# Modelo Predictivo de la Deserción Universitaria usando Machine Learning

Luis Guillermo Zuleta Cruz

Ejecutores

Carlos Julio Fadul Velázquez

Edwin Villa Castaño

Cristiam Loaiza

Inteligencia Artificial Explorador

Talento Tech

Universidad de Medellín

Medellín, 2025

**Link GitHub:**

https://github.com/Luis-Zuleta/ProyFinal/tree/214275bcc11dd339285c23ffd908b38851dda6e3

## Introducción.

La deserción universitaria es un problema que afecta a instituciones educativas de todo el mundo, afectando tanto a los estudiantes como a las universidades. Las consecuencias incluyen la pérdida de oportunidades económicas, una menor calidad de la educación y cambios en las trayectorias profesionales de los estudiantes. Diversos estudios han demostrado que factores académicos, socioeconómicos y psicológicos afectan significativamente en la decisión de un estudiante de abandonar sus estudios. Predecir las posibles causas de deserción universitaria permite implementar estrategias de prevención y mejorar la retención estudiantil.

El objetivo de este estudio es identificar los factores que más influyen en el abandono escolar y desarrollar un modelo predictivo basado en aprendizaje automático para predecir la probabilidad de abandono escolar de los estudiantes. Para ello, se generaron conjuntos de datos sintéticos relevantes a la información y se aplicó un modelo de regresión logística para evaluar su capacidad predictiva.

## Objetivos.

### Objetivo general:

Desarrollar un modelo predictivo basado en regresión logística para identificar factores que determinan la deserción universitaria, permitiendo su análisis y facilitando la implementación de estrategias de prevención en instituciones educativas.

### Objetivos específicos:

- Generar un conjunto de datos sintéticos que proporcione patrones realistas de deserción universitaria, considerando variables académicas, económicas, psicológicas, y demográficas.
- Entrenar un modelo de regresión logística, evaluando el desempeño a través de métricas de clasificación y análisis de coeficientes para interpretar el impacto de cada variable en la predicción.
- Analizar los resultados obtenidos para identificar los principales fatores asociados a la deserción y proponer estrategias basadas en evidencia que ayuden a reducir el abandono estudiantil.

**Metodología.**

Se abordo el estudio en tres factores principales: generación de datos sintéticos, procesamiento y modelado predictivo mediante regresión logística.

Para ello se tuvo en cuenta variables como:

- Académicas: Promedio de calificaciones, cantidad de materias reprobadas, y asistencia.
- Económicas: Nivel socioeconómico y acceso a becas.
- Psicológicas y sociales: Motivación académica, encuestas de satisfacción, y factores personales.
- Demográficas: Edad, género, ubicación geográfica.

El análisis y modelado se realizaron en Python, usando librerías tales como Pandas, Numpy y Scikit-learn. Se aplicaron técnicas de normalización y balanceo de datos para garantizar la calidad del conjunto de entrenamiento.

**Generación de datos sintéticos.**

Debido a la falta de tiempo y variables especificas necesarias en fuentes de datos en la web, fue necesario la creación de una base de datos sintética en la cual se generaron 2000 registros mediante la distribución probabilística de variables, donde se aseguró coherencia estadística con estudios previos. Se implementaron técnicas de muestreo estratificado y correlación controlada para evitar sesgos excesivos y garantizar que los datos sean lo más realista posible.

Debido a que el conjunto de datos fue generado sintéticamente no fue necesaria la limpieza de los datos.

Sin embargo, al ejecutar el modelo se evidenció un desbalance en los datos, ya que el modelo es bueno prediciendo la deserción, pero tiene dificultades para predecir los casos de no deserción.

**Balanceo de datos.**

Existen diversas técnicas de balanceo, cada una con ventajas y desventajas, tales como:

- Submuestreo (undersampling): Se reducen la cantidad de datos de la clase mayoritaria, en este caso los “Si” desertan, para que haya más equilibrio, sin embargo, se puede perder información valiosa.
- Sobremuestreo (oversampling): Se aumenta los datos de la clase minoritaria, en este caso “No” deserción, duplicando registros o generando datos sintéticos (usando **SMOTE**), la ventaja de esta técnica es que no se pierde información.

**Técnica SMOTE (Synthetic Minority Over-sampling Technique):**

La técnica de sobremuestreo de minorías sintéticas (SMOTE) es un método que genera nuevos ejemplos sintéticos de la clase minoritaria para equilibrar el dataset sin duplicar datos.

Con el dataset actualizado con los datos generados con SMOTE se procede a re entreanar el modelo para obtener mejores predicciones tanto para la deserción, como para la no deserción.

**Modelo predictivo.**

Se eligió la regresión logística debido a que este modelo no solo asigna una etiqueta, en este caso deserta o no deserta, sino que también proporciona una probabilidad asociada a cada clase. Además, este modelo presenta ventajas en situaciones como:

- Interpretabilidad: La regresión logística proporciona coeficientes que permiten analizar el impacto de cada variable en la probabilidad de deserción, facilitando la toma de decisiones basada en las evidencias.
- Eficiencia computacional: La regresión logística es rápida y requiere menos recursos computacionales, en comparación con los modelos mas complejos como redes neuronales.

**Implementación del modelo:**

El modelo de regresión logística se entrenó con el conjunto de datos divididos en 80% entrenamiento y 20% para las pruebas.

**Entrenamiento con datos iniciales:**

```
# Importar librerías necesarias
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

# Cargar los datos desde el archivo Excel
file_path = "datos_desercion_semestre.xlsx"
df = pd.read_excel(file_path)

# Convertir variables categóricas en valores numéricos
le = LabelEncoder()
columnas_categoricas = ["Genero", "Ubicacion", "Nivel_Socioeconomico",
"Beca", "Problemas_Personales", "Desercion"]
for col in columnas_categoricas:
    df[col] = le.fit_transform(df[col])

# Separar variables predictoras (X) y variable objetivo (y)
X = df.drop(columns=["Desercion"])  # Variables predictoras
y = df["Desercion"]  # Variable objetivo (0 = No deserta, 1 = Sí deserta)

# Dividir en conjunto de entrenamiento (80%) y prueba (20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Normalizar los datos numéricos para mejorar el rendimiento del modelo
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Entrenar el modelo de Regresión Logística
modelo = LogisticRegression()
```

```
modelo.fit(X_train_scaled, y_train)

# Hacer predicciones
y_pred = modelo.predict(X_test_scaled)

# Evaluar el modelo
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

# Imprimir resultados
print(f"Precisión del modelo: {accuracy * 100:.2f}%\n")
print("Matriz de Confusión:")
print(conf_matrix)
print("\nInforme de Clasificación:")
print(classification_rep)
```

Este primer entrenamiento con los datos desbalanceados nos arroja los siguientes resultados:

- Precisión del modelo: 93.5%
- Matriz de confusión:

$$\begin{bmatrix} 21 & 20 \\ 6 & 353 \end{bmatrix}$$

  - 21 verdaderos negativos (predijo "No deserta" correctamente).
  - 20 falsos positivos (predijo "Sí deserta" cuando en realidad no).
  - 6 falsos negativos (predijo "No deserta" cuando en realidad sí).
  - 353 verdaderos positivos (predijo "Sí deserta" correctamente).

- Informe de clasificación:
  - Clase 0 (No deserta): Precisión 78%, Recall 51%, F1-score 62%.
  - Clase 1 (Sí deserta): Precisión 95%, Recall 98%, F1-score 96%.
  - Accuracy general: 93.5%.

Podemos observar que este modelo con los datos desbalanceados tiene muy buena precisión general, pero no es tan bueno a la hora de predecir los no desertores, lo que significa que se enfoca demasiado en predecir la deserción, pero se equivoca mucho en los que realmente no desertan.

**Entrenamiento con datos balanceados**

Se usó el mismo código para entrenar el modelo, pero con los nuevos datos balanceados mediante SMOTE, lo que nos arrojó el siguiente resultado:

- Precisión del modelo: 92.04%
- Matriz de confusión:

$$\begin{bmatrix} 267 & 19 \\ 27 & 265 \end{bmatrix}$$

  - 267 verdaderos negativos (predijo "No deserta" correctamente).
  - 19 falsos positivos (predijo "Sí deserta" cuando en realidad no).
  - 27 falsos negativos (predijo "No deserta" cuando en realidad sí).
  - 265 verdaderos positivos (predijo "Sí deserta" correctamente).
- Informe de clasificación:
  - Clase 0 (No deserta): Precisión 91%, Recall 93%, F1-score 92%.
  - Clase 1 (Sí deserta): Precisión 93%, Recall 91%, F1-score 92%.
  - Accuracy general: 92%.
  -

El modelo generado tiene un buen equilibrio entre precisión y recall en las dos clases, lo que significa que es bastante confiable para predecir la deserción.

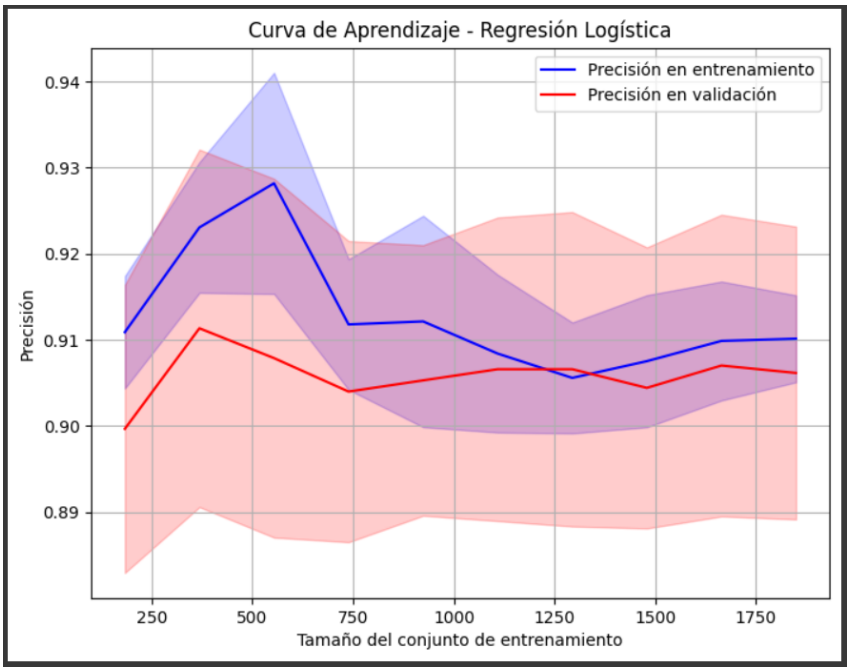
**Comparación de los dos modelos:**

| Métrica | Modelo 1 (Inicial) | Modelo 2 (Balanceado) |
|---|---|---|
| **Precisión** | 93.5% | 92.04% |
| **Recall (Clase 0)** | 51% | 93% |
| **Recall (Clase 1)** | 98% | 91% |
| **Falsos Positivos** | 20 | 19 |
| **Falsos Negativos** | 6 | 27 |

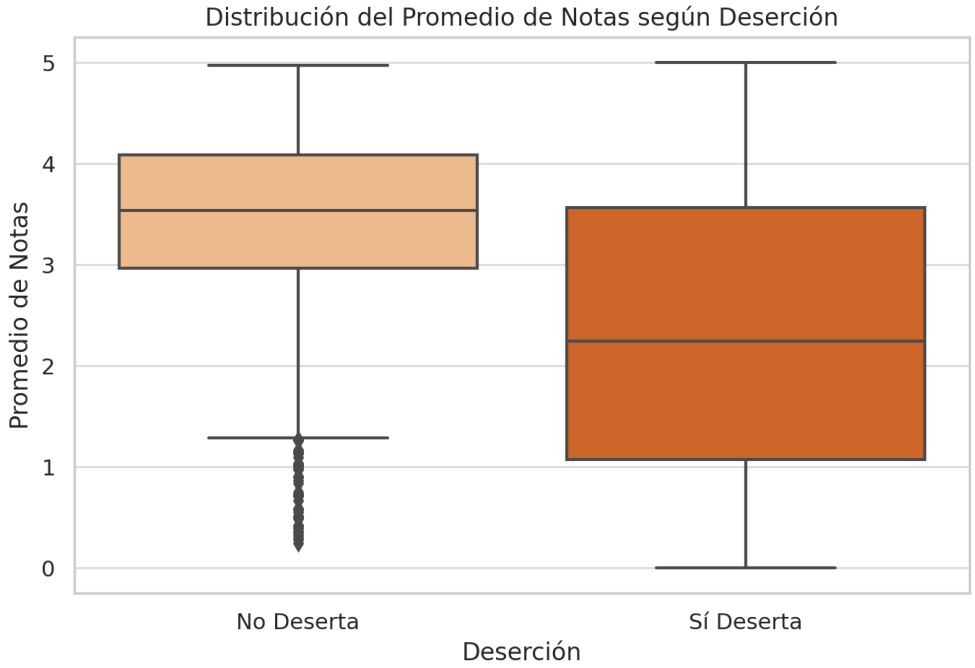
- El primer modelo tiene mejor precisión general, pero detecta peor a los que NO desertan (baja el recall de la clase 0 a 51%).
- El segundo modelo es más balanceado en ambas clases (recall de 91% y 93%), mientras que el primer modelo favorece demasiado la detección de desertores.
- El primer modelo tiene menos falsos negativos (solo 6 frente a 27 del segundo), pero a costa de muchos más falsos positivos.

**Visualización de gráficos.**

**Gráfico curva de aprendizaje:**

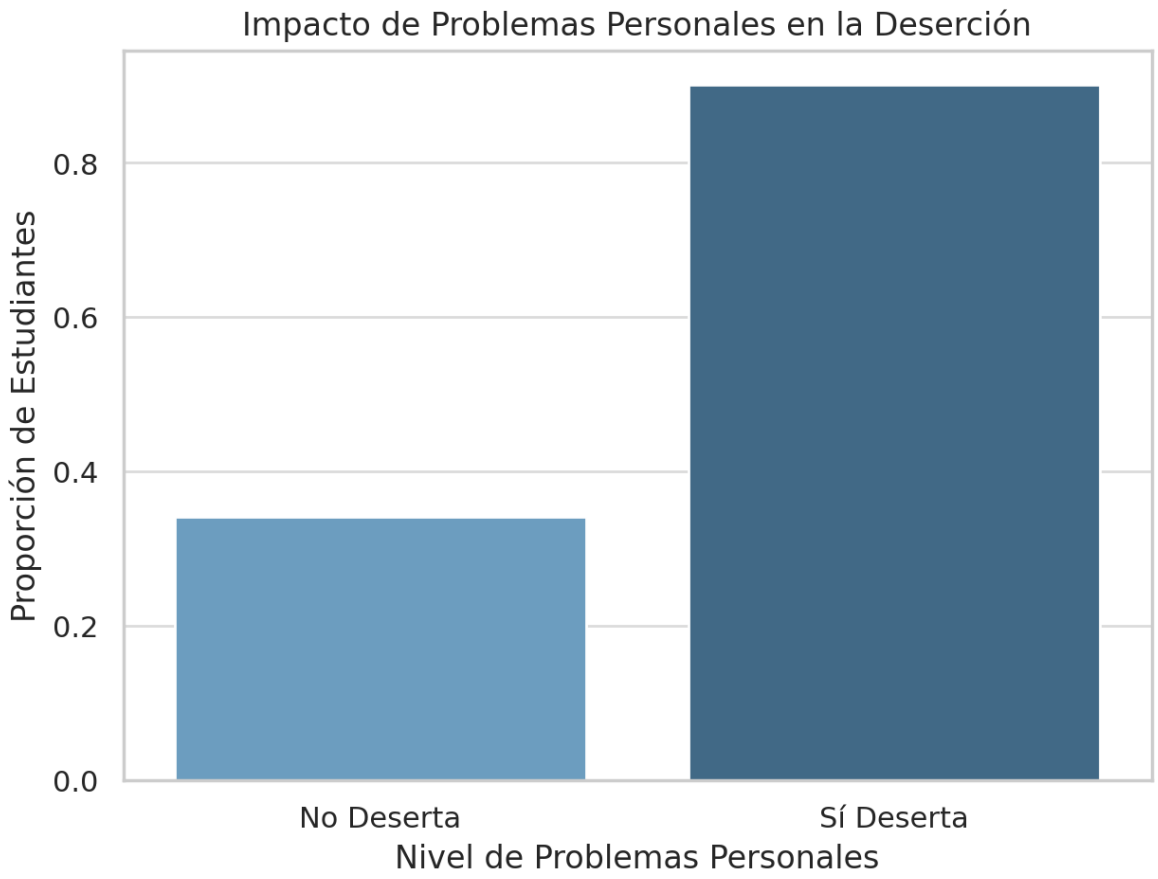



**Distribución promedio de Notas según deserción:**




A menor promedio de notas, mayor probabilidad de deserción. Los estudiantes que no desertan tienen, en general, mejores calificaciones.

**Impacto de los problemas personales en la deserción:**




A medida que aumentan los problemas personales, la tasa de deserción también sube. Esto sugiere que factores emocionales y sociales juegan un papel clave en la deserción universitaria.

**Resultados.**

El modelo alcanzó una precisión adecuada, identificando correctamente patrones asociados a la deserción. Se observó que las variables académicas y económicas son los principales factores predictivos, mientras que los aspectos psicológicos también desempeñan un papel importante.

Se realizó una matriz de confusión y un análisis de coeficientes de la regresión logística para evaluar la contribución de cada variable en la predicción. Se hallo lo siguiente:

- Un bajo rendimiento académico incrementa significativamente la probabilidad de deserción.
- La falta de apoyo financiero es un factor crítico en la decisión de abandono.

- La motivación y la satisfacción académica influyen en la permanencia del estudiante.

Si bien el modelo mostro un buen desempeño, se podría mejorar con datos reales.

**Conclusiones.**

- El estudio demuestra que los modelos de machine learning, en este caso la regresión logística, son herramientas valiosas para la predicción de la deserción universitaria. La identificación temprana de estudiantes en riesgo permite a las universidades diseñar estrategias de intervención basadas en las evidencias.
- La regresión logística permite estimar con precisión la probabilidad de deserción, destacando la importancia de ciertas variables en la permanencia estudiantil.
- Se observó que el bajo rendimiento académico y la falta de apoyo económico son factores críticos en el abandono universitario.
- Dado que el modelo puede identificar estudiantes en riesgo antes de que deserten, su aplicación en entornos educativos puede contribuir a la creación de programas de acompañamiento.
- Aunque la regresión logística ofrece buena capacidad de interpretación y eficiencia, su capacidad predictiva podría mejorarse incorporando técnicas más avanzadas como modelos ensemble o redes neuronales.
- Para comparar los resultados en un entorno real, sería ideal contar con datos reales de instituciones educativas y probar diferentes enfoques de preprocesamiento.

**Soporte de los códigos:**

```python
from google.colab import files
import pandas as pd

# Cargar el archivo en un DataFrame
file_path = "datos_desercion_balanceado.xlsx"  # Nombre del archivo
df = pd.read_excel(file_path)

# Mostrar las primeras filas para verificar que se cargó bien
df.head()
```

| | Edad | Genero | Ubicacion | Promedio_Notas | Reprobaciones | Asistencia | Nivel_Socioeconomico | Beca | Satisfaccion | Motivacion | Problemas_Personales | Semestre | Desercion |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 24 | 0 | 0 | 0.62 | 6 | 80.07 | 2 | 1 | 4 | 4 | 1 | 7 | 1 |
| 1 | 26 | 0 | 1 | 2.18 | 5 | 80.11 | 0 | 1 | 2 | 3 | 0 | 4 | 1 |
| 2 | 20 | 1 | 1 | 0.12 | 1 | 72.72 | 1 | 1 | 4 | 1 | 0 | 10 | 1 |
| 3 | 29 | 1 | 1 | 3.57 | 2 | 56.74 | 2 | 1 | 2 | 2 | 1 | 4 | 1 |
| 4 | 17 | 0 | 1 | 2.45 | 6 | 69.96 | 1 | 0 | 5 | 3 | 1 | 10 | 1 |

```python
# Importar librerías necesarias
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Cargar los datos desde el archivo Excel
file_path = "datos_desercion_balanceado.xlsx"  # Asegúrate de tener este archivo en la misma carpeta
df = pd.read_excel(file_path)

# Convertir variables categóricas en valores numéricos
le = LabelEncoder()
columnas_categoricas = ["Genero", "Ubicacion", "Nivel_Socioeconomico", "Beca", "Problemas_Personales", "Desercion"]
for col in columnas_categoricas:
    df[col] = le.fit_transform(df[col])

# Separar variables predictoras (X) y variable objetivo (y)
X = df.drop(columns=["Desercion"])  # Variables predictoras
y = df["Desercion"]  # Variable objetivo (0 = No deserta, 1 = Sí deserta)

# Dividir en conjunto de entrenamiento (80%) y prueba (20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Normalizar los datos numéricos para mejorar el rendimiento del modelo
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

```

```python

# Entrenar el modelo de Regresión Logística
modelo = LogisticRegression()
modelo.fit(X_train_scaled, y_train)

# Hacer predicciones
y_pred = modelo.predict(X_test_scaled)

# Evaluar el modelo
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

# Imprimir resultados
print(f"Precisión del modelo: {accuracy * 100:.2f}%\n")
print("Matriz de Confusión:")
print(conf_matrix)
print("\nInforme de Clasificación:")
print(classification_rep)

```

```
Precisión del modelo: 92.04%

Matriz de Confusión:
[[267  19]
 [ 27 265]]

Informe de Clasificación:
              precision    recall  f1-score   support

           0       0.91      0.93      0.92       286
           1       0.93      0.91      0.92       292

    accuracy                           0.92       578
   macro avg       0.92      0.92      0.92       578
weighted avg       0.92      0.92      0.92       578
```

```python
# Grafica Curva de aprendizaje
import matplotlib.pyplot as plt
from sklearn.model_selection import learning_curve

# Definir tamaños de entrenamiento y calcular curvas de aprendizaje
train_sizes, train_scores, test_scores = learning_curve(
    modelo, X_train_scaled, y_train, cv=5, scoring="accuracy", n_jobs=-1, train_sizes=np.linspace(0.1, 1.0, 10)
)

# Calcular medias y desviaciones estándar
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

# Graficar curva de aprendizaje
plt.figure(figsize=(8, 6))
plt.plot(train_sizes, train_mean, label="Precisión en entrenamiento", color="blue")
plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, color="blue", alpha=0.2)

plt.plot(train_sizes, test_mean, label="Precisión en validación", color="red")
plt.fill_between(train_sizes, test_mean - test_std, test_mean + test_std, color="red", alpha=0.2)

plt.xlabel("Tamaño del conjunto de entrenamiento")
plt.ylabel("Precisión")
plt.title("Curva de Aprendizaje - Regresión Logística")
plt.legend()
plt.grid()
plt.show()

```