

Abandono de carrera por estudiantes mediante implementación de clasificación con Feed-Forward en PyTorch

Asignatura: INF395 - Introducción a las Redes Neuronales y Deep Learning.

Integrantes:

- Bruno Morici - ROL: 202373555-8
- Juan Pablo Fuenzalida - ROL: 202373102-1
- Tutor: Alejandro Veloz

Usando el dataset de la UCL Machine Learning "Students' Dropout and Academic Success Dataset". Acceder mediante el siguiente link :

<https://archive.ics.uci.edu/dataset/697/predict+students+dropout+and+academic+!>

Importación e instalación de librerías necesarias

```
In [22]: %pip install pandas
          %pip install numpy
          %pip install scikit-learn
          %pip install matplotlib
          %pip install seaborn
          %pip install torch
          %pip install tqdm
          %pip install ucimlrepo
```

Requirement already satisfied: pandas in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (2.3.3)

Requirement already satisfied: numpy>=1.23.2 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from pandas) (2.3.4)

Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from pandas) (2.9.0.post0)

Requirement already satisfied: pytz>=2020.1 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from pandas) (2025.2)

Requirement already satisfied: tzdata>=2022.7 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from pandas) (2025.2)

Requirement already satisfied: six>=1.5 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)

Note: you may need to restart the kernel to use updated packages.

[notice] A new release of pip is available: 23.1.2 -> 25.3

[notice] To update, run: python.exe -m pip install --upgrade pip

Requirement already satisfied: numpy in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (2.3.4)

Note: you may need to restart the kernel to use updated packages.

[notice] A new release of pip is available: 23.1.2 -> 25.3

[notice] To update, run: python.exe -m pip install --upgrade pip

Requirement already satisfied: scikit-learn in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (1.7.2)

Requirement already satisfied: numpy>=1.22.0 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from scikit-learn) (2.3.4)

Requirement already satisfied: scipy>=1.8.0 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from scikit-learn) (1.16.3)

Requirement already satisfied: joblib>=1.2.0 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from scikit-learn) (1.5.2)

Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from scikit-learn) (3.6.0)

Note: you may need to restart the kernel to use updated packages.

[notice] A new release of pip is available: 23.1.2 -> 25.3

[notice] To update, run: python.exe -m pip install --upgrade pip

Requirement already satisfied: matplotlib in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (3.10.7)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cyclor>=0.10 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from matplotlib) (4.60.1)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from matplotlib) (1.4.9)
Requirement already satisfied: numpy>=1.23 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from matplotlib) (2.3.4)
Requirement already satisfied: packaging>=20.0 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from matplotlib) (12.0.0)
Requirement already satisfied: pyparsing>=3 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from matplotlib) (3.2.5)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from python-dateutil>=2.7->matplotlib) (1.17.0)
Note: you may need to restart the kernel to use updated packages.

[notice] A new release of pip is available: 23.1.2 -> 25.3

[notice] To update, run: python.exe -m pip install --upgrade pip

Requirement already satisfied: seaborn in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (0.13.2)

Requirement already satisfied: numpy!=1.24.0,>=1.20 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from seaborn) (2.3.4)

Requirement already satisfied: pandas>=1.2 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from seaborn) (2.3.3)

Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from seaborn) (3.10.7)

Requirement already satisfied: contourpy>=1.0.1 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.3.3)

Requirement already satisfied: cycler>=0.10 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)

Requirement already satisfied: fonttools>=4.22.0 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.60.1)

Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4.9)

Requirement already satisfied: packaging>=20.0 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (25.0)

Requirement already satisfied: pillow>=8 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (12.0.0)

Requirement already satisfied: pyparsing>=3 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.2.5)

Requirement already satisfied: python-dateutil>=2.7 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (2.9.0.post0)

Requirement already satisfied: pytz>=2020.1 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from pandas>=1.2->seaborn) (2025.2)

Requirement already satisfied: tzdata>=2022.7 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from pandas>=1.2->seaborn) (2025.2)

Requirement already satisfied: six>=1.5 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.17.0)

Note: you may need to restart the kernel to use updated packages.

[notice] A new release of pip is available: 23.1.2 -> 25.3

[notice] To update, run: python.exe -m pip install --upgrade pip

Requirement already satisfied: torch in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (2.9.1)
Requirement already satisfied: filelock in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from torch) (3.20.0)
Requirement already satisfied: typing-extensions>=4.10.0 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from torch) (4.15.0)
Requirement already satisfied: sympy>=1.13.3 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from torch) (1.14.0)
Requirement already satisfied: networkx>=2.5.1 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from torch) (3.5)
Requirement already satisfied: jinja2 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from torch) (3.1.6)
Requirement already satisfied: fsspec>=0.8.5 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from torch) (2025.10.0)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from sympy>=1.13.3->torch) (1.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from jinja2->torch) (3.0.3)
Note: you may need to restart the kernel to use updated packages.

[notice] A new release of pip is available: 23.1.2 -> 25.3

[notice] To update, run: python.exe -m pip install --upgrade pip

Requirement already satisfied: tqdm in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (4.67.1)
Requirement already satisfied: colorama in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from tqdm) (0.4.6)
Note: you may need to restart the kernel to use updated packages.

[notice] A new release of pip is available: 23.1.2 -> 25.3

[notice] To update, run: python.exe -m pip install --upgrade pip

Requirement already satisfied: ucimlrepo in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (0.0.7)

Requirement already satisfied: pandas>=1.0.0 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from ucimlrepo) (2.3.3)

Requirement already satisfied: certifi>=2020.12.5 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from ucimlrepo) (2025.10.5)

Requirement already satisfied: numpy>=1.23.2 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from pandas>=1.0.0->ucimlrepo) (2.3.4)

Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from pandas>=1.0.0->ucimlrepo) (2.9.0.post0)

Requirement already satisfied: pytz>=2020.1 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from pandas>=1.0.0->ucimlrepo) (2025.2)

Requirement already satisfied: tzdata>=2022.7 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from pandas>=1.0.0->ucimlrepo) (2025.2)

Requirement already satisfied: six>=1.5 in c:\users\bruno\desktop\inf395 - irn\proyecto_redes_neuronales\venv\lib\site-packages (from python-dateutil>=2.8.2->pandas>=1.0.0->ucimlrepo) (1.17.0)

Note: you may need to restart the kernel to use updated packages.

[notice] A new release of pip is available: 23.1.2 -> 25.3

[notice] To update, run: python.exe -m pip install --upgrade pip

```
In [23]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import Dataset, DataLoader
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.utils.class_weight import compute_class_weight
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from scipy.stats import zscore
from ucimlrepo import fetch_ucirepo
from tqdm import tqdm

# Configurar dispositivo
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print(f'Dispositivo utilizado: {device}')
```

Dispositivo utilizado: cpu

Limpieza y carga del Dataset

```
In [24]: # Cargar el dataset (por ID)
dataset = fetch_ucirepo(id=697)
```

```

# Acceder a los datos como DataFrame de pandas
df = dataset.data.features
target = dataset.data.targets

# Unimos features con targets en un solo DataFrame
df = pd.concat([df, target], axis=1)

# Mostrar las primeras filas
df.head()

```

Out[24]:

	Marital Status	Application mode	Application order	Course	Daytime/evening attendance	Previous qualification	q
0	1	17	5	171	1	1	
1	1	15	1	9254	1	1	
2	1	1	5	9070	1	1	
3	1	17	2	9773	1	1	
4	2	39	1	8014	0	1	

5 rows × 37 columns

Información del dataset:

In [25]: `df.info()`

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4424 entries, 0 to 4423
Data columns (total 37 columns):
 #   Column                                                                 Non-Null Count  Dtype
---  -
 0   Marital Status                                                         4424 non-null   int64
 1   Application mode                                                         4424 non-null   int64
 2   Application order                                                         4424 non-null   int64
 3   Course                                                                  4424 non-null   int64
 4   Daytime/evening attendance                                              4424 non-null   int64
 5   Previous qualification                                                  4424 non-null   int64
 6   Previous qualification (grade)                                          4424 non-null   float64
 7   Nacionality                                                            4424 non-null   int64
 8   Mother's qualification                                                  4424 non-null   int64
 9   Father's qualification                                                  4424 non-null   int64
10   Mother's occupation                                                     4424 non-null   int64
11   Father's occupation                                                     4424 non-null   int64
12   Admission grade                                                         4424 non-null   float64
13   Displaced                                                              4424 non-null   int64
14   Educational special needs                                              4424 non-null   int64
15   Debtor                                                                  4424 non-null   int64
16   Tuition fees up to date                                                4424 non-null   int64
17   Gender                                                                  4424 non-null   int64
18   Scholarship holder                                                     4424 non-null   int64
19   Age at enrollment                                                      4424 non-null   int64
20   International                                                           4424 non-null   int64
21   Curricular units 1st sem (credited)                                    4424 non-null   int64
22   Curricular units 1st sem (enrolled)                                    4424 non-null   int64
23   Curricular units 1st sem (evaluations)                                4424 non-null   int64
24   Curricular units 1st sem (approved)                                    4424 non-null   int64
25   Curricular units 1st sem (grade)                                        4424 non-null   float64
26   Curricular units 1st sem (without evaluations)                        4424 non-null   int64
27   Curricular units 2nd sem (credited)                                    4424 non-null   int64
28   Curricular units 2nd sem (enrolled)                                    4424 non-null   int64
29   Curricular units 2nd sem (evaluations)                                4424 non-null   int64
30   Curricular units 2nd sem (approved)                                    4424 non-null   int64
31   Curricular units 2nd sem (grade)                                        4424 non-null   float64
32   Curricular units 2nd sem (without evaluations)                        4424 non-null   int64
33   Unemployment rate                                                      4424 non-null   float64
34   Inflation rate                                                         4424 non-null   float64
35   GDP                                                                    4424 non-null   float64
36   Target                                                                  4424 non-null   object
dtypes: float64(7), int64(29), object(1)
memory usage: 1.2+ MB

```

Revisando si existen valores nulos en el dataset:

```
In [26]: df.isna().sum()
```



```

Out[26]: Marital Status                                0
         Application mode                              0
         Application order                             0
         Course                                         0
         Daytime/evening attendance                   0
         Previous qualification                         0
         Previous qualification (grade)                0
         Nacionality                                   0
         Mother's qualification                       0
         Father's qualification                       0
         Mother's occupation                           0
         Father's occupation                           0
         Admission grade                               0
         Displaced                                     0
         Educational special needs                     0
         Debtor                                         0
         Tuition fees up to date                      0
         Gender                                         0
         Scholarship holder                            0
         Age at enrollment                             0
         International                                 0
         Curricular units 1st sem (credited)           0
         Curricular units 1st sem (enrolled)           0
         Curricular units 1st sem (evaluations)        0
         Curricular units 1st sem (approved)           0
         Curricular units 1st sem (grade)              0
         Curricular units 1st sem (without evaluations) 0
         Curricular units 2nd sem (credited)           0
         Curricular units 2nd sem (enrolled)           0
         Curricular units 2nd sem (evaluations)        0
         Curricular units 2nd sem (approved)           0
         Curricular units 2nd sem (grade)              0
         Curricular units 2nd sem (without evaluations) 0
         Unemployment rate                             0
         Inflation rate                               0
         GDP                                             0
         Target                                         0
         dtype: int64

```

Convertimos los nombres de las columnas a minúsculas para facilitar su manejo:

```

In [27]: df.columns = [col.replace(' ', '_').replace("'s", '').replace('\t', '').replace
df.columns

```

```
Out[27]: Index(['marital_status', 'application_mode', 'application_order', 'course',
               'daytime_evening_attendance', 'previous_qualification',
               'previous_qualification_grade', 'nationality', 'mother_qualificatio
n',
               'father_qualification', 'mother_occupation', 'father_occupation',
               'admission_grade', 'displaced', 'educational_special_needs', 'debt_o
r',
               'tuition_fees_up_to_date', 'gender', 'scholarship_holder',
               'age_at_enrollment', 'international',
               'curricular_units_1st_sem_credited',
               'curricular_units_1st_sem_enrolled',
               'curricular_units_1st_sem_evaluations',
               'curricular_units_1st_sem_approved', 'curricular_units_1st_sem_grad
e',
               'curricular_units_1st_sem_without_evaluations',
               'curricular_units_2nd_sem_credited',
               'curricular_units_2nd_sem_enrolled',
               'curricular_units_2nd_sem_evaluations',
               'curricular_units_2nd_sem_approved', 'curricular_units_2nd_sem_grad
e',
               'curricular_units_2nd_sem_without_evaluations', 'unemployment_rate',
               'inflation_rate', 'gdp', 'target'],
              dtype='object')
```

Explicación para cada columna de la tabla:

Columna	Significado
<code>marital_status</code>	Estado civil del alumno/a al momento de la inscripción.
<code>application_mode</code>	Modo de aplicación al curso (por ejemplo, presencial vs. en línea, u otra categoría).
<code>application_order</code>	Orden de la solicitud de ingreso (por ejemplo, cuántas aplicaciones realizó antes, o prioridad de la solicitud).
<code>course</code>	Código o identificador del curso al que se inscribió el estudiante.
<code>daytime_evening_attendance</code>	Asistencia: diurna o vespertina (día vs. noche) para el curso.
<code>previous_qualification</code>	Tipo de cualificación/título anterior que tenía el estudiante antes de este nivel educativo.
<code>previous_qualification_grade</code>	Nota o calificación obtenida en la cualificación anterior.
<code>nationality</code>	Nacionalidad del estudiante.

Columna	Significado
<code>mother_qualification</code>	Nivel de formación de la madre del estudiante.
<code>father_qualification</code>	Nivel de formación del padre del estudiante.
<code>mother_occupation</code>	Ocupación laboral de la madre del estudiante.
<code>father_occupation</code>	Ocupación laboral del padre del estudiante.
<code>admission_grade</code>	Nota de admisión al curso/universidad del estudiante.
<code>displaced</code>	Si el estudiante es desplazado (por ejemplo, geográfica o socialmente) o no.
<code>educational_special_needs</code>	Si el estudiante tiene necesidades educativas especiales.
<code>debtor</code>	Si el estudiante adeuda pagos (por ejemplo, matrícula u otros cargos) o no.
<code>tuition_fees_up_to_date</code>	Si las tasas de matrícula están al día o no.
<code>gender</code>	Género del estudiante.
<code>scholarship_holder</code>	Si el estudiante tiene una beca o no.
<code>age_at_enrollment</code>	Edad del estudiante al momento de la inscripción.
<code>international</code>	Si el estudiante es internacional (extranjero) o no.
<code>curricular_units_1st_sem_credited</code>	Unidades curriculares acreditadas en el primer semestre.
<code>curricular_units_1st_sem_enrolled</code>	Unidades curriculares en que se matriculó el estudiante en el primer semestre.
<code>curricular_units_1st_sem_evaluations</code>	Número de unidades curriculares que fueron evaluadas en el primer semestre.
<code>curricular_units_1st_sem_approved</code>	Unidades curriculares aprobadas en el primer semestre.

Columna	Significado
<code>curricular_units_1st_sem_grade</code>	Nota promedio o de alguna unidad en el primer semestre (o quizá nota global del semestre).
<code>curricular_units_1st_sem_without_evaluations</code>	Unidades curriculares del primer semestre sin evaluación (por ejemplo exentas u otro motivo).
<code>curricular_units_2nd_sem_credited</code>	Unidades curriculares acreditadas en el segundo semestre.
<code>curricular_units_2nd_sem_enrolled</code>	Unidades curriculares en que se matriculó el estudiante en el segundo semestre.
<code>curricular_units_2nd_sem_evaluations</code>	Número de unidades curriculares que fueron evaluadas en el segundo semestre.
<code>curricular_units_2nd_sem_approved</code>	Unidades curriculares aprobadas en el segundo semestre.
<code>curricular_units_2nd_sem_grade</code>	Nota promedio o de alguna unidad en el segundo semestre (o nota global).
<code>curricular_units_2nd_sem_without_evaluations</code>	Unidades curriculares del segundo semestre sin evaluación.
<code>unemployment_rate</code>	Tasa de desempleo correspondiente al contexto económico del estudiante (o región) en el momento.
<code>inflation_rate</code>	Tasa de inflación correspondiente al contexto económico del estudiante (o región) en el momento.
<code>gdp</code>	Producto interno bruto (PIB) correspondiente al contexto económico del estudiante (o región) en el momento.
<code>target</code>	Si el estudiante abandonó, se graduó o sigue cursando en la actualidad.

Análisis Exploratorio de datos

```
In [28]: df.describe()
```

Out[28]:

	marital_status	application_mode	application_order	course	dayti
count	4424.000000	4424.000000	4424.000000	4424.000000	
mean	1.178571	18.669078	1.727848	8856.642631	
std	0.605747	17.484682	1.313793	2063.566416	
min	1.000000	1.000000	0.000000	33.000000	
25%	1.000000	1.000000	1.000000	9085.000000	
50%	1.000000	17.000000	1.000000	9238.000000	
75%	1.000000	39.000000	2.000000	9556.000000	
max	6.000000	57.000000	9.000000	9991.000000	

8 rows × 36 columns

Eliminamos Outliers con el método de Z-Score

```
In [29]: threshold = 3

z_scores = zscore(df.select_dtypes(include=[np.number]))
outliers = (np.abs(z_scores) > threshold).any(axis=1)
df_2 = df[~outliers].copy()

print(f"Se eliminaron { len(df) - len(df_2) } filas del dataset")
df_2
```

Se eliminaron 1343 filas del dataset

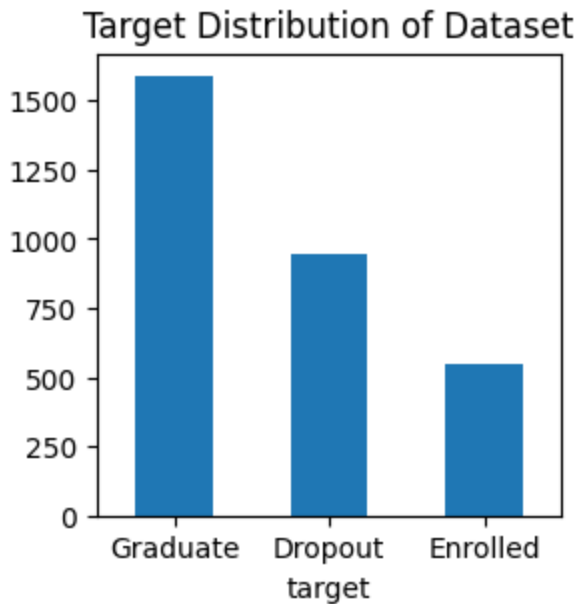
Out[29]:

	marital_status	application_mode	application_order	course	daytime_ev
1	1	15	1	9254	
2	1	1	5	9070	
3	1	17	2	9773	
4	2	39	1	8014	
6	1	1	1	9500	
...	
4410	1	1	1	9070	
4412	1	43	1	9070	
4414	1	1	1	9130	
4417	1	1	1	9070	
4421	1	1	1	9500	

3081 rows × 37 columns

Verificamos que no haya desbalanceo de clases

```
In [30]: df_2['target'].value_counts()
df_2["target"].value_counts().plot(kind="bar", figsize=(3,3), title="Target
plt.xticks(rotation=0)
plt.show()
```



Como existe desbalanceo de clases, usaremos el método de **Balanceo Artificial por vía de Modificación de Función Objetivo** para las muestras que sean de las clases "Dropout" y "Enrolled"

Separación del dataset en entrenamiento y testing

Eliminamos la columna de Enrolled, ya que no aporta información relevante para el análisis

```
In [31]: df_2 = df_2.drop(df_2[df_2['target'] == 'Enrolled'].index)

# Verificamos que se eliminó
print(df_2['target'].unique())
```

```
['Graduate' 'Dropout']
```

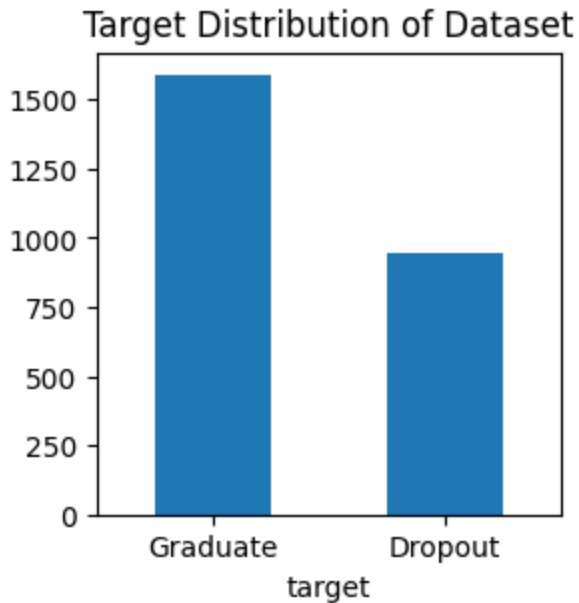
Le asignamos un número a cada clase para que PyTorch pueda trabajar con ellas:

```
In [33]: class_names, y = np.unique(df_2['target'], return_inverse=True)
print(y)
print(class_names)
X = df_2.drop('target', axis=1)
```

```
[1 0 1 ... 1 1 0]
['Dropout' 'Graduate']
```

Visualización de la distribución de la variable objetivo 'target' sin 'Enrolled'

```
In [34]: df_2['target'].value_counts()
df_2["target"].value_counts().plot(kind="bar", figsize=(3,3), title="Target
plt.xticks(rotation=0)
plt.show()
```



One Hot Encoding

```
In [35]: one_hot_cols = ['marital_status',
                        'daytime_evening_attendance',
                        'tuition_fees_up_to_date',
                        'educational_special_needs',
                        'displaced',
                        'scholarship_holder',
                        'gender',
                        'debtor',
                        'international',
                        ]

for i in one_hot_cols:
    X[i] = X[i].apply(lambda x: float(x))
    X[i] = X[i].apply(lambda x: int(x))

for i in one_hot_cols:
    dum = pd.get_dummies(X[i], prefix=i)
    X.drop([i], axis=1, inplace=True)
    X = pd.concat([X, dum], axis=1)
```

Dividimos el Dataset en entrenamiento y testing

```
In [36]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rar
X_train_corr = X_train.copy()
```

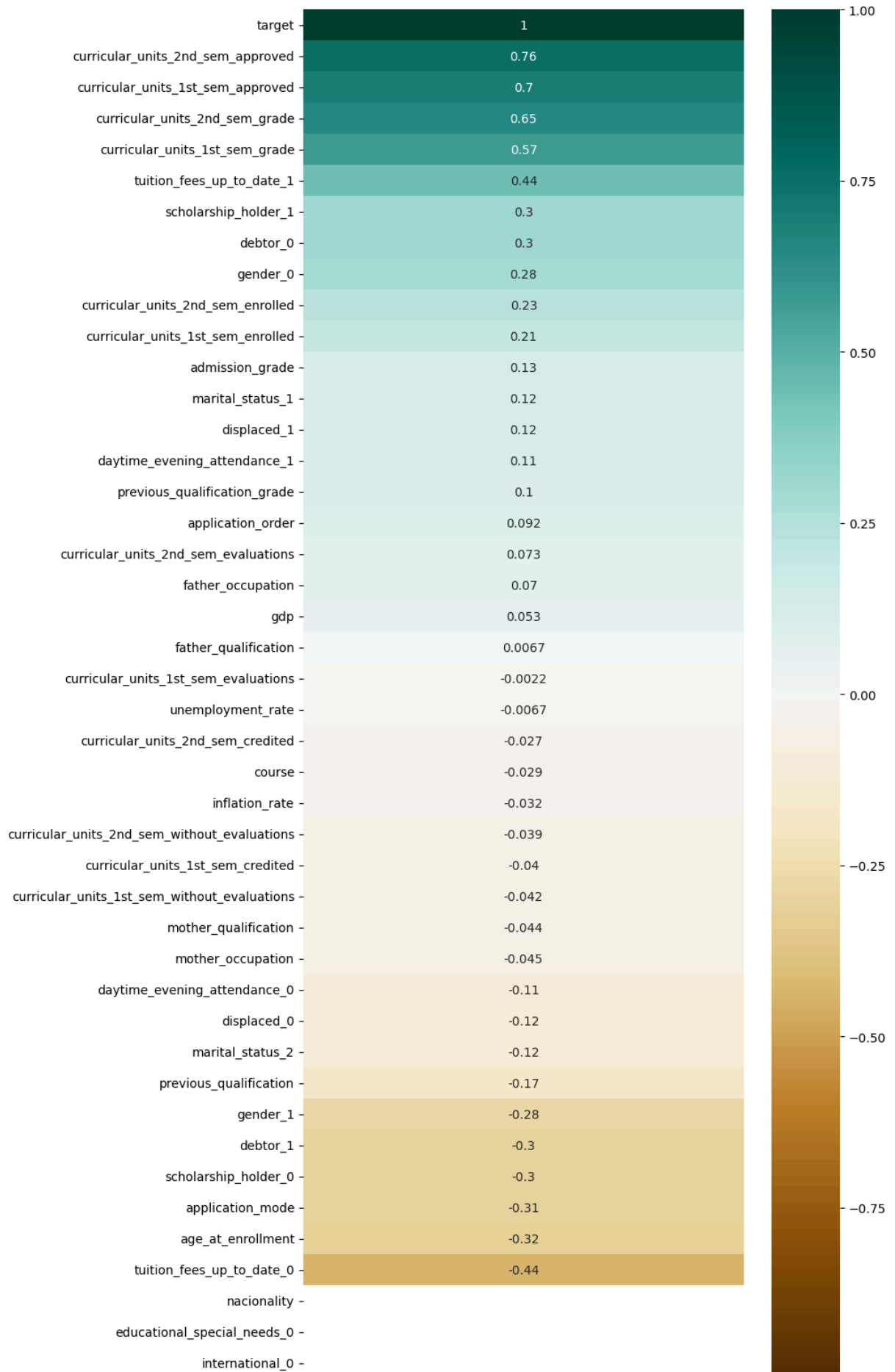
```
X_train_corr['target'] = y_train
```

Mapa de correlación de la variable target en base a las diferentes columnas de el Dataset

```
In [37]: plt.figure(figsize=(8, 20))  
heatmap = sns.heatmap(X_train_corr.corr()[['target']].sort_values(by='target'  
heatmap.set_title('Features Correlation with target', fontdict={'fontsize':1
```

```
Out[37]: Text(0.5, 1.0, 'Features Correlation with target')
```


Features Correlation with target



Seleccionamos las features que deseamos analizar.

```
In [38]: features_list = ['curricular_units_2nd_sem_approved',
                        'curricular_units_1st_sem_approved',
                        'curricular_units_2nd_sem_grade',
                        'curricular_units_1st_sem_grade',
                        'tuition_fees_up_to_date_1',
                        'scholarship_holder_1',
                        'debtor_0',
                        'gender_0',
                        'curricular_units_2nd_sem_enrolled',
                        'curricular_units_1st_sem_enrolled',
                        'tuition_fees_up_to_date_0',
                        'age_at_enrollment',
                        'application_mode',
                        'scholarship_holder_0',
                        'debtor_1',
                        'gender_1']
```

Recortamos el entrenamiento y testing a esas features

```
In [39]: X_train_fs = X_train[features_list]
        X_test_fs = X_test[features_list]
```

Normalizamos de forma estándar los datos, con el fin de no darle más relevancia al número por su tamaño en sí, si no que por su impacto respecto a los otros datos

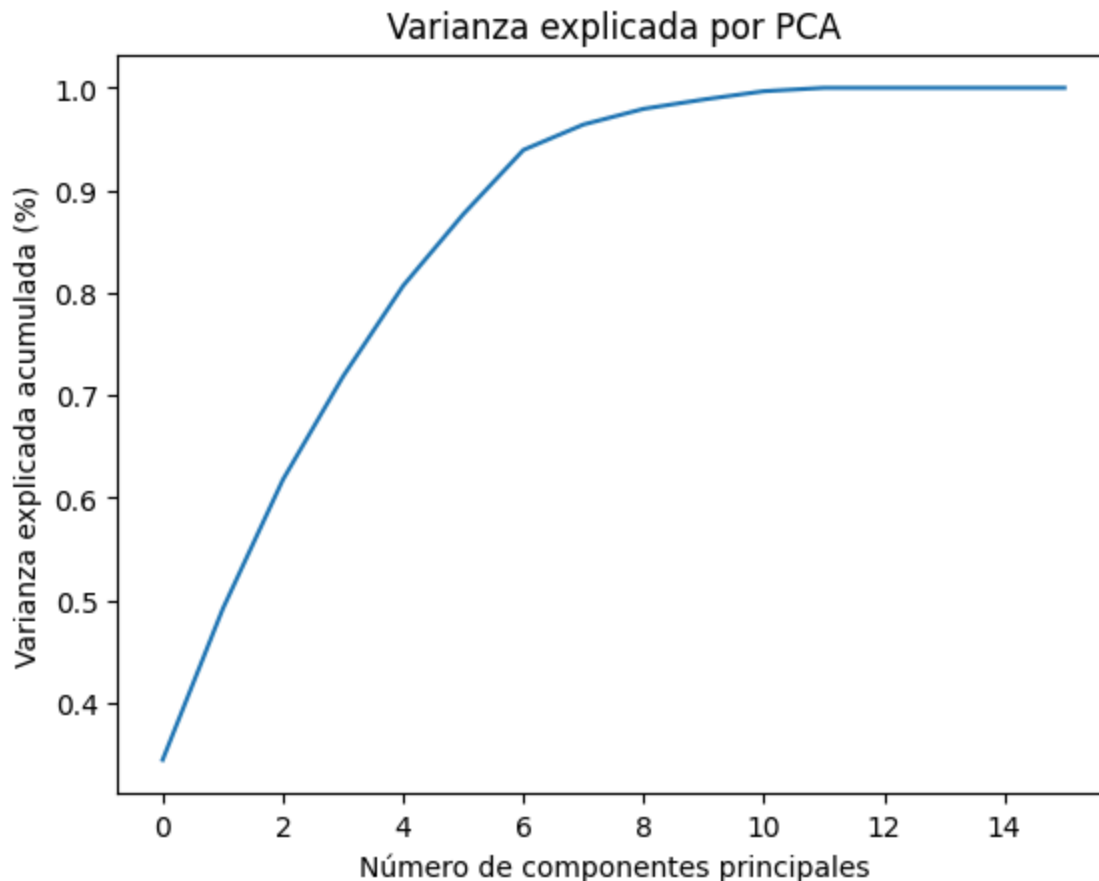
```
In [40]: scaler = StandardScaler()
        scaler.fit(X_train_fs)

        X_train_sc = scaler.transform(X_train_fs)
        X_test_sc = scaler.transform(X_test_fs)
        X_train_sc = pd.DataFrame(X_train_sc, columns=X_train_fs.columns)
        X_test_sc = pd.DataFrame(X_test_sc, columns=X_test_fs.columns)
```

Aplicamos PCA para disminuir la complejidad del modelo

```
In [41]: # Ajustar el modelo PCA a los datos de entrenamiento ya escalados
        pca = PCA().fit(X_train_sc)

        # Crear una figura y graficar la varianza acumulada explicada por los compor
        plt.figure()
        plt.plot(np.cumsum(pca.explained_variance_ratio_))
        plt.xlabel('Número de componentes principales')
        plt.ylabel('Varianza explicada acumulada (%)')
        plt.title('Varianza explicada por PCA')
        plt.show()
```



```
In [42]: num_pca = 16
pca = PCA(n_components=num_pca, whiten=True).fit(X_train_sc)
X_train_pc = pca.transform(X_train_sc)
X_test_pc = pca.transform(X_test_sc)
```

Balanceamos mediante modificación de la Función Objetivo, haciendo variar los pesos

```
In [43]: classes = np.unique(y_train)
weights = compute_class_weight(class_weight='balanced', classes=classes, y=y)
class_weights = dict(zip(classes, weights))
class_weights
```

```
Out[43]: {np.int64(0): np.float64(1.3448275862068966),
np.int64(1): np.float64(0.7959183673469388)}
```

Modelo de predicción

Configuración de Hiperparámetros

Definimos los hiperparámetros que definirán la creación y modelamiento de nuestro modelo de predicción, siendo estas:

1. input_size que define el tamaño de la primera capa de la red neuronal.

2. `num_layers` especifica el número de capas ocultas en la arquitectura de la red neuronal.
3. `hidden_sizes` define el número de neuronas de cada capa oculta del modelo.
4. `dropout_rates` especifica la tasa de dropout para cada capa oculta, siendo esta una técnica de regularización que desactiva aleatoriamente un porcentaje de neuronas durante el entrenamiento para prevenir el sobreajuste.
5. `activation` define la función de activación utilizada en las capas ocultas.
6. `learning_rate` controla el tamaño de los pasos que toma el optimizador durante el entrenamiento.
7. `batch_size` define el número de muestras que se procesan antes de actualizar los pesos del modelo.
8. `num_epochs` especifica el número máximo de iteraciones completas sobre todo el conjunto de entrenamiento.
9. `weight_decay` implementa regularización L2, que penaliza los pesos grandes del modelo, previniendo el sobreajuste.
10. `patience` define el número de épocas que el modelo puede entrenar sin mejorar antes de activar el early stopping.

```
In [44]: HYPERPARAMETERS = {  
    'input_size': 16,  
    'num_layers': 4,  
    'hidden_sizes': [128, 64, 32, 16],  
    'dropout_rates': [0.4, 0.3, 0.2, 0.1],  
    'activation': 'relu',  
    'learning_rate': 0.0005,  
    'batch_size': 32,  
    'num_epochs': 200,  
    'weight_decay': 0.0005,  
    'patience': 25,  
}
```

Resumen del Modelo

Nuestro modelo de predicción es una red neuronal feedforward diseñada para clasificar estudiantes en tres categorías: Dropout, Enrolled y Graduate.

Utiliza dropout progresivo para regularización efectiva, además implementa función de activación ReLU para introducir no-linealidad. Aplica Softmax en la salida para generar probabilidades de pertenencia a cada clase, para finalmente producir una distribución de probabilidades sobre las tres clases posibles.

```

In [45]: class StudentDropoutModel(nn.Module):
    def __init__(self, input_size, hidden_sizes, dropout_rates, activation='relu'):
        super(StudentDropoutModel, self).__init__()

        self.layers = nn.ModuleList()
        self.dropouts = nn.ModuleList()

        if activation == 'relu':
            self.activation = nn.ReLU()
        elif activation == 'tanh':
            self.activation = nn.Tanh()
        elif activation == 'sigmoid':
            self.activation = nn.Sigmoid()
        else:
            self.activation = nn.ReLU()

        prev_size = input_size
        for hidden_size, dropout_rate in zip(hidden_sizes, dropout_rates):
            self.layers.append(nn.Linear(prev_size, hidden_size))
            self.dropouts.append(nn.Dropout(dropout_rate))
            prev_size = hidden_size

        self.output = nn.Linear(prev_size, num_classes)
        self.softmax = nn.Softmax(dim=1)

    def forward(self, x):
        for layer, dropout in zip(self.layers, self.dropouts):
            x = layer(x)
            x = self.activation(x)
            x = dropout(x)

        x = self.output(x)
        return self.softmax(x)

    def predict_class(self, x):
        with torch.no_grad():
            probs = self.forward(x)
            return torch.argmax(probs, dim=1)

model = StudentDropoutModel(
    input_size=HYPERPARAMETERS['input_size'],
    hidden_sizes=HYPERPARAMETERS['hidden_sizes'],
    dropout_rates=HYPERPARAMETERS['dropout_rates'],
    activation=HYPERPARAMETERS['activation'],
    num_classes=2
).to(device)

print(f"\nArquitectura del modelo:")
print(model)
print(f"\nNúmero total de parámetros: {sum(p.numel() for p in model.parameters())}")

```

Arquitectura del modelo:

```
StudentDropoutModel(
  (layers): ModuleList(
    (0): Linear(in_features=16, out_features=128, bias=True)
    (1): Linear(in_features=128, out_features=64, bias=True)
    (2): Linear(in_features=64, out_features=32, bias=True)
    (3): Linear(in_features=32, out_features=16, bias=True)
  )
  (dropouts): ModuleList(
    (0): Dropout(p=0.4, inplace=False)
    (1): Dropout(p=0.3, inplace=False)
    (2): Dropout(p=0.2, inplace=False)
    (3): Dropout(p=0.1, inplace=False)
  )
  (activation): ReLU()
  (output): Linear(in_features=16, out_features=2, bias=True)
  (softmax): Softmax(dim=1)
)
```

Número total de parámetros: 13,074

Dataset y DataLoader para PyTorch

Preparamos los datasets apropiados para el modelo para el set de entrenamiento y testeo, y sus respectivo dataloaders.

```
In [46]: # Clase Dataset personalizada
class StudentDataset(Dataset):
    def __init__(self, features, labels):
        self.features = torch.FloatTensor(features)
        self.labels = torch.LongTensor(labels)

    def __len__(self):
        return len(self.labels)

    def __getitem__(self, idx):
        return self.features[idx], self.labels[idx]

# Crear datasets
train_dataset = StudentDataset(X_train_pc, y_train)
test_dataset = StudentDataset(X_test_pc, y_test)

# Crear DataLoaders
train_loader = DataLoader(
    train_dataset,
    batch_size=HYPERPARAMETERS['batch_size'],
    shuffle=True
)
test_loader = DataLoader(
    test_dataset,
    batch_size=HYPERPARAMETERS['batch_size'],
    shuffle=False
)
```

```
print(f"Tamaño del conjunto de entrenamiento: {len(train_dataset)}")
print(f"Tamaño del conjunto de prueba: {len(test_dataset)}")
```

Tamaño del conjunto de entrenamiento: 2028

Tamaño del conjunto de prueba: 507

Entrenamiento del modelo

El proceso de entrenamiento del modelo implementa balanceo de clases, utilizando pesos de clases calculados para manejar el desbalance en el dataset, la función de pérdida CrossEntropyLoss con pesos de clases para penalizar más los errores en clases desbalanceadas. Utilizamos el optimizador Adam para prevenir sobreajuste y un Learning Rate Scheduler ReduceLROnPlateau que reduce el learning rate automáticamente cuando la pérdida de validación deja de mejorar, finalmente el Early Stopping, que detiene el entrenamiento si no hay mejora en la pérdida de validación por 25 épocas consecutivas, restaurando automáticamente el mejor modelo encontrado.

Podemos describir el proceso de entrenamiento en tres fases:

1. Fase de entrenamiento:

- El modelo procesa batches de datos de entrenamiento
- Calcula la pérdida mediante forward pass
- Actualiza los pesos mediante backward pass y optimizador
- Calcula precisión y pérdida promedio

2. Fase de validación:

- Evalúa el modelo en el conjunto de validación sin actualizar pesos
- Calcula métricas de validación
- Actualiza el learning rate scheduler basado en la pérdida de validación

3. Monitoreo y control:

- Guarda el mejor modelo basado en la pérdida de validación
- Implementa early stopping para prevenir sobreajuste
- Muestra métricas en tiempo real mediante barras de progreso

```
In [47]: class_weights_tensor = torch.FloatTensor([class_weights[i] for i in sorted(c
criterion = nn.CrossEntropyLoss(weight=class_weights_tensor)

optimizer = optim.Adam(
    model.parameters(),
    lr=HYPERPARAMETERS['learning_rate'],
    weight_decay=HYPERPARAMETERS['weight_decay']
)

scheduler = optim.lr_scheduler.ReduceLROnPlateau(
    optimizer, mode='min', factor=0.5, patience=10
)
```

```

train_losses = []
train_accuracies = []
val_losses = []
val_accuracies = []
best_val_loss = float('inf')
best_val_acc = 0.0
patience_counter = 0
best_model_state = None
best_epoch = 0

print("Iniciando entrenamiento...\n")
epoch_pbar = tqdm(range(HYPERPARAMETERS['num_epochs']), desc='Entrenamiento')

for epoch in epoch_pbar:
    model.train()
    train_loss = 0.0
    train_correct = 0
    train_total = 0
    train_pbar = tqdm(train_loader, desc=f'Epoch {epoch+1}/{HYPERPARAMETERS["num_epochs"]}')
    for features, labels in train_pbar:
        features, labels = features.to(device), labels.to(device)
        optimizer.zero_grad()
        outputs = model(features)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        train_loss += loss.item()
        _, predicted = torch.max(outputs.data, 1)
        train_total += labels.size(0)
        train_correct += (predicted == labels).sum().item()
        train_acc_batch = 100 * train_correct / train_total
        train_pbar.set_postfix({
            'Loss': f'{loss.item():.4f}',
            'Acc': f'{train_acc_batch:.2f}%'
        })

    train_loss /= len(train_loader)
    train_acc = 100 * train_correct / train_total
    train_losses.append(train_loss)
    train_accuracies.append(train_acc)
    model.eval()
    val_loss = 0.0
    val_correct = 0
    val_total = 0

    val_pbar = tqdm(test_loader, desc=f'Epoch {epoch+1}/{HYPERPARAMETERS["num_epochs"]}',
                    leave=False, unit='batch')

    with torch.no_grad():
        for features, labels in val_pbar:
            features, labels = features.to(device), labels.to(device)
            outputs = model(features)
            loss = criterion(outputs, labels)

            val_loss += loss.item()
            _, predicted = torch.max(outputs.data, 1)

```



```

        val_total += labels.size(0)
        val_correct += (predicted == labels).sum().item()

    val_acc_batch = 100 * val_correct / val_total
    val_pbar.set_postfix({
        'Loss': f'{loss.item():.4f}',
        'Acc': f'{val_acc_batch:.2f}%'
    })

val_loss /= len(test_loader)
val_acc = 100 * val_correct / val_total
val_losses.append(val_loss)
val_accuracies.append(val_acc)

epoch_pbar.set_postfix({
    'Train Loss': f'{train_loss:.4f}',
    'Train Acc': f'{train_acc:.2f}%',
    'Val Loss': f'{val_loss:.4f}',
    'Val Acc': f'{val_acc:.2f}%',
    'Best Val Acc': f'{best_val_acc:.2f}%',
    'Patience': f'{patience_counter}/{HYPERPARAMETERS["patience"]}'
})

if val_loss < best_val_loss:
    improvement = best_val_loss - val_loss
    best_val_loss = val_loss
    best_val_acc = val_acc
    patience_counter = 0
    best_model_state = model.state_dict().copy()
    best_epoch = epoch + 1
    epoch_pbar.write(f'✓ Mejora en epoch {epoch+1}: Val Loss mejoró de {
else:
    patience_counter += 1
    if patience_counter >= HYPERPARAMETERS['patience']:
        epoch_pbar.write(f'\n⚠ Early stopping activado en epoch {epoch+1}
        epoch_pbar.write(f'    Mejor modelo encontrado en epoch {best_epc
        epoch_pbar.write(f'    Mejor Val Loss: {best_val_loss:.4f}')
        epoch_pbar.write(f'    Mejor Val Acc: {best_val_acc:.2f}%',
        epoch_pbar.write(f'    Razón: No hubo mejora en Val Loss por {HYPERPARAMETERS["patience"]} épocas')
        epoch_pbar.write(f'    El modelo se detuvo para prevenir sobreajuste')
        model.load_state_dict(best_model_state)
        break

epoch_pbar.close()
print("\n" + "="*60)
print("ENTRENAMIENTO COMPLETADO")
print("="*60)
print(f"Mejor modelo guardado de epoch {best_epoch}")
print(f"Mejor Val Loss: {best_val_loss:.4f}")
print(f"Mejor Val Acc: {best_val_acc:.2f}%")
print(f"Total de épocas entrenadas: {len(train_losses)}")

```

Iniciando entrenamiento...

Entrenamiento: 0%| | 1/200 [00:00<01:56, 1.71epoch/s, Train Loss=0.6806, Train Acc=37.82%, Val Loss=0.6427, Val Acc=40.43%, Best Val Acc=0.00%, Patience=0/25]
✓ Mejora en epoch 1: Val Loss mejoró de inf a 0.6427 (Val Acc: 40.43%)

Entrenamiento: 1%| | 2/200 [00:01<02:00, 1.65epoch/s, Train Loss=0.5704, Train Acc=76.87%, Val Loss=0.4459, Val Acc=91.72%, Best Val Acc=40.43%, Patience=0/25]
✓ Mejora en epoch 2: Val Loss mejoró de 0.6427 a 0.4459 (Val Acc: 91.72%)

Entrenamiento: 2%|| | 3/200 [00:01<01:55, 1.70epoch/s, Train Loss=0.4466, Train Acc=89.00%, Val Loss=0.4040, Val Acc=92.11%, Best Val Acc=91.72%, Patience=0/25]
✓ Mejora en epoch 3: Val Loss mejoró de 0.4459 a 0.4040 (Val Acc: 92.11%)

Entrenamiento: 2%|| | 4/200 [00:02<01:56, 1.68epoch/s, Train Loss=0.4260, Train Acc=90.43%, Val Loss=0.3989, Val Acc=91.91%, Best Val Acc=92.11%, Patience=0/25]
✓ Mejora en epoch 4: Val Loss mejoró de 0.4040 a 0.3989 (Val Acc: 91.91%)

Entrenamiento: 3%|| | 6/200 [00:03<02:06, 1.53epoch/s, Train Loss=0.4185, Train Acc=90.83%, Val Loss=0.3985, Val Acc=92.31%, Best Val Acc=91.91%, Patience=1/25]
✓ Mejora en epoch 6: Val Loss mejoró de 0.3989 a 0.3985 (Val Acc: 92.31%)

Entrenamiento: 4%|| | 8/200 [00:04<01:59, 1.60epoch/s, Train Loss=0.4133, Train Acc=91.17%, Val Loss=0.3965, Val Acc=91.91%, Best Val Acc=92.31%, Patience=1/25]
✓ Mejora en epoch 8: Val Loss mejoró de 0.3985 a 0.3965 (Val Acc: 91.91%)

Entrenamiento: 4%|| | 9/200 [00:05<01:55, 1.65epoch/s, Train Loss=0.4095, Train Acc=91.47%, Val Loss=0.3955, Val Acc=92.31%, Best Val Acc=91.91%, Patience=0/25]
✓ Mejora en epoch 9: Val Loss mejoró de 0.3965 a 0.3955 (Val Acc: 92.31%)

Entrenamiento: 6%|| | 12/200 [00:07<01:59, 1.57epoch/s, Train Loss=0.4064, Train Acc=92.26%, Val Loss=0.3947, Val Acc=92.11%, Best Val Acc=92.31%, Patience=2/25]
✓ Mejora en epoch 12: Val Loss mejoró de 0.3955 a 0.3947 (Val Acc: 92.11%)

Entrenamiento: 6%|| | 13/200 [00:08<01:57, 1.59epoch/s, Train Loss=0.4072, Train Acc=91.77%, Val Loss=0.3918, Val Acc=93.29%, Best Val Acc=92.11%, Patience=0/25]
✓ Mejora en epoch 13: Val Loss mejoró de 0.3947 a 0.3918 (Val Acc: 93.29%)

Entrenamiento: 8%|| | 15/200 [00:09<01:49, 1.69epoch/s, Train Loss=0.4012, Train Acc=92.36%, Val Loss=0.3912, Val Acc=93.10%, Best Val Acc=93.29%, Patience=1/25]
✓ Mejora en epoch 15: Val Loss mejoró de 0.3918 a 0.3912 (Val Acc: 93.10%)

Entrenamiento: 8%|| | 17/200 [00:10<01:37, 1.87epoch/s, Train Loss=0.4026, Train Acc=91.81%, Val Loss=0.3875, Val Acc=93.29%, Best Val Acc=93.10%, Patience=1/25]
✓ Mejora en epoch 17: Val Loss mejoró de 0.3912 a 0.3875 (Val Acc: 93.29%)

Entrenamiento: 12%|| | 23/200 [00:13<01:29, 1.99epoch/s, Train Loss=0.4019, Train Acc=91.81%, Val Loss=0.3857, Val Acc=93.49%, Best Val Acc=93.29%, Patience=5/25]
✓ Mejora en epoch 23: Val Loss mejoró de 0.3875 a 0.3857 (Val Acc: 93.49%)

Entrenamiento: 14%|| | 27/200 [00:15<01:16, 2.25epoch/s, Train Loss=0.4017, Train Acc=92.16%, Val Loss=0.3832, Val Acc=93.69%, Best Val Acc=93.49%, Patience=3/25]
✓ Mejora en epoch 27: Val Loss mejoró de 0.3857 a 0.3832 (Val Acc: 93.69%)

Entrenamiento: 20% | 39/200 [00:23<01:43, 1.56epoch/s, Train Loss=0.3968, Train Acc=93.00%, Val Loss=0.3822, Val Acc=94.28%, Best Val Acc=93.69%, Patience=11/25]

✓ Mejora en epoch 39: Val Loss mejoró de 0.3832 a 0.3822 (Val Acc: 94.28%)

Entrenamiento: 24% | 47/200 [00:27<01:23, 1.84epoch/s, Train Loss=0.3962, Train Acc=92.70%, Val Loss=0.3813, Val Acc=93.69%, Best Val Acc=94.28%, Patience=7/25]

✓ Mejora en epoch 47: Val Loss mejoró de 0.3822 a 0.3813 (Val Acc: 93.69%)

Entrenamiento: 36% | 71/200 [00:41<01:15, 1.72epoch/s, Train Loss=0.3916, Train Acc=93.29%, Val Loss=0.3828, Val Acc=93.89%, Best Val Acc=93.69%, Patience=24/25]

△ Early stopping activado en epoch 72

Mejor modelo encontrado en epoch 47

Mejor Val Loss: 0.3813

Mejor Val Acc: 93.69%

Razón: No hubo mejora en Val Loss por 25 épocas consecutivas

El modelo se detuvo para prevenir sobreajuste (overfitting)

=====
ENTRENAMIENTO COMPLETADO
=====

Mejor modelo guardado de epoch 47

Mejor Val Loss: 0.3813

Mejor Val Acc: 93.69%

Total de épocas entrenadas: 72

Gráfico de pérdida y precision por épocas

Una vez entrenado el modelo generamos los siguientes graficos que describen la evolucion tanto de la pérdida, como de la precision a medida que crecen los epochs, comparandolos con el dataset de testeo/validacion.

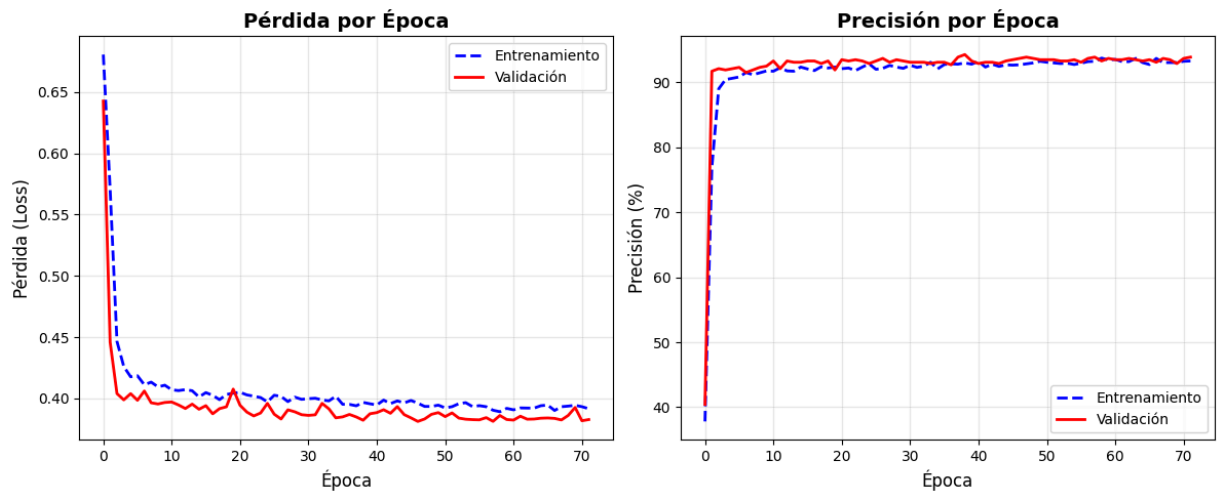
```
In [48]: plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(train_losses, 'b--', label='Entrenamiento', linewidth=2)
plt.plot(val_losses, 'r-', label='Validación', linewidth=2)
plt.xlabel('Época', fontsize=12)
plt.ylabel('Pérdida (Loss)', fontsize=12)
plt.title('Pérdida por Época', fontsize=14, fontweight='bold')
plt.legend(fontsize=10)
plt.grid(True, alpha=0.3)

plt.subplot(1, 2, 2)
plt.plot(train_accuracies, 'b--', label='Entrenamiento', linewidth=2)
plt.plot(val_accuracies, 'r-', label='Validación', linewidth=2)
plt.xlabel('Época', fontsize=12)
plt.ylabel('Precisión (%)', fontsize=12)
plt.title('Precisión por Época', fontsize=14, fontweight='bold')
plt.legend(fontsize=10)
plt.grid(True, alpha=0.3)

plt.tight_layout()
plt.show()
```

```
print(f"Mejor pérdida de validación: {min(val_losses):.4f}")
print(f"Mejor precisión de validación: {max(val_accuracies):.2f}%")
```



Mejor pérdida de validación: 0.3813
 Mejor precisión de validación: 94.28%

Evaluación final y métricas

Se muestran los resultados finales para los parametros del modelo, consecuencia de su entrenamiento.

```
In [49]: model.eval()
all_predictions = []
all_labels = []

with torch.no_grad():
    for features, labels in test_loader:
        features = features.to(device)
        outputs = model(features)
        _, predicted = torch.max(outputs, 1)
        all_predictions.extend(predicted.cpu().numpy())
        all_labels.extend(labels.numpy())

y_pred = np.array(all_predictions)
y_true = np.array(all_labels)
accuracy = accuracy_score(y_true, y_pred)
precision = precision_score(y_true, y_pred, average='weighted', zero_division=0)
recall = recall_score(y_true, y_pred, average='weighted', zero_division=0)
f1 = f1_score(y_true, y_pred, average='weighted', zero_division=0)

print("=" * 60)
print("RESULTADOS FINALES DEL MODELO")
print("=" * 60)
print(f"\nPrecisión (Accuracy): {accuracy:.4f} ({accuracy*100:.2f}%)")
print(f"Precisión (Precision - Weighted): {precision:.4f}")
print(f"Recall (Weighted): {recall:.4f}")
print(f"F1-Score (Weighted): {f1:.4f}")
print("\n" + "=" * 60)
print("\nReporte de Clasificación Detallado:")
```

```
print("=" * 60)
print(classification_report(y_true, y_pred, target_names=class_names, zero_c
```

RESULTADOS FINALES DEL MODELO

```
Precisión (Accuracy): 0.9389 (93.89%)
Precisión (Precision - Weighted): 0.9393
Recall (Weighted): 0.9389
F1-Score (Weighted): 0.9384
```

Reporte de Clasificación Detallado:

	precision	recall	f1-score	support
Dropout	0.95	0.88	0.92	190
Graduate	0.93	0.97	0.95	317
accuracy			0.94	507
macro avg	0.94	0.93	0.93	507
weighted avg	0.94	0.94	0.94	507

Matriz de Confusión

Generamos la matriz de confusión, que permite visualizar cómo se comporta el modelo al re-clasificar cada una de las clases durante el proceso de evaluación o validación, para visualizar si efectivamente las clasifica de acuerdo a su etiqueta real.

```
In [50]: cm = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_names, yticklabels=class_names,
            cbar_kws={'label': 'Cantidad'})
plt.title('Matriz de Confusión', fontsize=16, fontweight='bold', pad=20)
plt.ylabel('Etiqueta Real', fontsize=12)
plt.xlabel('Etiqueta Predicha', fontsize=12)
plt.tight_layout()
plt.show()
print("\nMatriz de Confusión:")
print("=" * 60)
print(f"{'':15}", end="")
for name in class_names:
    print(f"{name:15}", end="")
print()
for i, name in enumerate(class_names):
    print(f"{name:15}", end="")
    for j in range(len(class_names)):
        print(f"{cm[i, j]:15}", end="")
    print()
print("=" * 60)
```

```

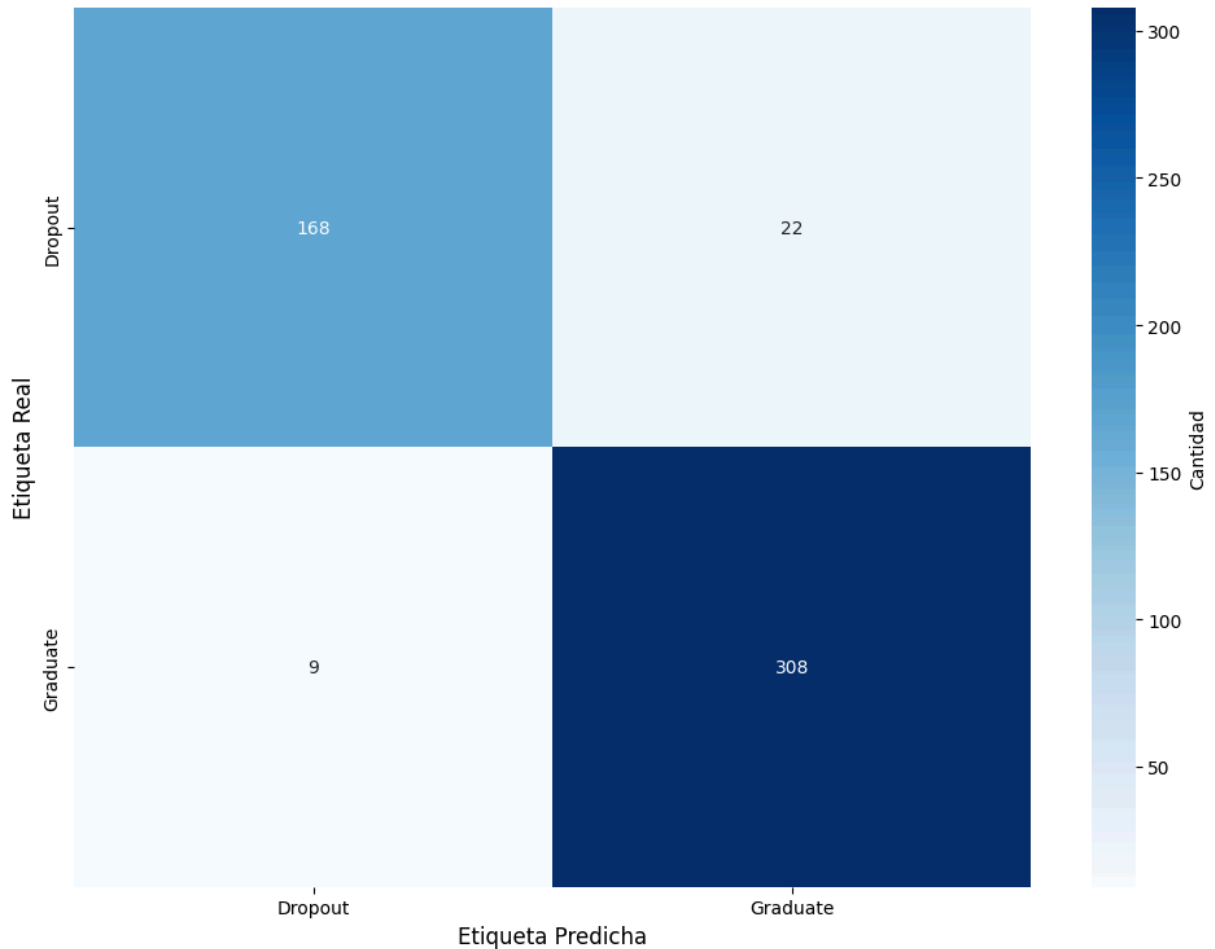
print("\nMétricas por Clase:")
print("=" * 60)
for i, class_name in enumerate(class_names):
    tp = cm[i, i]
    fp = cm[:, i].sum() - tp
    fn = cm[i, :].sum() - tp
    tn = cm.sum() - tp - fp - fn

    precision_class = tp / (tp + fp) if (tp + fp) > 0 else 0
    recall_class = tp / (tp + fn) if (tp + fn) > 0 else 0
    f1_class = 2 * (precision_class * recall_class) / (precision_class + recall_class)

    print(f"\n{class_name}:")
    print(f" Precision: {precision_class:.4f}")
    print(f" Recall: {recall_class:.4f}")
    print(f" F1-Score: {f1_class:.4f}")
    print(f" Soporte: {cm[i, :].sum()}")

```

Matriz de Confusión



Matriz de Confusión:

	Dropout	Graduate
Dropout	168	22
Graduate	9	308

Métricas por Clase:

Dropout:

Precision: 0.9492
Recall: 0.8842
F1-Score: 0.9155
Soporte: 190

Graduate:

Precision: 0.9333
Recall: 0.9716
F1-Score: 0.9521
Soporte: 317

Ejemplo de aplicación del Modelo

```
In [51]: def predict_student(student_data, model, scaler, pca, class_names, device):  
    model.eval()  
    df = pd.DataFrame([student_data])  
  
    missing_cols = set(X_train_fs.columns) - set(df.columns)  
    for col in missing_cols:  
        df[col] = 0  
  
    df = df[X_train_fs.columns]  
    df_scaled = scaler.transform(df)  
    df_pc = pca.transform(df_scaled)  
  
    with torch.no_grad():  
        df_tensor = torch.FloatTensor(df_pc).to(device)  
        probs = model(df_tensor)  
        pred_class_idx = torch.argmax(probs, dim=1).item()  
        probabilities = probs[0].cpu().numpy()  
  
    return class_names[pred_class_idx], probabilities  
  
student = {  
    'curricular_units_2nd_sem_approved': 5,  
    'curricular_units_1st_sem_approved': 6,  
    'curricular_units_2nd_sem_grade': 12.5,  
    'curricular_units_1st_sem_grade': 13.0,  
    'tuition_fees_up_to_date_1': 1,  
    'scholarship_holder_1': 0,  
    'debtor_0': 1,  
    'gender_0': 1,  
    'curricular_units_2nd_sem_enrolled': 6,  
    'curricular_units_1st_sem_enrolled': 6,  
    'tuition_fees_up_to_date_0': 0,
```

```

    'age_at_enrollment': 20,
    'application_mode': 1,
    'scholarship_holder_0': 1,
    'debtor_1': 0,
    'gender_1': 0
}

resultado, probabilidades = predict_student(student, model, scaler, pca, cla
print(f"Predicción del estudiante: {resultado}")
print("\nProbabilidades por clase:")
for i, class_name in enumerate(class_names):
    print(f" {class_name}: {probabilidades[i]:.4f} ({probabilidades[i]*100:

```

Predicción del estudiante: Graduate

Probabilidades por clase:

Dropout: 0.0001 (0.01%)

Graduate: 0.9999 (99.99%)

c:\Users\Bruno\Desktop\INF395 - IRN\proyecto_redes_neuronales\venv\Lib\site-packages\sklearn\utils\validation.py:2749: UserWarning: X does not have valid feature names, but PCA was fitted with feature names
warnings.warn(

Exportación del modelo entrenado y aplicación en caso real

```

In [52]: if torch.cuda.is_available():
          torch.save(model.state_dict(), "student_model.pt")
        else:
          torch.save(model.cpu().state_dict(), "student_model.pt")

```