Propuesta BI -Deserción en Educación Superior

Definir el contexto

La organización es una **Institución de Educación Superior (IES)**, la cual opera en el **sector de la educación universitaria y profesional**. Este tipo de institución gestiona múltiples programas académicos (carreras) y atiende a una población estudiantil diversa que proviene de distintos contextos socioeconómicos y niveles de preparación previa.

El contexto operacional se caracteriza por:

- **Altos volúmenes de matrícula:**

  La IES maneja miles de estudiantes en diferentes cohortes y modalidades.

- **Necesidad de sostenibilidad:**

  La retención de estudiantes y la tasa de graduación son críticas, ya que impactan directamente los ingresos por matrícula, la reputación institucional, la acreditación y el acceso a financiamiento externo.

- **Entorno competitivo y regulado:**

  El sector de educación superior enfrenta presión por parte de distintas universidades e institutos profesionales, además de regulaciones estatales sobre calidad, retención, deserción y empleabilidad.

En este contexto, la **deserción estudiantil** no solo representa una pérdida económica para la institución, sino también un fracaso en términos de misión social y de acompañamiento al estudiante. Contar con un sistema de **Business Intelligence (BI)** específicamente orientado a monitorear, explicar y anticipar la deserción es, por tanto, una prioridad estratégica.

Identificar la necesidad

A pesar de contar con diversos sistemas que almacenan información académica, socioeconómica y administrativa de los estudiantes, la institución enfrenta un **reto**

de información clave:

> No dispone de una visión integrada, oportuna y analítica que permita **monitorear, explicar y anticipar la deserción estudiantil**

En la práctica, esto se traduce en que:

- Los datos están **fragmentados** en distintas fuentes (gestión académica, finanzas, admisión, becas, etc.), lo que dificulta responder preguntas simples como:
  - ¿Qué carreras concentran mayor deserción?
  - ¿Cual es la tasa de deserción por el estado civil?
  - ¿Qué perfiles de estudiantes presentan mayor riesgo?
- La toma de decisiones se apoya en **reportes manuales y estáticos**, con poco detalle histórico y escasa capacidad de segmentación (por cohorte, modalidad, perfil socioeconómico, etc.).
- No existe un **sistema de alertas tempranas** que permita identificar estudiantes en riesgo antes de que abandonen, limitando las posibilidades de intervención oportuna (tutorías, apoyos económicos, acompañamiento académico).

Frente a este escenario, surge la necesidad de implementar una solución de **Business Intelligence (BI)** que permita:

- **Integrar** en un único modelo de datos la información relevante del estudiante (perfil, desempeño, situación económica, contexto externo).
- **Visualizar y analizar** de forma dinámica los indicadores de deserción, retención y rendimiento académico, a distintos niveles (institución, carrera, cohorte, grupo vulnerable, etc.).
- **Soportar decisiones estratégicas y tácticas**, desde la planificación institucional hasta la gestión diaria de directores de carrera, unidades de apoyo y bienestar estudiantil.

Fuentes de datos

Datos necesarios:

- **Datos académicos**:
  - Calificaciones por semestre, unidades curriculares aprobadas, y evaluaciones previas.
  - Modalidad de asistencia (diurna/nocturna).
  - Historial académico del estudiante.
- **Datos demográficos**:
  - Edad, género, estado civil, nacionalidad, nivel educativo de los padres, y tipo de carrera.
- **Factores socioeconómicos externos**:
  - **Tasa de desempleo**: Relacionada con la situación económica general que podría influir en la estabilidad de los estudiantes.
  - **Tasa de inflación y PIB**: Datos económicos relevantes que podrían afectar la capacidad de los estudiantes para continuar con sus estudios.

Fuentes de datos:

1. **Bases de datos internas** de la universidad o institución educativa, como sistemas de gestión académica (ERP) y sistemas de información estudiantil (CRM).
2. **Sistemas de gestión de estudiantes** (LMS o plataformas académicas).
3. **APIs externas** o **datos abiertos** proporcionados por entidades gubernamentales o internacionales para obtener información económica como la tasa de desempleo, la inflación y el PIB .
4. **Archivos históricos** de rendimiento académico almacenados en formatos CSV o Excel que contengan la información histórica de los estudiantes.

Proceso ETL Completo

2.Proceso ETL – Extracción (Extract)

En este proyecto, la fase de **Extracción** consiste en obtener y dejar listo para su uso el dataset de deserción estudiantil almacenado en un **archivo CSV local**,

utilizando **Python y pandas** como herramientas principales.

El objetivo es:

- Leer el dataset de forma **segura y repetible**.
- Asegurar que la estructura (columnas, tipos básicos) sea la esperada.
- Detectar de forma temprana **problemas de integridad** (archivos corruptos, columnas faltantes, filas vacías, etc.).

Herramientas de extracción

La extracción se realiza exclusivamente con:

- **Python 3.x** como lenguaje de scripting.
- **pandas** para leer y manejar el dataset tabular (read_csv, DataFrame).
- **pathlib / os** para manejar rutas y existencia de archivos.

Validar la estructura básica

- Comprobar:
  - Número de columnas.
  - Nombres de columnas clave (por ejemplo: Marital status, Course, Age at enrollment, Admission grade, Curricular units 1st sem (approved), Unemployment rate, GDP, Target, etc.).
  - Cantidad de registros.

2.Proceso ETL – Transformación (Transform)

Este bloque limpia y transforma la columna Marital status: primero trata los códigos como numéricos, corrige outliers y valores nulos usando un rango válido [1, 6], y luego los convierte en categorías textuales interpretables para el análisis.

```
# Definir la columna a trabajar
columna_a_trabajar = "Marital status"
```

```
# Asegurar que la columna sea numérica (convierte valores no válidos a NaN)
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
ors="coerce")

# Boxplot para visualizar la distribución y posibles outliers
dataset[columna_a_trabajar].plot(kind="box")

# Definir rango válido para la variable (códigos 1 a 6)
UMBRAL_MINIMO = 1
UMBRAL_MAXIMO = 6

# Calcular la media de la columna (se usará para imputar outliers y NaN)
media_columna = dataset[columna_a_trabajar].mean()

# Reemplazar valores menores al mínimo por la media
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trab
ajar] = media_columna

# Reemplazar valores mayores al máximo por la media
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trab
ajar] = media_columna

# Imputar valores NaN restantes con la media
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_colu
mna)

# Transformar a entero y luego a string para aplicar el mapeo categórico
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].astype(int).astype
(str)

# Definir diccionario de mapeo de código numérico a categoría textual
diccionario_mapeo_marital_status = {
    "1": "single",          # 1 – single
    "2": "married",         # 2 – married
    "3": "widower",         # 3 – widower
```

```
    "4": "divorced",          # 4 – divorced
    "5": "facto union",       # 5 – facto union
    "6": "legally separated"   # 6 – legally separated
}

# Aplicar el mapeo categórico a la columna
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].map(diccionario_
mapeo_marital_status)
```

Esta transformación limpia primero la columna Application mode como código numérico (corrige valores fuera del rango válido y nulos) y luego la convierte a categorías textuales interpretables según el modo de ingreso del estudiante.

```
# Definir la columna a trabajar
columna_a_trabajar = "Application mode"

# Asegurar que la columna sea numérica (valores no válidos se convierten a
NaN)
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
ors="coerce")

# Boxplot para visualizar la distribución y posibles outliers
dataset[columna_a_trabajar].plot(kind="box")

# Definir un rango numérico amplio de códigos válidos observados (1 a 57)
UMBRAL_MINIMO = 1
UMBRAL_MAXIMO = 57

# Calcular la media de la columna (se usará para imputar outliers y NaN)
media_columna = dataset[columna_a_trabajar].mean()

# Reemplazar valores menores al mínimo por la media
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trab
ajar] = media_columna

# Reemplazar valores mayores al máximo por la media
```

```
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trabajar] = media_columna

# Imputar valores NaN restantes con la media
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_columna)

# Transformar a entero y luego a string para aplicar el mapeo categórico
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].astype(int).astype(str)

# Definir diccionario de mapeo de código numérico a categoría textual
diccionario_mapeo_application_mode = {
    "1":  "1st phase - general contingent",
    "2":  "Ordinance No. 612/93",
    "5":  "1st phase - special contingent (Azores Island)",
    "7":  "Holders of other higher courses",
    "10": "Ordinance No. 854-B/99",
    "15": "International student (bachelor)",
    "16": "1st phase - special contingent (Madeira Island)",
    "17": "2nd phase - general contingent",
    "18": "3rd phase - general contingent",
    "26": "Ordinance No. 533-A/99, item b2) (Different Plan)",
    "27": "Ordinance No. 533-A/99, item b3 (Other Institution)",
    "39": "Over 23 years old",
    "42": "Transfer",
    "43": "Change of course",
    "44": "Technological specialization diploma holders",
    "51": "Change of institution/course",
    "53": "Short cycle diploma holders",
    "57": "Change of institution/course (International)"
}

# Aplicar el mapeo categórico a la columna
```

```
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].map(diccionario_
mapeo_application_mode)
```

Esta transformación verifica la calidad de la columna Application, identifica y corrige outliers numéricos usando el rango válido [0, 9], e imputa valores nulos con la media para dejar la variable lista para análisis.

```
# Definir la columna a trabajar
columna_a_trabajar = "Application"

# Asegurar que la columna sea numérica (valores no válidos se convierten a
NaN)
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
ors="coerce")

# Boxplot para visualizar la distribución y posibles outliers
dataset[columna_a_trabajar].plot(kind="box")

# Definir rango válido para la variable (de 0 = primera opción a 9 = última)
UMBRAL_MINIMO = 0
UMBRAL_MAXIMO = 9

# Calcular la media de la columna (se usará para imputar outliers y NaN)
media_columna = dataset[columna_a_trabajar].mean()

# Reemplazar valores menores al mínimo por la media
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trab
ajar] = media_columna

# Reemplazar valores mayores al máximo por la media
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trab
ajar] = media_columna

# Imputar valores NaN restantes con la media
```

```
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_colu
mna)
```

Esta transformación depura la columna Course como código numérico (corrige valores fuera del rango observado y nulos) y luego convierte esos códigos en descripciones textuales de la carrera, haciéndola interpretable para el análisis.

```
# Definir la columna a trabajar
columna_a_trabajar = "Course"

# Asegurar que la columna sea numérica (valores no válidos se convierten a
NaN)
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
ors="coerce")

# Boxplot para visualizar la distribución y posibles outliers
dataset[columna_a_trabajar].plot(kind="box")

# Definir un rango numérico amplio de códigos válidos observados
UMBRAL_MINIMO = 33
UMBRAL_MAXIMO = 9991

# Calcular la media de la columna (se usará para imputar outliers y NaN)
media_columna = dataset[columna_a_trabajar].mean()

# Reemplazar valores menores al mínimo por la media
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trab
ajar] = media_columna

# Reemplazar valores mayores al máximo por la media
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trab
ajar] = media_columna

# Imputar valores NaN restantes con la media
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_colu
mna)
```

```
# Transformar a entero y luego a string para aplicar el mapeo categórico
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].astype(int).astype
(str)

# Definir diccionario de mapeo de código numérico a descripción textual de la
carrera
diccionario_mapeo_course = {
    "33":   "Biofuel Production Technologies",
    "171":  "Animation and Multimedia Design",
    "8014": "Social Service (evening attendance)",
    "9003": "Agronomy",
    "9070": "Communication Design",
    "9085": "Veterinary Nursing",
    "9119": "Informatics Engineering",
    "9130": "Equinculture",
    "9147": "Management",
    "9238": "Social Service",
    "9254": "Tourism",
    "9500": "Nursing",
    "9556": "Oral Hygiene",
    "9670": "Advertising and Marketing Management",
    "9773": "Journalism and Communication",
    "9853": "Basic Education",
    "9991": "Management (evening attendance)"
}

# Aplicar el mapeo categórico a la columna
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].map(diccionario_
mapeo_course)
```

Esta transformación verifica y limpia la columna Daytime/evening attendance como variable binaria en el rango [0, 1], y luego convierte los códigos en categorías textuales (daytime, evening) para facilitar el análisis.

```
# Definir la columna a trabajar
columna_a_trabajar = "Daytime/evening attendance"

# Asegurar que la columna sea numérica (valores no válidos se convierten a NaN)
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], errors="coerce")

# Boxplot para visualizar la distribución y posibles outliers
dataset[columna_a_trabajar].plot(kind="box")

# Definir rango válido para la variable (0 = evening, 1 = daytime)
UMBRAL_MINIMO = 0
UMBRAL_MAXIMO = 1

# Calcular la media de la columna (se usará para imputar outliers y NaN)
media_columna = dataset[columna_a_trabajar].mean()

# Reemplazar valores menores al mínimo por la media
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trabajar] = media_columna

# Reemplazar valores mayores al máximo por la media
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trabajar] = media_columna

# Imputar valores NaN restantes con la media
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_columna)

# Transformar a entero y luego a string para aplicar el mapeo categórico
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].astype(int).astype(str)

# Definir diccionario de mapeo de código numérico a categoría textual
```

```
diccionario_mapeo_daytime_evening = {
    "1": "daytime",  # 1 – daytime
    "0": "evening"   # 0 – evening
}

# Aplicar el mapeo categórico a la columna
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].map(diccionario_
mapeo_daytime_evening)
```

Esta transformación limpia la columna Previous qualification usando un rango numérico válido y luego convierte los códigos en descripciones textuales del nivel educativo previo, haciéndola interpretable como variable categórica.

```
# Definir la columna a trabajar
columna_a_trabajar = "Previous qualification"

# Asegurar que la columna sea numérica (valores no válidos se convierten a
NaN)
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
ors="coerce")

# Boxplot para visualizar la distribución y posibles outliers
dataset[columna_a_trabajar].plot(kind="box")

# Definir rango válido aproximado según los códigos observados (1 a 43)
UMBRAL_MINIMO = 1
UMBRAL_MAXIMO = 43

# Calcular la media de la columna (se usará para imputar outliers y NaN)
media_columna = dataset[columna_a_trabajar].mean()

# Reemplazar valores menores al mínimo por la media
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trab
ajar] = media_columna
```

```
# Reemplazar valores mayores al máximo por la media
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trab
ajar] = media_columna

# Imputar valores NaN restantes con la media
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_colu
mna)

# Transformar a entero y luego a string para aplicar el mapeo categórico
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].astype(int).astype
(str)

# Definir diccionario de mapeo de código numérico a nivel educativo previo
diccionario_mapeo_previous_qualification = {
    "1":  "Secondary education",
    "2":  "Higher education - bachelor's degree",
    "3":  "Higher education - degree",
    "4":  "Higher education - master's",
    "5":  "Higher education - doctorate",
    "6":  "Frequency of higher education",
    "9":  "12th year of schooling - not completed",
    "10": "11th year of schooling - not completed",
    "12": "Other - 11th year of schooling",
    "14": "10th year of schooling",
    "15": "10th year of schooling - not completed",
    "19": "Basic education 3rd cycle (9th/10th/11th year) or equiv.",
    "38": "Basic education 2nd cycle (6th/7th/8th year) or equiv.",
    "39": "Technological specialization course",
    "40": "Higher education - degree (1st cycle)",
    "42": "Professional higher technical course",
    "43": "Higher education - master (2nd cycle)"
}

# Aplicar el mapeo categórico a la columna
```

```
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].map(diccionario_
mapeo_previous_qualification)
```

Esta transformación asegura la calidad de la variable continua Previous qualification (grade): se identifican outliers con un boxplot, se corrigen valores fuera del rango [0, 200] sustituyéndolos por la media y se imputan valores nulos con la misma media para estabilizar la distribución.

```
# Definir la columna a trabajar
columna_a_trabajar = "Previous qualification (grade)"

# Asegurar que la columna sea numérica (valores no válidos se convierten a
NaN)
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
ors="coerce")

# Boxplot para visualizar la distribución y posibles outliers
dataset[columna_a_trabajar].plot(kind="box")

# Definir rango válido para la nota previa (entre 0 y 200)
UMBRAL_MINIMO = 0
UMBRAL_MAXIMO = 200

# Calcular la media de la columna (se usará para imputar outliers y NaN)
media_columna = dataset[columna_a_trabajar].mean()

# Reemplazar valores menores al mínimo por la media
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trab
ajar] = media_columna

# Reemplazar valores mayores al máximo por la media
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trab
ajar] = media_columna

# Imputar valores NaN restantes con la media
```

```
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_colu
mna)
```

Esta transformación limpia la columna Nacionality dentro del rango válido de códigos y luego los convierte en descripciones textuales del país, dejándola como variable categórica interpretable.

```
# Definir la columna a trabajar
columna_a_trabajar = "Nacionality"

# Asegurar que la columna sea numérica (valores no válidos se convierten a
NaN)
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
ors="coerce")

# Boxplot para visualizar la distribución y posibles outliers
dataset[columna_a_trabajar].plot(kind="box")

# Definir rango válido aproximado según los códigos observados (1 a 109)
UMBRAL_MINIMO = 1
UMBRAL_MAXIMO = 109

# Calcular la media de la columna (se usará para imputar outliers y NaN)
media_columna = dataset[columna_a_trabajar].mean()

# Reemplazar valores menores al mínimo por la media
dataset.loc[dataset[columna_a_trbajar] < UMBRAL_MINIMO, columna_a_trabaj
ar] = media_columna

# Reemplazar valores mayores al máximo por la media
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trab
ajar] = media_columna

# Imputar valores NaN restantes con la media
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_colu
mna)
```

```
# Transformar a entero y luego a string para aplicar el mapeo categórico
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].astype(int).astype
(str)

# Definir diccionario de mapeo de código numérico a nacionalidad textual
diccionario_mapeo_nacionality = {
   "1":   "Portuguese",
   "2":   "German",
   "6":   "Spanish",
   "11":  "Italian",
   "13":  "Dutch",
   "14":  "English",
   "17":  "Lithuanian",
   "21":  "Angolan",
   "22":  "Cape Verdean",
   "24":  "Guinean",
   "25":  "Mozambican",
   "26":  "Santomean",
   "32":  "Turkish",
   "41":  "Brazilian",
   "62":  "Romanian",
   "100": "Moldovan",
   "101": "Mexican",
   "103": "Ukrainian",
   "105": "Russian",
   "108": "Cuban",
   "109": "Colombian"
}

# Aplicar el mapeo categórico a la columna
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].map(diccionario_
mapeo_nacionality)
```

Esta transformación estandariza el código educativo de la madre: primero limpia la variable numérica Mother's qualification usando un rango válido [1, 44] con

imputación por media, y luego convierte los códigos en descripciones textuales del nivel educativo para usarla como categoría.

```
# Definir la columna a trabajar
columna_a_trabajar = "Mother's qualification"

# Asegurar que la columna sea numérica (valores no válidos se convierten a NaN)
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], errors="coerce")

# Boxplot para visualizar la distribución y posibles outliers
dataset[columna_a_trabajar].plot(kind="box")

# Definir rango válido aproximado según los códigos observados (1 a 44)
UMBRAL_MINIMO = 1
UMBRAL_MAXIMO = 44

# Calcular la media de la columna (se usará para imputar outliers y NaN)
media_columna = dataset[columna_a_trabajar].mean()

# Reemplazar valores menores al mínimo por la media
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trabajar] = media_columna

# Reemplazar valores mayores al máximo por la media
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trabajar] = media_columna

# Imputar valores NaN restantes con la media
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_columna)

# Transformar a entero y luego a string para aplicar el mapeo categórico
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].astype(int).astype(str)
```

```
# Definir diccionario de mapeo: código numérico → descripción del nivel educ
ativo de la madre
diccionario_mapeo_mother_qualification = {
    "1":  "Secondary Education - 12th Year of Schooling or Eq.",
    "2":  "Higher Education - Bachelor's Degree",
    "3":  "Higher Education - Degree",
    "4":  "Higher Education - Master's",
    "5":  "Higher Education - Doctorate",
    "6":  "Frequency of Higher Education",
    "9":  "12th Year of Schooling - Not Completed",
    "10": "11th Year of Schooling - Not Completed",
    "11": "7th Year (Old)",
    "12": "Other - 11th Year of Schooling",
    "14": "10th Year of Schooling",
    "18": "General commerce course",
    "19": "Basic Education 3rd Cycle (9th/10th/11th Year) or Equiv.",
    "22": "Technical-professional course",
    "26": "7th year of schooling",
    "27": "2nd cycle of the general high school course",
    "29": "9th Year of Schooling - Not Completed",
    "30": "8th year of schooling",
    "34": "Unknown",
    "35": "Can't read or write",
    "36": "Can read without having a 4th year of schooling",
    "37": "Basic education 1st cycle (4th/5th year) or equiv.",
    "38": "Basic Education 2nd Cycle (6th/7th/8th Year) or Equiv.",
    "39": "Technological specialization course",
    "40": "Higher education - degree (1st cycle)",
    "41": "Specialized higher studies course",
    "42": "Professional higher technical course",
    "43": "Higher Education - Master (2nd cycle)",
    "44": "Higher Education - Doctorate (3rd cycle)"
}

# Aplicar el mapeo categórico a la columna
```

```
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].map(diccionario_
mapeo_mother_qualification)
```

Esta transformación hace lo mismo para Father's qualification: limpia la variable numérica en el rango [1, 44] e imputa por media, y después convierte los códigos en descripciones textuales del nivel educativo del padre para facilitar la interpretación.

```
# Definir la columna a trabajar
columna_a_trabajar = "Father's qualification"

# Asegurar que la columna sea numérica (valores no válidos se convierten a
NaN)
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
ors="coerce")

# Boxplot para visualizar la distribución y posibles outliers
dataset[columna_a_trabajar].plot(kind="box")

# Definir rango válido aproximado según los códigos observados (1 a 44)
UMBRAL_MINIMO = 1
UMBRAL_MAXIMO = 44

# Calcular la media de la columna (se usará para imputar outliers y NaN)
media_columna = dataset[columna_a_trabajar].mean()

# Reemplazar valores menores al mínimo por la media
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trab
ajar] = media_columna

# Reemplazar valores mayores al máximo por la media
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trab
ajar] = media_columna

# Imputar valores NaN restantes con la media
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_colu
```

```
mna)

# Transformar a entero y luego a string para aplicar el mapeo categórico
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].astype(int).astype
(str)

# Definir diccionario de mapeo: código numérico → descripción del nivel educ
ativo del padre
diccionario_mapeo_father_qualification = {
    "1":  "Secondary Education - 12th Year of Schooling or Eq.",
    "2":  "Higher Education - Bachelor's Degree",
    "3":  "Higher Education - Degree",
    "4":  "Higher Education - Master's",
    "5":  "Higher Education - Doctorate",
    "6":  "Frequency of Higher Education",
    "9":  "12th Year of Schooling - Not Completed",
    "10": "11th Year of Schooling - Not Completed",
    "11": "7th Year (Old)",
    "12": "Other - 11th Year of Schooling",
    "13": "2nd year complementary high school course",
    "14": "10th Year of Schooling",
    "18": "General commerce course",
    "19": "Basic Education 3rd Cycle (9th/10th/11th Year) or Equiv.",
    "20": "Complementary High School Course",
    "22": "Technical-professional course",
    "25": "Complementary High School Course - not concluded",
    "26": "7th year of schooling",
    "27": "2nd cycle of the general high school course",
    "29": "9th Year of Schooling - Not Completed",
    "30": "8th year of schooling",
    "31": "General Course of Administration and Commerce",
    "33": "Supplementary Accounting and Administration",
    "34": "Unknown",
    "35": "Can't read or write",
    "36": "Can read without having a 4th year of schooling",
    "37": "Basic education 1st cycle (4th/5th year) or equiv.",
```

```
    "38": "Basic Education 2nd Cycle (6th/7th/8th Year) or Equiv.",
    "39": "Technological specialization course",
    "40": "Higher education - degree (1st cycle)",
    "41": "Specialized higher studies course",
    "42": "Professional higher technical course",
    "43": "Higher Education - Master (2nd cycle)",
    "44": "Higher Education - Doctorate (3rd cycle)"
}

# Aplicar el mapeo categórico a la columna
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].map(diccionario_
mapeo_father_qualification)
```

Esta transformación limpia y estandariza la columna Mother's occupation: se validan códigos numéricos en un rango amplio y se imputan nulos con la media; después se mapea cada código a una descripción textual de la ocupación de la madre para usarla como variable categórica.

```
# Definir la columna a trabajar
columna_a_trabajar = "Mother's occupation"

# Asegurar que la columna sea numérica (valores no válidos se convierten a
NaN)
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
ors="coerce")

# Boxplot para visualizar la distribución y posibles outliers
dataset[columna_a_trabajar].plot(kind="box")

# Definir rango válido aproximado según los códigos observados (0 a 194)
UMBRAL_MINIMO = 0
UMBRAL_MAXIMO = 194

# Calcular la media de la columna (se usará para imputar outliers y NaN)
media_columna = dataset[columna_a_trabajar].mean()
```

```
# Reemplazar valores menores al mínimo por la media
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trab
ajar] = media_columna

# Reemplazar valores mayores al máximo por la media
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trab
ajar] = media_columna

# Imputar valores NaN restantes con la media
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_colu
mna)

# Transformar a entero y luego a string para aplicar el mapeo categórico
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].astype(int).astype
(str)

# Diccionario de mapeo código → descripción ocupación de la madre
diccionario_mapeo_mother_occupation = {
    "0":   "Student",
    "1":   "Representatives of the Legislative Power and Executive Bodies, Direc
tors, Directors and Executive Managers",
    "2":   "Specialists in Intellectual and Scientific Activities",
    "3":   "Intermediate Level Technicians and Professions",
    "4":   "Administrative staff",
    "5":   "Personal Services, Security and Safety Workers and Sellers",
    "6":   "Farmers and Skilled Workers in Agriculture, Fisheries and Forestry",
    "7":   "Skilled Workers in Industry, Construction and Craftsmen",
    "8":   "Installation and Machine Operators and Assembly Workers",
    "9":   "Unskilled Workers",
    "10":  "Armed Forces Professions",
    "90":  "Other Situation",
    "99":  "(blank)",
    "122": "Health professionals",
    "123": "Teachers",
    "125": "Specialists in information and communication technologies (ICT)",
    "131": "Intermediate level science and engineering technicians and professi
```

```
ons",
    "132": "Technicians and professionals, of intermediate level of health",
    "134": "Intermediate level technicians from legal, social, sports, cultural and similar services",
    "141": "Office workers, secretaries in general and data processing operators",
    "143": "Data, accounting, statistical, financial services and registry-related operators",
    "144": "Other administrative support staff",
    "151": "Personal service workers",
    "152": "Sellers",
    "153": "Personal care workers and the like",
    "171": "Skilled construction workers and the like, except electricians",
    "173": "Skilled workers in printing, precision instrument manufacturing, jewellers, artisans and the like",
    "175": "Workers in food processing, woodworking, clothing and other industries and crafts",
    "191": "Cleaning workers",
    "192": "Unskilled workers in agriculture, animal production, fisheries and forestry",
    "193": "Unskilled workers in extractive industry, construction, manufacturing and transport",
    "194": "Meal preparation assistants"
}

# Aplicar el mapeo categórico a la columna
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].map(diccionario_mapeo_mother_occupation)
```

Esta transformación hace lo mismo para Father's occupation: se limpia el código numérico en un rango razonable y se imputan nulos con la media; luego se convierten los códigos en descripciones textuales de la ocupación del padre para análisis categórico.

```
# Definir la columna a trabajar
columna_a_trabajar = "Father's occupation"
```

```
# Asegurar que la columna sea numérica (valores no válidos se convierten a NaN)
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], errors="coerce")

# Boxplot para visualizar la distribución y posibles outliers
dataset[columna_a_trabajar].plot(kind="box")

# Definir rango válido aproximado según los códigos observados (0 a 195)
UMBRAL_MINIMO = 0
UMBRAL_MAXIMO = 195

# Calcular la media de la columna (se usará para imputar outliers y NaN)
media_columna = dataset[columna_a_trabajar].mean()

# Reemplazar valores menores al mínimo por la media
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trabajar] = media_columna

# Reemplazar valores mayores al máximo por la media
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trabajar] = media_columna

# Imputar valores NaN restantes con la media
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_columna)

# Transformar a entero y luego a string para aplicar el mapeo categórico
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].astype(int).astype(str)

# Diccionario de mapeo código → descripción ocupación del padre
diccionario_mapeo_father_occupation = {
    "0":   "Student",
    "1":   "Representatives of the Legislative Power and Executive Bodies, Direc
```

tors, Directors and Executive Managers",
"2": "Specialists in Intellectual and Scientific Activities",
"3": "Intermediate Level Technicians and Professions",
"4": "Administrative staff",
"5": "Personal Services, Security and Safety Workers and Sellers",
"6": "Farmers and Skilled Workers in Agriculture, Fisheries and Forestry",
"7": "Skilled Workers in Industry, Construction and Craftsmen",
"8": "Installation and Machine Operators and Assembly Workers",
"9": "Unskilled Workers",
"10": "Armed Forces Professions",
"90": "Other Situation",
"99": "(blank)",
"101": "Armed Forces Officers",
"102": "Armed Forces Sergeants",
"103": "Other Armed Forces personnel",
"112": "Directors of administrative and commercial services",
"114": "Hotel, catering, trade and other services directors",
"121": "Specialists in the physical sciences, mathematics, engineering and related techniques",
"122": "Health professionals",
"123": "Teachers",
"124": "Specialists in finance, accounting, administrative organization, public and commercial relations",
"131": "Intermediate level science and engineering technicians and professions",
"132": "Technicians and professionals, of intermediate level of health",
"134": "Intermediate level technicians from legal, social, sports, cultural and similar services",
"135": "Information and communication technology technicians",
"141": "Office workers, secretaries in general and data processing operators",
"143": "Data, accounting, statistical, financial services and registry-related operators",
"144": "Other administrative support staff",
"151": "Personal service workers",
"152": "Sellers",

```
    "153": "Personal care workers and the like",
    "154": "Protection and security services personnel",
    "161": "Market-oriented farmers and skilled agricultural and animal producti
on workers",
    "163": "Farmers, livestock keepers, fishermen, hunters and gatherers, subsi
stence",
    "171": "Skilled construction workers and the like, except electricians",
    "172": "Skilled workers in metallurgy, metalworking and similar",
    "174": "Skilled workers in electricity and electronics",
    "175": "Workers in food processing, woodworking, clothing and other indust
ries and crafts",
    "181": "Fixed plant and machine operators",
    "182": "Assembly workers",
    "183": "Vehicle drivers and mobile equipment operators",
    "192": "Unskilled workers in agriculture, animal production, fisheries and for
estry",
    "193": "Unskilled workers in extractive industry, construction, manufacturin
g and transport",
    "194": "Meal preparation assistants",
    "195": "Street vendors (except food) and street service providers"
}

# Aplicar el mapeo categórico a la columna
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].map(diccionario_
mapeo_father_occupation)
```

Esta transformación controla la calidad de la variable continua Admission grade, eliminando outliers fuera del rango [0, 200] mediante sustitución por la media e imputando valores nulos con la misma media para estabilizar la distribución.

```
# Definir la columna a trabajar
columna_a_trabajar = "Admission grade"

# Asegurar que la columna sea numérica (valores no válidos se convierten a
NaN)
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
```

```
ors="coerce")

# Boxplot para visualizar la distribución y posibles outliers
dataset[columna_a_trabajar].plot(kind="box")

# Definir rango válido para la nota de admisión (entre 0 y 200)
UMBRAL_MINIMO = 0
UMBRAL_MAXIMO = 200

# Calcular la media de la columna (se usará para imputar outliers y NaN)
media_columna = dataset[columna_a_trabajar].mean()

# Reemplazar valores menores al mínimo por la media
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trab
ajar] = media_columna

# Reemplazar valores mayores al máximo por la media
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trab
ajar] = media_columna

# Imputar valores NaN restantes con la media
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_colu
mna)
```

Las siguientes transformaciones tratan variables binarias (0/1) como indicadores: se valida el rango [0, 1], se corrigen valores anómalos imputando con la media y luego se mapean los códigos a etiquetas textuales para mejorar la interpretabilidad.

```
# Displaced: 1 – yes, 0 – no
columna_a_trabajar = "Displaced"
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
ors="coerce")
dataset[columna_a_trabajar].plot(kind="box")
```

```
UMBRAL_MINIMO = 0
UMBRAL_MAXIMO = 1
media_columna = dataset[columna_a_trabajar].mean()
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trab
ajar] = media_columna
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trab
ajar] = media_columna
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_colu
mna)
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].astype(int).astype
(str)
diccionario_mapeo_displaced = {"1": "yes", "0": "no"}
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].map(diccionario_
mapeo_displaced)

# Educational special needs: 1 – yes, 0 – no
columna_a_trabajar = "Educational special needs"
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
ors="coerce")
dataset[columna_a_trabajar].plot(kind="box")
UMBRAL_MINIMO = 0
UMBRAL_MAXIMO = 1
media_columna = dataset[columna_a_trabajar].mean()
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trab
ajar] = media_columna
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trab
ajar] = media_columna
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_colu
mna)
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].astype(int).astype
(str)
diccionario_mapeo_esn = {"1": "yes", "0": "no"}
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].map(diccionario_
mapeo_esn)

# Debtor: 1 – yes, 0 – no
```

```
columna_a_trabajar = "Debtor"
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
ors="coerce")
dataset[columna_a_trabajar].plot(kind="box")
UMBRAL_MINIMO = 0
UMBRAL_MAXIMO = 1
media_columna = dataset[columna_a_trabajar].mean()
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trab
ajar] = media_columna
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trab
ajar] = media_columna
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_colu
mna)
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].astype(int).astype
(str)
diccionario_mapeo_debtor = {"1": "yes", "0": "no"}
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].map(diccionario_
mapeo_debtor)

# Tuition fees up to date: 1 – yes, 0 – no
columna_a_trabajar = "Tuition fees up to date"
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
ors="coerce")
dataset[columna_a_trabajar].plot(kind="box")
UMBRAL_MINIMO = 0
UMBRAL_MAXIMO = 1
media_columna = dataset[columna_a_trabajar].mean()
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trab
ajar] = media_columna
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trab
ajar] = media_columna
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_colu
mna)
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].astype(int).astype
(str)
diccionario_mapeo_tuition = {"1": "yes", "0": "no"}
```

```
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].map(diccionario_
mapeo_tuition)

# Scholarship holder: 1 – yes, 0 – no
columna_a_trabajar = "Scholarship holder"
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
ors="coerce")
dataset[columna_a_trabajar].plot(kind="box")
UMBRAL_MINIMO = 0
UMBRAL_MAXIMO = 1
media_columna = dataset[columna_a_trabajar].mean()
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trab
ajar] = media_columna
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trab
ajar] = media_columna
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_colu
mna)
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].astype(int).astype
(str)
diccionario_mapeo_scholar = {"1": "yes", "0": "no"}
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].map(diccionario_
mapeo_scholar)
```

Esta transformación trata la variable de género como binaria: se valida el rango [0, 1], se corrigen valores anómalos e imputan nulos con la media y luego se mapea a las etiquetas textuales male y female.

```
# Gender: 1 – male, 0 – female
columna_a_trabajar = "Gender"

# Asegurar que la columna sea numérica
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
ors="coerce")

# Boxplot para visualizar la distribución y posibles outliers
dataset[columna_a_trabajar].plot(kind="box")
```

```
# Rango válido
UMBRAL_MINIMO = 0
UMBRAL_MAXIMO = 1

# Media para imputar
media_columna = dataset[columna_a_trabajar].mean()

# Reemplazar valores fuera del rango
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trab
ajar] = media_columna
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trab
ajar] = media_columna

# Imputar NaN con la media
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_colu
mna)

# Convertir a entero y luego a string
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].astype(int).astype
(str)

# Mapeo numérico → textual
diccionario_mapeo_gender = {"1": "male", "0": "female"}
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].map(diccionario_
mapeo_gender)
```

Esta transformación para Age at enrollment garantiza que la variable de edad sea numérica, permite visualizar outliers con un boxplot y, sin definir aún un rango específico, se encarga al menos de imputar valores nulos con la media para no perder registros.

```
# Definir la columna a trabajar
columna_a_trabajar = "Age at enrollment"

# Asegurar que la columna sea numérica
```

```
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
ors="coerce")

# Boxplot para visualizar la distribución y posibles outliers
dataset[columna_a_trabajar].plot(kind="box")

# Calcular la media de la columna
media_columna = dataset[columna_a_trabajar].mean()

# Imputar valores NaN con la media
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_colu
mna)
```

Para estas variables se siguen dos estrategias: para International se trata como indicador binario 0/1 y se mapea a texto; para las variables de unidades curriculares del 1.er y 2.º semestre se limpian como enteras, controlando outliers con un rango razonable y rellenando valores faltantes con la media.

```
# ----------------------------------------------------------
# International: 1 – yes, 0 – no (variable binaria con mapeo a texto)
# ----------------------------------------------------------

columna_a_trabajar = "International"

# Asegurar que la columna sea numérica
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
ors="coerce")

# Boxplot para visualizar la distribución y posibles outliers
dataset[columna_a_trabajar].plot(kind="box")

# Definir rango válido [0, 1]
UMBRAL_MINIMO = 0
UMBRAL_MAXIMO = 1

# Calcular la media de la columna
```

```
media_columna = dataset[columna_a_trabajar].mean()

# Reemplazar valores menores al mínimo por la media
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trab
ajar] = media_columna

# Reemplazar valores mayores al máximo por la media
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trab
ajar] = media_columna

# Imputar valores NaN con la media
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_colu
mna)

# Convertir a entero y luego a string para mapear
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].astype(int).astype
(str)

# Diccionario de mapeo 0/1 a texto
diccionario_mapeo_international = {
    "1": "yes",
    "0": "no"
}

# Aplicar el mapeo categórico
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].map(diccionario_
mapeo_international)



# ----------------------------------------------------------
# Curricular units 1st sem (credited): variable de conteo con rango razonable
# ----------------------------------------------------------

columna_a_trabajar = "Curricular units 1st sem (credited)"

# Asegurar que la columna sea numérica
```

```
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
ors="coerce")

# Boxplot para visualizar la distribución y posibles outliers
dataset[columna_a_trabajar].plot(kind="box")

# Definir rango válido (entre 0 y 20 unidades como rango razonable)
UMBRAL_MINIMO = 0
UMBRAL_MAXIMO = 20

# Calcular la media de la columna
media_columna = dataset[columna_a_trabajar].mean()

# Reemplazar valores menores al mínimo por la media
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trab
ajar] = media_columna

# Reemplazar valores mayores al máximo por la media
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trab
ajar] = media_columna

# Imputar valores NaN con la media
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_colu
mna)



# ---------------------------------------------------------
# Curricular units 1st sem (enrolled): variable de conteo con rango razonable
# ---------------------------------------------------------

columna_a_trabajar = "Curricular units 1st sem (enrolled)"

# Asegurar que la columna sea numérica
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
ors="coerce")
```

```
# Boxplot para visualizar la distribución y posibles outliers
dataset[columna_a_trabajar].plot(kind="box")

# Definir rango válido (entre 0 y 20 unidades)
UMBRAL_MINIMO = 0
UMBRAL_MAXIMO = 20

# Calcular la media de la columna
media_columna = dataset[columna_a_trabajar].mean()

# Reemplazar valores menores al mínimo por la media
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trab
ajar] = media_columna

# Reemplazar valores mayores al máximo por la media
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trab
ajar] = media_columna

# Imputar valores NaN con la media
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_colu
mna)



# ---------------------------------------------------------
# Curricular units 1st sem (evaluations): número de evaluaciones en el 1.er se
mestre
# ---------------------------------------------------------

columna_a_trabajar = "Curricular units 1st sem (evaluations)"

# Asegurar que la columna sea numérica
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
ors="coerce")

# Boxplot para visualizar la distribución y posibles outliers
dataset[columna_a_trabajar].plot(kind="box")
```

```
# Definir rango válido (entre 0 y 40 evaluaciones como rango razonable)
UMBRAL_MINIMO = 0
UMBRAL_MAXIMO = 40

# Calcular la media de la columna
media_columna = dataset[columna_a_trabajar].mean()

# Reemplazar valores menores al mínimo por la media
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trab
ajar] = media_columna

# Reemplazar valores mayores al máximo por la media
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trab
ajar] = media_columna

# Imputar valores NaN con la media
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_colu
mna)



# ------------------------------------------------------------
# Curricular units 1st sem (approved): número de unidades aprobadas en el 1.
er semestre
# ------------------------------------------------------------

columna_a_trabajar = "Curricular units 1st sem (approved)"

# Asegurar que la columna sea numérica
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
ors="coerce")

# Boxplot para visualizar la distribución y posibles outliers
dataset[columna_a_trabajar].plot(kind="box")

# Definir rango válido (entre 0 y 20 unidades aprobadas)
```

```
UMBRAL_MINIMO = 0
UMBRAL_MAXIMO = 20

# Calcular la media de la columna
media_columna = dataset[columna_a_trabajar].mean()

# Reemplazar valores menores al mínimo por la media
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trab
ajar] = media_columna

# Reemplazar valores mayores al máximo por la media
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trab
ajar] = media_columna

# Imputar valores NaN con la media
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_colu
mna)



# ---------------------------------------------------------
# Curricular units 1st sem (grade): nota media del 1.er semestre, continua en
[0, 20]
# ---------------------------------------------------------

columna_a_trabajar = "Curricular units 1st sem (grade)"

# Asegurar que la columna sea numérica
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
ors="coerce")

# Boxplot para visualizar la distribución y posibles outliers
dataset[columna_a_trabajar].plot(kind="box")

# Definir rango válido para la nota promedio (entre 0 y 20)
UMBRAL_MINIMO = 0
UMBRAL_MAXIMO = 20
```

```
# Calcular la media de la columna
media_columna = dataset[columna_a_trabajar].mean()

# Reemplazar valores menores al mínimo por la media
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trab
ajar] = media_columna

# Reemplazar valores mayores al máximo por la media
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trab
ajar] = media_columna

# Imputar valores NaN con la media
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_colu
mna)
```

```
# ---------------------------------------------------------
# Curricular units 1st sem (without evaluations): unidades sin evaluación en el
1.er semestre
# ---------------------------------------------------------

columna_a_trabajar = "Curricular units 1st sem (without evaluations)"

# Asegurar que la columna sea numérica
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
ors="coerce")

# Boxplot para visualizar la distribución y posibles outliers
dataset[columna_a_trabajar].plot(kind="box")

# Definir rango válido (entre 0 y 20 unidades sin evaluación)
UMBRAL_MINIMO = 0
UMBRAL_MAXIMO = 20

# Calcular la media de la columna
```

```
media_columna = dataset[columna_a_trabajar].mean()

# Reemplazar valores menores al mínimo por la media
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trab
ajar] = media_columna

# Reemplazar valores mayores al máximo por la media
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trab
ajar] = media_columna

# Imputar valores NaN con la media
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_colu
mna)



# ---------------------------------------------------------
# Curricular units 2nd sem (credited): número de unidades acreditadas en el
2.º semestre
# ---------------------------------------------------------

columna_a_trabajar = "Curricular units 2nd sem (credited)"

# Asegurar que la columna sea numérica
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
ors="coerce")

# Boxplot para visualizar la distribución y posibles outliers
dataset[columna_a_trabajar].plot(kind="box")

# Definir rango válido (entre 0 y 20 unidades acreditadas)
UMBRAL_MINIMO = 0
UMBRAL_MAXIMO = 20

# Calcular la media de la columna
media_columna = dataset[columna_a_trabajar].mean()
```

```
# Reemplazar valores menores al mínimo por la media
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trab
ajar] = media_columna

# Reemplazar valores mayores al máximo por la media
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trab
ajar] = media_columna

# Imputar valores NaN con la media
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_colu
mna)



# ---------------------------------------------------------
# Curricular units 2nd sem (enrolled): unidades inscritas en el 2.º semestre
# ---------------------------------------------------------

columna_a_trabajar = "Curricular units 2nd sem (enrolled)"

# Asegurar que la columna sea numérica
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
ors="coerce")

# Boxplot para visualizar la distribución y posibles outliers
dataset[columna_a_trabajar].plot(kind="box")

# Definir rango válido (entre 0 y 20 unidades)
UMBRAL_MINIMO = 0
UMBRAL_MAXIMO = 20

# Calcular la media de la columna
media_columna = dataset[columna_a_trabajar].mean()

# Reemplazar valores menores al mínimo por la media
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trab
ajar] = media_columna
```

```
# Reemplazar valores mayores al máximo por la media
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trab
ajar] = media_columna

# Imputar valores NaN con la media
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_colu
mna)



# ---------------------------------------------------------
# Curricular units 2nd sem (evaluations): número de evaluaciones en el 2.º se
mestre
# ---------------------------------------------------------

columna_a_trabajar = "Curricular units 2nd sem (evaluations)"

# Asegurar que la columna sea numérica
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
ors="coerce")

# Boxplot para visualizar la distribución y posibles outliers
dataset[columna_a_trabajar].plot(kind="box")

# Definir rango válido (entre 0 y 40 evaluaciones como rango razonable)
UMBRAL_MINIMO = 0
UMBRAL_MAXIMO = 40

# Calcular la media de la columna
media_columna = dataset[columna_a_trabajar].mean()

# Reemplazar valores menores al mínimo por la media
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trab
ajar] = media_columna

# Reemplazar valores mayores al máximo por la media
```

```
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trab
ajar] = media_columna

# Imputar valores NaN con la media
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_colu
mna)
```

Esta transformación asegura la calidad de la variable de conteo Curricular units 2nd sem (approved), controlando outliers en un rango razonable de unidades aprobadas e imputando valores faltantes con la media.

```
# Curricular units 2nd sem (approved) - número de unidades aprobadas en el
2.º semestre

columna_a_trabajar = "Curricular units 2nd sem (approved)"

# Asegurar que la columna sea numérica
dataset[columna_a_trabajar] = pd.to_numeric(dataset[columna_a_trabajar], err
ors="coerce")

# Boxplot para revisar distribución y posibles outliers
dataset[columna_a_trabajar].plot(kind="box")

# Definir rango válido (entre 0 y 20 unidades aprobadas)
UMBRAL_MINIMO = 0
UMBRAL_MAXIMO = 20

# Calcular la media de la columna
media_columna = dataset[columna_a_trabajar].mean()

# Reemplazar valores menores al mínimo por la media
dataset.loc[dataset[columna_a_trabajar] < UMBRAL_MINIMO, columna_a_trab
ajar] = media_columna

# Reemplazar valores mayores al máximo por la media
dataset.loc[dataset[columna_a_trabajar] > UMBRAL_MAXIMO, columna_a_trab
```

```
ajar] = media_columna

# Imputar valores NaN con la media
dataset[columna_a_trabajar] = dataset[columna_a_trabajar].fillna(media_colu
mna)
```

Proceso ETL – Fase de Carga (Load)

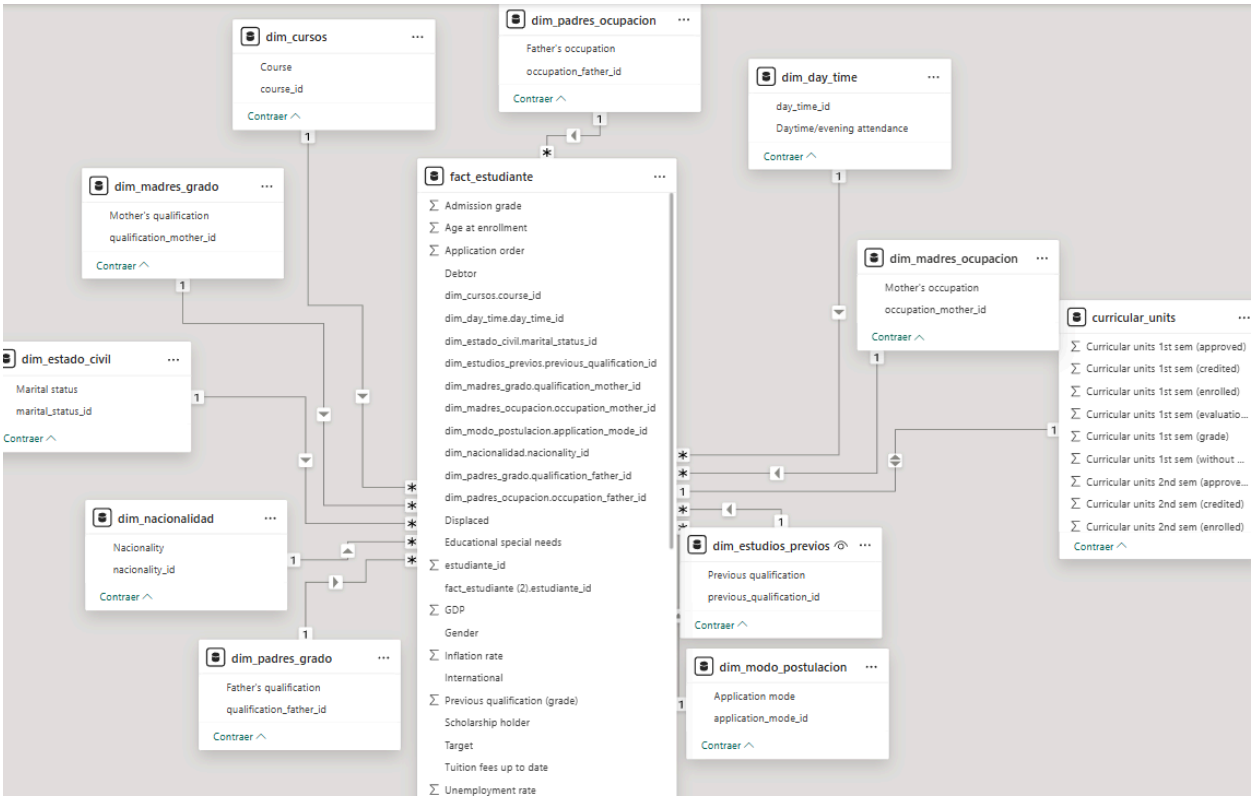
Una vez aplicadas todas las transformaciones (limpieza de outliers, imputación de valores nulos, conversión de códigos numéricos a categorías textuales, etc.), el DataFrame final se persiste en una **zona de salida analítica** mediante archivos estructurados. En este proyecto se sigue el siguiente esquema:

- Se genera un archivo **CSV limpio** (por ejemplo, dataset_final_limpio.csv) que contiene todas las columnas ya tratadas:
  - Variables demográficas y socioeconómicas transformadas a texto interpretables.
  - Variables académicas sin outliers fuera de sus rangos teóricos y sin valores faltantes.
  - Indicadores macroeconómicos normalizados.
  - La variable objetivo Target lista para tareas de clasificación (dropout, enrolled, graduate).

Estos archivos constituyen la **fuente oficial de datos para BI**:

- Power BI se conecta directamente al archivo CSV para construir los dashboards de:
  - deserción y retención;
  - rendimiento académico por semestre;
  - análisis de factores socioeconómicos y macroeconómicos.
- Al estar el dataset ya consolidado y documentado, se garantiza que todos los análisis partan de la **misma versión única de la verdad** (single source of truth), evitando discrepancias entre usuarios y reportes.

Arquitectura BI propuesta



1. Fuentes de Datos (Extract)

Datos necesarios:

- **Datos académicos**:
  - Calificaciones por semestre, unidades curriculares aprobadas, y evaluaciones previas.
  - Modalidad de asistencia (diurna/nocturna).
  - Historial académico del estudiante.
- **Datos demográficos**:
  - Edad, género, estado civil, nacionalidad, nivel educativo de los padres, y tipo de carrera.
- **Factores socioeconómicos externos**:

- **Tasa de desempleo**: Relacionada con la situación económica general que podría influir en la estabilidad de los estudiantes.
- **Tasa de inflación y PIB**: Datos económicos relevantes que podrían afectar la capacidad de los estudiantes para continuar con sus estudios.

Fuentes de datos:

1. **Bases de datos internas** de la universidad o institución educativa, como sistemas de gestión académica (ERP) y sistemas de información estudiantil (CRM).
2. **Sistemas de gestión de estudiantes** (LMS o plataformas académicas).
3. **APIs externas** o **datos abiertos** proporcionados por entidades gubernamentales o internacionales para obtener información económica como la tasa de desempleo, la inflación y el PIB (por ejemplo, **API de estadísticas económicas** de gobiernos o organizaciones como el Banco Mundial o el INEGI).
4. **Archivos históricos** de rendimiento académico almacenados en formatos CSV o Excel que contengan la información histórica de los estudiantes.

2. Almacenamiento de Datos (Data Warehouse)

- **Data Warehouse**:
  - **Hechos**: La **tabla de hechos** (en tu caso, fact_estudiante) contiene las métricas clave del análisis, como las calificaciones por semestre, la modalidad de estudio, la tasa de deserción, etc.
  - **Dimensiones**: Las **tablas de dimensiones** contienen detalles adicionales relacionados con las métricas, como el **estado civil**, **nivel educativo de los padres**, **ocupación del padre y madre**, **nacionalidad**, entre otros.

3. Modelo de Datos (Modelo Estrella)

El Modelo Estrella es el más adecuado para el Data Mart de Retención Estudiantil, utilizando una tabla central de hechos (`fact_estudiante`) y un conjunto de tablas de dimensiones directamente conectadas a ella.

1. Justificación Técnica: Rendimiento y Agilidad

El objetivo principal de un modelo de BI es servir métricas complejas rápidamente. El Modelo Estrella logra esto mejor que su alternativa, el Modelo Copo de Nieve (Snowflake).

- **Rendimiento en Consultas:**
  - En el Modelo Estrella, las consultas para calcular KPIs (ej. Tasa de Deserción por Carrera) solo requieren una **única unión (*join*)** entre la `fact_estudiante` y la dimensión relevante (ej. `dim_cursos`).
  - En el Modelo Copo de Nieve, esta consulta requeriría múltiples *joins* anidados lo que **aumenta la complejidad y el tiempo de ejecución**.
  - Para los Dashboards en vivo, cada milisegundo cuenta. El Modelo Estrella ofrece la ruta de consulta más corta y rápida, esencial para la **experiencia del usuario**.
- **Simplicidad del Modelo y Mantenimiento:**
  - La estructura plana del Modelo Estrella es **fácil de entender** por los analistas de BI y por el motor de la herramienta de visualización (Power BI).
  - El **mantenimiento es más sencillo**, ya que los cambios en una dimensión no requieren modificar la estructura de otras dimensiones o la tabla de hechos.

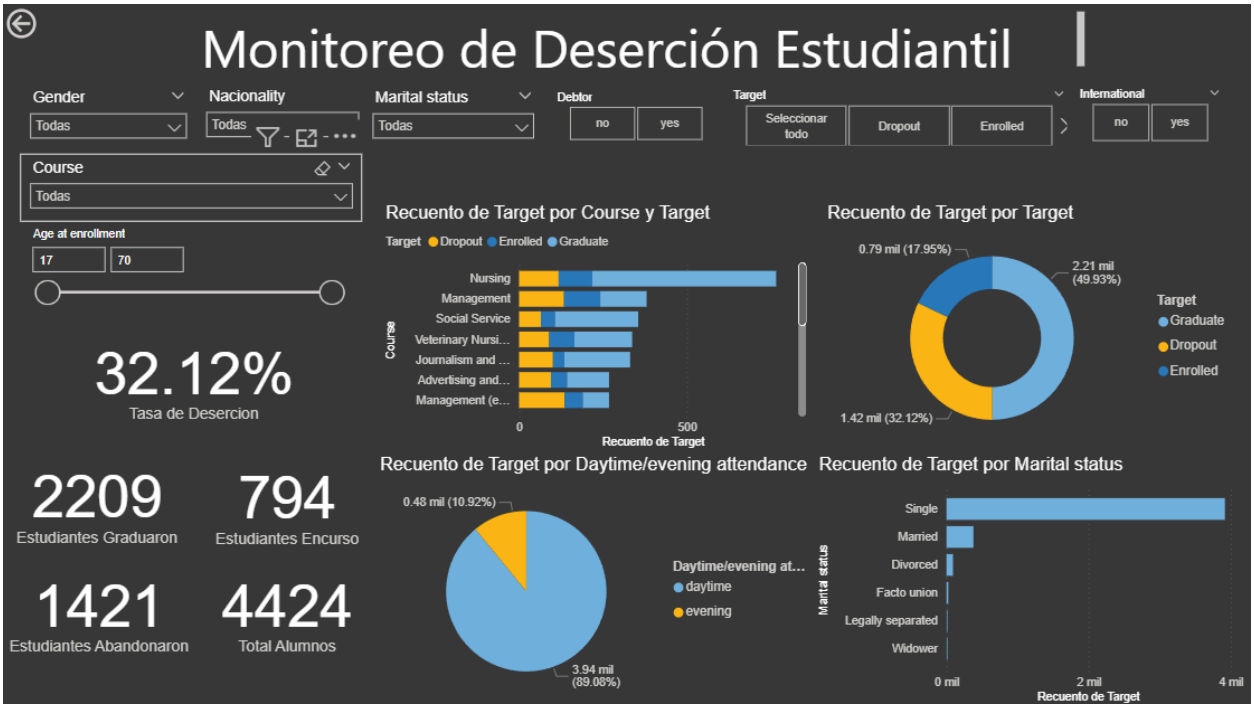
4. Transformación de Datos (ETL - Transform)

En esta fase, los datos extraídos de las fuentes son transformados para ser cargados en el Data Warehouse. El proceso de **transformación** incluye:

- **Limpieza de datos**: Como eliminar valores nulos o vacíos y corregir datos erróneos.
- **Imputación de valores faltantes**: Se imputan valores faltantes en variables categóricas y numéricas.
- **Normalización de datos**: Los datos numéricos y categóricos se ajustan para facilitar su análisis (por ejemplo, convertir valores numéricos en categorías textuales).

- **Eliminación de outliers**: Se identifican y eliminan valores atípicos para evitar que afecten el análisis.
- **Cálculos agregados**: Se realizan cálculos adicionales, como el promedio de calificaciones, la tasa de deserción por carrera, etc.

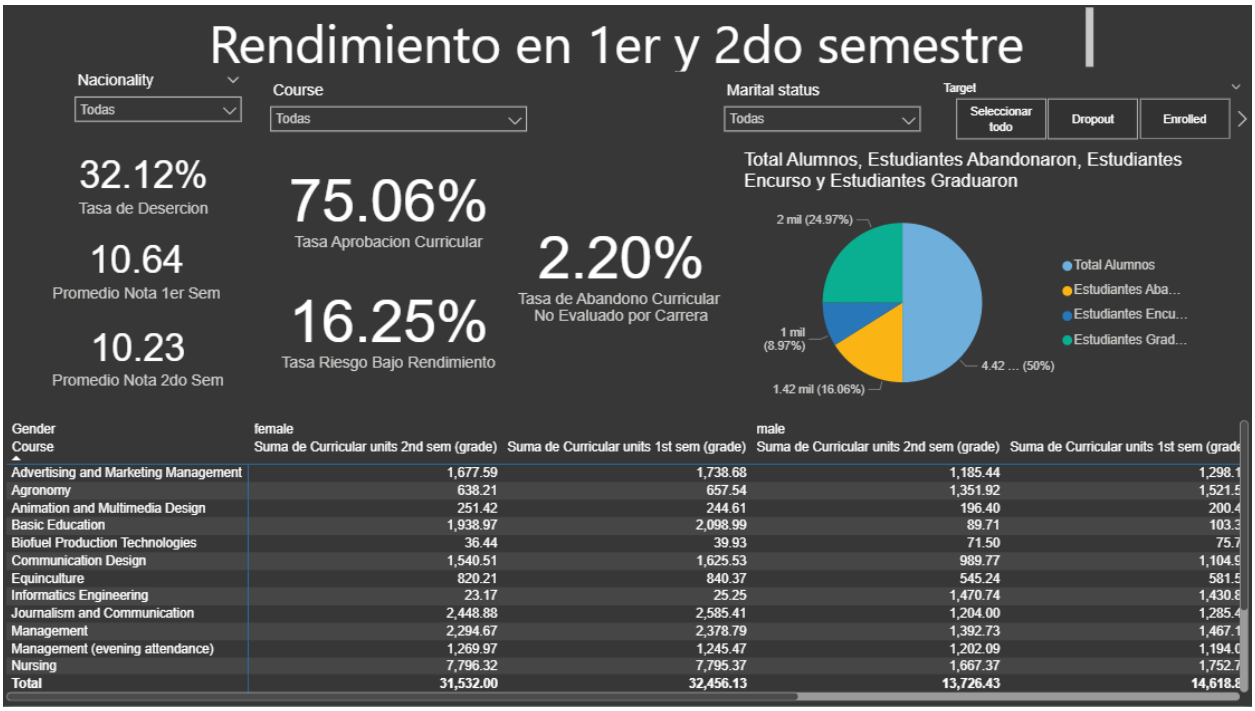
5. Visualización de Datos (Power BI)



Este dashboard ofrece una visión integral sobre el **rendimiento académico** y el **problema de deserción** dentro de la institución.

El objetivo principal es medir la **magnitud del abandono estudiantil** y señalar las áreas y grupos demográficos donde el problema es más prevalente.

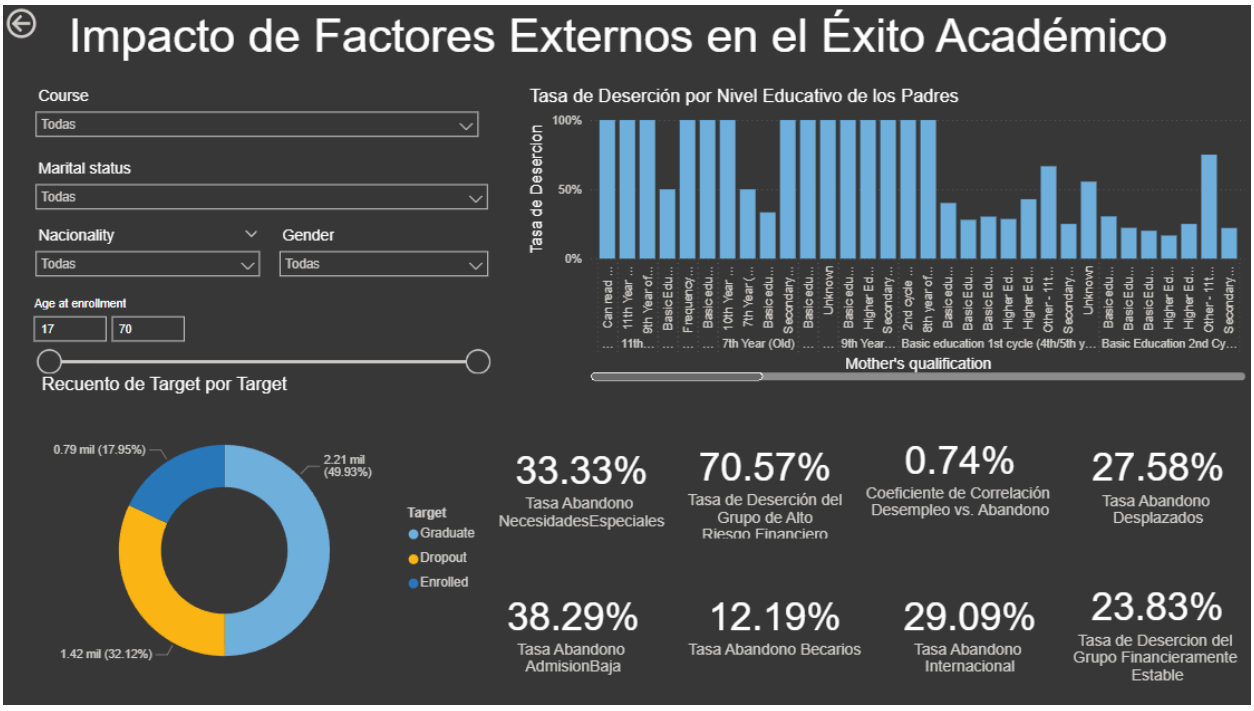
- **Métrica Central:** Se destaca de forma prominente la **Tasa de Deserción** para establecer la gravedad general del problema.
- **Volumen de Estudiantes:** Se muestran los números absolutos de estudiantes que se han **graduado**, han **abandonado**, y están **actualmente en curso**.



Este dashboard ofrece una visión integral sobre el rendimiento académico y el problema de deserción dentro de la institución.

El objetivo principal es medir la **magnitud del abandono estudiantil** y señalar las áreas y grupos demográficos donde el problema es más prevalente.

- **Métrica Central:** Se destaca de forma prominente la **Tasa de Deserción** para establecer la gravedad general del problema. Además, incorpora métricas de desempeño cruciales como la **Tasa de Aprobación Curricular**, el **Promedio de Notas** del 1er y 2do semestre, y la **Tasa de Riesgo por Bajo Rendimiento**.
- **Volumen de Estudiantes:** Se muestran los números absolutos de estudiantes que se han **graduado**, han **abandonado**, y están **actualmente en curso** a través de un gráfico de distribución.



Este dashboard ofrece una visión integral sobre el rendimiento académico y el problema de deserción dentro de la institución.

El objetivo principal es medir la **magnitud del abandono estudiantil** y señalar las áreas y grupos demográficos donde el problema es más prevalente, con un énfasis particular en las **influencias externas** y los **factores socioeconómicos**.

Objetivos

1. **Predecir y Reducir la Deserción Estudiantil**:
    - Analizar y predecir qué factores están más relacionados con la deserción estudiantil para implementar medidas preventivas.
2. **Mejorar el Rendimiento Académico**:
    - Identificar qué factores, como la preparación académica previa, el nivel educativo de los padres o la modalidad de asistencia, influyen en el rendimiento académico y diseñar estrategias de apoyo para los estudiantes.
3. **Identificar Factores de Riesgo de Abandono**:

- Analizar el impacto de factores socioeconómicos, como el estado civil, la nacionalidad y el nivel educativo de los padres, en el éxito académico.

4. **Optimizar los Recursos Académicos y Administrativos**:
   - Determinar qué áreas de apoyo académico necesitan ser reforzadas (por ejemplo, tutorías, programas de orientación) para reducir la deserción y mejorar el rendimiento.
5. **Monitorear el Progreso Académico en Tiempo Real**:
   - Crear un **dashboard interactivo** que permita monitorear las tasas de deserción y graduación en tiempo real, con filtros dinámicos para obtener insights según diferentes dimensiones.

KPIs (Indicadores Clave de Desempeño)

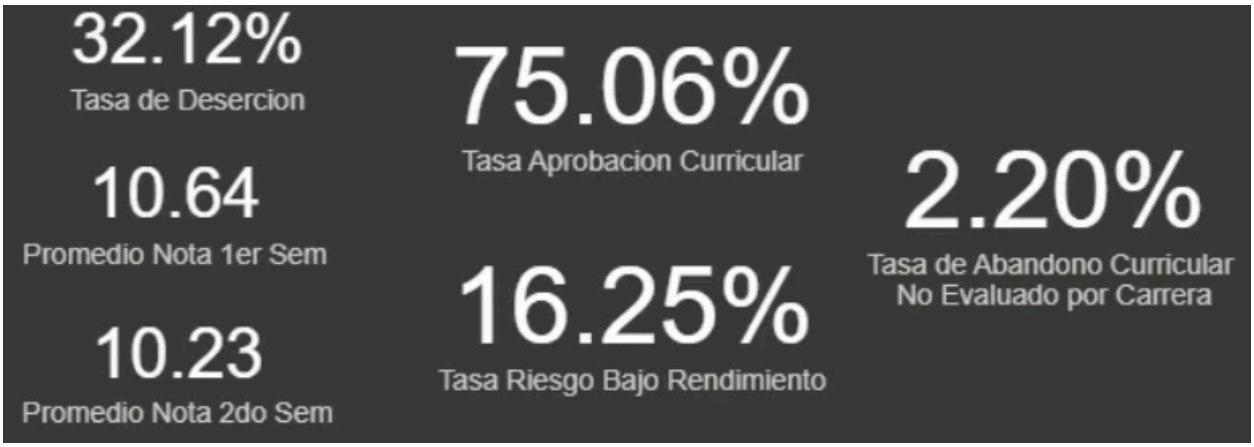
1. Deserción Estudiantil

- **Tasa de Deserción**

    - **Objetivo:** Medir la proporción de estudiantes que abandonan la universidad en un período específico.
    - **Insight esperado:** Identificar las **carreras o programas académicos** que concentran las tasas de deserción más altas, permitiendo priorizar recursos por área de estudio.
- **Numero de Riesgo de Deserción**
    - **Objetivo:** Calcular elnumero de deserción de los estudiantes y permitir intervenciones proactivas.
    - **Insight esperado:** Proporcionar un numero predictivo para que el director de cada programa pueda focalizar los esfuerzos y recursos en sus estudiantes más vulnerables.
- **Numero de Retención**
    - **Objetivo:** Medir el numero de estudiantes que permanecen en la institución a lo largo de un período académico.
    - **Insight esperado:** Evaluar la **efectividad de los planes de estudio y las estrategias de apoyo** implementadas por cada departamento o programa académico para mantener a los estudiantes matriculados.

2. Rendimiento Académico y Riesgo Temprano

Esta sección se enfoca en las métricas del primer semestre que actúan como **señales de alerta** para la gerencia de cada programa (Carrera), sin repetir las tasas de resultado.



- **Promedio de Calificaciones (GPA) por Carrera**
  - **Objetivo:** Mostrar el GPA promedio para cada programa académico.
  - **Insight esperado:** Identificar qué carreras tienen un **rendimiento académico inicial bajo**, lo que suele correlacionarse con una alta Tasa de Deserción (vista en el Punto 1). Esto ayuda a enfocar los recursos de tutoría en las carreras más afectadas.
- **Tasa de Riesgo por Bajo Rendimiento por Carrera**
  - **Objetivo:** Cuantificar el porcentaje de estudiantes en cada carrera que obtuvieron un GPA inferior al umbral mínimo de aprobación (ej. < 10) en el primer semestre.
  - **Insight esperado:** Determinar dónde se concentra la **población de alto riesgo académico**. Una alta tasa aquí, combinada con una alta Tasa de Deserción, indica un problema en el ajuste inicial a la carga académica de ese programa.
- **Tasa de Aprobación Curricular por Carrera**
  - **Objetivo:** Medir la eficiencia académica de cada carrera: el porcentaje de unidades inscritas que fueron aprobadas por sus estudiantes.
  - **Insight esperado:** Identificar las carreras donde el **programa de estudio es percibido como demasiado difícil** o donde la metodología de enseñanza no es efectiva. Una tasa baja aquí es un indicador de cuellos de botella curriculares.
- **Tasa de Abandono Curricular No Evaluado por Carrera**

- **Objetivo:** Cuantificar el porcentaje de unidades curriculares inscritas que los estudiantes de cada carrera no presentaron a evaluación.
- **Insight esperado:** Medir el **riesgo motivacional y de inasistencia** por carrera. Un valor alto aquí sugiere que los estudiantes de ese programa se están rindiendo temprano, incluso antes de formalizar el abandono.

3. Impacto de Factores Externos en el Éxito Académico

Esta sección se enfoca en métricas externas (económicas, demográficas y sociales) que el estudiante trae consigo o que experimenta fuera del campus. El objetivo es cuantificar el riesgo social y económico para diseñar intervenciones de soporte que no son puramente académicas.

- **Tasa de Abandono del Grupo con Alto Riesgo Financiero y Tasa de Abandono del Grupo Financieramente Estable.**
  - **Objetivo:** Determinar la **sensibilidad de la deserción** a la inestabilidad económica personal (ser deudor o tener pagos pendientes). La comparación con el Grupo Estable sirve para dimensionar el problema.
  - **Insight Esperado:** Si la **Tasa de Alto Riesgo** es sustancialmente mayor que la **Tasa Estable**, se confirma que la presión económica es el **factor de**

expulsión primario. Esto justifica la priorización de fondos para becas de retención o el desarrollo de programas de *counseling* financiero.

- **Tasa de Abandono de Estudiantes con Necesidades Especiales**
  - **Objetivo:** Medir la Tasa de Abandono en el segmento de estudiantes que requieren **apoyo educativo especializado**.
  - **Insight esperado:** Si esta tasa es alta, indica una posible **deficiencia en los recursos de inclusión o de accesibilidad** del campus. Esto justifica la revisión de los programas de soporte a la discapacidad y necesidades educativas especiales.
- **Coeficiente de Correlación Desempleo vs. Abandono.**
  - **Objetivo:** Cuantificar la **fuerza y dirección** de la relación estadística entre la Tasa de Desempleo (un factor macroeconómico externo) y la Tasa de Deserción de la institución. Este coeficiente sirve como un **indicador de alerta temprana**.
  - **Insight esperado:** El coeficiente debe ser cercano a **+1** (correlación positiva). Esto significa que cuando la **Tasa de Desempleo sube**, la **Tasa de Deserción también sube** (y viceversa).
- **Tasa Abandono Desplazados**
  - **Objetivo:** Medir si el esfuerzo y los desafíos logísticos de **reubicación geográfica** (ser un estudiante clasificado como "desplazado")
  - **Insight esperado:** Si la tasa de abandono de los estudiantes **Desplazados** es notablemente superior a la de los estudiantes no desplazados, se confirma que la **adaptación geográfica y el soporte logístico/social** son un punto de falla en la retención.
- **Tasa Abandono Internacional**
  - **Objetivo:** Cuantificar el riesgo de deserción en el segmento de estudiantes que no son nacionales. Busca identificar si las barreras específicas de este grupo (idioma, adaptación cultural, problemas de visa o falta de redes de apoyo) son un factor de riesgo significativo.
  - **Insight esperado:** Si la tasa de abandono de los **Estudiantes Internacionales** es alta, señala una deficiencia en los **servicios de**

integración y apoyo específico del campus.

- **Tasa Abandono Becarios.**
  - **Objetivo:** Medir la **eficacia de la intervención** (la beca) como estrategia de retención. Cuantifica la tasa de deserción específica en el segmento de estudiantes que recibieron apoyo financiero
  - **Insight esperado:** Si la tasa de abandono de los **Becarios** es **alta o similar** a la tasa promedio o a la del grupo de alto riesgo financiero, indica que el **apoyo financiero solo no es suficiente** para asegurar la permanencia.
- **Tasa Abandono AdmisionBaja.**
  - **Objetivo:** Determinar si el **rendimiento académico inicial** (nota de admisión inferior a 120) es un predictor confiable de riesgo de deserción. Aísla a este grupo para medir su vulnerabilidad.
  - **Insight esperado:** Si la tasa de abandono de los estudiantes con notas de ingreso bajas es **sustancialmente superior** al promedio, se confirma una **debilidad de preparación inicial** y se valida la necesidad de intervención.

Resultados Esperados

El resultado del proyecto es un conjunto de tres Dashboards especializados, diseñados para que cada nivel de la organización pueda tomar decisiones basadas en datos específicos.

1. Monitoreo de Deserción Estudiantil

Este Dashboard es el **Resumen Ejecutivo de Resultados**. Ofrece una visión agregada y el diagnóstico inicial del problema.

- **Tipo de Reporte:** Dashboard de Monitoreo y Resultados Ejecutivos.
- **Contenido Clave:**
  - **Métricas Absolutas:** Tasa de Deserción general (ej., 32.12%), Total Graduaron (ej., 2209), Total Abandonaron (ej., 1421), y Total En Curso (ej., 794).
  - **Análisis de Distribución:** Gráficos de barras que muestran la deserción segmentada por **Carrera** (identificando las más afectadas) y por variables

logísticas (ej., asistencia diurna/nocturna).

- **Usuarios que lo Consultarán: Rectoría, Alta Dirección** (para establecer metas de retención), y **Planificación Estratégica** (para evaluación de impacto global).
- **Propósito:** Responder *¿Cuál es la magnitud del problema?* y *¿Qué carreras o grupos concentran la mayor pérdida de estudiantes?*

2. Rendimiento Académico y Riesgo Temprano

Este Dashboard es el **Sistema de Alerta Temprana** enfocado en la gestión del programa académico.

- **Tipo de Reporte:** Dashboard Táctico y de Alerta Temprana.
- **Contenido Clave:**
  - **Rendimiento Inicial:** Promedio de Notas del 1er y 2do Semestre.
  - **Riesgo Curricular:** Tasa de Aprobación Curricular (mide cuellos de botella) y Tasa de Abandono Curricular No Evaluado (mide riesgo motivacional).
  - **Riesgo Individual:** Tasa Riesgo Bajo Rendimiento (porcentaje de estudiantes con GPA inferior al umbral mínimo).
- **Usuarios que lo Consultarán: Directores de Carrera, Coordinadores Académicos, Tutores,** y **Equipo Operativo** (para acciones de tutoría e intervención inmediata).
- **Propósito:** Responder *¿Qué está fallando dentro del campus y en qué momento (1er o 2do semestre)?* y *¿Dónde debemos enfocar los recursos de tutoría?*

3. Impacto de Factores Externos

Este Dashboard es la **Herramienta de Justificación Financiera y Soporte Social**. Mide el riesgo inherente al perfil socioeconómico del estudiante.

- **Tipo de Reporte:** Dashboard Estratégico y de Análisis de Causa Raíz Externa.
- **Contenido Clave:**

    - **Riesgo Financiero:** Comparación crítica entre **Tasa de Deserción del Grupo de Alto Riesgo Financiero**
    - **Riesgos Focales:** Tarjetas KPI para segmentos vulnerables: Tasa Abandono Becarios, Desplazados, Internacionales, AdmisionBaja, y Necesidades Especiales.
    - **Predicción:** Coeficiente de Correlación Desempleo vs. Abandono
- **Usuarios que lo Consultarán: Dirección Financiera** (para presupuestar becas), **Bienestar Estudiantil** (para diseñar programas de apoyo social), y **Planificación Estratégica** (para decisiones sobre políticas de admisión).
- **Propósito:** Responder *¿La causa del abandono es económica o académica?* y *¿Cómo podemos anticipar y mitigar el riesgo externo?*

Conclusiones

El éxito del proyecto de BI no solo radica en la **construcción técnica** de un modelo y sus *dashboards*, sino en su capacidad de transformar la cultura organizacional y la toma de decisiones en la institución de educación superior. Esta herramienta pasa de ser un repositorio de datos históricos a un motor de **acción proactiva** y **planificación estratégica**.

El modelo de BI redefine la gestión de la deserción, elevándola de un problema meramente operativo (registros y encuestas) a un **imperativo estratégico** para la sostenibilidad y calidad institucional.

La implementación efectiva del modelo requiere una integración fluida con los procesos diarios de los equipos académicos, de bienestar y administrativos.

El *dashboard* permite a los **coordinadores y tutores** identificar cohortes y estudiantes individuales con alta probabilidad de deserción **antes** de que el problema se manifieste.

En resumen, el proyecto ha logrado crear un **activo institucional** que va más allá de los informes estáticos. Ha sentado las bases para un ecosistema de gestión educativa donde los datos son la **moneda de cambio** para la **proactividad**, la **eficiencia operativa** y, en última instancia, el **éxito estudiantil**.