Chicago Crime

Xin Guan, Vera Hudak, Yuqi Zhang

2023-11-24

```r
# Packages required
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

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

```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(magrittr)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats 1.0.0     v stringr 1.5.0
## v ggplot2 3.4.3     v tibble  3.2.1
## v purrr   1.0.2     v tidyr   1.3.0
## v readr   2.1.4

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::extract()   masks magrittr::extract()
## x dplyr::filter()    masks stats::filter()
## x dplyr::lag()       masks stats::lag()
## x purrr::set_names() masks magrittr::set_names()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(ggplot2)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```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(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(ROSE)
```

```
## Loaded ROSE 0.0-4
```

```r
library(knitr)
```

Data Processing

```r
# Read data
Crime_data <- read.csv("OriginalData.csv")
```

Processing variable 'Date'

```r
# We want to firstly convert the format of date for the `lubridate` package, then extract information f
Crime_data$newDate <- mdy_hms(Crime_data$Date)
Crime_data$Hour <- hour(Crime_data$newDate)
Crime_data$WeekDay <- weekdays(Crime_data$newDate)
Crime_data$DayOfMonth <- day(Crime_data$newDate)
Crime_data$DayOfYear <- yday(Crime_data$newDate)
Crime_data$Month <- month(Crime_data$newDate, label = TRUE, abbr = FALSE)
Crime_data$Time <- hour(Crime_data$newDate)*100 + minute(Crime_data$newDate) #This format of `Time` var
Crime_data$TimeOfDay <- cut(
  hour(Crime_data$newDate),
  breaks= c(-Inf, 5, 12, 17, 20, Inf),
  labels = c("Night", "Early Morning", "Morning", "Afternoon", "Evening"),
  include.lowest = TRUE
)
```

Processing missing data

```r
colSums(is.na(Crime_data))
```

```
##                   ID          Case.Number                 Date
##                    0                    0                    0
##                Block                 IUCR         Primary.Type
##                    0                    0                    0
##          Description Location.Description               Arrest
##                    0                    0                    0
##             Domestic                 Beat             District
##                    0                    0                    0
##                 Ward       Community.Area             FBI.Code
##                   15                    0                    0
##         X.Coordinate         Y.Coordinate                 Year
##                 2205                 2205                    0
##           Updated.On             Latitude            Longitude
##                    0                 2205                 2205
##             Location              newDate                 Hour
##                    0                    0                    0
##              WeekDay           DayOfMonth            DayOfYear
##                    0                    0                    0
##                Month                 Time            TimeOfDay
##                    0                    0                    0
```

We can see that the number of missing values for the variables `X.Coordinate`, `Y.Coordinate`, `Latitude`, and `Longtitude` are exactly the same, we can deduce that the coordinates are calculated from the latitude and longitude, so we don't need to include both pairs of location information. Also since the number of missing value is small compare to the number of data size, we will just eliminate the rows with missing values.

```r
Crime_data %<>%  na.omit()
```

Expalnatory Data Analysis

Feature Selection

Our first task is to predict whether arrest or not given time, location, and the crime type.

```
Binary_pred_df <- Crime_data %>% select(c("ID","X.Coordinate", "Y.Coordinate","Hour","Time","WeekDay","
#???location variables
```

Define Cross-validation

```
# Define a 5-fold cross validation
ctrl <- trainControl(method = "cv", number = 5, summaryFunction = twoClassSummary, classProbs = TRUE)
```

Imbalance Data Experiments with a Baseline Model

```
# Separate positive and negative classes
true_class <- Binary_pred_df %>% filter(Arrest == "true")
false_class <- Binary_pred_df %>% filter(Arrest == "false")

# Set the desired ratio of positive to negative samples
desired_positive_ratio <- 0.05

# Calculate the number of positive samples needed for downsampling
num_true_samples <- 0.05*nrow(false_class) / (1 - desired_positive_ratio)

# Randomly sample positive samples
true_class_downsampled <- true_class %>% sample_n(num_true_samples, replace = FALSE, seed = 42)

# Combine positive and downsampled negative samples
downsampled_dataset <- bind_rows(false_class, true_class_downsampled)

# Shuffle the dataset to mix positive and negative samples
set.seed(42)
downsampled_dataset <- downsampled_dataset[sample(nrow(downsampled_dataset)), ]

sum(downsampled_dataset[,"Arrest"]=="true")/dim(df)[1]
```

```
## numeric(0)
```

```
#use 80% of dataset as training set and 20% as test set
variables_to_convert <- c("Arrest", "Primary.Type", "WeekDay")
imbalanced_train <- downsampled_dataset %>% dplyr::sample_frac(0.80)
imbalanced_train[, variables_to_convert] <- lapply(imbalanced_train[, variables_to_convert], as.factor)
imbalanced_test  <- dplyr::anti_join(downsampled_dataset, imbalanced_train, by = 'ID')
imbalanced_test[, variables_to_convert] <- lapply(imbalanced_test[, variables_to_convert], as.factor)
imbalanced_train %<>% select(-"ID")
imbalanced_test %<>% select(-"ID")
```

```r
base_m <- train(Arrest ~ X.Coordinate + Y.Coordinate + DayOfYear + Hour +
    Primary.Type, data = imbalanced_train, method = "glm", family = "binomial", trControl = ctrl, metric
```

```r
pred <- predict(base_m, imbalanced_test[,c("X.Coordinate", "Y.Coordinate", "DayOfYear", "Hour", "Primar
```

```r
binary_predictions <- ifelse(pred[,2] >= 0.5, "true", "false")
# create confusion matrix
confusionMatrix(as.factor(binary_predictions), imbalanced_test$Arrest,
                mode = "everything",
                positive="true")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction false  true
##      false 40639  1467
##      true     15   629
##
##                Accuracy : 0.9653
##                  95% CI : (0.9636, 0.967)
##     No Information Rate : 0.951
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4464
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.30010
##             Specificity : 0.99963
##          Pos Pred Value : 0.97671
##          Neg Pred Value : 0.96516
##               Precision : 0.97671
##                  Recall : 0.30010
##                      F1 : 0.45912
##              Prevalence : 0.04903
##          Detection Rate : 0.01471
##    Detection Prevalence : 0.01506
##       Balanced Accuracy : 0.64986
##
##        'Positive' Class : true
##
```

```r
#over sampling
train_balanced_over <- ovun.sample(Arrest ~ ., data = imbalanced_train, method = "over")$data
train_balanced_over$Arrest <- as.factor(train_balanced_over$Arrest)
train_balanced_under <- ovun.sample(Arrest ~ ., data = imbalanced_train, method = "under")$data
train_balanced_under$Arrest <- as.factor(train_balanced_under$Arrest)
train.rose <- ROSE(Arrest ~ ., data = imbalanced_train)$data
train.rose$Arrest <- as.factor(train.rose$Arrest)
table(train.rose$Arrest)
```

```
##
```

```
## false  true
## 85639 85359
```

```r
table(train_balanced_over$Arrest)
```

```
##
##  false   true
## 162407 162774
```

```r
table(train_balanced_under$Arrest)
```

```
##
## false  true
##  8594  8591
```

```r
base_m_under <- train(Arrest ~ X.Coordinate + Y.Coordinate + DayOfYear + Hour + Primary.Type, data = tra
base_m_over <- train(Arrest ~ X.Coordinate + Y.Coordinate + DayOfYear + Hour + Primary.Type, data = trai
base_m_rose <- train(Arrest ~ X.Coordinate + Y.Coordinate + DayOfYear + Hour + Primary.Type, data = trai
```

```r
pred_under <- predict(base_m_under, imbalanced_test[,c("X.Coordinate", "Y.Coordinate", "DayOfYear", "Hou
pred_over <- predict(base_m_over, imbalanced_test[,c("X.Coordinate", "Y.Coordinate", "DayOfYear", "Hour
pred_rose <- predict(base_m_rose, imbalanced_test[,c("X.Coordinate", "Y.Coordinate", "DayOfYear", "Hour
```

```r
# Create a ROC curve object for each prediction
roc_curve <- roc(imbalanced_test$Arrest, pred[,2])
```

```
## Setting levels: control = false, case = true

## Setting direction: controls < cases
```

```r
roc_curve_under <- roc(imbalanced_test$Arrest, pred_under[,2])
```

```
## Setting levels: control = false, case = true
## Setting direction: controls < cases
```

```r
roc_curve_over <- roc(imbalanced_test$Arrest, pred_over[,2])
```

```
## Setting levels: control = false, case = true
## Setting direction: controls < cases
```

```r
roc_curve_rose <- roc(imbalanced_test$Arrest, pred_rose[,2])
```

```
## Setting levels: control = false, case = true
## Setting direction: controls < cases
```

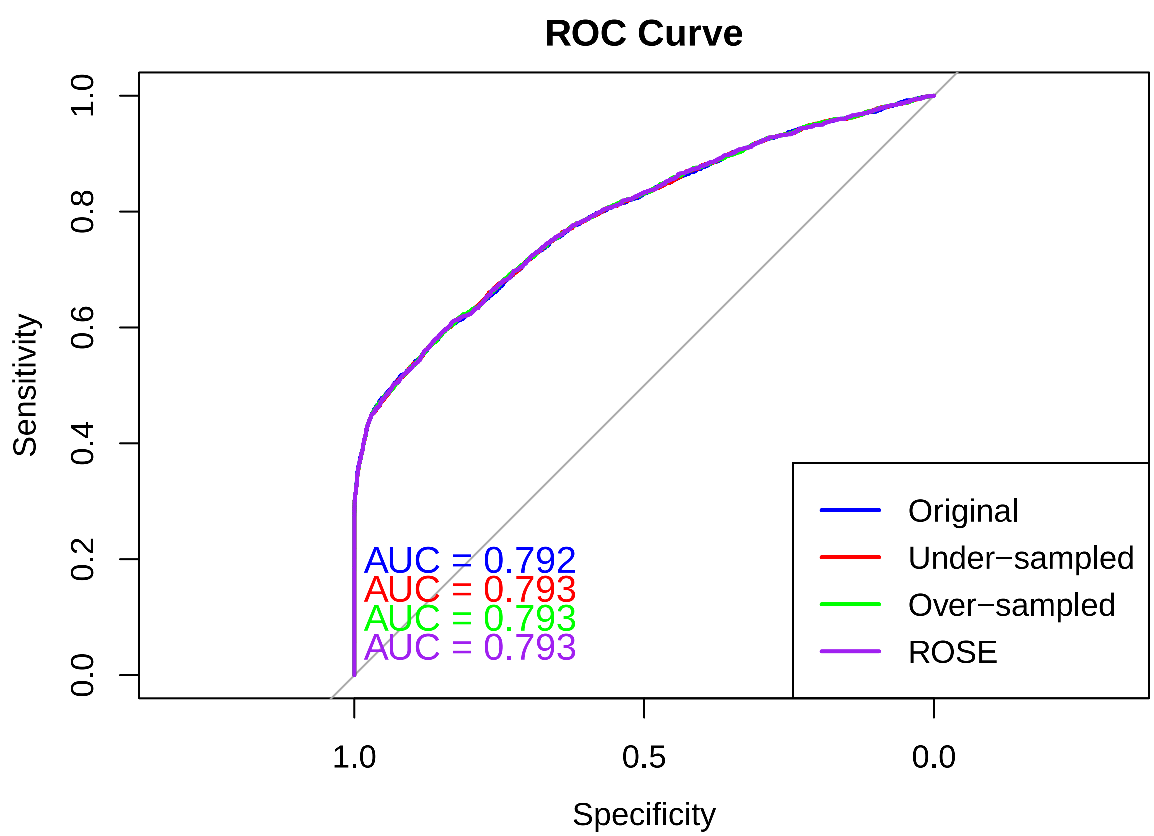
```r
# Plot the first ROC curve
plot(roc_curve, col = "blue", lwd = 2, main = "ROC Curve")

# Add AUC to the plot for the first curve
auc_value <- auc(roc_curve)
text(0.8, 0.2, paste("AUC =", round(auc_value, 3)), col = "blue", cex = 1.2)

# Add the other ROC curves to the plot
lines(roc_curve_under, col = "red", lwd = 2)
lines(roc_curve_over, col = "green", lwd = 2)
lines(roc_curve_rose, col = "purple", lwd = 2)

# Add AUC values for the other curves
text(0.8, 0.15, paste("AUC =", round(auc(roc_curve_under), 3)), col = "red", cex = 1.2)
text(0.8, 0.1, paste("AUC =", round(auc(roc_curve_over), 3)), col = "green", cex = 1.2)
text(0.8, 0.05, paste("AUC =", round(auc(roc_curve_rose), 3)), col = "purple", cex = 1.2)

legend("bottomright", legend = c("Original", "Under-sampled", "Over-sampled", "ROSE"),
       col = c("blue", "red", "green", "purple"), lty = 1, lwd = 2)
```



Given the current highly imbalanced nature of the data, the effectiveness of comparing models using the AUC curve diminishes. It's advisable to explore alternative evaluation metrics. The provided code below computes the F-β score, an generalization of the F-1 score, the F-1 score is included in the output of `confusionMatrix()` function. The parameter $\beta \geq 0$ is the weight assigned to recall, in the case, we should choose $\beta > 1$, so we prioritize the recall.

F-beta Score

```
f_beta_score <- function(X, y, model, beta, threshold=0.5) {
  predictions <- predict(model, X, type = "prob")[,2]
  predicted_classes <- ifelse(predictions > threshold, "true", "false")
  predicted_classes <- as.factor(predicted_classes)
  pr <- confusionMatrix(data = predicted_classes, reference = y,
                        mode = "everything", positive="true")

  precision <- as.numeric(pr$byClass["Precision"])
  recall <- as.numeric(pr$byClass["Recall"])

  f_beta <- ((1 + beta^2) * precision * recall) / (beta^2 * precision + recall)
  return(f_beta)
}
```

```
f2 <- f_beta_score(X=imbalanced_test[,c("X.Coordinate", "Y.Coordinate", "DayOfYear", "Hour", "Primary.Ty
f2_under <- f_beta_score(X=imbalanced_test[,c("X.Coordinate", "Y.Coordinate", "DayOfYear", "Hour", "Prin
f2_over <- f_beta_score(X=imbalanced_test[,c("X.Coordinate", "Y.Coordinate", "DayOfYear", "Hour", "Prima
f2_rose <- f_beta_score(X=imbalanced_test[,c("X.Coordinate", "Y.Coordinate", "DayOfYear", "Hour", "Prima
print(c(f2,f2_under,f2_over,f2_rose))
```

```
## [1] 0.3083101 0.5238314 0.5348198 0.5286002
```

Train Test Split

```
#use 80% of dataset as training set and 20% as test set
variables_to_convert <- c("Arrest", "Primary.Type", "WeekDay")
train <- Binary_pred_df %>% dplyr::sample_frac(0.80)
train[, variables_to_convert] <- lapply(train[, variables_to_convert], as.factor)
test  <- dplyr::anti_join(Binary_pred_df, train, by = 'ID')
test[, variables_to_convert] <- lapply(test[, variables_to_convert], as.factor)
train %<>% select(-"ID")
test %<>% select(-"ID")
```