imbalace

Alberto Armijo Ruiz

5 de febrero de 2019

Cargamos librerías necesarias.

```r
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(pROC)
```

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

##
## Attaching package: 'pROC'

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

```r
library(tidyr)
library(imbalance)
```

Cargamos funciones para entrenar los datos.

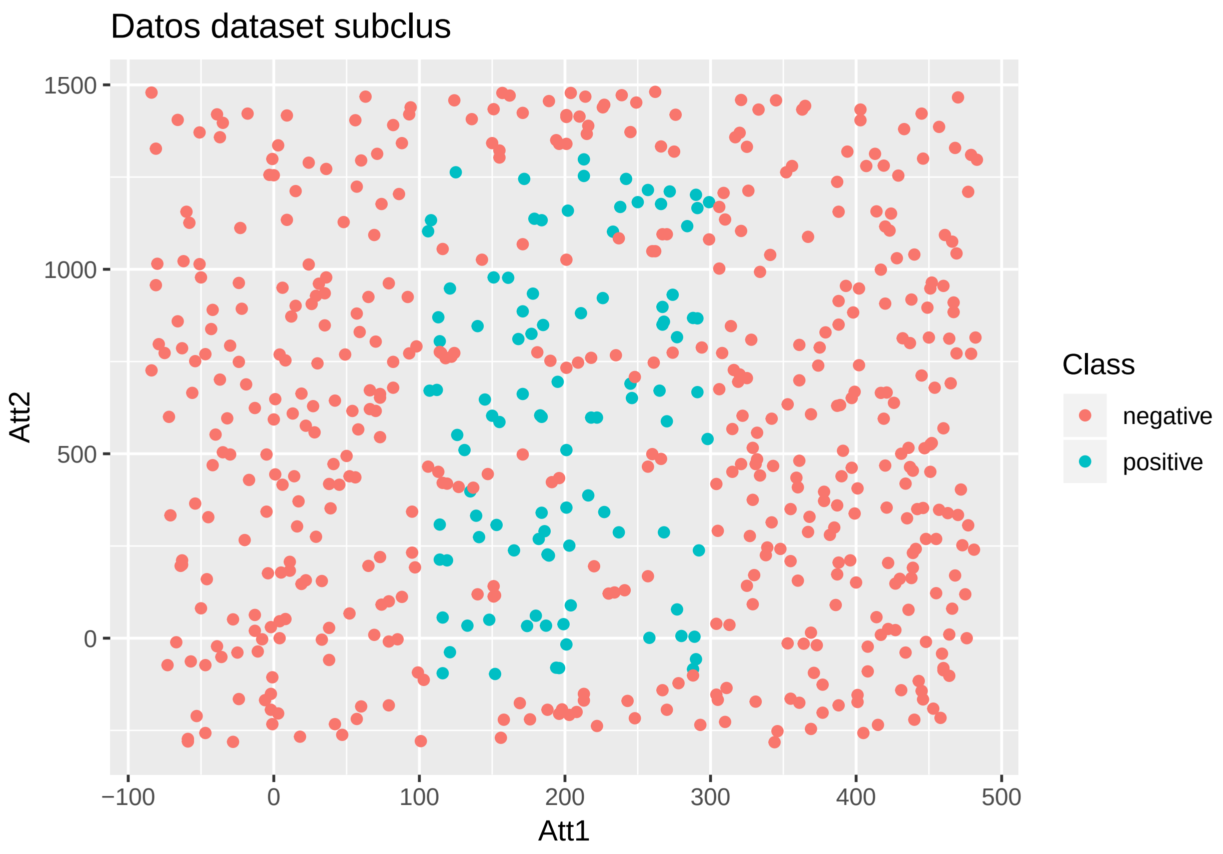
```r
learn_model <-function(dataset, ctrl,message){
  knn.fit <- train(Class ~ ., data = dataset, method = "knn",
                   trControl = ctrl, preProcess = c("center","scale"), metric="ROC",
                   tuneGrid = expand.grid(k = c(1,3,5,7,9,11)))
  knn.pred <- predict(knn.fit,newdata = dataset)
  #Get the confusion matrix to see accuracy value and other parameter values
  knn.cm <- confusionMatrix(knn.pred, dataset$Class,positive = "positive")
  knn.probs <- predict(knn.fit,newdata = dataset, type="prob")
  knn.roc <- roc(dataset$Class,knn.probs[,"positive"],color="green")
  return(knn.fit)
}

test_model <-function(dataset, knn.fit,message){
  knn.pred <- predict(knn.fit,newdata = dataset)
  #Get the confusion matrix to see accuracy value and other parameter values
  knn.cm <- confusionMatrix(knn.pred, dataset$Class,positive = "positive")
  print(knn.cm)
  knn.probs <- predict(knn.fit,newdata = dataset, type="prob")
  knn.roc <- roc(dataset$Class,knn.probs[,"positive"])
  #print(knn.roc)
  plot(knn.roc, type="S", print.thres= 0.5,main=c("ROC Test",message),col="blue")
  #print(paste0("AUC Test ",message,auc(knn.roc)))
  return(knn.cm)
}
```

Cargamos los conjuntos de datos que vamos a utilizar.

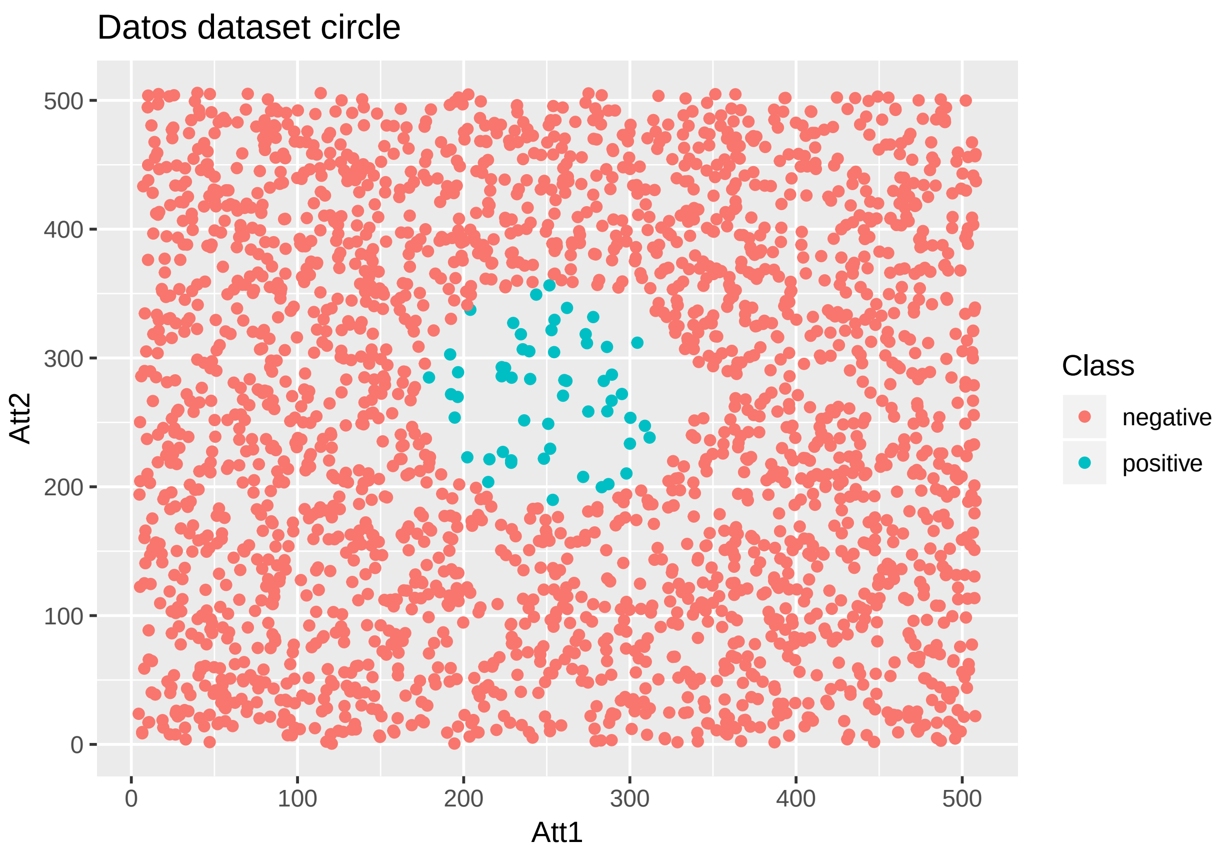
```
dataset <- read.table("subclus.txt", sep=",")
dataset2 <- read.table("circle.txt", sep=",")
colnames(dataset) <- c("Att1", "Att2", "Class")
colnames(dataset2) <- c("Att1", "Att2", "Class")
```

Visualizamos los conjuntos de datos.

```
library(ggplot2)
ggplot(dataset)+
  geom_point(aes(x=Att1,y=Att2,colour=Class))+
  ggtitle("Datos dataset subclus")
```



```
ggplot(dataset2)+
  geom_point(aes(x=Att1,y=Att2,colour=Class))+
  ggtitle("Datos dataset circle")
```



Como se puede ver, el primer dataset tiene pocos datos de una clase que de otra, además, los datos de la clase minoritaria están mezclados con algunos de los casos de la clase mayoritaria. Para el segundo dataset, los datos de la clase minoritaria se encuentran en el centro, pero no se mezclan con los datos de la clase mayoritaria.

Calculamos su ratio de imbalance.

```
imbalanceRatio(dataset)
```

```
## [1] 0.2
```

```
imbalanceRatio(dataset2)
```

```
## [1] 0.0235546
```

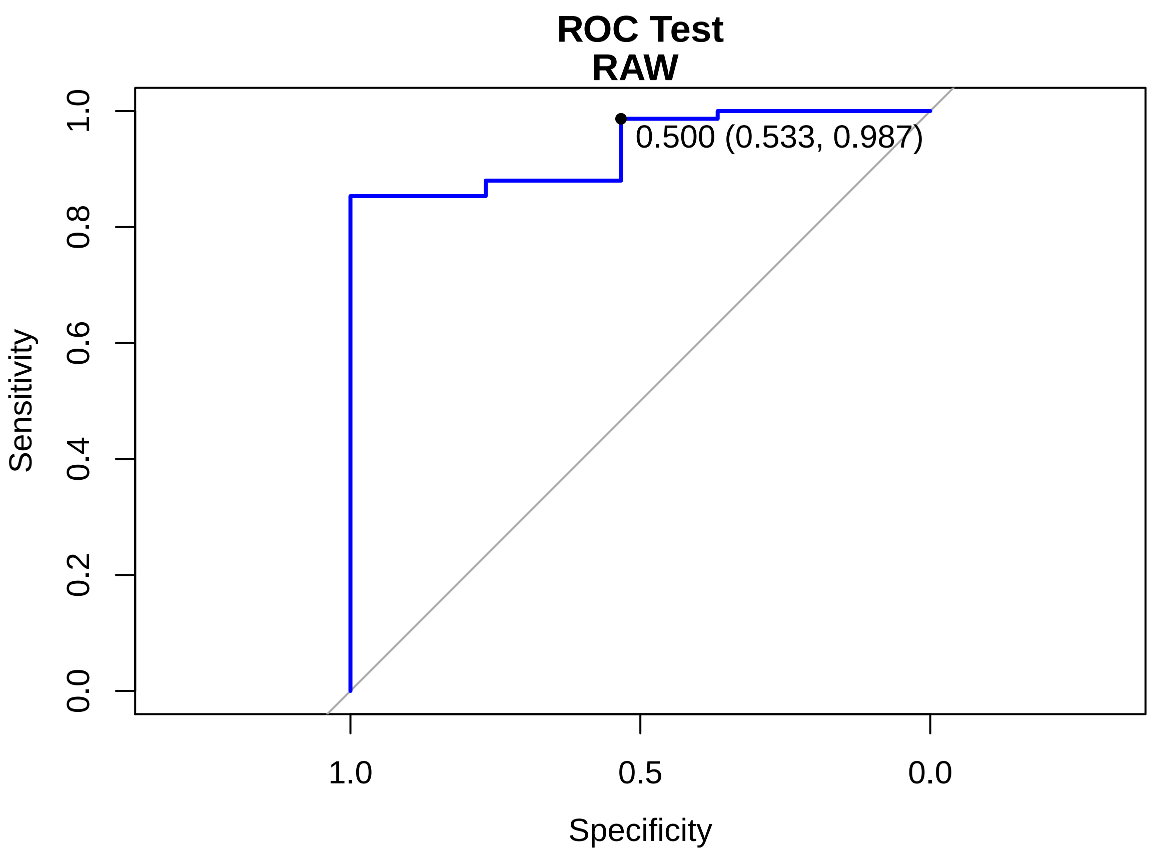
Como se puede ver, en ambos casos se puede ver que hay un gran desbalanceo entre las dos clases, siendo más exagerado para el dataset circle. Lo siguiente que vamos a hacer es crear particiones para cada uno de los dataset, y crear modelos y compararlos.

```
set.seed(42)
dataset$Class <- relevel(dataset$Class,"positive")
index <- createDataPartition(dataset$Class, p = 0.7, list = FALSE)
train_data <- dataset[index, ]
test_data  <- dataset[-index, ]

dataset2$Class <- relevel(dataset2$Class,"positive")
index2 <- createDataPartition(dataset2$Class, p = 0.7, list = FALSE)
train_data2 <- dataset2[index2, ]
test_data2  <- dataset2[-index2, ]
```

Modelos con sin modificaciones a los datos.

```r
ctrl <- trainControl(method="repeatedcv",number=5,repeats = 3,
                     classProbs=TRUE,summaryFunction = twoClassSummary)
model.raw <- learn_model(train_data,ctrl,"RAW ")
#plot(model,main="Grid Search RAW")
#print(model.raw)
cm.original <- test_model(test_data,model.raw,"RAW ")
```

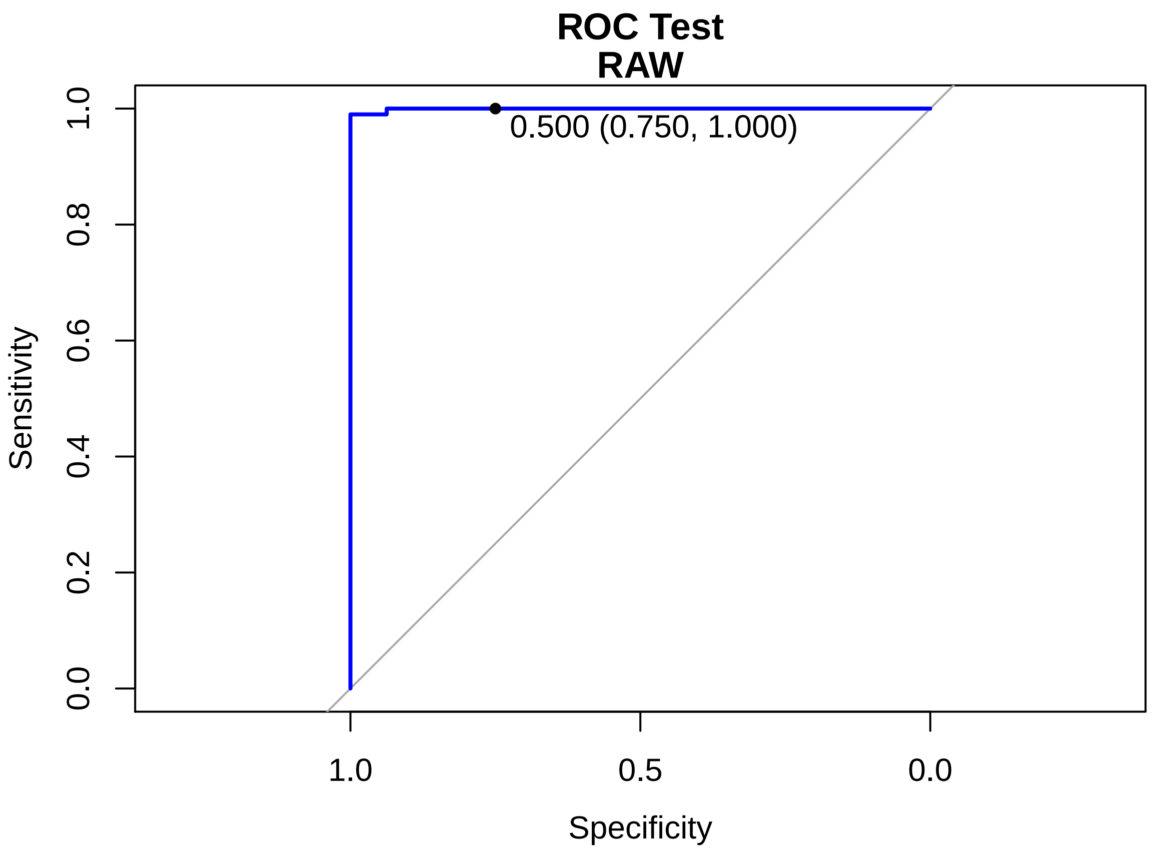
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction positive negative
##   positive       16        2
##   negative       14      148
##
##                Accuracy : 0.9111
##                  95% CI : (0.8597, 0.9483)
##     No Information Rate : 0.8333
##     P-Value [Acc > NIR] : 0.001979
##
##                   Kappa : 0.619
##  Mcnemar's Test P-Value : 0.005960
##
##             Sensitivity : 0.53333
##             Specificity : 0.98667
##          Pos Pred Value : 0.88889
##          Neg Pred Value : 0.91358
##              Prevalence : 0.16667
##          Detection Rate : 0.08889
##    Detection Prevalence : 0.10000
##       Balanced Accuracy : 0.76000
##
##        'Positive' Class : positive
##
```



```
model.raw.2 = learn_model(train_data2,ctrl,"RAW")
cm.original.2 = test_model(test_data2,model.raw.2,"RAW")
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction positive negative
##   positive       12        0
##   negative        4      700
##
##                Accuracy : 0.9944
##                  95% CI : (0.9858, 0.9985)
##     No Information Rate : 0.9777
##     P-Value [Acc > NIR] : 0.0003607
##
##                   Kappa : 0.8544
##  Mcnemar's Test P-Value : 0.1336144
##
##             Sensitivity : 0.75000
##             Specificity : 1.00000
##          Pos Pred Value : 1.00000
##          Neg Pred Value : 0.99432
##              Prevalence : 0.02235
##          Detection Rate : 0.01676
##    Detection Prevalence : 0.01676
##       Balanced Accuracy : 0.87500
##
```

```
##        'Positive' Class : positive
##
```

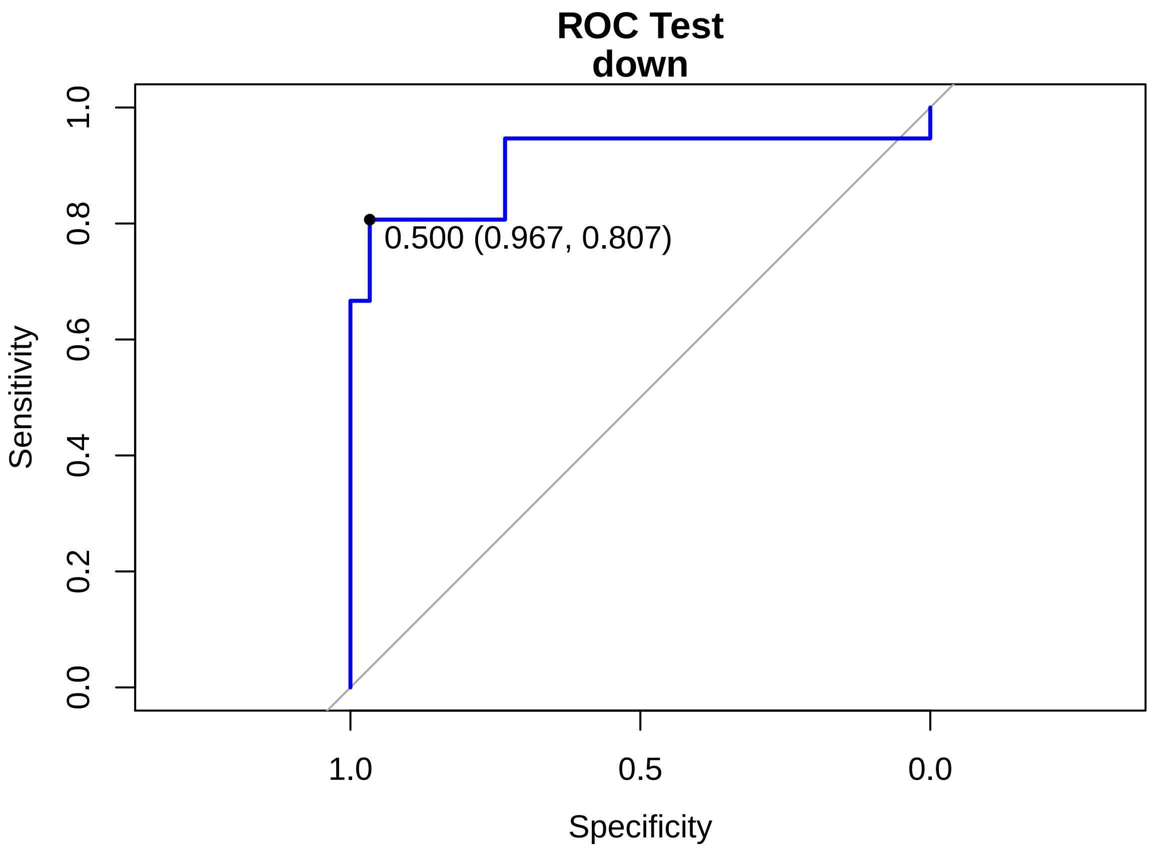


Modelos con undersampling.

```
ctrl <- trainControl(method="repeatedcv",number=5,repeats = 3,
                     classProbs=TRUE,summaryFunction = twoClassSummary,sampling = "down")

model.us <- learn_model(train_data,ctrl,"down ")
#plot(model,main="Grid Search RAW")
#print(model.raw)
cm.under <- test_model(test_data,model.us,"down")
```

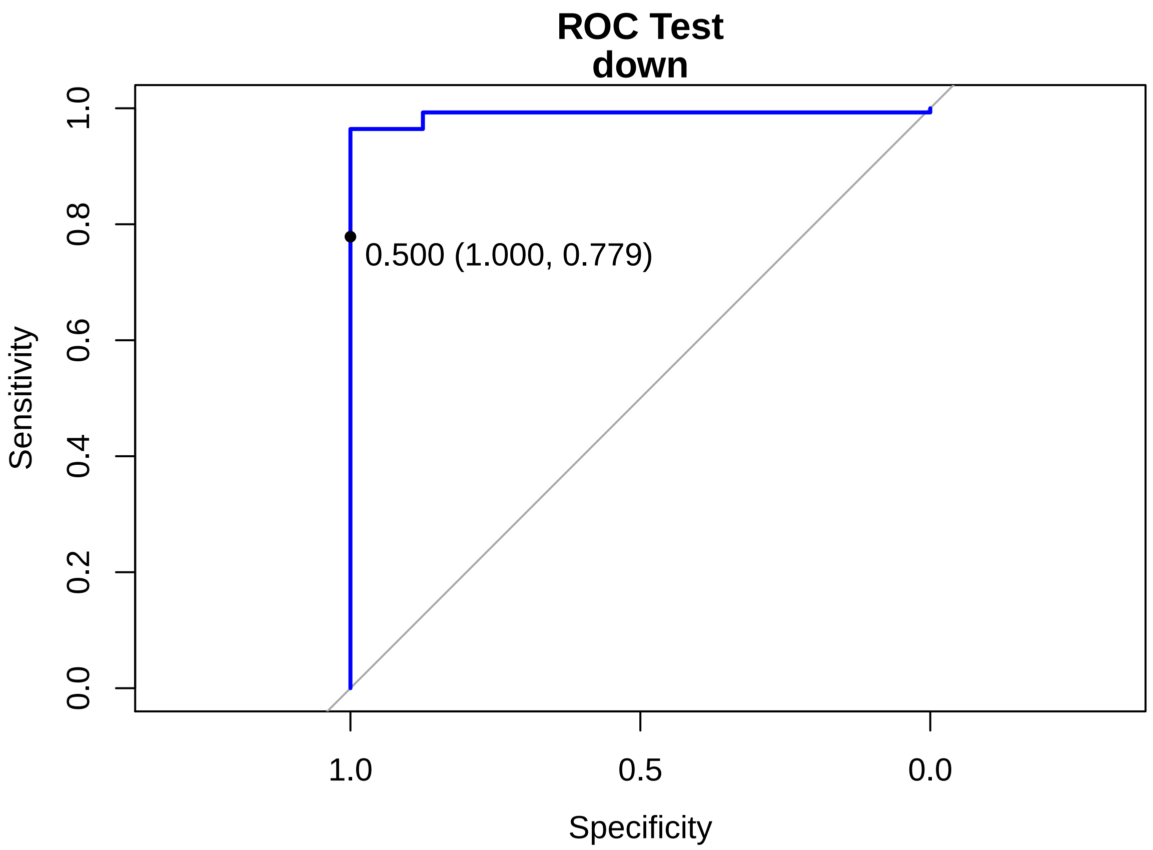
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction positive negative
##   positive       29       29
##   negative        1      121
##
##                Accuracy : 0.8333
##                  95% CI : (0.7707, 0.8846)
##     No Information Rate : 0.8333
##     P-Value [Acc > NIR] : 0.5486
##
##                   Kappa : 0.5631
##  Mcnemar's Test P-Value : 8.244e-07
##
```

```
##               Sensitivity : 0.9667
##               Specificity : 0.8067
##            Pos Pred Value : 0.5000
##            Neg Pred Value : 0.9918
##                Prevalence : 0.1667
##            Detection Rate : 0.1611
##      Detection Prevalence : 0.3222
##         Balanced Accuracy : 0.8867
##
##          'Positive' Class : positive
##
```



```
model.us.2 <- learn_model(train_data2,ctrl,"down ")
cm.under.2 <- test_model(test_data2,model.us.2,"down")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction positive negative
##   positive       16      155
##   negative        0      545
##
##                Accuracy : 0.7835
##                  95% CI : (0.7515, 0.8132)
##     No Information Rate : 0.9777
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1358
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 1.00000
##             Specificity : 0.77857
##          Pos Pred Value : 0.09357
##          Neg Pred Value : 1.00000
##              Prevalence : 0.02235
##          Detection Rate : 0.02235
##    Detection Prevalence : 0.23883
##       Balanced Accuracy : 0.88929
##
##        'Positive' Class : positive
##
```

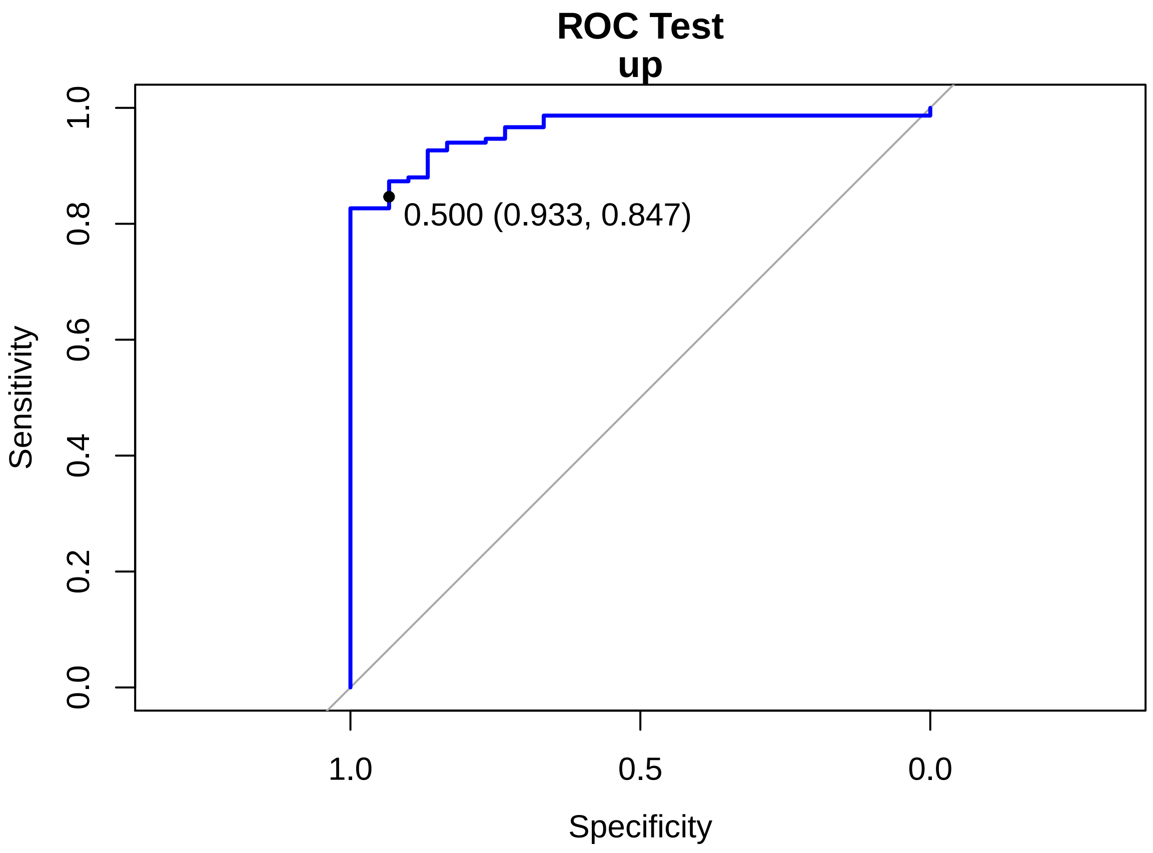


Modelos con oversampling.

```r
ctrl <- trainControl(method="repeatedcv",number=5,repeats = 3,
                     classProbs=TRUE,summaryFunction = twoClassSummary,sampling = "up")

model.os <- learn_model(train_data,ctrl,"up ")
cm.over <- test_model(test_data,model.os,"up")
```

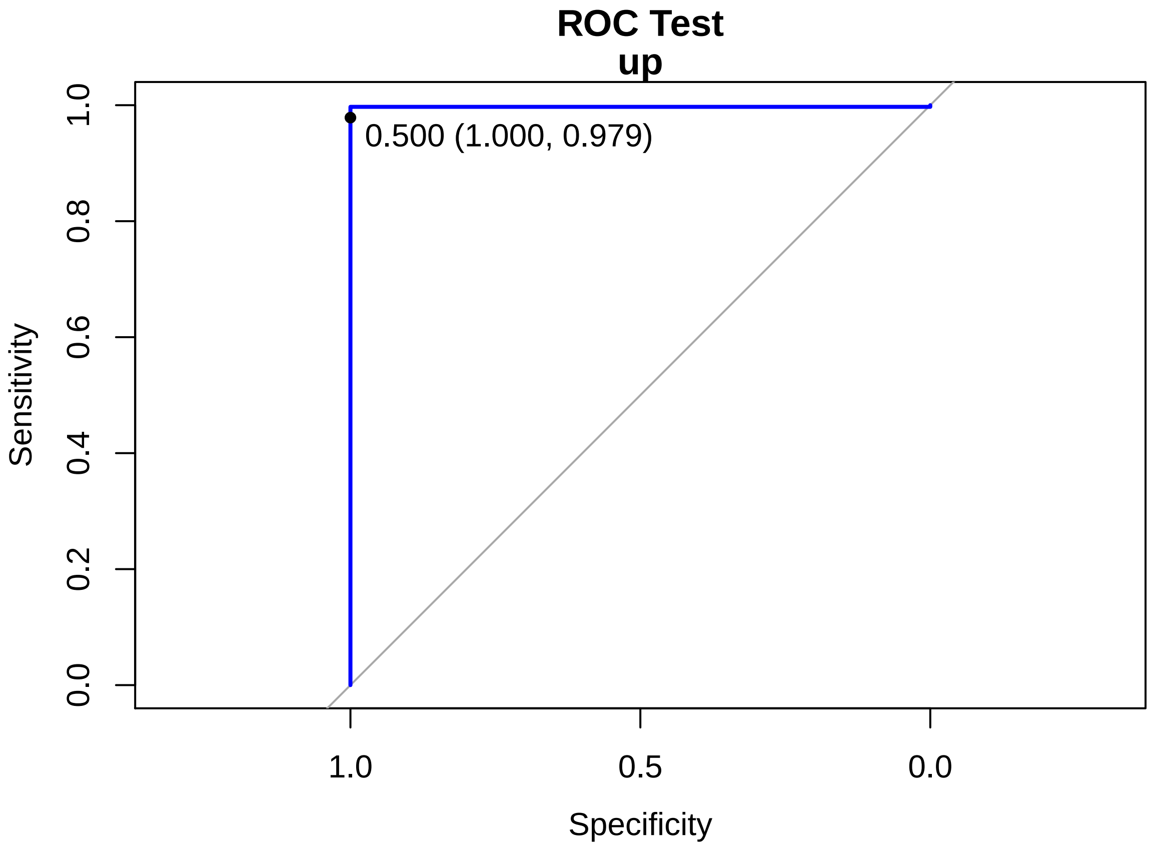
```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction positive negative
##   positive       28       23
##   negative        2      127
```

```
##
##                Accuracy : 0.8611
##                  95% CI : (0.8018, 0.9081)
##     No Information Rate : 0.8333
##     P-Value [Acc > NIR] : 0.1851
##
##                   Kappa : 0.6094
##  Mcnemar's Test P-Value : 6.334e-05
##
##             Sensitivity : 0.9333
##             Specificity : 0.8467
##          Pos Pred Value : 0.5490
##          Neg Pred Value : 0.9845
##              Prevalence : 0.1667
##          Detection Rate : 0.1556
##    Detection Prevalence : 0.2833
##       Balanced Accuracy : 0.8900
##
##        'Positive' Class : positive
##
```



```
model.os.2 <- learn_model(train_data2,ctrl,"up ")
cm.over.2 <- test_model(test_data2,model.os.2,"up")
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction positive negative
##   positive       16       15
##   negative        0      685
##
##                Accuracy : 0.9791
##                  95% CI : (0.9657, 0.9882)
##     No Information Rate : 0.9777
##     P-Value [Acc > NIR] : 0.4656219
##
##                   Kappa : 0.6712
##  Mcnemar's Test P-Value : 0.0003006
##
##             Sensitivity : 1.00000
##             Specificity : 0.97857
##          Pos Pred Value : 0.51613
##          Neg Pred Value : 1.00000
##              Prevalence : 0.02235
##          Detection Rate : 0.02235
##    Detection Prevalence : 0.04330
##       Balanced Accuracy : 0.98929
##
##        'Positive' Class : positive
##
```



Modelos con SMOTE

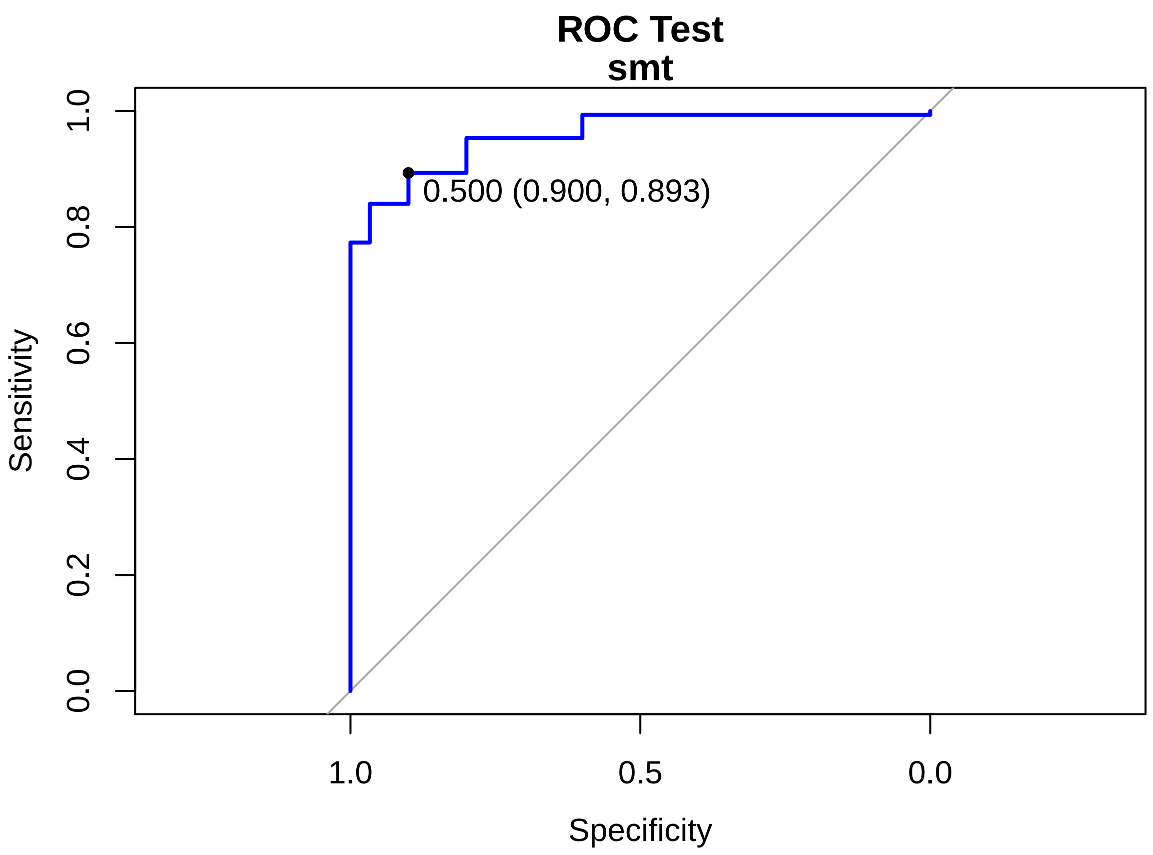
```
ctrl <- trainControl(method="repeatedcv",number=5,repeats = 3,
                     classProbs=TRUE,summaryFunction = twoClassSummary,sampling = "smote")
```

```
model.smt <- learn_model(train_data,ctrl,"smt")
```

```
## Loading required package: grid
```

```
cm.smote<- test_model(test_data,model.smt,"smt")
```

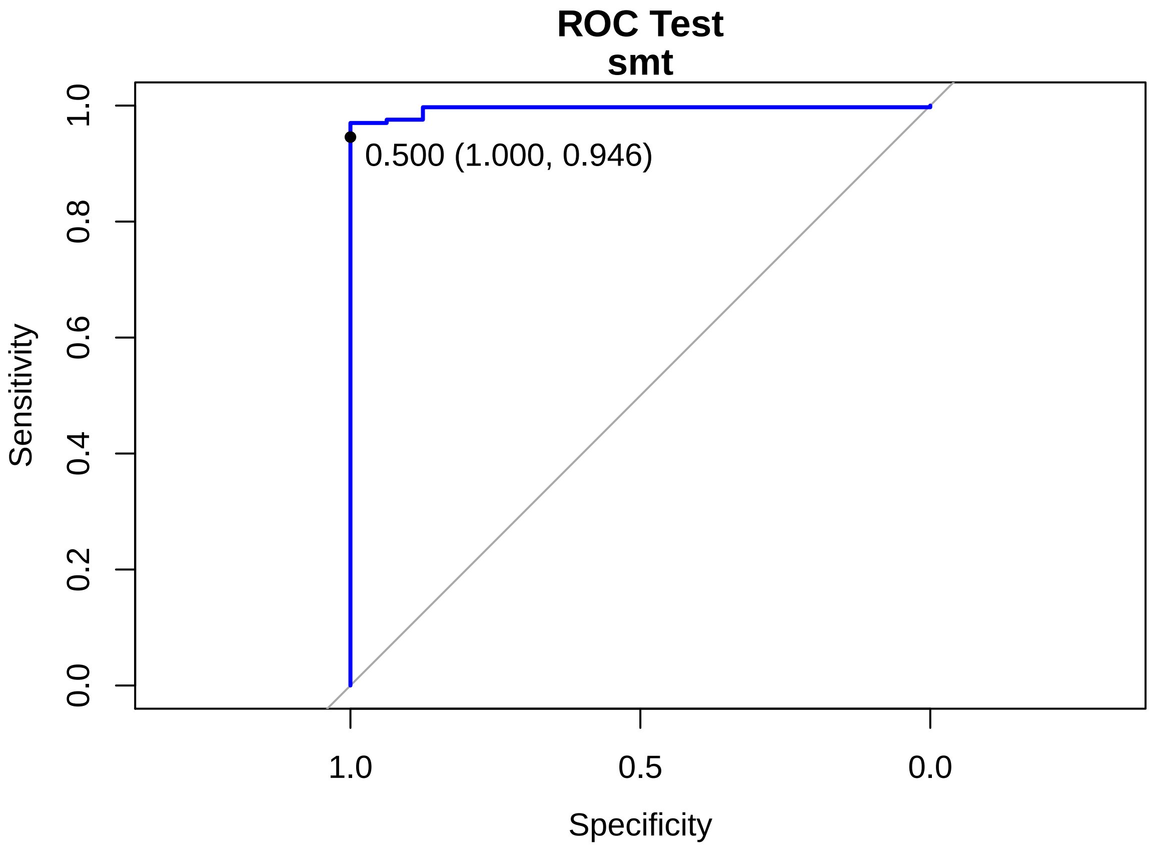
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction positive negative
##   positive       27       16
##   negative        3      134
##
##                Accuracy : 0.8944
##                  95% CI : (0.8401, 0.9352)
##     No Information Rate : 0.8333
##     P-Value [Acc > NIR] : 0.014158
##
##                   Kappa : 0.6761
##  Mcnemar's Test P-Value : 0.005905
##
##             Sensitivity : 0.9000
##             Specificity : 0.8933
##          Pos Pred Value : 0.6279
##          Neg Pred Value : 0.9781
##              Prevalence : 0.1667
##          Detection Rate : 0.1500
##    Detection Prevalence : 0.2389
##       Balanced Accuracy : 0.8967
##
##        'Positive' Class : positive
##
```



```r
model.smt.2 <- learn_model(train_data2,ctrl,"smt")
cm.smote.2<- test_model(test_data2,model.smt.2,"smt")
```

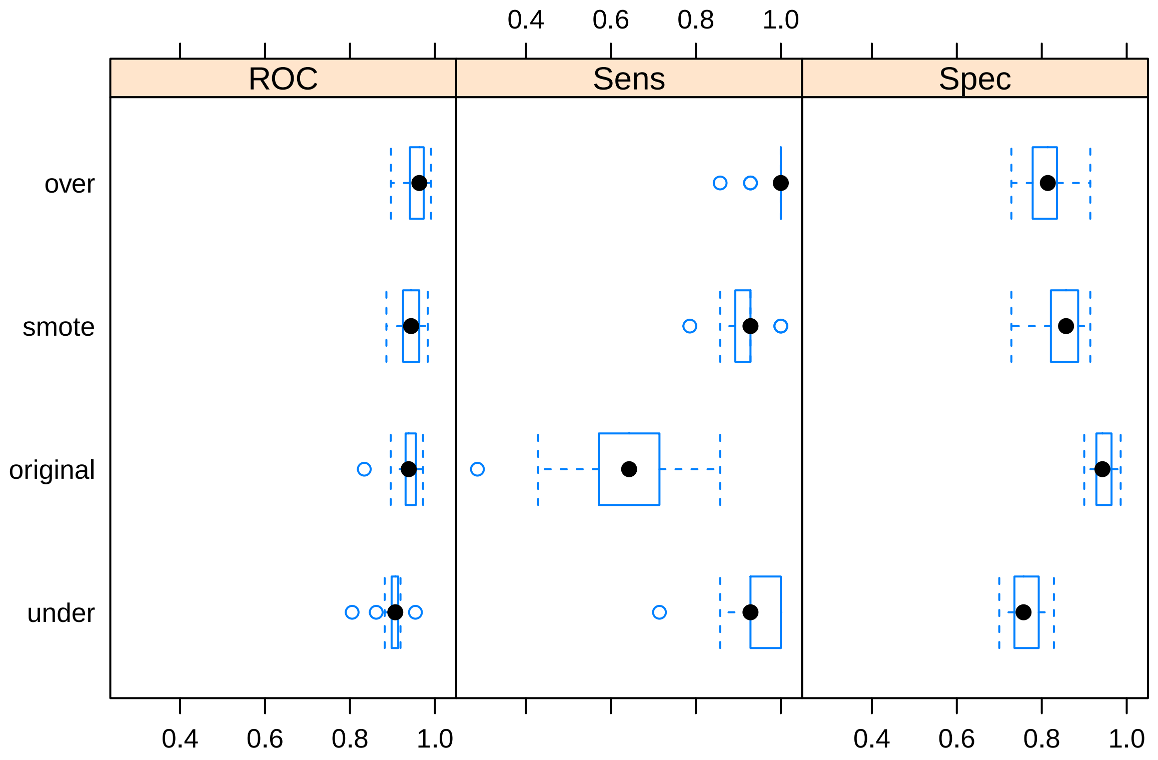
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction positive negative
##   positive       16       38
##   negative        0      662
##
##                Accuracy : 0.9469
##                  95% CI : (0.9279, 0.9622)
##     No Information Rate : 0.9777
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4378
##  Mcnemar's Test P-Value : 1.947e-09
##
##             Sensitivity : 1.00000
##             Specificity : 0.94571
##          Pos Pred Value : 0.29630
##          Neg Pred Value : 1.00000
##              Prevalence : 0.02235
##          Detection Rate : 0.02235
##    Detection Prevalence : 0.07542
##       Balanced Accuracy : 0.97286
##
```

```
##        'Positive' Class : positive
##
```

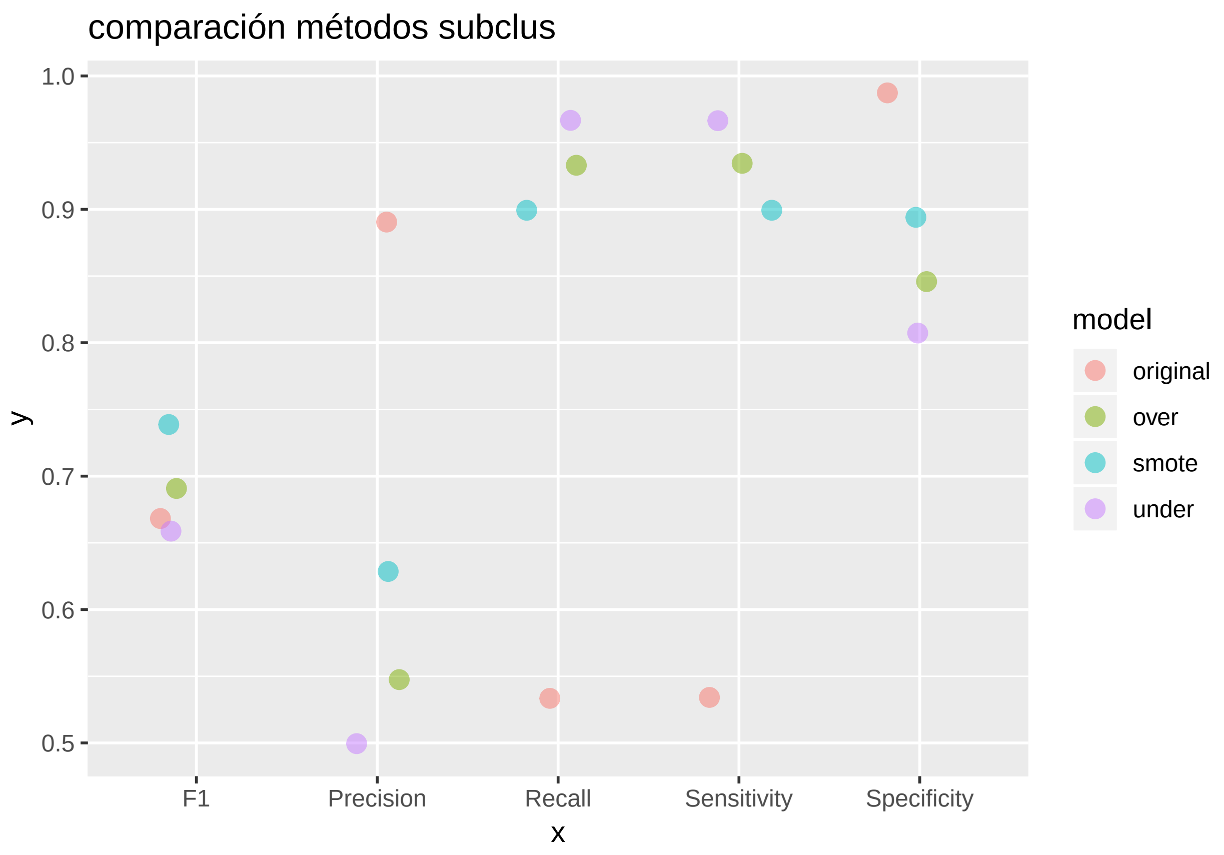


Unimos los datos en una lista y comparamos las diferentes medidas entre ellos.

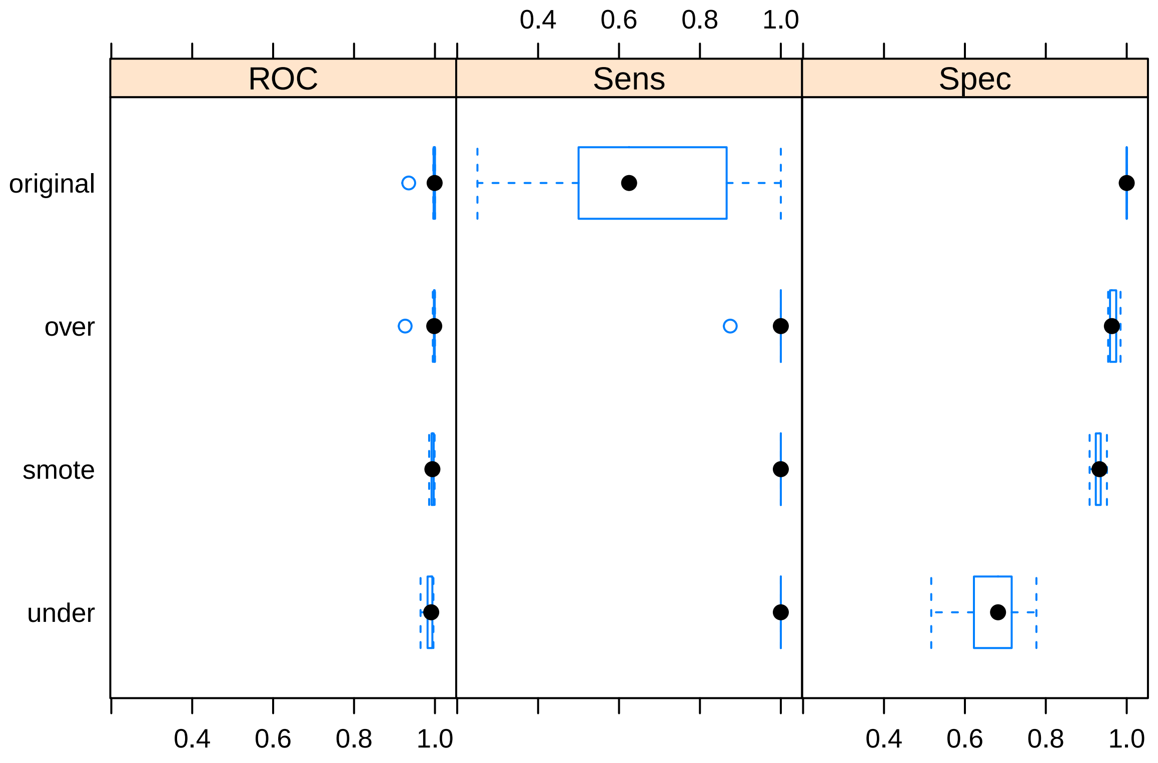
```
models <- list(original = model.raw,
               under = model.us,
               over = model.os,
               smote = model.smt)

resampling <- resamples(models)
bwplot(resampling)
```



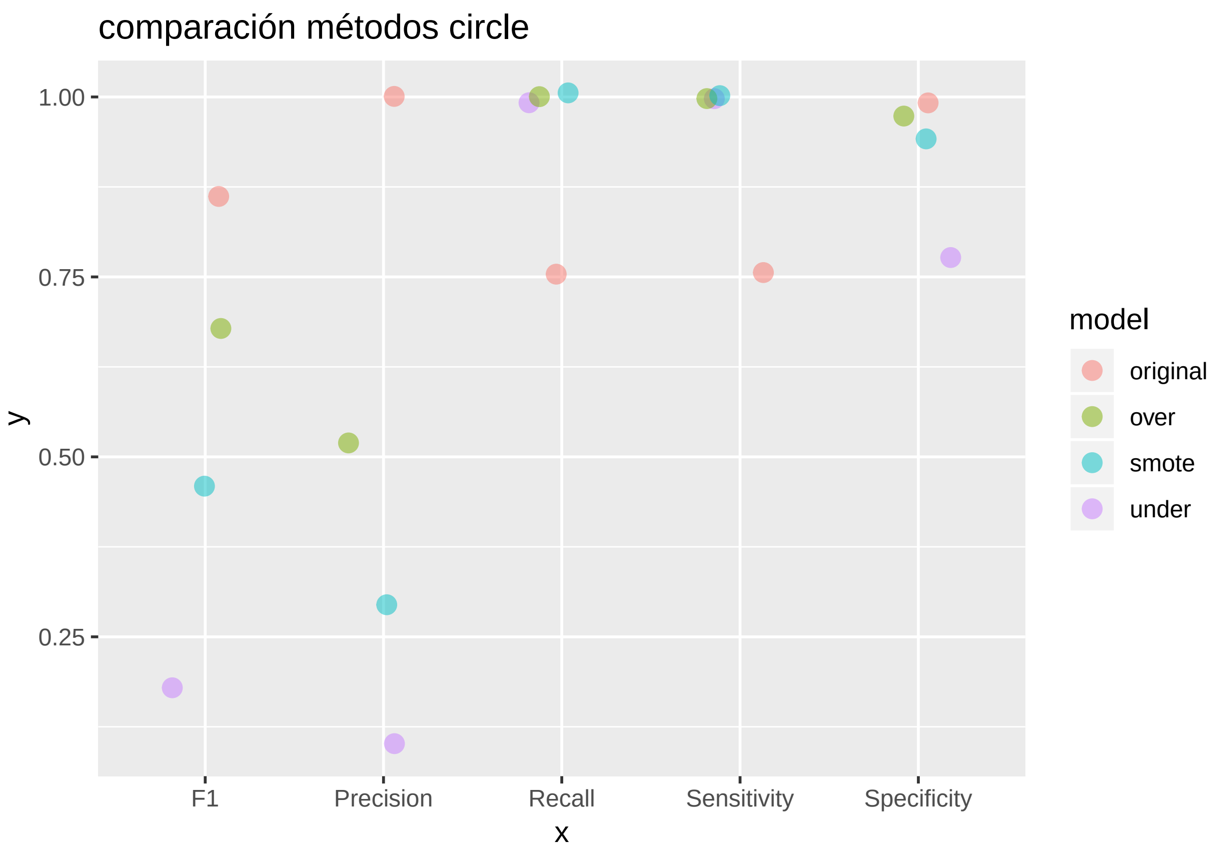
```
comparison <- data.frame(model = names(models),
                         Sensitivity = rep(NA, length(models)),
                         Specificity = rep(NA, length(models)),
                         Precision = rep(NA, length(models)),
                         Recall = rep(NA, length(models)),
                         F1 = rep(NA, length(models)))

for (name in names(models)) {
  cm_model <- get(paste0("cm.", name))

  comparison[comparison$model == name, ] <- filter(comparison, model == name) %>%
    mutate(Sensitivity = cm_model$byClass["Sensitivity"],
           Specificity = cm_model$byClass["Specificity"],
           Precision = cm_model$byClass["Precision"],
           Recall = cm_model$byClass["Recall"],
           F1 = cm_model$byClass["F1"])
}

comparison %>%
  gather(x, y, Sensitivity:F1) %>%
  ggplot(aes(x = x, y = y, color = model)) +
  geom_jitter(width = 0.2, alpha = 0.5, size = 3)+
  ggtitle("comparación métodos subclus")
```



```
models <- list(original = model.raw.2,
               under = model.us.2,
               over = model.os.2,
               smote = model.smt.2)

resampling <- resamples(models)
bwplot(resampling)
```

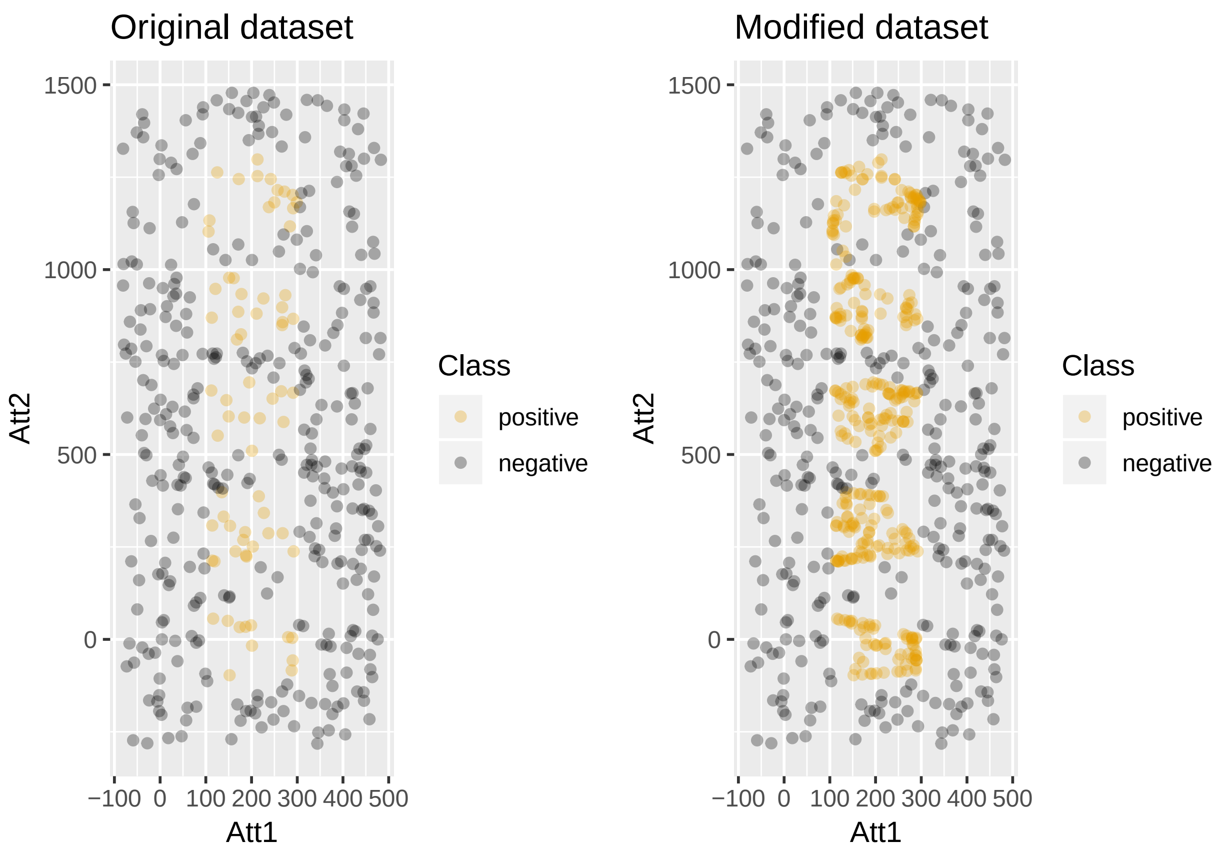


```
comparison <- data.frame(model = names(models),
                         Sensitivity = rep(NA, length(models)),
                         Specificity = rep(NA, length(models)),
                         Precision = rep(NA, length(models)),
                         Recall = rep(NA, length(models)),
                         F1 = rep(NA, length(models)))

for (name in names(models)) {
  cm_model <- get(paste0("cm.", name,".2"))

  comparison[comparison$model == name, ] <- filter(comparison, model == name) %>%
    mutate(Sensitivity = cm_model$byClass["Sensitivity"],
           Specificity = cm_model$byClass["Specificity"],
           Precision = cm_model$byClass["Precision"],
           Recall = cm_model$byClass["Recall"],
           F1 = cm_model$byClass["F1"])
}

comparison %>%
  gather(x, y, Sensitivity:F1) %>%
  ggplot(aes(x = x, y = y, color = model)) +
  geom_jitter(width = 0.2, alpha = 0.5, size = 3)+
  ggtitle("comparación métodos circle")
```

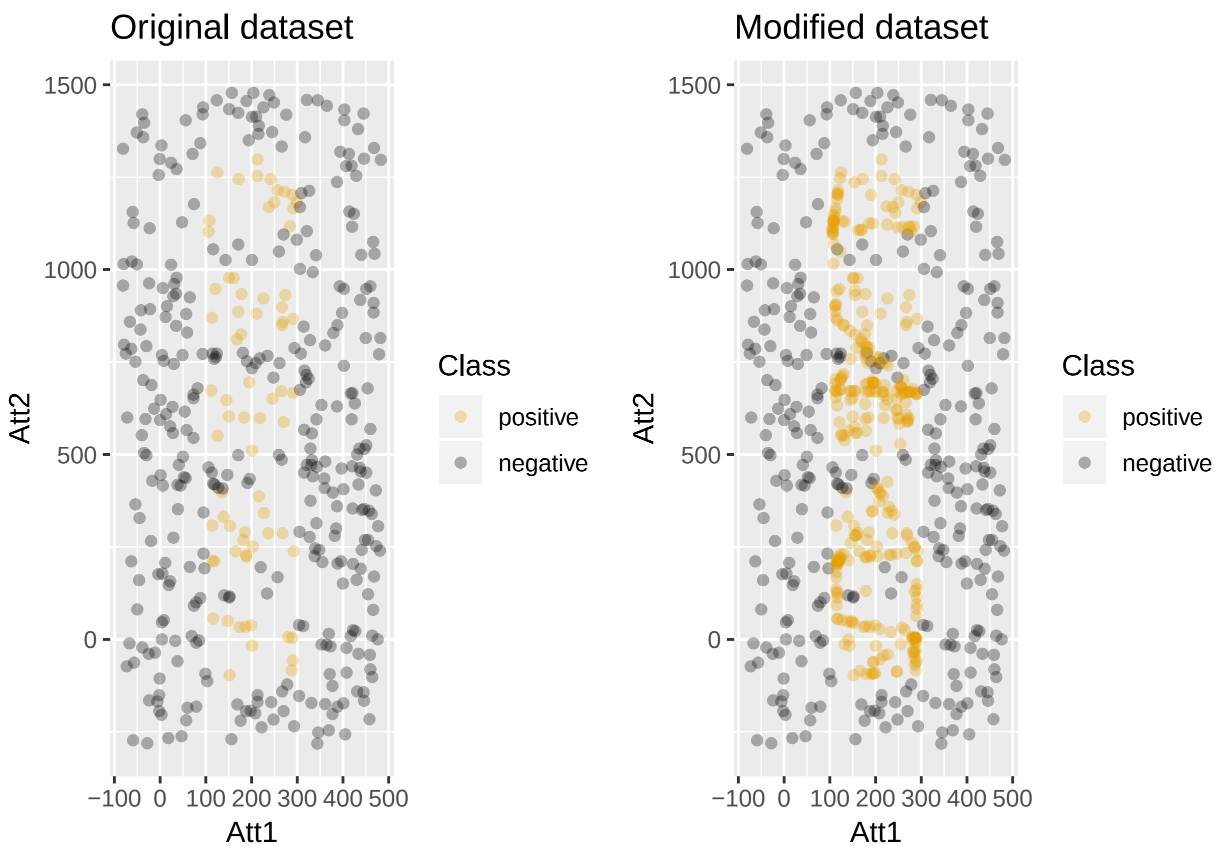


Ahora, probaremos dos modelos nuevos de oversampling diferentes de SMOTE y compararemos los resultados.

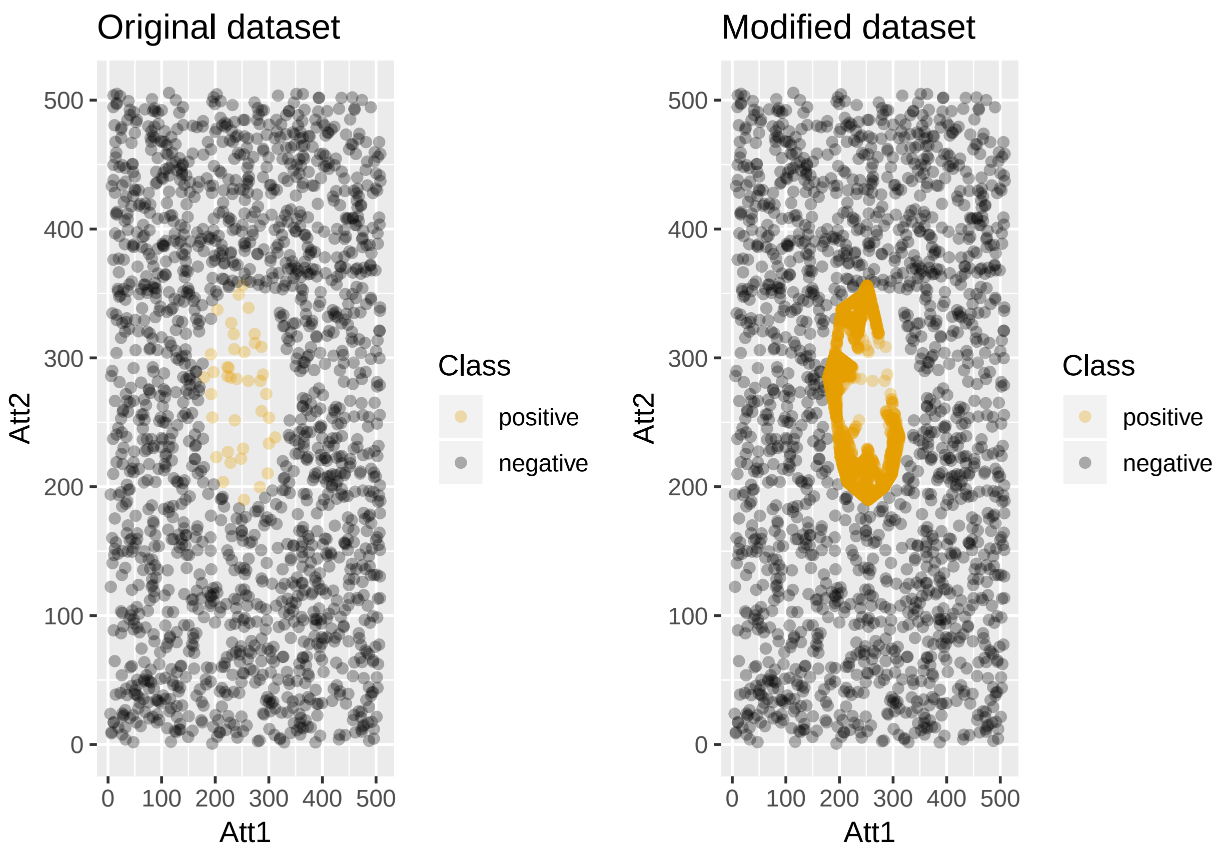
```
new_train = oversample(train_data,method="ADASYN",classAttr = "Class")
plotComparison(train_data,new_train,cols = 2,attrs = names(dataset)[1:2],classAttr = "Class")
```



```
new_train_blsmote = oversample(train_data,method="BLSMOTE",ratio=0.9,classAttr = "Class")

## [1] "Borderline-SMOTE done"

plotComparison(train_data,new_train_blsmote,cols = 2,attrs = names(dataset)[1:2],classAttr = "Class")
```



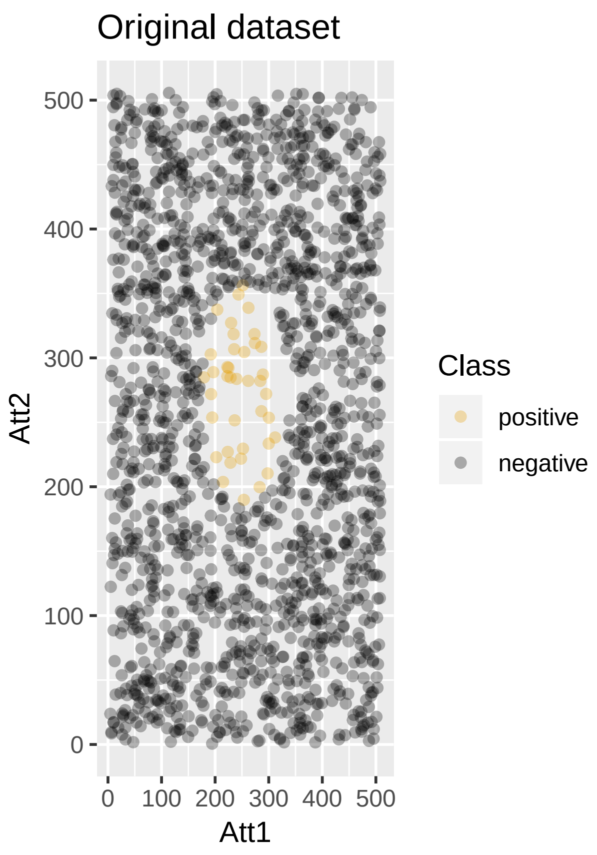
```
new_train2 = oversample(train_data2,method="ADASYN",classAttr = "Class")
plotComparison(train_data2,new_train2,cols = 2,attrs = names(dataset2)[1:2],classAttr = "Class")
```

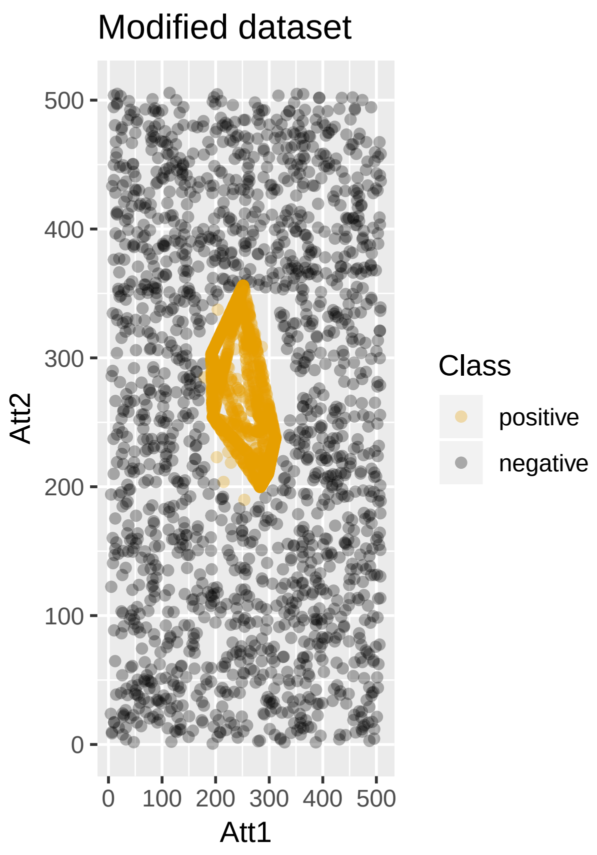


```
new_train_blsmote2 = oversample(train_data2,method="BLSMOTE",ratio=0.9,classAttr = "Class")
```

```
## [1] "Borderline-SMOTE done"
```

```
plotComparison(train_data2,new_train_blsmote2,cols = 2,attrs = names(dataset2)[1:2],classAttr = "Class"
```



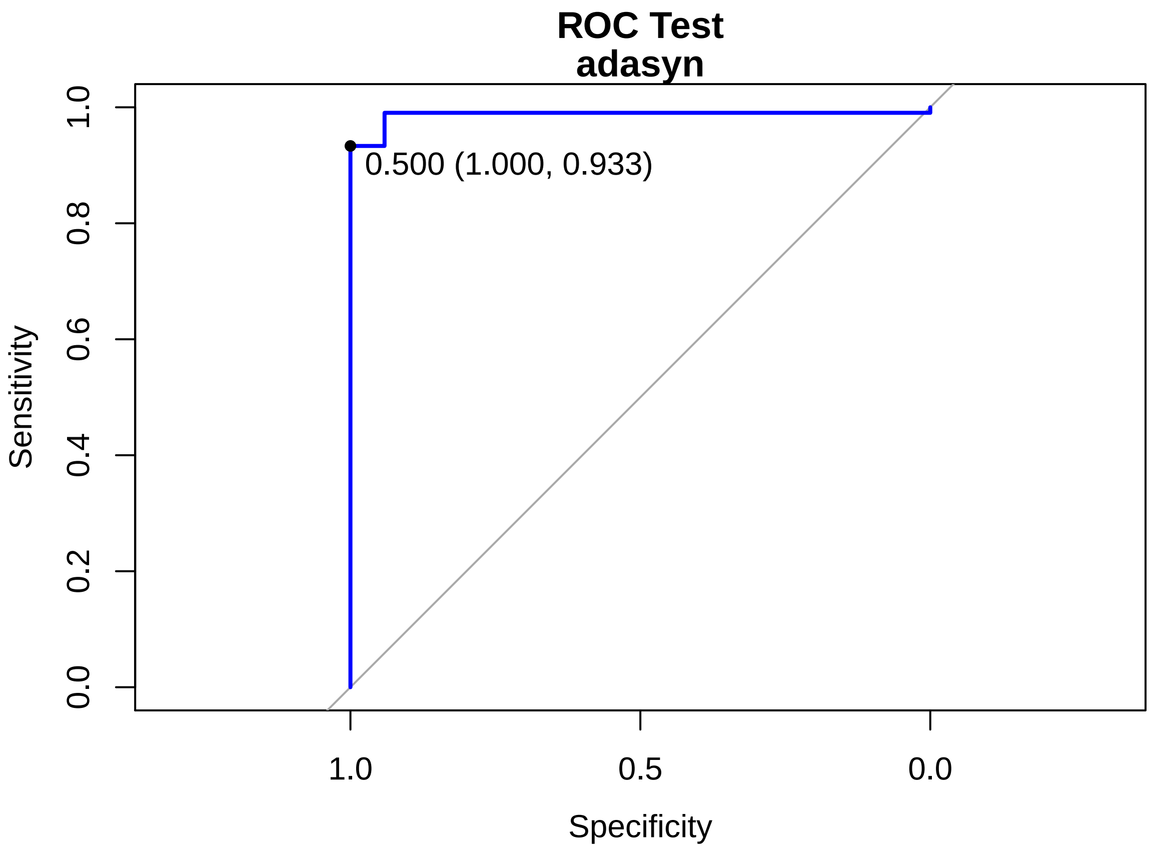


Primero crearemos particiones para los dataset con oversampling, después creamos modelos para ambos datasets, y añadimos los resultados a las gráficas que hemos generado anteriormente.

```
new_train$Class <- relevel(new_train$Class,"positive")
index <- createDataPartition(new_train$Class, p = 0.7, list = FALSE)
train_data <- new_train[index, ]
test_data  <- new_train[-index, ]

new_train2$Class <- relevel(new_train2$Class,"positive")
index2 <- createDataPartition(new_train2$Class, p = 0.7, list = FALSE)
train_data2 <- new_train2[index2, ]
test_data2  <- new_train2[-index2, ]
```

```
ctrl <- trainControl(method="repeatedcv",number=5,repeats = 3,
                     classProbs=TRUE,summaryFunction = twoClassSummary)

model.adasyn <- learn_model(train_data,ctrl,"adasyn")
cm.adasyn<- test_model(test_data,model.adasyn,"adasyn")
```

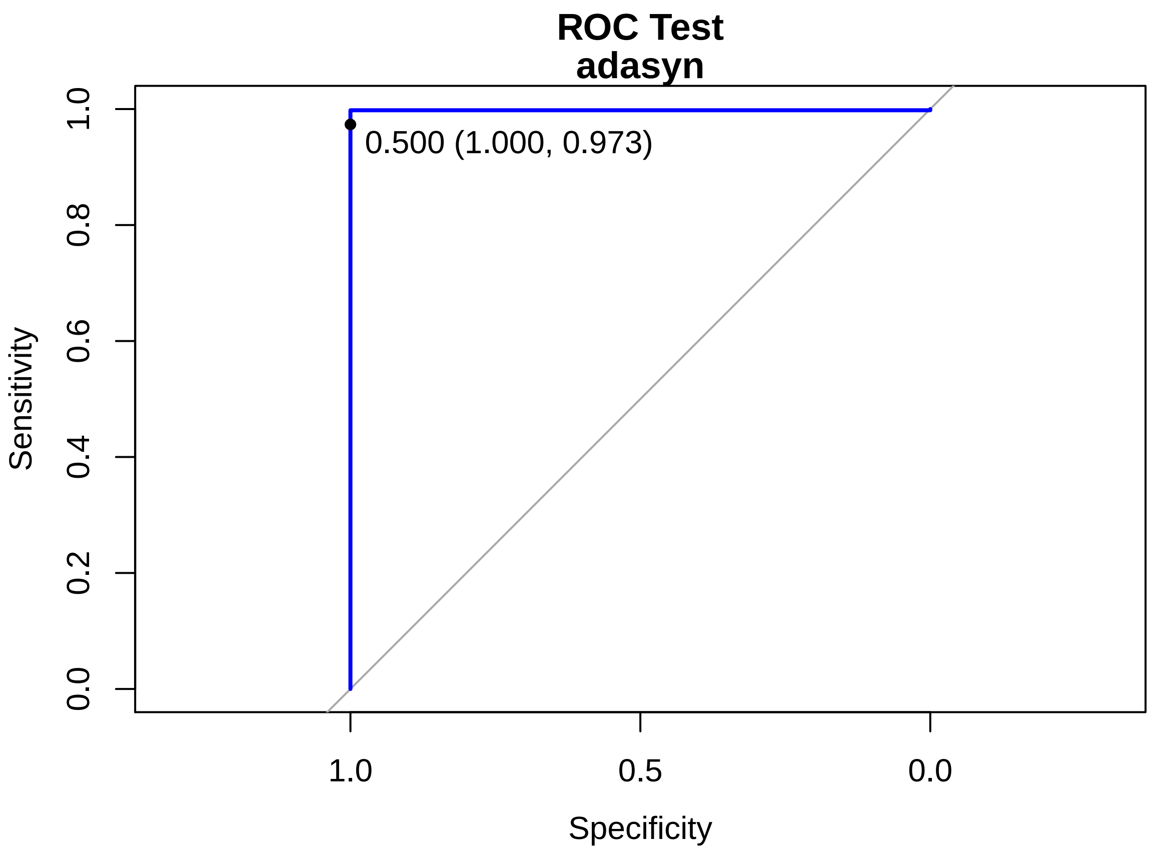
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction positive negative
##   positive      102        7
##   negative        0       98
##
##                Accuracy : 0.9662
##                  95% CI : (0.9316, 0.9863)
##     No Information Rate : 0.5072
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9324
##  Mcnemar's Test P-Value : 0.02334
##
##             Sensitivity : 1.0000
##             Specificity : 0.9333
##          Pos Pred Value : 0.9358
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4928
##          Detection Rate : 0.4928
##    Detection Prevalence : 0.5266
##       Balanced Accuracy : 0.9667
##
##        'Positive' Class : positive
##
```



```
model.adasyn.2 <- learn_model(train_data2,ctrl,"adasyn")
cm.adasyn.2<- test_model(test_data2,model.adasyn.2,"adasyn")
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction positive negative
##   positive      489       13
##   negative        0      477
##
```

```
##                Accuracy : 0.9867
##                  95% CI : (0.9774, 0.9929)
##     No Information Rate : 0.5005
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9734
##  Mcnemar's Test P-Value : 0.0008741
##
##             Sensitivity : 1.0000
##             Specificity : 0.9735
##          Pos Pred Value : 0.9741
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4995
##          Detection Rate : 0.4995
##    Detection Prevalence : 0.5128
##       Balanced Accuracy : 0.9867
##
##        'Positive' Class : positive
##
```



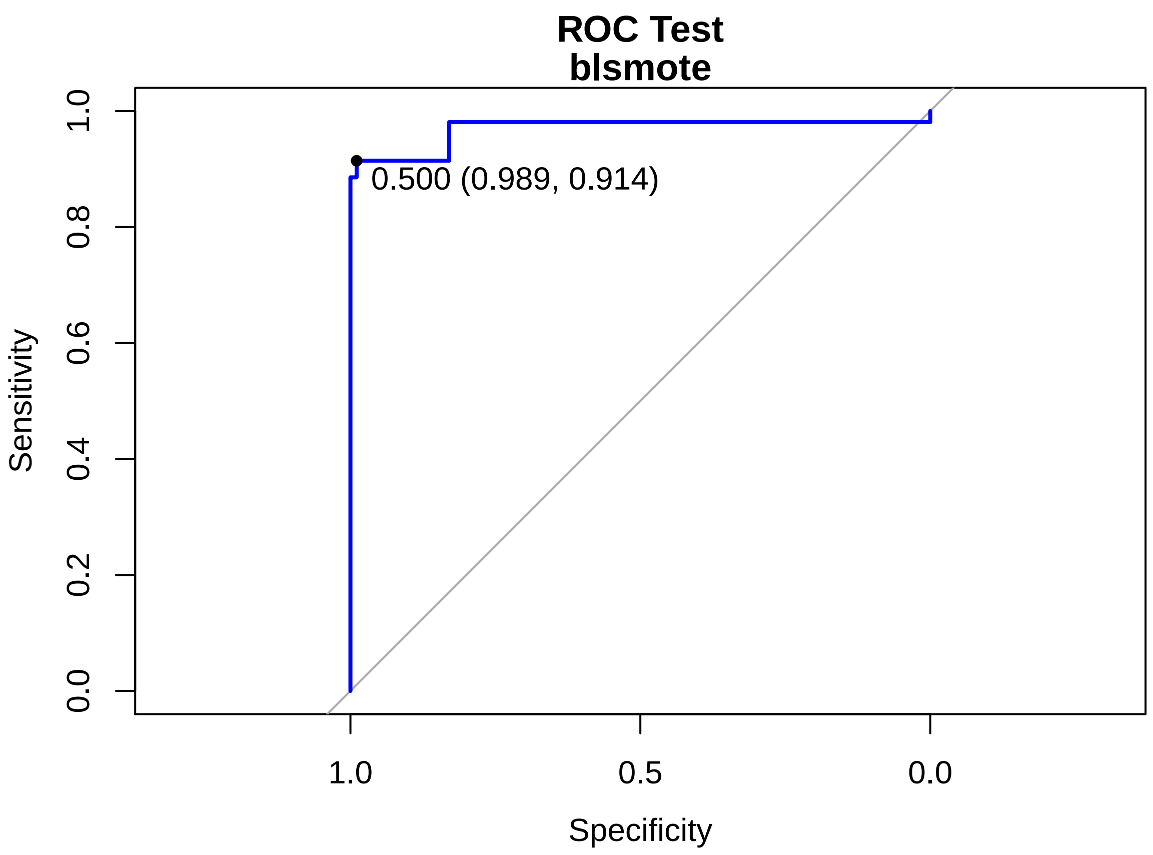
Repetimos el mismo proceso para los dataset generados con *BLSMOTE*.

```
new_train_blsmote$Class <- relevel(new_train_blsmote$Class,"positive")
index <- createDataPartition(new_train_blsmote$Class, p = 0.7, list = FALSE)
train_data <- new_train_blsmote[index, ]
test_data  <- new_train_blsmote[-index, ]

new_train_blsmote2$Class <- relevel(new_train_blsmote2$Class,"positive")
```

```r
index2 <- createDataPartition(new_train_blsmote2$Class, p = 0.7, list = FALSE)
train_data2 <- new_train_blsmote2[index2, ]
test_data2  <- new_train_blsmote2[-index2, ]
```

```r
ctrl <- trainControl(method="repeatedcv",number=5,repeats = 3,
                     classProbs=TRUE,summaryFunction = twoClassSummary)

model.blsmote <- learn_model(train_data,ctrl,"blsmote")
cm.blsmote <- test_model(test_data,model.blsmote,"blsmote")
```

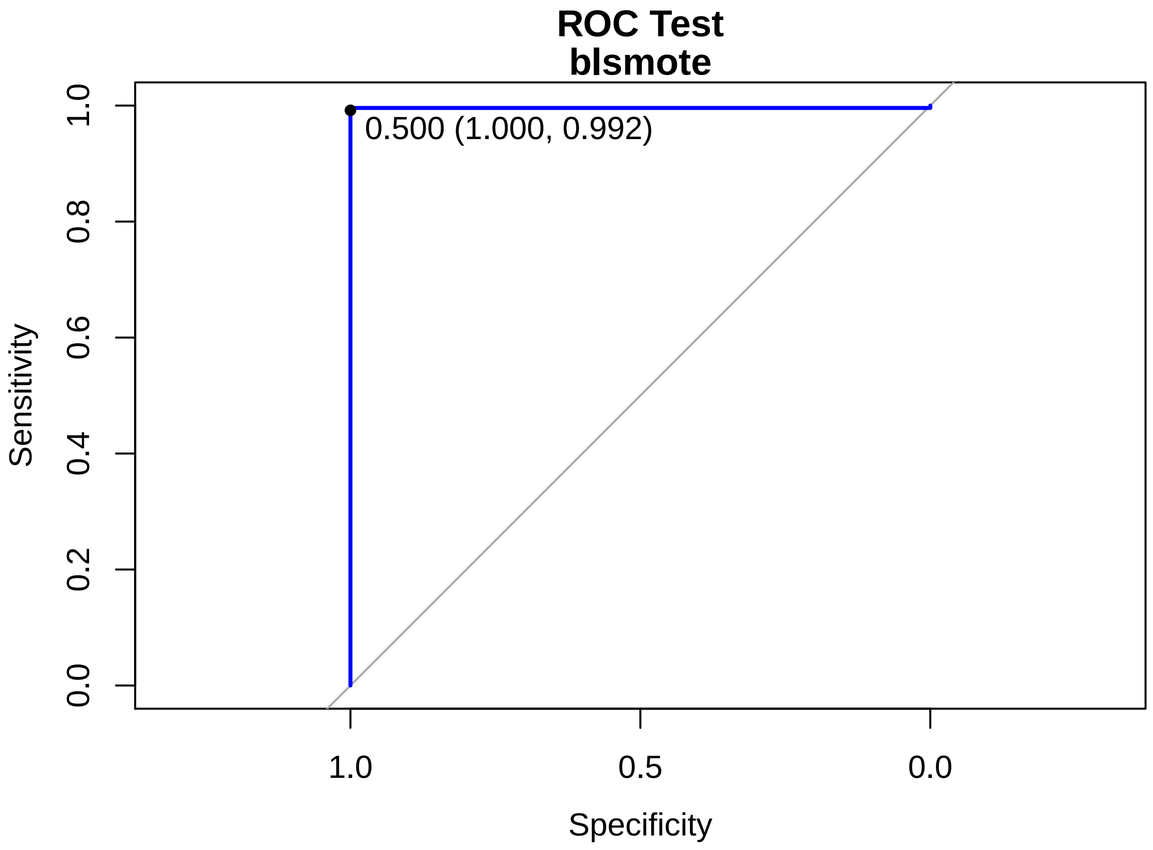
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction positive negative
##   positive       93        9
##   negative        1       96
##
##                Accuracy : 0.9497
##                  95% CI : (0.9095, 0.9756)
##     No Information Rate : 0.5276
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8996
##  Mcnemar's Test P-Value : 0.02686
##
##             Sensitivity : 0.9894
##             Specificity : 0.9143
##          Pos Pred Value : 0.9118
##          Neg Pred Value : 0.9897
##              Prevalence : 0.4724
##          Detection Rate : 0.4673
##    Detection Prevalence : 0.5126
##       Balanced Accuracy : 0.9518
##
##        'Positive' Class : positive
##
```



```
model.blsmote.2 <- learn_model(train_data2,ctrl,"blsmote")
cm.blsmote.2<- test_model(test_data2,model.blsmote.2,"blsmote")
```

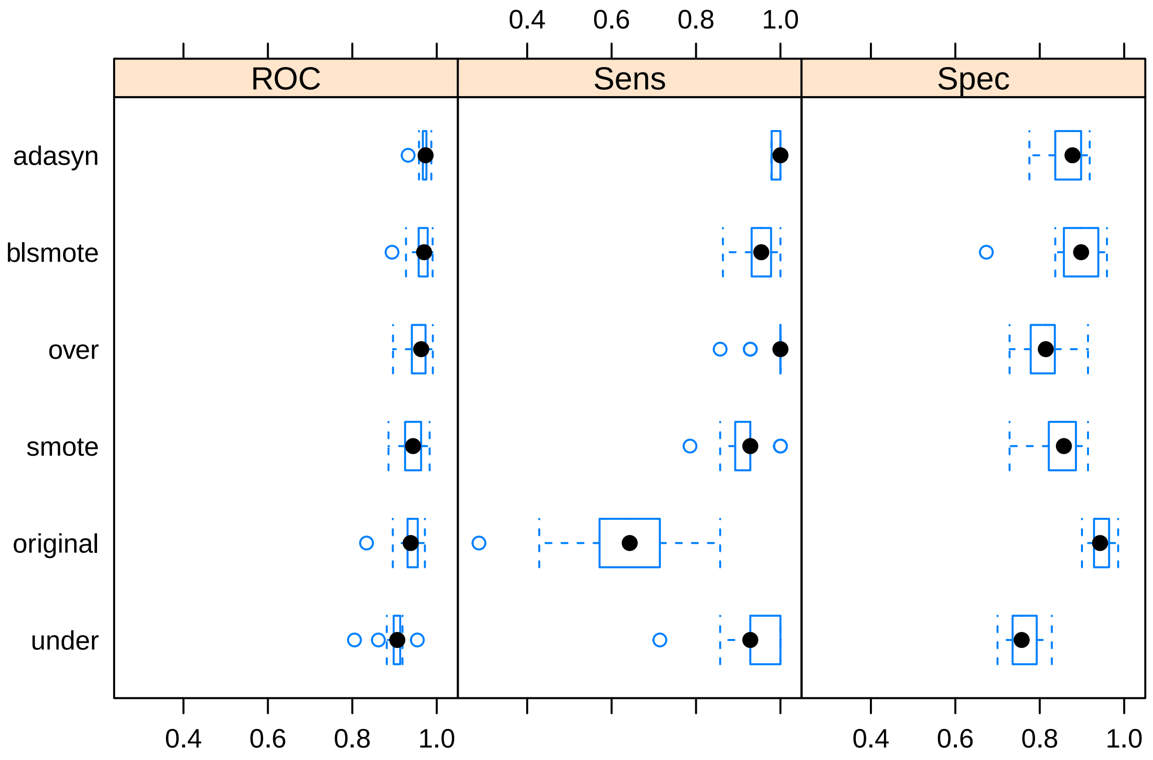
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction positive negative
##   positive      441        4
##   negative        0      486
##
##                Accuracy : 0.9957
##                  95% CI : (0.989, 0.9988)
##     No Information Rate : 0.5263
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9914
##  Mcnemar's Test P-Value : 0.1336
##
##             Sensitivity : 1.0000
##             Specificity : 0.9918
##          Pos Pred Value : 0.9910
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4737
##          Detection Rate : 0.4737
##    Detection Prevalence : 0.4780
##       Balanced Accuracy : 0.9959
##
```

```
##        'Positive' Class : positive
##
```

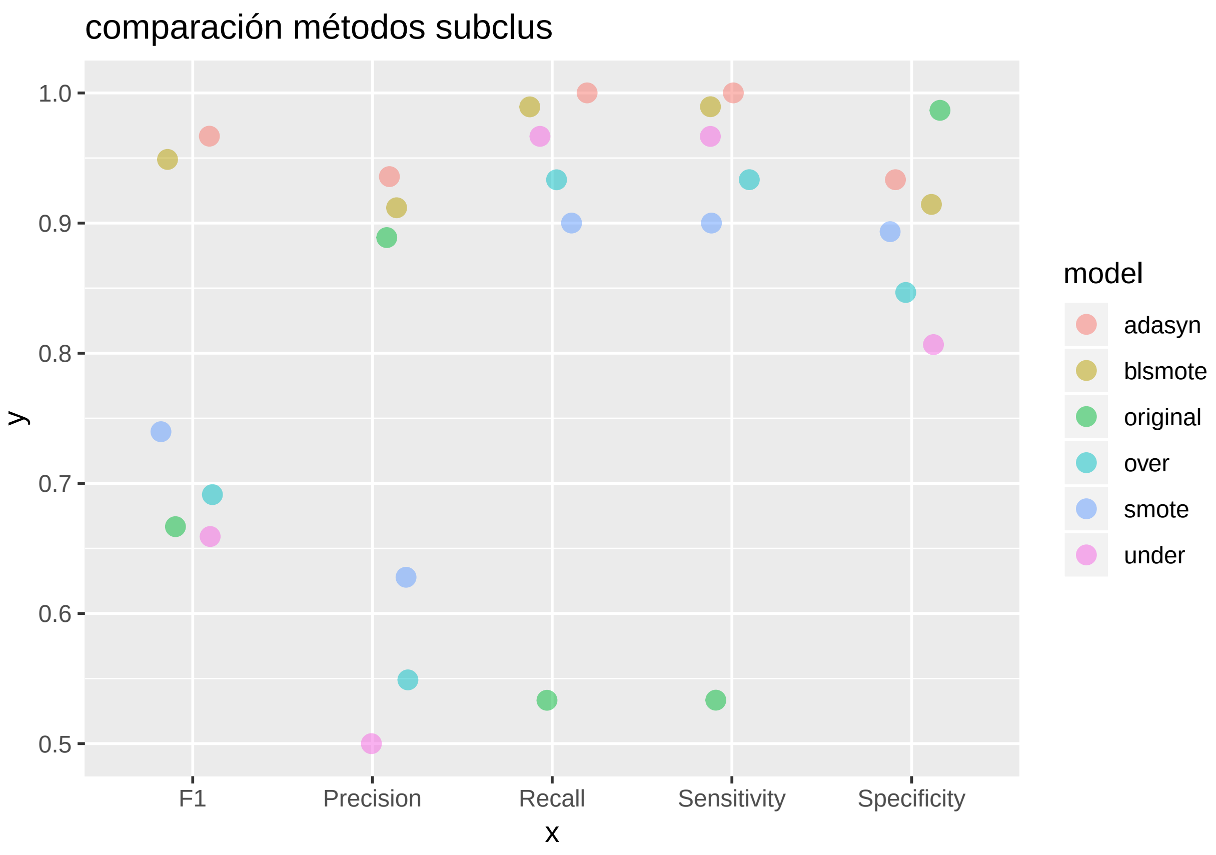


Ahora creamos de nuevo las gráficas comparativas de los modelos añadiendo los nuevos modelos generados.

```r
models <- list(original = model.raw,
               under = model.us,
               over = model.os,
               smote = model.smt,
               adasyn = model.adasyn,
               blsmote = model.blsmote)

resampling <- resamples(models)
bwplot(resampling)
```

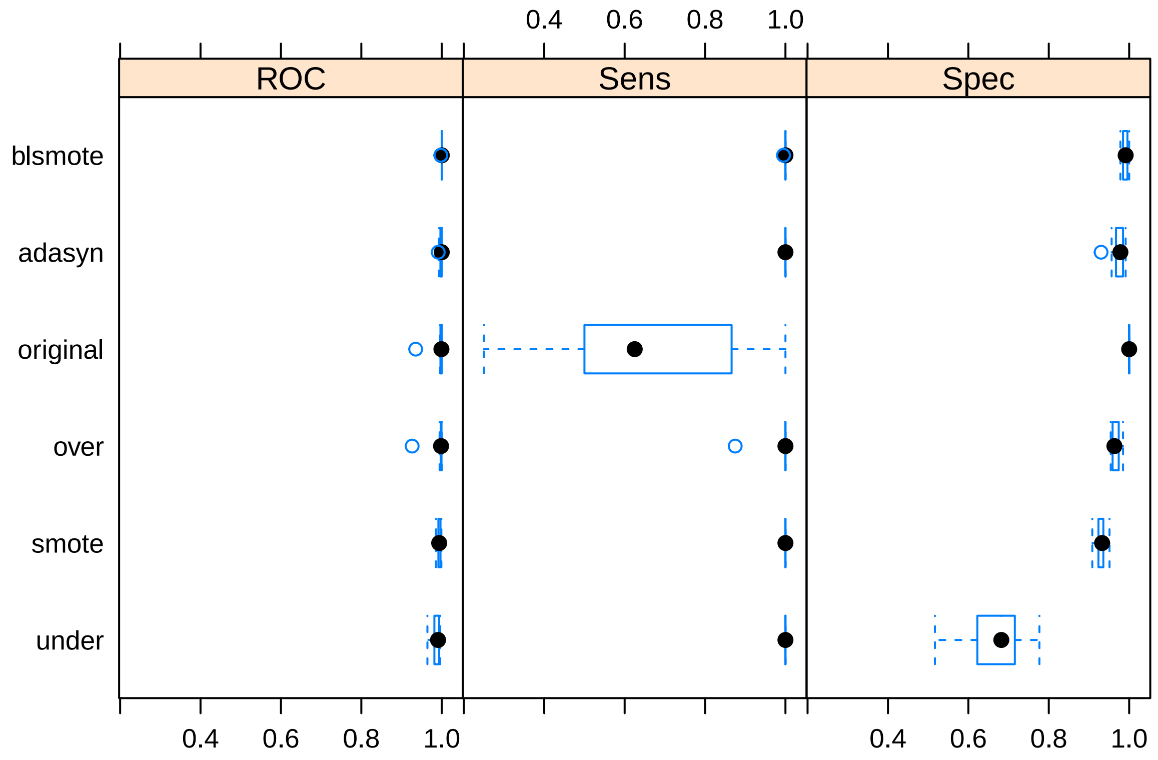


```
comparison <- data.frame(model = names(models),
                         Sensitivity = rep(NA, length(models)),
                         Specificity = rep(NA, length(models)),
                         Precision = rep(NA, length(models)),
                         Recall = rep(NA, length(models)),
                         F1 = rep(NA, length(models)))

for (name in names(models)) {
  cm_model <- get(paste0("cm.", name))

  comparison[comparison$model == name, ] <- filter(comparison, model == name) %>%
    mutate(Sensitivity = cm_model$byClass["Sensitivity"],
           Specificity = cm_model$byClass["Specificity"],
           Precision = cm_model$byClass["Precision"],
           Recall = cm_model$byClass["Recall"],
           F1 = cm_model$byClass["F1"])
}

comparison %>%
  gather(x, y, Sensitivity:F1) %>%
  ggplot(aes(x = x, y = y, color = model)) +
  geom_jitter(width = 0.2, alpha = 0.5, size = 3)+
  ggtitle("comparación métodos subclus")
```

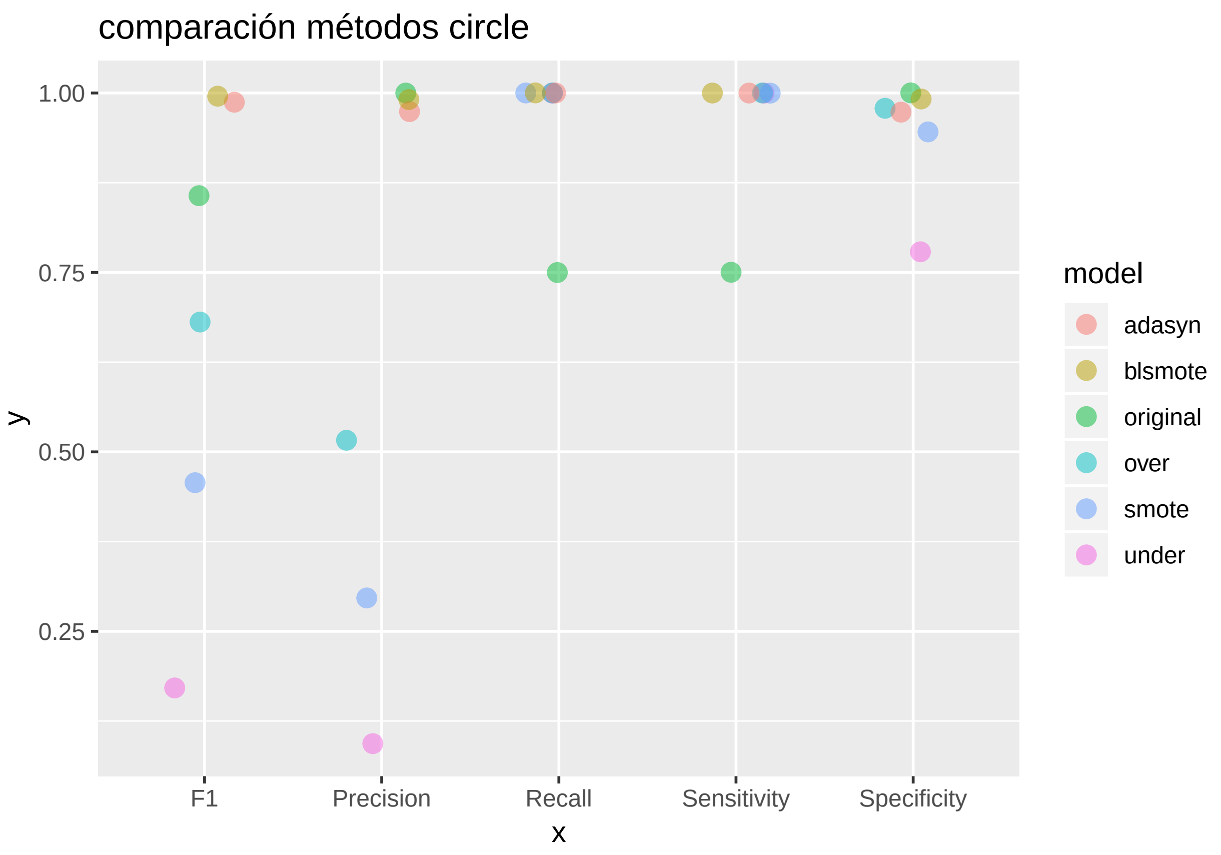


Como se puede ver en la gráfica, para el dataset *subclus* el modelo de adasyn y blsmote obtienen unos resultados muy parejos los dos. Esto es normal ya que las transformaciones que han realizado a los datos son muy parecidas.

```
models <- list(original = model.raw.2,
               under = model.us.2,
               over = model.os.2,
               smote = model.smt.2,
               adasyn = model.adasyn.2,
               blsmote = model.blsmote.2)

resampling <- resamples(models)
bwplot(resampling)
```



```
comparison <- data.frame(model = names(models),
                         Sensitivity = rep(NA, length(models)),
                         Specificity = rep(NA, length(models)),
                         Precision = rep(NA, length(models)),
                         Recall = rep(NA, length(models)),
                         F1 = rep(NA, length(models)))

for (name in names(models)) {
  cm_model <- get(paste0("cm.", name,".2"))

  comparison[comparison$model == name, ] <- filter(comparison, model == name) %>%
    mutate(Sensitivity = cm_model$byClass["Sensitivity"],
           Specificity = cm_model$byClass["Specificity"],
           Precision = cm_model$byClass["Precision"],
           Recall = cm_model$byClass["Recall"],
           F1 = cm_model$byClass["F1"])
}

comparison %>%
  gather(x, y, Sensitivity:F1) %>%
  ggplot(aes(x = x, y = y, color = model)) +
  geom_jitter(width = 0.2, alpha = 0.5, size = 3)+
  ggtitle("comparación métodos circle")
```



Para el caso del dataset *circle* nos ocurre los mismo, los datos que obtenemos con ambos algoritmos son iguales; la razón es la misma que en el caso anterior, ambos algoritmos han realizado transformaciones muy parecidas a los datos (si se miran los scatter plots, se puede ver que ambos datos han reforzado la clase minoritaria en la frontera) y por ello los resultados son también muy parecidos.