Comparison of Sampling methods

CHINDU

5/4/2020

#A basic analysis of different sampling methods that deals with data imbalance

```r
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```r
library(rpart)
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(reshape)
```

```
##
## Attaching package: 'reshape'

## The following object is masked from 'package:dplyr':
##
##     rename
```

```r
library(MLmetrics)
```

```
##
## Attaching package: 'MLmetrics'

## The following objects are masked from 'package:caret':
##
##     MAE, RMSE
```

```
## The following object is masked from 'package:base':
##
##     Recall
```

```r
library(DMwR)
```

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

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

```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(PRROC)
library(ROSE)
```

```
## Loaded ROSE 0.0-3
```

```
##
## Attaching package: 'ROSE'
```

```
## The following object is masked from 'package:PRROC':
##
##     roc.curve
```

```r
library(plyr)
```

```
## ------------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## ------------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

```
## The following object is masked from 'package:DMwR':
##
##     join
```

```
## The following objects are masked from 'package:reshape':
##
##     rename, round_any
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```r
library(DMwR)
```

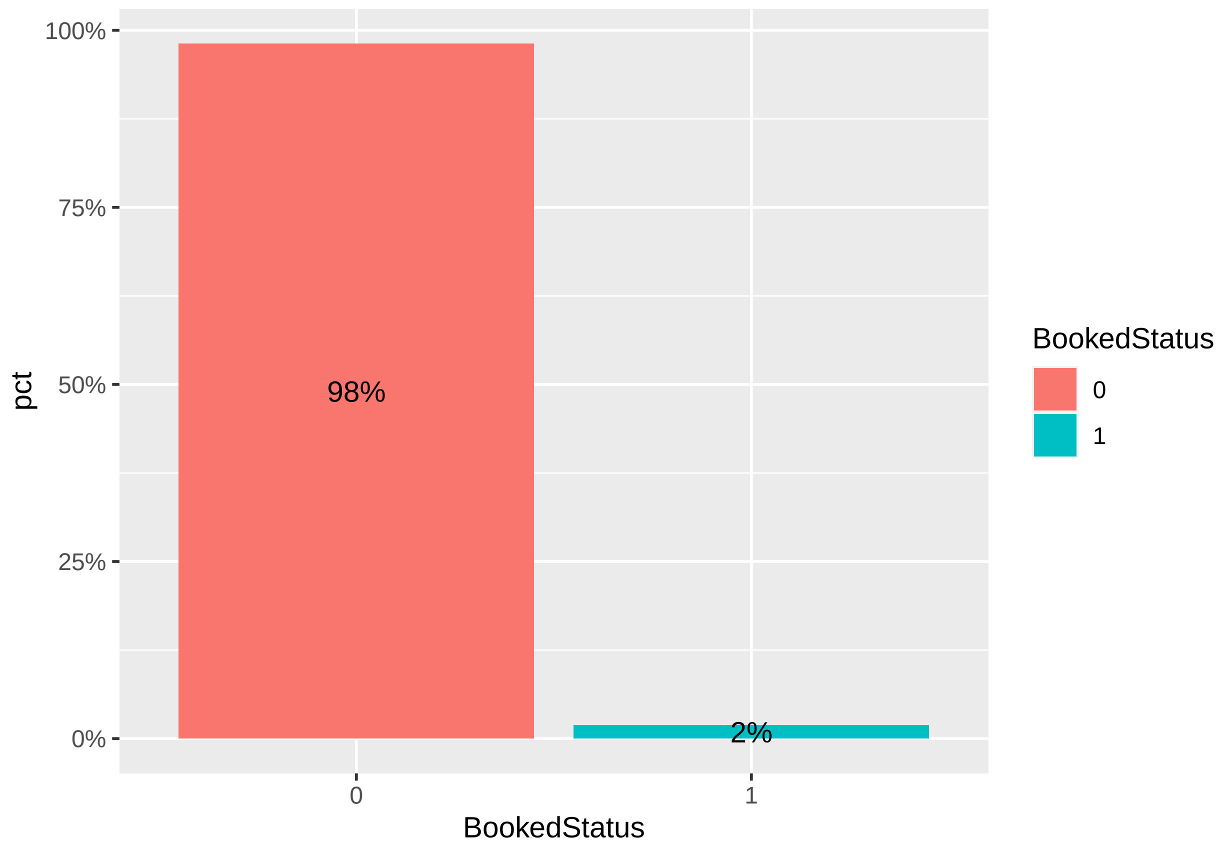
```r
data<- read.csv("EnquiriesClean.csv")
data$BookedStatus<-factor(data$BookedStatus)
data$EnquiryMonth<-factor(data$EnquiryMonth)
data$Hotkey<-factor(data$Hotkey)
data$DepartureMonth<-factor(data$DepartureMonth)
data$TempSent<-factor(data$TempSent)
data$ConversationRCD<-factor(data$ConversationRCD)
str(data)
```

```
## 'data.frame':    115830 obs. of  27 variables:
##  $ X                : int  31 54 57 59 115 213 216 276 291 350 ...
##  $ EnquiryMonth     : Factor w/ 12 levels "1","2","3","4",..: 5 5 11 1 9 9 1 10 1 5 ...
##  $ Enquiry.Day      : Factor w/ 7 levels "Friday","Monday",..: 2 2 4 7 1 2 2 4 6 4 ...
##  $ Web.or.Phone     : Factor w/ 2 levels "PHONE","WEB": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Hotkey           : Factor w/ 2 levels "0","1": 2 1 2 1 2 1 1 2 2 2 ...
##  $ ConversationRCD  : Factor w/ 21 levels "0","1","2","3",..: 2 13 8 4 4 12 1 8 2 2 ...
##  $ TempSent         : Factor w/ 11 levels "0","1","2","3",..: 6 3 4 2 5 7 2 4 2 2 ...
##  $ Holiday.Type     : Factor w/ 4 levels "Fly Drive","Multi Centre",..: 2 3 4 4 2 4 4 1 4 4 ...
##  $ Accom.type       : Factor w/ 4 levels "Apartment","Hotel",..: 2 4 4 2 2 4 4 3 2 2 ...
##  $ Dep.Airport      : Factor w/ 10 levels "Any Airport",..: 9 1 8 10 5 8 6 5 5 7 ...
##  $ DepartureMonth   : Factor w/ 12 levels "1","2","3","4",..: 4 10 5 7 8 3 3 2 8 3 ...
##  $ Lead.Time        : int  48 74 26 27 47 27 62 14 85 44 ...
##  $ Destination      : Factor w/ 10 levels " Disney Area",..: 6 2 4 3 10 7 7 10 9 2 ...
##  $ Duration         : int  14 14 14 14 17 14 14 14 14 10 ...
##  $ Adults           : int  2 4 2 2 2 2 3 3 1 4 ...
##  $ Children         : int  0 2 0 2 2 2 2 0 1 0 ...
##  $ Infants          : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ Transport.Type   : Factor w/ 3 levels "Fully comp car hire",..: 3 3 1 3 1 1 3 1 3 2 ...
##  $ Answered.Q       : Factor w/ 2 levels "NO","YES": 1 1 2 2 2 1 2 2 2 1 ...
##  $ Notes.Completed  : Factor w/ 2 levels "NO","YES": 1 1 1 1 1 1 2 2 1 1 ...
##  $ Title            : Factor w/ 2 levels "F","M": 1 1 2 1 1 1 2 2 1 2 ...
##  $ Enquiry.Comments : Factor w/ 2 levels "NO","YES": 1 1 2 1 2 1 1 1 1 1 ...
##  $ Enquiry.Timecat  : Factor w/ 2 levels "Business_Hour",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Enquiry.Time_class: Factor w/ 3 levels "afternoon","morning",..: 2 2 1 2 1 3 2 1 1 3 ...
##  $ EnquirySeason    : Factor w/ 4 levels "fall","spring",..: 2 2 1 4 1 1 4 1 4 2 ...
##  $ DepartureSeason  : Factor w/ 4 levels "fall","spring",..: 2 1 2 3 3 2 2 4 3 2 ...
##  $ BookedStatus     : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

```r
data$X<-NULL
table(data$BookedStatus)
```

```
##
##      0      1
## 113630   2200
```

```r
DataNew <- data %>% group_by(BookedStatus) %>%
 dplyr::summarize(count = n()) %>%
 mutate(pct = count/sum(count))
ggplot(DataNew, aes(BookedStatus, pct, fill = BookedStatus)) +
  geom_bar(stat='identity') +
  geom_text(aes(label=scales::percent(pct)), position = position_stack(vjust = .5))+
  scale_y_continuous(labels = scales::percent)
```



The data has 98% cases of 0(negative) and 2% of 1 (positive)

Prep Training and Test data.

```r
set.seed(666)
trainDataIndex <- createDataPartition(data$BookedStatus, p=0.7, list = F)  # 70% training data
trainData <- data[trainDataIndex, ]
```

```
testData <- data[-trainDataIndex, ]
table(trainData$BookedStatus)
```

```
##
##     0     1
## 79541  1540
```

Sampling using RUS,ROS,SMOTE

```
down_train1 <- upSample(x = trainData[,-ncol(trainData)],
                        y = trainData$BookedStatus)
down_train1$BookedStatus<- NULL
down_train1<-rename(down_train1,c(Class="BookedStatus"))

down_train2 <- downSample(x = trainData[,-ncol(trainData)],
                          y = trainData$BookedStatus)
down_train2$BookedStatus<- NULL
down_train2<-rename(down_train2,c(Class="BookedStatus"))

down_train3 <- SMOTE(BookedStatus ~ ., trainData,perc.over = 100, perc.under=200,k=5)
```

Create models using the sampled datasets and unsampled data to understand the importance of sampling when data is imbalanced

```
m1 <- rpart(BookedStatus~., data=down_train1, method="class")
m2 <- rpart(BookedStatus~., data=down_train2, method="class")
m3 <- rpart(BookedStatus~., data=down_train3, method="class")
m6 <- rpart(BookedStatus~., data=trainData,   method="class")
```

```
table(testData$BookedStatus)
```

```
##
##     0     1
## 34089   660
```

```
m1 <- rpart(BookedStatus~.,
            method="class", data=down_train1)
m2 <- rpart(BookedStatus~.,
            method="class", data=down_train2)
m3 <- rpart(BookedStatus~.,
            method="class", data=down_train3)
m4 <- rpart(BookedStatus~.,
            method="class", trainData)
```

Accuracy, Specificity, Sensitivity

```
#ROS
pdata <- as.data.frame(predict(m1, newdata = testData, type = "p"))
pdata$my_custom_predicted_class <- ifelse(pdata$`1` > .5, 1, 0)
```

```r
pdata$my_custom_predicted_class<-factor(pdata$my_custom_predicted_class)
testData$BookedStatus<-factor(testData$BookedStatus)
caret::confusionMatrix(data = pdata$my_custom_predicted_class,
                   reference = testData$BookedStatus, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 25556   106
##          1  8533   554
##
##                Accuracy : 0.7514
##                  95% CI : (0.7468, 0.7559)
##     No Information Rate : 0.981
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0811
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.83939
##             Specificity : 0.74968
##          Pos Pred Value : 0.06097
##          Neg Pred Value : 0.99587
##              Prevalence : 0.01899
##          Detection Rate : 0.01594
##    Detection Prevalence : 0.26150
##       Balanced Accuracy : 0.79454
##
##        'Positive' Class : 1
##
```

```r
#RUS
pdata2 <- as.data.frame(predict(m2, newdata = testData, type = "p"))
pdata2$my_custom_predicted_class <- ifelse(pdata2$`1` > .5, 1, 0)
pdata2$my_custom_predicted_class<-factor(pdata2$my_custom_predicted_class)
testData$BookedStatus<-factor(testData$BookedStatus)
caret::confusionMatrix(data = pdata2$my_custom_predicted_class,
                       reference = testData$BookedStatus, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 25554   106
##          1  8535   554
##
##                Accuracy : 0.7513
##                  95% CI : (0.7468, 0.7559)
##     No Information Rate : 0.981
##     P-Value [Acc > NIR] : 1
```

```
##
##                   Kappa : 0.0811
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.83939
##             Specificity : 0.74963
##          Pos Pred Value : 0.06095
##          Neg Pred Value : 0.99587
##              Prevalence : 0.01899
##          Detection Rate : 0.01594
##    Detection Prevalence : 0.26156
##       Balanced Accuracy : 0.79451
##
##        'Positive' Class : 1
##
```

```r
#SMOTE
pdata3 <- as.data.frame(predict(m3, newdata = testData, type = "p"))
pdata3$my_custom_predicted_class <- ifelse(pdata3$`1` > .5, 1, 0)
pdata3$my_custom_predicted_class<-factor(pdata3$my_custom_predicted_class)
testData$BookedStatus<-factor(testData$BookedStatus)
caret::confusionMatrix(data = pdata3$my_custom_predicted_class,
                       reference = testData$BookedStatus, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 24075    91
##          1 10014   569
##
##                Accuracy : 0.7092
##                  95% CI : (0.7044, 0.714)
##     No Information Rate : 0.981
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0679
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.86212
##             Specificity : 0.70624
##          Pos Pred Value : 0.05377
##          Neg Pred Value : 0.99623
##              Prevalence : 0.01899
##          Detection Rate : 0.01637
##    Detection Prevalence : 0.30456
##       Balanced Accuracy : 0.78418
##
##        'Positive' Class : 1
##
```

```r
#No Sampling
pdata4 <- as.data.frame(predict(m4, newdata = testData, type = "p"))
pdata4$my_custom_predicted_class <- ifelse(pdata4$`1` > .5, 1, 0)
pdata4$my_custom_predicted_class<-factor(pdata4$my_custom_predicted_class)
testData$BookedStatus<-factor(testData$BookedStatus)
caret::confusionMatrix(data = pdata4$my_custom_predicted_class,
                       reference = testData$BookedStatus, positive = "1")
```

```
## Warning in confusionMatrix.default(data = pdata4$my_custom_predicted_class, :
## Levels are not in the same order for reference and data. Refactoring data to
## match.
```

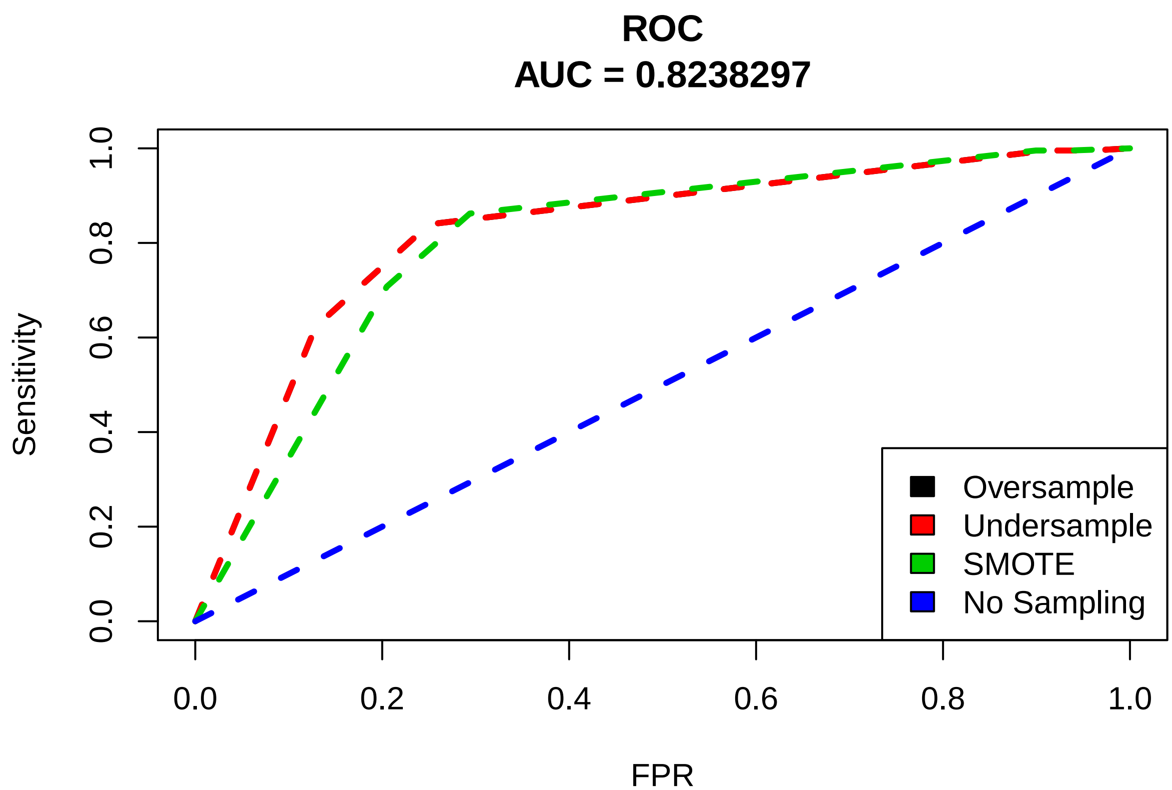
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 34089   660
##          1     0     0
##
##                Accuracy : 0.981
##                  95% CI : (0.9795, 0.9824)
##     No Information Rate : 0.981
##     P-Value [Acc > NIR] : 0.5104
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.00000
##             Specificity : 1.00000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.98101
##              Prevalence : 0.01899
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.00000
##       Balanced Accuracy : 0.50000
##
##        'Positive' Class : 1
##
```

Notice that the no sampling model gives us an accuracy of 98%, but has a sensitivity of 0%. This means that this model failed to predict any positive cases and predicted all cases as negative. This model is basically useless.

Model comparison

```r
fg1 <- pdata$`1`[testData$BookedStatus == 1]
bg1 <- pdata$`1`[testData$BookedStatus == 0]
fg2 <- pdata2$`1`[testData$BookedStatus == 1]
bg2 <- pdata2$`1`[testData$BookedStatus == 0]
fg3 <- pdata3$`1`[testData$BookedStatus == 1]
```

```r
bg3 <- pdata3$`1`[testData$BookedStatus == 0]
fg4 <- pdata4$`1`[testData$BookedStatus == 1]
bg4 <- pdata4$`1`[testData$BookedStatus == 0]


roc1 <- PRROC::roc.curve(scores.class0 = fg1, scores.class1 = bg1, curve = T)
pr1 <- pr.curve(scores.class0 = fg1, scores.class1 = bg1, curve = T)


roc2 <- PRROC::roc.curve(scores.class0 = fg2, scores.class1 = bg2, curve = T)
pr2 <- pr.curve(scores.class0 = fg2, scores.class1 = bg2, curve = T)


roc3 <- PRROC::roc.curve(scores.class0 = fg3, scores.class1 = bg3, curve = T)
pr3 <- pr.curve(scores.class0 = fg3, scores.class1 = bg3, curve = T)

# The ROC will show a straight line with 0.5 AUC as the algorithm was unable to deal with the data imba
roc4 <- PRROC::roc.curve(scores.class0 = fg4, scores.class1 = bg4, curve = T)

#you will not be able to plot a PR curve as the algorithm was unable to handle the data imbalance
#pr4 <- pr.curve(scores.class0 = fg4, scores.class1 = bg4, curve = T)

plot(roc1, col = 1, lty = 2, main = "ROC")
plot(roc2, col = 2, lty = 2, add=TRUE)
plot(roc3, col = 3, lty = 2, add=TRUE)
plot(roc4,col=4 ,lty=2, add=TRUE)

legend(x="bottomright",
       legend= c("Oversample",
                 "Undersample",
                 "SMOTE",
                 "No Sampling"),
       fill = 1:5)
```



Without sampling the model basically was unable to do any logical prediction on the data, this is displayed by the straight line with AUC of 0.5. Once sampling was applied the model was able to perform faily well on the data with oversampling and undersampling perfoming better than SMOTE. Note that the sampling method that should be used depends on many cases example the distribution of data etc..

#A more reliable analysis would be to use Precision and recall and a Precision recall curve. To know more about precision recall curve refer to "Machine Learing Ensemble.pdf"