Imbalanced classification

Minería de datos: aspectos avanzados

Francisco Luque Sánchez

21/12/2019

1 Introduction

In this report, the problem of imbalanced classification will be addressed. In the first section, we will show a typical workflow to solve the imbalance classification problem, using the dataset *Subclus*. This dataset is an artificially generated two dimensional dataset whose positive class is grouped in a few small subgroups. In the second section, the performance of some SMOTE-based oversampling methods will be tested over that same dataset. Finally, in the third section, we will dig deeper in the classic version of SMOTE, trying to understand how the solution of the problem is influenced by its parameters.

2 Standard imbalanced classification pipeline

In this section, a classical pipeline of imbalanced classification will be shown. We begin by loading the dataset and renaming the variables properly:

```r
## Dataset loading and column names setting
dataset <- read.csv("subclus.csv")
colnames(dataset) <- c("Att1", "Att2", "Class")
dataset$Class <- relevel(dataset$Class, "positive")
```

At first, we are interested in knowing about the dataset (variables, types, dimensions…):

```r
## DATASET SUMMARY
## Dimensions
dim(dataset)
```

```
## [1] 599   3
```

```r
## Structure and type
str(dataset)
```

```
## 'data.frame':    599 obs. of  3 variables:
##  $ Att1 : int  187 290 194 204 196 201 289 116 199 174 ...
##  $ Att2 : int  34 -57 -80 89 -81 -17 4 -95 38 33 ...
##  $ Class: Factor w/ 2 levels "positive","negative": 1 1 1 1 1 1 1 1 1 1 ...
```

```r
## First rows of the data
kable(head(dataset))
```

Att1	Att2	Class
187	34	positive
290	-57	positive
194	-80	positive
204	89	positive
196	-81	positive
201	-17	positive

```r
## Class levels
levels(dataset$Class)
```

```
## [1] "positive" "negative"
```

As we can sy in the output of previous commands, our dataset is composed of 599 examples of 3 variables (two numeric and the class). It is binary classificacion problem, with classes named *negative* and *positive*.

```r
## Columns summarization
summary(dataset)
```

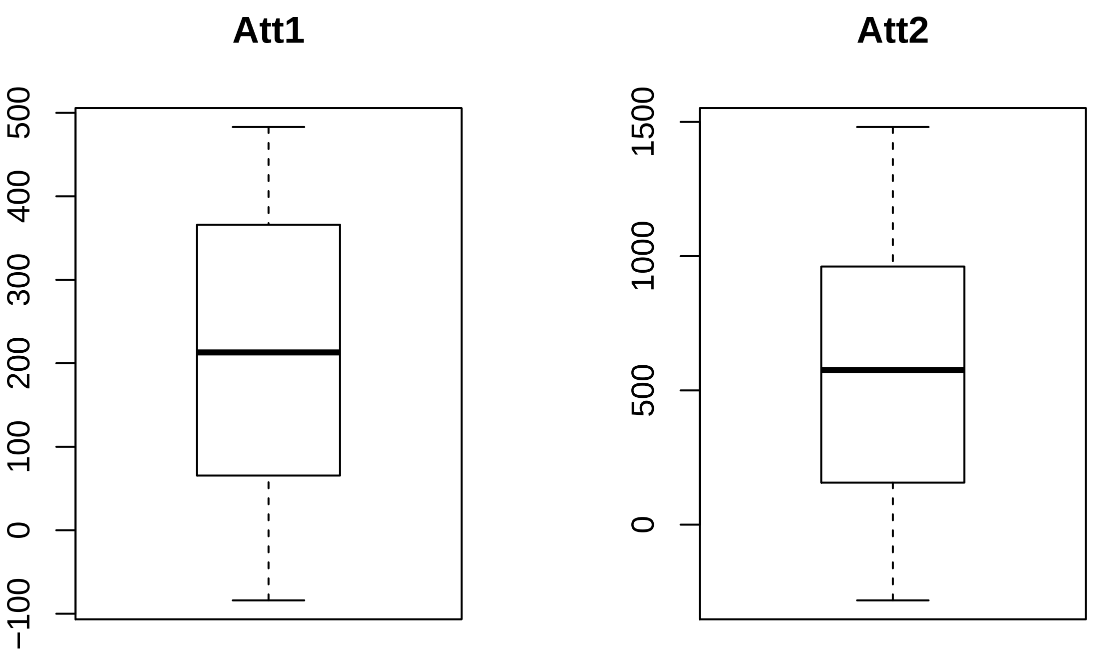
```
##       Att1             Att2               Class
##  Min.   :-84.0   Min.   :-282.0   positive: 99
##  1st Qu.: 65.5   1st Qu.: 156.5   negative:500
##  Median :213.0   Median : 576.0
##  Mean   :214.2   Mean   : 575.3
##  3rd Qu.:366.0   3rd Qu.: 961.5
##  Max.   :483.0   Max.   :1481.0
```

```r
## Imbalance ratio
imbalanceRatio(dataset)
```

```
## [1] 0.198
```

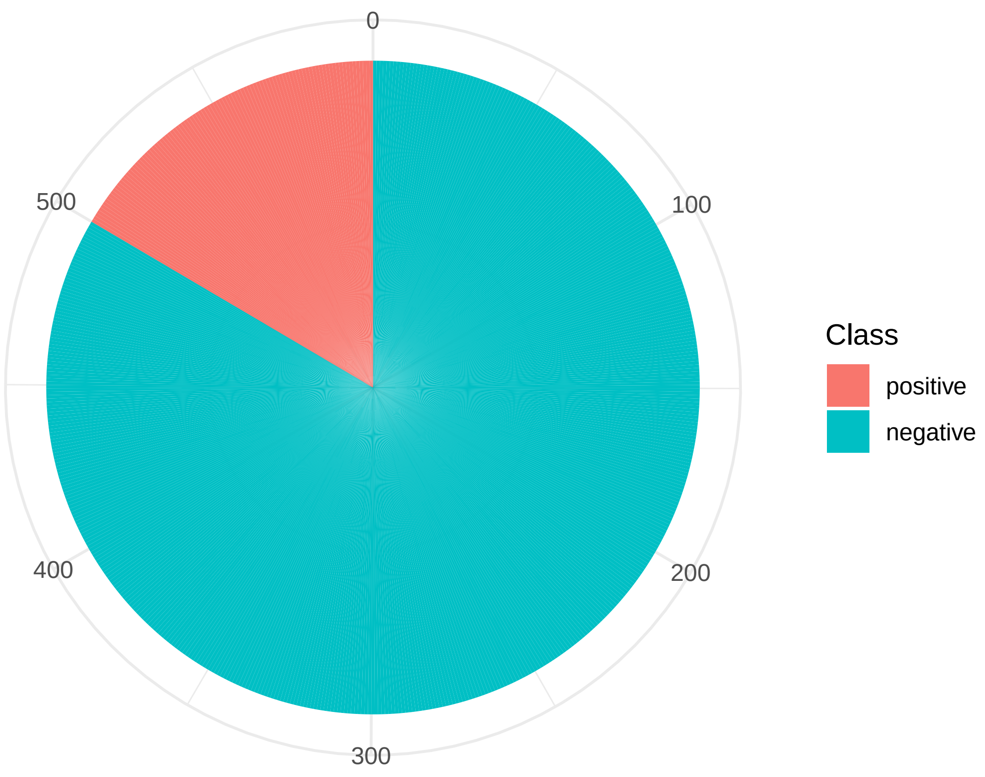
The imbalance ratio of the dataset is not very pronounced (approximately 1 to 5). It is far from the 1 to 40 that we had in other examples. However, it is important enough to be addressed as an imbalanced dataset. Now, we will try and visualize the data. We begin with a boxplot of the attributes and a piechart of the classes distribution:

```r
## Dataset visualization
x <- dataset[,1:2]
y <- dataset[,3]

## Attributes boxplot
par(mfrow=c(1,2))
out <- sapply(1:2, function(i) boxplot(x[,i], main=names(dataset)[i]))
```



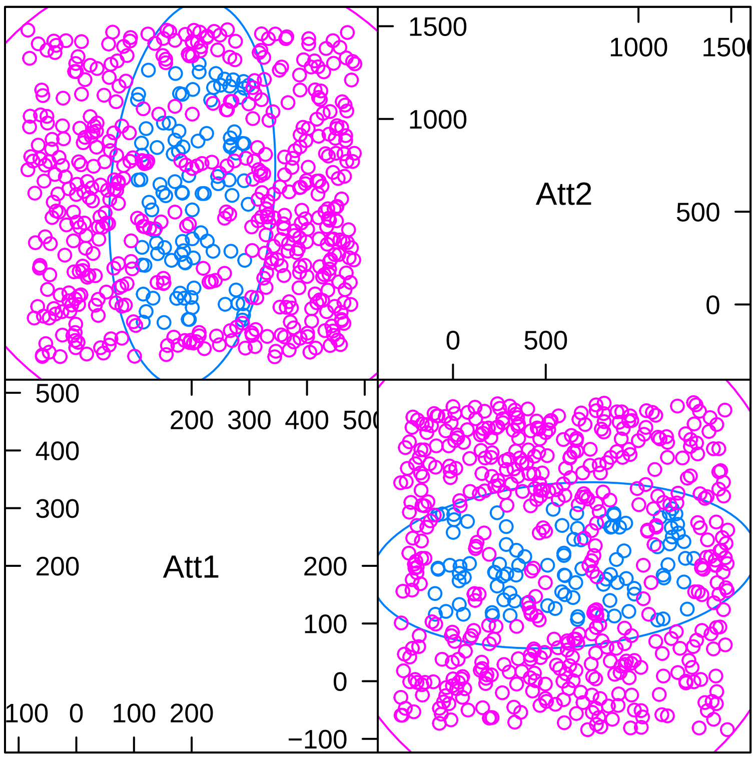
The data distribution along the variables is slightly different. The second attribute is much sparser than the first, with a range three times bigger.

```
## Classes piechart
ggplot(dataset, aes(x="", y=1, fill=Class))+
    geom_bar(width = 1, stat = "identity")+
    coord_polar("y", start=0)+ theme_minimal()+
    theme(axis.title.x=element_blank(), axis.title.y = element_blank())
```



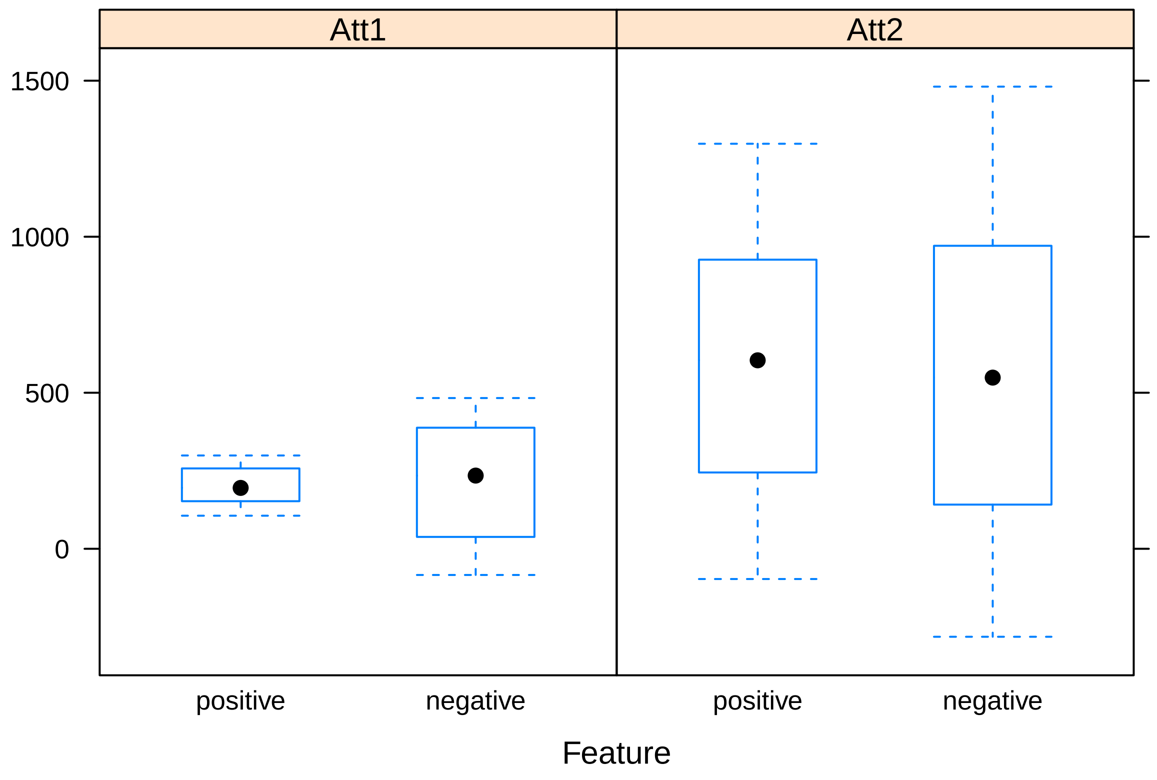
As we said before, in the chart can be seen that there are approximately 5 times more data in the negative class than in the positive one. Now, we will plot the data in

```
## Scatterplot
featurePlot(x=x, y=y, plot="ellipse")
```



Scatter Plot Matrix

```
## Per class boxplot
featurePlot(x=x, y=y, plot="box")
```



```r
set.seed(42) #To ensure the same output

## An easy way to create split "data partitions":
trainIndex <- createDataPartition(dataset$Class, p = .75,
                                  list = FALSE,
                                  times = 1)
trainData <- dataset[ trainIndex,]
testData  <- dataset[-trainIndex,]

## Check IR to ensure a stratified partition
imbalanceRatio(trainData)
```

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

```r
imbalanceRatio(testData)
```

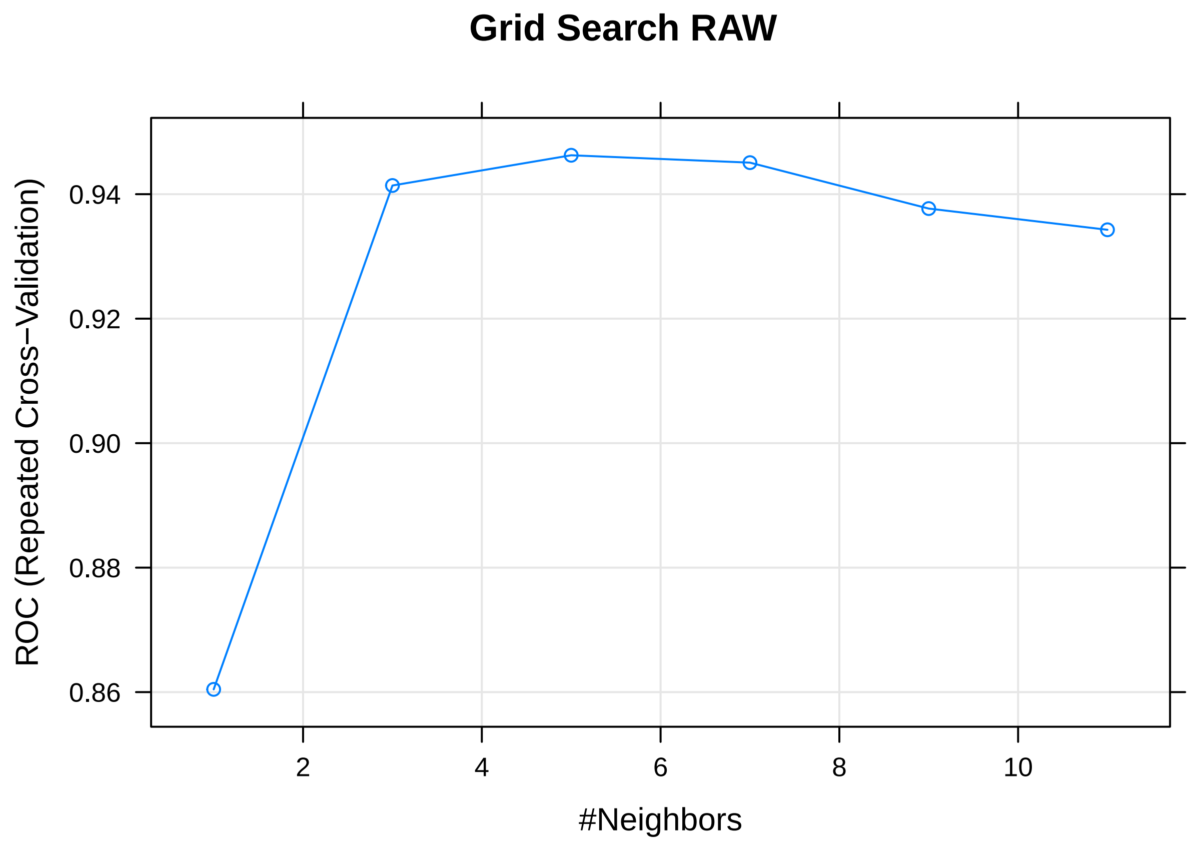
```
## [1] 0.192
```

```
learn_model <-function(dataset, ctrl, message){
    model.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)))
    model.pred <- predict(model.fit,newdata = dataset)
    ## Get the confusion matrix to see accuracy value and other parameter
↪   values
    model.cm <- confusionMatrix(model.pred, dataset$Class,positive =
↪   "positive")
    model.probs <- predict(model.fit,newdata = dataset, type="prob")
    model.roc <- roc(dataset$Class,model.probs[,"positive"],color="green")
    return(model.fit)
}

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

```
## Execute model ("raw" data)
ctrl <- trainControl(method="repeatedcv",number=5,repeats = 3,
                     classProbs=TRUE,summaryFunction = twoClassSummary)
model.raw <- learn_model(trainData,ctrl,"RAW ")
```

```
## Setting levels: control = positive, case = negative

## Setting direction: controls > cases
```

```r
## We may decide to plot the results from the grid search of the
## model's parameters
plot(model.raw,main="Grid Search RAW")
```



```r
print(model.raw)
```

```
## k-Nearest Neighbors
##
## 450 samples
##   2 predictor
##   2 classes: 'positive', 'negative'
##
## Pre-processing: centered (2), scaled (2)
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 360, 360, 360, 360, 360, 360, ...
## Resampling results across tuning parameters:
##
```

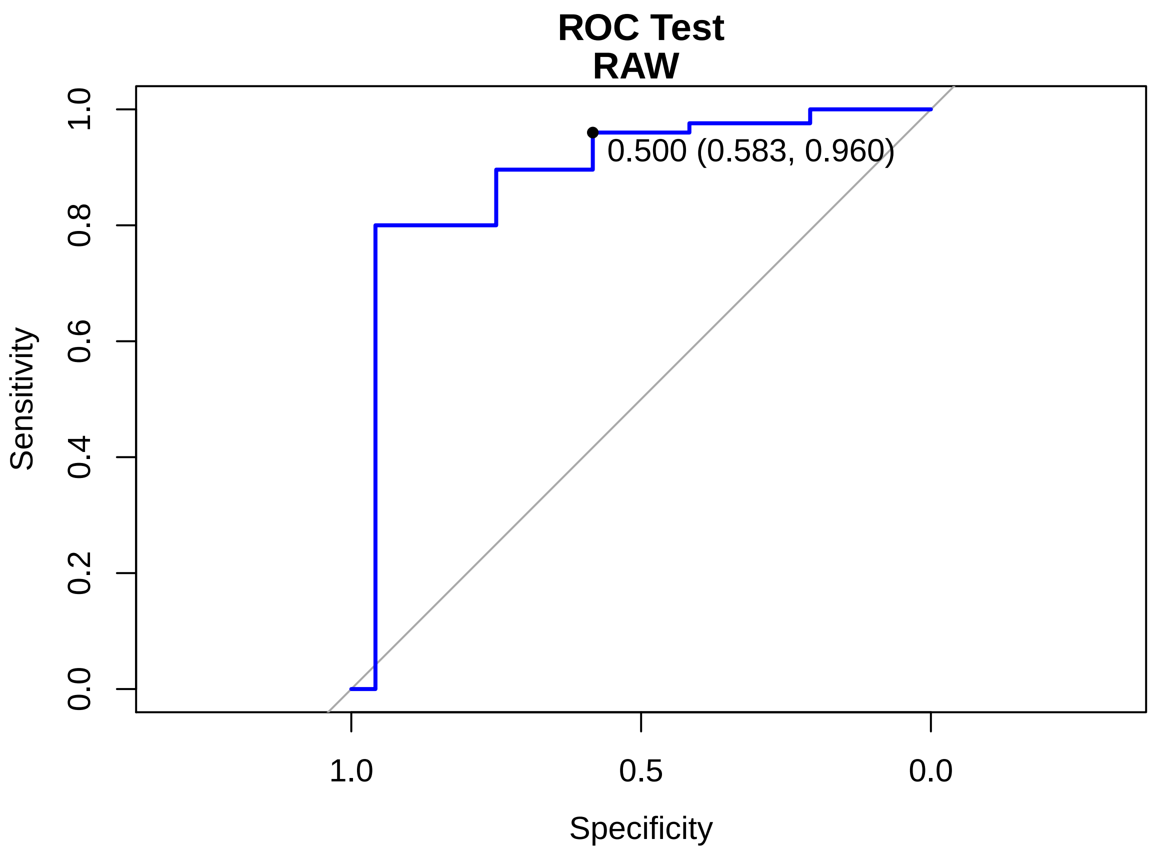
```
##    k   ROC        Sens       Spec     
##    1  0.8604444  0.7644444  0.9564444
##    3  0.9413926  0.6977778  0.9502222
##    5  0.9462519  0.6977778  0.9475556
##    7  0.9450667  0.6622222  0.9404444
##    9  0.9376889  0.6488889  0.9288889
##   11  0.9342815  0.6222222  0.9271111
## 
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

```
cm.raw <- test_model(testData,model.raw,"RAW ")
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction positive negative
##   positive       14        5
##   negative       10      120
##                                           
##                Accuracy : 0.8993          
##                  95% CI : (0.8394, 0.9426)
##     No Information Rate : 0.8389          
##     P-Value [Acc > NIR] : 0.02414         
##                                           
##                   Kappa : 0.5933          
##                                           
##  Mcnemar's Test P-Value : 0.30170         
##                                           
##             Sensitivity : 0.58333         
##             Specificity : 0.96000         
##          Pos Pred Value : 0.73684         
##          Neg Pred Value : 0.92308         
##              Prevalence : 0.16107         
##          Detection Rate : 0.09396         
##    Detection Prevalence : 0.12752         
##       Balanced Accuracy : 0.77167         
## 
```

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

```
## Setting levels: control = positive, case = negative
## Setting direction: controls > cases
```



```
## Execute model ("preprocessed" data)
## Undersampling
ctrl <- trainControl(method="repeatedcv",number=5,repeats = 3,
                     classProbs=TRUE,summaryFunction =
                     ↪  twoClassSummary,sampling = "down")

model.us <- learn_model(trainData,ctrl,"US ")
```

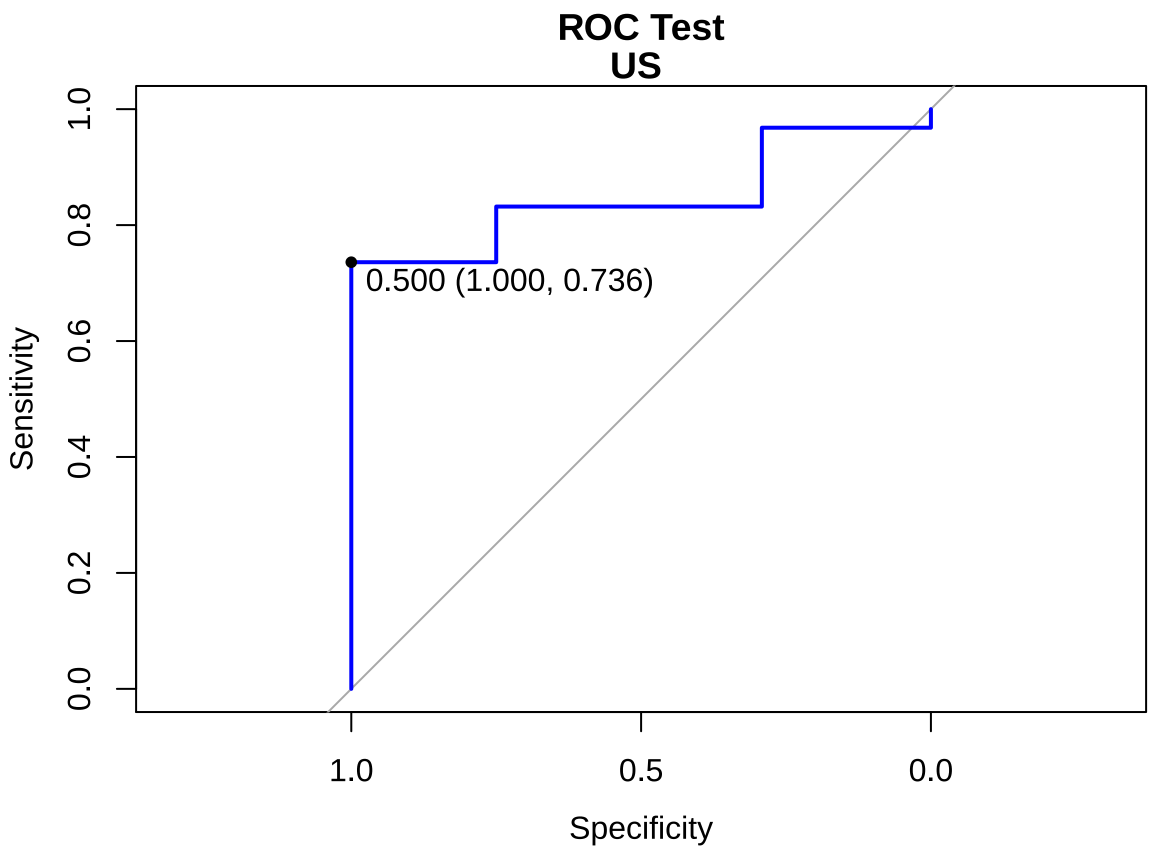
```
## Setting levels: control = positive, case = negative
```

```
## Setting direction: controls > cases
```

```r
cm.us <- test_model(testData,model.us,"US ")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction positive negative
##   positive       24       33
##   negative        0       92
##
##                Accuracy : 0.7785
##                  95% CI : (0.7033, 0.8424)
##     No Information Rate : 0.8389
##     P-Value [Acc > NIR] : 0.9795
##
##                   Kappa : 0.4732
##
##  Mcnemar's Test P-Value : 2.54e-08
##
##             Sensitivity : 1.0000
##             Specificity : 0.7360
##          Pos Pred Value : 0.4211
##          Neg Pred Value : 1.0000
##              Prevalence : 0.1611
##          Detection Rate : 0.1611
##    Detection Prevalence : 0.3826
##       Balanced Accuracy : 0.8680
##
##        'Positive' Class : positive
##
```

```
## Setting levels: control = positive, case = negative
## Setting direction: controls > cases
```



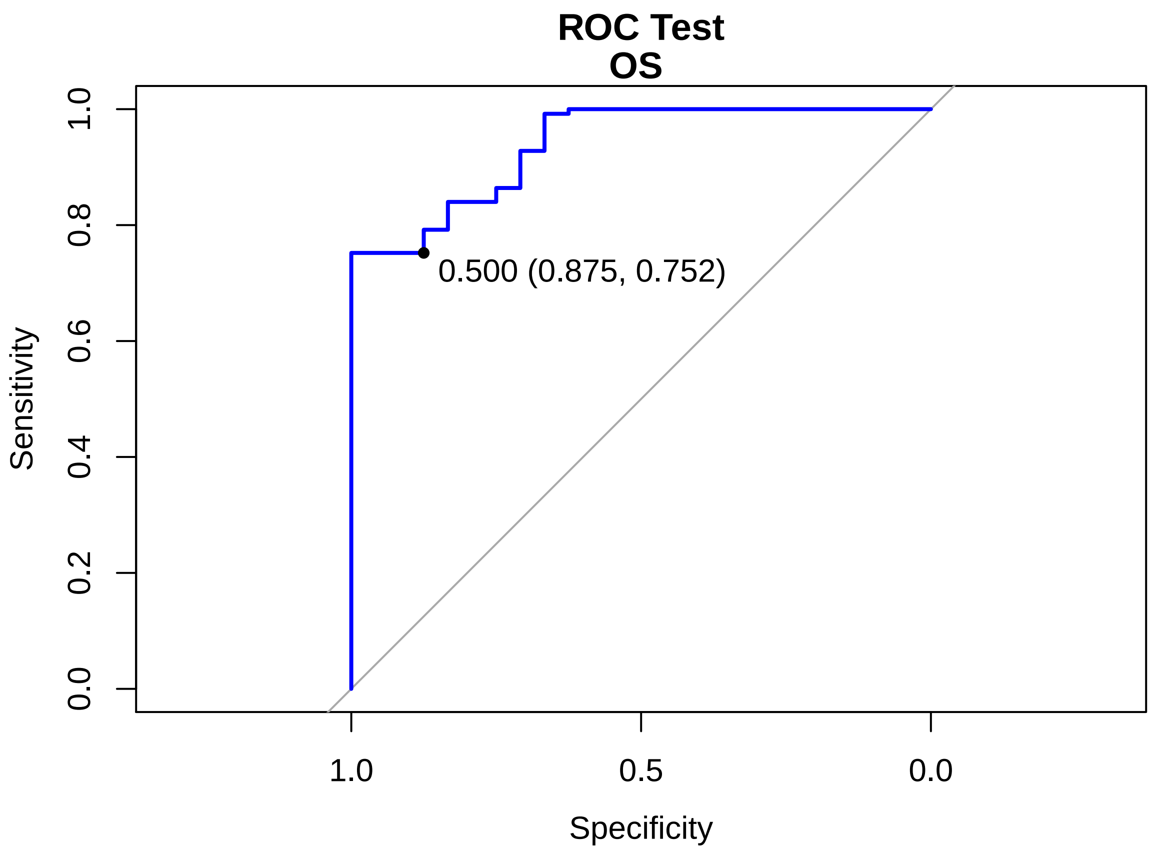
```r
## Oversampling
ctrl <- trainControl(method="repeatedcv",number=5,repeats = 3,
                     classProbs=TRUE,summaryFunction =
                      ↪  twoClassSummary,sampling = "up")
model.os <- learn_model(trainData,ctrl,"OS ")
```

```
## Setting levels: control = positive, case = negative

## Setting direction: controls > cases
```

```r
cm.os <- test_model(testData,model.os,"OS ")
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction positive negative
##   positive       21       31
##   negative        3       94
##
##                Accuracy : 0.7718
##                  95% CI : (0.696, 0.8365)
##     No Information Rate : 0.8389
##     P-Value [Acc > NIR] : 0.9877
##
##                   Kappa : 0.4261
##
##  Mcnemar's Test P-Value : 3.649e-06
##
##             Sensitivity : 0.8750
##             Specificity : 0.7520
##          Pos Pred Value : 0.4038
##          Neg Pred Value : 0.9691
##              Prevalence : 0.1611
##          Detection Rate : 0.1409
##    Detection Prevalence : 0.3490
##       Balanced Accuracy : 0.8135
##
##        'Positive' Class : positive
##
```

```
## Setting levels: control = positive, case = negative
## Setting direction: controls > cases
```



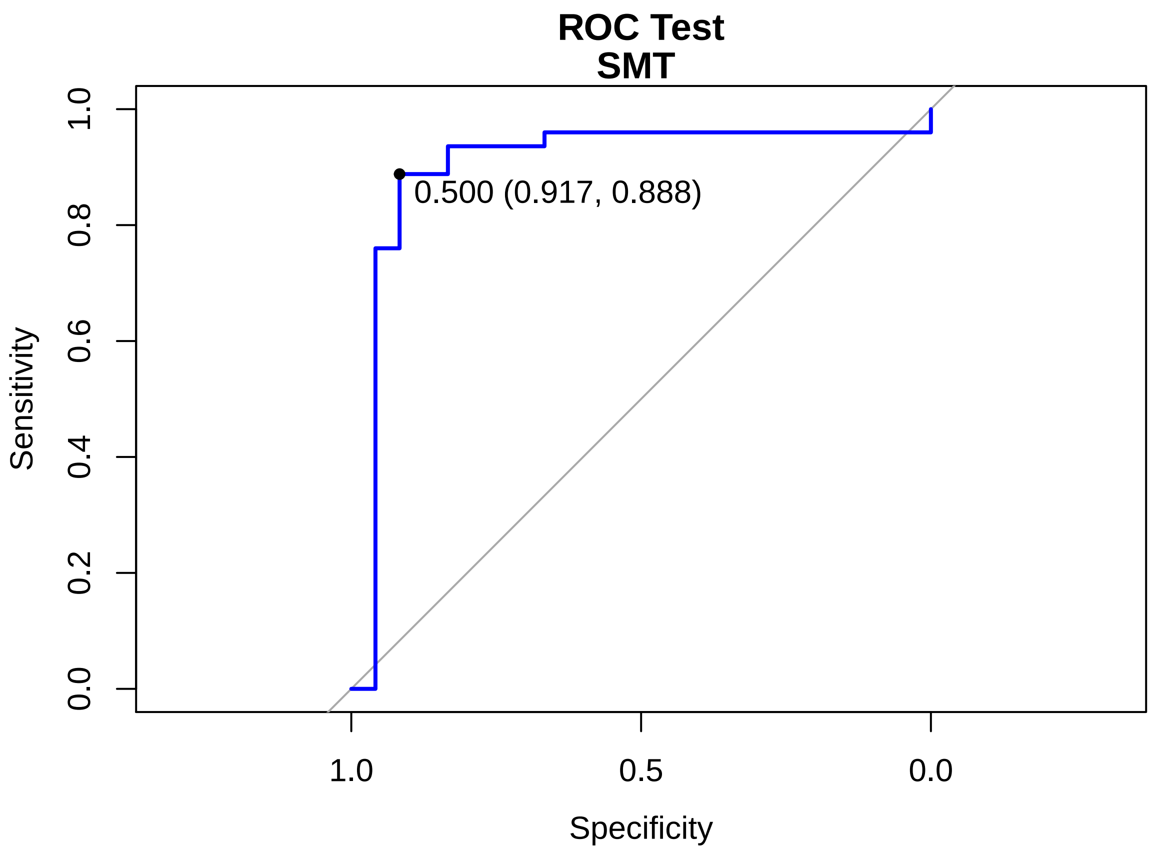
```
## SMOTE
ctrl <- trainControl(method="repeatedcv",number=5,repeats = 3,
                     classProbs=TRUE,summaryFunction =
                      ↪  twoClassSummary,sampling = "smote")
model.smt <- learn_model(trainData,ctrl,"SMT ")
```

```
## Loading required package: grid

## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

## Setting levels: control = positive, case = negative
## Setting direction: controls > cases
```

```
cm.smt <- test_model(testData,model.smt,"SMT ")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction positive negative
##   positive       22       14
##   negative        2      111
##
##                Accuracy : 0.8926
##                  95% CI : (0.8315, 0.9374)
##     No Information Rate : 0.8389
##     P-Value [Acc > NIR] : 0.04215
##
##                   Kappa : 0.6694
##
##  Mcnemar's Test P-Value : 0.00596
##
##             Sensitivity : 0.9167
##             Specificity : 0.8880
##          Pos Pred Value : 0.6111
##          Neg Pred Value : 0.9823
##              Prevalence : 0.1611
##          Detection Rate : 0.1477
##    Detection Prevalence : 0.2416
##       Balanced Accuracy : 0.9023
##
##        'Positive' Class : positive
##
```

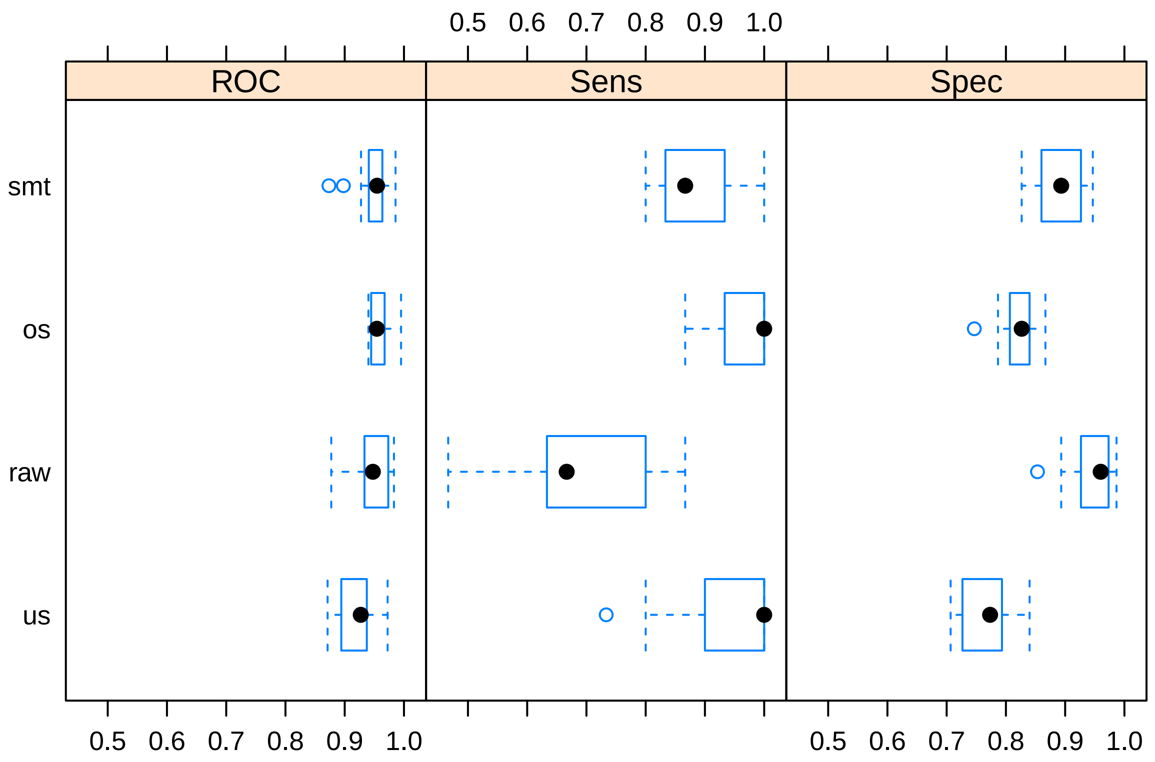
```
## Setting levels: control = positive, case = negative
## Setting direction: controls > cases
```



```
## summarize accuracy of models
models <- list(raw = model.raw,us = model.us,os = model.os,smt = model.smt)
results <- resamples(models)
summary(results)
```

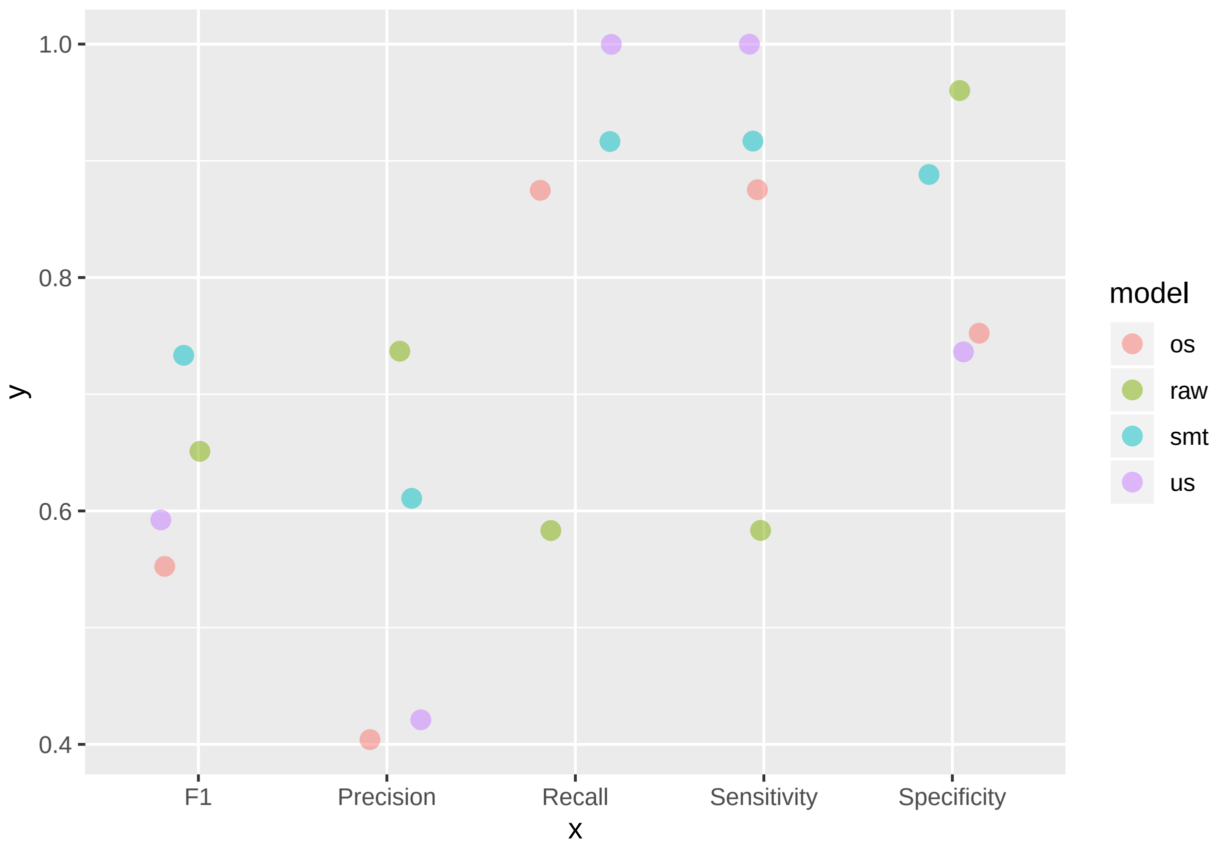
```
##
## Call:
## summary.resamples(object = results)
##
## Models: raw, us, os, smt
## Number of resamples: 15
##
## ROC
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## raw 0.8773333 0.9333333 0.9475556 0.9462519 0.9735556 0.9831111    0
## us  0.8711111 0.8942222 0.9271111 0.9188444 0.9373333 0.9724444    0
## os  0.9400000 0.9446667 0.9542222 0.9590815 0.9673333 0.9951111    0
## smt 0.8733333 0.9406667 0.9546667 0.9480593 0.9635556 0.9857778    0
##
## Sens
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## raw 0.4666667 0.6333333 0.6666667 0.6977778 0.8000000 0.8666667    0
## us  0.7333333 0.9000000 1.0000000 0.9333333 1.0000000 1.0000000    0
## os  0.8666667 0.9333333 1.0000000 0.9733333 1.0000000 1.0000000    0
## smt 0.8000000 0.8333333 0.8666667 0.8800000 0.9333333 1.0000000    0
##
## Spec
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## raw 0.8533333 0.9266667 0.9600000 0.9475556 0.9733333 0.9866667    0
## us  0.7066667 0.7266667 0.7733333 0.7688889 0.7933333 0.8400000    0
## os  0.7466667 0.8066667 0.8266667 0.8222222 0.8400000 0.8666667    0
## smt 0.8266667 0.8600000 0.8933333 0.8915556 0.9266667 0.9466667    0
```

```
## Compare accuracy of models
bwplot(results)
```



```
#dotplot(results)
```

```
## Carry out a comparison over all imbalanced metrics
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)
```



2.1 Package imbalance utilization

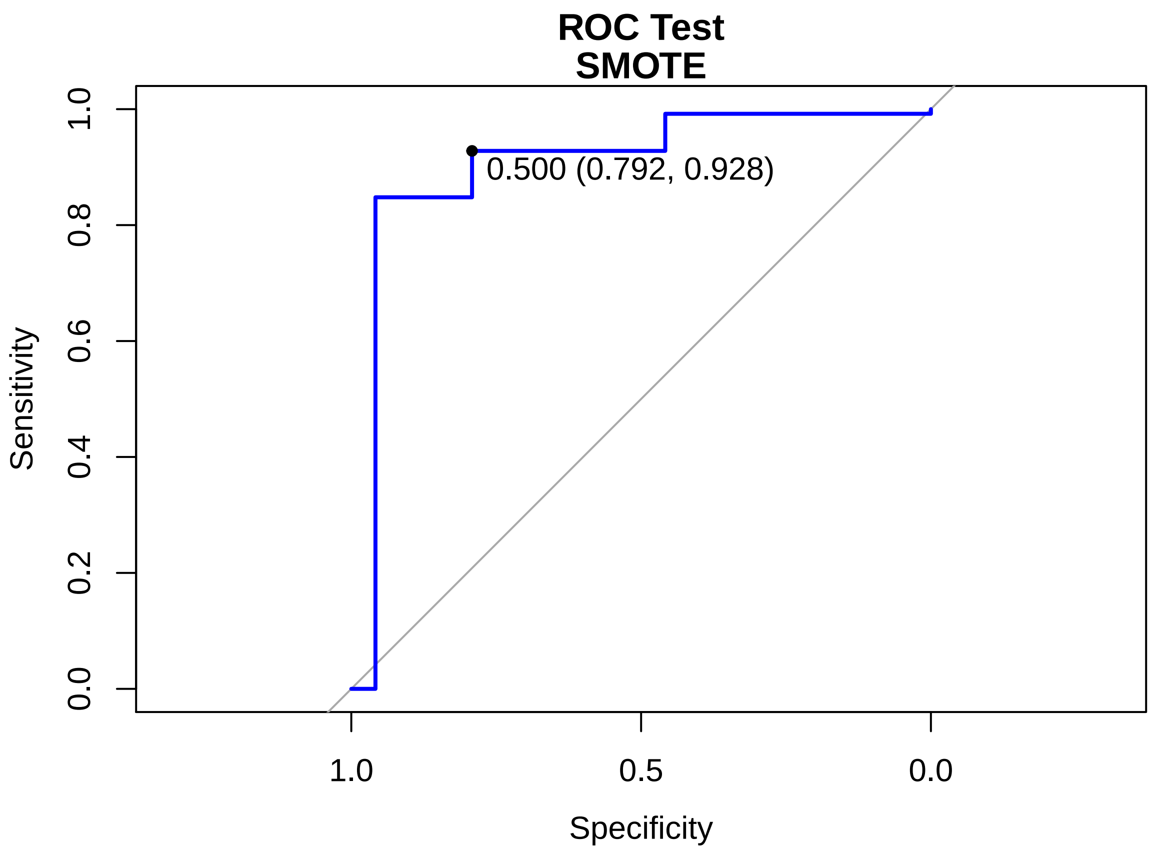
```
## Oversampling using classic SMOTE
trainData.smote <- oversample(trainData, ratio=0.4, method="SMOTE")

## TrainControl without
ctrl <- trainControl(method="repeatedcv",number=5,repeats = 3,
                     classProbs=TRUE, summaryFunction = twoClassSummary)


model.smt <- learn_model(trainData.smote, ctrl, "SMOTE")
```

```
## Setting levels: control = positive, case = negative

## Setting direction: controls > cases
```

```r
cm.smt <- test_model(testData, model.smt, "SMOTE")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction positive negative
##   positive       19        9
##   negative        5      116
##
##                Accuracy : 0.906
##                  95% CI : (0.8474, 0.9477)
##     No Information Rate : 0.8389
##     P-Value [Acc > NIR] : 0.01295
##
##                   Kappa : 0.6743
##
##  Mcnemar's Test P-Value : 0.42268
##
##             Sensitivity : 0.7917
##             Specificity : 0.9280
##          Pos Pred Value : 0.6786
##          Neg Pred Value : 0.9587
##              Prevalence : 0.1611
##          Detection Rate : 0.1275
##    Detection Prevalence : 0.1879
##       Balanced Accuracy : 0.8598
##
##        'Positive' Class : positive
##
```

```
## Setting levels: control = positive, case = negative
## Setting direction: controls > cases
```



```r
## Oversampling using classic
trainData.smote <- oversample(trainData, ratio=0.6, method="MWMOTE")

## TrainControl without
ctrl <- trainControl(method="repeatedcv",number=5,repeats = 3,
                     classProbs=TRUE, summaryFunction = twoClassSummary)


model.smt <- learn_model(trainData.smote, ctrl, "SMOTE")
```

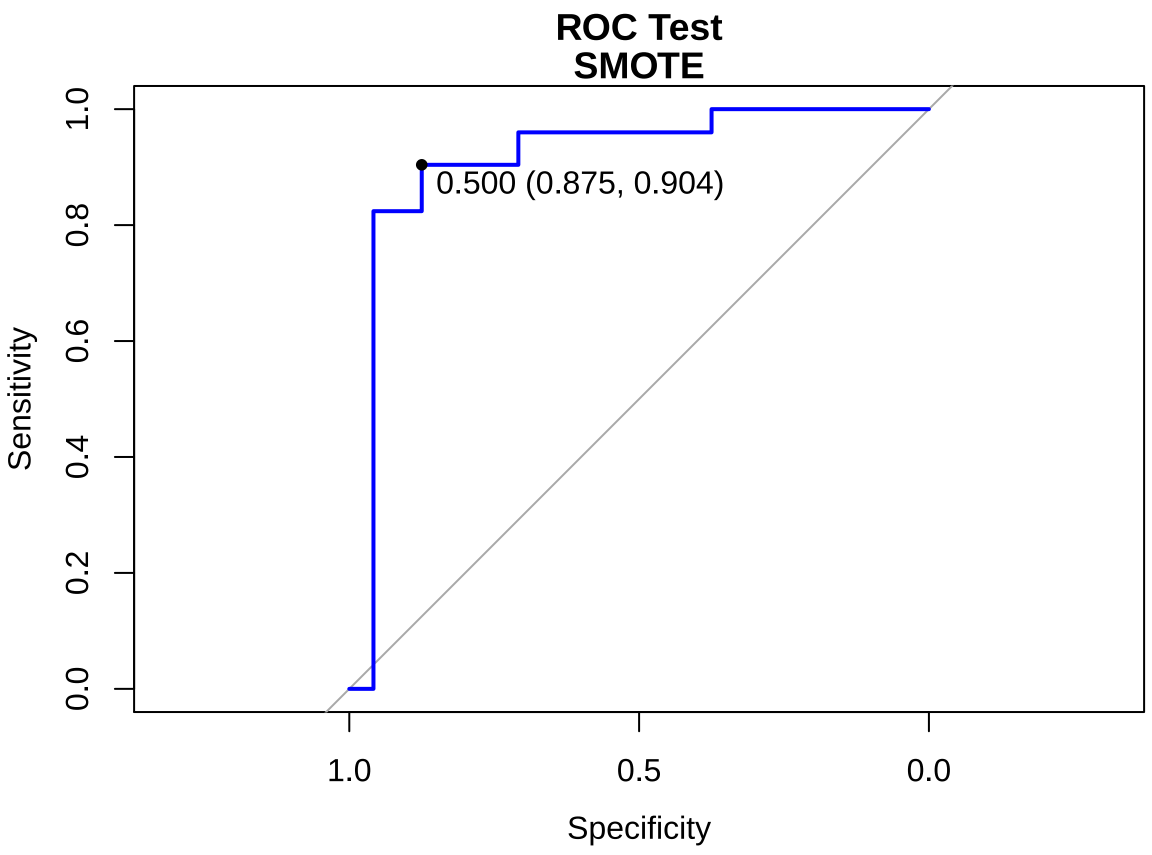
```
## Setting levels: control = positive, case = negative

## Setting direction: controls > cases
```

```r
cm.smt <- test_model(testData, model.smt, "SMOTE")
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction positive negative
##   positive       21       13
##   negative        3      112
##
##                Accuracy : 0.8926
##                  95% CI : (0.8315, 0.9374)
##     No Information Rate : 0.8389
##     P-Value [Acc > NIR] : 0.04215
##
##                   Kappa : 0.6599
##
##  Mcnemar's Test P-Value : 0.02445
##
##             Sensitivity : 0.8750
##             Specificity : 0.8960
##          Pos Pred Value : 0.6176
##          Neg Pred Value : 0.9739
##              Prevalence : 0.1611
##          Detection Rate : 0.1409
##    Detection Prevalence : 0.2282
##       Balanced Accuracy : 0.8855
##
##        'Positive' Class : positive
##
```

```
## Setting levels: control = positive, case = negative
## Setting direction: controls > cases
```


ROC Test
SMOTE
Sensitivity
Specificity
0.500 (0.875, 0.904)
1.0
0.8
0.6
0.4
0.2
0.0
1.0
0.5
0.0