Untitled

2025-11-21

En este script se realiza un modelo de clasificación binario con caret (cv=10) Y CON loocv.

Se utiliza la base de datos desbalanceada y se juega con el cutoff para hacer frente al desbalanceo.

Caret

Trabajaremos a partir de datatrain, donde tenemos la respuesta Exited para todas las filas

```r
load("~/GitHub/Mineria/DATA/dataaaaaaaaaaaaaaa.RData")
datatrain <- data_reducida_plus[1:7000, !(names(data_reducida_plus) %in% "group")]
datatest<-data_reducida_plus[7001:10000,!(names(data_reducida_plus) %in% "group")]
```

Con Caret se genera el modelo glm de respuesta binaria

```r
library(caret)
```

```
## Cargando paquete requerido: ggplot2

## Cargando paquete requerido: lattice
```

```r
set.seed(123)

datatrain$Exited <- factor(datatrain$Exited, levels=c(0,1), labels=c("neg","pos"))

# Control de cross-validation
train_ctrl <- trainControl(
  method = "cv",
  number = 10,
  classProbs = TRUE,
  summaryFunction = twoClassSummary,
  savePredictions = TRUE
)

# Entrenem el model logístic amb CV
fit_cv <- train(
  Exited ~ .,
  data = datatrain,
  method = "glm",
  family = binomial,
  trControl = train_ctrl,
  metric = "ROC"
```

```r
)

# Obtenim les probabilitats out-of-fold
prob_cv <- fit_cv$pred$pos
y_cv    <- fit_cv$pred$obs
```

Ara evaluem mitjançant els kpi explicats a classe els resultats del model. Es proporciona una funció que permet obtenir les mètriques modificant el llindar o cutoff

```r
performance_metrics <- function(y_true, prob, cutoff=0.5, dec=3){

  # Convertim y_true a neg/pos
  y_true <- factor(y_true, levels=c(0,1), labels=c("neg","pos"))

  # Predicció com a factor amb nivells fixos
  pred <- factor(ifelse(prob > cutoff, "pos", "neg"),
                 levels=c("neg","pos"))

  # Calculem la taula manualment garantint files/columnes
  cm <- matrix(0, nrow=2, ncol=2,
               dimnames=list(Predicted=c("neg","pos"),
                             Actual=c("neg","pos")))

  tab <- table(Predicted=pred, Actual=y_true)

  cm[rownames(tab), colnames(tab)] <- tab

  TN <- cm["neg","neg"]
  FP <- cm["pos","neg"]
  FN <- cm["neg","pos"]
  TP <- cm["pos","pos"]

  Sens <- TP/(TP+FN)
  Spec <- TN/(TN+FP)
  PPV  <- TP/(TP+FP)
  NPV  <- TN/(TN+FN)
  PLR  <- PPV/(1-NPV)
  NLR  <- NPV/(1-PPV)
  Acc  <- (TP+TN) / sum(cm)

  list(
    Sensitivity = round(Sens, dec),
    Specificity = round(Spec, dec),
    PPV = round(PPV, dec),
    NPV = round(NPV, dec),
    PLR = round(PLR, dec),
    NLR = round(NLR, dec),
    Accuracy = round(Acc, dec),
    ConfusionMatrix = cm
  )
}
#Per tenir les probabilitats en l'ordre de la variable sortida:
prob_alineada <- numeric(nrow(datatrain))
```

```r
prob_alineada[fit_cv$pred$rowIndex] <- fit_cv$pred$pos

#tornem a transformar a numèrica la variable sortida
y_numeric <- ifelse(datatrain$Exited == "pos", 1, 0)
```

Evaluem els KPI per a diferents cutoff per observar les variacions que es donen

```r
cutpoints <- seq(0.1, 0.9, by=0.05)
results   <- data.frame()

results <- data.frame(
  Cutpoint    = numeric(),
  Sensitivity = numeric(),
  Specificity = numeric(),
  PPV         = numeric(),
  NPV         = numeric(),
  PLR         = numeric(),
  NLR         = numeric(),
  Accuracy    = numeric(),
  F1          = numeric()
)


f1_score_cm <- function(cm){
  TP <- cm["pos","pos"]
  FP <- cm["pos","neg"]
  FN <- cm["neg","pos"]

  precision <- TP / (TP + FP)
  recall    <- TP / (TP + FN)

  f1 <- 2 * (precision * recall) / (precision + recall)
  return(round(f1, 3))
}


for(c in cutpoints){
  m <- performance_metrics(y_numeric, prob_alineada, cutoff=c)

  results <- rbind(results, data.frame(
    Cutpoint    = c,
    Sensitivity = m$Sensitivity,
    Specificity = m$Specificity,
    PPV         = m$PPV,
    NPV         = m$NPV,
    PLR         = m$PLR,
    NLR         = m$NLR,
    Accuracy    = m$Accuracy,
    F1          = f1_score_cm(m$ConfusionMatrix)
  ))
}

results
```

```
##    Cutpoint Sensitivity Specificity   PPV   NPV   PLR    NLR Accuracy    F1
## 1      0.10       0.915       0.321 0.261 0.935 4.037  1.265    0.444 0.406
## 2      0.15       0.788       0.523 0.301 0.904 3.142  1.294    0.578 0.436
## 3      0.20       0.681       0.670 0.350 0.889 3.165  1.369    0.672 0.463
## 4      0.25       0.569       0.779 0.402 0.874 3.183  1.461    0.735 0.471
## 5      0.30       0.457       0.853 0.448 0.857 3.141  1.554    0.771 0.452
## 6      0.35       0.363       0.903 0.493 0.844 3.171  1.667    0.791 0.419
## 7      0.40       0.277       0.937 0.537 0.832 3.201  1.797    0.801 0.366
## 8      0.45       0.212       0.960 0.581 0.823 3.292  1.966    0.805 0.311
## 9      0.50       0.174       0.973 0.630 0.818 3.471  2.212    0.808 0.272
## 10     0.55       0.132       0.981 0.642 0.812 3.420  2.270    0.805 0.220
## 11     0.60       0.102       0.987 0.670 0.808 3.487  2.446    0.804 0.177
## 12     0.65       0.084       0.992 0.726 0.806 3.736  2.942    0.804 0.151
## 13     0.70       0.062       0.996 0.789 0.802 3.997  3.812    0.802 0.115
## 14     0.75       0.046       0.998 0.835 0.800 4.178  4.862    0.800 0.086
## 15     0.80       0.028       0.999 0.930 0.797 4.590 11.428    0.798 0.054
## 16     0.85       0.013       1.000 0.905 0.795 4.413  8.347    0.795 0.026
## 17     0.90       0.003       1.000 0.833 0.793 4.033  4.760    0.793 0.007
```

```
results
```

```
##    Cutpoint Sensitivity Specificity   PPV   NPV   PLR    NLR Accuracy    F1
## 1      0.10       0.915       0.321 0.261 0.935 4.037  1.265    0.444 0.406
## 2      0.15       0.788       0.523 0.301 0.904 3.142  1.294    0.578 0.436
## 3      0.20       0.681       0.670 0.350 0.889 3.165  1.369    0.672 0.463
## 4      0.25       0.569       0.779 0.402 0.874 3.183  1.461    0.735 0.471
## 5      0.30       0.457       0.853 0.448 0.857 3.141  1.554    0.771 0.452
## 6      0.35       0.363       0.903 0.493 0.844 3.171  1.667    0.791 0.419
## 7      0.40       0.277       0.937 0.537 0.832 3.201  1.797    0.801 0.366
## 8      0.45       0.212       0.960 0.581 0.823 3.292  1.966    0.805 0.311
## 9      0.50       0.174       0.973 0.630 0.818 3.471  2.212    0.808 0.272
## 10     0.55       0.132       0.981 0.642 0.812 3.420  2.270    0.805 0.220
## 11     0.60       0.102       0.987 0.670 0.808 3.487  2.446    0.804 0.177
## 12     0.65       0.084       0.992 0.726 0.806 3.736  2.942    0.804 0.151
## 13     0.70       0.062       0.996 0.789 0.802 3.997  3.812    0.802 0.115
## 14     0.75       0.046       0.998 0.835 0.800 4.178  4.862    0.800 0.086
## 15     0.80       0.028       0.999 0.930 0.797 4.590 11.428    0.798 0.054
## 16     0.85       0.013       1.000 0.905 0.795 4.413  8.347    0.795 0.026
## 17     0.90       0.003       1.000 0.833 0.793 4.033  4.760    0.793 0.007
```

#LOOCV (leave One Out Cross Validation)

```
library(caret)

set.seed(123)
datatrain <- data_reducida_plus[1:7000, !(names(data_reducida_plus) %in% "group")]
datatest<-data_reducida_plus[7001:10000,!(names(data_reducida_plus) %in% "group")]
#  Convertimos Exited a factor neg/pos
datatrain$Exited <- factor(datatrain$Exited, levels=c(0,1), labels=c("neg","pos"))

#  Control LOOCV
ctrl_loocv <- trainControl(
  method = "LOOCV",
```

```r
  classProbs = TRUE,
  summaryFunction = twoClassSummary,
  savePredictions = TRUE   # importante para obtener predicciones out-of-fold
)

# Entrenamos modelo logístico con LOOCV
fit_loocv <- train(
  Exited ~ .,
  data = datatrain,
  method = "glm",
  family = "binomial",
  trControl = ctrl_loocv,
  metric = "ROC"
)

#  Probabilidades out-of-fold en el orden original
prob_loocv <- numeric(nrow(datatrain))
prob_loocv[fit_loocv$pred$rowIndex] <- fit_loocv$pred$pos

#  Convertimos Exited a 0/1 para métricas
y_numeric <- ifelse(datatrain$Exited == "pos", 1, 0)

cutpoints <- seq(0.1, 0.9, by=0.05)
results   <- data.frame()

results <- data.frame(
  Cutpoint    = numeric(),
  Sensitivity = numeric(),
  Specificity = numeric(),
  PPV         = numeric(),
  NPV         = numeric(),
  PLR         = numeric(),
  NLR         = numeric(),
  Accuracy    = numeric(),
  F1          = numeric()
)


f1_score_cm <- function(cm){
  TP <- cm["pos","pos"]
  FP <- cm["pos","neg"]
  FN <- cm["neg","pos"]

  precision <- TP / (TP + FP)
  recall    <- TP / (TP + FN)

  f1 <- 2 * (precision * recall) / (precision + recall)
  return(round(f1, 3))
}


for(c in cutpoints){
  m <- performance_metrics(y_numeric, prob_alineada, cutoff=c)
```

```r
  results <- rbind(results, data.frame(
    Cutpoint    = c,
    Sensitivity = m$Sensitivity,
    Specificity = m$Specificity,
    PPV         = m$PPV,
    NPV         = m$NPV,
    PLR         = m$PLR,
    NLR         = m$NLR,
    Accuracy    = m$Accuracy,
    F1          = f1_score_cm(m$ConfusionMatrix)
  ))
}

results
```

```
##    Cutpoint Sensitivity Specificity   PPV   NPV   PLR    NLR Accuracy    F1
## 1      0.10       0.915       0.321 0.261 0.935 4.037  1.265    0.444 0.406
## 2      0.15       0.788       0.523 0.301 0.904 3.142  1.294    0.578 0.436
## 3      0.20       0.681       0.670 0.350 0.889 3.165  1.369    0.672 0.463
## 4      0.25       0.569       0.779 0.402 0.874 3.183  1.461    0.735 0.471
## 5      0.30       0.457       0.853 0.448 0.857 3.141  1.554    0.771 0.452
## 6      0.35       0.363       0.903 0.493 0.844 3.171  1.667    0.791 0.419
## 7      0.40       0.277       0.937 0.537 0.832 3.201  1.797    0.801 0.366
## 8      0.45       0.212       0.960 0.581 0.823 3.292  1.966    0.805 0.311
## 9      0.50       0.174       0.973 0.630 0.818 3.471  2.212    0.808 0.272
## 10     0.55       0.132       0.981 0.642 0.812 3.420  2.270    0.805 0.220
## 11     0.60       0.102       0.987 0.670 0.808 3.487  2.446    0.804 0.177
## 12     0.65       0.084       0.992 0.726 0.806 3.736  2.942    0.804 0.151
## 13     0.70       0.062       0.996 0.789 0.802 3.997  3.812    0.802 0.115
## 14     0.75       0.046       0.998 0.835 0.800 4.178  4.862    0.800 0.086
## 15     0.80       0.028       0.999 0.930 0.797 4.590 11.428    0.798 0.054
## 16     0.85       0.013       1.000 0.905 0.795 4.413  8.347    0.795 0.026
## 17     0.90       0.003       1.000 0.833 0.793 4.033  4.760    0.793 0.007
```

```r
results
```


```
##    Cutpoint Sensitivity Specificity   PPV   NPV   PLR    NLR Accuracy    F1
## 1      0.10       0.915       0.321 0.261 0.935 4.037  1.265    0.444 0.406
## 2      0.15       0.788       0.523 0.301 0.904 3.142  1.294    0.578 0.436
## 3      0.20       0.681       0.670 0.350 0.889 3.165  1.369    0.672 0.463
## 4      0.25       0.569       0.779 0.402 0.874 3.183  1.461    0.735 0.471
## 5      0.30       0.457       0.853 0.448 0.857 3.141  1.554    0.771 0.452
## 6      0.35       0.363       0.903 0.493 0.844 3.171  1.667    0.791 0.419
## 7      0.40       0.277       0.937 0.537 0.832 3.201  1.797    0.801 0.366
## 8      0.45       0.212       0.960 0.581 0.823 3.292  1.966    0.805 0.311
## 9      0.50       0.174       0.973 0.630 0.818 3.471  2.212    0.808 0.272
## 10     0.55       0.132       0.981 0.642 0.812 3.420  2.270    0.805 0.220
## 11     0.60       0.102       0.987 0.670 0.808 3.487  2.446    0.804 0.177
## 12     0.65       0.084       0.992 0.726 0.806 3.736  2.942    0.804 0.151
## 13     0.70       0.062       0.996 0.789 0.802 3.997  3.812    0.802 0.115
## 14     0.75       0.046       0.998 0.835 0.800 4.178  4.862    0.800 0.086
## 15     0.80       0.028       0.999 0.930 0.797 4.590 11.428    0.798 0.054
```

```
## 16      0.85         0.013        1.000 0.905 0.795 4.413  8.347    0.795 0.026
## 17      0.90         0.003        1.000 0.833 0.793 4.033  4.760    0.793 0.007
```

Bootsrap