Practica 3 | Diplomado en ciencia de datos

Luis Manuel Álamo Díaz

In [19]:
```
import pandas as pd
import numpy as np
from sklearn.impute import KNNImputer
import seaborn as sns
import matplotlib.pyplot as plt
```

In [20]:
```
#df = pd.read_csv(r'E:\Users\1070911\Luis DCD\CTG.csv')
df = pd.read_csv('CTG.csv')
#df.head()
df.info()
df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2129 entries, 0 to 2128
Data columns (total 40 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   FileName  2126 non-null   object 
 1   Date      2126 non-null   object 
 2   SegFile   2126 non-null   object 
 3   b         2126 non-null   float64
 4   e         2126 non-null   float64
 5   LBE       2126 non-null   float64
 6   LB        2126 non-null   float64
 7   AC        2126 non-null   float64
 8   FM        2127 non-null   float64
 9   UC        2127 non-null   float64
 10  ASTV      2127 non-null   float64
 11  MSTV      2127 non-null   float64
 12  ALTV      2127 non-null   float64
 13  MLTV      2127 non-null   float64
 14  DL        2128 non-null   float64
 15  DS        2128 non-null   float64
 16  DP        2128 non-null   float64
 17  DR        2128 non-null   float64
 18  Width     2126 non-null   float64
 19  Min       2126 non-null   float64
 20  Max       2126 non-null   float64
 21  Nmax      2126 non-null   float64
 22  Nzeros    2126 non-null   float64
 23  Mode      2126 non-null   float64
 24  Mean      2126 non-null   float64
 25  Median    2126 non-null   float64
 26  Variance  2126 non-null   float64
 27  Tendency  2126 non-null   float64
 28  A         2126 non-null   float64
 29  B         2126 non-null   float64
 30  C         2126 non-null   float64
 31  D         2126 non-null   float64
 32  E         2126 non-null   float64
 33  AD        2126 non-null   float64
 34  DE        2126 non-null   float64
 35  LD        2126 non-null   float64
 36  FS        2126 non-null   float64
 37  SUSP      2126 non-null   float64
 38  CLASS     2126 non-null   float64
 39  NSP       2126 non-null   float64
dtypes: float64(37), object(3)
memory usage: 665.4+ KB
```

Out[20]:

	b	e	LBE	LB	AC	
count	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	212
mean	878.439793	1702.877234	133.303857	133.303857	2.722484	
std	894.084748	930.919143	9.840844	9.840844	3.560850	3
min	0.000000	287.000000	106.000000	106.000000	0.000000	
25%	55.000000	1009.000000	126.000000	126.000000	0.000000	
50%	538.000000	1241.000000	133.000000	133.000000	1.000000	
75%	1521.000000	2434.750000	140.000000	140.000000	4.000000	
max	3296.000000	3599.000000	160.000000	160.000000	26.000000	56

8 rows × 37 columns

1. Preprocesamiento

Eliminar columnas con más del 20% de valores nulos

In [21]:
```
#conteo de valores nulos

df.isna().mean()*100 #porcentaje
#df.isna().sum()#conteo de valores nulos
```

Out[21]:
```
FileName    0.140911
Date        0.140911
SegFile     0.140911
b           0.140911
e           0.140911
LBE         0.140911
LB          0.140911
AC          0.140911
FM          0.093941
UC          0.093941
ASTV        0.093941
MSTV        0.093941
ALTV        0.093941
MLTV        0.093941
DL          0.046970
DS          0.046970
DP          0.046970
DR          0.046970
Width       0.140911
Min         0.140911
Max         0.140911
Nmax        0.140911
Nzeros      0.140911
Mode        0.140911
Mean        0.140911
Median      0.140911
Variance    0.140911
Tendency    0.140911
A           0.140911
B           0.140911
C           0.140911
D           0.140911
E           0.140911
AD          0.140911
DE          0.140911
LD          0.140911
FS          0.140911
SUSP        0.140911
CLASS       0.140911
NSP         0.140911
dtype: float64
```

Ninguna columan supera el 1% de valores nulos, por lo que conservamos todas las columnas.

Imputar valores faltantes restantes con métodos adecuados:

In [22]:
```
#imputación de valores

unique_counts = df.nunique()
```

```
df_imp = df.copy()

# Clasificamos (columanas numericas con 10 o. más valores unicos son n
discretas = [col for col in df_imp.columns if unique_counts[col] < 10]
continuas = [col for col in df_imp.columns if unique_counts[col] >= 10

cols_num = df_imp.select_dtypes(include=["number"]).columns
#cols_cat = df_imp.select_dtypes(exclude=["number"]).columns

#discretas = [col for col in discretas if col in cols_cat]
continuas = [col for col in continuas if col in cols_num]

for col in continuas:
    valor = df_imp[col].mean()
    df_imp[col] = df_imp[col].fillna(valor)

print(df_imp.isna().mean()*100)
print('Se imputaron todas las columnas continuas')
```

```
FileName    0.140911
Date        0.140911
SegFile     0.140911
b           0.000000
e           0.000000
LBE         0.000000
LB          0.000000
AC          0.000000
FM          0.000000
UC          0.000000
ASTV        0.000000
MSTV        0.000000
ALTV        0.000000
MLTV        0.000000
DL          0.000000
DS          0.046970
DP          0.046970
DR          0.046970
Width       0.000000
Min         0.000000
Max         0.000000
Nmax        0.000000
Nzeros      0.140911
Mode        0.000000
Mean        0.000000
Median      0.000000
Variance    0.000000
Tendency    0.140911
A           0.140911
B           0.140911
C           0.140911
D           0.140911
E           0.140911
AD          0.140911
DE          0.140911
LD          0.140911
FS          0.140911
SUSP        0.140911
CLASS       0.000000
NSP         0.140911
dtype: float64
Se imputaron todas las columnas continuas
```

In [23]:
```
# Categóricos
for col in discretas:
    if df_imp[col].isna().all():
        continue
    moda = df_imp[col].mode(dropna=True)
    if len(moda) > 0:
        valor = moda.iloc[0]
        df_imp[col] = df_imp[col].fillna(valor)

print(df_imp.isna().mean()*100)
```

```
print('\n Se imputaron todas las columnas discretas')
```

```
FileName    0.140911
Date        0.140911
SegFile     0.140911
b           0.000000
e           0.000000
LBE         0.000000
LB          0.000000
AC          0.000000
FM          0.000000
UC          0.000000
ASTV        0.000000
MSTV        0.000000
ALTV        0.000000
MLTV        0.000000
DL          0.000000
DS          0.000000
DP          0.000000
DR          0.000000
Width       0.000000
Min         0.000000
Max         0.000000
Nmax        0.000000
Nzeros      0.000000
Mode        0.000000
Mean        0.000000
Median      0.000000
Variance    0.000000
Tendency    0.000000
A           0.000000
B           0.000000
C           0.000000
D           0.000000
E           0.000000
AD          0.000000
DE          0.000000
LD          0.000000
FS          0.000000
SUSP        0.000000
CLASS       0.000000
NSP         0.000000
dtype: float64
```

```
 Se imputaron todas las columnas discretas
```

KNN Imputer (reto adicional) 5% adicional

In [24]:
```
df_imp_knn = df.copy()
df_imp_knn.isna().mean()
```

Out[24]:
```
FileName    0.001409
Date        0.001409
SegFile     0.001409
b           0.001409
e           0.001409
LBE         0.001409
LB          0.001409
AC          0.001409
FM          0.000939
UC          0.000939
ASTV        0.000939
MSTV        0.000939
ALTV        0.000939
MLTV        0.000939
DL          0.000470
DS          0.000470
DP          0.000470
DR          0.000470
Width       0.001409
Min         0.001409
Max         0.001409
Nmax        0.001409
Nzeros      0.001409
Mode        0.001409
Mean        0.001409
Median      0.001409
Variance    0.001409
Tendency    0.001409
A           0.001409
B           0.001409
C           0.001409
D           0.001409
E           0.001409
AD          0.001409
DE          0.001409
LD          0.001409
FS          0.001409
SUSP        0.001409
CLASS       0.001409
NSP         0.001409
dtype: float64
```

In [25]:
```
cols_numericas = None
if cols_numericas is None:
    cols_numericas = df_imp_knn.select_dtypes(include=["number"]).colu

imputer = KNNImputer(n_neighbors=5)
datos_num = df_imp_knn[cols_numericas]
datos_imputados = imputer.fit_transform(datos_num)
df_imp_knn[cols_numericas] = datos_imputados
print('Se imputo usando el metodo KNN')
```

```
Se imputo usando el metodo KNN
```

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

Out[26]:
```
FileName    3
Date        3
SegFile     3
b           0
e           0
LBE         0
LB          0
AC          0
FM          0
UC          0
ASTV        0
MSTV        0
ALTV        0
MLTV        0
DL          0
DS          0
DP          0
DR          0
Width       0
Min         0
Max         0
Nmax        0
Nzeros      0
Mode        0
Mean        0
Median      0
Variance    0
Tendency    0
A           0
B           0
C           0
D           0
E           0
AD          0
DE          0
LD          0
FS          0
SUSP        0
CLASS       0
NSP         0
dtype: int64
```

Detectar y tratar valores atípicos (outliers) con IQR o z-score

In [27]:
```
#OUTLIERS

df_proc = df_imp.copy()
columnas = None
factor = 1.5
```

```
print(f"Tamaño original: {len(df_proc)}")

if columnas is None:
    columnas = df_proc.select_dtypes(include=["number"]).columns.tolis

filas_a_eliminar = np.zeros(len(df_proc), dtype=bool)

for col in columnas:
    serie = df_proc[col].dropna()
    q1 = serie.quantile(0.25)
    q3 = serie.quantile(0.75)
    iqr = q3 - q1
    lim_inf = q1 - factor * iqr
    lim_sup = q3 + factor * iqr

    mask_out = (df_proc[col] < lim_inf) | (df_proc[col] > lim_sup)
    filas_a_eliminar = filas_a_eliminar | mask_out
df_proc_iqr = df_proc.loc[~filas_a_eliminar].reset_index(drop=True)
print(f"Outliers eliminados: {len(df_proc) - len(df_proc_iqr)}")

print(f"Tamaño del df despues de sacar outliers: {len(df_proc_iqr)}")
```

```
Tamaño original: 2129
Outliers eliminados: 1774
Tamaño del df despues de sacar outliers: 355
Outliers eliminados: 1774
Tamaño del df despues de sacar outliers: 355
```

In [28]:
```
#Z-score
umbral = 3
# Seleccionamos solo columnas numéricas
cols_num = df_proc.select_dtypes(include=["number"])

# Calculamos Z-score
z_scores = (cols_num - cols_num.mean()) / cols_num.std()

# Creamos máscara de outliers (True = outlier)
mask_outliers = np.abs(z_scores) > umbral

# Filtramos filas que contengan algún outlier
outliers_df = df_proc[mask_outliers.any(axis=1)]
df_sin_outliers = df_proc[~mask_outliers.any(axis=1)].reset_index(drop

print(f"Outliers eliminados: {len(df_imp) - len(df_sin_outliers)}")
len(df_sin_outliers)
```

```
Outliers eliminados: 743
```

Out[28]: 1386

2. Análisis de Datos

Crear una función general check_data_completeness_nomnbrecompleto(df) que retorne:

Conteo de nulos

Porcentaje de completitud

Tipo de dato

Estadísticos de dispersión

Clasificar automáticamente columnas en:

Continuas (más de 10 valores únicos y tipo numérico)

Discretas (menos de 10 valores únicos)

In [29]:
```python
#creación de funcion
def check_data_completeness_luis_manuel_alamo_diaz(df: pd.DataFrame) -

    filas_resumen = []

    unique_counts = df.nunique(dropna=True)
    cols_num = df.select_dtypes(include=["number"]).columns

    for col in df.columns:
        serie = df[col]
        n_nulos = serie.isna().sum()
        prop_nulos = n_nulos / len(serie)
        porcentaje_completitud = (1 - prop_nulos) * 100
        tipo_dato = serie.dtype
        n_unicos = unique_counts[col]


        if col in cols_num:
            if n_unicos < 10:
                tipo_variable = "discreta"
            else:
                tipo_variable = "continua"
        else:
            tipo_variable = "no_numerica"# No numéricas (texto,fechas,


        fila = {
            "columna": col,
            "n_nulos": n_nulos,
            "porcentaje_completitud": porcentaje_completitud,
            "tipo_dato": str(tipo_dato),
            "n_unicos": n_unicos,
            "tipo_variable": tipo_variable,
        }
```

```
        if pd.api.types.is_numeric_dtype(serie):
            fila.update(
                {
                    "min": serie.min(),
                    "max": serie.max(),
                    "media": serie.mean(),
                    "std": serie.std(),
                    "var": serie.var(),
                }
            )
        else:
            fila.update(
                {
                    "min": None,
                    "max": None,
                    "media": None,
                    "std": None,
                    "var": None,
                }
            )
        filas_resumen.append(fila)

    resumen = pd.DataFrame(filas_resumen)
    return resumen.set_index("columna")

check_data_completeness_luis_manuel_alamo_diaz(df)
```

Out[29]:

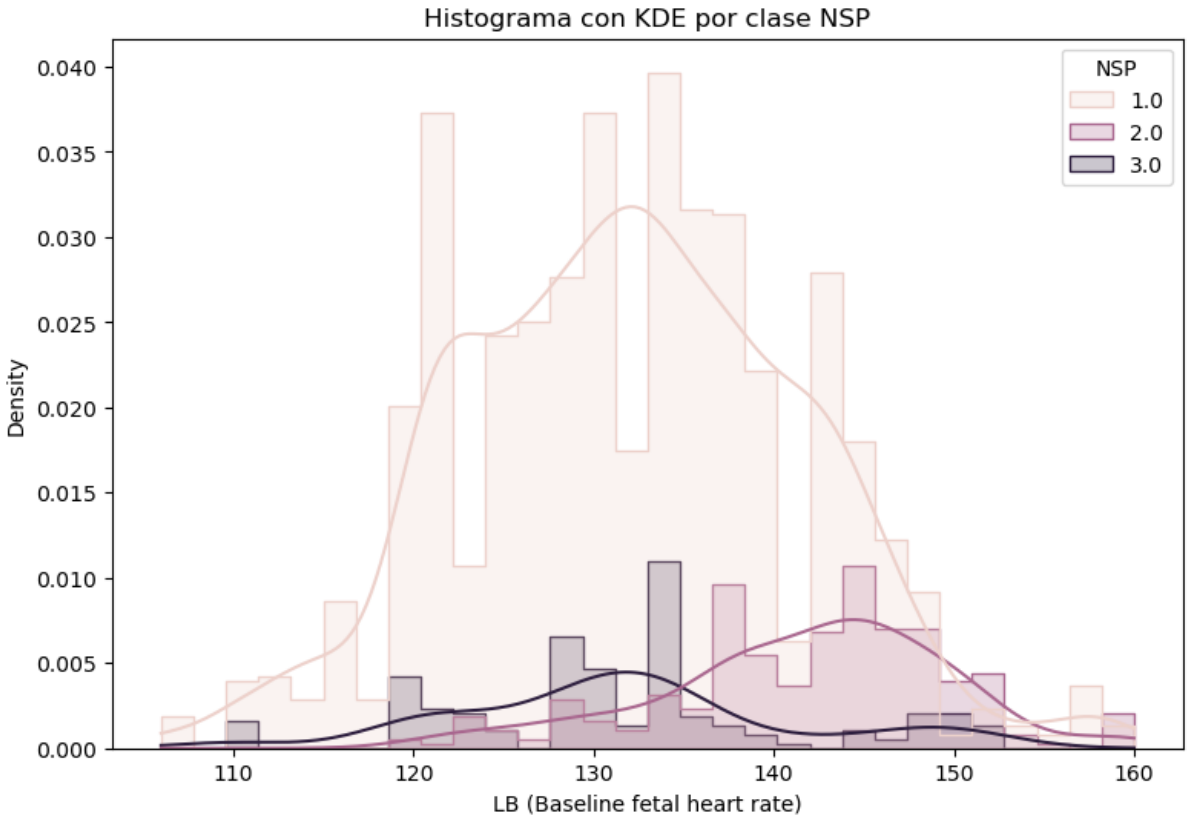
columna	n_nulos	porcentaje_completitud	tipo_dato	n_unicos	tipo_variable
FileName	3	99.859089	object	352	no_numerica
Date	3	99.859089	object	48	no_numerica
SegFile	3	99.859089	object	2126	no_numerica
b	3	99.859089	float64	979	continua
e	3	99.859089	float64	1064	continua
LBE	3	99.859089	float64	48	continua
LB	3	99.859089	float64	48	continua
AC	3	99.859089	float64	22	continua
FM	2	99.906059	float64	96	continua
UC	2	99.906059	float64	19	continua
ASTV	2	99.906059	float64	75	continua
MSTV	2	99.906059	float64	57	continua

ALTV	2	99.906059	float64	87	continua
MLTV	2	99.906059	float64	249	continua
DL	1	99.953030	float64	15	continua
DS	1	99.953030	float64	2	discreta
DP	1	99.953030	float64	5	discreta
DR	1	99.953030	float64	1	discreta
Width	3	99.859089	float64	154	continua
Min	3	99.859089	float64	109	continua
Max	3	99.859089	float64	86	continua
Nmax	3	99.859089	float64	18	continua
Nzeros	3	99.859089	float64	9	discreta
Mode	3	99.859089	float64	88	continua
Mean	3	99.859089	float64	103	continua
Median	3	99.859089	float64	95	continua
Variance	3	99.859089	float64	133	continua
Tendency	3	99.859089	float64	3	discreta
A	3	99.859089	float64	2	discreta
B	3	99.859089	float64	2	discreta
C	3	99.859089	float64	2	discreta
D	3	99.859089	float64	2	discreta
E	3	99.859089	float64	2	discreta
AD	3	99.859089	float64	2	discreta
DE	3	99.859089	float64	2	discreta
LD	3	99.859089	float64	2	discreta
FS	3	99.859089	float64	2	discreta
SUSP	3	99.859089	float64	2	discreta
CLASS	3	99.859089	float64	10	continua
NSP	3	99.859089	float64	3	discreta

In [30]:
```
columnas_numericas = df.select_dtypes(include="number").columns
```

3. Visualizaciones (eleva dificultad agregando interactividad, estadísticas o múltiples variables)

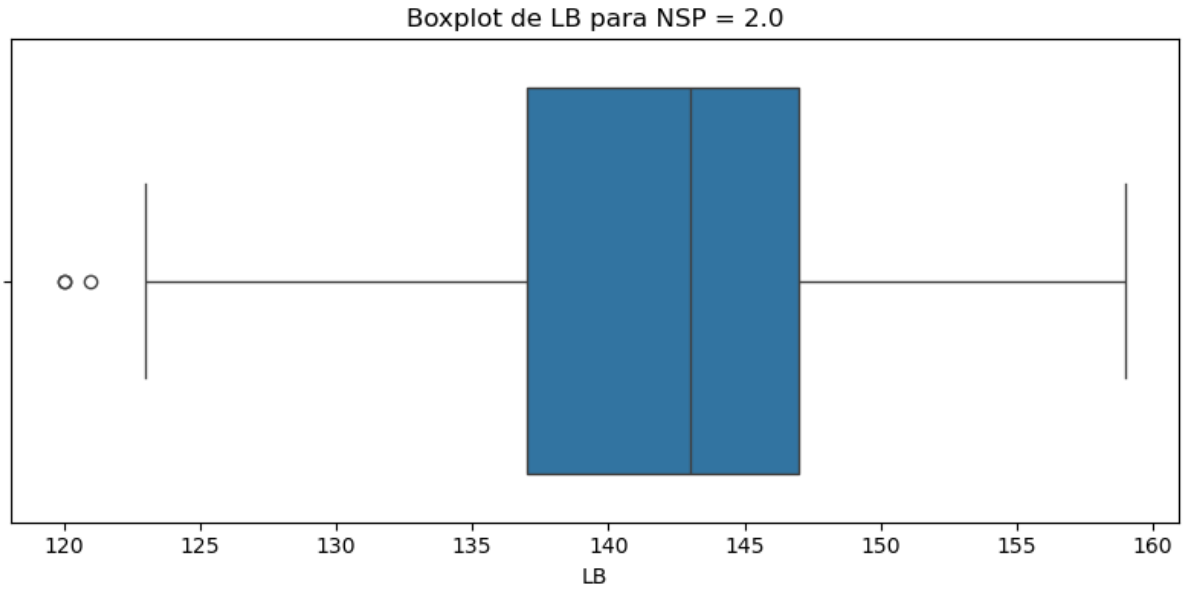
In [31]:
```
# HHistogramas Añadir línea de densidad + KDE + customizable por grupo

plt.figure(figsize=(9,6))
sns.histplot(
    data=df_proc,
    x="LB",             # frecuencia cardiaca fetal basal
    hue="NSP",          # clases Normal / Sospechoso / Patológico
    kde=True,
    stat="density",
    bins=30,
    element="step"
)
plt.title("Histograma con KDE por clase NSP")
plt.xlabel("LB (Baseline fetal heart rate)")
plt.show()
```

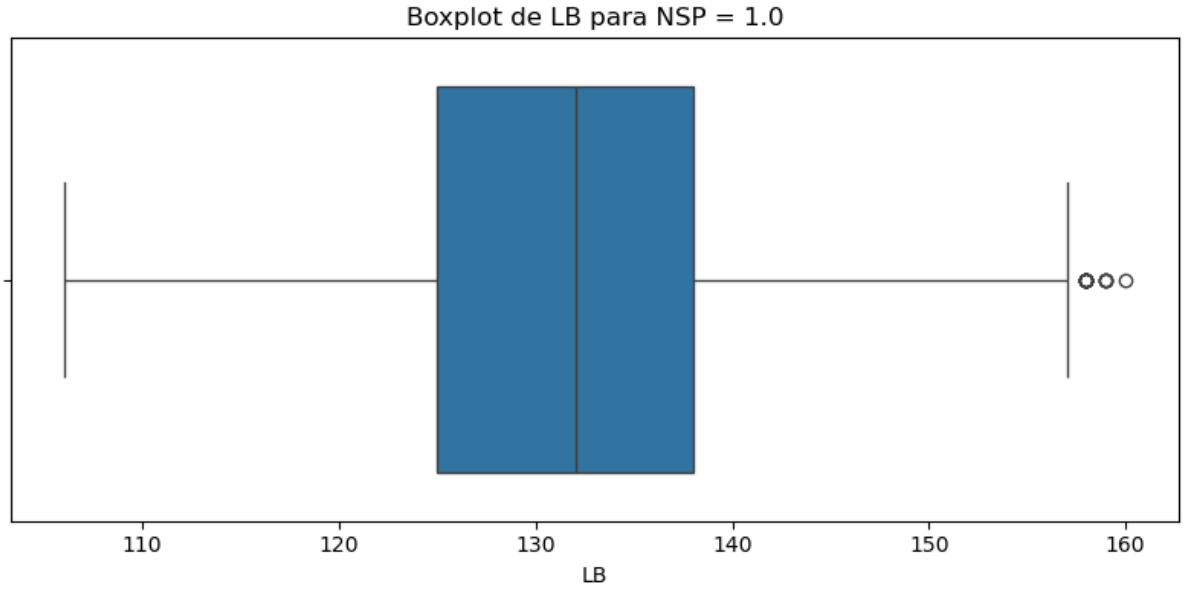


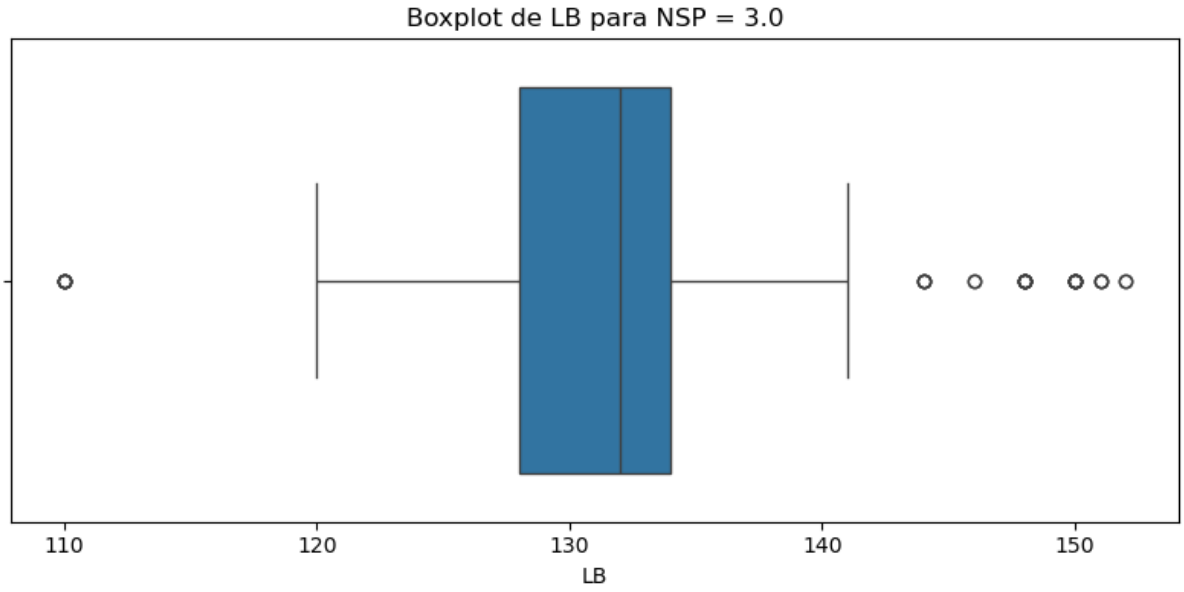
Use el Df con outliers pues si los quito no puedo comparar las distribuciones pues desapareecen los valores para otras categorias diferente a "normal" comparamos las distribuciones para cada cetegoria, use NSP, la cual considera normal, sospechos y patologico, la distribucion de sospechoso esta mas cargada a la derecha y tiene una cola izquierda mas cargada

In [32]:
```
# Boxplots Incluir subgráficos por clase objetivo
```

```
clases = df_imp["NSP"].unique()

fig, ejes = plt.subplots(len(clases), 1, figsize=(8, 12))

for i, clase in enumerate(clases):
    df_clase = df[df["NSP"] == clase]
    sns.boxplot(
        data=df_clase,
        x="LB",
        ax=ejes[i]
    )
    ejes[i].set_title(f"Boxplot de LB para NSP = {clase}")

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



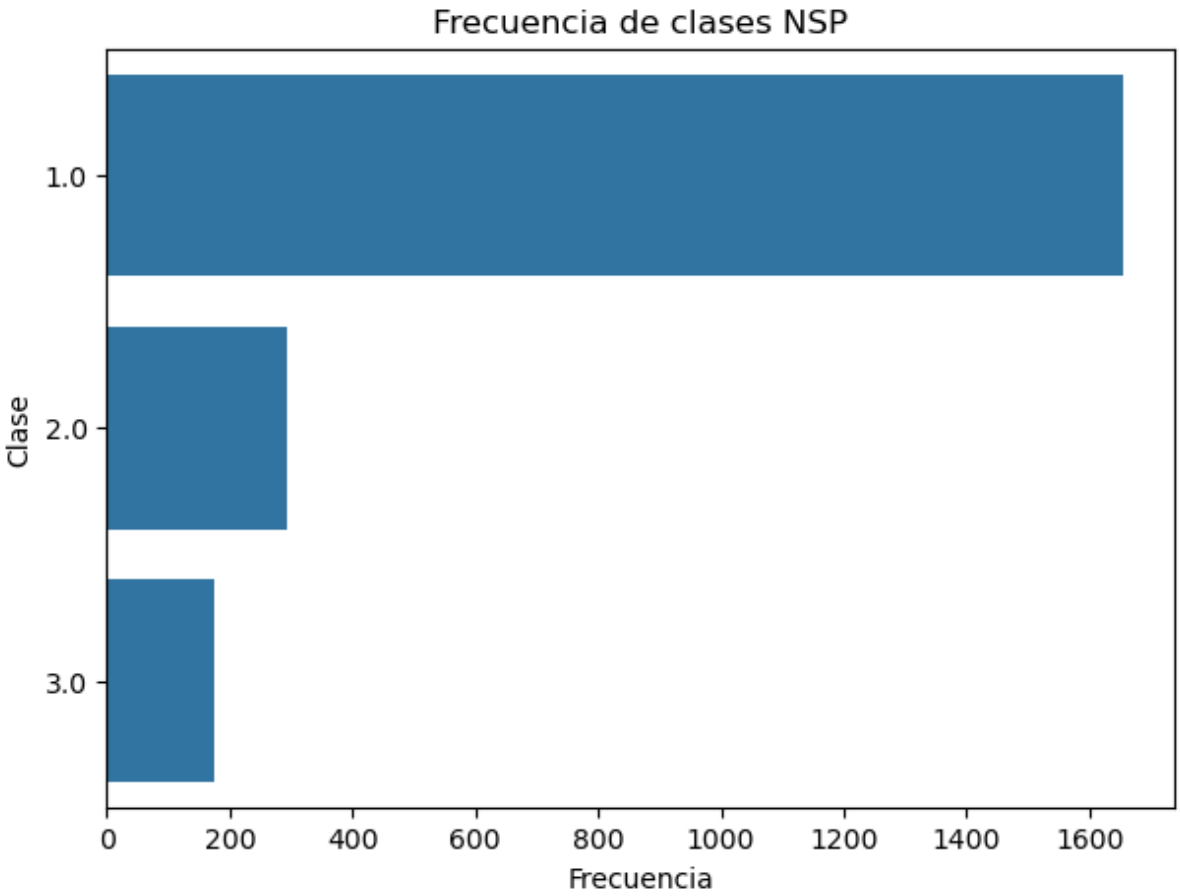




Podemos observar que se conservan los outliers para la ultima categoria, y las

primeras dos tienen una distribucion y simetria similar

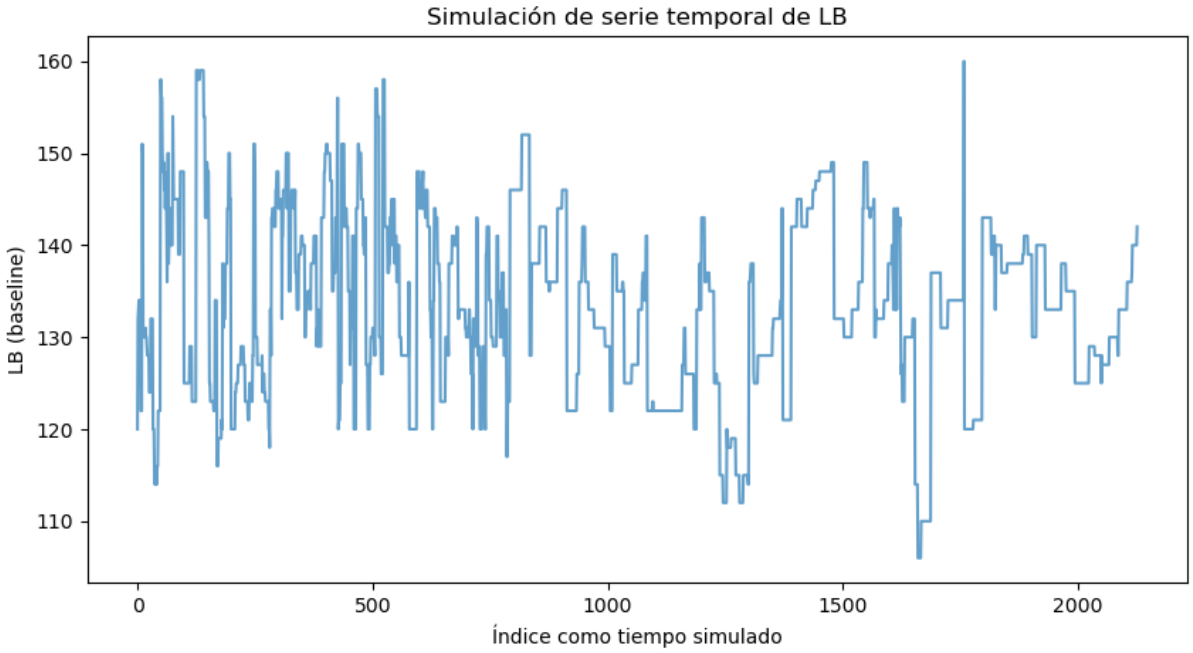
In [33]:
```
#Barras Horizontales Ordenadas por frecuencia descendente
conteos = df["NSP"].value_counts()

plt.figure(figsize=(7,5))
sns.barplot(
    x=conteos.values,
    y=conteos.index,
    orient='h'
)
plt.title("Frecuencia de clases NSP")
plt.xlabel("Frecuencia")
plt.ylabel("Clase")
plt.show()
```



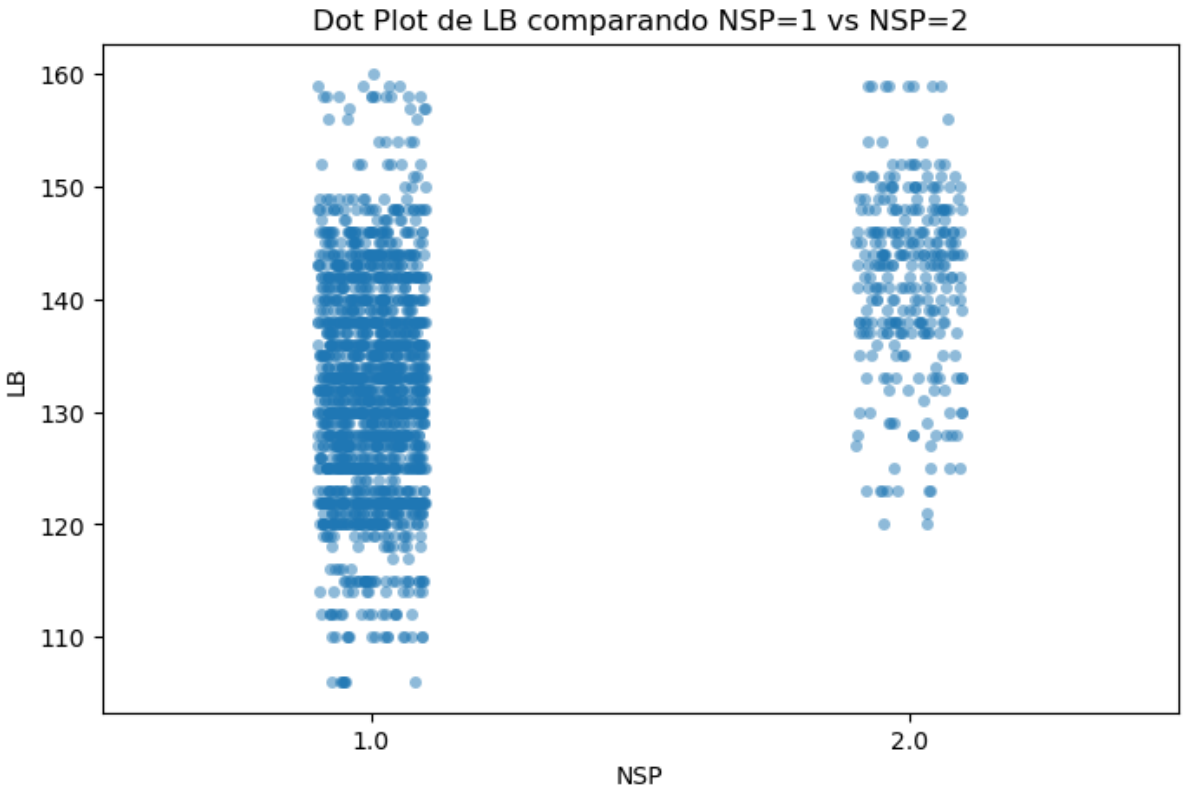
Camparamos los conteos para categoria y la normal tiene la mayoria, mientras que la patologica es la que menos observaciones tiene

In [34]:
```
#Líneas Simular serie temporal (ordenar por índice o simular variable
df_ordenado = df.sort_index()   # simulamos tiempo

plt.figure(figsize=(10,5))
plt.plot(df_ordenado.index, df_ordenado["LB"], alpha=0.7)
plt.title("Simulación de serie temporal de LB")
```

```
plt.xlabel("Índice como tiempo simulado")
plt.ylabel("LB (baseline)")
plt.show()
```

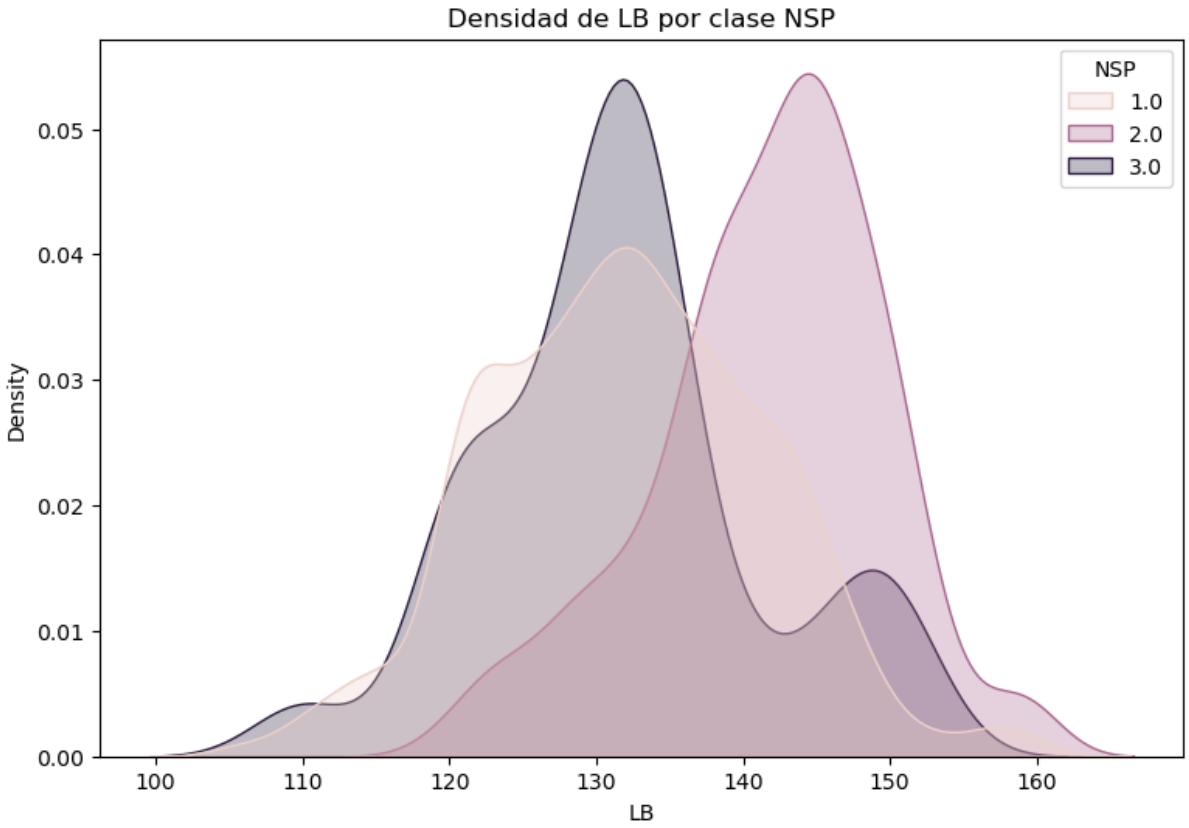


In [35]:
```
#Dot Plots Comparación entre 2 grupos (overlay)
df_dos = df[df["NSP"].isin([1,2])]

plt.figure(figsize=(8,5))
sns.stripplot(
    data=df_dos,
    x="NSP",
    y="LB",
    jitter=True,
    alpha=0.5
)
plt.title("Dot Plot de LB comparando NSP=1 vs NSP=2")
plt.show()
```



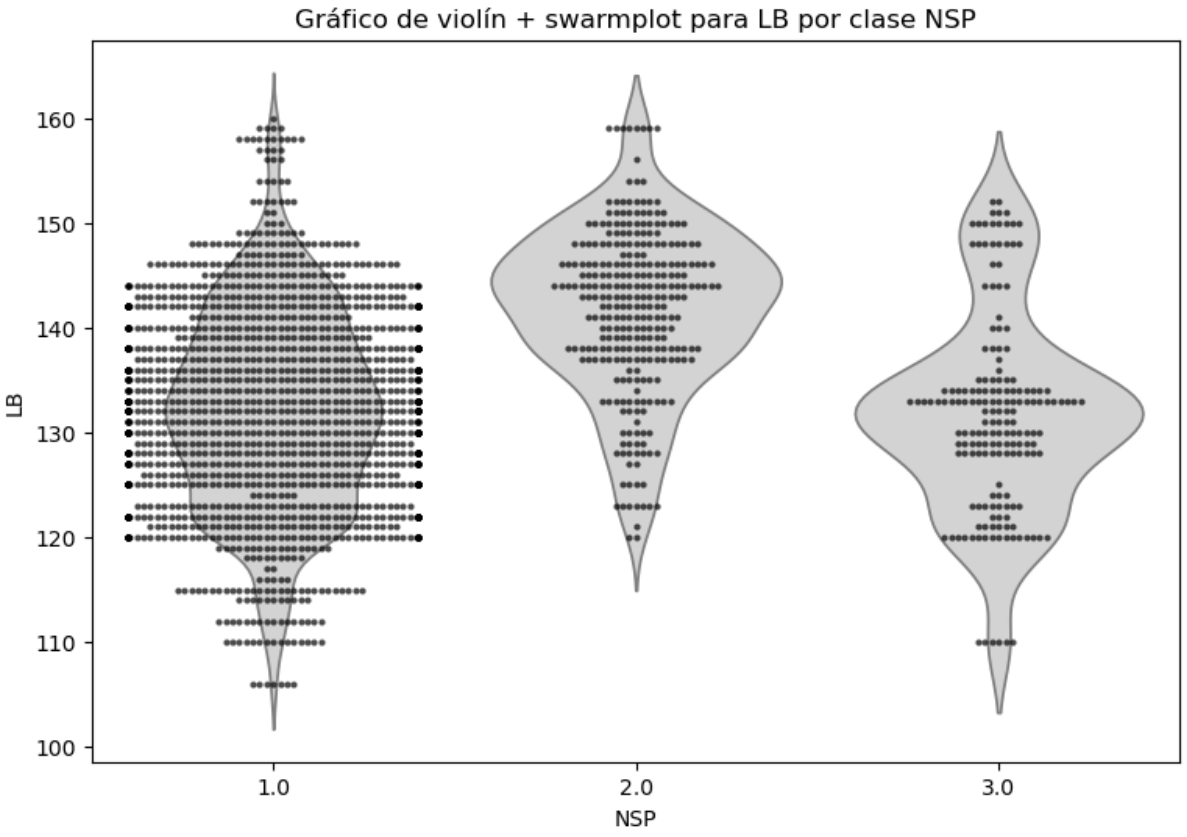
Otra forma de ver la distribución de los datos y comprar , se nota la carga de la segunda categoria hacia valores medios sin observaciones el los valores bajos de LB

In [36]:
```
#Densidad Múltiples clases con diferentes colores
plt.figure(figsize=(9,6))
sns.kdeplot(
    data=df,
    x="LB",
    hue="NSP",
    common_norm=False,
    fill=True,
    alpha=0.3
)
plt.title("Densidad de LB por clase NSP")
plt.show()
```

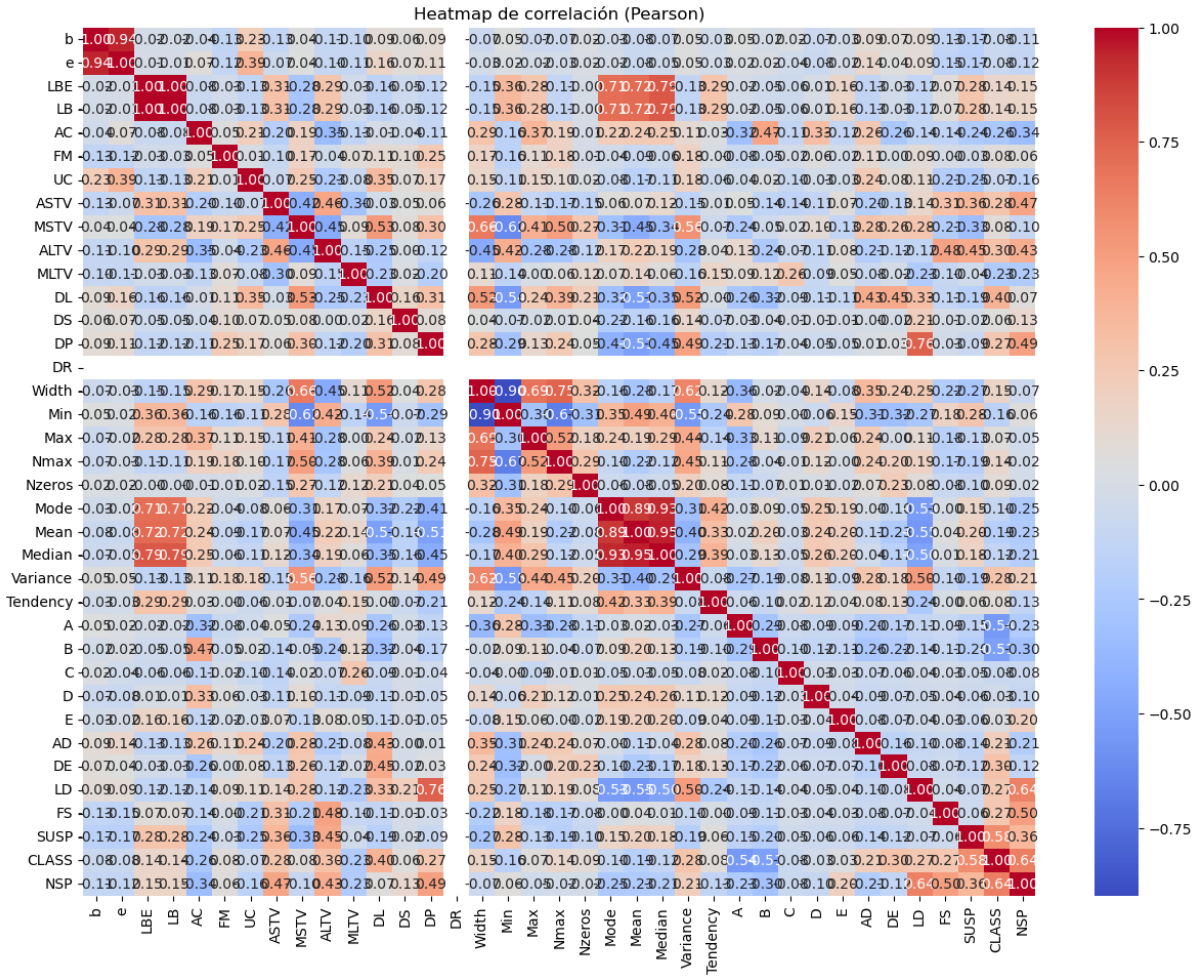


In [37]:
```
#Violín Overlay con swarmplot
plt.figure(figsize=(9,6))

sns.violinplot(
    data=df,
    x="NSP",
    y="LB",
    inner=None,
    color="lightgray"
)

sns.swarmplot(
    data=df,
    x="NSP",
    y="LB",
    color="black",
    size=3,
    alpha=0.7
)

plt.title("Gráfico de violín + swarmplot para LB por clase NSP")
plt.show()
```

```
/Users/luisalamo/miniforge3/envs/ds/lib/python3.11/site-packages/seaborn/categorical.py:3399: UserWarning: 25.2% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```



In [38]:
```
#Heatmap Correlación + anotaciones y selección de método ('pearson', '
corr_pearson = df[columnas_numericas].corr(method="pearson")

plt.figure(figsize=(15,11))
sns.heatmap(
    corr_pearson,
    annot=True,
    fmt=".2f",
    cmap="coolwarm"
)
plt.title("Heatmap de correlación (Pearson)")
plt.show()
```



Hay varias variables con una alta correlacion Son redundantes; podemos considerar reducción de dimensionalidad (PCA) o eliminar algunas.

4. Construcción de una Librería Python

Se creo la libreria con:

Tipado estático (def func(x: pd.DataFrame) -> pd.DataFrame)

Docstrings (estilo Google o NumPy)

Pruebas básicas con pytest

5. Validación y Reporte

Crear pruebas unitarias para al menos 4 funciones clave

Generar un Jupyter Notebook de ejemplo con uso de cada función

Exportar el reporte de análisis como PDF con:

Descripción de funciones

Visualizaciones generadas

Recomendaciones analíticas breves

In [52]:
```
# Ejemplo de uso de funciones
import os, sys
sys.path.append(os.path.abspath(".."))
sys.path.append(os.path.abspath("../.."))

from ctg_viz import (
    eliminar_nulos_altos,
    imputar_valores,
    imputar_knn,
    eliminar_outliers_iqr,
    eliminar_outliers_zscore,
    check_data_completeness_luis_manuel_alamo_diaz
)

df_sin_nulos_altos = eliminar_nulos_altos(df, umbral=0.20)
df_imputado = imputar_valores(df)
df_imputado_knn = imputar_knn(df)
df_sin_outliers_iqr = eliminar_outliers_iqr(df)
df_sin_outliers_z = eliminar_outliers_zscore(df, umbral=3)

reporte_completitud = check_data_completeness_luis_manuel_alamo_diaz(d
display(reporte_completitud)
```

columna	tipo_dato	nulos	completitud_%	n_unicos	tipo_variable	min	ma
FileName	object	3	99.859089	352	no_numerica	NaN	Na
Date	object	3	99.859089	48	no_numerica	NaN	Na
SegFile	object	3	99.859089	2126	no_numerica	NaN	Na
b	float64	3	99.859089	979	continua	0.0	3296
e	float64	3	99.859089	1064	continua	287.0	3599
LBE	float64	3	99.859089	48	continua	106.0	160
LB	float64	3	99.859089	48	continua	106.0	160
AC	float64	3	99.859089	22	continua	0.0	26
FM	float64	2	99.906059	96	continua	0.0	564
UC	float64	2	99.906059	19	continua	0.0	23
ASTV	float64	2	99.906059	75	continua	12.0	87

MSTV	float64	2	99.906059	57	continua	0.2	7
ALTV	float64	2	99.906059	87	continua	0.0	91
MLTV	float64	2	99.906059	249	continua	0.0	50
DL	float64	1	99.953030	15	continua	0.0	16
DS	float64	1	99.953030	2	discreta	0.0	1
DP	float64	1	99.953030	5	discreta	0.0	4
DR	float64	1	99.953030	1	discreta	0.0	0
Width	float64	3	99.859089	154	continua	3.0	180
Min	float64	3	99.859089	109	continua	50.0	159
Max	float64	3	99.859089	86	continua	122.0	238
Nmax	float64	3	99.859089	18	continua	0.0	18
Nzeros	float64	3	99.859089	9	discreta	0.0	10
Mode	float64	3	99.859089	88	continua	60.0	187
Mean	float64	3	99.859089	103	continua	73.0	182
Median	float64	3	99.859089	95	continua	77.0	186
Variance	float64	3	99.859089	133	continua	0.0	269
Tendency	float64	3	99.859089	3	discreta	-1.0	1
A	float64	3	99.859089	2	discreta	0.0	1
B	float64	3	99.859089	2	discreta	0.0	1
C	float64	3	99.859089	2	discreta	0.0	1
D	float64	3	99.859089	2	discreta	0.0	1
E	float64	3	99.859089	2	discreta	0.0	1
AD	float64	3	99.859089	2	discreta	0.0	1
DE	float64	3	99.859089	2	discreta	0.0	1
LD	float64	3	99.859089	2	discreta	0.0	1
FS	float64	3	99.859089	2	discreta	0.0	1
SUSP	float64	3	99.859089	2	discreta	0.0	1
CLASS	float64	3	99.859089	10	continua	1.0	10
NSP	float64	3	99.859089	3	discreta	1.0	3

Las graficas ya estan generadas arriba con sus descripciones y analisis

Concluciones

La imputación ayudó a completar los valores faltantes sin afectar mucho la forma general del dataset. Esto permite trabajar con todas las variables sin perder información. Aun así, hay columnas que están muy correlacionadas entre sí, por lo que sería posible reducir dimensiones eliminando algunas que aportan lo mismo o aplicando algo como PCA para quedarnos con menos variables que representen el mismo comportamiento. Las columnas LB (latidos por minuto) y NSP (la clase del feto) son de las más relevantes: LB tiene mucha relación con varias medidas de variabilidad del latido. NSP es la variable objetivo y varias columnas se mueven junto con ella, aunque ninguna por sí sola la explica al 100%. En resumen, la imputación deja el dataset más limpio, varias variables se pueden resumir en menos dimensiones porque son muy parecidas entre sí, y LB y NSP son claves para entender el comportamiento general del conjunto.