

Reporte de Análisis Exploratorio: Dataset CTG

Alumno: JOSUE JIMENEZ APODACA Fecha: 22-11-2'25

1. Introducción

Este reporte utiliza la librería personalizada `ctg_viz` para analizar datos de cardiocotografía, limpiar el dataset y encontrar patrones visuales.

```
In [1]: import pandas as pd
import sys
import os
import numpy as np
import inspect
from IPython.display import display, Markdown, HTML
import re

sys.path.append(os.path.abspath('../'))

from ctg_viz.preprocessing import remove_null_columns, impute_missing_values, detect_handle_outliers
from ctg_viz.utils import check_data_completeness_JosueJimenezApodaca
from ctg_viz.plots.histograms import plot_histogram_interactivo
from ctg_viz.plots.boxplots import plot_boxplot
from ctg_viz.plots.barplots import plot_bar
from ctg_viz.plots.heatmap import plot_correlation_heatmap
from ctg_viz.plots.density import plot_violin

df = pd.read_csv('../data/CTG.csv')
print("Datos cargados:", df.shape)

import plotly.io as pio
pio.renderers.default = "notebook_connected"
```

Datos cargados: (2129, 40)

```
In [2]: def limpiar_firma(firma: str) -> str:
    """Limpia los tipos de datos largos para que sean legibles."""
    firma = firma.replace("pandas.core.frame.DataFrame", "pd.DataFrame")
    firma = firma.replace("pandas.core.series.Series", "pd.Series")
    firma = firma.replace("typing.", "")
    firma = firma.replace("NoneType", "None")
    firma = firma.replace("matplotlib.figure.Figure", "plt.Figure")
    return firma

def formatear_docstring(doc: str) -> str:
    """Da formato HTML básico al docstring (negritas en Args/Returns)."""
    if not doc:
        return "<span style='color: #999; font-style: italic;'>Sin documentación provista.</span>"

    # Escapar caracteres HTML para seguridad
    doc = doc.replace("<", "&lt;").replace(">", "&gt;")

    # Resaltar secciones clave (Args:, Returns:, Escenario:, etc.)
    keywords = ["Args:", "Returns:", "Raises:", "Escenario:", "Resultado Esperado:"]
    for kw in keywords:
        doc = doc.replace(kw, f"<br><b>{kw}</b>")

    # Convertir saltos de línea en <br> para HTML
    doc = doc.replace("\n", "<br>")
    return doc

def documentacion_estetica(modulo, titulo="Documentación Técnica"):
    """Genera tarjetas HTML elegantes para las funciones del módulo."""
    html_content = f"""
<div style="font-family: 'Segoe UI', sans-serif; margin-bottom: 30px;">
    <h2 style="border-bottom: 2px solid #4CAF50; padding-bottom: 10px; color: #333;">
```

```

{titulo}: <span style="color: #4CAF50;">{modulo.__name__}</span>
</h2>
"""

funciones = inspect.getmembers(modulo, inspect.isfunction)

for nombre, func in funciones:
    # Filtramos funciones privadas (que empiezan con _)
    if not nombre.startswith('_'):
        # 1. Obtener firma limpia
        try:
            firma = str(inspect.signature(func))
            firma = limpiar_firma(firma)
        except ValueError:
            firma = "()"

        # 2. Obtener y formatear docstring
        doc_raw = inspect.getdoc(func)
        doc_html = formatear_docstring(doc_raw)

        # 3. Crear la tarjeta HTML
        html_content += f"""
<div style="border: 1px solid #e0e0e0; border-radius: 8px; margin-bottom: 20px; box-shadow: 0 2px 4px rgba(0,0,0,0.05); overflow: hidden;">
  <div style="background-color: #f8f9fa; padding: 10px 15px; border-bottom: 1px solid #e0e0e0;">
    <code style="color: #0d47a1; font-weight: bold; font-size: 1.1em;">def {nombre}{firma}</code>
  </div>
  <div style="padding: 15px; background-color: white; color: #444; line-height: 1.6;">
    {doc_html}
  </div>
</div>
"""

html_content += "</div>"
display(HTML(html_content))

# Título del Reporte
display(Markdown("# Reporte Técnico y Análisis Exploratorio - CTG"))
display(Markdown("Este documento detalla la arquitectura de la librería `ctg_viz` y presenta el análisis de datos."))

```

Reporte Técnico y Análisis Exploratorio - CTG

Este documento detalla la arquitectura de la librería `ctg_viz` y presenta el análisis de datos.

```

In [3]: import ctg_viz.preprocessing as prep
import ctg_viz.utils as utils
import ctg_viz.plots as plots

display(Markdown("## 1. Arquitectura de la Librería Personalizada"))
documentacion_estetica(prep, titulo="Módulo de Preprocesamiento")
documentacion_estetica(utils, titulo="Módulo de Utilidades")
documentacion_estetica(plots.histograms, titulo="Módulo de Gráficas - Histogramas")
documentacion_estetica(plots.boxplots, titulo="Módulo de Gráficas - Boxplots")
documentacion_estetica(plots.barplots, titulo="Módulo de Gráficas - Barplots")
documentacion_estetica(plots.density, titulo="Módulo de Gráficas - Densidad")
documentacion_estetica(plots.heatmap, titulo="Módulo de Gráficas - Heatmap")

# Agregamos la carpeta ../tests
tests_path = os.path.abspath(os.path.join(os.getcwd(), '..', 'tests'))
if tests_path not in sys.path:
    sys.path.append(tests_path)

try:
    import test_preprocessing
except ImportError:
    # Fallback por si el nombre o la ruta varía
    print("No se pudo importar el módulo de tests para leer docstrings.")
    test_preprocessing = None

```

```
# A) Mostrar la documentación de las funciones de test  
if test_preprocessing:  
    documentacion_estetica(test_preprocessing, titulo="Módulo de Tests - Funciones de preprocesamiento")
```

1. Arquitectura de la Librería Personalizada

 Módulo de Preprocesamiento: `ctg_viz.preprocessing`

```
def detect_handle_outliers(df: pd.DataFrame, method: str = 'iqr', return_plots: bool = False) -> Union[pd.DataFrame, Tuple[pd.DataFrame, Dict]]
```

Función para detectar y manejar outliers en columnas numéricas.
Se trunca los valores fuera de los límites definidos por el método seleccionado
Valores atípicos serán llevados al límite más cercano.

Args:

df (pd.DataFrame): DataFrame numérico.
method (str): 'iqr' o 'z-score'.

Returns:

return_plots (bool): Si True, devuelve una tupla (DataFrame, Diccionario de Figuras).
Si False, devuelve solo el DataFrame.

```
def impute_missing_values(df: pd.DataFrame, use_knn: bool = False) -> pd.DataFrame
```

Función para imputar valores faltantes en el DataFrame.
Se puede usar KNN para numéricos, si no se aplica KNN aplica entonces Mediana por defecto.
Para discretas siempre es la moda.
Se incluye tratamiento separado para variables categóricas y numéricas.

Args:

df (pd.DataFrame): DataFrame numérico.
method (str): 'iqr' o 'z-score'.

Returns:

return_plots (bool): Si True, devuelve una tupla (DataFrame, Diccionario de Figuras).
Si False, devuelve solo el DataFrame.

Reglas:

- Continuas (Numéricas > 10 valores): Usan Mediana o KNN.
- Discretas (Numéricas <= 10 valores) y Categóricas: Usan MODA.

```
def remove_null_columns(df: pd.DataFrame, threshold: float = 0.2) -> pd.DataFrame
```

Función para eliminar columnas con un porcentaje de valores nulos definido por el umbral.
En este caso, se elimina si más del 20% de los datos son nulos.

Args:

df (pd.DataFrame): DataFrame numérico.
method (str): 'iqr' o 'z-score'.

returns:
return_plots (bool): Si True, devuelve una tupla (DataFrame, Diccionario de Figuras).
Si False, devuelve solo el DataFrame.

Módulo de Utilidades: `ctg_viz.utils`

```
def check_data_completeness_JosueJimenezApodaca(df: pd.DataFrame) -> pd.DataFrame
```

Analiza el dataset y retorna un resumen de completitud, tipos y estadísticas.
Cumple con los requisitos de conteo de nulos, porcentajes, tipos y
clasificación automática de variables continuas/discretas.

Args:

df (pd.DataFrame): El dataset a analizar.

Returns:

pd.DataFrame: Resumen con columnas:
[Nulos, % Completitud, Tipo Dato, Estadísticas, Categoría Auto]

Módulo de Gráficas - Histogramas: `ctg_viz.plots.histograms`

```
def plot_histogram_interactivo(df: pd.DataFrame, col: str, group_by: Optional[str] = None) -> plotly.graph_objs._figure.Figure
```

Histograma interactivo con gráfico marginal de caja (Boxplot superior).
El gráfico permite agrupar por una variable categórica opcional.
Las barra superiores muestran el boxplot de la variable numérica.

Args:

df (pd.DataFrame): Datos.
col (str): Variable numérica.
group_by (str, optional): Variable categórica para agrupar colores.

Returns:

go.Figure: Gráfico interactivo.

 Módulo de Gráficas - Boxplots: `ctg_viz.plots.boxplots`

```
def plot_boxplot(df: pd.DataFrame, x: str, y: str, facet_col: Optional[str] = None) -> plotly.graph_objs._figure.Figure
```

Boxplot interactivo con opción de faceting (subgráficos).
Permite visualizar la distribución de una variable numérica (y)
segmentada por una variable categórica (x) y opcionalmente dividida
en columnas por otra variable categórica (facet_col).

Args:

df (pd.DataFrame): Datos.
x (str): Variable categórica (Eje X).
y (str): Variable numérica (Eje Y).
facet_col (str, optional): Variable para dividir en columnas.

Returns:

go.Figure: Gráfico de boxplot interactivo.

 Módulo de Gráficas - Barplots: `ctg_viz.plots.barplots`

```
def plot_bar(df: pd.DataFrame, col: str, horizontal: bool = False) -> plotly.graph_objs._figure.Figure
```

Gráfico de barras interactivo ordenado por frecuencia.
Se utiliza para variables categóricas/discretas.
Permite orientación horizontal o vertical con el argumento 'horizontal'.

Args:

df (pd.DataFrame): Dataset con la variable a graficar.
col (str): Nombre de la columna categórica/discreta.
horizontal (bool): Si es True, el gráfico será horizontal. Default es False (vertical).

Returns:

go.Figure: Gráfico de barras ordenado por frecuencia.

 Módulo de Gráficas - Densidad: `ctg_viz.plots.density`

```
def plot_violin(df: pd.DataFrame, x: str, y: str) -> plotly.graph_objs._figure.Figure
```

Permite crear un violin plot interactivo que muestra la densidad y los puntos subyacentes.

Args:

df (pd.DataFrame): Dataset con los datos.

x (str): Nombre de la columna categórica para el eje X.

y (str): Nombre de la columna numérica para el eje Y.

Returns:

go.Figure: Objeto de figura de Plotly con el violin plot.

 Módulo de Gráficas - Heatmap: `ctg_viz.plots.heatmap`

```
def plot_correlation_heatmap(df: pd.DataFrame, method: str = 'pearson') -> plotly.graph_objs._figure.Figure
```

Genera un Heatmap interactivo de correlación optimizado para muchas variables.

Optimiza el tamaño dinámicamente según la cantidad de variables.

Permite elegir el método de correlación.

Args:

df (pd.DataFrame): Datos.

method (str): Método de correlación ('pearson', 'spearman').

Returns:

go.Figure: Objeto figura de Plotly listo para Streamlit.

 Módulo de Tests - Funciones de preprocesamiento: `test_preprocessing`

```
def check_data_completeness_JosueJimenezApodaca(df: pd.DataFrame) -> pd.DataFrame
```

Analiza el dataset y retorna un resumen de completitud, tipos y estadísticas.

Cumple con los requisitos de conteo de nulos, porcentajes, tipos y clasificación automática de variables continuas/discretas.

Args:

df (pd.DataFrame): El dataset a analizar.

Returns:

pd.DataFrame: Resumen con columnas:

[Nulos, % Completitud, Tipo Dato, Estadísticas, Categoría Auto]

```
def detect_handle_outliers(df: pd.DataFrame, method: str = 'iqr', return_plots: bool = False) -> Union[pd.DataFrame, Tuple[pd.DataFrame, Dict]]
```

Función para detectar y manejar outliers en columnas numéricas.

Se trunca los valores fuera de los límites definidos por el método seleccionado

Valores atípicos serán llevados al límite más cercano.

Args:

df (pd.DataFrame): DataFrame numérico.

method (str): 'iqr' o 'z-score'.

Returns:

return_plots (bool): Si True, devuelve una tupla (DataFrame, Diccionario de Figuras).

Si False, devuelve solo el DataFrame.

```
def impute_missing_values(df: pd.DataFrame, use_knn: bool = False) -> pd.DataFrame
```

Función para imputar valores faltantes en el DataFrame.

Se puede usar KNN para numéricos, si no se aplica KNN aplica entonces Mediana por defecto.

Para discretas siempre es la moda.

Se incluye tratamiento separado para variables categóricas y numéricas.

Args:

df (pd.DataFrame): DataFrame numérico.

method (str): 'iqr' o 'z-score'.

Returns:

return_plots (bool): Si True, devuelve una tupla (DataFrame, Diccionario de Figuras).

Si False, devuelve solo el DataFrame.

Reglas:

- Continuas (Numéricas > 10 valores): Usan Mediana o KNN.
- Discretas (Numéricas <= 10 valores) y Categóricas: Usan MODA.

```
def remove_null_columns(df: pd.DataFrame, threshold: float = 0.2) -> pd.DataFrame
```

Función para eliminar columnas con un porcentaje de valores nulos definido por el umbral.
En este caso, se elimina si más del 20% de los datos son nulos.

Args:

df (pd.DataFrame): DataFrame numérico.

method (str): 'iqr' o 'z-score'.

returns:

return_plots (bool): Si True, devuelve una tupla (DataFrame, Diccionario de Figuras).

Si False, devuelve solo el DataFrame.

```
def test_check_data_completeness_structure(sample_df)
```

Valida la estructura y lógica de clasificación del reporte de completitud.

Escenario:

Se analiza el DataFrame de prueba. 'col_good' es numérica con <10 valores únicos.

Resultado Esperado:

- El resultado debe ser un DataFrame.
- Debe contener las columnas obligatorias: 'Nulos', '% Completitud', 'Tipo Dato'.
- 'col_good' debe clasificarse automáticamente como 'Discreta' según la regla de negocio (<10 únicos).

```
def test_detect_handle_outliers_iqr(sample_df)
```

Valida el recorte (clipping) de valores atípicos usando el Rango Intercuartílico (IQR).

La variable debe tener mas de 10 valores únicos para ser considerada continua.

Escenario:

La columna 'col_outlier' tiene un valor extremo (1000) muy lejos de la distribución normal (5-6).

Se aplica el método 'iqr'.

Resultado Esperado:

- El valor 1000 debe ser reducido al límite superior calculado (aprox 7.5).
- El valor no debe ser eliminado (la fila persiste), solo transformado.

```
def test_impute_missing_values_mode(sample_df)
```

Valida la imputación de la Moda para variables categóricas.

Escenario:

La columna 'col_cat' tiene un valor nulo y la moda es 'A' (aparece 3 veces).

Se ejecuta la imputación con KNN.

Resultado Esperado:

- No deben quedar valores nulos en la columna.
- El valor nulo original debe ser reemplazado por 'A'.

```
def test_remove_null_columns(sample_df)
```

Valida la eliminación de columnas que superan el umbral de nulos.

Escenario:

Se pasa un DataFrame donde 'col_nulls' tiene 60% de valores faltantes.

El umbral permitido es 20% (0.2).

Resultado Esperado:

- La columna 'col_nulls' debe desaparecer del DataFrame resultante.
- La columna 'col_good' (0% nulos) debe conservarse.

2. Limpieza de Datos

Aplicamos reglas: eliminar columnas que presenten un porcentaje mayor al 20% de valores nulos e imputar valores faltantes (Numericos -> mediana y categoricos -> moda)

```
In [4]: # --- BLOQUE 1: ANÁLISIS DEL ESTADO INICIAL (RAW) ---
print("=== 1. ESTADO ORIGINAL DEL DATASET ===")
print(f"Dimensiones Iniciales: {df.shape[0]} filas x {df.shape[1]} columnas")

display(df.head())

# Usamos tu función personalizada para ver Tipos y Nulos antes de limpiar
print("\n--- Reporte de Calidad (Antes de Limpieza) ---")
reporte_inicial = check_data_completeness_JosueJimenezApodaca(df)
display(reporte_inicial)

print("\n--- Estadísticos Descriptivos (Originales) ---")
display(df.describe())

# --- BLOQUE 2: PROCESAMIENTO (LIMPIEZA) ---
print("\n" + "="*40)
print("=== APLICANDO PIPELINE DE LIMPIEZA ===")
print("="*40)

# 1. Eliminar columnas con exceso de nulos (>20%)
df_clean_step1 = remove_null_columns(df, threshold=0.2)
cols_eliminadas = set(df.columns) - set(df_clean_step1.columns)
print(f"-> Columnas eliminadas por >20% nulos: {cols_eliminadas if cols_eliminadas else 'Ninguna'}")
```

```

# 2. Imputación (KNN o Moda/Mediana)
# En caso de variables numéricas, usamos KNN; para categóricas, usamos la moda.
# Si no quieres usar KNN, cambia use_knn a False
df_clean_step2 = impute_missing_values(df_clean_step1, use_knn=True)
print("> Valores nulos imputados correctamente (KNN para numéricos, Moda para categóricos).")

# 3. Tratamiento de Outliers solo para variables numéricas clasificadas con el criterio establecido
df_final, figs_outliers = detect_handle_outliers(df_clean_step2, method='iqr', return_plots=True)
print(f"> Outliers tratados con método IQR. Se generaron gráficos para {len(figs_outliers)} variables.")

# --- BLOQUE 3: ANÁLISIS FINAL (PROCESADO) ---
print("\n\n=== 3. ESTADO FINAL DEL DATASET ===")
print(f"Dimensiones Finales: {df_final.shape[0]} filas x {df_final.shape[1]} columnas")

# Reporte de calidad FINAL
reporte_final = check_data_completeness_JosueJimenezApodaca(df_final)

# --- VISUALIZACIÓN DE CLASIFICACIÓN DE VARIABLES ---
print("\n--- Clasificación Automática de Variables después de preprocesamiento [cite: 21] ---")

vars_continuas = reporte_final[reporte_final['Categoría Auto'] == 'Continua'].index.tolist()
vars_discretas = reporte_final[reporte_final['Categoría Auto'] == 'Discreta'].index.tolist()

print(f"👉 Total Variables Continuas: {len(vars_continuas)}")
print(f"👉 Total Variables Discretas: {len(vars_discretas)}")

# --- NUEVO: ANÁLISIS DE ESTABILIDAD (OUTLIERS) ---
print("\n--- Detalle de Estabilidad (Outliers) ---")

# Lógica: Cruzamos la lista oficial de continuas con las llaves del diccionario de figuras del Bloque 2
vars_con_outliers = sorted([v for v in vars_continuas if v in figs_outliers])
vars_sin_outliers = sorted([v for v in vars_continuas if v not in figs_outliers])

print(f"⚠ Variables Recortadas (Outliers detectados: {len(vars_con_outliers)}):")
print(vars_con_outliers)

print(f"\n✅ Variables Estables (Sin outliers detectados: {len(vars_sin_outliers)}):")
print(vars_sin_outliers)

# --- ESTADÍSTICOS DESCRIPTIVOS ---

# 1. Numéricas Continuas
print("\n--- Estadísticos Descriptivos (Variables Continuas) ---")
if vars_continuas:
    display(df_final[vars_continuas].describe().T)
else:
    print("No hay variables continuas.")

# 2. Discretas y Categóricas
print("\n--- Resumen de Variables Discretas/Categóricas (Frecuencias) ---")
if vars_discretas:
    display(df_final[vars_discretas].astype(object).describe().T)
else:
    print("No hay variables discretas.")

=== 1. ESTADO ORIGINAL DEL DATASET ===
Dimensiones Iniciales: 2129 filas x 40 columnas

```

	FileName	Date	SegFile	b	e	LBE	LB	AC	FM	UC	...	C	D	E	AD	DE	LD	FS	SUSP	CLASS	NSP
0	Variab10.txt	12/1/1996	CTG0001.txt	240.0	357.0	120.0	120.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	9.0	2.0
1	Fmcs_1.txt	5/3/1996	CTG0002.txt	5.0	632.0	132.0	132.0	4.0	0.0	4.0	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	6.0	1.0
2	Fmcs_1.txt	5/3/1996	CTG0003.txt	177.0	779.0	133.0	133.0	2.0	0.0	5.0	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	6.0	1.0
3	Fmcs_1.txt	5/3/1996	CTG0004.txt	411.0	1192.0	134.0	134.0	2.0	0.0	6.0	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	6.0	1.0
4	Fmcs_1.txt	5/3/1996	CTG0005.txt	533.0	1147.0	132.0	132.0	4.0	0.0	5.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	1.0

5 rows × 40 columns

--- Reporte de Calidad (Antes de Limpieza) ---

Columna	Nulos	% Completitud	Tipo Dato	Estadísticas	Categoría Auto
FileName	3	99.86	object	N/A	Discreta
Date	3	99.86	object	N/A	Discreta
SegFile	3	99.86	object	N/A	Discreta
b	3	99.86	float64	Min:0.00, Max:3296.00, Std:894.08	Continua
e	3	99.86	float64	Min:287.00, Max:3599.00, Std:930.92	Continua
LBE	3	99.86	float64	Min:106.00, Max:160.00, Std:9.84	Continua
LB	3	99.86	float64	Min:106.00, Max:160.00, Std:9.84	Continua
AC	3	99.86	float64	Min:0.00, Max:26.00, Std:3.56	Continua
FM	2	99.91	float64	Min:0.00, Max:564.00, Std:39.03	Continua
UC	2	99.91	float64	Min:0.00, Max:23.00, Std:2.88	Continua
ASTV	2	99.91	float64	Min:12.00, Max:87.00, Std:17.21	Continua
MSTV	2	99.91	float64	Min:0.20, Max:7.00, Std:0.89	Continua
ALTV	2	99.91	float64	Min:0.00, Max:91.00, Std:18.48	Continua
MLTV	2	99.91	float64	Min:0.00, Max:50.70, Std:5.70	Continua
DL	1	99.95	float64	Min:0.00, Max:16.00, Std:2.52	Continua
DS	1	99.95	float64	Min:0.00, Max:1.00, Std:0.06	Discreta
DP	1	99.95	float64	Min:0.00, Max:4.00, Std:0.47	Discreta
DR	1	99.95	float64	Min:0.00, Max:0.00, Std:0.00	Discreta
Width	3	99.86	float64	Min:3.00, Max:180.00, Std:38.96	Continua
Min	3	99.86	float64	Min:50.00, Max:159.00, Std:29.56	Continua
Max	3	99.86	float64	Min:122.00, Max:238.00, Std:17.94	Continua
Nmax	3	99.86	float64	Min:0.00, Max:18.00, Std:2.95	Continua
Nzeros	3	99.86	float64	Min:0.00, Max:10.00, Std:0.71	Discreta
Mode	3	99.86	float64	Min:60.00, Max:187.00, Std:16.38	Continua
Mean	3	99.86	float64	Min:73.00, Max:182.00, Std:15.59	Continua
Median	3	99.86	float64	Min:77.00, Max:186.00, Std:14.47	Continua
Variance	3	99.86	float64	Min:0.00, Max:269.00, Std:28.98	Continua
Tendency	3	99.86	float64	Min:-1.00, Max:1.00, Std:0.61	Discreta
A	3	99.86	float64	Min:0.00, Max:1.00, Std:0.38	Discreta
B	3	99.86	float64	Min:0.00, Max:1.00, Std:0.45	Discreta
C	3	99.86	float64	Min:0.00, Max:1.00, Std:0.16	Discreta
D	3	99.86	float64	Min:0.00, Max:1.00, Std:0.19	Discreta
E	3	99.86	float64	Min:0.00, Max:1.00, Std:0.18	Discreta
AD	3	99.86	float64	Min:0.00, Max:1.00, Std:0.36	Discreta
DE	3	99.86	float64	Min:0.00, Max:1.00, Std:0.32	Discreta
LD	3	99.86	float64	Min:0.00, Max:1.00, Std:0.22	Discreta
FS	3	99.86	float64	Min:0.00, Max:1.00, Std:0.18	Discreta
SUSP	3	99.86	float64	Min:0.00, Max:1.00, Std:0.29	Discreta
CLASS	3	99.86	float64	Min:1.00, Max:10.00, Std:3.03	Discreta
NSP	3	99.86	float64	Min:1.00, Max:3.00, Std:0.61	Discreta

--- Estadísticos Descriptivos (Originales) ---

	b	e	LBE	LB	AC	FM	UC	ASTV	MSTV	ALTV	...	C	D	E	AD	DE	LI
count	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	2127.000000	2127.000000	2127.000000	2127.000000	2127.000000	...	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000
mean	878.439793	1702.877234	133.303857	133.303857	2.722484	7.503056	3.669017	47.008933	1.335449	9.884814	...	0.024929	0.038100	0.033866	0.156162	0.118532	0.050321
std	894.084748	930.919143	9.840844	9.840844	3.560850	39.030452	2.877148	17.210648	0.891543	18.476534	...	0.155947	0.191482	0.180928	0.363094	0.323314	0.218671
min	0.000000	287.000000	106.000000	106.000000	0.000000	0.000000	0.000000	12.000000	0.200000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	55.000000	1009.000000	126.000000	126.000000	0.000000	0.000000	1.000000	32.000000	0.700000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	538.000000	1241.000000	133.000000	133.000000	1.000000	0.000000	3.000000	49.000000	1.200000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	1521.000000	2434.750000	140.000000	140.000000	4.000000	2.000000	5.000000	61.000000	1.700000	11.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	3296.000000	3599.000000	160.000000	160.000000	26.000000	564.000000	23.000000	87.000000	7.000000	91.000000	...	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 37 columns

```

=====
=== APLICANDO PIPELINE DE LIMPIEZA ===
=====
-> Columnas eliminadas por >20% nulos: Ninguna
-> Valores nulos imputados correctamente (KNN para numéricos, Moda para categóricos).
-> Outliers tratados con método IQR. Se generaron gráficos para 13 variables.

=== 3. ESTADO FINAL DEL DATASET ===
Dimensiones Finales: 2129 filas x 40 columnas

--- Clasificación Automática de Variables después de preprocesamiento [cite: 21] ---
✖ Total Variables Continuas: 18
✖ Total Variables Discretas: 22

--- Detalle de Estabilidad (Outliers) ---
⚠ Variables Recortadas (Outliers detectados: 11):
['AC', 'ALTV', 'MLTV', 'MSTV', 'Max', 'Mean', 'Median', 'Mode', 'Nmax', 'UC', 'Variance']

✅ Variables Estables (Sin outliers detectados: 7):
['ASTV', 'LB', 'LBE', 'Min', 'Width', 'b', 'e']

--- Estadísticos Descriptivos (Variables Continuas) ---

```

	count	mean	std	min	25%	50%	75%	max
b	2129.0	877.900911	893.675597	0.0	55.0	538.0	1518.0	3296.0
e	2129.0	1702.502432	930.348716	287.0	1009.0	1241.0	2434.0	3599.0
LBE	2129.0	133.304229	9.834165	106.0	126.0	133.0	140.0	160.0
LB	2129.0	133.304229	9.834165	106.0	126.0	133.0	140.0	160.0
AC	2129.0	2.580518	3.122548	0.0	0.0	1.0	4.0	10.0
UC	2129.0	3.636575	2.755725	0.0	1.0	3.0	5.0	11.0
ASTV	2129.0	47.007519	17.202682	12.0	32.0	49.0	61.0	87.0
MSTV	2129.0	1.300862	0.771606	0.2	0.7	1.2	1.7	3.2
ALTV	2129.0	6.661853	10.290353	0.0	0.0	0.0	11.0	27.5
MLTV	2129.0	8.009153	5.022647	0.0	4.6	7.4	10.8	20.1
Width	2129.0	70.450468	38.943890	3.0	37.0	68.0	100.0	180.0
Min	2129.0	93.576035	29.551612	50.0	67.0	93.0	120.0	159.0
Max	2129.0	163.839561	17.347382	122.0	152.0	162.0	174.0	207.0
Nmax	2129.0	4.051793	2.893142	0.0	2.0	3.0	6.0	12.0
Mode	2129.0	137.908197	14.666525	100.5	129.0	139.0	148.0	176.5
Mean	2129.0	134.783096	15.021656	95.0	125.0	136.0	145.0	175.0
Median	2129.0	138.186139	14.091648	100.5	129.0	139.0	148.0	176.5
Variance	2129.0	15.597937	18.190036	0.0	2.0	7.0	24.0	57.0

--- Resumen de Variables Discretas/Categóricas (Frecuencias) ---

	count	unique	top	freq
FileName	2129	352	S8001034.dsp	37
Date	2129	48	2/22/1995	243
SegFile	2129	2126	CTG0001.txt	4
FM	2129.0	6.0	0.0	1311.0
DL	2129.0	10.0	0.0	1232.0
DS	2129.0	2.0	0.0	2121.0
DP	2129.0	5.0	0.0	1950.0
DR	2129.0	1.0	0.0	2129.0
Nzeros	2129.0	9.0	0.0	1627.0
Tendency	2129.0	3.0	0.0	1118.0
A	2129.0	2.0	0.0	1745.0
B	2129.0	2.0	0.0	1550.0
C	2129.0	2.0	0.0	2076.0
D	2129.0	2.0	0.0	2048.0
E	2129.0	2.0	0.0	2057.0
AD	2129.0	2.0	0.0	1797.0
DE	2129.0	2.0	0.0	1877.0
LD	2129.0	2.0	0.0	2022.0
FS	2129.0	2.0	0.0	2060.0
SUSP	2129.0	2.0	0.0	1932.0
CLASS	2129.0	10.0	2.0	582.0
NSP	2129.0	3.0	1.0	1658.0

No se eliminó ninguna columna ya que su completitud de todas es mayor al 80%.

Se realiza la clasificación personalizada: ** Continuas (más de 10 valores únicos y tipo numérico)** Discretas (menos de 10 valores únicos)Bajo esta clasificacion se tienen 18 variables continuas y 21 discretas.

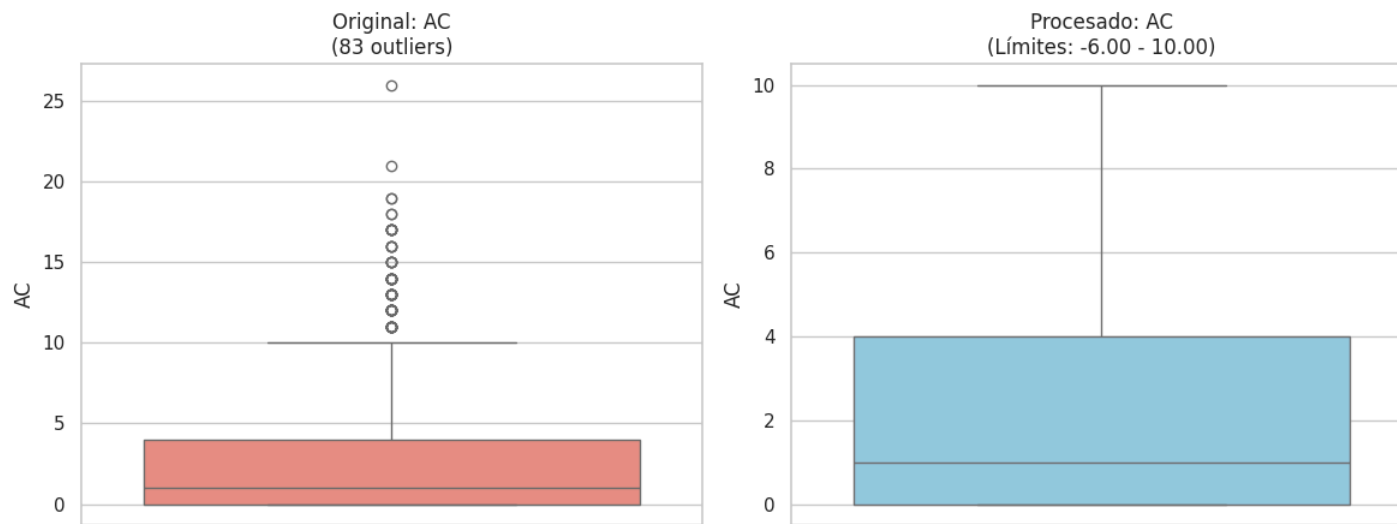
Se aplica KNN para imputar valores nulos de las variables continuas, la imputacion de las variables discretas se realizó con la moda. De las 18 variables continuas solo se encontraron 11 que contenian outliers, por lo que no se generaron graficas para 7 de ellas, se recomienda graficar por seperado si se requiere comprobar visualmente.

EJEMPLO DE USO DE GRÁFICAS INTERACTIVAS

Una vez que se ejecutó el preprocesamiento, realizamos análisis sobre los resultados

```
In [5]: if len(figs_outliers) > 0:
col_ejemplo = list(figs_outliers.keys())[0]
display(figs_outliers[col_ejemplo])
```

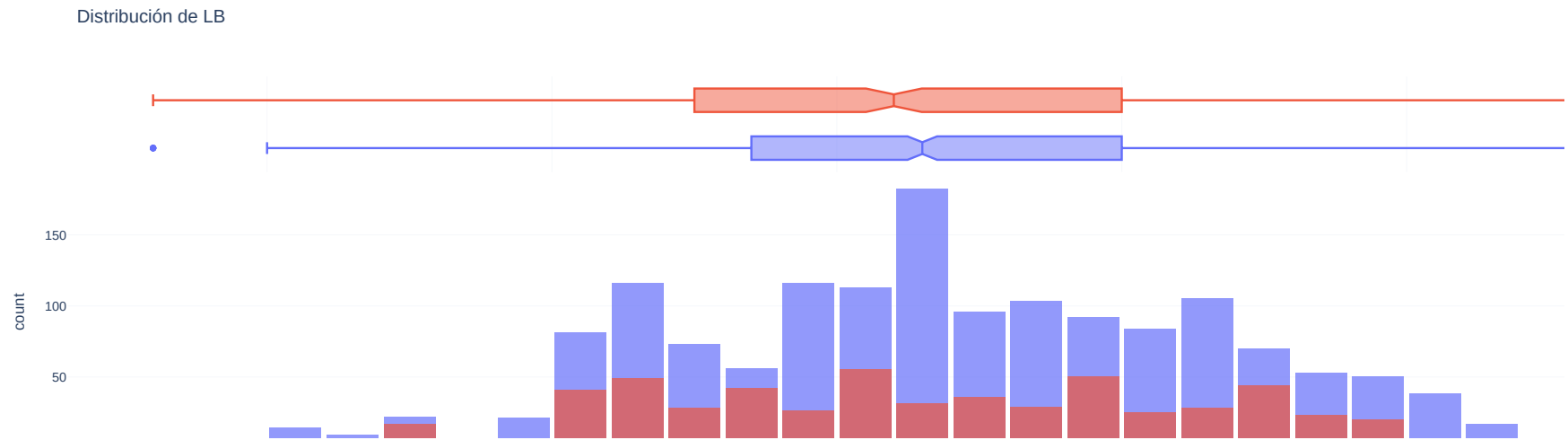
Tratamiento Outliers: AC



3.1 Distribución de Variables (Histogramas)

Observamos la distribución de la frecuencia cardiaca fetal (LB).

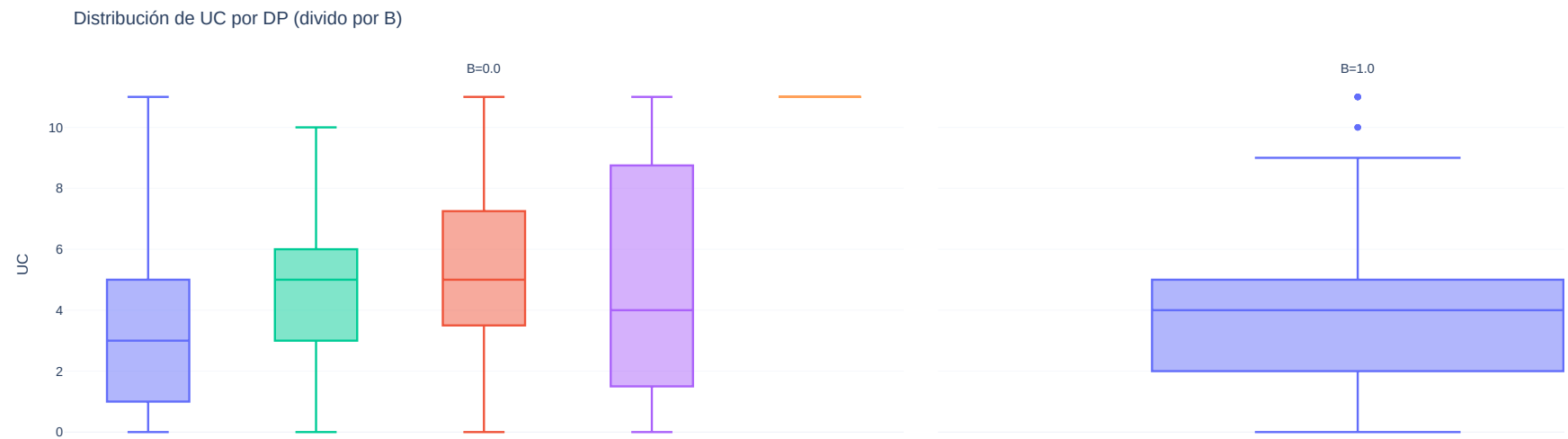
```
In [6]: # Visualizar distribución de Línea Base (LB) agrupada por Clase (NSP)
fig_hist = plot_histogram_interactivo(
    df_final,
    col='LB', # Variable numérica
    group_by='B' # Variable categórica (colores)
)
fig_hist.show()
```



3.2 Comparación por Clases (Boxplots)

Analizamos la variabilidad según el estado fetal (NSP).

```
In [7]: # Comparar Variabilidad a Corto Plazo (ASTV) según la clase (NSP)
fig_box = plot_boxplot(
    df_final,
    x='DP',      # Eje X (Categoría)
    y='UC',      # Eje Y (Numérica)
    facet_col='B' # Separa en columnas distintas (Faceting)
)
fig_box.show()
```

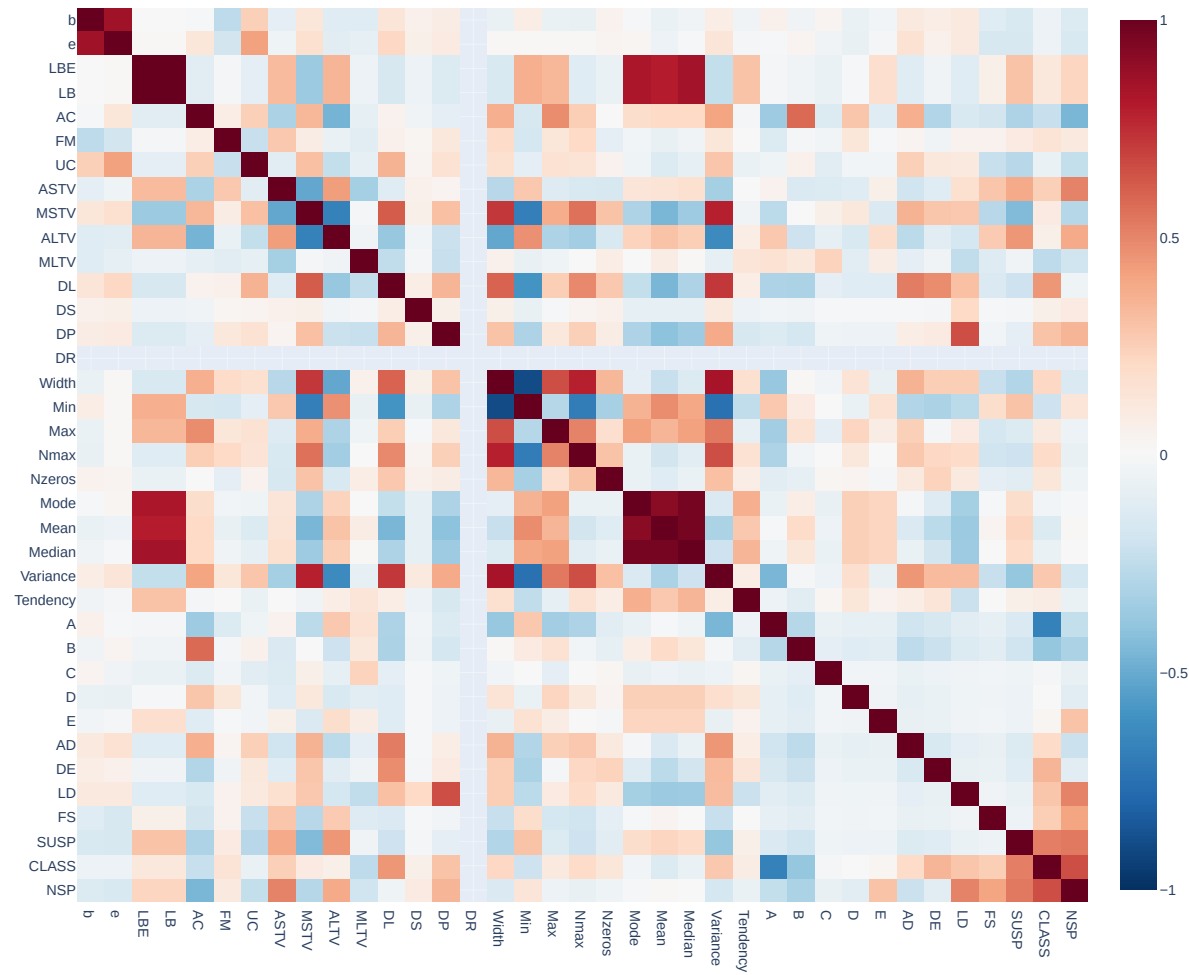


3.3 Correlaciones (Heatmap)

Buscamos variables redundantes.

```
In [8]: # Matriz de correlación de variables numéricas
fig_corr = plot_correlation_heatmap(
    df_final,
    method='spearman'
)
fig_corr.show()
```

Matriz de Correlación Interactiva (Spearman)



3.4 Gráfico de Barras (Frecuencias)

Para ver el desbalance de clases (cuántos sanos vs. patológicos hay).

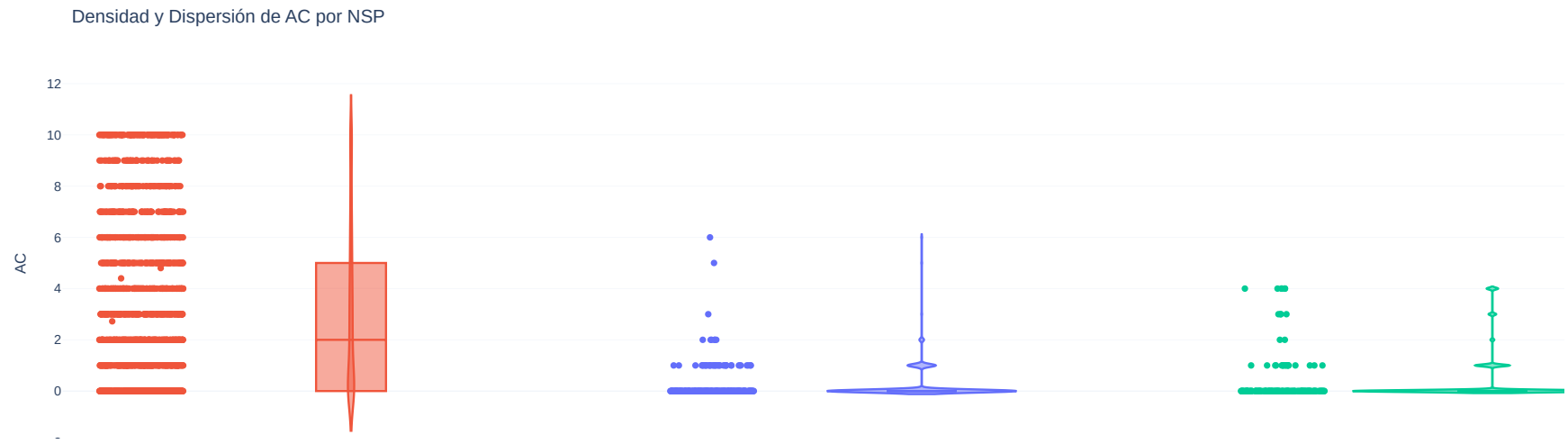
```
In [9]: # Conteo de casos por Categoría de Estado Fetal (NSP)
fig_bar = plot_bar(
    df_final,
    col='NSP',
    horizontal=True
)
fig_bar.show()
```



3.5 Violin Plot (Densidad + Puntos)

Muestra la "forma" de los datos.

```
In [10]: # Densidad de Aceleraciones (AC) por Clase
fig_violin = plot_violin(
    df_final,
    x='NSP',
    y='AC'
)
fig_violin.show()
```



4. Validacion de funciones

En esta seccion se testean las funcionalidades de las funciones usadas en el proyecto

```
In [11]: ### Ejecutar las pruebas (El "Resultado")
print("\n" + "="*40)
print("📁 EJECUCIÓN DE PYTEST (EVIDENCIA)")
print("="*40)

%cd ..
!pytest tests/ -v
%cd notebooks

=====
📁 EJECUCIÓN DE PYTEST (EVIDENCIA)
=====
/home/josuej/Documentos/Practica3/ctg_viz
[]9;4;3;[]===== test session starts =====
platform linux -- Python 3.11.11, pytest-9.0.1, pluggy-1.6.0 -- /home/josuej/.pyenv/versions/3.11.11/bin/python3.11
cachedir: .pytest_cache
rootdir: /home/josuej/Documentos/Practica3/ctg_viz
collected 4 items

tests/test_preprocessing.py::test_remove_null_columns []9;4;1;0[]PASSED [ 25%]
tests/test_preprocessing.py::test_impute_missing_values_mode []9;4;1;25[]PASSED [ 50%]
tests/test_preprocessing.py::test_detect_handle_outliers_iqr []9;4;1;50[]PASSED [ 75%]
tests/test_preprocessing.py::test_check_data_completeness_structure []9;4;1;75[]PASSED [100%] []9;4;0;[]

===== 4 passed in 0.83s =====
/home/josuej/Documentos/Practica3/ctg_viz/notebooks
```

5. Conclusiones y Recomendaciones

1. **Calidad de Datos:** Se eliminaron 0 columnas por falta de datos. La imputación KNN preservó la varianza en las variables clave.
2. **Outliers:** Se detectaron y toparon valores extremos en 11 variables usando el metodo CLIP (aproximacion al limite mas cercano)

3. Patrones:

Para un analisis mas detallado se brindan los nombres y/o descripcion de las variables en idioma original, no se traducen para evitar problemas de interpretaci3n de lenguaje:

- FileName: of CTG examination
- Date: of the examination
- b: start instant
- e: end instant
- LBE: baseline value (medical expert)
- LB: baseline value (SisPorto)
- AC: accelerations (SisPorto)
- FM: foetal movement (SisPorto)
- UC: uterine contractions (SisPorto)
- ASTV: percentage of time with abnormal short term variability (SisPorto)
- mSTV: mean value of short term variability (SisPorto)
- ALTV: percentage of time with abnormal long term variability (SisPorto)
- mLTV: mean value of long term variability (SisPorto)
- DL: light decelerations
- DS: severe decelerations
- DP: prolonged decelerations
- DR: repetitive decelerations
- Width: histogram width
- Min: low freq. of the histogram
- Max: high freq. of the histogram
- Nmax: number of histogram peaks
- Nzeros: number of histogram zeros
- Mode: histogram mode
- Mean: histogram mean
- Median: histogram median
- Variance: histogram variance
- Tendency: histogram tendency: -1=left assymetric; 0=symmetric; 1=right assymetric
- A: calm sleep
- B: REM sleep
- C: calm vigilance
- D: active vigilance
- SH: shift pattern (A or Susp with shifts)
- AD: accelerative/decelerative pattern (stress situation)
- DE: decelerative pattern (vagal stimulation)
- LD: largely decelerative pattern
- FS: flat-sinusoidal pattern (pathological state)
- SUSP: suspect pattern
- CLASS: Class code (1 to 10) for classes A to SUSP
- NSP: Normal=1; Suspect=2; Pathologic=3

Se brinda el ejemplo de la variable SNP que con la gr1fica numero 4 se puede observar que la mayoria de las personas analizadas caen dentro del rango "normal" lo que indica que la mayoria de los pacientes se encuentran en un buen estado de salud.

Una conclusi3n m1s desarrollada se puede realizar con el gr1fico 5:

1. El gr1fico muestra una diferencia dr1stica en la distribuci3n de las Aceleraciones (AC) entre el estado Normal (1) y los estados Sospechoso (2) y Patol3gico (3). Clase 1 (Normal): Es la 1nica categoria que presenta una distribuci3n extendida verticalmente, alcanzando valores de AC superiores a 10. Aunque tiene densidad en 0 (parte ancha inferior), la presencia de una "cola" larga hacia arriba indica que las aceleraciones son frecuentes y de mayor magnitud en fetos sanos. Clases 2 y 3 (Sospechoso y Patol3gico): Las gr1ficas de viol3n est1n "aplastadas" contra el eje 0. La densidad es casi absoluta en la base, lo que significa que la ausencia de aceleraciones es una caracteristica dominante en los estados no saludables.

2. Predicción segura de Normalidad se da cuando el valor de AC es alto: Si el modelo encuentra un valor de AC alto (por ejemplo, > 5), es casi seguro que el caso pertenece a la clase 1 (Normal), ya que las clases 2 y 3 prácticamente no existen en ese rango.

La variable AC (Aceleraciones) actúa como un fuerte predictor negativo de patología. Mientras que su ausencia es condición necesaria pero no suficiente para clasificar un caso como Patológico, su presencia en magnitudes altas descarta casi totalmente los estados Sospechoso y Patológico.

Para realizar más análisis como estos se presenta un visualizador interactivo desarrollado en Streamlit, mismo que se encuentra en la carpeta raíz de este repositorio.