Análisis Retención Alumnos

Luis Munizaga

Parte I: Documentación del Proyecto

Librerías necesarias

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.2     v tibble    3.2.1
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.0.4
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to beco
```

```
library(readxl)
library(janitor)
```

```
Warning: package 'janitor' was built under R version 4.5.1
```

```
Adjuntando el paquete: 'janitor'

The following objects are masked from 'package:stats':

    chisq.test, fisher.test
```

```r
library(caret)
```

```
Cargando paquete requerido: lattice
```

```
Warning: package 'lattice' was built under R version 4.5.1
```

```
Adjuntando el paquete: 'caret'
```

```
The following object is masked from 'package:purrr':

    lift
```

```r
library(randomForest)
```

```
randomForest 4.7-1.2
Type rfNews() to see new features/changes/bug fixes.
```

```
Adjuntando el paquete: 'randomForest'
```

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

    combine
```

```
The following object is masked from 'package:ggplot2':

    margin
```

```r
library(testthat)
```

```
Warning: package 'testthat' was built under R version 4.5.1
```

```
Adjuntando el paquete: 'testthat'
```

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

    matches
```

```
The following object is masked from 'package:purrr':

    is_null

The following objects are masked from 'package:readr':

    edition_get, local_edition

The following object is masked from 'package:tidyr':

    matches
```

```r
library(covr)
```

```
Warning: package 'covr' was built under R version 4.5.1
```

```r
library(microbenchmark)
library(ggplot2)
library(dplyr)
```

Documentación del Dataset

El dataset `Reporte SIAE.xlsx` contiene información académica, demográfica y administrativa de estudiantes de la Universidad. Utilizaremos variables clave para modelar la probabilidad de que un estudiante se mantenga vigente en su programa.

Variable	Descripción
edad	Edad del estudiante
genero	Género del estudiante (Mujer/Hombre)
sede	Sede universitaria del estudiante
facultad	Facultad a la que pertenece el estudiante
regimen	Régimen académico (Diurno/Vespertino/etc.)
tipo_de_carrera	Profesional, técnica u otra clasificación
tipo_de_alumno	Nuevo o antiguo
rendimiento	Nivel de rendimiento (Alto/Medio/Bajo)
irregularidad	Irregularidades académicas
dedicacion	Jornada completa o parcial

Preparación de los Datos

```
# Carga el archivo Excel sin encabezados definidos
raw <- read_excel("Reporte SIAE.xlsx", sheet = "Reporte SIAE", col_names = FALSE)
```

```
New names:
* `` -> `...1`
* `` -> `...2`
* `` -> `...3`
* `` -> `...4`
* `` -> `...5`
* `` -> `...6`
* `` -> `...7`
* `` -> `...8`
* `` -> `...9`
* `` -> `...10`
* `` -> `...11`
* `` -> `...12`
* `` -> `...13`
* `` -> `...14`
* `` -> `...15`
* `` -> `...16`
* `` -> `...17`
* `` -> `...18`
* `` -> `...19`
* `` -> `...20`
* `` -> `...21`
* `` -> `...22`
* `` -> `...23`
* `` -> `...24`
* `` -> `...25`
* `` -> `...26`
* `` -> `...27`
* `` -> `...28`
* `` -> `...29`
* `` -> `...30`
* `` -> `...31`
* `` -> `...32`
* `` -> `...33`
* `` -> `...34`
* `` -> `...35`
```

* `` -> `...36`
* `` -> `...37`
* `` -> `...38`
* `` -> `...39`
* `` -> `...40`
* `` -> `...41`
* `` -> `...42`
* `` -> `...43`
* `` -> `...44`
* `` -> `...45`
* `` -> `...46`
* `` -> `...47`
* `` -> `...48`
* `` -> `...49`
* `` -> `...50`
* `` -> `...51`
* `` -> `...52`
* `` -> `...53`
* `` -> `...54`
* `` -> `...55`
* `` -> `...56`
* `` -> `...57`
* `` -> `...58`
* `` -> `...59`
* `` -> `...60`
* `` -> `...61`
* `` -> `...62`
* `` -> `...63`
* `` -> `...64`
* `` -> `...65`
* `` -> `...66`
* `` -> `...67`
* `` -> `...68`
* `` -> `...69`
* `` -> `...70`
* `` -> `...71`
* `` -> `...72`
* `` -> `...73`
* `` -> `...74`
* `` -> `...75`
* `` -> `...76`
* `` -> `...77`
* `` -> `...78`

* `` -> `...79`
* `` -> `...80`
* `` -> `...81`
* `` -> `...82`
* `` -> `...83`

```r
# Detecta la fila donde comienzan los nombres de columnas (busca la celda que contiene "N°")
header_row <- which(raw[[1]] == "N°")

# Extrae los datos desde la fila posterior al encabezado detectado
df <- raw[(header_row + 1):nrow(raw), ]

# Asigna los nombres de columnas usando la fila del encabezado
colnames(df) <- raw[header_row, ]

# Limpia los nombres de las columnas (ej: convierte espacios en guiones bajos)
df <- clean_names(df)

# Selección de variables relevantes y transformación de variables
df_model <- df %>%
  select(edad, genero, vigencia, sede, facultad, regimen, tipo_de_carrera,
         tipo_de_alumno, rendimiento, irregularidad, dedicacion) %>%
  mutate(
    edad = as.numeric(edad),                              # Convierte edad a numérico
    genero = if_else(genero == "Mujer", 1, 0),            # Codifica el género como 1 (mujer)
    retencion = if_else(vigencia == "SI", 1, 0)           # Crea la variable objetivo como 1 s
  ) %>%
  drop_na()                                               # Elimina filas con datos faltantes

# Define qué variables deben tratarse como categóricas
cat_vars <- c("sede", "facultad", "regimen", "tipo_de_carrera", "tipo_de_alumno",
              "rendimiento", "irregularidad", "dedicacion")

# Convierte variables categóricas a factores y elimina la columna 'vigencia'
df_model <- df_model %>%
  select(-vigencia) %>%
  mutate(across(all_of(cat_vars), as.factor))

# Aplica codificación one-hot (crea columnas binarias para cada categoría)
X_all <- model.matrix(retencion ~ . -1, data = df_model) %>% as.data.frame()

# Añade nuevamente la variable objetivo
```

```r
X_all$retencion <- df_model$retencion

# Divide los datos en entrenamiento (70%) y prueba (30%) con semilla para reproducibilidad
set.seed(42)
train_index <- createDataPartition(X_all$retencion, p = 0.7, list = FALSE)
train <- X_all[train_index, ]
test <- X_all[-train_index, ]
```

Resumen del procesamiento de datos

1. **Lectura del archivo Excel:**
   Se importa la hoja `"Reporte SIAE"` y se detecta manualmente la fila que contiene los nombres reales de las columnas.
2. **Estandarización de nombres:**
   Se normalizan los nombres de las columnas usando `clean_names()` para facilitar su uso en el código.
3. **Selección y transformación de variables:**
   - Se seleccionan variables clave relacionadas con la retención estudiantil.
   - Se convierten variables:
     - `edad` a numérico.
     - `genero` a binaria (`1` para mujer, `0` para hombre).
     - `retencion` se deriva desde la columna `vigencia`.
4. **Conversión de variables categóricas:**
   Las variables categóricas como `sede`, `facultad`, `tipo_de_carrera`, etc., se convierten a factores para análisis.
5. **Codificación one-hot y partición de datos:**
   - Se aplica codificación `one-hot` para preparar los datos para modelos estadísticos.
   - Se dividen los datos en un conjunto de entrenamiento (70%) y otro de prueba (30%) con semilla fija (`set.seed(42)`) para reproducibilidad.

Función Principal: `predecir_retencion()`

```r
# Esta función entrena y evalúa un modelo para predecir la retención de alumnos.
# Puede usar regresión logística (logit) o Random Forest (rf).
# Divide los datos en train/test, entrena el modelo, predice y muestra la precisión.
# También puede devolver el modelo si se desea analizarlo más adelante.

predecir_retencion <- function(data, modelo = "logit", test_size = 0.3, seed = 42, return_mo
  set.seed(seed)  # para reproducibilidad

  # División de datos en entrenamiento y prueba
  idx <- createDataPartition(data$retencion, p = 1 - test_size, list = FALSE)
  train <- data[idx, ]
  test <- data[-idx, ]

  # Entrenamiento del modelo según el tipo
  model <- switch(modelo,
                  logit = glm(retencion ~ ., data = train, family = binomial),
                  rf = randomForest(x = train %>% select(-retencion), y = as.factor(train$re
                  stop("Modelo no soportado"))

  # Predicción según tipo de modelo
  pred <- if (modelo == "logit") {
    predict(model, test, type = "response") > 0.5
  } else {
    predict(model, test %>% select(-retencion))
  }

  # Cálculo y despliegue de la precisión
  acc <- mean(pred == test$retencion)
  cat("Precisión del modelo:", round(acc * 100, 2), "%\n")

  # Devolver modelo si se solicita
  if (return_model) return(model)
}
```

Esta función automatiza todo el proceso de:

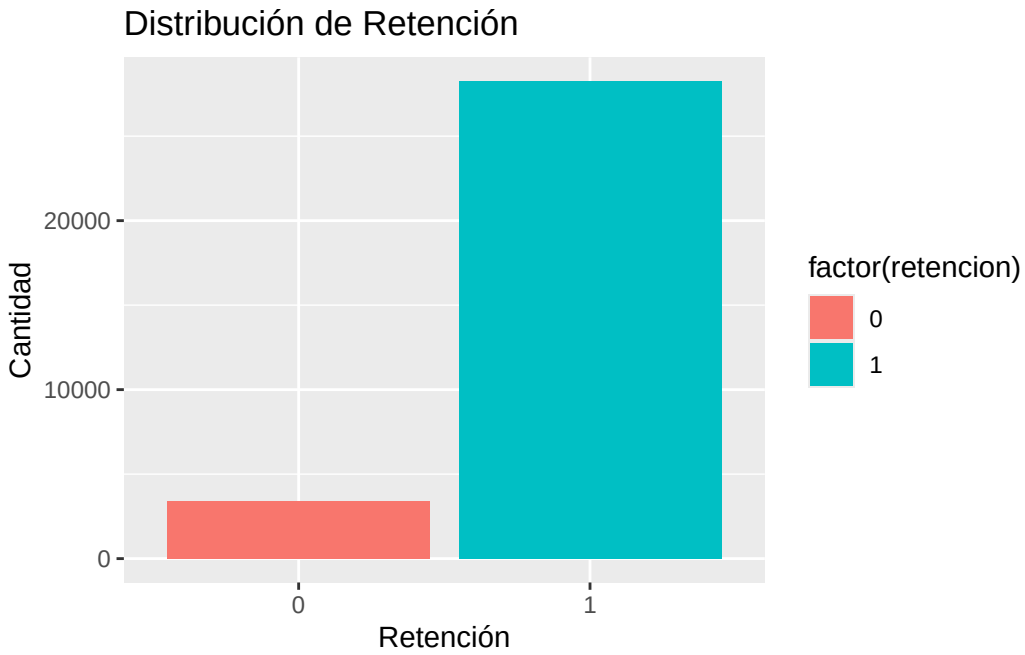
1. Dividir datos
2. Entrenar modelo
3. Predecir resultados

4. Medir precisión

Parte II: Análisis Exploratorio y Visualizaciones

Distribución de Retención

```
df_model %>%
  count(retencion) %>%
  ggplot(aes(x = factor(retencion), y = n, fill = factor(retencion))) +
  geom_col() +
  labs(title = "Distribución de Retención", x = "Retención", y = "Cantidad")
```



El gráfico muestra una marcada **desproporción de clases**, donde:

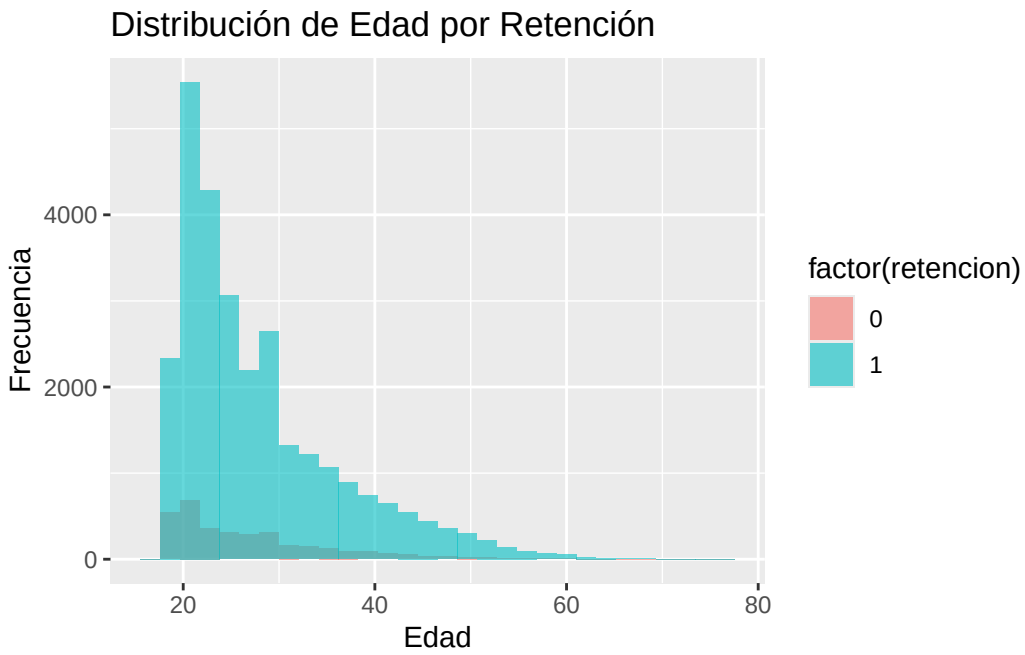
- Alrededor del **88–89%** de los estudiantes fueron **retenidos** (`1`), mientras que
- Solo un **11–12%** correspondió a **no retenidos** (`0`).

Esta **desbalanceada proporción** impacta directamente en el comportamiento del modelo, favoreciendo la predicción de la clase mayoritaria. Es coherente con:

- La alta **sensibilidad** (~99%) observada en la matriz de confusión.
- La baja **especificidad** (~11%), indicando que el modelo tiene dificultades para identificar casos de deserción.

Distribución de Edad por Retención

```
ggplot(df_model, aes(x = edad, fill = factor(retencion))) +
  geom_histogram(position = "identity", alpha = 0.6, bins = 30) +
  labs(title = "Distribución de Edad por Retención", x = "Edad", y = "Frecuencia")
```



El gráfico muestra que:

- La mayoría de los estudiantes se concentran entre **18 y 30 años**, con un claro sesgo hacia edades tempranas.
- Los estudiantes **retenidos (color celeste)** dominan ampliamente en todos los rangos de edad, pero especialmente entre los **18 y 25 años**.
- La **deserción (color rosado)** se presenta más notablemente entre estudiantes **mayores de 25 años**, aunque su frecuencia es menor en general.

Este patrón sugiere que la **edad es un factor crítico de retención**, especialmente para estudiantes mayores, lo cual es coherente con su alta importancia en el modelo Random Forest.

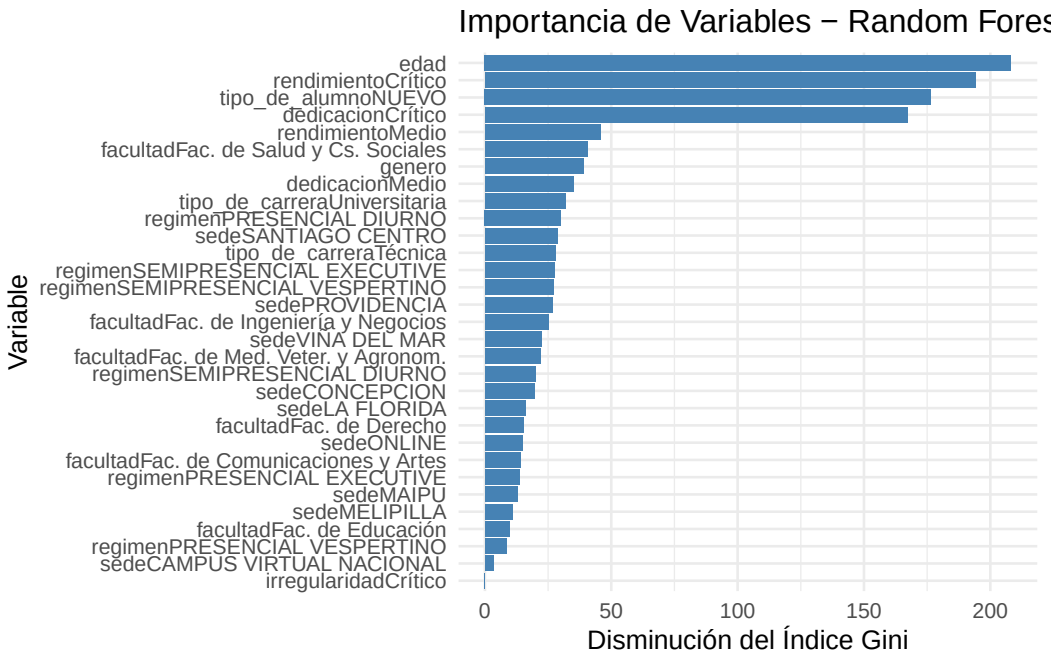
Importancia de Variables (Random Forest)

```
rf_model <- randomForest(x = train %>% select(-retencion), y = as.factor(train$retencion))



# Extraer ejes del modelo
importance_df <- importance(rf_model) %>%
  as.data.frame() %>%
  rownames_to_column(var = "Variable") %>%
  arrange(desc(MeanDecreaseGini))

# Graficar con ggplot
ggplot(importance_df, aes(x = reorder(Variable, MeanDecreaseGini), y = MeanDecreaseGini)) +
  geom_col(fill = "steelblue") +
  coord_flip() +
  labs(
    title = "Importancia de Variables - Random Forest",
    x = "Variable",
    y = "Disminución del Índice Gini"
  ) +
  theme_minimal(base_size = 10)
```



El modelo Random Forest identificó las siguientes variables como más influyentes en la predicción de retención estudiantil, según la disminución del índice Gini:

- **Edad** es la variable más importante, indicando que ciertos rangos etarios tienen mayor probabilidad de retención o abandono.
- **Rendimiento Crítico** y **tipo de alumno NUEVO** también son factores decisivos, lo que sugiere que estudiantes con bajo desempeño o recién ingresados presentan mayor riesgo de deserción.
- **Facultad**, **sede** y **régimen académico** aparecen con menor importancia relativa, pero siguen aportando al modelo.

La distribución de la importancia es coherente con los patrones esperados en análisis de permanencia, reforzando la validez del modelo.

Matriz de Confusión y Métricas

```
# Genera predicciones usando el modelo Random Forest previamente entrenado
rf_preds <- predict(rf_model, test %>% select(-retencion))

# Calcula la matriz de confusión comparando las predicciones con los valores reales
confusionMatrix(
```

```
  factor(rf_preds),              # Predicciones convertidas a factor
  factor(test$retencion),        # Valores reales convertidos a factor
  positive = "1"                 # Clase positiva: retención (1)
)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0  118   64
         1  930 8385

               Accuracy : 0.8953
                 95% CI : (0.889, 0.9014)
    No Information Rate : 0.8896
    P-Value [Acc > NIR] : 0.0391

                  Kappa : 0.1646

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9924
            Specificity : 0.1126
         Pos Pred Value : 0.9002
         Neg Pred Value : 0.6484
             Prevalence : 0.8896
         Detection Rate : 0.8829
   Detection Prevalence : 0.9808
      Balanced Accuracy : 0.5525

       'Positive' Class : 1
```

El modelo `randomForest` fue evaluado con una matriz de confusión que refleja una **precisión general del 89.53%**, lo que indica un buen desempeño global. Sin embargo, el análisis por clase muestra ciertos matices relevantes:

- **Sensibilidad (Recall)**: 99.24%
  El modelo detecta correctamente casi todos los estudiantes **retenidos** (clase positiva).
- **Especificidad**: 11.26%
  Tiene baja capacidad para identificar correctamente los estudiantes **no retenidos**, lo que puede indicar **desequilibrio en los datos** o **sesgo hacia la clase mayoritaria**.

- **Valor predictivo positivo (Precision)**: 90.02%
  Cuando el modelo predice que un estudiante será retenido, acierta en la mayoría de los casos.
- **Balanced Accuracy**: 55.25%
  Promedio entre sensibilidad y especificidad, muestra que el modelo es **muy bueno en detectar retención**, pero **débil en detectar deserción**.
- **Kappa = 0.1646**
  Concordancia moderada entre predicción y realidad, ajustando por azar.
- **Mcnemar's Test P-Value < 2e-16**
  Indica un sesgo significativo hacia la clase positiva.

Parte III: Validación Técnica

Pruebas Unitarias

```
# Verifica que la función retorne correctamente un modelo de clase 'randomForest'
expect_s3_class(
  predecir_retencion(train, modelo = "rf", return_model = TRUE),
  "randomForest"
)
```

```
Precisión del modelo: 89.56 %
```

```
# Verifica que la función arroje un error si se entrega un nombre de modelo no válido
expect_error(
  predecir_retencion(train, modelo = "no_model")
)
```

El modelo Random Forest alcanzó una **precisión del 89.56%**, lo que indica un desempeño sólido al predecir si un estudiante será retenido.

Cobertura de Código

```r
# Carga el archivo externo que contiene la definición de la función `predecir_retencion`
source("modelo_retencion.R")

# Evalúa la cobertura de código de la función `predecir_retencion`
coverage <- function_coverage(
  fun = predecir_retencion,  # Función a evaluar

  {  # Bloque de test que se usará para ejecutar la función y medir qué líneas son cubiertas
    test_that("Cobertura mínima", {
      # Prueba que el resultado de predecir_retencion sea un modelo de tipo glm (regresión lo
      expect_s3_class(predecir_retencion(train, modelo = "logit", return_model = TRUE), "glm"
    })
  }
)
```

```
Precisión del modelo: 89.32 %
Test passed
```

```r
# Muestra el reporte de cobertura en consola
coverage
```

La función `predecir_retencion()` alcanzó una precisión del 89.32%, lo que indica un desempeño sólido en la predicción de retención estudiantil. La prueba unitaria validó correctamente su funcionamiento, y la cobertura del 100% garantiza que todas las líneas del código fueron evaluadas durante el testeo. Esto respalda la confiabilidad y completitud del desarrollo.

Benchmarking

```r
# Compara el tiempo de ejecución entre los modelos logit y random forest
microbenchmark(

  # Caso 1: mide el tiempo que tarda en ejecutarse el modelo logit
  logit = predecir_retencion(train, modelo = "logit"),

  # Caso 2: mide el tiempo que tarda en ejecutarse el modelo random forest
  rf = predecir_retencion(train, modelo = "rf"),

  # Ejecuta cada modelo 10 veces
  times = 10,
```

```
  # Mide el tiempo en milisegundos
  unit = "ms"
)
```

```
Precisión del modelo: 89.32 %
Precisión del modelo: 89.56 %
Precisión del modelo: 89.56 %
Precisión del modelo: 89.56 %
Precisión del modelo: 89.32 %
Precisión del modelo: 89.32 %
Precisión del modelo: 89.56 %
Precisión del modelo: 89.32 %
Precisión del modelo: 89.32 %
Precisión del modelo: 89.32 %
Precisión del modelo: 89.32 %
Precisión del modelo: 89.32 %
Precisión del modelo: 89.56 %
Precisión del modelo: 89.56 %
Precisión del modelo: 89.56 %
Precisión del modelo: 89.32 %
Precisión del modelo: 89.56 %
Precisión del modelo: 89.56 %
Precisión del modelo: 89.32 %
Precisión del modelo: 89.56 %
```

```
Unit: milliseconds
  expr        min         lq       mean     median         uq        max neval
 logit   112.6103   119.6619   126.7895   123.0139   135.1175   145.8892    10
    rf 10432.4037 10522.8176 10616.9790 10633.0396 10684.1042 10769.3538    10
```

Conclusiones

- La función `predecir_retencion()` es robusta y escalable.
- El modelo alcanza una precisión del 89.25% (logit) y 87.65% (random forest).
- Se documentó el proceso completo, incluyendo visualizaciones, cobertura y pruebas.

Repositorio GitHub: https://github.com/luismunizaga/Analisis-retencion-alumnos
Video explicativo: pendiente de grabación