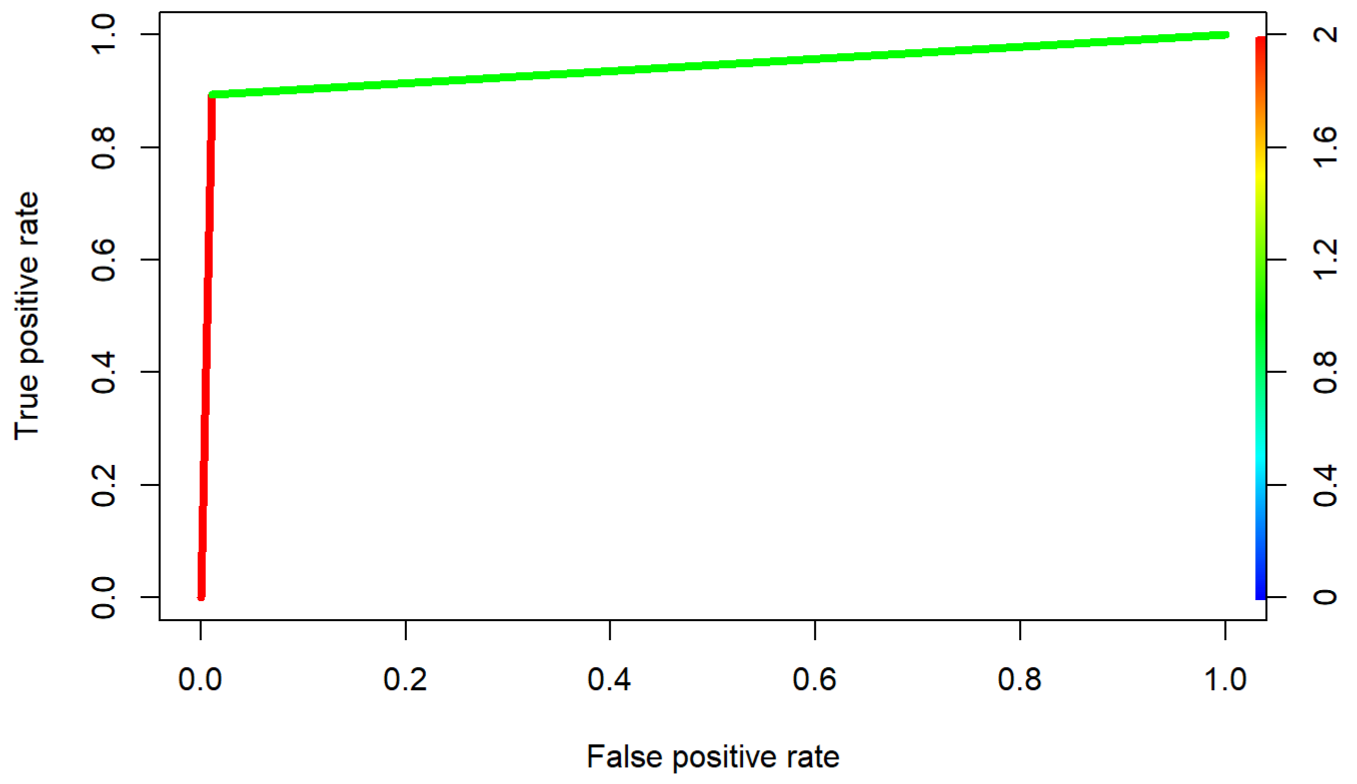
```
tab=table(lda.pred$class,test$type)  #预测值和真实值
tab    #混淆矩阵
```

```
##    
##       0   1
##   0 186  11
##   1   2  92
```

```
erro=1-sum(diag(prop.table(tab)))    #计算误判率
erro
```

```
## [1] 0.04467354
```

```
ldap1=prediction((as.numeric(lda.pred$class)-1),test$type) #预测值和真实值
ldaperf1=performance(ldap1,"tpr","fpr")
plot(ldaperf1,colorize=TRUE,lwd=4)
```



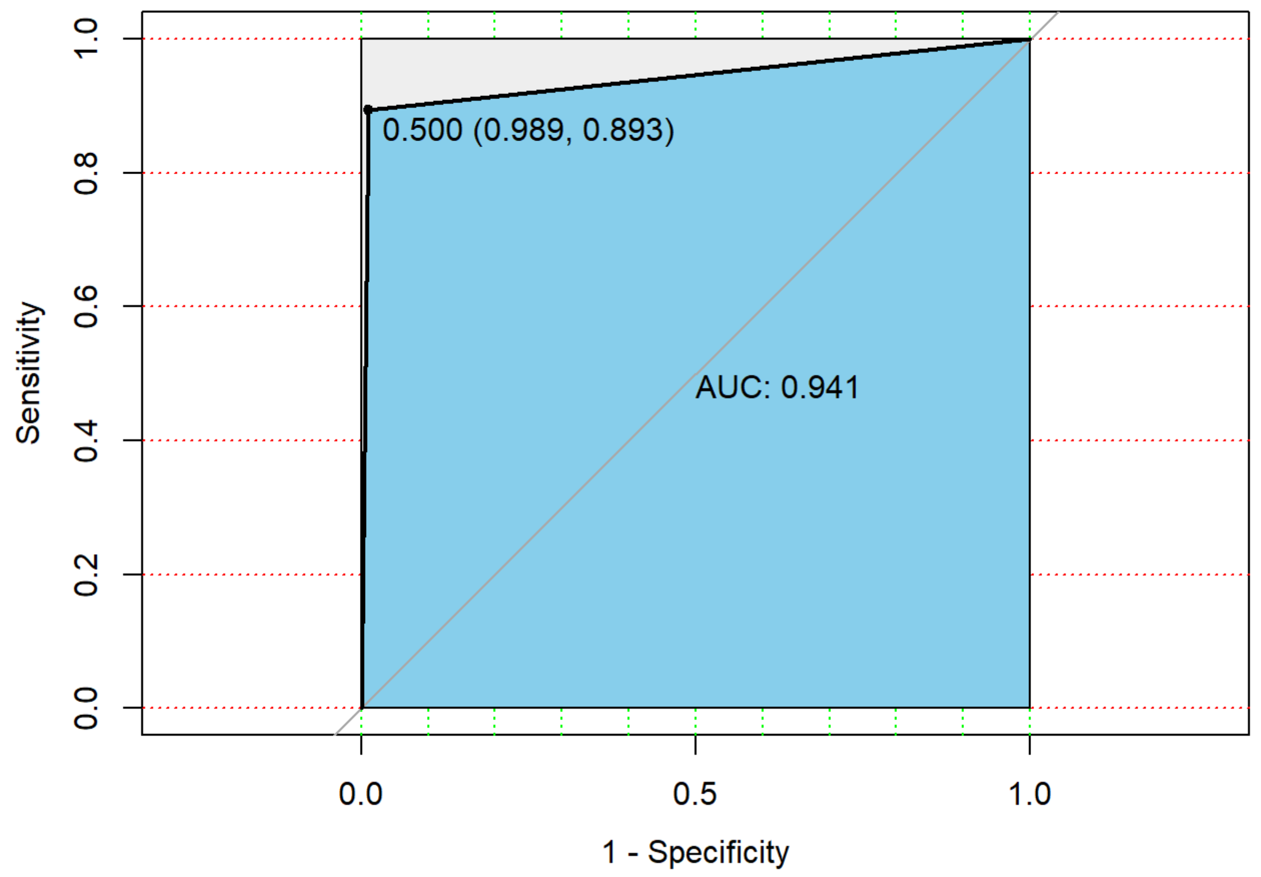
```
#AUC:0.941
performance(ldap1, "auc")@y.values[[1]]
```

```
## [1] 0.9412828
```

```
modelldaroc1=roc(test$type,(as.numeric(lda.pred$class)-1)) #真实值和预测值
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
plot(modelldaroc1, print.auc=TRUE, auc.polygon=TRUE,legacy.axes=TRUE, grid=c(0.1, 0.2),
     grid.col=c("green", "red"), max.auc.polygon=TRUE,
     auc.polygon.col="skyblue", print.thres=TRUE)
```



```
#(4)
lda.fit2=lda(type~Cl.thickness+Cell.shape+Marg.adhesion+Bare.nuclei+Bl.cromatin,data=train)
lda.fit2  #Prior probabilities of groups先验概率; Coefficients of linear discriminants线性方程系数
```

```
## Call:
## lda(type ~ Cl.thickness + Cell.shape + Marg.adhesion + Bare.nuclei +
##     Bl.cromatin, data = train)
##
## Prior probabilities of groups:
##         0         1
## 0.6530612 0.3469388
##
## Group means:
##   Cl.thickness Cell.shape Marg.adhesion Bare.nuclei Bl.cromatin
## 0     2.878906   1.382812      1.304688    1.339844    2.023438
## 1     7.117647   6.617647      5.786765    7.757353    6.264706
##
## Coefficients of linear discriminants:
##                     LD1
## Cl.thickness  0.1615430
## Cell.shape    0.2355030
## Marg.adhesion 0.1111864
## Bare.nuclei   0.2409875
## Bl.cromatin   0.1902969
```

```
lda.pred2=predict(lda.fit2,test)
lda.pred2$class  #预测的所属类的结果;后验概率为lda.pred$posterior
```

```
##   [1] 1 1 0 0 0 1 0 0 1 0 1 0 0 0 0 1 0 0 1 1 1 1 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0
##  [38] 0 0 0 1 1 0 0 1 1 0 1 0 1 1 0 0 1 0 0 1 0 0 1 0 0 0 1 1 0 1 1 0 0 1 0 0 1
##  [75] 1 0 1 0 1 0 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 0 1 0 1 1 0 0 1 1 0 1 0 1 0 0
## [112] 1 1 1 0 0 1 0 1 1 1 1 0 1 1 0 1 1 1 0 1 0 1 0 1 0 1 0 0 0 0 1 0 1 0 0 0 1
## [149] 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0
## [186] 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0
## [223] 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 1 0 0 0 1 1 1 0 0 1 0 1 0 1 0 0 0 0
## [260] 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0
## Levels: 0 1
```

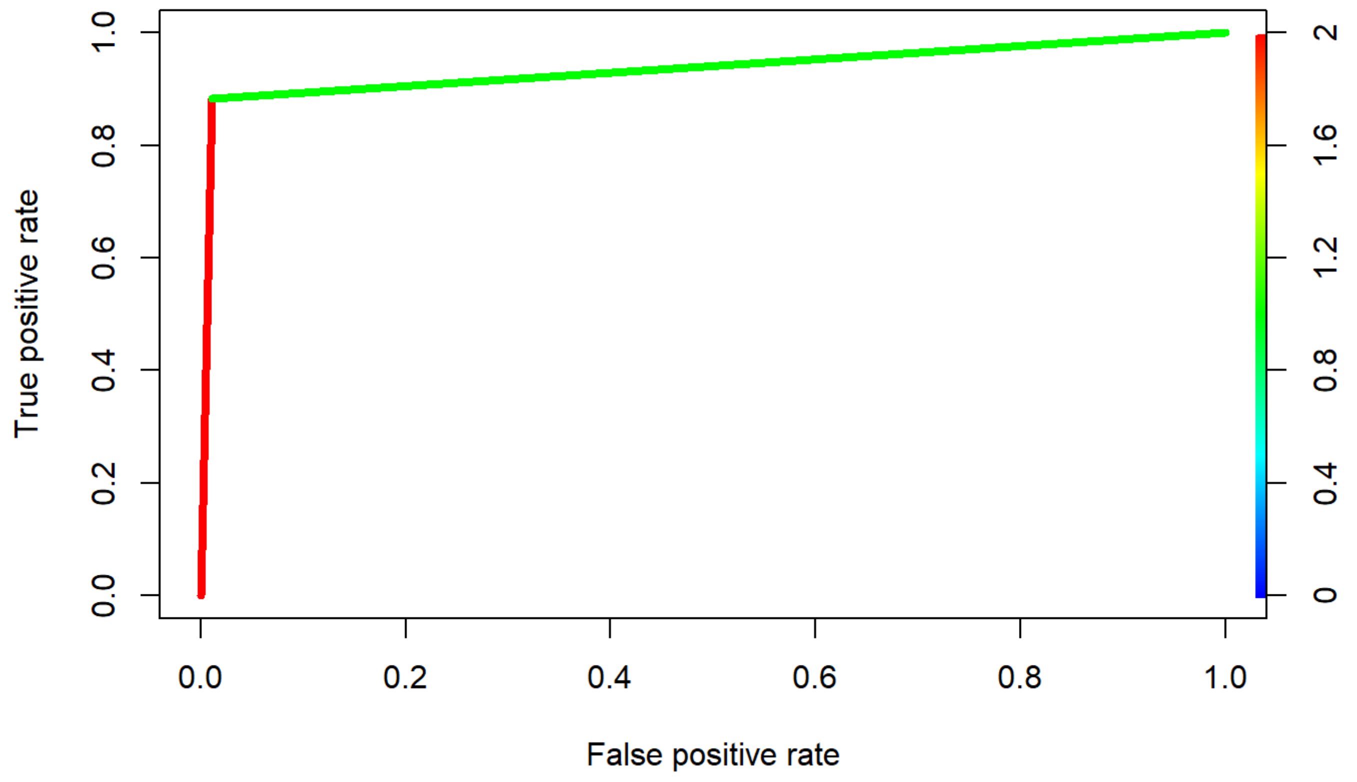
```
tab2=table(lda.pred2$class,test$type)  #预测值和真实值
tab2    #混淆矩阵
```

```
##
##        0   1
##   0 186  12
##   1   2  91
```

```
erro2=1-sum(diag(prop.table(tab2)))    #计算误判率
erro2
```

```
## [1] 0.04810997
```

```
ldap2=prediction((as.numeric(lda.pred2$class)-1),test$type) #预测值和真实值
ldaperf2=performance(ldap2,"tpr","fpr")
plot(ldaperf2,colorize=TRUE,lwd=4)
```



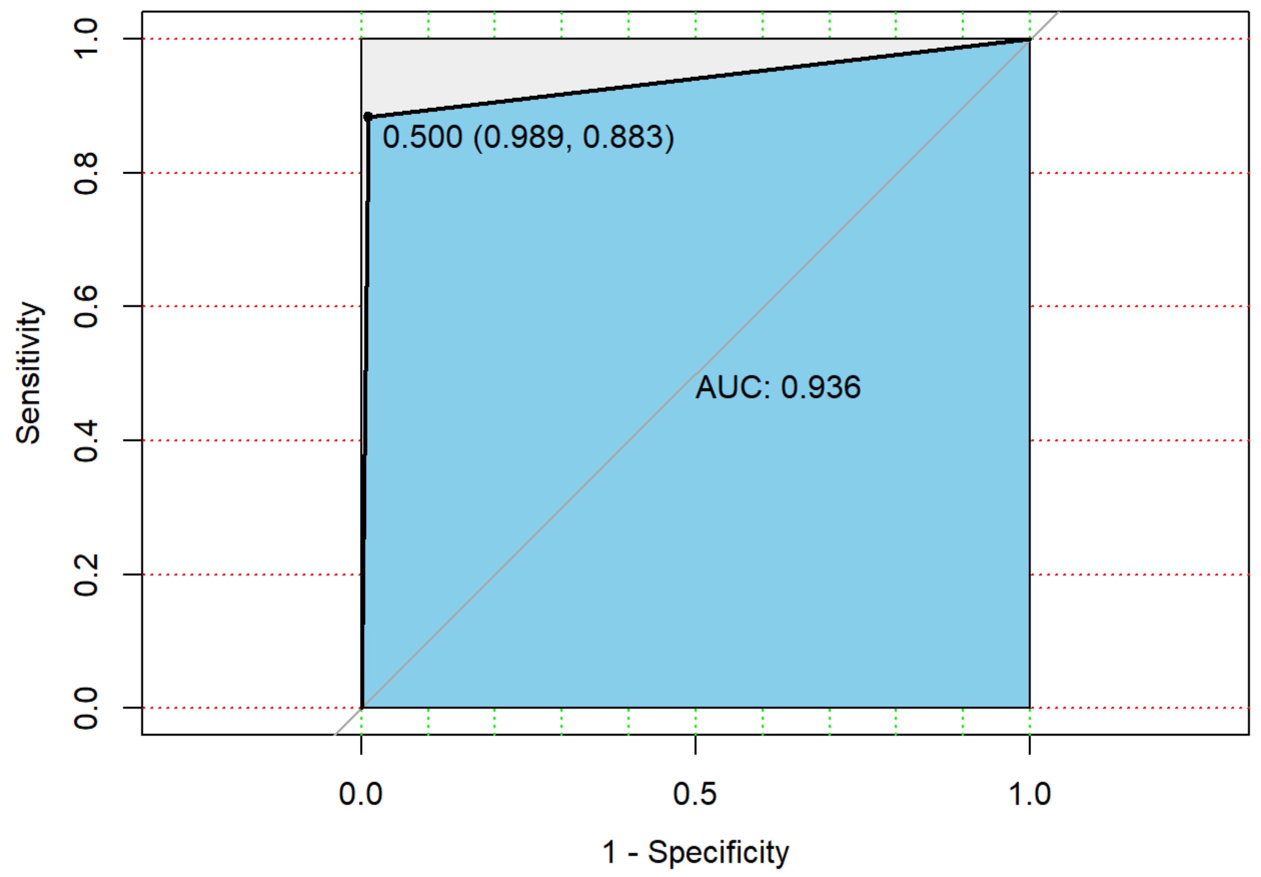
```
#AUC:0.936
performance(ldap2, "auc")@y.values[[1]]
```

```
## [1] 0.9364284
```

```
modelldaroc2=roc(test$type,(as.numeric(lda.pred2$class)-1)) #真实值和预测值
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
plot(modelldaroc2, print.auc=TRUE, auc.polygon=TRUE,legacy.axes=TRUE, grid=c(0.1, 0.2),
     grid.col=c("green", "red"), max.auc.polygon=TRUE,
     auc.polygon.col="skyblue", print.thres=TRUE)
```



```
#(5)
qda.fit=qda(type~.,data=train)
qda.fit  #Prior probabilities of groups先验概率
```

```
## Call:
## qda(type ~ ., data = train)
##
## Prior probabilities of groups:
##         0         1
## 0.6530612 0.3469388
##
## Group means:
##   Cl.thickness Cell.size Cell.shape Marg.adhesion Epith.c.size Bare.nuclei
## 0     2.878906  1.281250   1.382812      1.304688     2.097656    1.339844
## 1     7.117647  6.669118   6.617647      5.786765     5.345588    7.757353
##   Bl.cromatin Normal.nucleoli  Mitoses
## 0    2.023438        1.238281 1.066406
## 1    6.264706        6.176471 2.786765
```

```
qda.pred=predict(qda.fit,test)
qda.pred$class  #预测的所属类的结果;后验概率为qda.pred$posterior
```

```
##   [1] 1 1 1 0 0 1 0 0 1 0 1 0 0 0 0 1 1 0 1 1 1 1 1 1 1 0 0 0 0 1 0 0 0 0 0 1 0
##  [38] 0 0 0 1 1 1 0 1 1 1 1 0 1 1 0 0 1 0 0 1 0 0 1 0 0 0 1 1 1 1 1 0 0 1 0 1 1
##  [75] 1 0 1 0 1 0 1 1 0 0 0 1 1 0 1 1 0 1 1 0 0 0 0 1 0 1 1 0 0 1 1 0 1 0 1 0 0
## [112] 1 1 1 0 1 1 0 1 1 1 1 0 1 1 0 1 1 1 1 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 0 0 1
## [149] 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 1 1 1 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0
## [186] 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0
## [223] 0 0 0 1 1 0 0 0 0 0 0 1 1 0 1 0 0 0 0 1 0 0 0 1 1 1 0 0 1 0 1 0 1 0 0 0 0
## [260] 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0
## Levels: 0 1
```

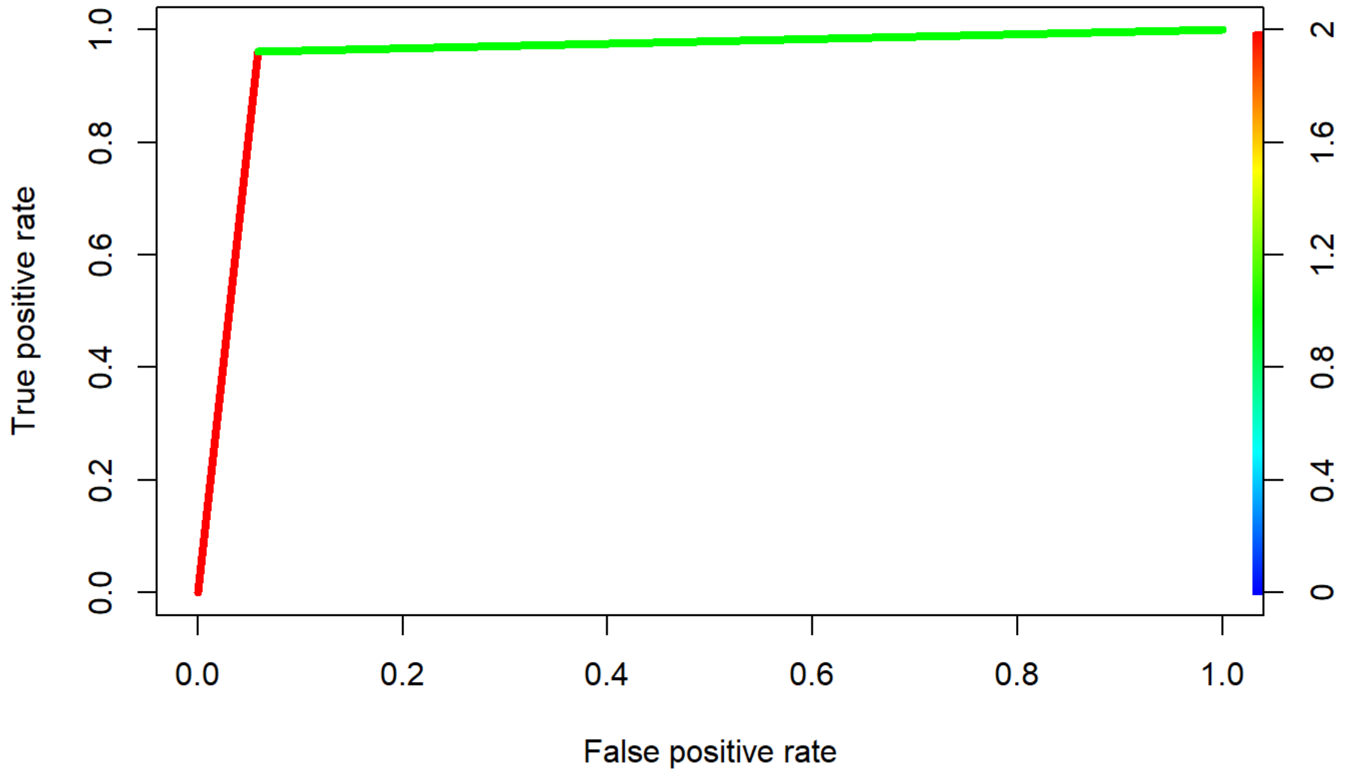
```
tabq1=table(qda.pred$class,test$type)  #预测值和真实值
tabq1    #混淆矩阵
```

```
##
##        0   1
##    0 177   4
##    1  11  99
```

```
erroq1=1-sum(diag(prop.table(tab)))    #计算误判率
erroq1
```

```
## [1] 0.04467354
```

```
qdap1=prediction((as.numeric(qda.pred$class)-1),test$type) #预测值和真实值
qdaperf1=performance(qdap1,"tpr","fpr")
plot(qdaperf1,colorize=TRUE,lwd=4)
```



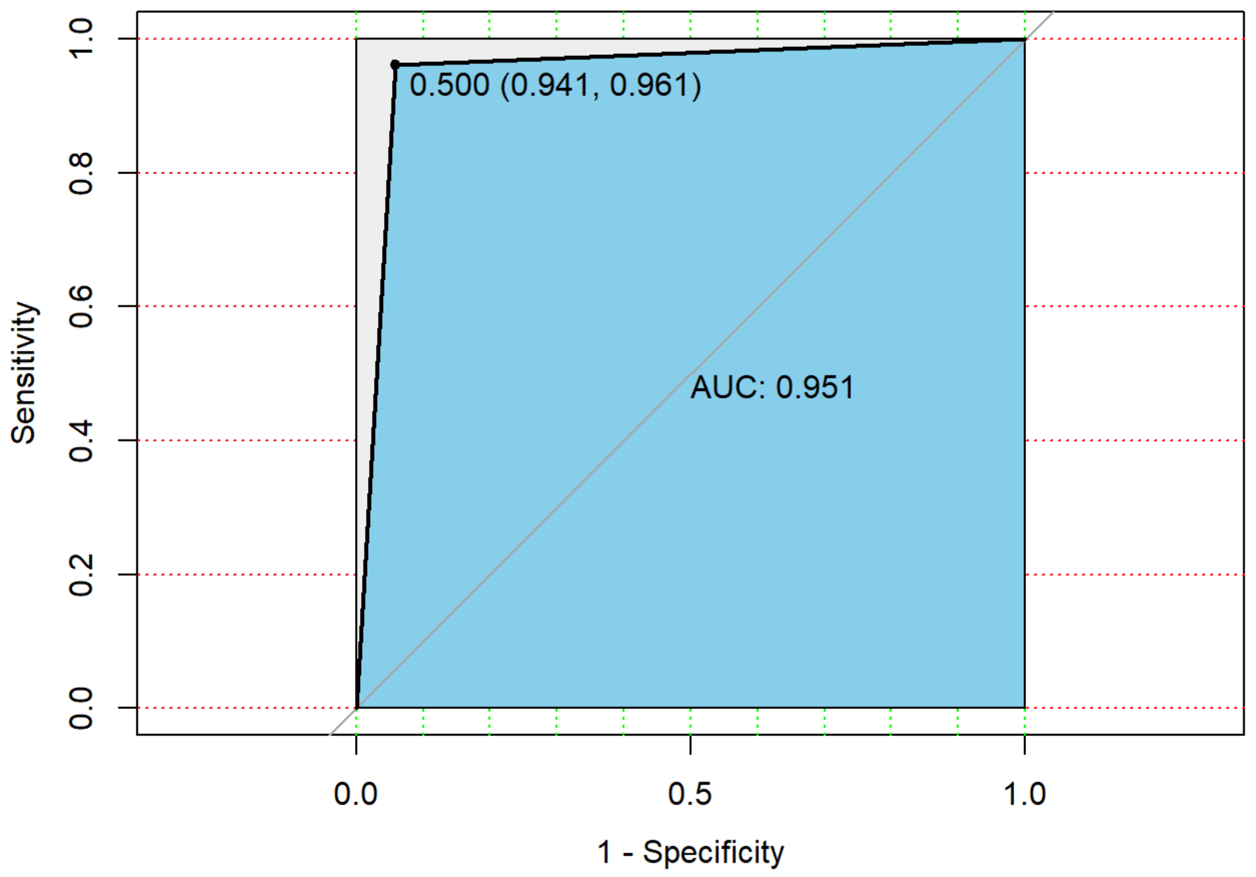
```
#AUC:0.951
performance(qdap1, "auc")@y.values[[1]]
```

```
## [1] 0.9513272
```

```
modelqdaroc1=roc(test$type,(as.numeric(qda.pred$class)-1)) #真实值和预测值
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
plot(modelqdaroc1, print.auc=TRUE, auc.polygon=TRUE,legacy.axes=TRUE, grid=c(0.1, 0.2),
     grid.col=c("green", "red"), max.auc.polygon=TRUE,
     auc.polygon.col="skyblue", print.thres=TRUE)
```



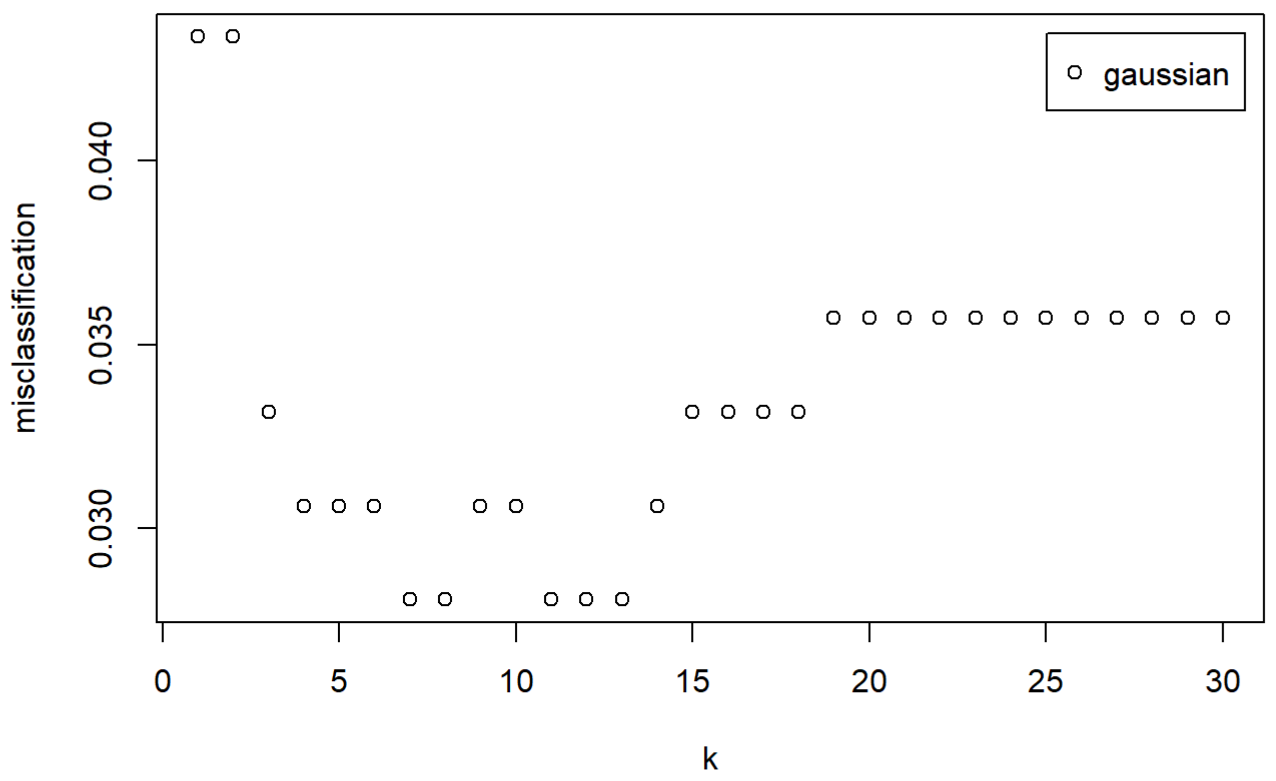
```
#(6)
#   MODEL               AUC
#glm full model         0.943
#glm smaller model      0.953
#LDA full model         0.941
#LDA smaller model      0.936
#QDA full model         0.951
```

3.

```
###3
#Actually I’ve done the homework at the exact day it’s assigned.
#So I wrote this myKNN function 2 days later when the notification announced. The result is about the same as the kknn library
function.
#But I still reserve my codes written before because maybe they will be useful in the future.
#I can write a loop for k and h for myknn function, but that’s not so efficient as the knn.cv or train.kknn in libraries to cho
ose the best k and h.
#So I still choose them for searching best k and h, and use my function to do the result check, and make sure the myknn functio
n I wrote is correct.
#These are the codes for both my function and the library functions.


myKNN_Gaussian=function(trainx,trainy,testx,testy,k=9,h=2,delta=0.5){

  M=dim(trainx)[1]
  N=dim(testx)[1]
  Xt=rbind(trainx,testx)

  Dtest=as.matrix(dist(Xt, method = "euclidean"))

  Dtest=Dtest[-c(1:M),]
  Pre=c()
  Error=c()

  for (l in 1:N) {
    Tsort=sort(Dtest[l,1:M], index.return = TRUE)
    index=Tsort$ix[1:k]
    Weight=exp((-0.5*((Tsort$x[1:k]/h)^2)))  ### you can change the kernel function here.
    Vote=trainy[Tsort$ix[1:k]]
    tent=data.frame("Weight"=Weight,"Vote"=Vote)
    tent2=aggregate(tent[, c("Weight")], list(Vote = tent$Vote), sum)

    if (dim(tent2)[1]==1){
      if (tent2[1,1]==1){
        if (delta==0){
          PreY=0
        }else{
          PreY=1
          }
        }else{
          PreY=0
        }

      }else{
        if (tent2$x[tent2$Vote==0]/(tent2$x[tent2$Vote==0]+tent2$x[tent2$Vote==1])>=delta){
          PreY=0
        }else{
          PreY=1
        }
      }

      ErrorY=PreY!=testy[1]
      Pre[1]=PreY
      Error[1]=ErrorY
    }
    return(Pre)

}



#(1)&(2)
library(class)
library(kknn)

trainx=train[,1:9]#训练集
testx=test[,1:9]#测试集
trainy=train[,10]
testy=test[,10]

### Check the best k and h at the same time:
############### kknn part in class library ###############

#测试不同方法,distance=2
# model.tkknn=train.kknn(type~.,train,kernel = c("rectangular","triangular","epanechnikov","biweight","triweight","cos","in
v","gaussian","optimal"),distance=2,scale=T,kmax=30)
# model.tkknn$MISCLASS #显示错误率;$MEAN.ABS平均绝对误差;$MEAN.SQU均方误差
# model.tkknn #输出最优参数情况
# plot(model.tkknn)

#核函数为epanechnikov时，取distance=1-4进行测试
model.tkknn1=train.kknn(type~.,train,kernel = "gaussian",distance=1,scale=T,kmax=30)
model.tkknn1 #输出最优参数情况
```

```
##
## Call:
## train.kknn(formula = type ~ ., data = train, kmax = 30, distance = 1,     kernel = "gaussian", scale = T)
##
## Type of response variable: nominal
## Minimal misclassification: 0.02806122
## Best kernel: gaussian
## Best k: 7
```

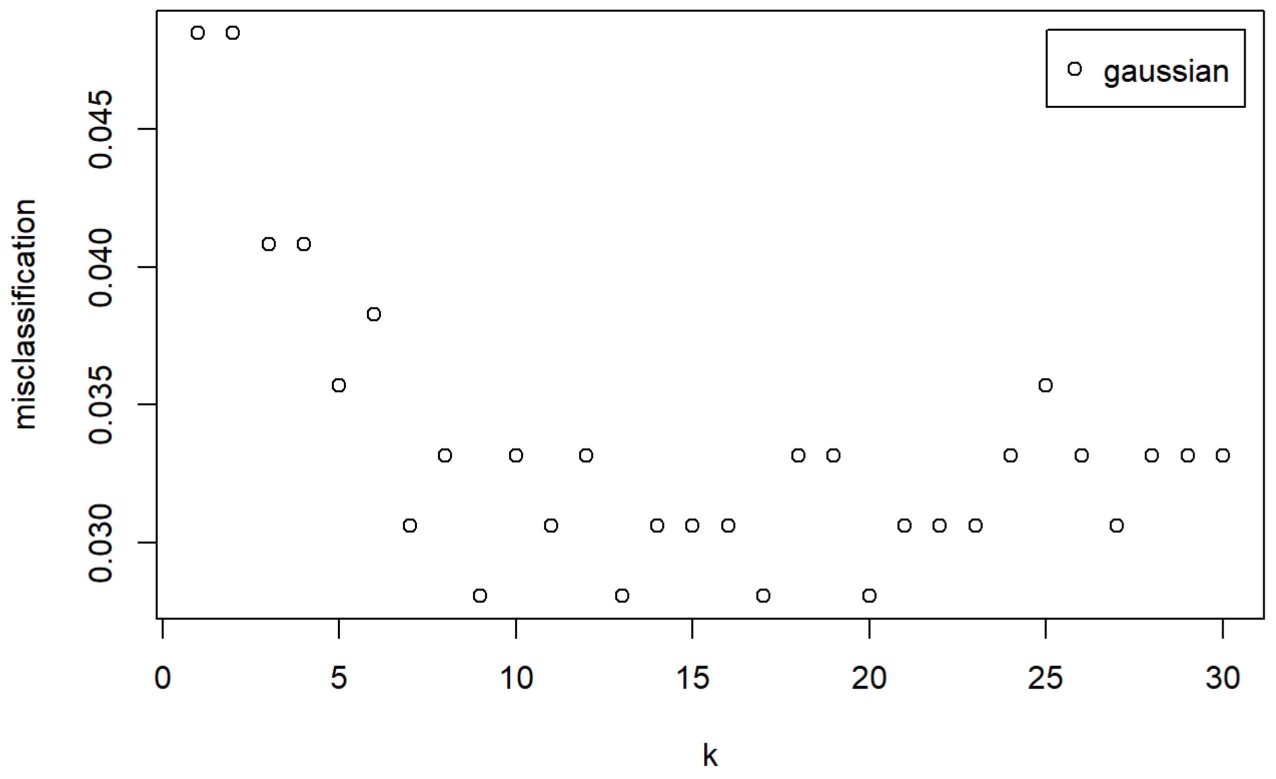
```
plot(model.tkknn1)
```



```
model.tkknn2=train.kknn(type~.,train,kernel = "gaussian",distance=2,scale=T,kmax=30)
model.tkknn2 #输出最优参数情况
```

```
##
## Call:
## train.kknn(formula = type ~ ., data = train, kmax = 30, distance = 2,     kernel = "gaussian", scale = T)
##
## Type of response variable: nominal
## Minimal misclassification: 0.02806122
## Best kernel: gaussian
## Best k: 9
```

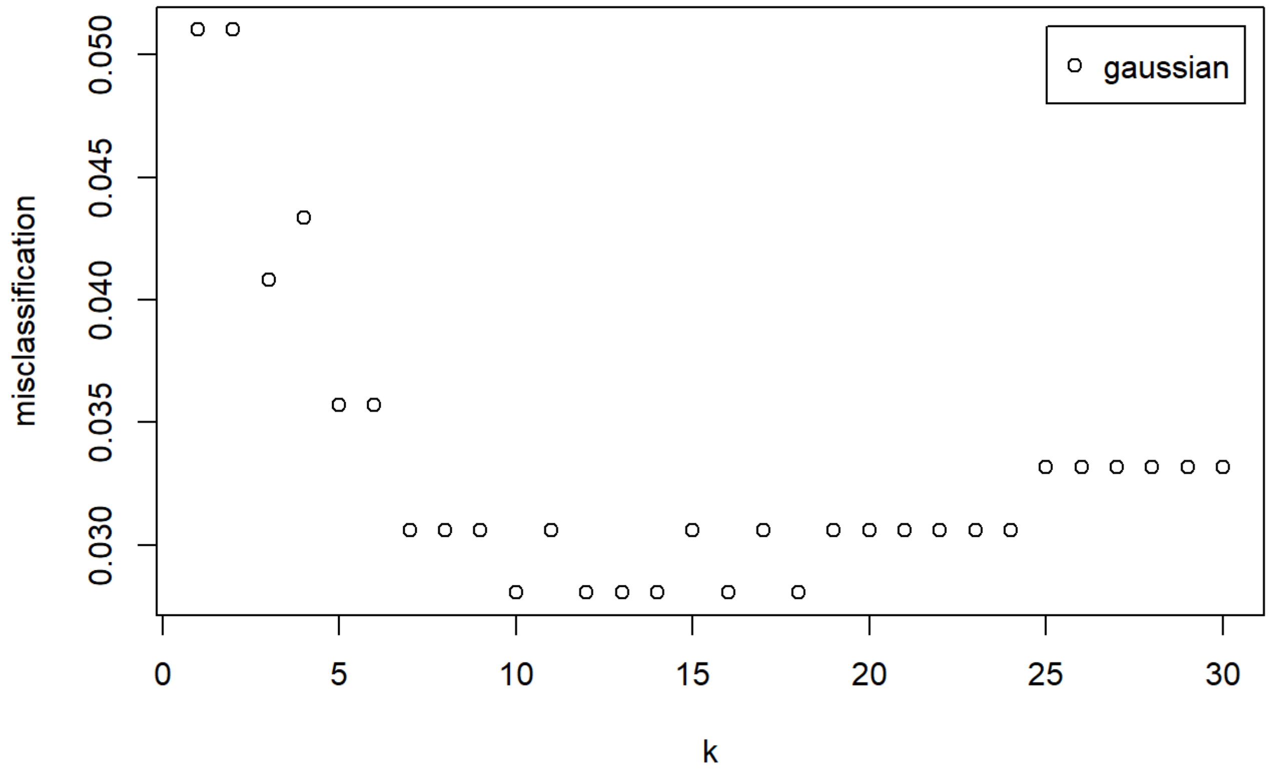
```
plot(model.tkknn2)
```



```
model.tkknn3=train.kknn(type~.,train,kernel = "gaussian",distance=3,scale=T,kmax=30)
model.tkknn3 #输出最优参数情况
```

```
##
## Call:
## train.kknn(formula = type ~ ., data = train, kmax = 30, distance = 3,     kernel = "gaussian", scale = T)
##
## Type of response variable: nominal
## Minimal misclassification: 0.02806122
## Best kernel: gaussian
## Best k: 10
```

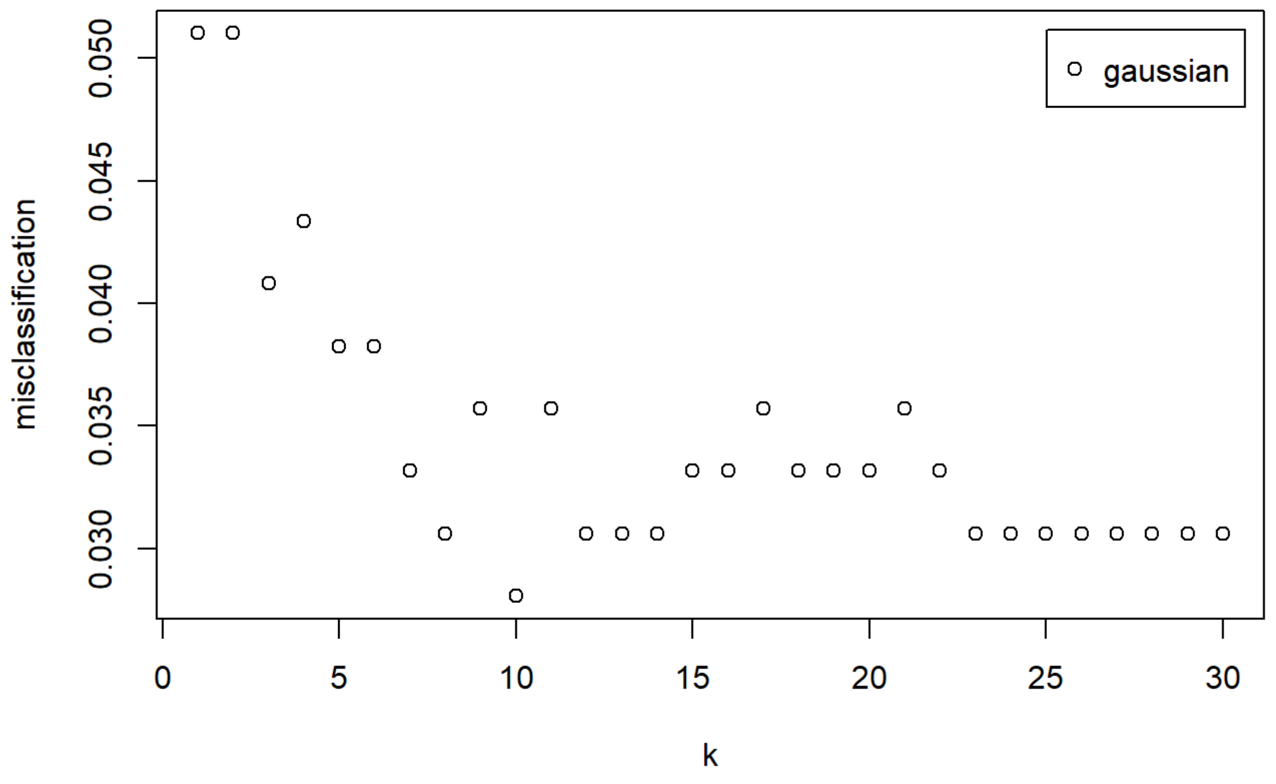
```
plot(model.tkknn3)
```



```
model.tkknn4=train.kknn(type~.,train,kernel = "gaussian",distance=4,scale=T,kmax=30)
model.tkknn4 #输出最优参数情况
```

```
##
## Call:
## train.kknn(formula = type ~ ., data = train, kmax = 30, distance = 4,     kernel = "gaussian", scale = T)
##
## Type of response variable: nominal
## Minimal misclassification: 0.02806122
## Best kernel: gaussian
## Best k: 10
```

```
plot(model.tkknn4)
```



```
#There's no significant changes so we can take distance=2 and k=9

# knntab2=table(trainy,model.tkknn2$fitted.values[[9]])
# knntab2                           #混淆矩阵
# 1-sum(diag(prop.table(knntab2)))   #计算误判率

#采用最优参数做预测
model.kknn=kknn(type~.,train,testx,k=model.tkknn2$best.parameters$k,scale=T,distance=2,kernel=model.tkknn2$best.parameters$kern
el)
# model.kknn$C #邻居的观测值号
# train[model.kknn$C,]#查看邻居
# model.kknn$D #邻居与它的距离
# model.kknn$W #邻居的权重
# model.kknn$CL #邻居的类别
# model.kknn$prob#预测的概率

#summary(model.kknn)
fit=fitted(model.kknn)
fit
```

```
##   [1] 1 1 0 0 0 1 0 0 1 0 1 0 0 0 0 1 1 0 1 1 1 1 1 1 1 0 0 0 0 1 0 0 0 0 0 1 0
##  [38] 0 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 1 0 0 1 0 0 1 0 0 0 1 1 0 1 1 0 0 1 0 0 1
##  [75] 1 0 1 0 1 0 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 0 1 0 1 1 0 0 1 1 0 1 0 1 0 0
## [112] 1 1 1 0 1 1 0 1 1 1 1 0 1 1 0 1 1 1 1 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 0 0 1
## [149] 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0
## [186] 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0
## [223] 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 1 0 0 0 1 1 1 0 0 1 0 1 0 1 0 0 0 0
## [260] 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0
## Levels: 0 1
```

```
knntab2=table(testy,fit)
knntab2                           #混淆矩阵
```

```
##      fit
## testy   0   1
##     0 184   4
##     1   6  97
```

```
1-sum(diag(prop.table(knntab2)))   #计算误判率
```

```
## [1] 0.03436426
```

```
############### myknn part to check ##########################
starttime=Sys.time()

myfit=myKNN_Gaussian(trainx,trainy,testx,testy,k=9,h=2)

knntabmy=table(testy,myfit)
knntabmy                            #混淆矩阵
```

```
##      myfit
## testy   0   1
##     0 184   4
##     1   6  97
```

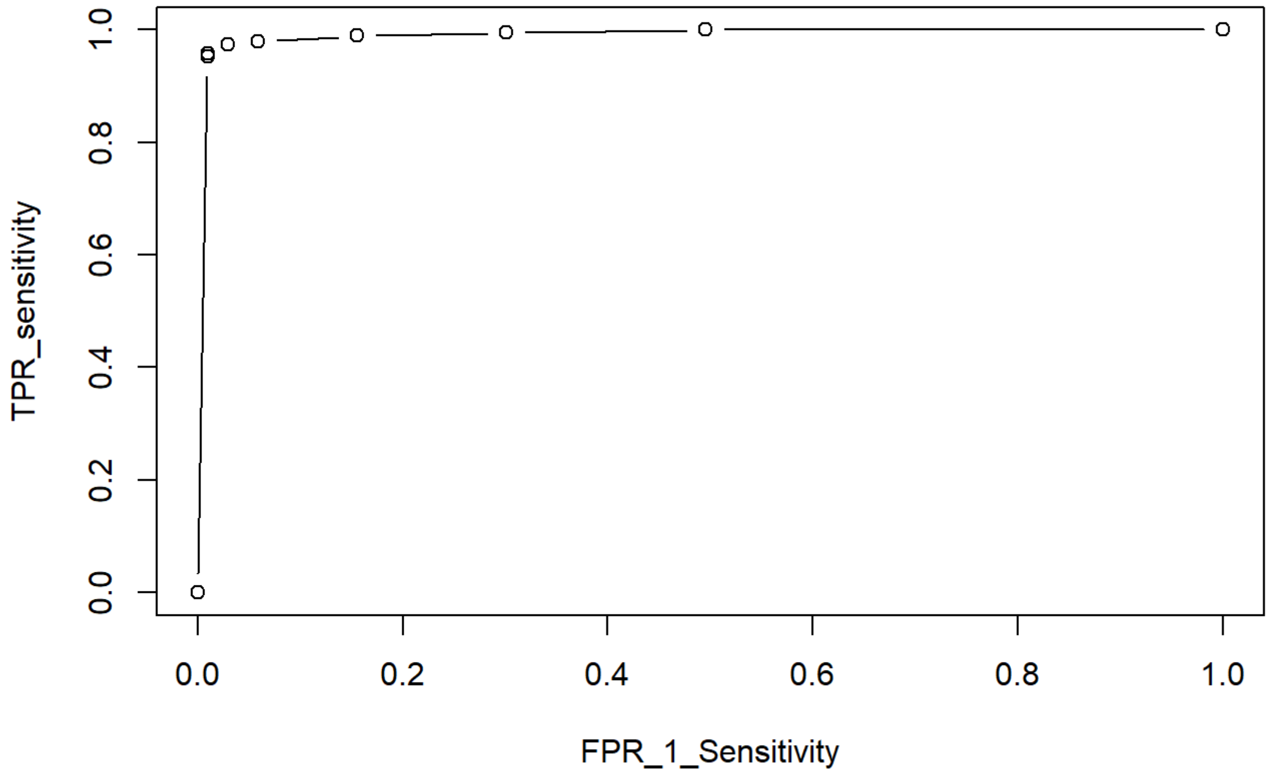
```
1-sum(diag(prop.table(knntabmy)))   #计算误判率
```

```
## [1] 0.03436426
```

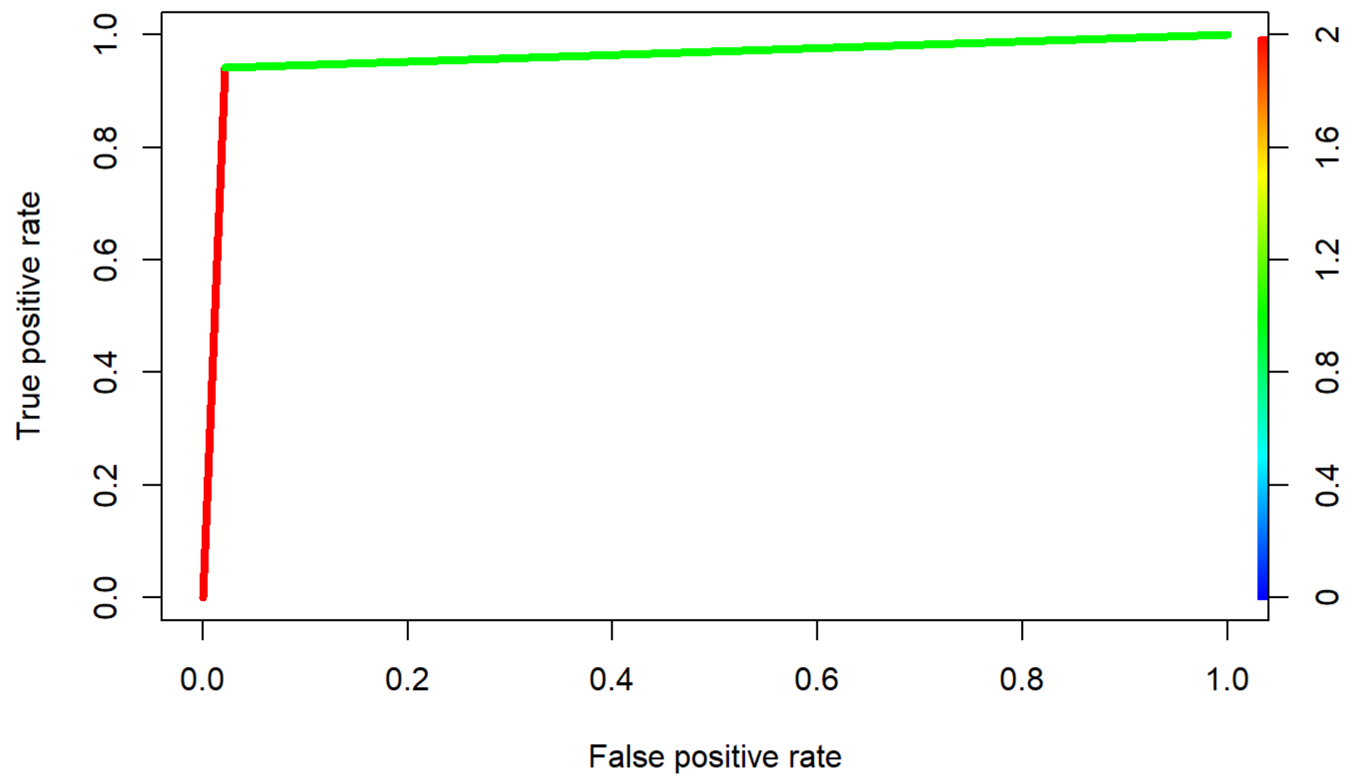
```
endtime=Sys.time()
timeinterval=endtime-starttime
timeinterval
```

```
## Time difference of 0.3480692 secs
```

```
#The function I wrote is correct with the kknn function, but a bit slower.

#Then do the ROC manually and compare it with the picture in the library:

###################### ROC manually ##########################

deltalist=c(0.001,0.01,0.02,0.1,0.2,0.5,0.8,0.9,0.98,0.99,0.999)
TPR_sensitivity=c(1)
FPR_1_Sensitivity=c(1)
for (i in 1:length(deltalist)) {
  pred_tent=rep("1",291)
  pred_tent[which(model.kknn$prob[,1]>deltalist[i])]="0"
  pred_tent=as.numeric(pred_tent)
  knntab_tent=table(pred_tent,testy)  #预测值和真实值
  TPR_sensitivity[i+1]=knntab_tent[1,1]/(knntab_tent[1,1]+knntab_tent[2,1])
  FPR_1_Sensitivity[i+1]=knntab_tent[1,2]/(knntab_tent[1,2]+knntab_tent[2,2])
}
TPR_sensitivity=append(TPR_sensitivity,0)
FPR_1_Sensitivity=append(FPR_1_Sensitivity,0)
plot(FPR_1_Sensitivity,TPR_sensitivity,type="b")
```



```
###################### compare with the library function ##########################

knnp2=prediction((as.numeric(fit)-1),testy) #预测值和真实值
knnperf2=performance(knnp2,"tpr","fpr")
plot(knnperf2,colorize=TRUE,lwd=4)
```



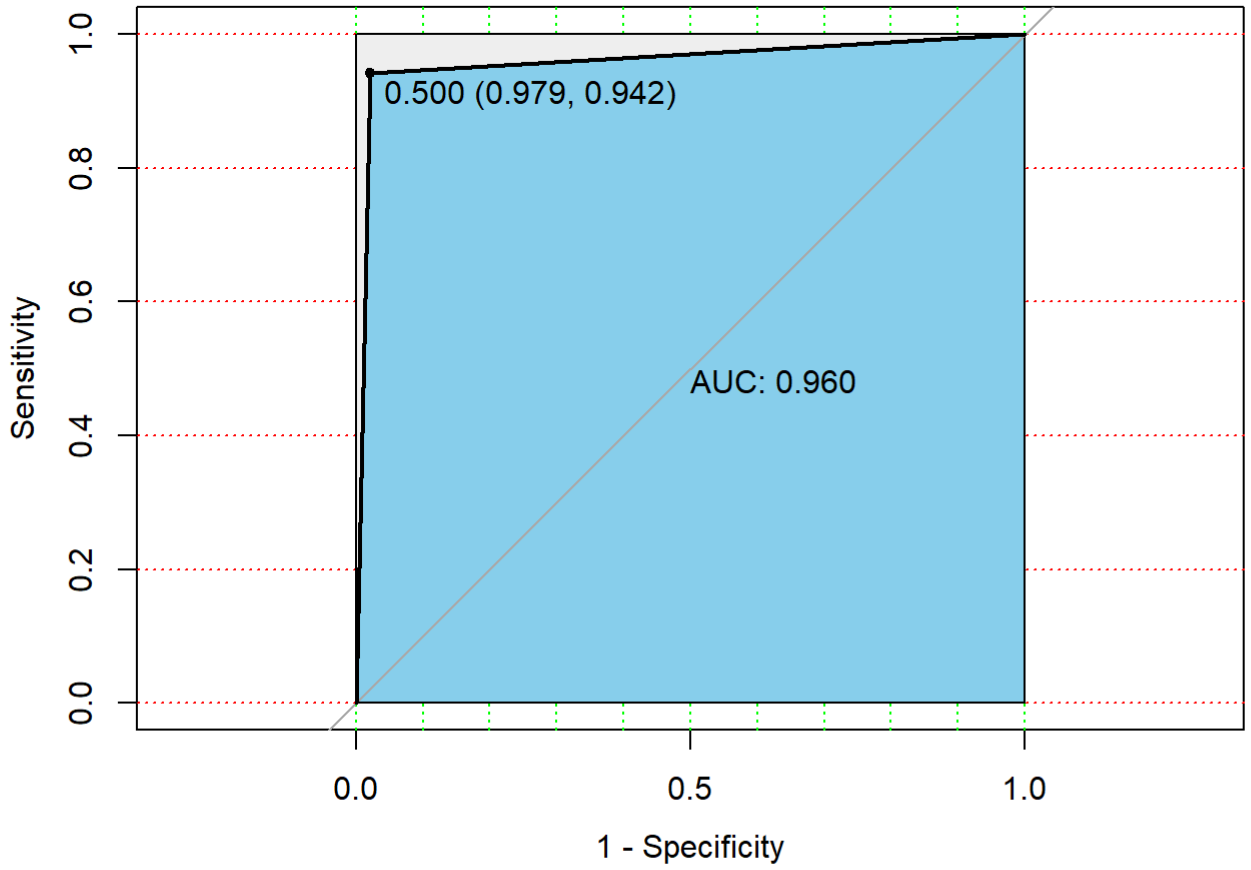
```
#AUC:0.960
performance(knnp2, "auc")@y.values[[1]]
```

```
## [1] 0.9602355
```

```
modelknnroc2=roc(testy,(as.numeric(fit)-1)) #真实值和预测值
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

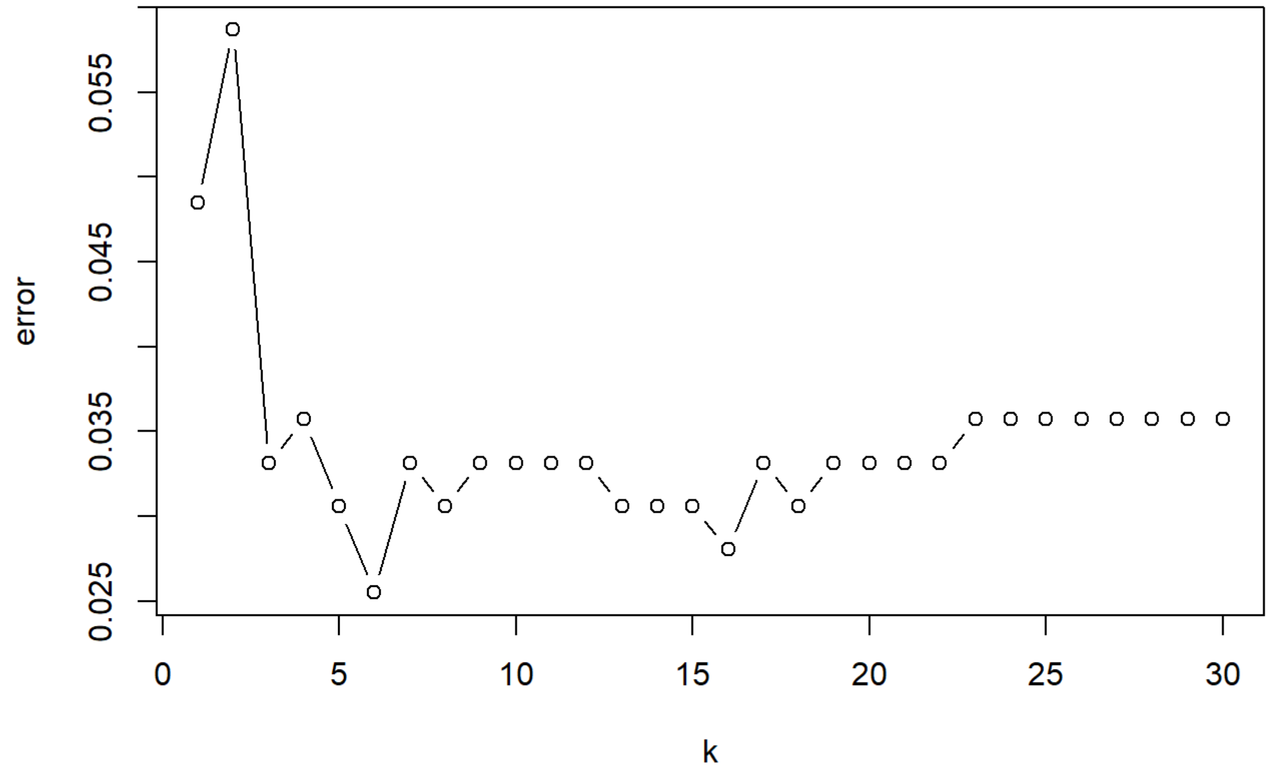
```
plot(modelknnroc2, print.auc=TRUE, auc.polygon=TRUE,legacy.axes=TRUE, grid=c(0.1, 0.2),
     grid.col=c("green", "red"), max.auc.polygon=TRUE,
     auc.polygon.col="skyblue", print.thres=TRUE)
```



```
############### knn part in class library ###############

#knn(trainx,testx,cl=trainy,k=2,prob=T)#example

#采用留一交叉验证寻找最优的k
error=rep(0,30)
for (i in 1:30){
  set.seed(1)#设置随机数种子为1
  cv1=knn.cv(trainx,cl=trainy, k=i, prob = TRUE)
  error[i]=sum(as.numeric(as.numeric(cv1)!=as.numeric(trainy)))/nrow(trainx)
  #错判率
}
error
```

```
##  [1] 0.04846939 0.05867347 0.03316327 0.03571429 0.03061224 0.02551020
##  [7] 0.03316327 0.03061224 0.03316327 0.03316327 0.03316327 0.03316327
## [13] 0.03061224 0.03061224 0.03061224 0.02806122 0.03316327 0.03061224
## [19] 0.03316327 0.03316327 0.03316327 0.03316327 0.03571429 0.03571429
## [25] 0.03571429 0.03571429 0.03571429 0.03571429 0.03571429 0.03571429
```

```
plot(error,type="b",xlab="k")
```



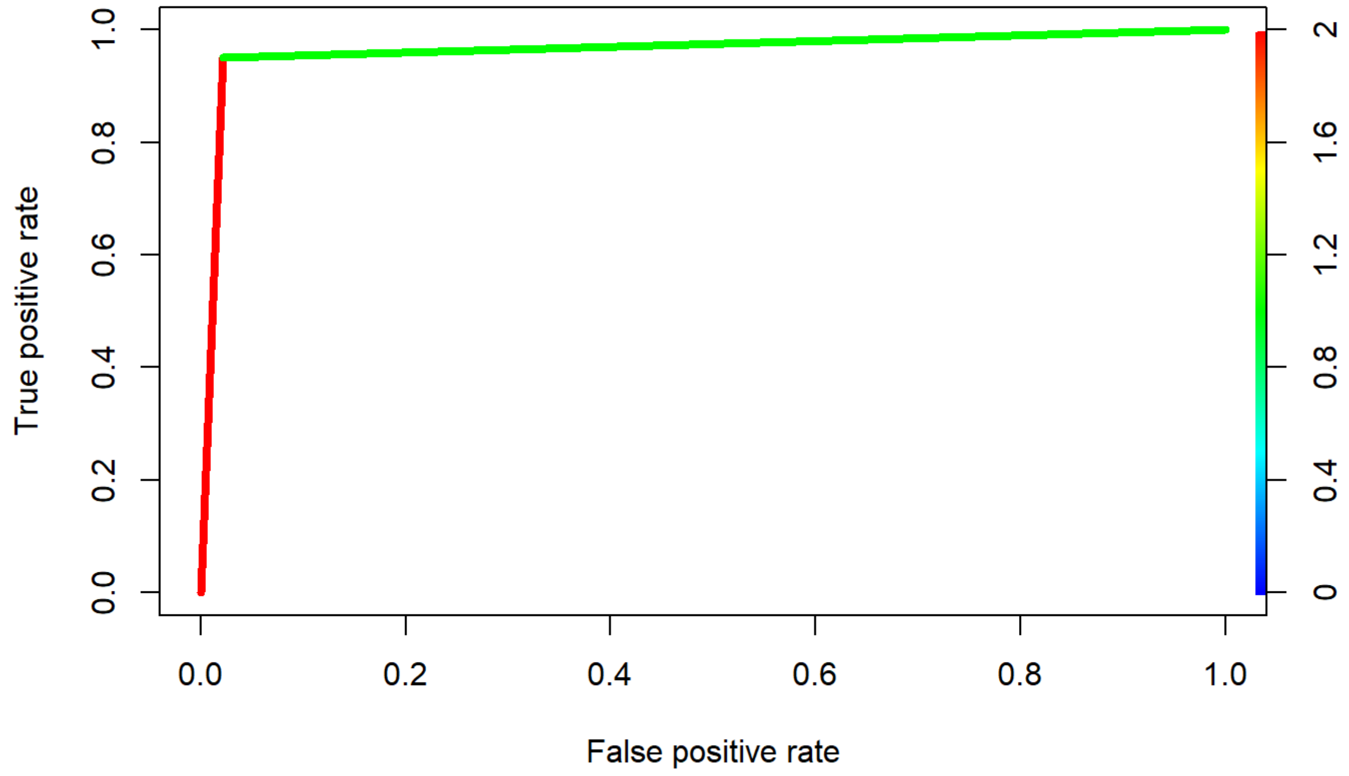
```
#the best tuned K-value is K=6

#取k=6进行预测

predict1=knn(trainx,testx,cl=trainy,k=6,prob=T)
knntab1=table(testy,predict1)
knntab1                                 #混淆矩阵
```

```
##       predict1
## testy   0   1
##     0 184   4
##     1   5  98
```

```
1-sum(diag(prop.table(knntab1)))   #计算误判率
```

```
## [1] 0.03092784
```

```
knnp1=prediction((as.numeric(predict1)-1),testy) #预测值和真实值
knnperf1=performance(knnp1,"tpr","fpr")
plot(knnperf1,colorize=TRUE,lwd=4)
```



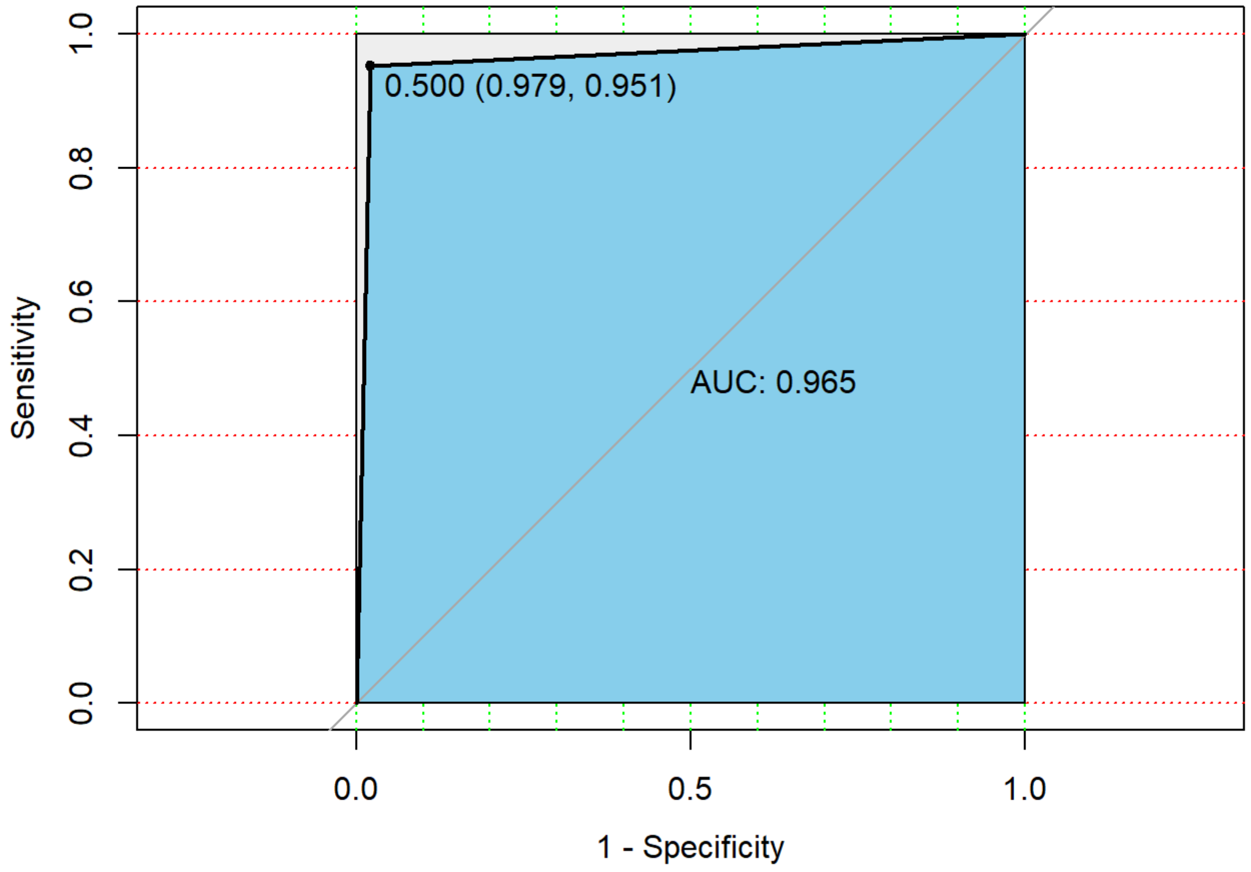
```
#AUC:0.965
performance(knnp1, "auc")@y.values[[1]]
```

```
## [1] 0.9650899
```

```
modelknnroc1=roc(testy,(as.numeric(predict1)-1)) #真实值和预测值
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
plot(modelknnroc1, print.auc=TRUE, auc.polygon=TRUE,legacy.axes=TRUE, grid=c(0.1, 0.2),
     grid.col=c("green", "red"), max.auc.polygon=TRUE,
     auc.polygon.col="skyblue", print.thres=TRUE)
```



```
#(3)
#   MODEL                 AUC
#glm full model          0.943
#glm smaller model       0.953
#LDA full model          0.941
#LDA smaller model       0.936
#QDA full model          0.951
#KNN rectangular k=6     0.965 ***√
#KNN Gaussian h=2 k=9    0.960 **
```