INTRODUCCION AL ANALISIS MULTIVARIADO

Lab. No.5 - INDICADORES DE DESEMPEÑO

1. Cargue los paquetes `caret`, `rpart`, `rattle`, `DT`, `ROCR` y `plotly`.

```
suppressMessages(library(caret))
suppressMessages(library(rpart))
suppressMessages(library(rattle))
suppressMessages(library(DT))
suppressMessages(library(ROCR))
suppressMessages(library(plotly))
```

2. Cargue la base de Credit. Declare la variable sobreendeudado como factor. Divida el archivo de datos de base2: uno para entrenar llamado `train` (80%) y el otro para predecir llamado `test` (20%). Use la instrucción `RNGkind(sample.kind = "Rounding")` y semilla igual a 10.

```
load("Credit.Rdata")
base2$sobreendeudado=factor(base2$sobreendeudado)
RNGkind(sample.kind = "Rounding")
```

```
## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
set.seed(10)
n=nrow(base2)
s=sample(1:n,round(0.8*n))
train=base2[s,]
test=base2[-s,]
```

- Haga un árbol de decisión con la base de entrenamiento (`mod1`). Obtenga la clasificación de la base de validación y llámele `pred1` . Recuerde que en el predict debe usar `type="class"` .

```
mod1 = rpart(cliente ~ ., method="class", data=train)
pred1 = predict(mod1,newdata=test,type="class")
```

- Haga un modelo de regresión logística con la base de entrenamiento y todas las variables (`mod2`). Obtenga la clasificación de la base de validación (recuerde que en el predict debe usar `type="response"` para obtener la probabilidad de éxito y luego asignar aquellos que tienen una probabilidad mayor a 0.5 al grupo 1), llámele `pred2` .

```
mod2 = glm(cliente~.,family=binomial,data=train)
pred2 = predict(mod2,newdata=test,type="response")>0.5
```

- Compare las dos clasificaciones.

```
table(pred1,pred2)
```

```
##        pred2
## pred1   FALSE TRUE
##   bueno   676   49
##   malo     49  115
```

- Obtenga la matriz de distancias entre los elementos de la base de entrenamiento y la base de validación. Use la distancia de Gower en la función `daisy` en la librería `cluster`. Pegue primero las dos bases y obtenga la matriz de distancias de todos contra todos, luego extraiga solo la parte que compara los elementos de entrenamiento contra los de validación.

```
library(cluster)
d=as.matrix(daisy(rbind(train[,-1],test[,-1])))
d=d[1:nrow(train),(nrow(train)+1):(nrow(train)+nrow(test))]
dim(d)
```

```
## [1] 3556  889
```

- Haga la clasificación de la base de validación con k=5 vecinos más cercanos y llámele `pred3`.

```
library(modeest)
```

```
## Warning: package 'modeest' was built under R version 4.1.3
```

```
## Registered S3 method overwritten by 'rmutil':
##   method         from
##   print.response httr
```

```
## Registered S3 method overwritten by 'statip':
##   method         from
##   predict.kmeans rattle
```

```
k=5
pred3=c()
for(i in 1:nrow(test)){
  d1=d[,i]
  o=order(d1)
  clas=train[o,][1:k,1]
  pred3[i]= mfv(clas)
}
```

- Compare esta clasificación con las 2 anteriores.

```
table(pred1,pred3)
```

```
##        pred3
## pred1     1   2
##   bueno 636  89
##   malo   64 100
```

```
table(pred2,pred3)
```

```
##        pred3
## pred2     1   2
##   FALSE 656  69
##   TRUE   44 120
```

3. Desarrolle un función en R llamada `eval` que le permita calcular los indicadores de desempeño (`e,FP y FNC`) derivados de la matriz de confusión para un modelo de clasificación de dos clases. La función debe recibir la variable respuesta de la base de validación y la clasificación de esa misma base obtenida con cualquier método.

```
eval=function(y,pred){
    confu= table(y,pred)
    e= 1-sum(diag(confu))/sum(confu)
    falsos=1-diag(confu)/apply(confu,1,sum)
    error=c(e,falsos)*100
    names(error)=c("e","FP","FN")
    return(list(Matriz=confu,Error=error))
}
```

- Ejecute la función para el modelo generado con el árbol de decisión y la clasificación de la base de validación (`pred1`).

```
(e1=eval(test$cliente,pred1))
```

```
## $Matriz
##        pred
## y       bueno malo
##   bueno   583   60
##   malo    142  104
##
## $Error
##        e       FP       FN
## 22.72216  9.33126 57.72358
```

- Ejecute la función para el modelo generado con la regresión logística y la clasificación de la base de validación (`pred2`).

```
(e2=eval(test$cliente,pred2))
```

```
## $Matriz
##        pred
## y       FALSE TRUE
##   bueno   584   59
##   malo    141  105
##
## $Error
##         e        FP        FN
## 22.497188  9.175739 57.317073
```

- Ejecute la función para el modelo generado con 5 vecinos más cercanos y la clasificación de la base de validación (`pred3`).

```
(e3=eval(test$cliente,pred3))
```

```
## $Matriz
##        pred
## y         1   2
##   bueno 562  81
##   malo  138 108
##
## $Error
##        e       FP       FN
## 24.63442 12.59720 56.09756
```

- Haga una tabla con los resultados de los 3 modelos.

```
E=cbind(e1$Error,e2$Error,e3$Error)
colnames(E)=c("Arbol","Logística","knn")
round(E,1)
```

```
##    Arbol Logística  knn
## e   22.7      22.5 24.6
## FP   9.3       9.2 12.6
## FN  57.7      57.3 56.1
```

6. Use la función `prediction` de la librería `ROCR` para obtener los elementos para hacer la Curva ROC y obtener el AUC. Debe dar dos variables para esta función, primero la predicción en forma numérica y el vector de respuesta: `prediction(pred,y)` . Aplique la función con el primer modelo, ponga el resultado en `predict1` .

```
predict1=prediction(as.numeric(pred1),test$cliente)
```

- Extraiga el auc con `attributes(performance(predict1,"auc"))$y.values[[1]]*100` .

```
attributes(performance(predict1,"auc"))$y.values[[1]]*100
```

```
## [1] 66.47258
```

- Para extraer los falsos positivos y la precisión positiva haga `performance(predict1,"tpr","fpr")` . Guarde esto en `des` y luego extraiga los falsos positivos con `attributes(des)$x.values[[1]]*100` y la precisión positiva con `attributes(des)$y.values[[1]]*100` .

```
des = performance(predict1,"tpr","fpr")
attributes(des)$x.values[[1]]*100
```

```
## [1]   0.00000   9.33126 100.00000
```

```
attributes(des)$y.values[[1]]*100
```

```
## [1]   0.00000  42.27642 100.00000
```

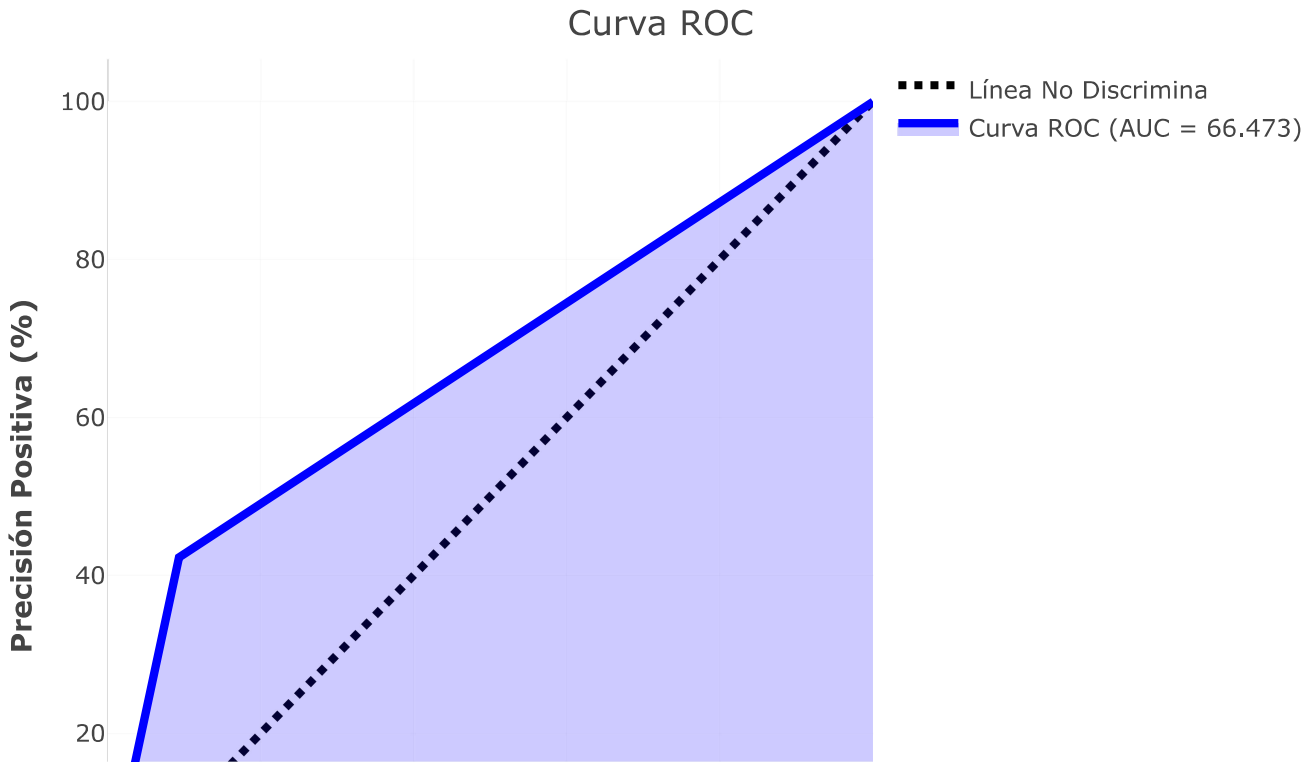
- Escriba la función que hace el gráfico de la Curva ROC y calcula el AUC. Use la siguiente función:

```
curvaROC = function(pred,y, grafico = F) {
  predict = prediction(pred,y)
  auc = attributes(performance(predict,"auc"))$y.values[[1]]*100
  des = performance(predict,"tpr","fpr")
  p = NULL
  if(grafico){
    FP = attributes(des)$x.values[[1]]*100
    PP = attributes(des)$y.values[[1]]*100
    p <- plot_ly(x = FP, y = FP, name = 'Línea No Discrimina',
                 type = 'scatter', mode = 'lines',
                 line = list(color = 'rgba(0, 0, 0, 1)',
                             width = 4, dash = 'dot'),
                 fill = 'tozeroy',  fillcolor = 'rgba(0, 0, 0, 0)') %>%
      add_trace(y = PP, name = paste('Curva ROC (AUC = ', round(auc,3),')', sep =""),
                line = list(color = 'rgba(0, 0, 255, 1)', width = 4,
                dash = 'line'),  fillcolor = 'rgba(0, 0, 255, 0.2)')%>%
      layout(title = "Curva ROC",
             xaxis = list(title = "<b>Falsos Positivos (%)<b>"),
             yaxis = list (title = "<b>Precisión Positiva (%)<b>"))
  }
  return(list(auc = auc,grafico = p))
}
```

- Haga el gráfico de la Curva ROC y calcule el AUC para los 3 modelos generados anteriormente.

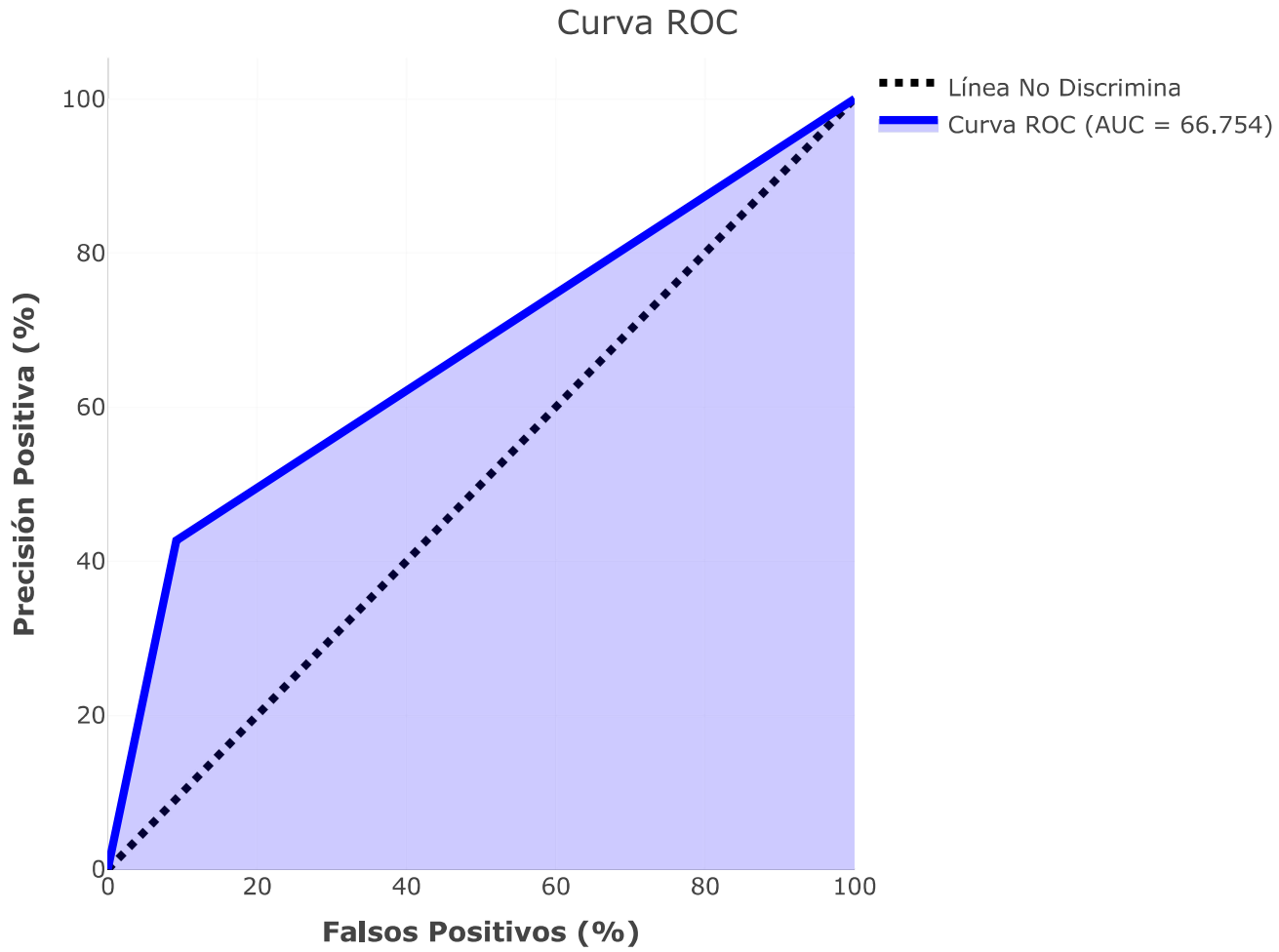
```
ROC1=curvaROC(as.numeric(pred1),test$cliente, grafico = TRUE)
ROC2=curvaROC(as.numeric(pred2),test$cliente, grafico = TRUE)
ROC3=curvaROC(pred3,test$cliente, grafico = TRUE)

ROC1$g
```

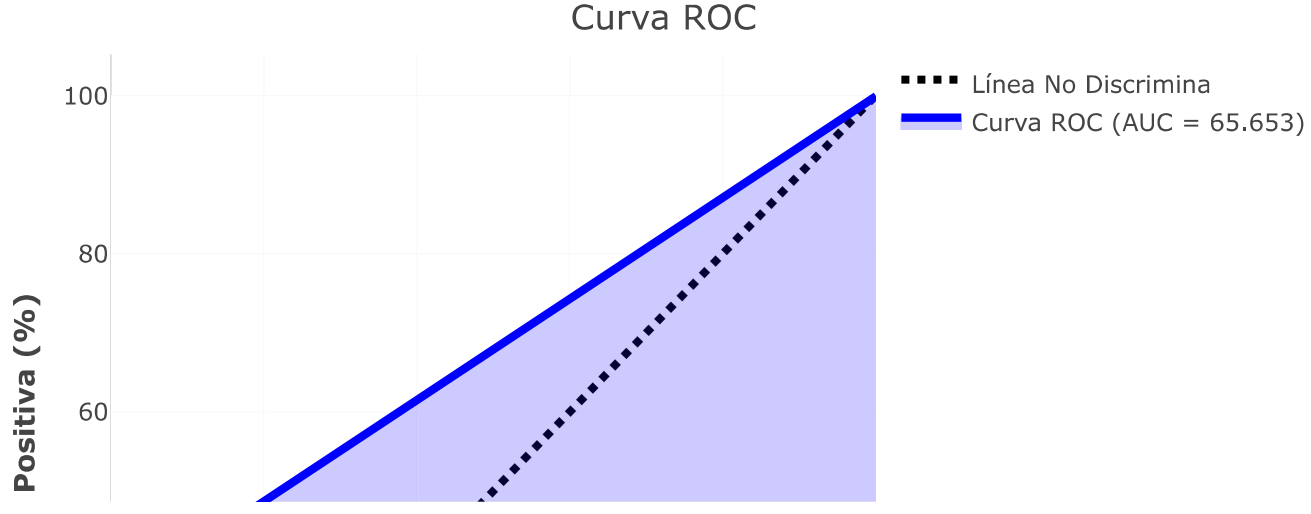


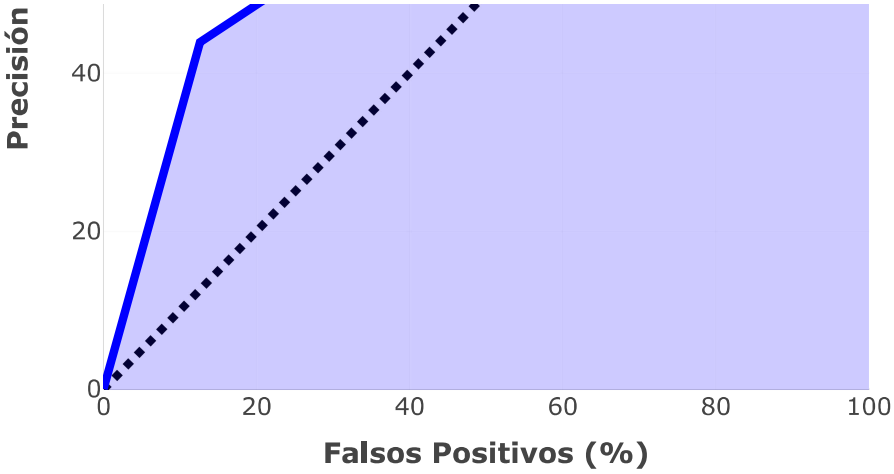

0
0
20
40
60
80
100
Falsos Positivos (%)

ROC2$g


Curva ROC
Precisión Positiva (%)
100
80
60
40
20
0
0
20
40
60
80
100
Falsos Positivos (%)
Línea No Discrimina
Curva ROC (AUC = 66.754)

ROC3$g


Curva ROC
Positiva (%)
100
80
60
Línea No Discrimina
Curva ROC (AUC = 65.653)




```
c(ROC1$auc,ROC2$auc,ROC3$auc)
```

```
## [1] 66.47258 66.75359 65.65262
```

7. Para calcular el KS del primer modelo obtenga el máximo de las diferencias entre precisión positiva menos los falsos positivos.

```
max(attributes(des)$y.values[[1]]*100 -
          attributes(des)$x.values[[1]]*100)
```

```
## [1] 32.94516
```

- Calcule el KS para los modelos generados en los ejercicios anteriores. Use la siguiente función:

```
KS = function(pred,y) {
  predictions = prediction(pred,y)
  des = performance(predictions,"tpr","fpr")
  ks = max(attributes(des)$y.values[[1]]*100 -
           attributes(des)$x.values[[1]]*100)
  return(ks)
}
```

- Calcule el KS para los 3 modelos generados anteriormente.

```
KS1=KS(as.numeric(pred1),test$cliente)
KS2=KS(as.numeric(pred2),test$cliente)
KS3=KS(pred3,test$cliente)

round(c(KS1,KS2,KS3),1)
```

```
## [1] 32.9 33.5 31.3
```

- Modifique la función `eval` para que devuelva el `AUC` y el `KS`.

```
eval=function(y,pred){
    confu= table(y,pred)
    e= 1-sum(diag(confu))/sum(confu)
    falsos=1-diag(confu)/apply(confu,1,sum)
    error=c(e,falsos)*100
    auc = curvaROC(pred,y)$auc
    KS = KS(pred,y)
    indicadores=c(error,auc,KS)
    names(indicadores)=c("e","FP","FN","AUC","KS")
    return(list(Matriz=confu,indicadores=indicadores))
}
```

- Haga una tabla con los resultados de los 3 modelos.

```
(f1=eval(test$cliente,as.numeric(pred1)))
```

```
## $Matriz
##        pred
## y         1   2
##   bueno 583  60
##   malo  142 104
##
## $indicadores
##        e       FP       FN      AUC       KS
## 22.72216  9.33126 57.72358 66.47258 32.94516
```

```
(f2=eval(test$cliente,as.numeric(pred2)))
```

```
## $Matriz
##        pred
## y         0   1
##   bueno 584  59
##   malo  141 105
##
## $indicadores
##         e        FP        FN       AUC        KS
## 22.497188  9.175739 57.317073 66.753594 33.507188
```

```
(f3=eval(test$cliente,pred3))
```

```
## $Matriz
##        pred
## y         1   2
##   bueno 562  81
##   malo  138 108
##
## $indicadores
##        e       FP       FN      AUC       KS
## 24.63442 12.59720 56.09756 65.65262 31.30524
```

```
ID=cbind(f1$indicadores,f2$indicadores,f3$indicadores)
colnames(ID)=c("Arbol","Logística","knn")
round(ID,1)
```

```
##     Arbol Logística  knn
## e    22.7      22.5 24.6
## FP    9.3       9.2 12.6
## FN   57.7      57.3 56.1
## AUC  66.5      66.8 65.7
## KS   32.9      33.5 31.3
```

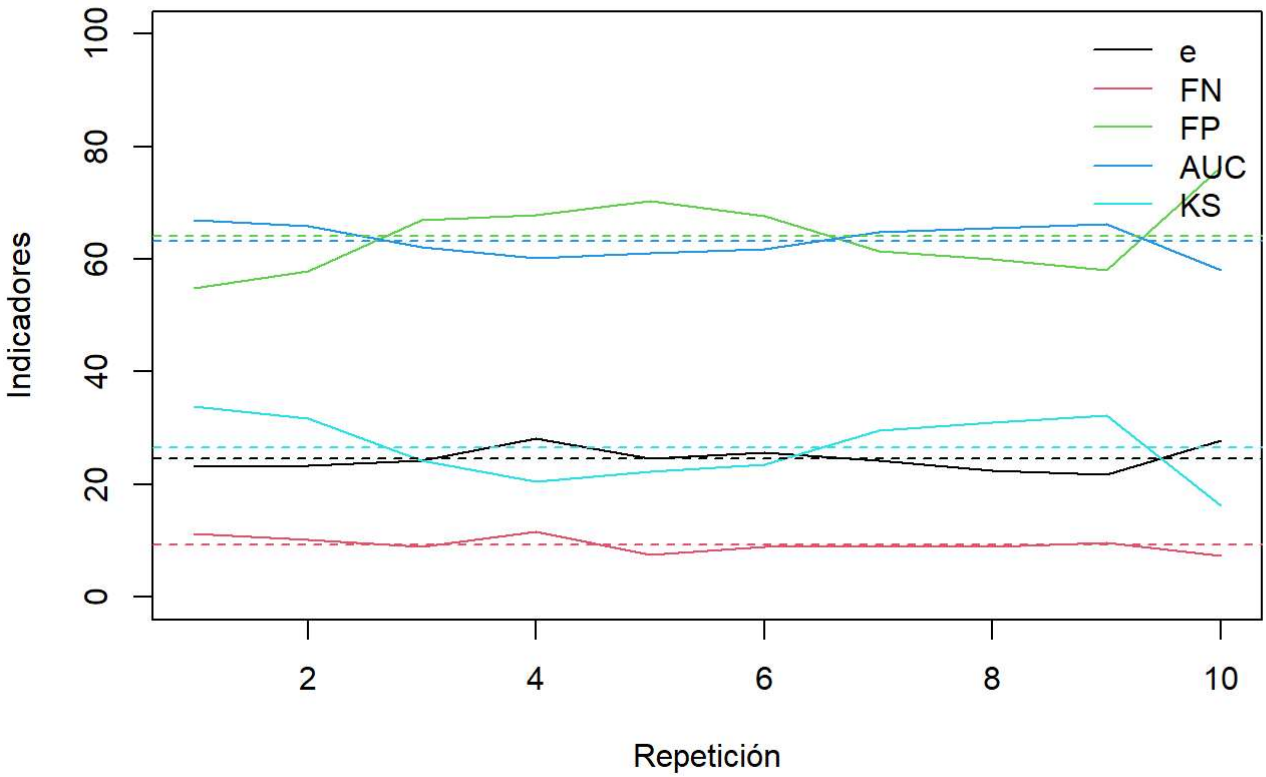
8. Haga 10 veces la partición de base2 en conjuntos de entrenamiento (80%) y prueba (20%) y en cada caso genere un árbol de decisión basado en el conjunto de entrenamiento, calcule los indicadores de desempeño para el conjunto de validación: e, FN, FP, AUC y KS. Haga un gráfico de estos indicadores para ver qué tanto varían e indique la media de los mismos.

```
res4=matrix(nrow=10,ncol=5)
colnames(res4)=c("e","FN","FP","AUC","KS")

for(i in 1:10){
  s=sample(1:n,round(0.8*n))
  train4=base2[s,]
  test4=base2[-s,]
  mod4 = rpart(cliente ~ ., method="class", data=train4)
  pred4 = predict(mod4,newdata=test4,type="class")
  f4=eval(test4$cliente,as.numeric(pred4))
  res4[i,]=f4$indicadores
}
round(res4,1)
```

```
##          e   FN   FP  AUC   KS
##  [1,] 23.2 11.2 54.9 67.0 33.9
##  [2,] 23.4 10.2 58.0 65.9 31.8
##  [3,] 24.3  8.9 66.9 62.1 24.2
##  [4,] 28.1 11.6 67.8 60.3 20.6
##  [5,] 24.6  7.4 70.4 61.1 22.2
##  [6,] 25.6  9.0 67.6 61.7 23.4
##  [7,] 24.2  9.0 61.5 64.8 29.5
##  [8,] 22.5  9.0 60.0 65.5 31.0
##  [9,] 21.8  9.6 58.0 66.2 32.3
## [10,] 27.8  7.3 76.4 58.1 16.2
```

```
matplot(res4,type="l",lty=1,ylim=c(0,100),ylab="Indicadores",xlab="Repetición")
legend("topright",colnames(res4),col=1:5,lty=1,bty="n")
medias4=apply(res4,2,mean)
abline(h=medias4,col=1:5,lty=2)
```



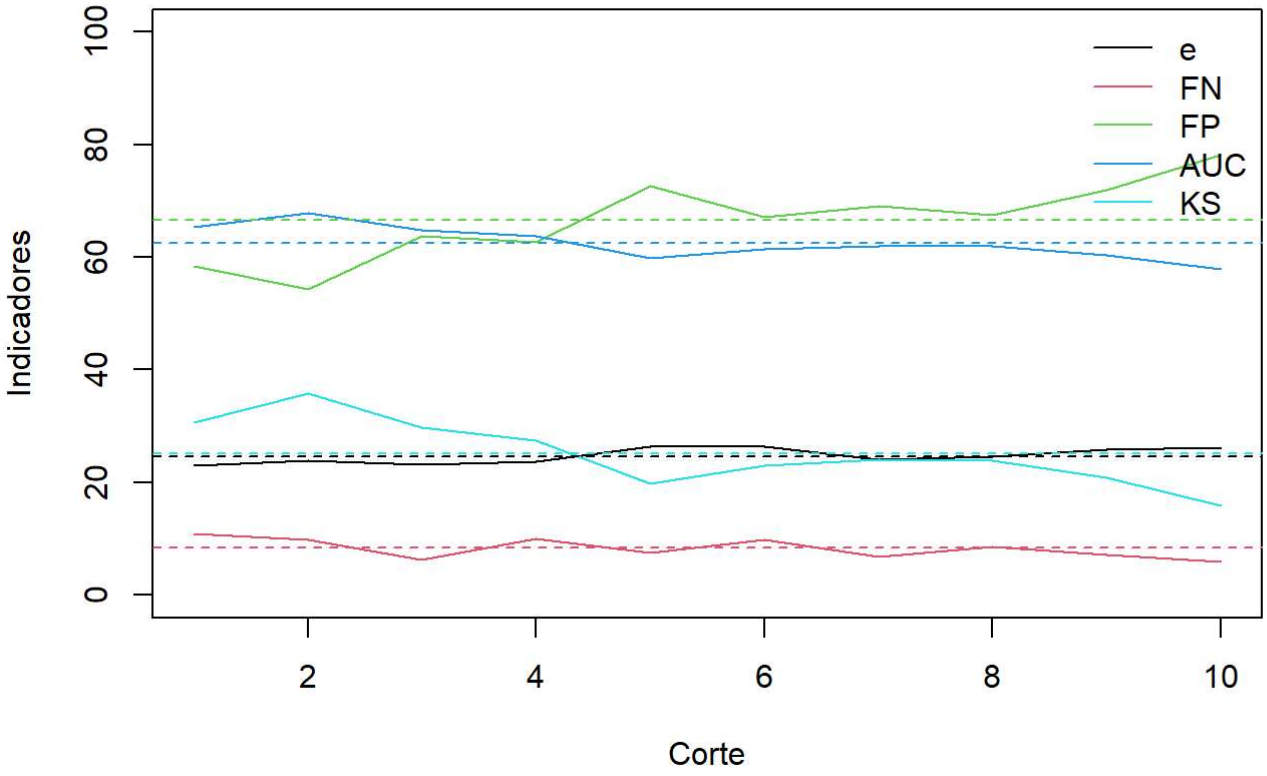
9. Haga la validación cruzada partiendo la base2 en 10 partes aproximadamente del mismo número de datos. Obtenga los indicadores de desempeño. Use la función `createFolds` de la librería `caret` . Haga un gráfico de estos indicadores para ver qué tanto varían e indique la media de los mismos.

```
cortes=createFolds(1:nrow(base2),k=10)
res5=matrix(nrow=10,ncol=5)
colnames(res5)=c("e","FN","FP","AUC","KS")

for(i in 1:10){
  train5=base2[-cortes[[i]],]
  test5=base2[cortes[[i]],]
  mod5 = rpart(cliente ~ ., method="class", data=train5)
  pred5 = predict(mod5,newdata=test5,type="class")
  f5=eval(test5$cliente,as.numeric(pred5))
  res5[i,]=f5$indicadores
}
round(res5,1)
```

```
##          e   FN   FP  AUC   KS
##  [1,] 23.0 10.9 58.4 65.4 30.7
##  [2,] 23.8  9.8 54.3 67.9 35.9
##  [3,] 23.1  6.3 63.8 64.9 29.8
##  [4,] 23.6 10.0 62.6 63.7 27.4
##  [5,] 26.4  7.6 72.7 59.9 19.7
##  [6,] 26.4  9.8 67.2 61.5 23.0
##  [7,] 24.0  6.8 69.1 62.0 24.1
##  [8,] 24.5  8.6 67.5 61.9 23.9
##  [9,] 25.8  7.2 71.9 60.4 20.9
## [10,] 26.1  5.9 78.2 57.9 15.8
```

```
matplot(res5,type="l",lty=1,ylim=c(0,100),ylab="Indicadores",xlab="Corte")
legend("topright",colnames(res5),col=1:5,lty=1,bty="n")
medias5=apply(res5,2,mean)
abline(h=medias5,col=1:5,lty=2)
```



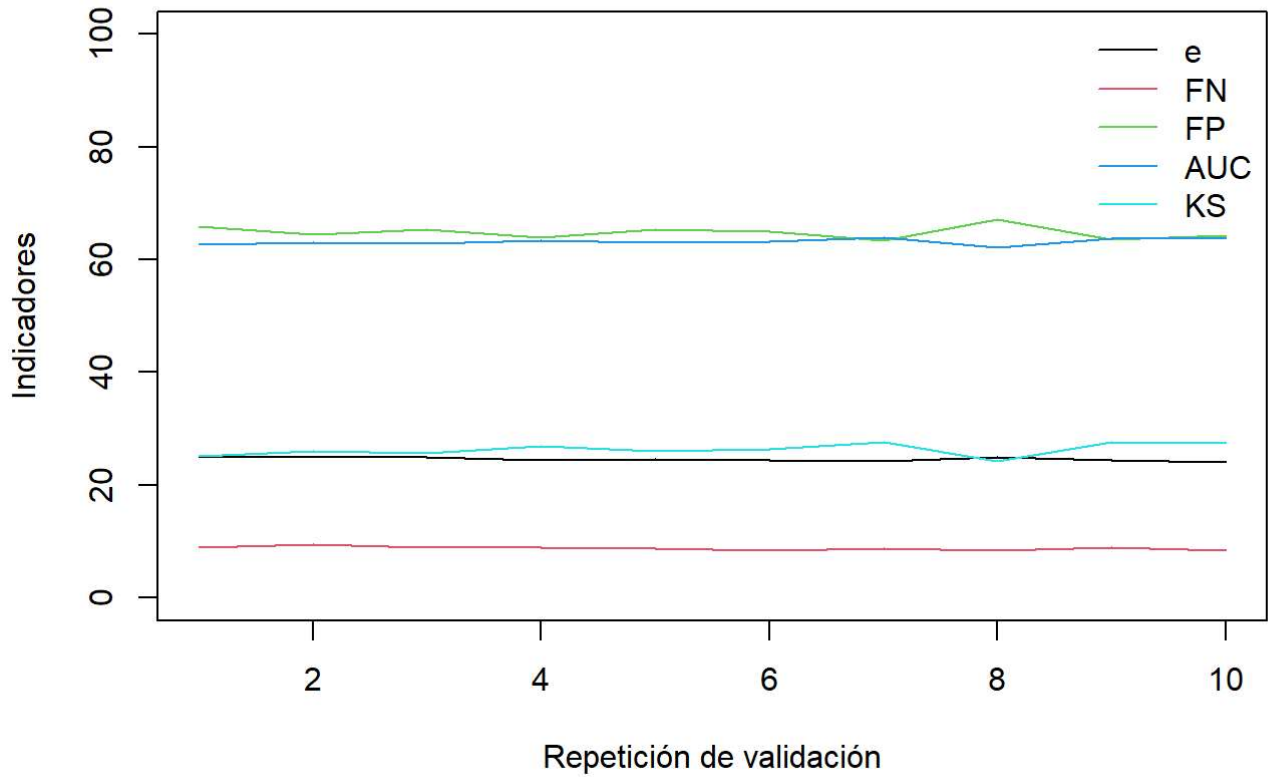
10. Haga 10 veces la validación cruzada de base2. Haga un gráfico de las medias de los indicadores de desempeño para ver qué tanto varían.

```
medias6=matrix(nrow=10,ncol=5)
for(j in 1:10){
  cortes=createFolds(1:nrow(base2),k=10)
  res6=matrix(nrow=10,ncol=5)
  colnames(res6)=c("e","FN","FP","AUC","KS")

  for(i in 1:10){
    train6=base2[-cortes[[i]],]
    test6=base2[cortes[[i]],]
    mod6 = rpart(cliente ~ ., method="class", data=train6)
    pred6 = predict(mod6,newdata=test6,type="class")
    f6=eval(test6$cliente,as.numeric(pred6))
    res6[i,]=f6$indicadores
  }
  medias6[j,]=apply(res6,2,mean)
}
colnames(medias6)=colnames(res6)
round(medias6,1)
```

```
##          e  FN   FP  AUC   KS
##  [1,] 24.9 8.9 65.8 62.6 25.2
##  [2,] 24.9 9.5 64.5 63.0 26.0
##  [3,] 25.0 9.0 65.4 62.8 25.7
##  [4,] 24.5 9.0 64.0 63.5 27.0
##  [5,] 24.6 8.7 65.3 63.0 26.0
##  [6,] 24.3 8.5 65.1 63.2 26.5
##  [7,] 24.2 8.8 63.5 63.9 27.7
##  [8,] 25.0 8.5 67.2 62.2 24.3
##  [9,] 24.3 8.9 63.5 63.8 27.5
## [10,] 24.1 8.3 64.2 63.7 27.5
```

```
matplot(medias6,type="l",lty=1,ylim=c(0,100),ylab="Indicadores",xlab="Repetición de validación")
legend("topright",colnames(medias6),col=1:5,lty=1,bty="n")
```



Pongo los mismos límites en el eje Y que en los gráficos anteriores para poder apreciar cómo ahora el rango es más corto.