Lista 02 – Métodos de Preprocesamiento de Datos

Análisis de datos y estrategias de balanceo

Víctor Mauricio Ochoa García, Salvador Enrique Rodríguez Hernández

06 de noviembre de 2025

Contents

1 Datos y utilidades **3**

2 (a) ¿Por qué imputación kNN en todas las predictoras con NA? **5**

3 (b) Proporción de clases de la respuesta (barras con porcentajes) **5**

4 (c) SMOTE + Regresión logística (corte 0.5) **7**

5 (d) ENN + Regresión logística (corte = prop. minoritaria) **7**

6 (e) SMOTE + ENN + Regresión logística (corte 0.5) **8**

7 (f) Base desbalanceada + Regresión logística (dos cortes) **9**

8 (g) Discusión y conclusión **10**

9 Conclusión principal. **10**

9.1 Influencia del punto de corte. 11

9.2 Qué usaría en producción. 11

```r
knitr::opts_chunk$set(
echo = TRUE, message = FALSE, warning = FALSE, fig.width = 7, fig.height = 5
)

suppressPackageStartupMessages({
library(readr)
library(dplyr)
library(ggplot2)
library(rsample)
library(recipes)
library(themis)   # SMOTE
library(FNN)      # knn/ENN
library(knitr)
library(tibble)
})
set.seed(2026)
```

1 Datos y utilidades

```r
# Dataset UCI (trata "?" como NA)

url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/00383/risk_factors_cervical_
cc    <- read_csv(url, na = "?")
datos <- as.data.frame(cc)

target_var <- "Biopsy"
stopifnot(target_var %in% names(datos))

# Asegurar factor binario "0"/"1"

y <- datos[[target_var]]
if (!is.factor(y)) y <- factor(as.character(y))
if (!all(levels(y) %in% c("0","1"))) {
y <- factor(as.character(y), levels = c("0","1"))
}
datos[[target_var]] <- y
```

```r
# Métricas (idénticas al estilo de clase)

calc_metrics <- function(y_true, y_pred) {
tab <- table(Pred = y_pred, Real = y_true)
TP <- ifelse("1" %in% rownames(tab) & "1" %in% colnames(tab), tab["1","1"], 0)
TN <- ifelse("0" %in% rownames(tab) & "0" %in% colnames(tab), tab["0","0"], 0)
FP <- ifelse("1" %in% rownames(tab) & "0" %in% colnames(tab), tab["1","0"], 0)
```

```r
FN <- ifelse("0" %in% rownames(tab) & "1" %in% colnames(tab), tab["0","1"], 0)

acc   <- ifelse((TP + TN + FP + FN) > 0, (TP + TN)/(TP + TN + FP + FN), NA)
sens  <- ifelse((TP + FN) > 0, TP / (TP + FN), NA)
esp   <- ifelse((TN + FP) > 0, TN / (TN + FP), NA)
ppv   <- ifelse((TP + FP) > 0, TP / (TP + FP), NA)
npv   <- ifelse((TN + FN) > 0, TN / (TN + FN), NA)
gmean <- ifelse(!is.na(sens) & !is.na(esp), sqrt(sens * esp), NA)
f1    <- ifelse((ppv + sens) > 0, 2 * ppv * sens / (ppv + sens), NA)

denom <- sqrt( (TP + FP) * (TP + FN) * (TN + FP) * (TN + FN) )
mcc   <- ifelse(denom > 0, (TP*TN - FP*FN) / denom, NA)

data.frame(Accuracy = acc, Sensitivity = sens, Specificity = esp,
PPV = ppv, NPV = npv, Gmean = gmean, F1 = f1, MCC = mcc)
}

# ENN manual (edita ejemplos de la clase mayoritaria que discrepan con vecinos)

ENN_manual <- function(data, target, k = 3, majority_class) {
X <- data[, setdiff(names(data), target), drop = FALSE]
y <- data[[target]]
X_num <- as.data.frame(lapply(X, function(z) if(is.numeric(z)) z else as.numeric(as.factor(z))))
nn <- knnx.index(as.matrix(X_num), as.matrix(X_num), k = k + 1)[, -1]
remove_idx <- sapply(1:nrow(X_num), function(i) {
if (y[i] == majority_class) {
neigh_classes <- y[nn[i, ]]
maj_class <- names(sort(table(neigh_classes), decreasing = TRUE))[1]
return(maj_class != y[i])
} else FALSE
})
data[!remove_idx, , drop = FALSE]
}

# Helper: ajustar y evaluar regresión logística

eval_logit <- function(train_df, test_df, cutoff = 0.5, target = target_var) {
form <- as.formula(paste(target, "~ ."))
mod  <- glm(form, data = train_df, family = binomial)
prob <- predict(mod, newdata = test_df, type = "response")
pred <- ifelse(prob >= cutoff, "1", "0")
pred <- factor(pred, levels = c("0","1"))
list(metrics = calc_metrics(test_df[[target]], pred),
cutoff = cutoff,
n_train = nrow(train_df))
}
```

2 (a) ¿Por qué imputación kNN en todas las predictoras con NA?

```
na_counts <- sapply(datos, function(v) sum(is.na(v)))
na_pct    <- round(100 * na_counts / nrow(datos), 2)
na_tbl    <- data.frame(var = names(na_counts), NA_count = na_counts, NA_percent = na_pct) |>
arrange(desc(NA_percent))
kable(head(na_tbl, 12), caption = "Top 12 variables con mayor porcentaje de NA")
```

Table 1: Top 12 variables con mayor porcentaje de NA

	var	NA_count	NA_percent
STDs: Time since first diagnosis	STDs: Time since first diagnosis	787	91.72
STDs: Time since last diagnosis	STDs: Time since last diagnosis	787	91.72
IUD	IUD	117	13.64
IUD (years)	IUD (years)	117	13.64
Hormonal Contraceptives	Hormonal Contraceptives	108	12.59
Hormonal Contraceptives (years)	Hormonal Contraceptives (years)	108	12.59
STDs	STDs	105	12.24
STDs (number)	STDs (number)	105	12.24
STDs:condylomatosis	STDs:condylomatosis	105	12.24
STDs:cervical condylomatosis	STDs:cervical condylomatosis	105	12.24
STDs:vaginal condylomatosis	STDs:vaginal condylomatosis	105	12.24
STDs:vulvo-perineal condylomatosis	STDs:vulvo-perineal condylomatosis	105	12.24

Respuesta (a). Varias variables presentan faltantes considerables (dos por arriba de 90% y varias en 10–15%). Eliminar filas con NA reduciría drásticamente el tamaño muestral y sesgaría el análisis. La imputación por k-vecinos (kNN) aprovecha la estructura multivariada: cada NA se reemplaza usando observaciones similares en el espacio de variables, preservando coherencia entre predictores. Por ello se decidió imputar con kNN en todas las predictoras con NA (y luego estandarizar).

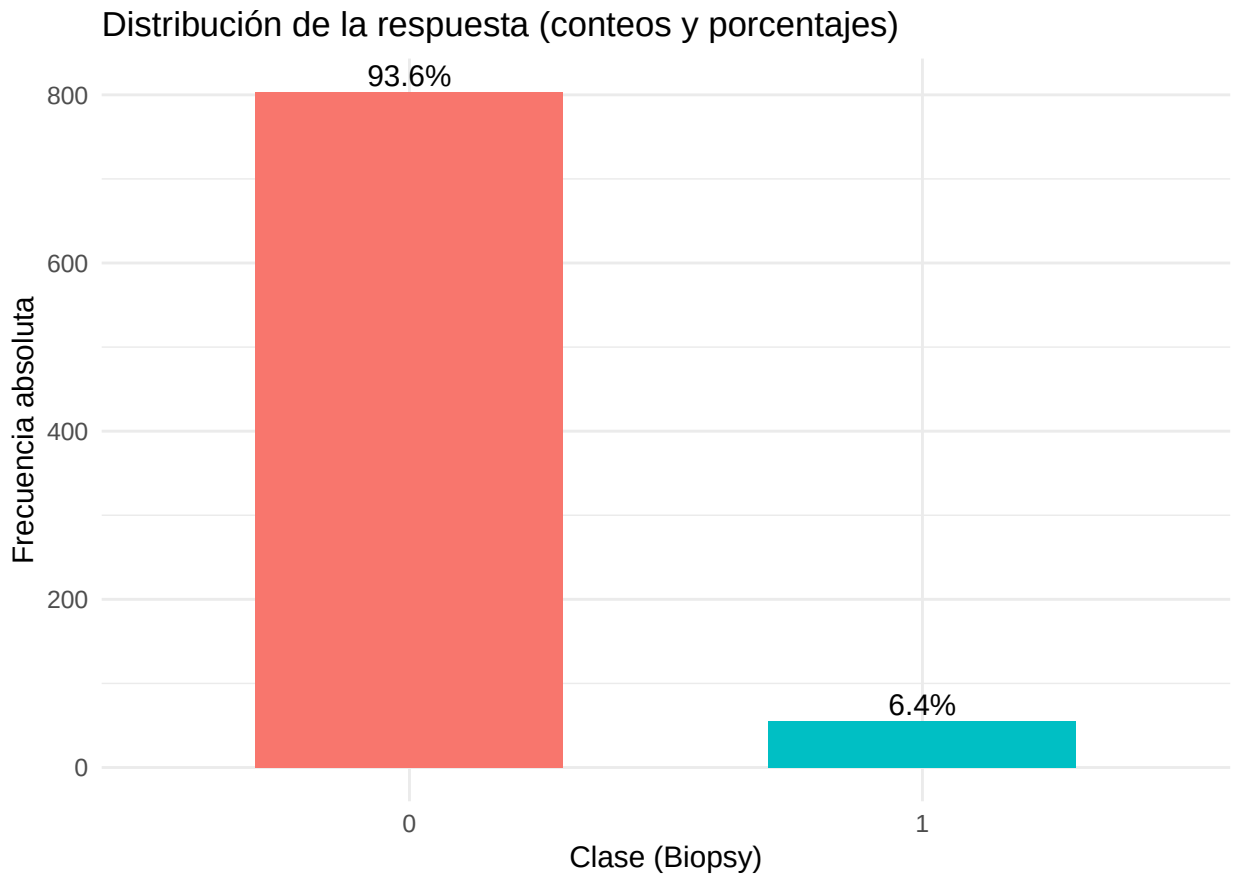
3 (b) Proporción de clases de la respuesta (barras con porcentajes)

```
g_base <- datos %>%
count(!!sym(target_var)) %>%
mutate(prop = 100 * n / sum(n))
kable(g_base, caption = "Frecuencia y porcentaje por clase de Biopsy")
```

Table 2: Frecuencia y porcentaje por clase de Biopsy

Biopsy	n	prop
0	803	93.589744
1	55	6.410256

```
ggplot(g_base, aes(x = !!sym(target_var), y = n, fill = !!sym(target_var))) +
geom_col(width = 0.6, show.legend = FALSE) +
geom_text(aes(label = paste0(round(prop,1), "%")), vjust = -0.3, size = 4.2) +
labs(x = "Clase (Biopsy)", y = "Frecuencia absoluta",
title = "Distribución de la respuesta (conteos y porcentajes)") +
theme_minimal(base_size = 12)
```



La clase positiva (1) representa r sprintf('%.1f%%', min(g_base$prop)) del total: el problema está fuertemente desbalanceado.

Preparación común (imputar, normalizar, splits)

```
# Split estratificado 70/30

split_obj <- initial_split(datos, prop = 0.70, strata = all_of(target_var))
```

```r
treino0 <- training(split_obj)
teste0  <- testing(split_obj)

# Imputar primero, normalizar después (evita warnings y es el orden correcto)

rec_imp <- recipe(as.formula(paste(target_var, "~ .")), data = treino0) %>%
step_impute_knn(all_numeric_predictors(), neighbors = 5) %>%
step_zv(all_predictors()) %>%                    # quita predictores constantes
step_normalize(all_numeric_predictors())

imp_prep   <- prep(rec_imp)
treino_imp <- bake(imp_prep, new_data = NULL)
teste_imp  <- bake(imp_prep, new_data = teste0)

minority_prop <- min(g_base$prop / 100)
```

4 (c) SMOTE + Regresión logística (corte 0.5)

```r
rec_smote <- recipe(as.formula(paste(target_var, "~ .")), data = treino_imp) %>%
step_smote(all_outcomes()) %>%
step_zv(all_predictors())
smote_prep   <- prep(rec_smote)
treino_smote <- bake(smote_prep, new_data = NULL)

res_c  <- eval_logit(treino_smote, teste_imp, cutoff = 0.5)
row_c  <- cbind(Modelo = "SMOTE (cut=0.5)", Ntrain = res_c$n_train, Cutoff = res_c$cutoff, res
kable(row_c, caption = "Resultados (c): SMOTE + logística (cut=0.5)")
```

Table 3: Resultados (c): SMOTE + logística (cut=0.5)

Modelo	Ntrain	Cutoff	Accuracy	Sensitivity	Specificity	PPV	NPV	Gmean	F1	MCC
SMOTE (cut=0.5)	1120	0.5	0.9534884	0.9333333	0.9547325	0.56	0.9957082	0.9439723	0.7	0.7024994

SMOTE aumenta la presencia de la clase minoritaria en entrenamiento y la logística mejora el equilibrio sensibilidad–especificidad.

5 (d) ENN + Regresión logística (corte = prop. minoritaria)

```r
maj_class  <- names(sort(table(treino_imp[[target_var]]), decreasing = TRUE))[1]
treino_ENN <- ENN_manual(treino_imp, target = target_var, k = 3, majority_class = maj_class)

prop_ENN <- prop.table(table(treino_ENN[[target_var]]))
kable(as.data.frame(prop_ENN), col.names = c("Clase", "Proporción"),
caption = "Proporciones en entrenamiento después de ENN")
```

Table 4: Proporciones en entrenamiento después de ENN

Clase	Proporción
0	0.9322034
1	0.0677966

```r
cut_ENN <- as.numeric(min(prop.table(table(treino_imp[[target_var]]))))
res_d   <- eval_logit(treino_ENN, teste_imp, cutoff = cut_ENN)
row_d   <- cbind(Modelo = sprintf("ENN (cut=%.3f)", cut_ENN), Ntrain = res_d$n_train, Cutoff =
kable(row_d, caption = "Resultados (d): ENN + logística (cut = prop. minoritaria)")
```

Table 5: Resultados (d): ENN + logística (cut = prop. minoritaria)

Modelo	Ntrain	Cutoff	Accuracy	Sensitivity	Specificity	PPV	NPV	Gmean	F1	MCC
ENN (cut=0.067)	590	0.0666667	0.9496124	0.9333333	0.9506173	0.5384615	0.9956897	0.9419357	0.6829268	0.6871414

ENN elimina mayoritarios "ruidosos". Con corte igual a la proporción minoritaria se empuja el modelo a recuperar más positivos.

6 (e) SMOTE + ENN + Regresión logística (corte 0.5)

```r
treino_ENN2 <- ENN_manual(treino_imp, target = target_var, k = 3, majority_class = maj_class)
rec_smote2  <- recipe(as.formula(paste(target_var, "~ .")), data = treino_ENN2) %>%
step_smote(all_outcomes()) %>%
step_zv(all_predictors())
smote2_prep      <- prep(rec_smote2)
treino_ENN_SMOTE <- bake(smote2_prep, new_data = NULL)

res_e <- eval_logit(treino_ENN_SMOTE, teste_imp, cutoff = 0.5)
row_e <- cbind(Modelo = "SMOTE+ENN (cut=0.5)", Ntrain = res_e$n_train, Cutoff = res_e$cutoff,
kable(row_e, caption = "Resultados (e): SMOTE+ENN + logística (cut=0.5)")
```

Table 6: Resultados (e): SMOTE+ENN + logística (cut=0.5)

Modelo	Ntrain	Cutoff	Accuracy	Sensitivity	Specificity	PPV	NPV	Gmean	F1	MCC
SMOTE+ENN (cut=0.5)	1100	0.5	0.9457364	0.9333333	0.9465021	0.5185185	0.995671	0.9398946	0.6666667	0.6726085

La combinación no necesariamente supera a SMOTE solo; si ENN elimina casos informativos antes del balanceo, la señal puede debilitarse.

7 (f) Base desbalanceada + Regresión logística (dos cortes)

```
# Sin SMOTE/ENN: solo imputación/normalización

res_f1 <- eval_logit(treino_imp, teste_imp, cutoff = 0.5)
row_f1 <- cbind(Modelo = "Desbalanceada (cut=0.5)", Ntrain = res_f1$n_train, Cutoff = res_f1$cu

res_f2 <- eval_logit(treino_imp, teste_imp, cutoff = minority_prop)
row_f2 <- cbind(Modelo = sprintf("Desbalanceada (cut=%.3f)", minority_prop), Ntrain = res_f2$n_

kable(row_f1, caption = "Resultados (f1): Desbalanceada + logística (cut=0.5)")
```

Table 7: Resultados (f1): Desbalanceada + logística (cut=0.5)

Modelo	Ntrain	Cutoff	Accuracy	Sensitivity	Specificity	PPV	NPV	Gmean	F1	MCC
Desbalanceada (cut=0.5)	600	0.5	0.9457364	0.6666667	0.962963	0.5263158	0.9790795	0.8012336	0.5882353	0.5641027

```
kable(row_f2, caption = "Resultados (f2): Desbalanceada + logística (cut=prop. minoritaria)")
```

Table 8: Resultados (f2): Desbalanceada + logística (cut=prop. minoritaria)

Modelo	Ntrain	Cutoff	Accuracy	Sensitivity	Specificity	PPV	NPV	Gmean	F1	MCC
Desbalanceada (cut=0.064)	600	0.064102	0.9379845	0.9333333	0.9382716	0.4827586	0.9956332	0.9357992	0.6363636	0.6457311

Tabla comparativa final

```r
tabla_final <- dplyr::bind_rows(row_c, row_d, row_e, row_f1, row_f2) %>%
as_tibble()

kable(tabla_final, digits = 3, caption = "Resumen de desempeño por estrategia")
```

Table 9: Resumen de desempeño por estrategia

Modelo	Ntrain	Cutoff	Accuracy	Sensitivity	Specificity	PPV	NPV	Gmean	F1	MCC
SMOTE (cut=0.5)	1120	0.500	0.953	0.933	0.955	0.560	0.996	0.944	0.700	0.702
ENN (cut=0.067)	590	0.067	0.950	0.933	0.951	0.538	0.996	0.942	0.683	0.687
SMOTE+ENN (cut=0.5)	1100	0.500	0.946	0.933	0.947	0.519	0.996	0.940	0.667	0.673
Desbalanceada (cut=0.5)	600	0.500	0.946	0.667	0.963	0.526	0.979	0.801	0.588	0.564
Desbalanceada (cut=0.064)	600	0.064	0.938	0.933	0.938	0.483	0.996	0.936	0.636	0.646

8 (g) Discusión y conclusión

Criterios robustos en desbalance. En datos desbalanceados, F1, G-mean y MCC son métricas más informativas que la exactitud. F1 resume precisión–recobrado, G-mean evalúa el balance sensibilidad–especificidad y MCC mide la correlación entre predicción y verdad (–1 a 1), siendo muy exigente.

```r
# Elegimos el "mejor" por F1 y, como desempate, por MCC y G-mean

pick <- tabla_final %>%
mutate(across(c(F1, MCC, Gmean), as.numeric)) %>%
arrange(desc(F1), desc(MCC), desc(Gmean)) %>%
slice(1)
best_model   <- pick$Modelo
best_f1      <- pick$F1
best_mcc     <- pick$MCC
best_gmean   <- pick$Gmean
```

9 Conclusión principal.

La estrategia con mejor compromiso global fue r best_model, con F1 = r sprintf('%.3f', best_f1), MCC = r sprintf('%.3f', best_mcc) y G-mean = r sprintf('%.3f', best_gmean). Esto indica que logra detectar más positivos sin deteriorar en exceso la precisión ni la especificidad, y mantiene una correlación alta entre las predicciones y la realidad.

9.1 Influencia del punto de corte.

Con corte = 0.5 en base desbalanceada, el clasificador favorece la clase mayoritaria (alta especificidad, baja sensibilidad).

Con corte = proporción minoritaria, aumenta la sensibilidad (recupera más casos positivos) a costa de cierta pérdida en precisión.

Tras SMOTE, un corte = 0.5 vuelve razonable porque el entrenamiento está balanceado; de allí el buen F1/G-mean/MCC observados.

9.2 Qué usaría en producción.

Adoptaría SMOTE + logística (con pipeline de imputación kNN $\rightarrow$ normalización, y chequeo de predictores constantes), dejando el punto de corte calibrado en validación según la métrica prioritaria clínica (p. ej., maximizar F1 o un costo asimétrico de falso negativo).

ENN puede ser útil cuando hay mucho ruido en la mayoría, pero en estos datos no superó a SMOTE solo.

El preprocesamiento importa: con datos clínicos desbalanceados y con NA, la imputación multivariada, el balanceo y la calibración del umbral hacen la diferencia entre un modelo "con buena exactitud" y uno realmente útil para detectar casos positivos.