Blog 3: Class Imbalance - How to Fix

Eric Lehmphul

5/22/2022

```
library(ROSE)
library(skimr)
library(ggpubr)
library(caret)
```

Introduction

Binary classification models can sometime be affected by target variable class imbalance. Some datasets where this is present are customer churn prediction, stroke prediction, and credit card fraud prediction. There are methods to improve modeling results for imbalanced datasets. There are three significant developing techniques that have been applied to balancing imbalanced classes in literature: External, Algorithmic / internal, and Cost-sensitive (Ali et al., 2019). The external approach focuses on rebalancing the data through modifying the dataset rather than adjusting the learning method of the machine learning model (Davarynejad, 2017). Internal approaches adjust or create algorithms to be able to handle imbalanced classification problems, such as modifying the decision threshold or creating a new cost function to add bias toward the minority class (Davarynejad, 2017). The last technique combines both the external and internal approaches (Davarynejad, 2017).

Load Data

The dataset being used for this blog was found on kaggle.com at https://www.kaggle.com/datasets/shashwatwork/cerebral-stroke-predictionimbalaced-dataset.

```
imbalanced.data <- read.csv("https://raw.githubusercontent.com/SaneSky109/DATA621/main/Blogs/imbalanced
imbalanced.data <- imbalanced.data[,-c(1,2,7)]

head(imbalanced.data)
```

```
##   age hypertension heart_disease ever_married Residence_type avg_glucose_level
## 1   3            0             0           No          Rural             95.12
## 2  58            1             0          Yes          Urban             87.96
## 3   8            0             0           No          Urban            110.89
## 4  70            0             0          Yes          Rural             69.04
## 5  14            0             0           No          Rural            161.28
## 6  47            0             0          Yes          Urban            210.95
##    bmi  smoking_status stroke
## 1 18.0                      0
## 2 39.2    never smoked      0
```

```
## 3 17.6                      0
## 4 35.9 formerly smoked      0
## 5 19.1                      0
## 6 50.1                      0
```

```
imbalanced.data$stroke <- gsub("0","no", imbalanced.data$stroke)
imbalanced.data$stroke <- gsub("1","yes", imbalanced.data$stroke)

imbalanced.data$hypertension <- as.factor(imbalanced.data$hypertension)
imbalanced.data$heart_disease <- as.factor(imbalanced.data$heart_disease)
imbalanced.data$ever_married <- as.factor(imbalanced.data$ever_married)

imbalanced.data$Residence_type <- as.factor(imbalanced.data$Residence_type)
imbalanced.data$smoking_status <- as.factor(imbalanced.data$smoking_status)
imbalanced.data$stroke <- as.factor(imbalanced.data$stroke)
```

```
table(imbalanced.data$stroke)
```

```
##
##    no   yes
## 42617   783
```

Create Train and Test Set

The training set will be used to train the models and create the under and over sampled datasets. The testing set will be used for model evaluation.

```
set.seed(15)

trainIndex <- createDataPartition(imbalanced.data$stroke, p = .7,
                                  list = FALSE,
                                  times = 1)

train <- imbalanced.data[ trainIndex,]
test <- imbalanced.data[-trainIndex,]
```

Using External Data Balancing techniques to create balanced datasets

The ROSE package provides the ovun.sample function to perform data balancing on the data.

```
over <- ovun.sample(stroke~., data = train, method = "over")$data

under <- ovun.sample(stroke~., data=train, method = "under")$data
```

Compare Stroke class distributions

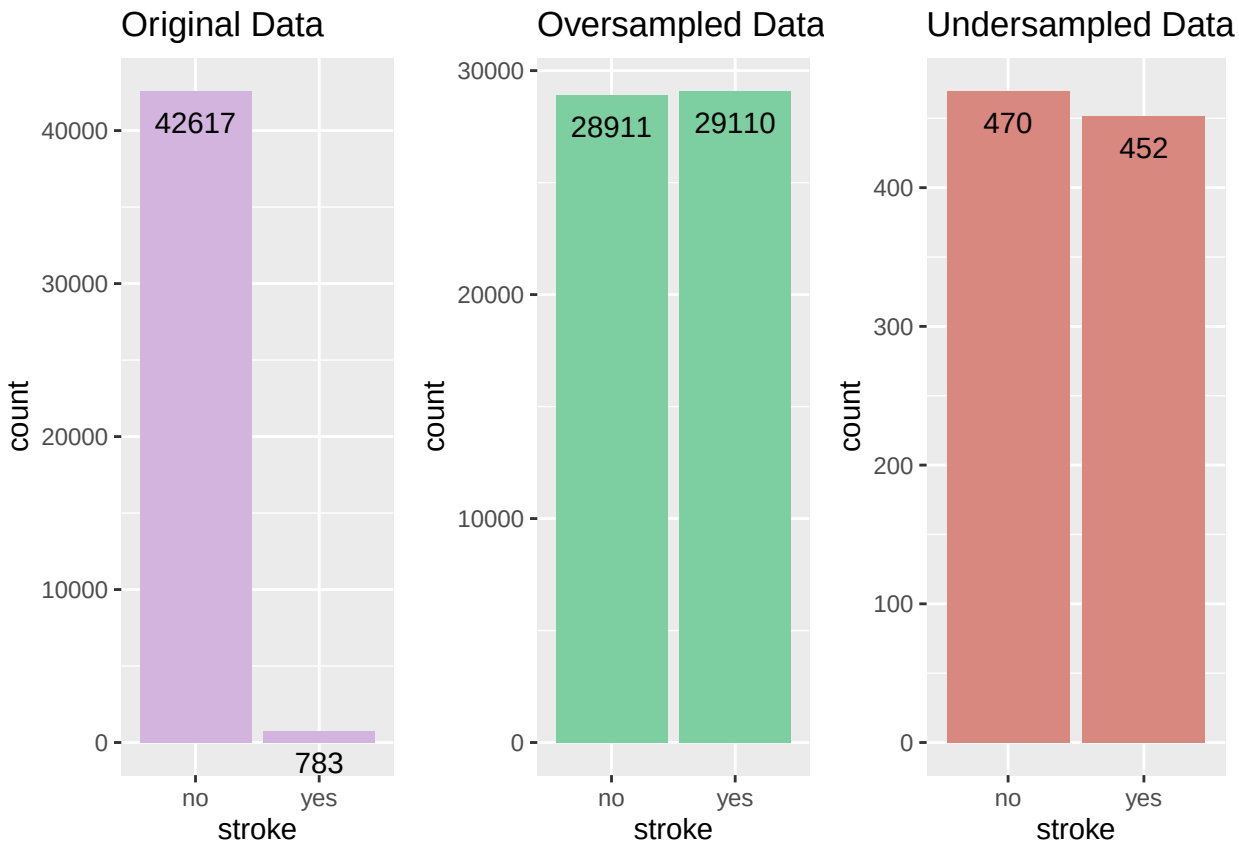
```
p1 <- imbalanced.data %>%
  ggplot(aes(x=stroke)) + geom_bar(fill = "#D2B4DE") +
  geom_text(stat='count', aes(label=..count..), vjust=2) +
  ggtitle("Original Data")

p2 <- over %>%
  ggplot(aes(x=stroke)) + geom_bar(fill = "#7DCEA0") +
  geom_text(stat='count', aes(label=..count..), vjust=2) +
  ggtitle("Oversampled Data")

p3 <- under %>%
  ggplot(aes(x=stroke)) + geom_bar(fill = "#D98880") +
  geom_text(stat='count', aes(label=..count..), vjust=2) +
  ggtitle("Undersampled Data")



ggarrange(p1,p2,p3,
          ncol = 3, nrow = 1)
```



Create Logistic Regression Models for each dataset

Imbalanced Data

This first model is the results for training a logistic regression using the unbalanced data. Notice that this model was terrible at identifying if someone experienced a stroke, even though it have a high accuracy.

```
m1 <- glm(stroke ~ ., family = "binomial", data = train)
```

```
pred.1.raw <- predict(m1, type = "response", newdata = test)
pred.1 <- as.factor(ifelse(pred.1.raw < .5, "no", "yes"))

cm <- table(test$stroke, pred.1)
yes <- c(0,0)

cm <- cbind(cm,yes)
cm1 <- confusionMatrix(cm, positive="yes")

cm1
```

```
## Confusion Matrix and Statistics
##
##        no   yes
## no  12384     0
## yes   191     0
##
##                Accuracy : 0.9848
##                  95% CI : (0.9825, 0.9869)
##     No Information Rate : 1
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity :      NA
##             Specificity : 0.98481
##          Pos Pred Value :      NA
##          Neg Pred Value :      NA
##              Prevalence : 0.00000
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.01519
##       Balanced Accuracy :      NA
##
##        'Positive' Class : yes
##
```

Oversampled Data

The oversampled data was used to fit the data. It was able to predict the minority class with a high degree of precision.

```r
m2 <- glm(stroke ~ ., family = "binomial", data = over)
```

```r
pred.2.raw <- predict(m2, type = "response", newdata = test)
pred.2 <- as.factor(ifelse(pred.2.raw < .5, "no", "yes"))

cm <- table(test$stroke, pred.2)

cm2 <- confusionMatrix(cm, positive="yes")

cm2
```

```
## Confusion Matrix and Statistics
##
##      pred.2
##         no  yes
##   no  9320 3064
##   yes   31  160
##
##                Accuracy : 0.7539
##                  95% CI : (0.7463, 0.7614)
##     No Information Rate : 0.7436
##     P-Value [Acc > NIR] : 0.004214
##
##                   Kappa : 0.0669
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.04963
##             Specificity : 0.99668
##          Pos Pred Value : 0.83770
##          Neg Pred Value : 0.75258
##              Prevalence : 0.25638
##          Detection Rate : 0.01272
##    Detection Prevalence : 0.01519
##       Balanced Accuracy : 0.52316
##
##        'Positive' Class : yes
##
```

Undersampled Data

The undersampled data was used to fit the data. It was able to predict the minority class with a high degree of precision.

```r
m3 <- glm(stroke ~ ., family = "binomial", data = under)
```

```r
pred.3.raw <- predict(m3, type = "response", newdata = test)
pred.3 <- as.factor(ifelse(pred.3.raw < .5, "no", "yes"))

cm <- table(test$stroke, pred.3)
```

```
cm3 <- confusionMatrix(cm, positive="yes")

cm3
```

```
## Confusion Matrix and Statistics
##
##      pred.3
##         no  yes
##   no  9417 2967
##   yes   30  161
##
##                Accuracy : 0.7617
##                  95% CI : (0.7541, 0.7691)
##     No Information Rate : 0.7513
##     P-Value [Acc > NIR] : 0.003435
##
##                   Kappa : 0.0704
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.05147
##             Specificity : 0.99682
##          Pos Pred Value : 0.84293
##          Neg Pred Value : 0.76042
##              Prevalence : 0.24875
##          Detection Rate : 0.01280
##    Detection Prevalence : 0.01519
##       Balanced Accuracy : 0.52415
##
##        'Positive' Class : yes
##
```

Evaluate Results

It is clear based on these results that imbalanced data needs to be rebalanced in order to sucessfully predict the rare event (minority class). Both oversampling and undersampling had much higher precision when determining the minority class.

```
get_metrics <- function(confusion.matrix){
by.class <- round(confusion.matrix$byClass, 3)
metrics <- data.frame(t(by.class))

Accuracy <- round(confusion.matrix$overall[1], 3)

classification.metrics <- cbind(Accuracy, metrics)

return(classification.metrics)
}
```

```
metric1 <- get_metrics(cm1)
metric2 <- get_metrics(cm2)
```

```
metric3 <- get_metrics(cm3)

model.metrics <- rbind(metric1, metric2, metric3)
rownames(model.metrics) <- c("Unbalanced", "Oversampled", "Undersampled")

model.metrics <- t(model.metrics)
kableExtra::kable(model.metrics)
```

	Unbalanced	Oversampled	Undersampled
Accuracy	0.985	0.754	0.762
Sensitivity	NA	0.050	0.051
Specificity	0.985	0.997	0.997
Pos.Pred.Value	NA	0.838	0.843
Neg.Pred.Value	NA	0.753	0.760
Precision	0.000	0.838	0.843
Recall	NA	0.050	0.051
F1	NA	0.094	0.097
Prevalence	0.000	0.256	0.249
Detection.Rate	0.000	0.013	0.013
Detection.Prevalence	0.015	0.015	0.015
Balanced.Accuracy	NA	0.523	0.524

Conclusion

It is important to rebalance imbalanced classes when performing a classification task. The results prove that the rebalanced data will predict the minority class with much greater precision than to be left unbalanced.