

```
from google.colab import drive
import os


# Montar Google Drive
drive.mount('/content/drive')

# Definir la ruta de la carpeta
base_path = "/content/drive/My Drive/Classroom/TCD"
```


 Mounted at /content/drive

```
import pandas as pd
data_clinic = pd.read_csv(f"{base_path}/data_clinic.csv")
```

```
data_clinic.head()
```



	subject_id	date	time	age	gender	temperature	abp_systolic	abp_diastolic
0	1000	2001-04-05	00:11:00	57	F	39.17	106.10	37.94
1	1000	2001-04-05	04:26:00	57	F	38.18	138.53	72.41
2	1000	2001-04-05	06:25:00	57	F	36.15	76.05	53.58

◀  ▶

Para el desarrollo de este estudio se emplea un conjunto de datos clínicos provenientes de registros electrónicos de salud (EHR) de pacientes ingresados en unidades de cuidados intensivos (UCI) con diagnóstico principal relacionado a enfermedades cardiovasculares. Este tipo de información es crítica para entender la evolución de los pacientes en estado delicado y posibilita el análisis de patrones asociados al reingreso a UCI, evolución clínica, respuesta a tratamientos y factores de riesgo.

El dataset incluye información estructurada, registrada de forma periódica durante la hospitalización del paciente. La metadata asociada permite comprender la naturaleza, origen, periodicidad y tipo de cada atributo presente en el conjunto de datos.

✓ Paso 1: Analizar el comportamiento de tus datos

Para este conjunto de datos de registro electrónico de salud de UCI se está usando la herramienta AutoProfiler para ayudarnos con el análisis de los datos.

✓ Cantidad de registros

Según los resultados de la consulta con la herramienta AutoProfiler tiene:

- Filas: 516 413
- Columnas: 18

Esta data según los autores es la fusión de varias datas y mencionan que esta limpia, pero debemos verificar si es cierto

▼ **data_clinic** 516,413 x 18



- Esto es mucho para un Excel, pero manejable para Python (pandas) si tu PC tiene al menos 8 a 12 GB de RAM. Tu laptop tiene 12 GB RAM, así que puedes trabajar con esta cantidad sin problemas usando pandas, aunque para deep learning podría necesitarse preprocesamiento o muestreo.
- No son demasiados para tareas de ciencia de datos o entrenamiento de modelos si se gestiona adecuadamente la memoria.
- Si se plantea para entrenar modelos de deep learning (como RNNs o LSTMs), se podría considerar usar subconjuntos o trabajar por lotes (batching) y técnicas como DataLoader en PyTorch.

✓ ¿Qué representa un registro?

Un registro representa una medición o conjunto de observaciones clínicas en un instante específico para un paciente en UCI. Es decir, cada fila del DataFrame es un conjunto de

variables biomédicas medidas para un paciente (subject_id) en un día y hora particular (date, time).

Esto es común en datasets de monitoreo en UCI, donde se registran múltiples mediciones por día por paciente. Significado de cada perfil de columna:

- subject_id: Identificador único y anonimizado del paciente. Permite rastrear registros individuales a lo largo del tiempo sin revelar su identidad.
- date: Fecha en la que se registraron los signos vitales o exámenes clínicos. Formato: YYYY-MM-DD.
- time: Hora del día en que se tomó la muestra o se midió el dato clínico. Es importante para estudios temporales o evolución de parámetros.
- age: Edad del paciente al momento del registro. Generalmente en años. Este dato es crucial para análisis de riesgo y segmentación
- gender: Sexo biológico del paciente. Normalmente codificado como M (masculino) o F (femenino). Puede influir en la presentación clínica y pronóstico
- temperature: Temperatura corporal del paciente, normalmente en grados Celsius. Indicador de infecciones o respuesta inflamatoria.
- abp_systolic: Presión arterial sistólica (valor más alto durante un latido). Mide la presión en las arterias cuando el corazón late. Unidad: mmHg.
- abp_diastolic: Presión arterial diastólica (valor más bajo entre latidos). Mide la presión cuando el corazón está en reposo entre latidos. Unidad: mmHg.
- abp_mean: Presión arterial media (MAP). Es una medida ponderada del ciclo cardíaco, muy importante en UCI para evaluar perfusión.
- heart_rate: Frecuencia cardíaca en latidos por minuto (bpm). Indicador clave de estado hemodinámico.
- oxygen_saturation: Saturación de oxígeno en sangre (SpO₂), expresado en porcentaje. Refleja la oxigenación del cuerpo. Un valor < 90% puede indicar hipoxemia.
- weight: Peso del paciente, usualmente en kilogramos (kg). Útil para calcular dosis de medicamentos o índices clínicos como el IMC.
- creatine: Nivel de creatinina en sangre (mg/dL o $\mu\text{mol/L}$). Indicador de función renal; valores altos pueden indicar daño renal agudo.
- ph: Medida del pH sanguíneo. El valor normal está entre 7.35 y 7.45. Desequilibrios pueden indicar acidosis o alcalosis, condiciones críticas en UCI.
- sodium: Concentración de sodio en sangre, normalmente en mEq/L. Esencial para el equilibrio electrolítico. Desviaciones pueden causar convulsiones o arritmias.
- potassium: Nivel de potasio en sangre (mEq/L). Crucial para la función cardíaca. Valores anormales están asociados a riesgo de paro cardíaco.
- hematocrit: Porcentaje de volumen de glóbulos rojos en la sangre. Indicador de anemia o deshidratación.
- bilirubin: Nivel de bilirrubina en sangre (mg/dL). Marcador de función hepática. Elevaciones pueden indicar falla hepática o hemólisis.

✓ ¿Qué datos son discretos, continuos, nominales o ordinales?

Columna	Tipo	Naturaleza	Límites	Unidad (si aplica)
subject_id	int / str	Discreto, categórico	1000 – 1499	*
date	fecha	Discreto temporal (2,191 fechas únicas)	2001-03-31 – 2007-03-29	*
time	hora	Discreto temporal (1,440 valores únicos)	00:00:00 – 23:59:00	*
age	int	Discreto (puede tratarse como continuo)	19 – 89	años
gender	str / categórico	Nominal (2 valores únicos: 'M' y 'F')	M, F	*
temperature	float	Continuo	36.0 – 40.0	°C
abp_systolic	float	Continuo	70.0 – 170.0	mmHg
abp_diastolic	float	Continuo	30.0 – 80.0	mmHg
abp_mean	float	Continuo	43.4 – 110.0	mmHg
heart_rate	float	Continuo	50.0 – 157.0	bpm (latido)
oxygen_saturation	float	Continuo	90.0 – 100.0	%
weight	float	Continuo (con posible error de signos)	-329.0 – 157.0	kg
creatinine	float	Continuo	0.40 – 2.60	mg/dL o µm
ph	float	Continuo	6.8 – 7.7	adimension
sodium	float	Continuo	117.0 – 166.0	mEq/L
potassium	float	Continuo	2.0 – 8.8	mEq/L
hematocrit	float	Continuo	8.9 – 53.3	%
bilirubin	float	Continuo	0.1 – 45.0	mg/dL o µm

Observaciones:

- En la columna weight se observa valores negativos, cosa que es imposible

✓ ¿Niveles de granularidad temporal?

Sí, hay tres niveles:

- Diario (date)
- Por minuto o segundo (time)
- Posiblemente múltiples observaciones por día por paciente

Esto se debe considerar para análisis temporales, reagrupamiento (resampling) o entrada a modelos secuenciales (LSTM, GRU).

✓ ¿Están todas las filas completas o tenemos campos con valores nulos?

▼ **data_clinic** 516,413 x 18



Según la información de AutoProfiler se puede visualizar que ninguna columna contiene valores vacíos

▼ ¿Todos los datos están en su formato adecuado?

`data_clinic.dtypes`



0

subject_id	int64
date	object
time	object
age	int64
gender	object
temperature	float64
abp_systolic	float64
abp_diastolic	float64
abp_mean	float64
heart_rate	float64
oxygen_saturation	float64
weight	int64
creatinine	float64
ph	float64
sodium	float64
potassium	float64
hematocrit	float64
bilirubin	float64

dtype: object

```
print(data_clinic['date'].apply(type).value_counts(), "\n")
print(data_clinic['time'].apply(type).value_counts(), "\n")
print(data_clinic['gender'].apply(type).value_counts(), "\n")
```



```
date
<class 'str'>    516413
Name: count, dtype: int64

time
<class 'str'>    516413
Name: count, dtype: int64

gender
<class 'str'>    516413
Name: count, dtype: int64
```

Conclusiones de este análisis:

- Que la columna date esta en formato "str" y debemos pasarlo a date

- Que la columna time esta en formato "str" y debemos pasarlo a un formato adecuado

✓ ¿Hay datos duplicados?

```
print(data_clinic.duplicated().sum())
```

0

No hay filas duplicadas

✓ ¿Siguen alguna distribución?

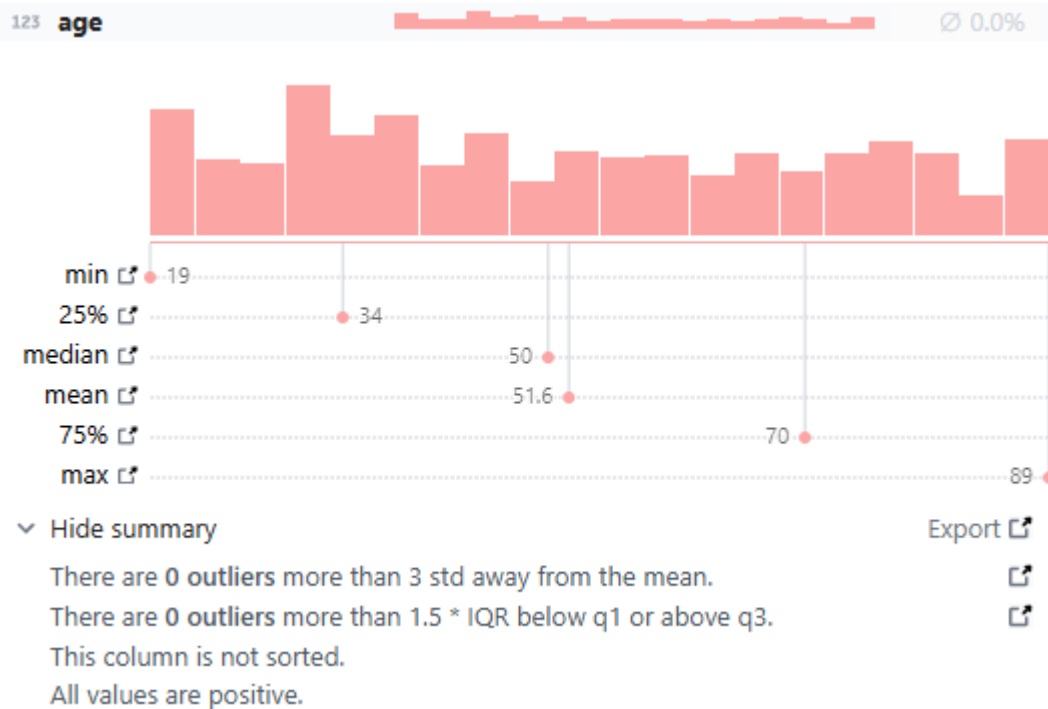
```
data_clinic.describe()
```

	subject_id	age	temperature	abp_systolic	abp_diastolic	
count	516413.000000	516413.000000	516413.000000	516413.000000	516413.000000	5164
mean	1247.625770	51.609483	38.002019	120.041427	55.010305	
std	144.847974	20.653736	1.154592	28.871779	14.430982	
min	1000.000000	19.000000	36.000000	70.000000	30.000000	
25%	1121.000000	34.000000	37.000000	95.030000	42.500000	
50%	1248.000000	50.000000	38.000000	120.090000	55.030000	
75%	1376.000000	70.000000	39.000000	145.050000	67.500000	
max	1499.000000	89.000000	40.000000	170.000000	80.000000	10

Con la ayuda de AutoProfiler podemos determinar los mismo valores que usando la función describe() de los valores flotantes y enteros

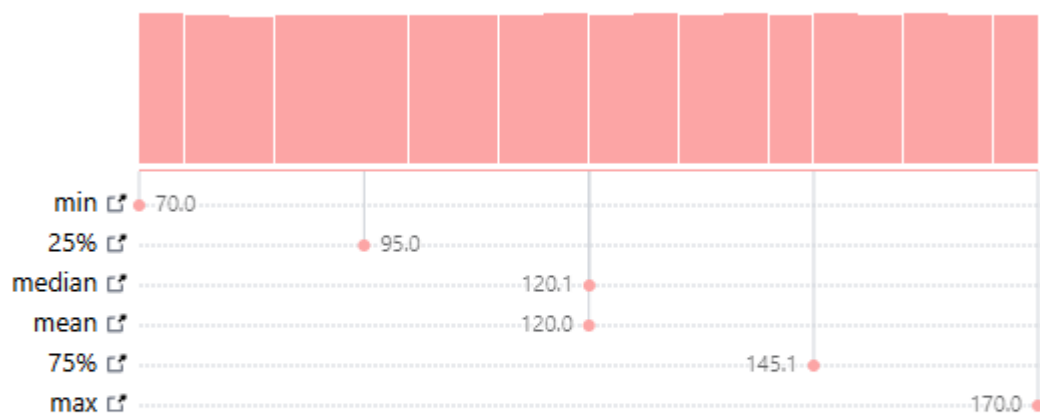
✓ Paso 2: Análisis de outliers

Esto tambien se puede determiar con los graficos proporcionados de Autoprofiler para ver si tenemos valores atipicos, con histogramas. Pero se esta aplicando boxplots para poder visualizar mejor



abp_systolic

0.0%



▼ Hide summary

Export

There are 0 outliers more than 3 std away from the mean.

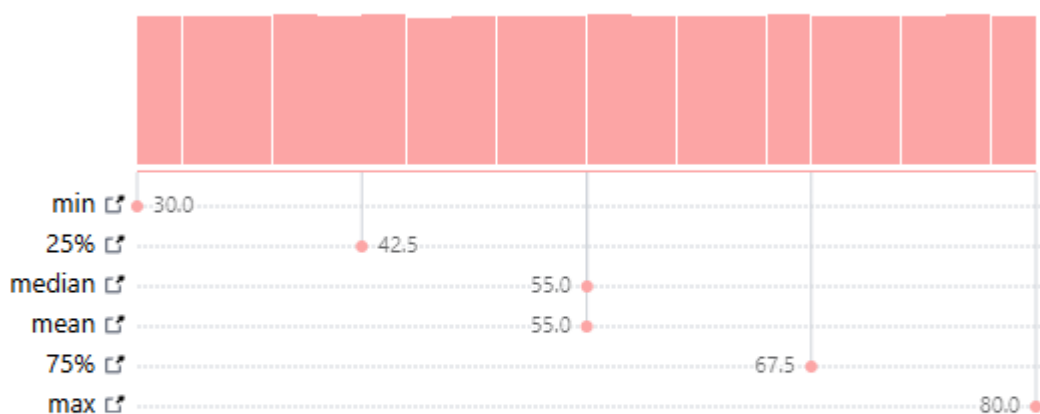
There are 0 outliers more than 1.5 * IQR below q1 or above q3.

This column is not sorted.

All values are positive.

abp_diastolic

0.0%



▼ Hide summary

Export

There are 0 outliers more than 3 std away from the mean.

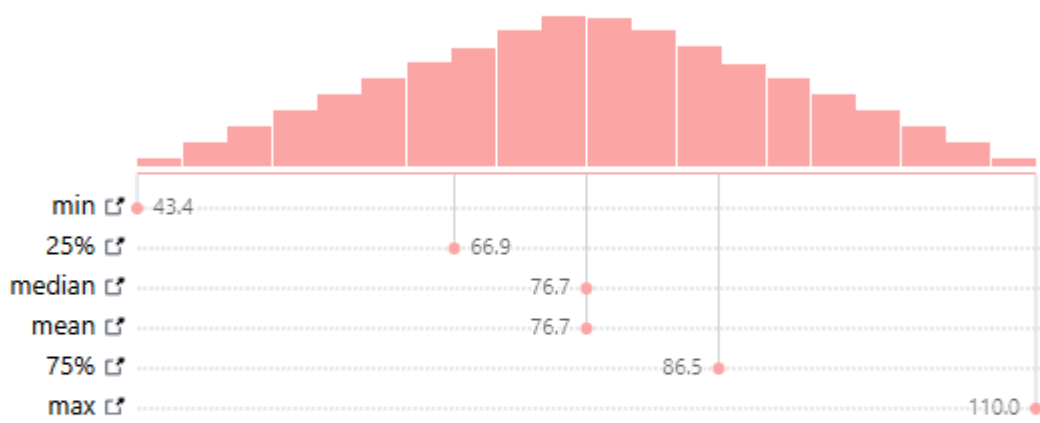
There are 0 outliers more than 1.5 * IQR below q1 or above q3.

This column is not sorted.

All values are positive.

abp_mean

0.0%



Hide summary

Export

There are 0 outliers more than 3 std away from the mean.

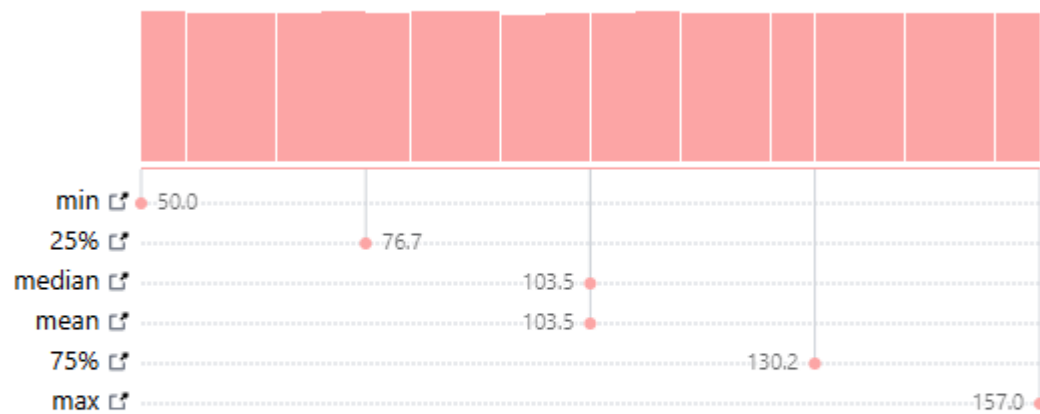
There are 0 outliers more than 1.5 * IQR below q1 or above q3.

This column is not sorted.

All values are positive.

heart_rate

0.0%



Hide summary

Export

There are 0 outliers more than 3 std away from the mean.

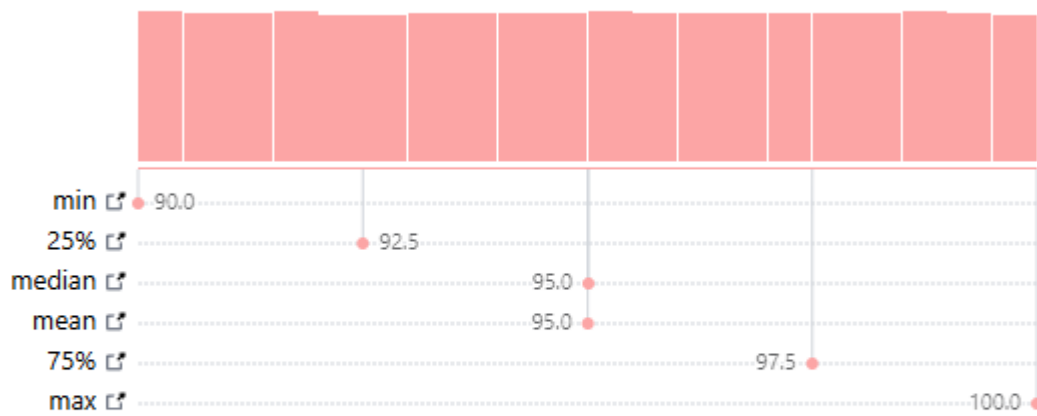
There are 0 outliers more than 1.5 * IQR below q1 or above q3.

This column is not sorted.

All values are positive.

oxygen_saturation

0.0%



Hide summary

Export

There are 0 outliers more than 3 std away from the mean.

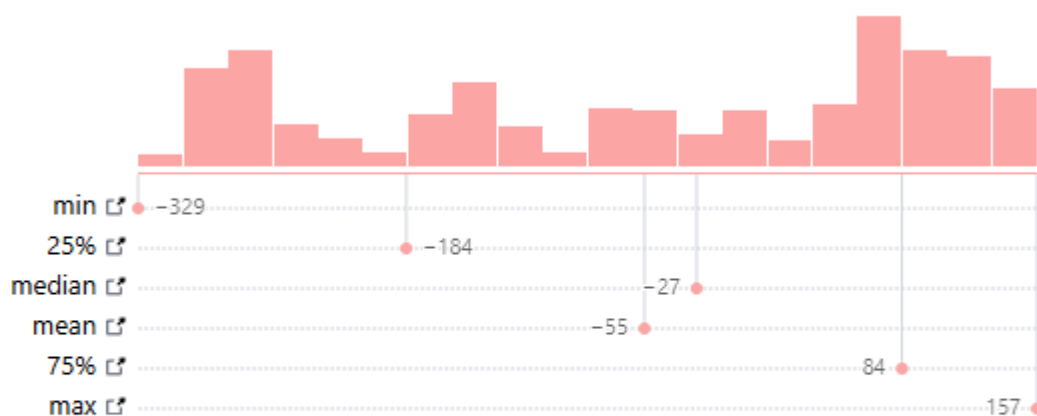
There are 0 outliers more than 1.5 * IQR below q1 or above q3.

This column is not sorted.

All values are positive.

123 weight

0.0%



Hide summary

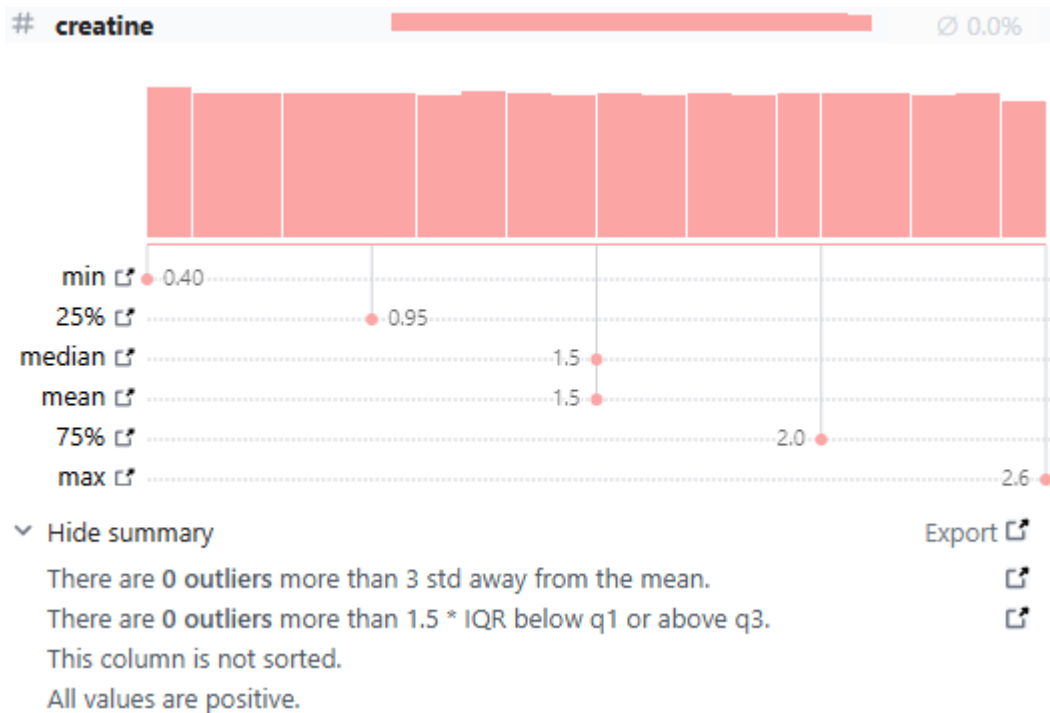
Export

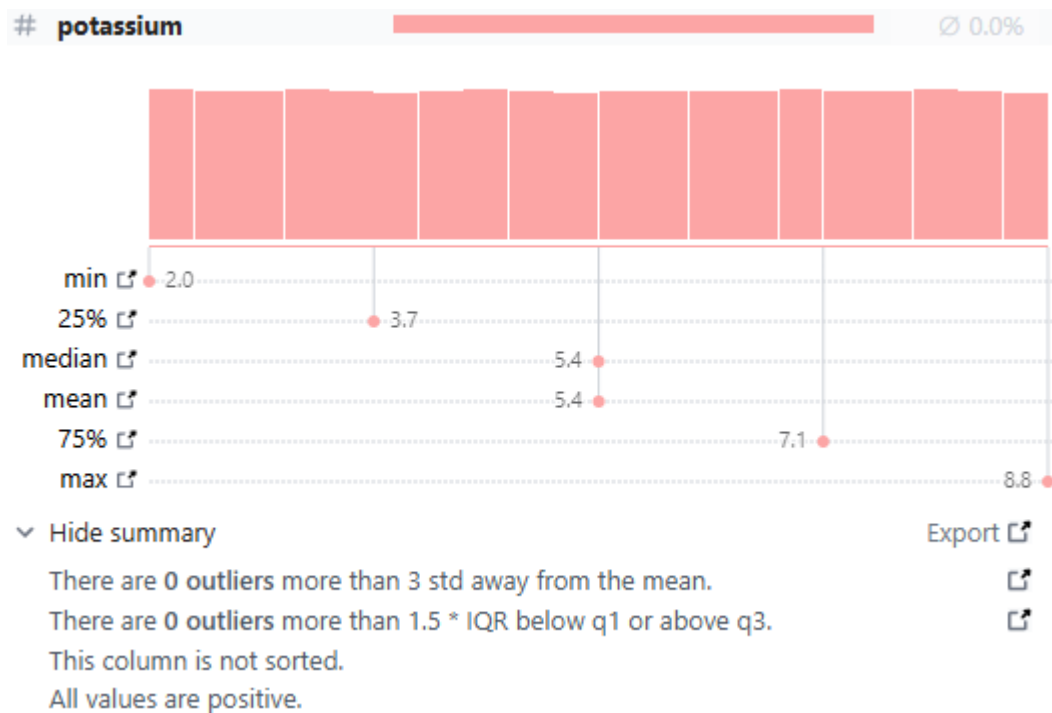
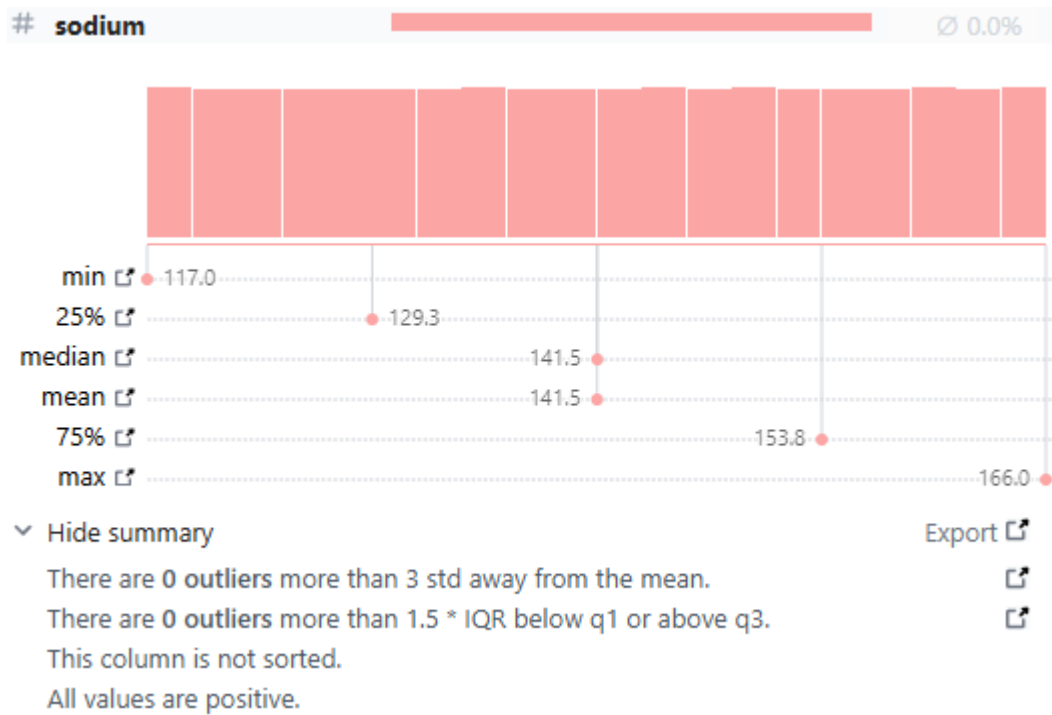
There are 0 outliers more than 3 std away from the mean.

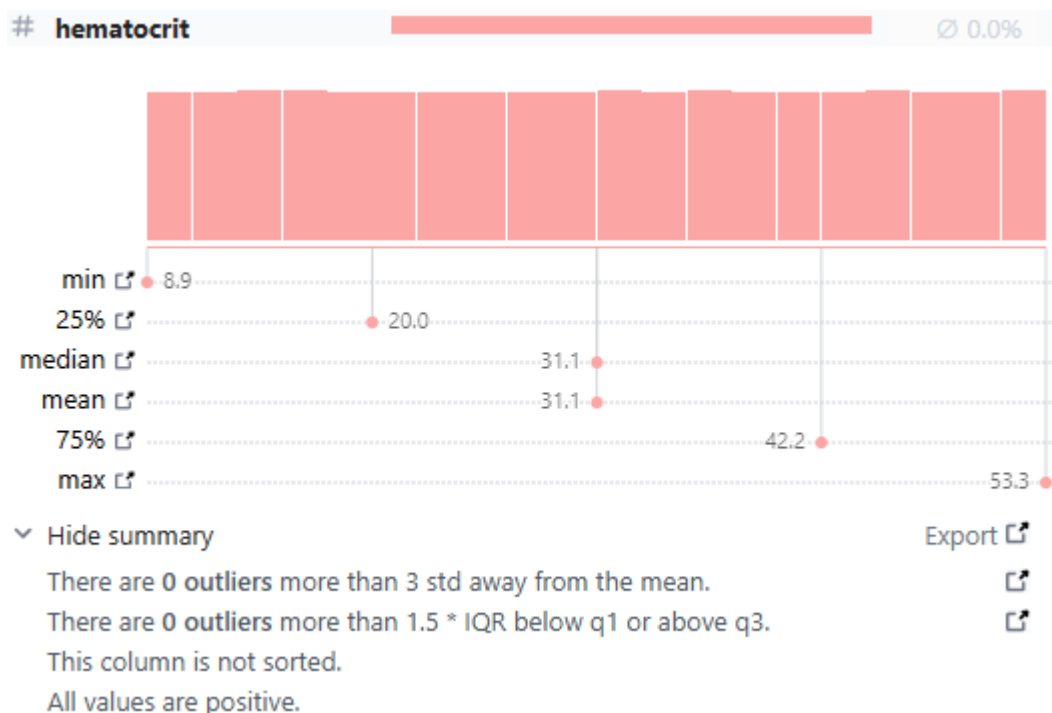
There are 0 outliers more than 1.5 * IQR below q1 or above q3.

This column is not sorted.

There are 746 zero values, 234051 positive values, and 281,616 negative values.







```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

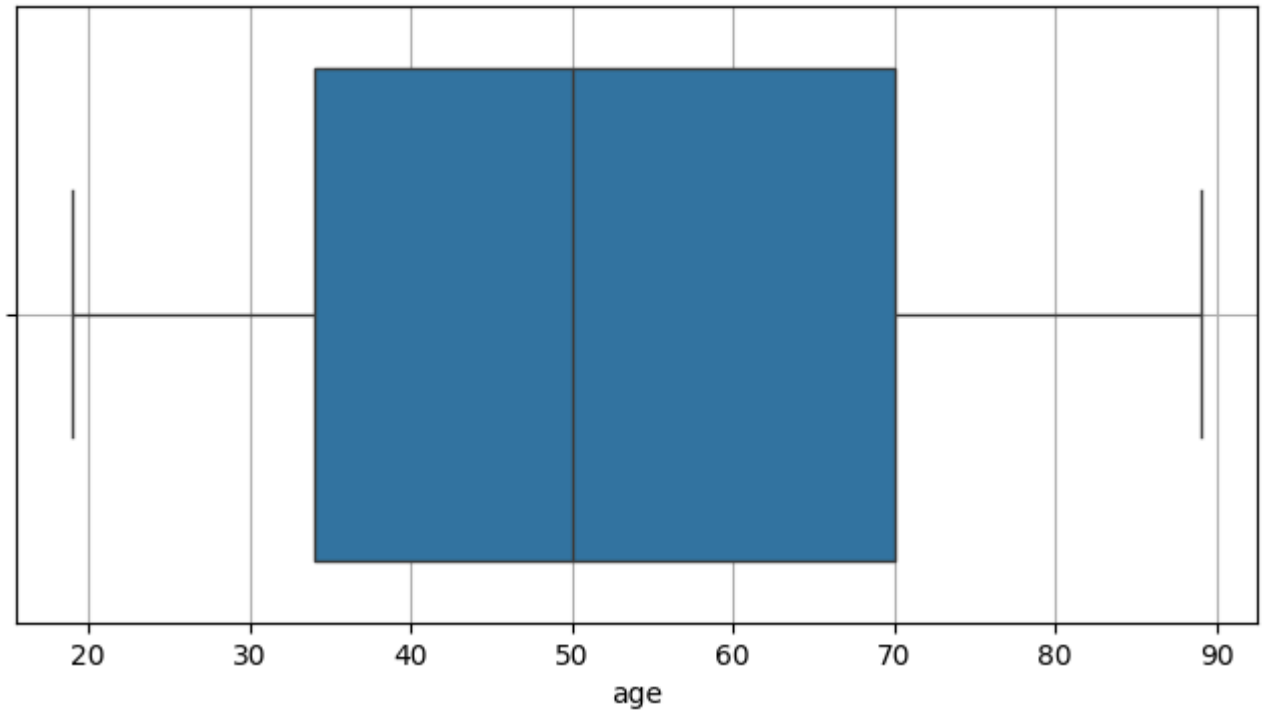
```
df = data_clinic.copy()
numeric_cols = ['age', 'temperature', 'abp_systolic', 'abp_diastolic', 'abp_mean', 'heart

for col in numeric_cols:
    plt.figure(figsize=(8, 4))
    sns.boxplot(x=df[col])
    plt.title(f'Distribución y outliers de: {col}')
```

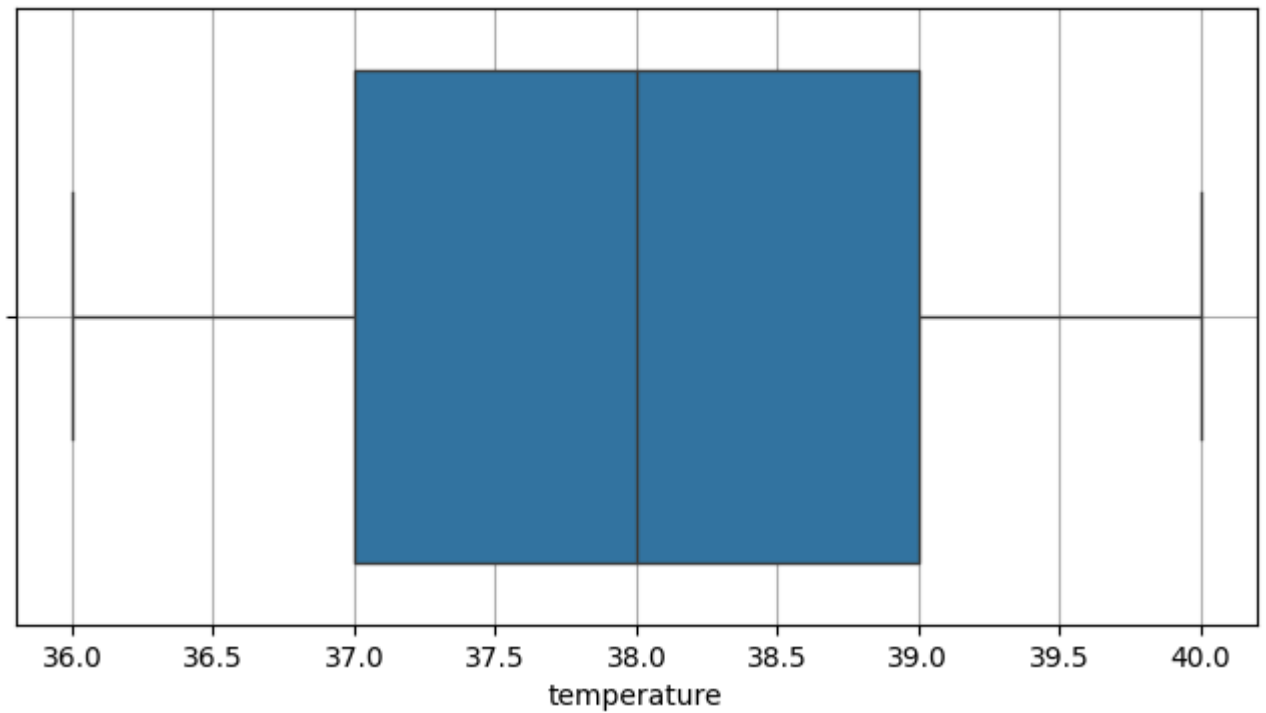
```
plt.xlabel(col)
plt.grid(True)
plt.show()
```



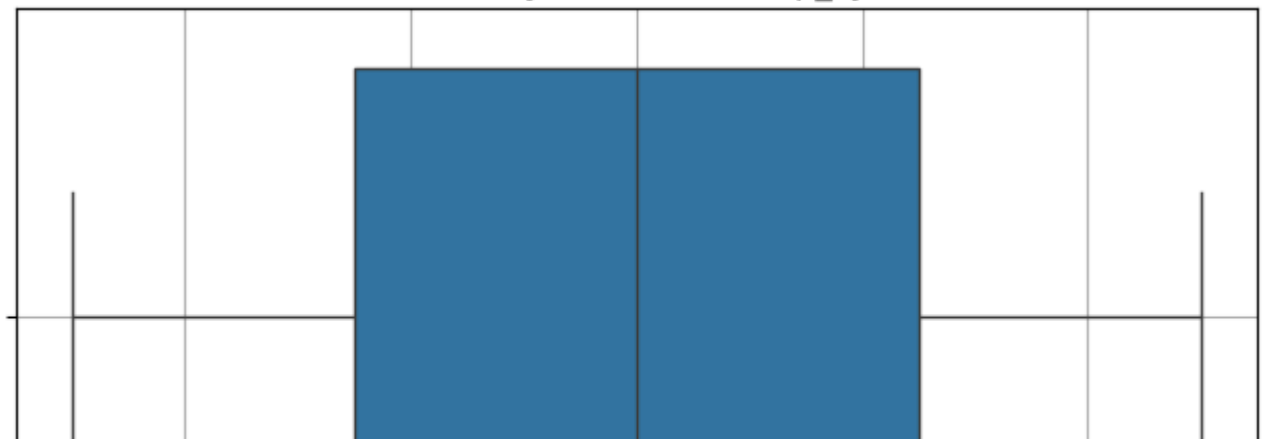
Distribución y outliers de: age

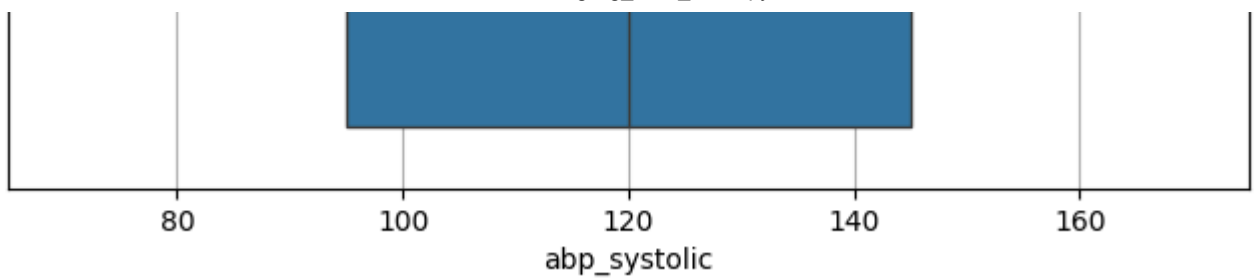


Distribución y outliers de: temperature

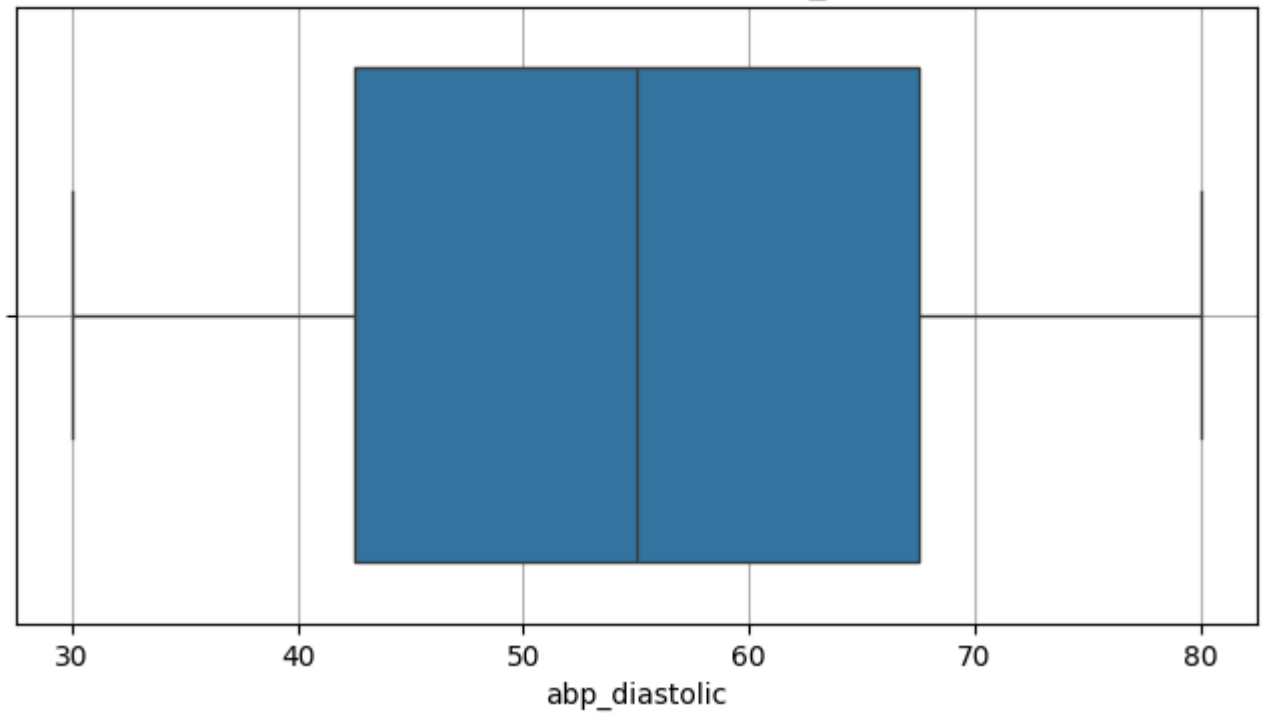


Distribución y outliers de: abp_systolic

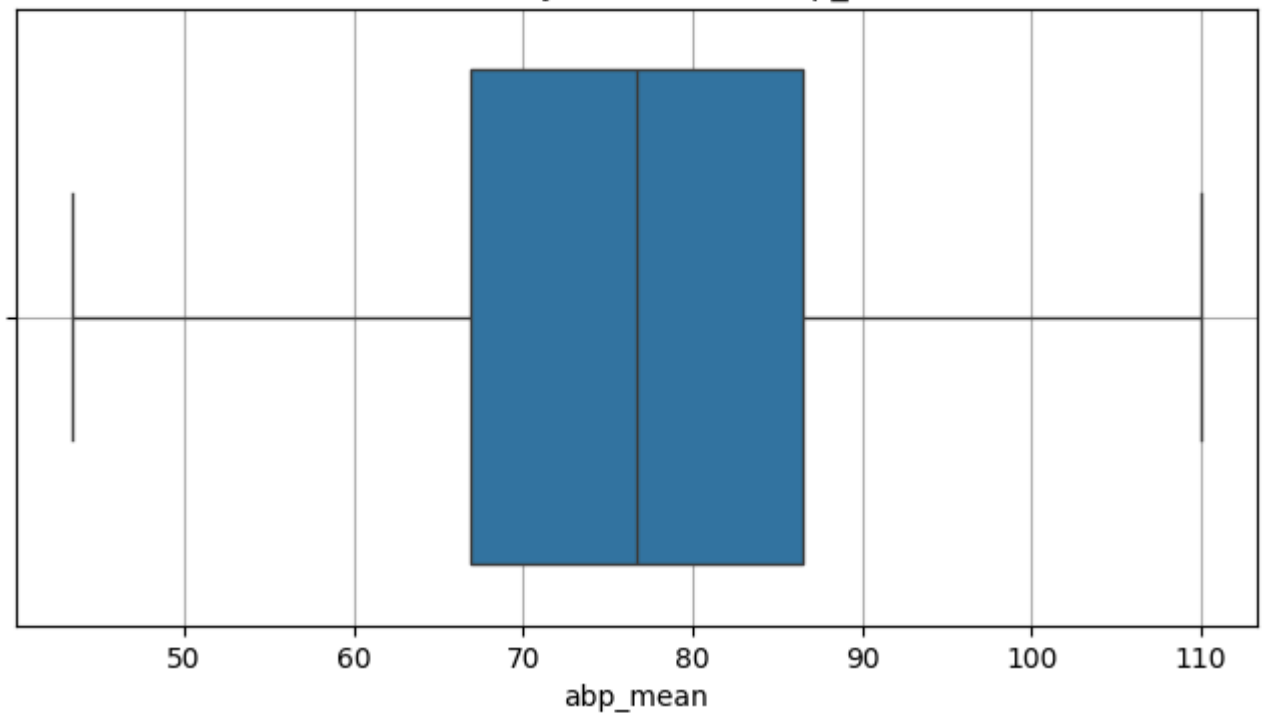




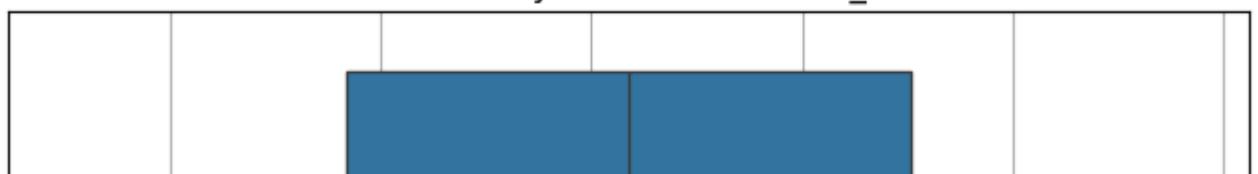
Distribución y outliers de: abp_diastolic

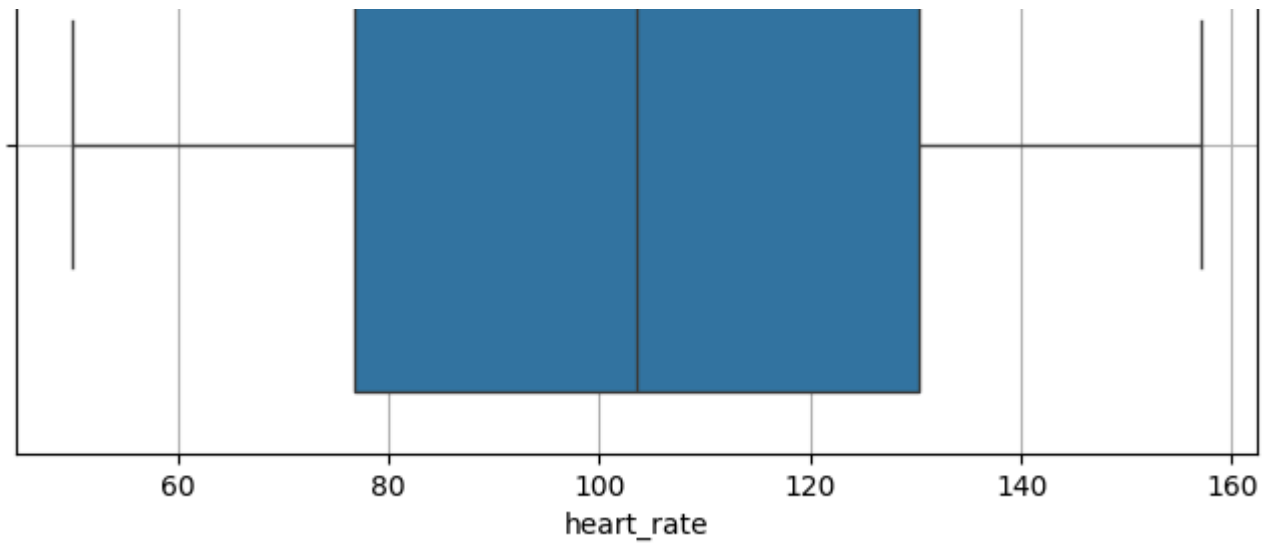


Distribución y outliers de: abp_mean

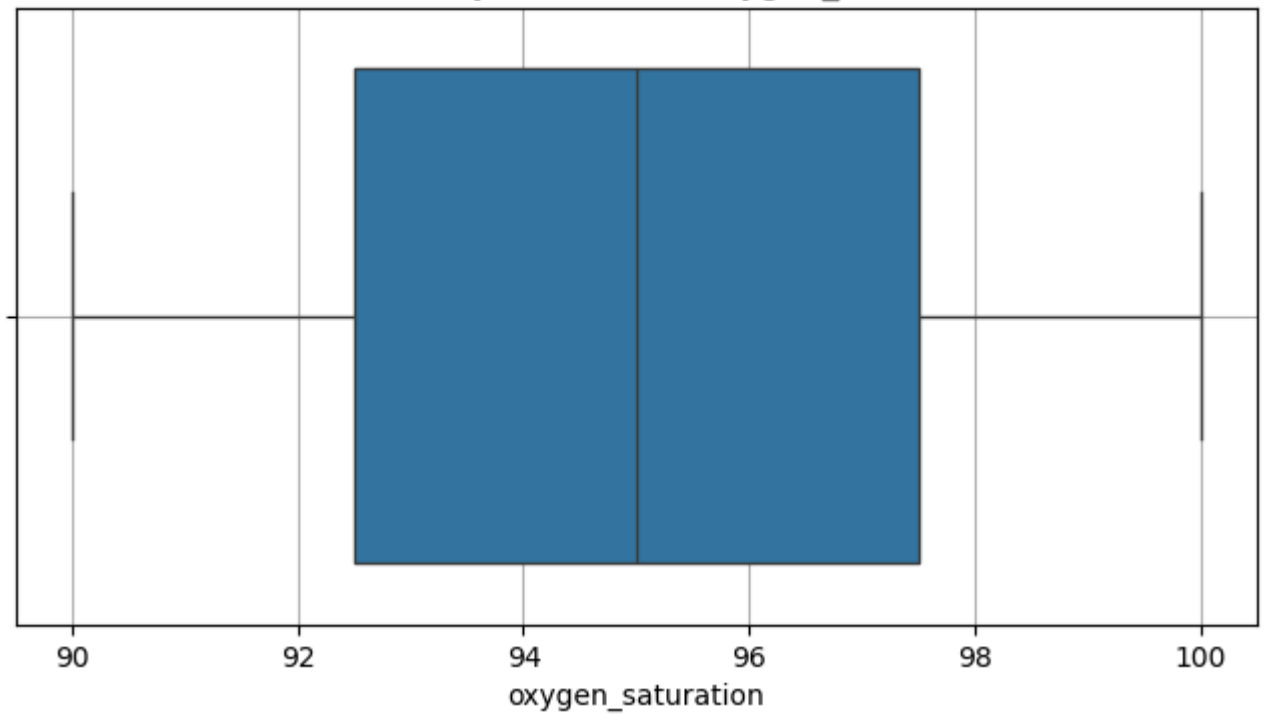


Distribución y outliers de: heart_rate

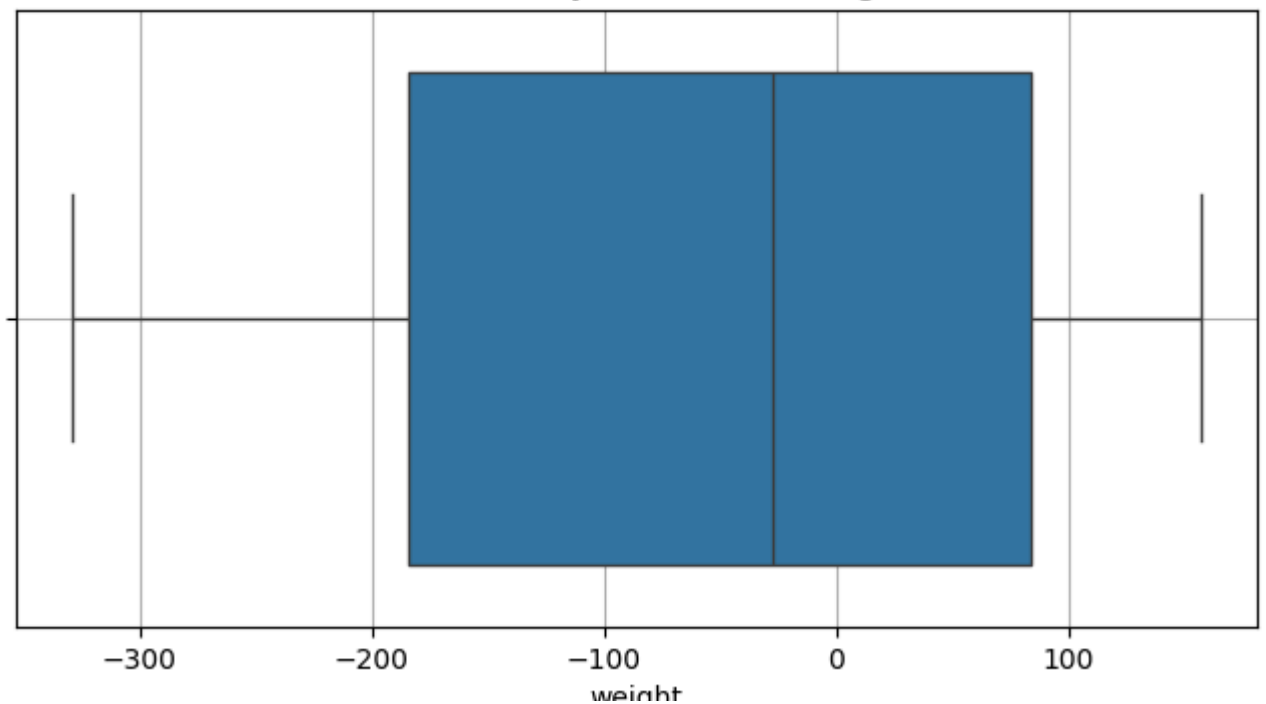




Distribución y outliers de: oxygen_saturation

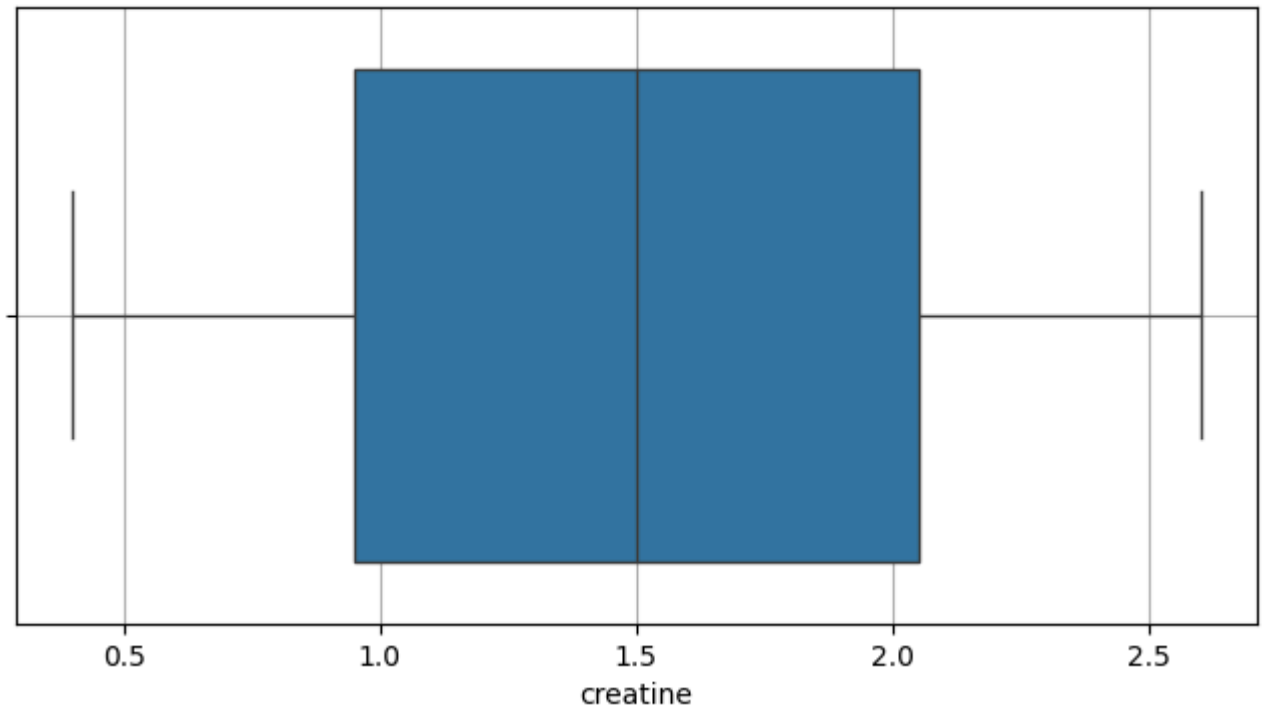


Distribución y outliers de: weight

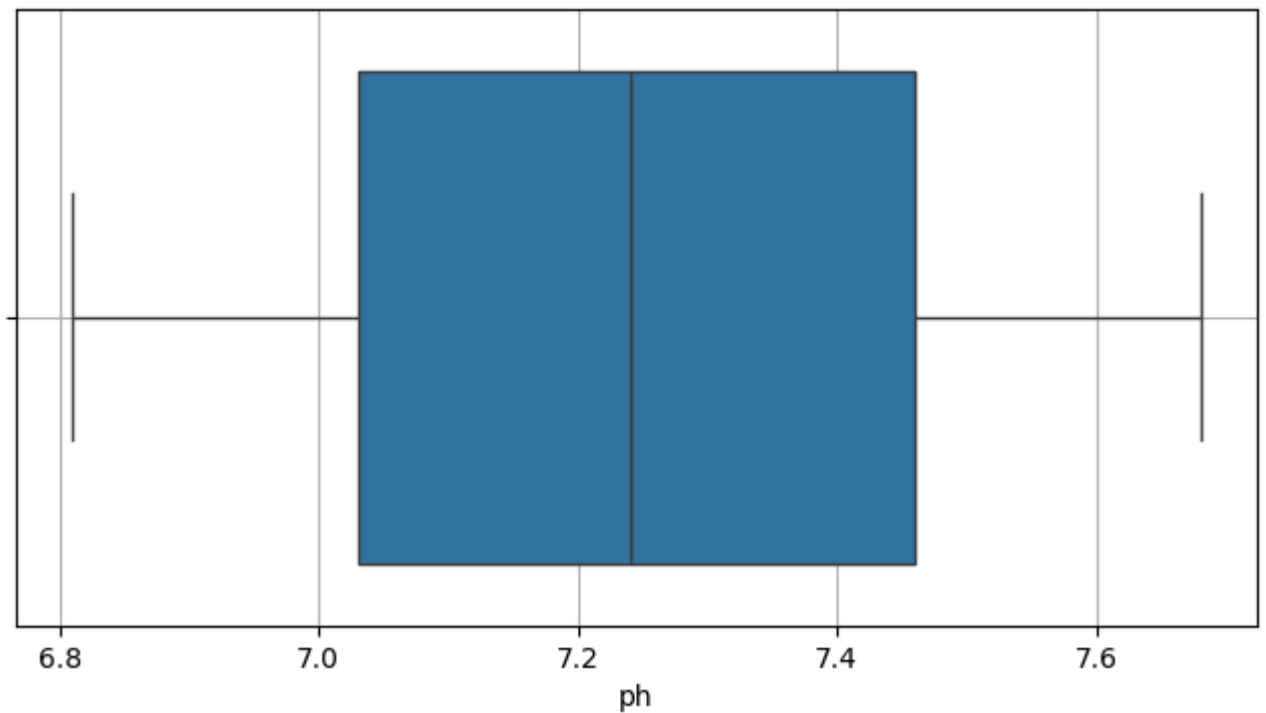


weight

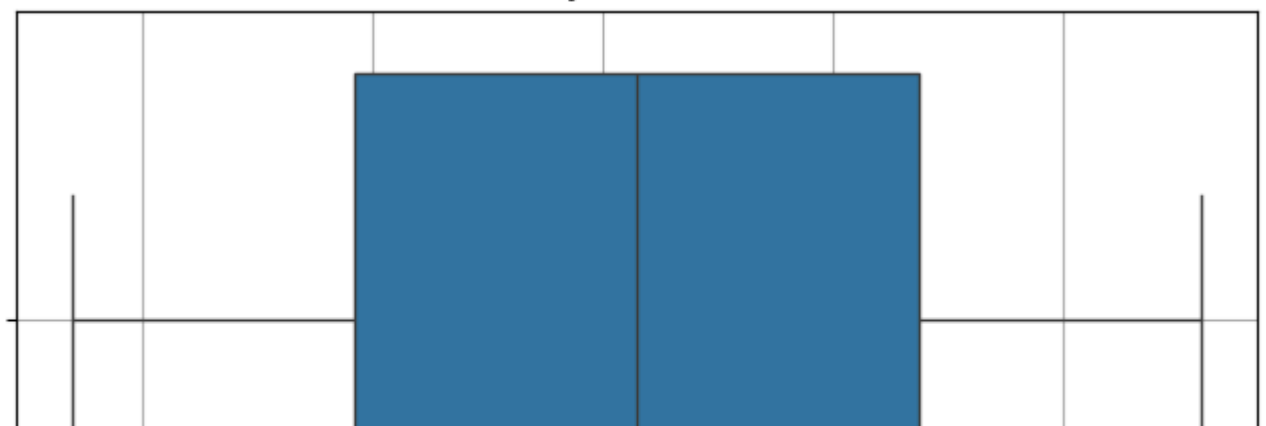
Distribución y outliers de: creatine

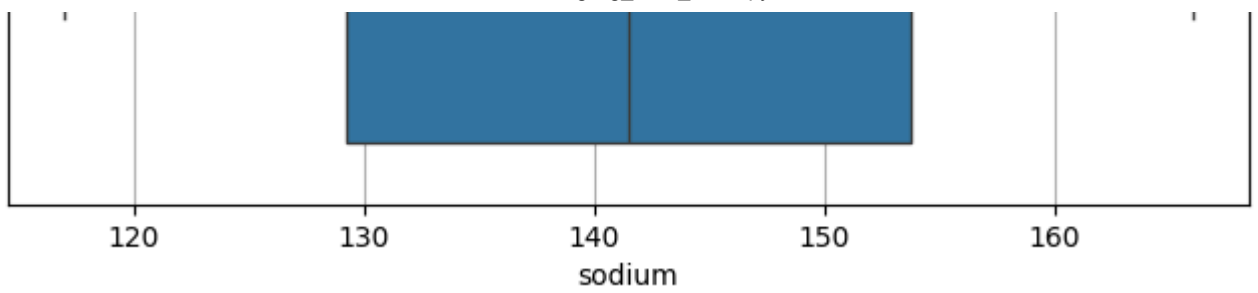


Distribución y outliers de: ph

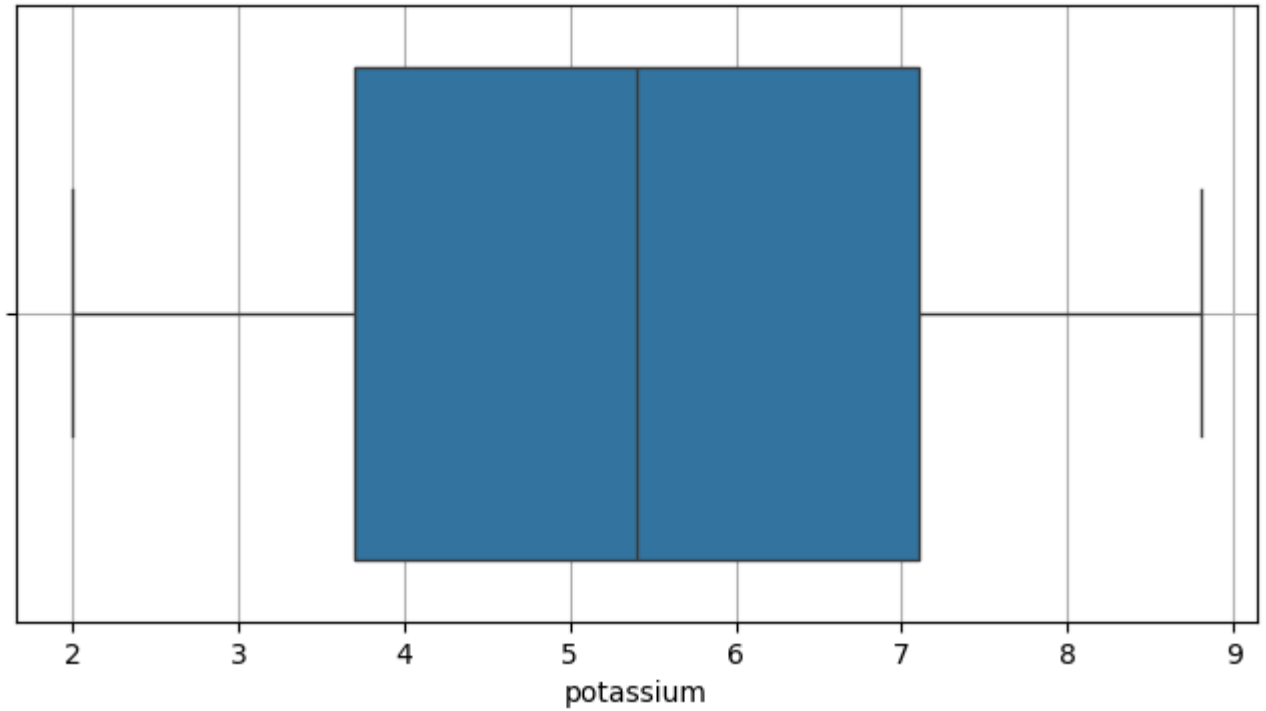


Distribución y outliers de: sodium

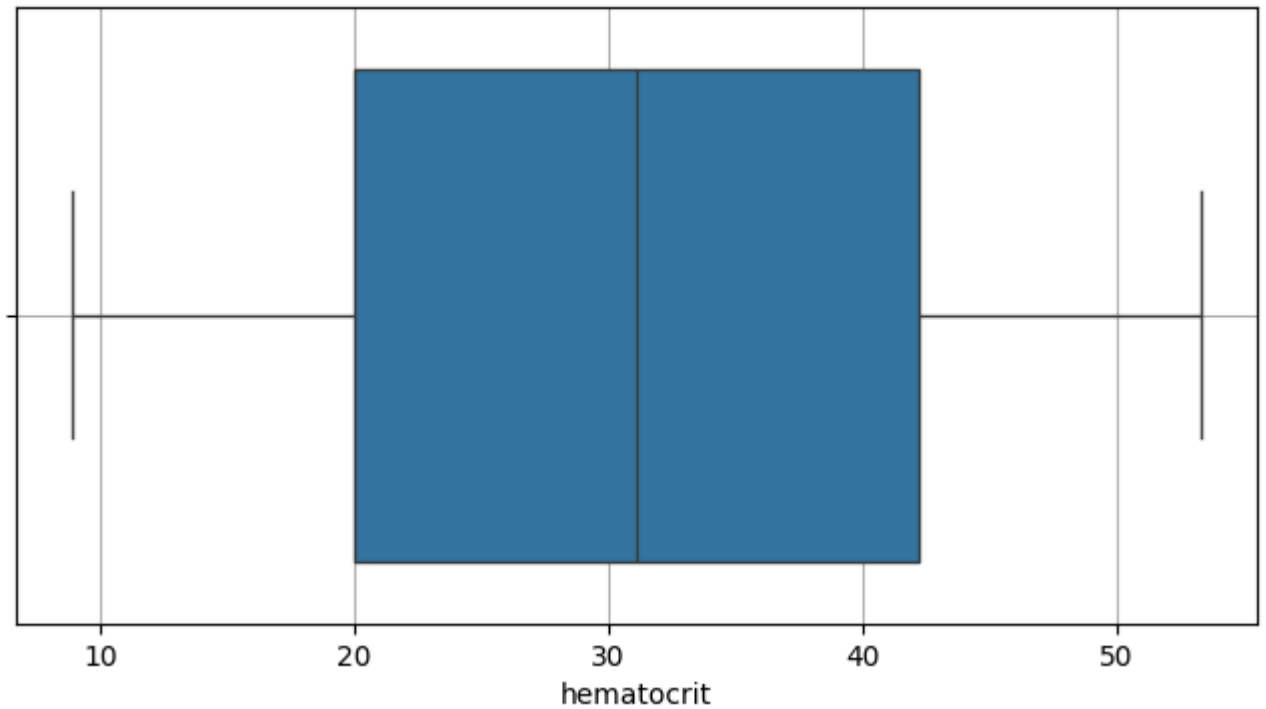




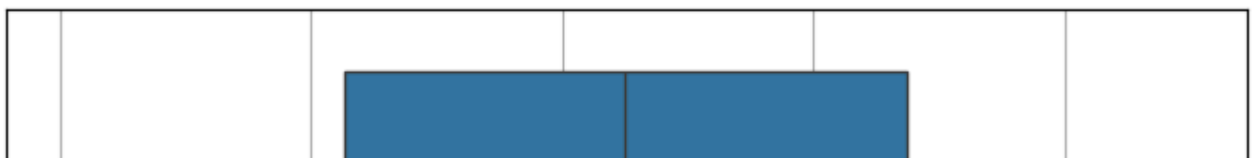
Distribución y outliers de: potassium

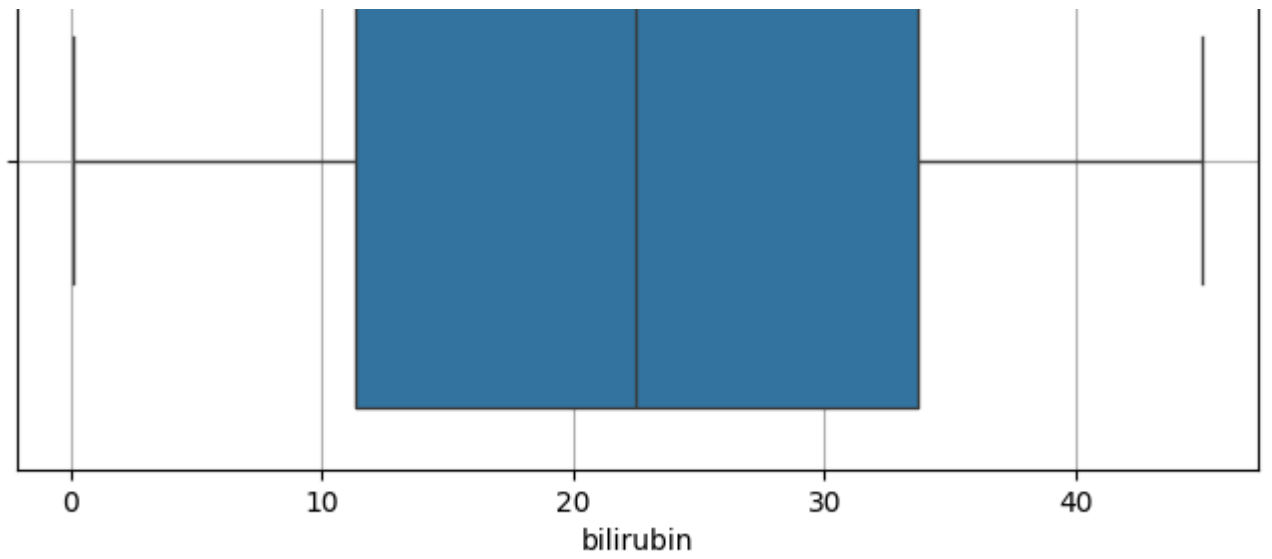


Distribución y outliers de: hematocrit



Distribución y outliers de: bilirubin





Se puede determinar que:

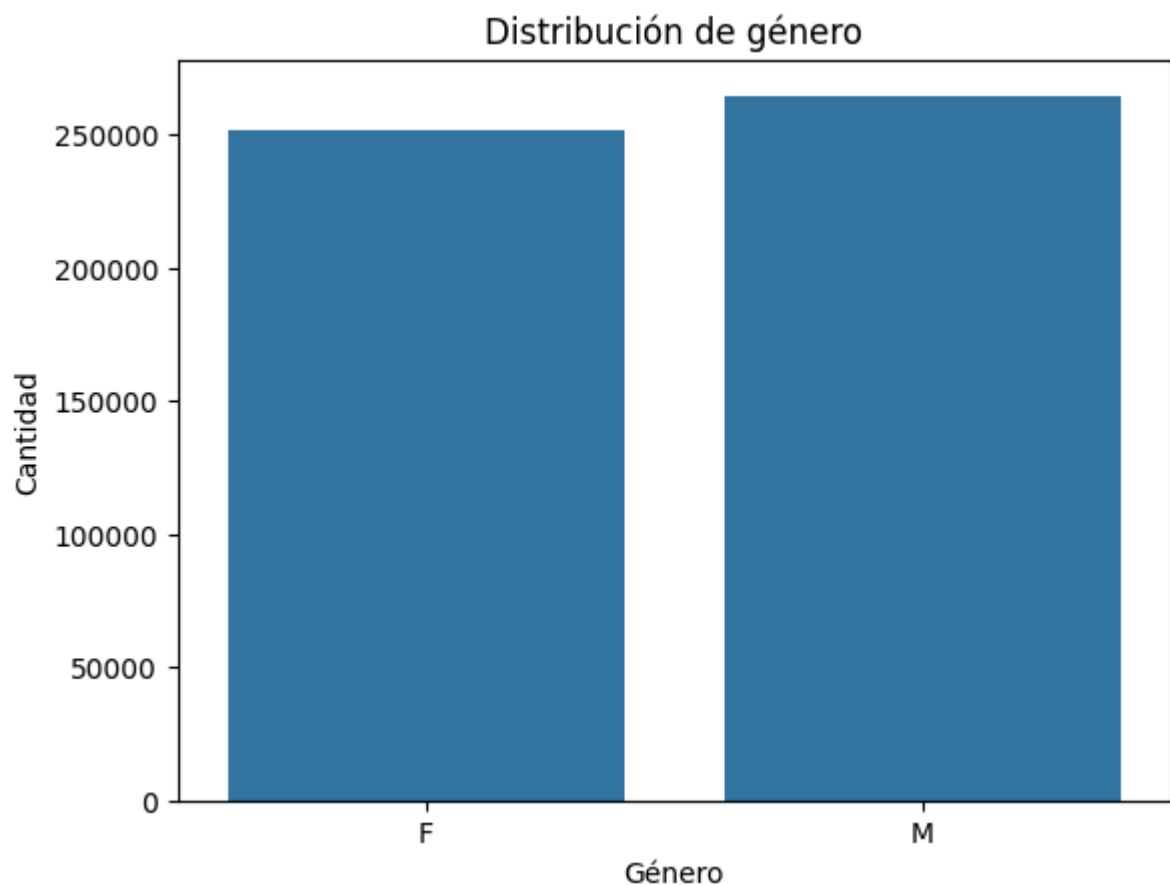
- No hay Outliers

✓ Paso 3: Visualización

✓ Variables categóricas

```
import seaborn as sns
import matplotlib.pyplot as plt

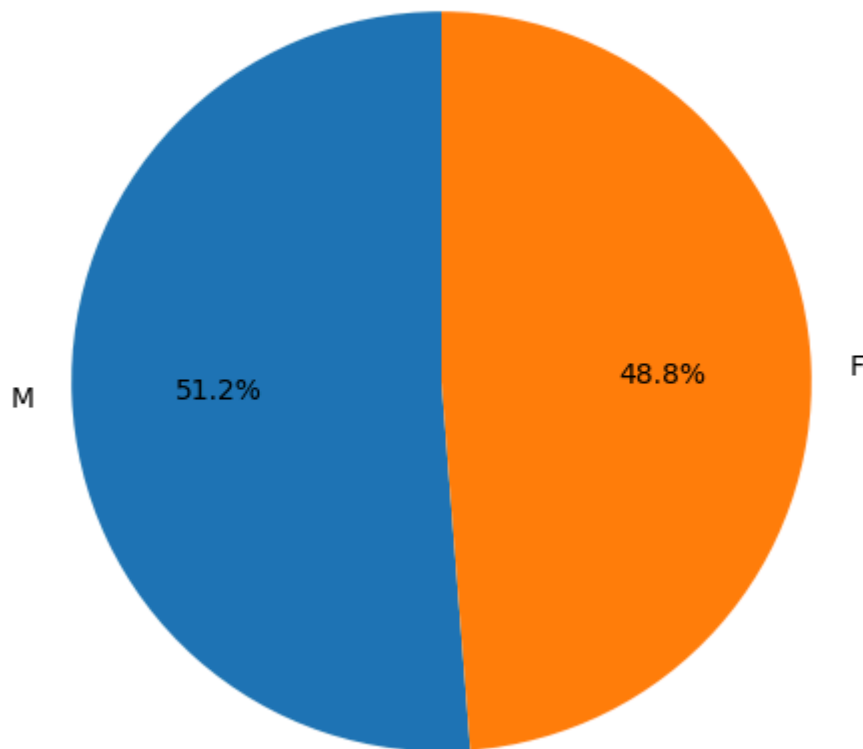
sns.countplot(x='gender', data=data_clinic)
plt.title('Distribución de género')
plt.xlabel('Género')
plt.ylabel('Cantidad')
plt.show()
```



```
data_clinic['gender'].value_counts().plot.pie(autopct='%1.1f%%', startangle=90, figsize=(
plt.title('Proporción por género')
plt.ylabel('')
plt.show())
```

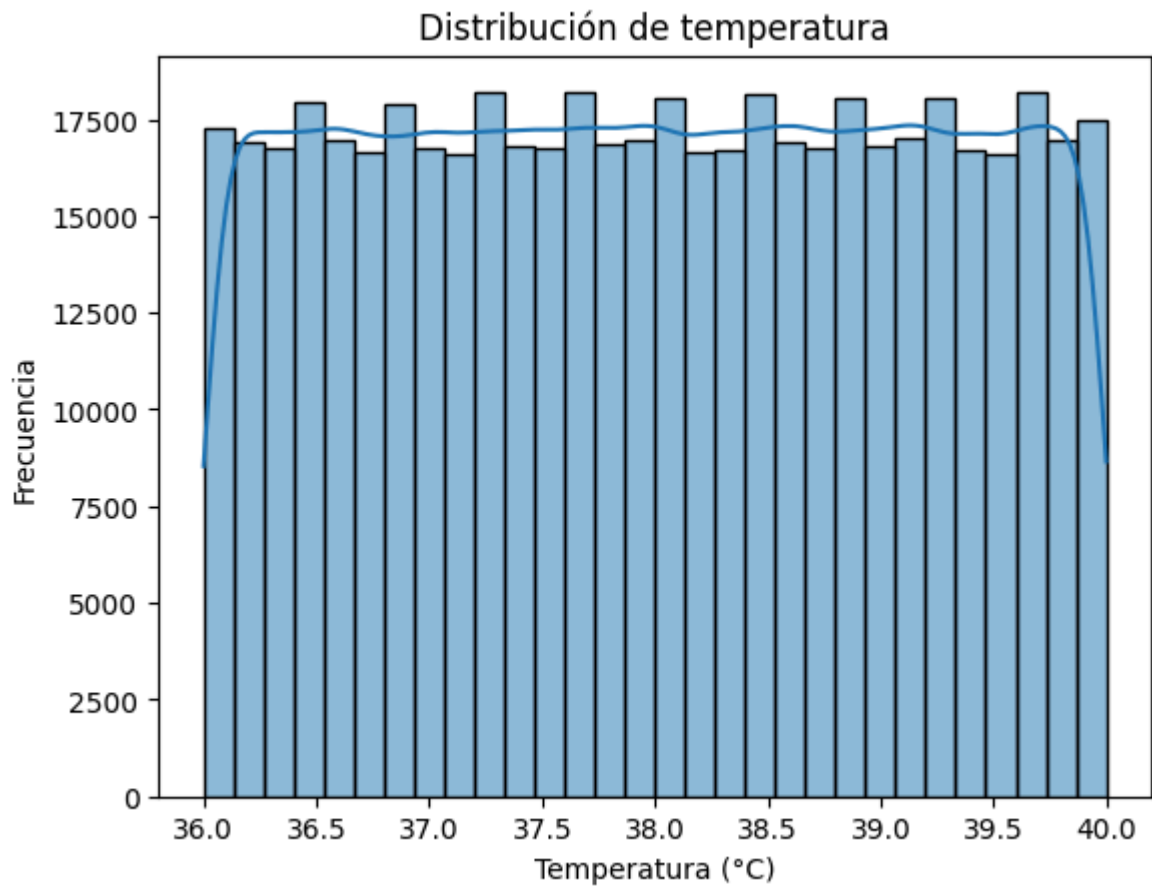


Proporción por género



✓ Variables numéricas

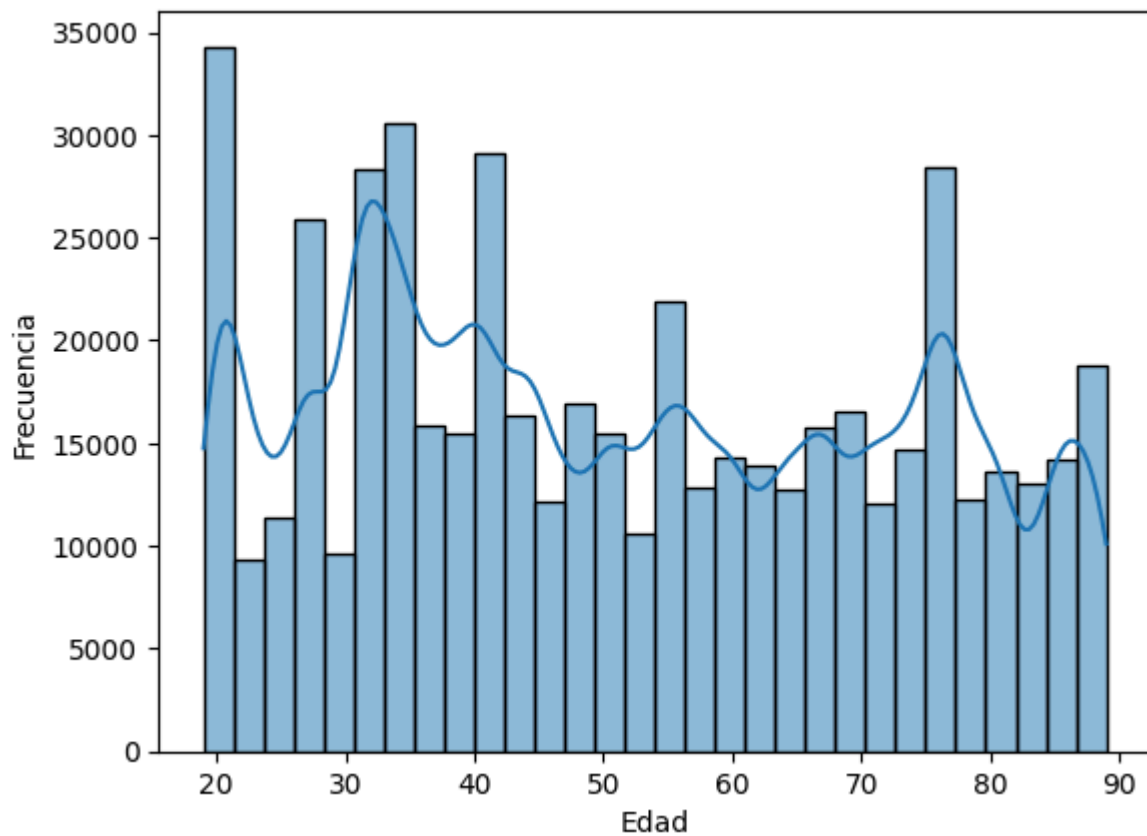
```
sns.histplot(data_clinic['temperature'], bins=30, kde=True)
plt.title('Distribución de temperatura')
plt.xlabel('Temperatura (°C)')
plt.ylabel('Frecuencia')
plt.show()
```



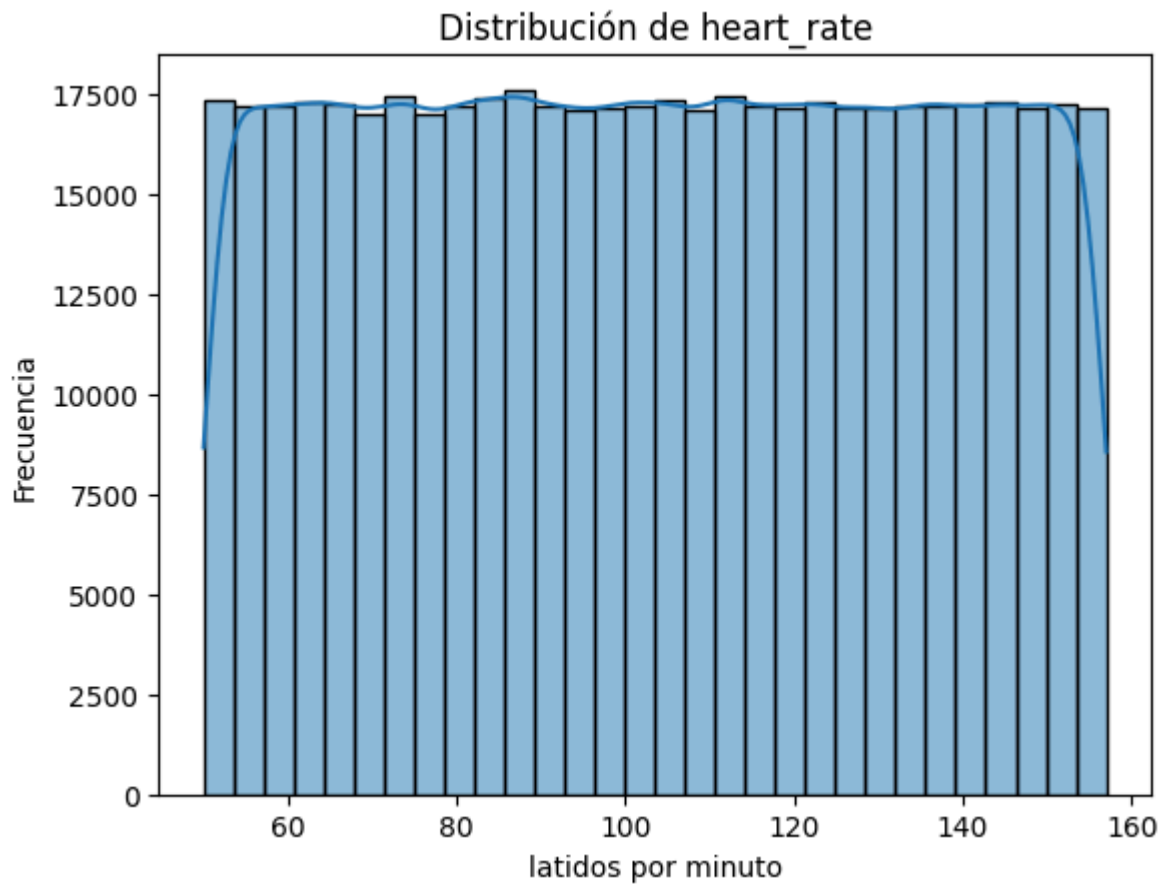
```
sns.histplot(data_clinic['age'], bins=30, kde=True)
plt.title('Distribución de edad')
plt.xlabel('Edad')
plt.ylabel('Frecuencia')
plt.show()
```



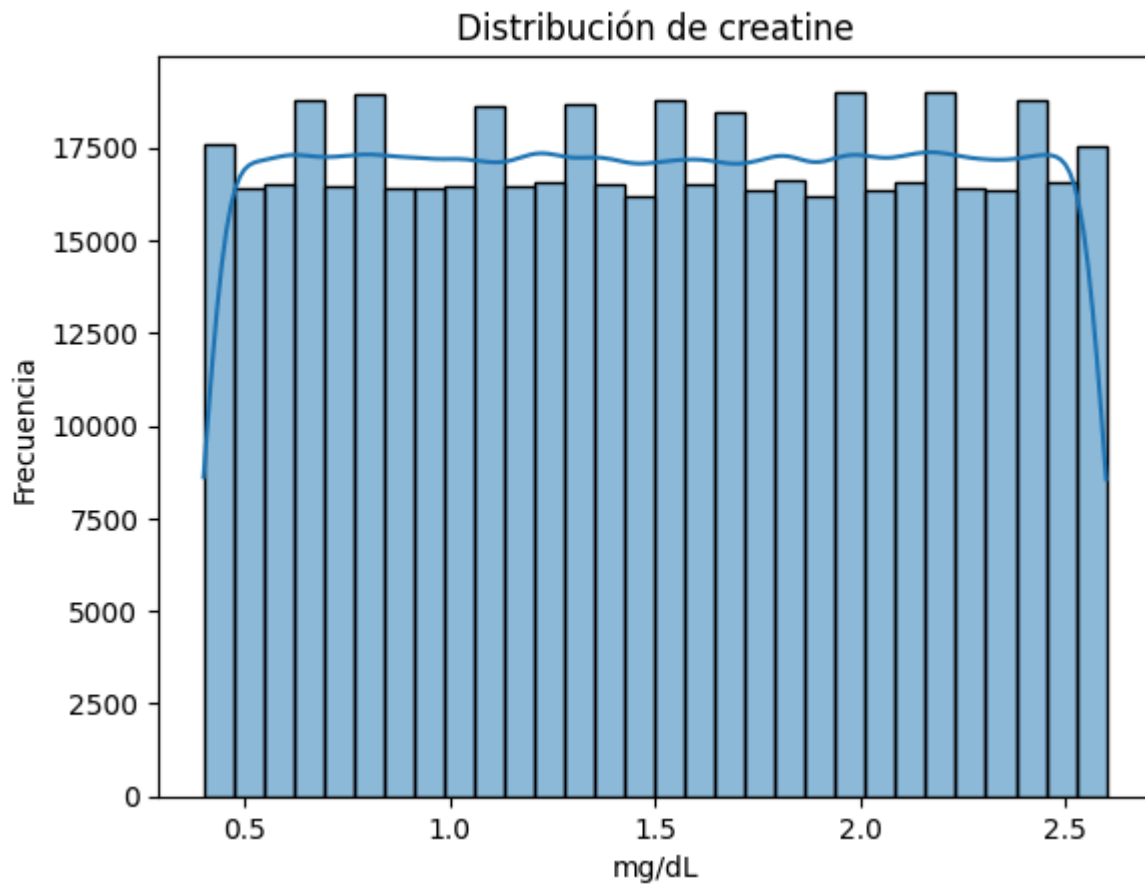

Distribución de edad



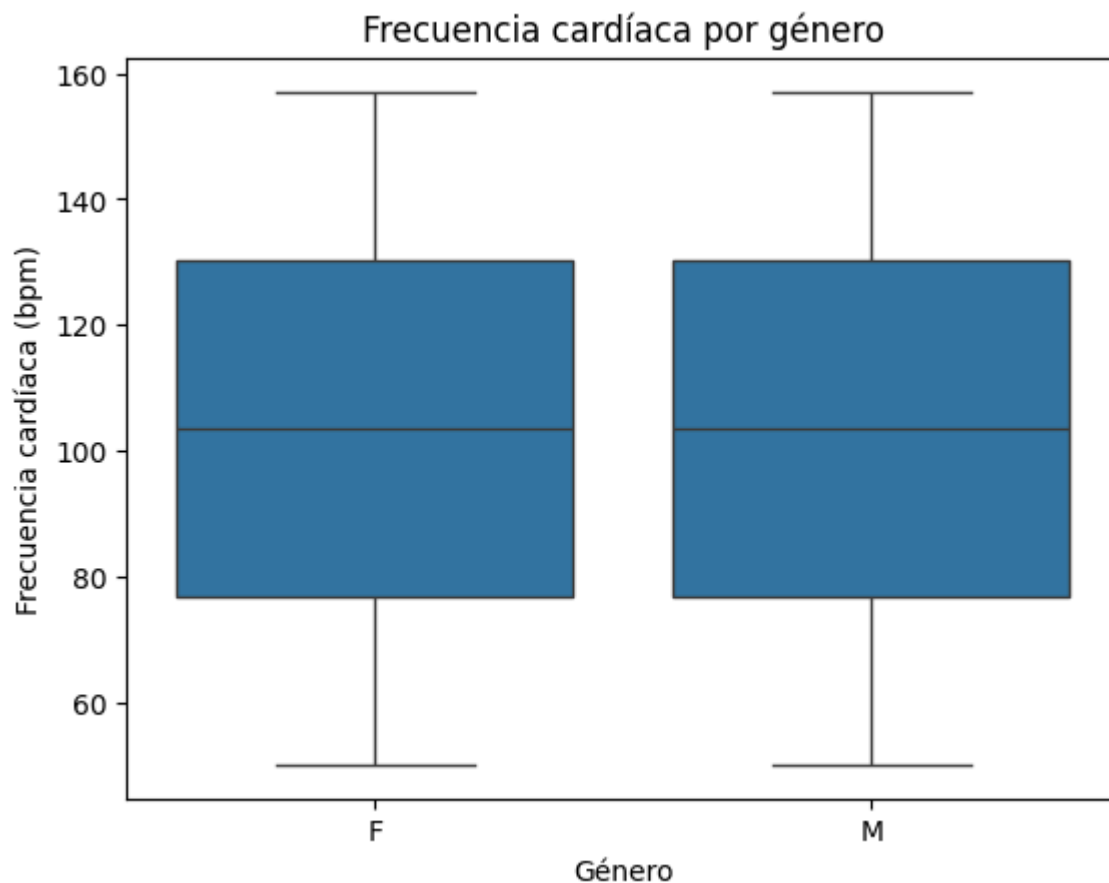
```
sns.histplot(data_clinic['heart_rate'], bins=30, kde=True)
plt.title('Distribución de heart_rate')
plt.xlabel('latidos por minuto')
plt.ylabel('Frecuencia')
plt.show()
```



```
sns.histplot(data_clinic['heart_rate'], bins=30, kde=True)
plt.title('Distribución de heart_rate')
plt.xlabel('latidos por minuto')
plt.ylabel('Frecuencia')
plt.show()
```



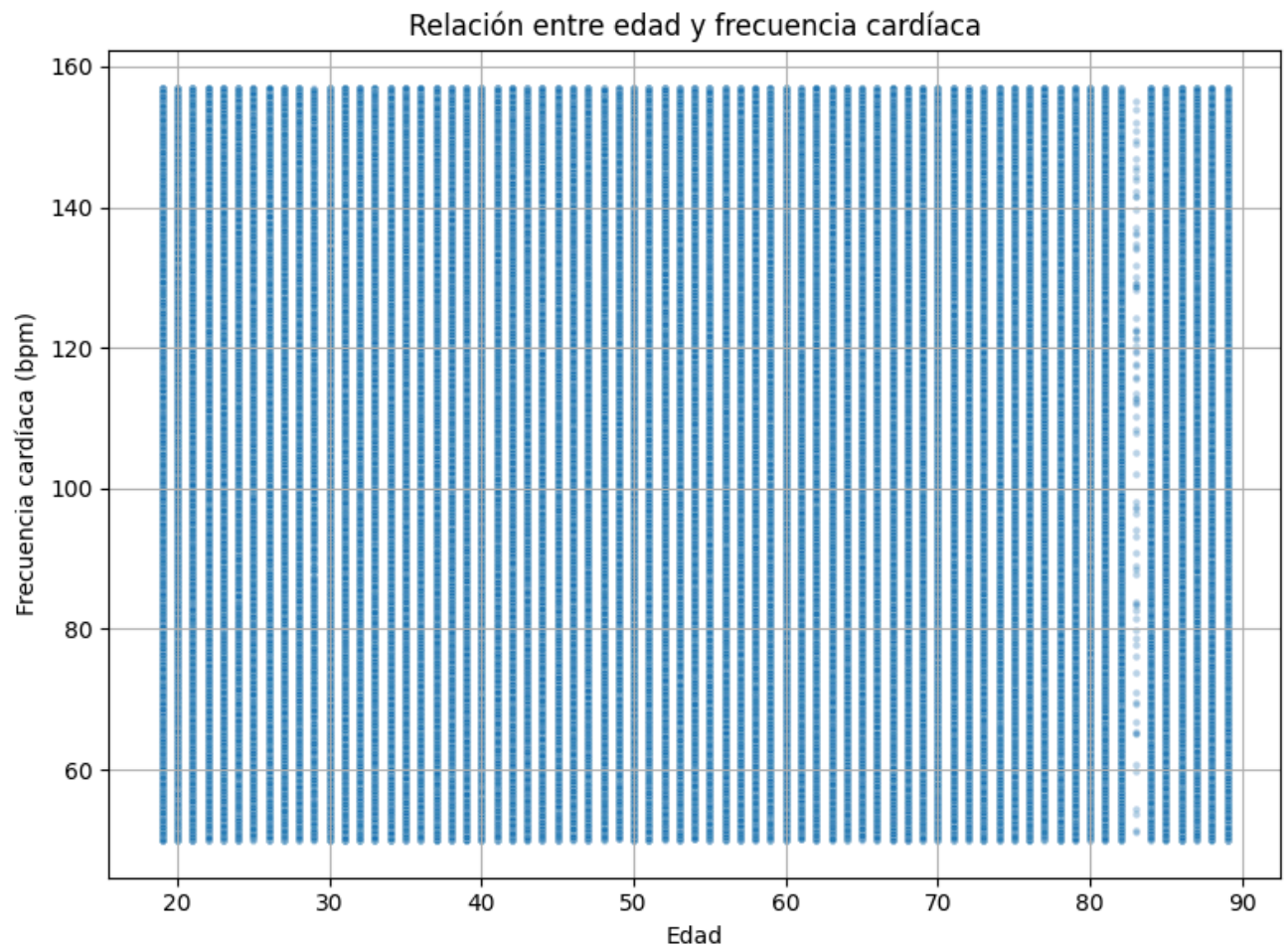
```
sns.boxplot(x='gender', y='heart_rate', data=data_clinic)
plt.title('Frecuencia cardíaca por género')
plt.xlabel('Género')
plt.ylabel('Frecuencia cardíaca (bpm)')
plt.show()
```



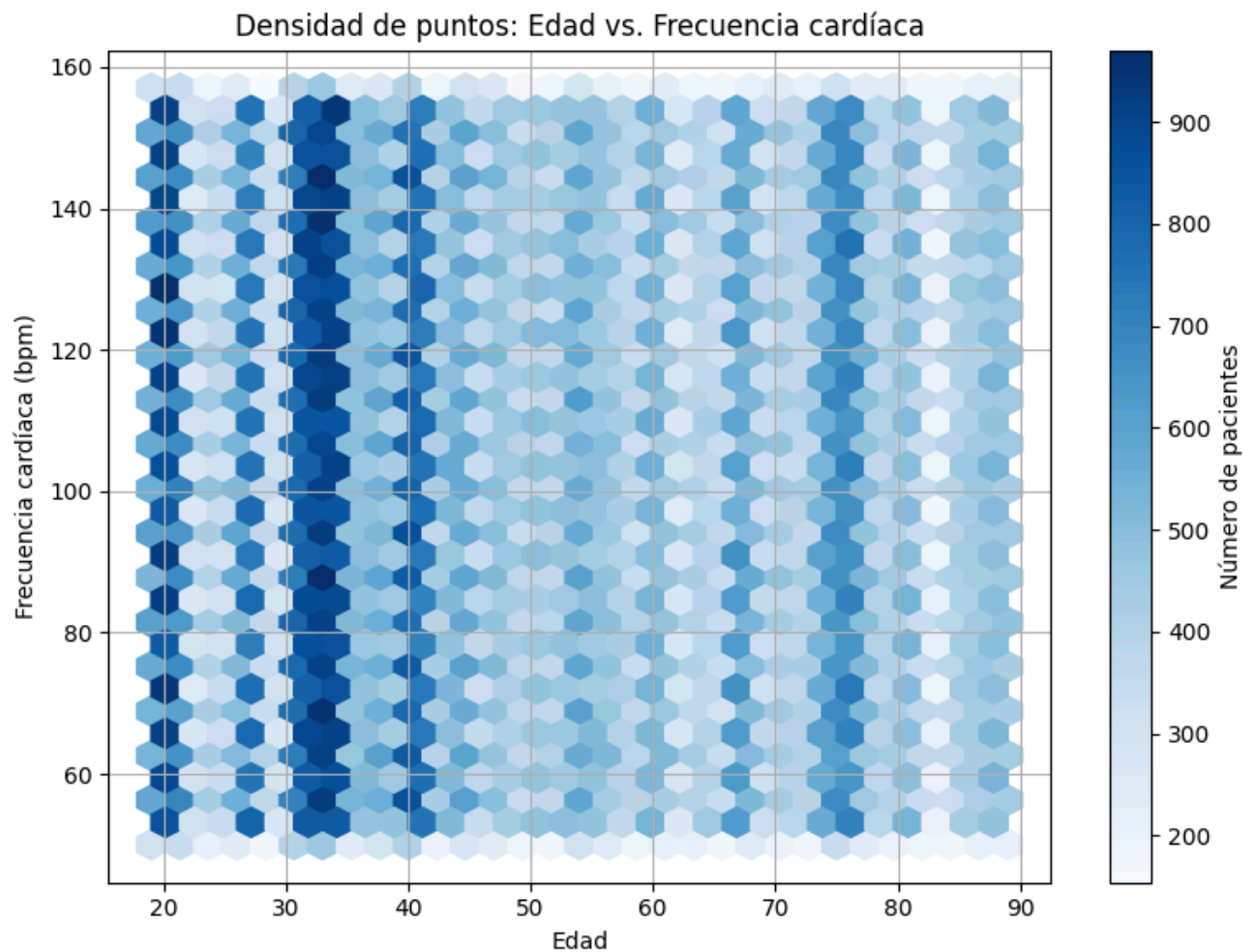
✓ Relación entre dos variables numéricas

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(8, 6))
sns.scatterplot(data=data_clinic, x='age', y='heart_rate', s=10, alpha=0.3)
plt.title('Relación entre edad y frecuencia cardíaca')
plt.xlabel('Edad')
plt.ylabel('Frecuencia cardíaca (bpm)')
plt.grid(True)
plt.tight_layout()
plt.show()
```



```
plt.figure(figsize=(8, 6))
plt.hexbin(data_clinic['age'], data_clinic['heart_rate'], gridsize=30, cmap='Blues', minc
plt.colorbar(label='Número de pacientes')
plt.title('Densidad de puntos: Edad vs. Frecuencia cardíaca')
plt.xlabel('Edad')
plt.ylabel('Frecuencia cardíaca (bpm)')
plt.grid(True)
plt.tight_layout()
plt.show()
```



✓ Heatmap de correlación

```
import numpy as np

plt.figure(figsize=(12, 10))
correlation = data_clinic.corr(numeric_only=True)

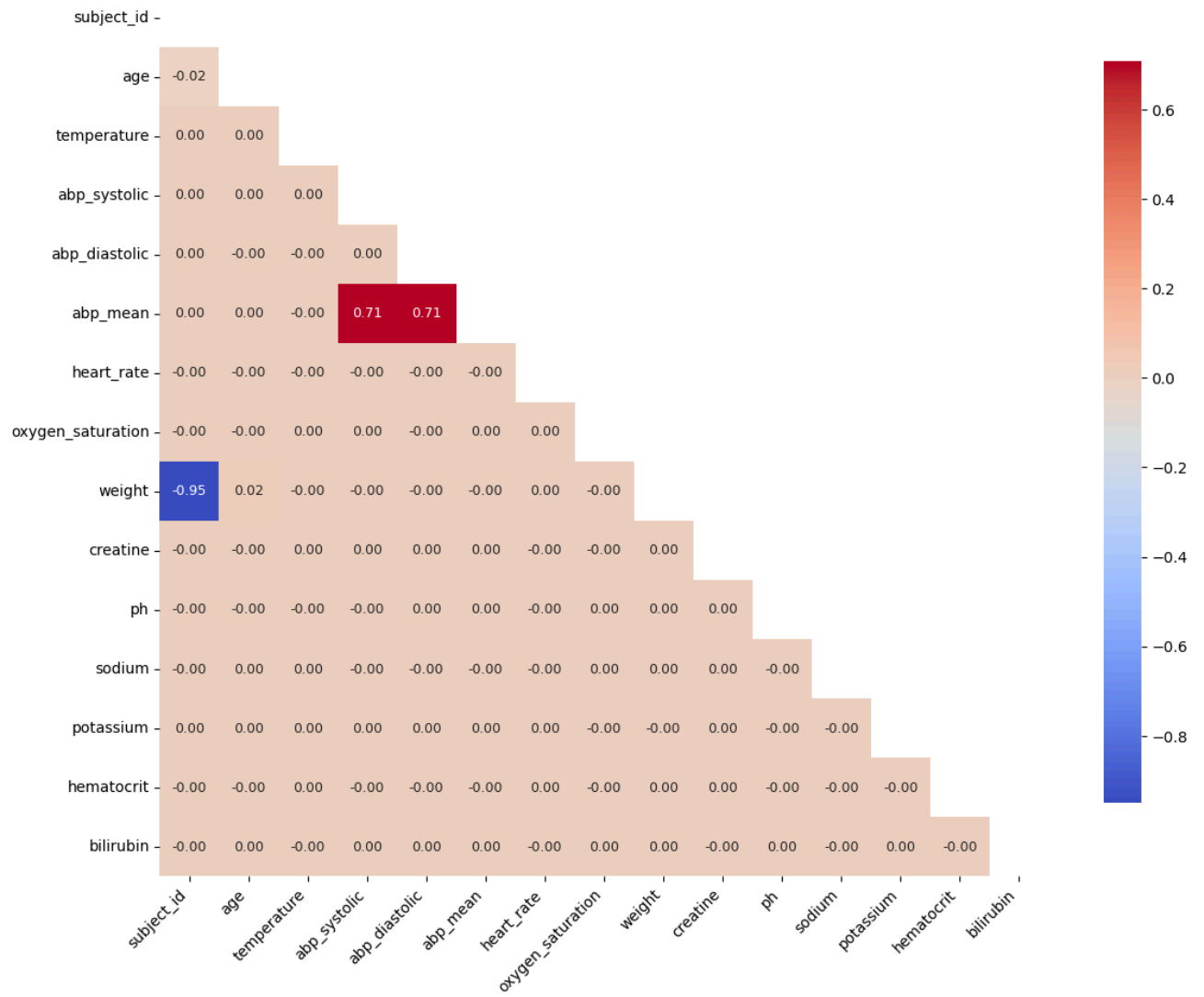
# Crear una máscara para ocultar la mitad superior
mask = np.triu(np.ones_like(correlation, dtype=bool))

sns.heatmap(
    correlation,
    mask=mask,
    annot=True,
    fmt=".2f",
    cmap='coolwarm',
    square=True,
    cbar_kws={"shrink": 0.8},
    annot_kws={"size": 9}
)
```

```
plt.title('Mapa de calor de correlación entre variables numéricas', fontsize=14)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.show()
```



Mapa de calor de correlación entre variables numéricas



✓ Paso 4: Encuentra un problema potencial en tus datos

✓ ¿Cuál es la columna de salida? ¿Es binaria o multiclase?

Aun no se tiene la columna de salida, pero se determinara tras entrenar el modelo con esta data. Será multiclase, ya que al ser binario solo hay dos opciones, se podria devolver el porcentaje para si un paciente necesita ser readmitido o no. Si un valor domina demasiado (por ejemplo, 95% "no readmitido"), el dataset está desbalanceado y necesitarás técnicas como:

- Submuestreo (undersampling).
- Sobremuestreo (SMOTE).
- Ponderar clases en el modelo.

✓ Identifica el objetivo del análisis

¿Qué quieres que tu modelo prediga?

- Si el paciente será readmitido a UCI.
- Si un paciente está estable o crítico.
- Si sobrevivirá 30 días después de salir de UCI.
- Si el tratamiento fue exitoso o fallido.

✓ ¿Tienes variables temporales?

Se tiene columna date, que nos indica el registro de un paciente según su estadia en UCI

```
readmit_count = data_clinic['subject_id'].value_counts()  
data_clinic['readmitted'] = data_clinic['subject_id'].map(lambda x: 1 if readmit_count[x]
```

```
data_clinic.head()
```



	subject_id	date	time	age	gender	temperature	abp_systolic	abp_diastolic
0	1000	2001-04-05	00:11:00	57	F	39.17	106.10	37.94
1	1000	2001-04-05	04:26:00	57	F	38.18	138.53	72.41
2	1000	2001-04-05	06:25:00	57	F	36.15	76.05	53.58
3	1000	2001-04-05	06:46:00	57	F	36.76	132.29	33.54
4	1000	2001-04-05	10:51:00	57	F	39.87	152.82	71.14

✓ Problema calidad de datos

Se observo valores negativos en la columna weight, por ello se hace un seguimiento

✓ Filtrar valores de peso negativos

```
pesos_negativos = data_clinic[data_clinic['weight'] < 0]
print(pesos_negativos[['subject_id', 'date', 'time', 'weight']])
```



	subject_id	date	time	weight
222619	1209	2005-06-14	08:44:00	-2
222620	1209	2005-06-14	12:19:00	-2
222621	1209	2005-06-14	14:03:00	-2
222622	1209	2005-06-14	14:36:00	-2
222623	1209	2005-06-14	15:15:00	-2
...
516408	1499	2006-12-17	06:26:00	-295
516409	1499	2006-12-17	06:40:00	-295
516410	1499	2006-12-17	11:02:00	-295
516411	1499	2006-12-17	13:35:00	-295
516412	1499	2006-12-17	16:37:00	-295

[281616 rows x 4 columns]

Visualizar la evolución del peso por paciente

Graficar el peso en el tiempo por paciente

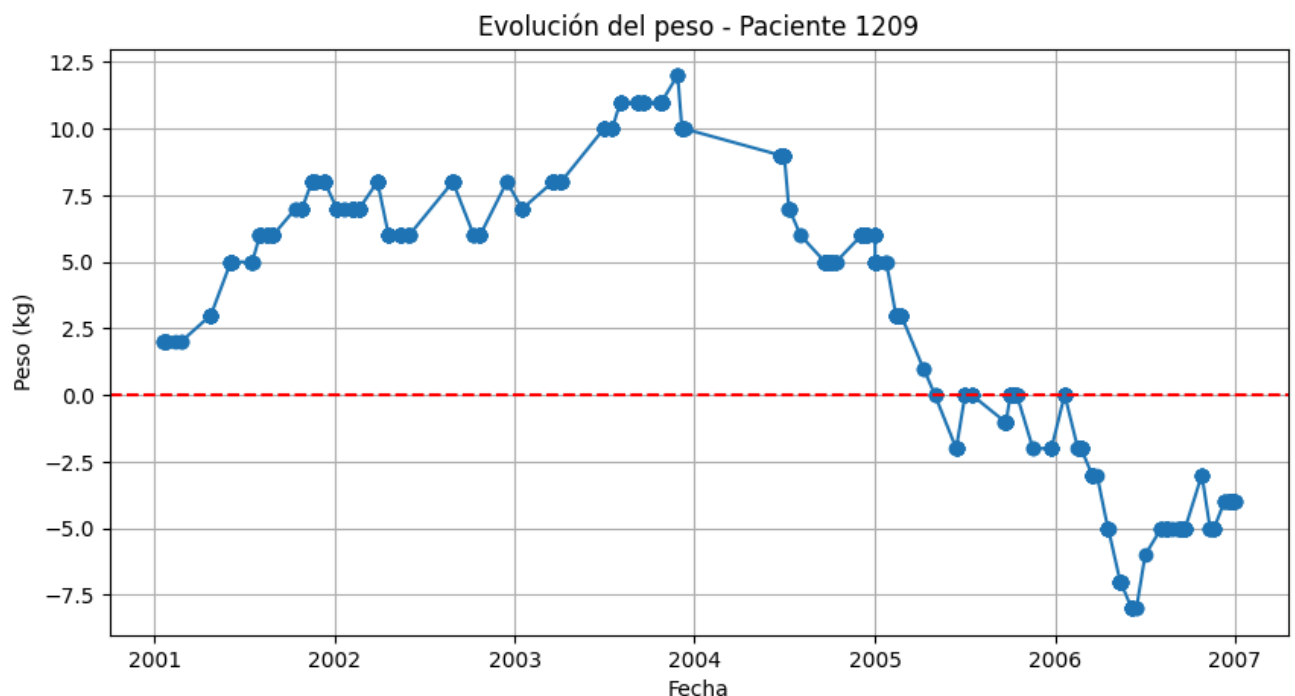
```
import matplotlib.pyplot as plt
```

```
# Seleccionar un paciente con pesos negativos
id_con_peso_negativo = pesos_negativos['subject_id'].unique()[0] # El primero
```

```
# Filtrar datos del paciente
paciente = data_clinic[data_clinic['subject_id'] == id_con_peso_negativo].copy()
paciente['datetime'] = pd.to_datetime(paciente['date'] + ' ' + paciente['time'])

# Ordenar por tiempo
paciente = paciente.sort_values('datetime')

# Graficar evolución del peso
plt.figure(figsize=(10, 5))
plt.plot(paciente['datetime'], paciente['weight'], marker='o')
plt.title(f'Evolución del peso - Paciente {id_con_peso_negativo}')
plt.xlabel('Fecha')
plt.ylabel('Peso (kg)')
plt.axhline(0, color='red', linestyle='--') # línea de referencia en 0
plt.grid()
plt.show()
```



```
import matplotlib.pyplot as plt

# Seleccionar un paciente con pesos negativos
id_con_peso_negativo = pesos_negativos['subject_id'].unique()[1] # El primero

# Filtrar datos del paciente
paciente = data_clinic[data_clinic['subject_id'] == id_con_peso_negativo].copy()
paciente['datetime'] = pd.to_datetime(paciente['date'] + ' ' + paciente['time'])
```