

# PRA 2: ¿Cómo realizar la limpieza y análisis de datos?

**Nombre y apellidos:** José Miguel Gamarro Tornay

Para la elaboración de este trabajo, se utilizará el lenguaje de programación Python, como herramienta que permita apoyarnos en ella para automatizar procesos de limpieza y de análisis.

Para ello se importarán las siguientes librerías:

```
In [ ]: import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import shapiro, wilcoxon, kruskal, ttest_ind
```

## 1. Descripción del dataset.

Para el presente trabajo académico, el dataset elegido es el que se indica en la documentación de la práctica: "Heart Attack Analysis & Prediction dataset".

Este dataset contiene dos csv, en el principal, denominado "heart.csv", se reflejan para un total de 303 pacientes, una serie de información médica y personal (año y sexo) junto a si tiene probabilidad mayor de infarto o no.

Se establece como objetivo, el análisis de la posible relación entre diferentes parámetros médicos (presión arterial en reposo, colesterol, frecuencia máxima alcanzada, glucemia en ayunas y los resultados electrocardiográficos) y la correlación que estos existen con mayor probabilidad de infarto.

Este análisis puede servir para que médicos puedan detectar de una forma sencilla si un paciente tiene más o menos probabilidad de sufrir un infarto según los indicadores.

```
In [ ]: heart_dataset = pd.read_csv("../dataset/heart.csv")
heart_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
--- 
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trtbps    303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalachh  303 non-null    int64  
 8   exng     303 non-null    int64  
 9   oldpeak   303 non-null    float64 
 10  slp       303 non-null    int64  
 11  caa       303 non-null    int64  
 12  thall     303 non-null    int64  
 13  output    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

## 2. Integración y selección

Integración y selección de los datos de interés a analizar. Puede ser el resultado de adicionar diferentes datasets o una subselección útil de los datos originales, en base al objetivo que se quiera conseguir.

Para el objetivo del problema, vamos a seleccionar un subconjunto de datos según los factores más relevantes para el objeto del estudio:

- Tipo de dolor torácico (cp)
  - Valor 1: angina típica
  - Valor 2: angina atípica
  - Valor 3: dolor no anginoso
  - Valor 4: asintomático
- Presión arterial en reposo en mmHg (trtbps)
- Colesterol en mg/dl (chol)
- Glucemia en ayunas mayor a 120 mg/dl (fbs)
- Resultados electrocardiográficos en reposo (restecg)
  - Valor 0: normal.
  - Valor 1: con anomalía de la onda ST-T (inversiones de la onda T y/o elevación o depresión del ST de > 0,05 mV).
  - Valor 2: hipertrofia ventricular izquierda probable o definida.
- Frecuencia cardiaca máxima alcanzada (thalachh)
- Probabilidad de infarto (output)

```
In [ ]: chest_pain = "cp"
resting_blood_pressure = "trtbps"
```

```

cholestorol = "chol"
fasting_blood_sugar = "fbs"
resting_electrocardiographic_results = "restecg"
maximum_heart_rate = "thalachh"
chance_heart_attack = "output"

heart_dataset = heart_dataset[
    [chest_pain,
     resting_blood_pressure,
     cholestorol,
     fasting_blood_sugar,
     resting_electrocardiographic_results,
     maximum_heart_rate,
     chance_heart_attack]]

print(heart_dataset.head())

```

	cp	trtbps	chol	fbs	restecg	thalachh	output
0	3	145	233	1	0	150	1
1	2	130	250	0	1	187	1
2	1	130	204	0	0	172	1
3	1	120	236	0	1	178	1
4	0	120	354	0	1	163	1

Si el problema quisiese determinar si los síntomas son similares para ambos géneros, si sería interesante quedarnos con datos de género, pero para el problema planteado podemos obviar el género y edad al querer únicamente basarnos en ver la relación de datos médicos con la probabilidad de infarto.

## 3. Limpieza de los datos

### 3.1. Elementos ceros y vacíos

Primero, vamos a observar si existe algún elemento vacío o nulo. Para ello, vamos a apoyarnos de la función `isnull` presente en los objetos DataFrame de Python, tras ejecutarlo, se puede observar que para este data set no hay ningún elemento nulo.

```
In [ ]: # Contar los valores nulos en cada columna
print(heart_dataset.isnull().sum())
```

```

cp          0
trtbps      0
chol         0
fbs          0
restecg      0
thalachh     0
output        0
dtype: int64

```

Por lo tanto, los valores nulos no requieren de ninguna acción correctiva sobre los que existiese como podría ser re-calcularlos según los valores vecinos o directamente eliminarlos.

Por otro lado, vamos a estudiar si hay elementos ceros entre las columnas que hemos seleccionado. Como el tipo de dolor torácico, la glucemia en ayunas mayor a 120 mg/dl, los resultados electrocardiográficos en reposo y el resultado son en realidad valores categóricos en los que algunos de sus estados pueden ser 0, no tiene sentido revisar si tienen elementos ceros, así que vamos a centrarnos en el resto.

```
In [ ]: selected_data = heart_dataset[[
    resting_blood_pressure,
    cholestorol,
    maximum_heart_rate
]]

print((selected_data == 0).sum())

trtbps      0
chol        0
thalachh    0
dtype: int64
```

Como se puede observar, no existen elementos ceros entre las variables numéricas, por lo tanto no se debe hacer ninguna acción correctora.

## 3.2. Identificar y gestionar los valores extremos

Para obtener los valores extremos, vamos a buscar todos los registros que se encuentren al menos a 3 desviaciones estándar alejados de la media.

Definiremos la siguiente función para encontrarlos.

```
In [ ]: def get_outliers(data_set, column_name, num_std):

    column_mean = data_set[column_name].mean()
    column_std = data_set[column_name].std()

    column_upper_limit = column_mean + num_std * column_std
    column_lower_limit = column_mean - num_std * column_std

    # Contar la cantidad de valores extremos en la columna
    return ((data_set[column_name] > column_upper_limit) | (data_set[column_name] <
```

```
In [ ]: resting_blood_pressure_outliers = get_outliers(heart_dataset, resting_blood_pressure)
print("Número de valores extremos de presión arterial en reposo: ", resting_blood_p

cholestorol_outliers = get_outliers(heart_dataset, cholestorol, 3)
print("Número de valores extremos de colesterol: ", cholestorol_outliers.sum())

maximum_heart_rate_outliers = get_outliers(heart_dataset, maximum_heart_rate, 3)
print("Número de valores extremos de Frecuencia cardiaca máxima alcanzada: ", maxim
```

Número de valores extremos de presión arterial en reposo: 2

Número de valores extremos de colesterol: 4

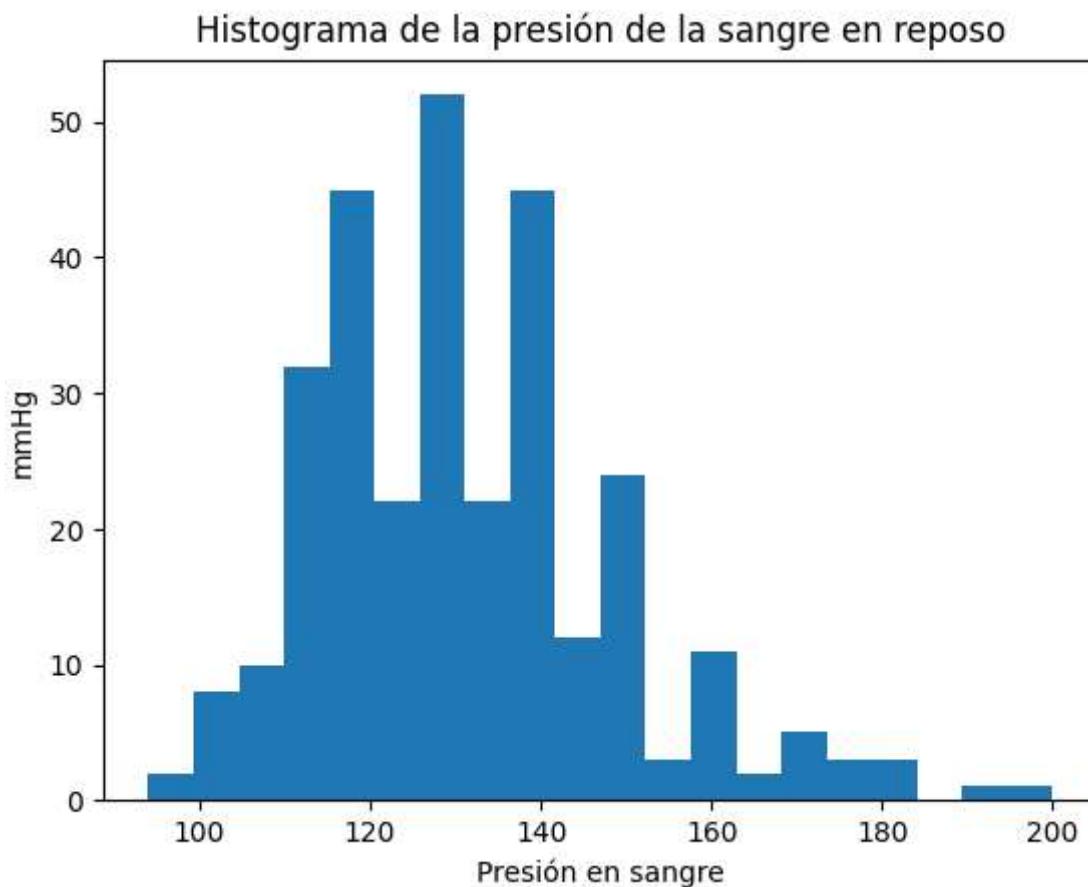
Número de valores extremos de Frecuencia cardiaca máxima alcanzada: 1

Se observa por lo tanto que existen algunos valores extremos.

```
In [ ]: def print_histogram(column, title, xlabel, ylabel):
    plt.hist(column, bins=20)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.title(title)
    plt.show()

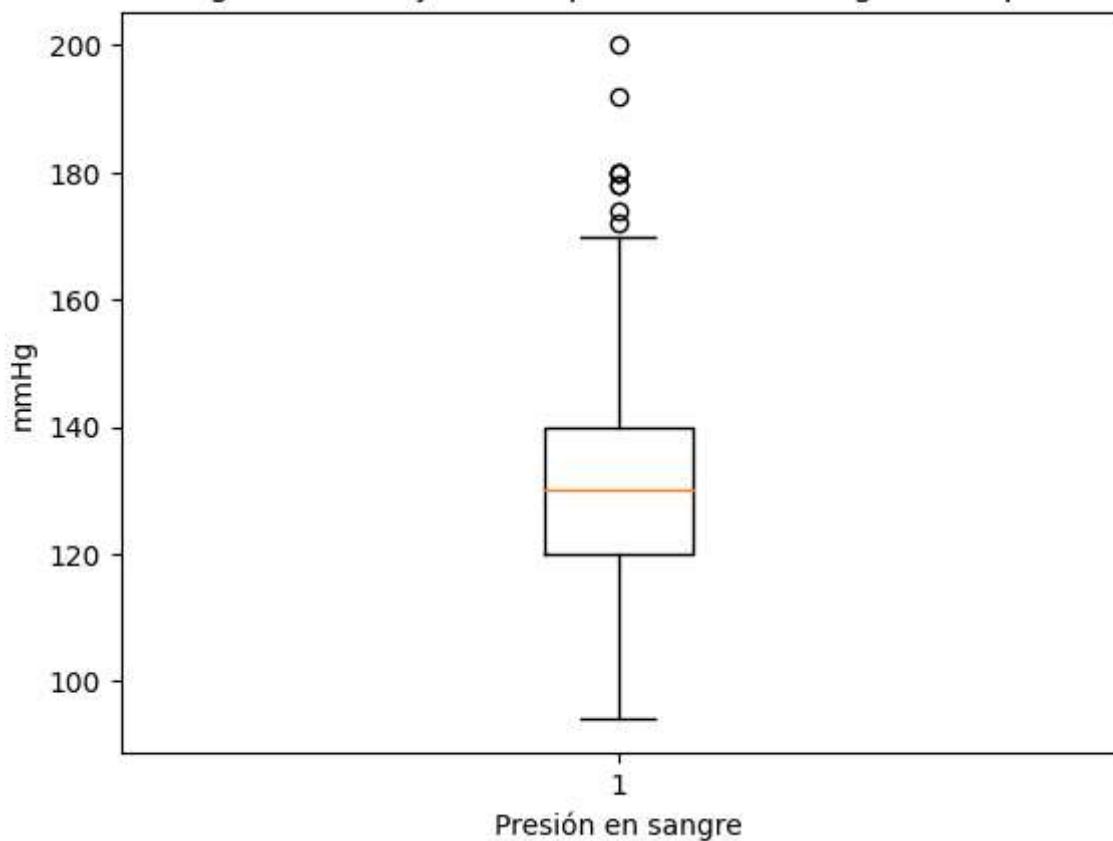
def print_boxplot(column, title, xlabel, ylabel):
    plt.boxplot(column)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.title(title)
    plt.show()
```

```
In [ ]: print_histogram(
    heart_dataset[resting_blood_pressure],
    "Histograma de la presión de la sangre en reposo",
    "Presión en sangre",
    "mmHg")
```



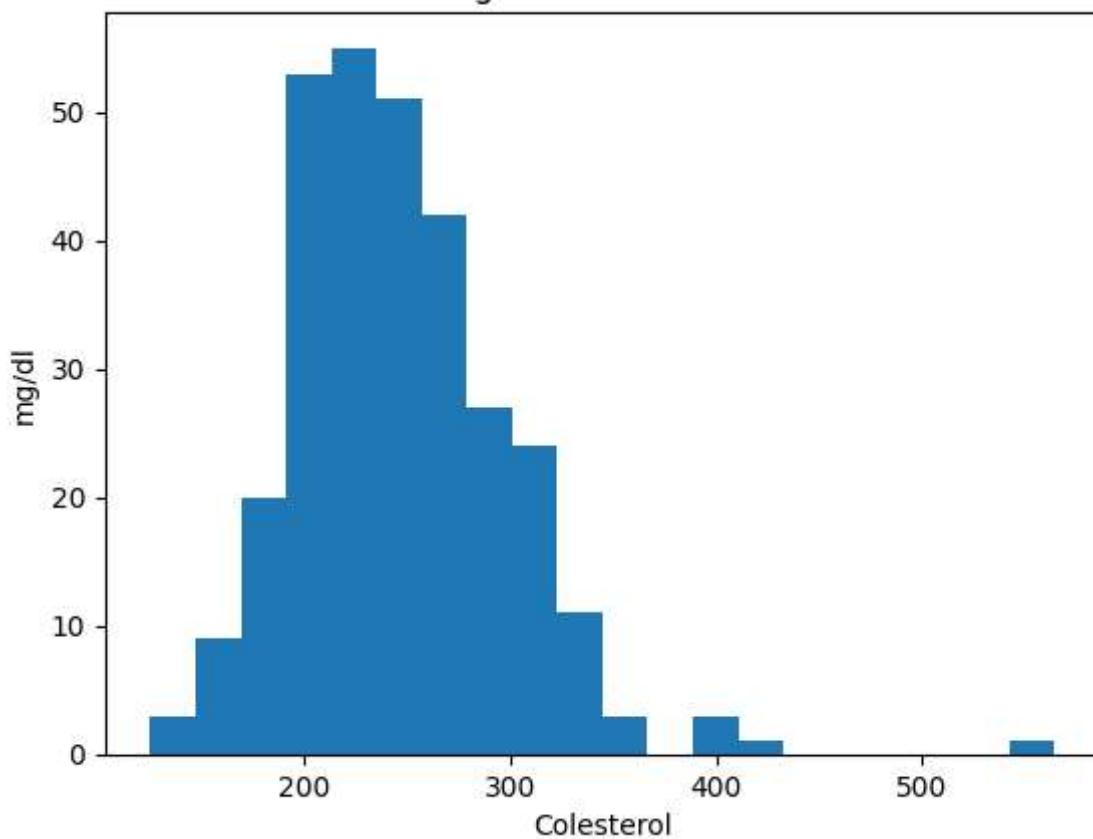
```
In [ ]: print_boxplot(
    heart_dataset[resting_blood_pressure],
    "Diagrama de cajas de la presión de la sangre en reposo",
    "Presión en sangre",
    "mmHg")
```

Diagrama de cajas de la presión de la sangre en reposo



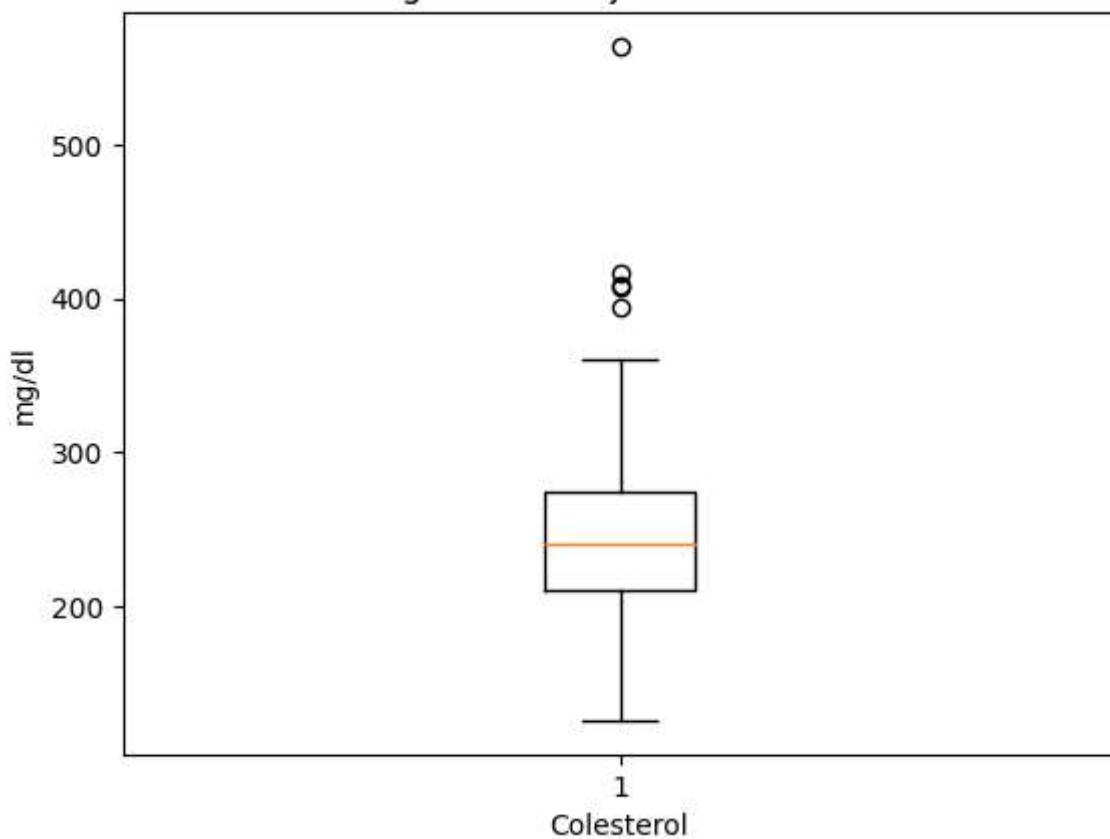
```
In [ ]: print_histogram(  
    heart_dataset[cholesterol],  
    "Histograma del colesterol",  
    "Colesterol",  
    "mg/dl")
```

### Histograma del colesterol



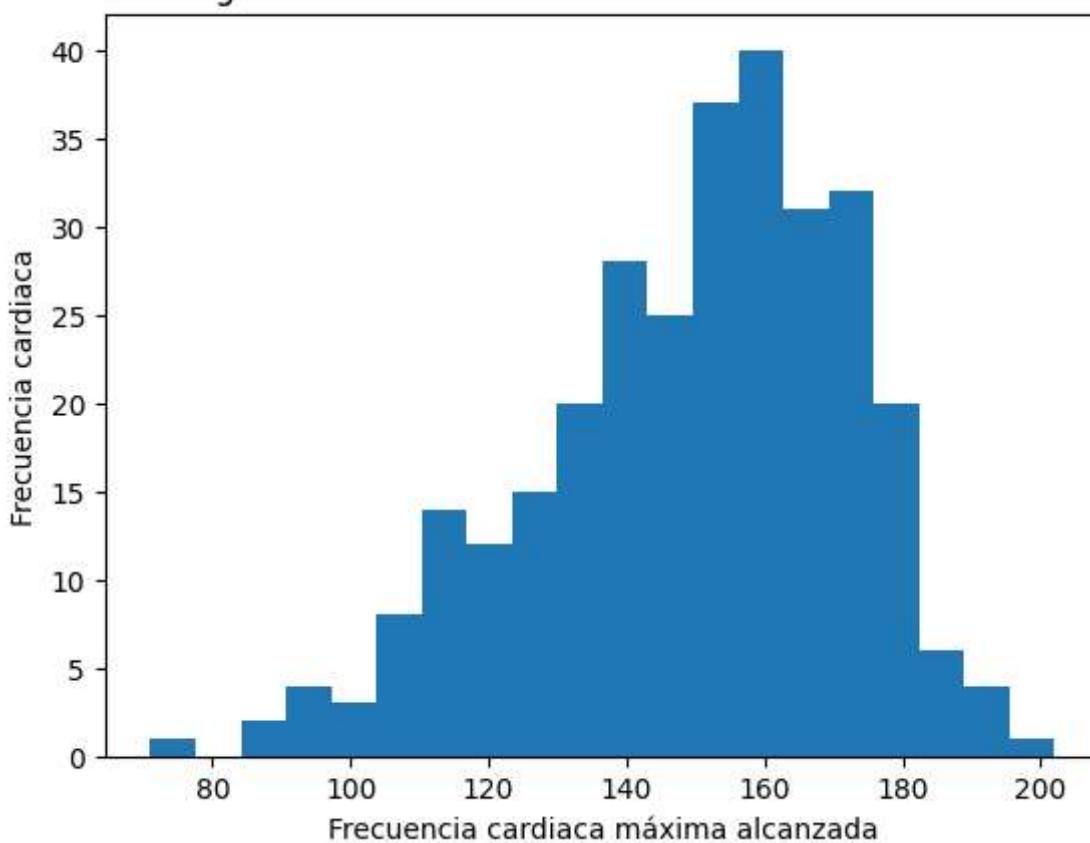
```
In [ ]: print_boxplot(  
    heart_dataset[cholesterol],  
    "Diagrama de cajas del colesterol",  
    "Colesterol",  
    "mg/dl")
```

Diagrama de cajas del colesterol

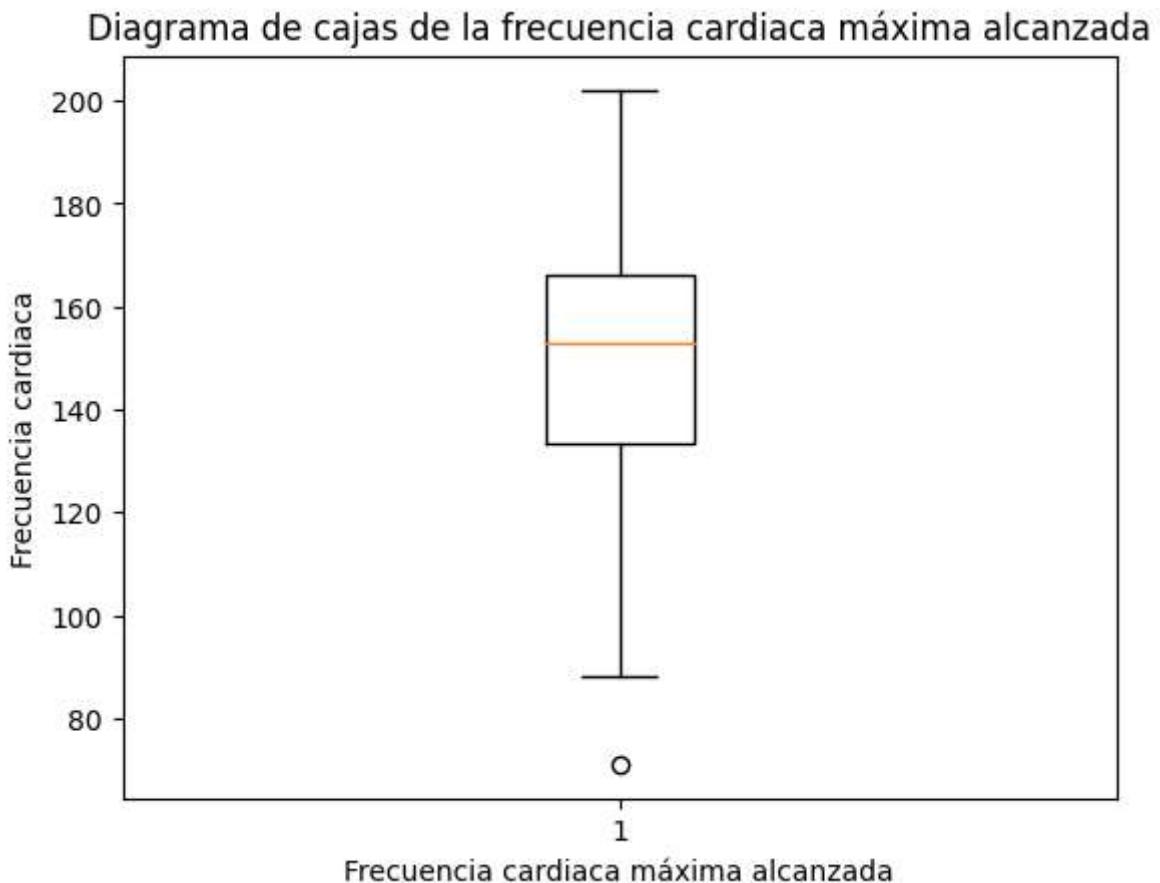


```
In [ ]: print_histogram(  
    heart_dataset[maximum_heart_rate],  
    "Histograma de la frecuencia cardiaca máxima alcanzada",  
    "Frecuencia cardiaca máxima alcanzada",  
    "Frecuencia cardiaca")
```

Histograma de la frecuencia cardiaca máxima alcanzada



```
In [ ]: print_boxplot(  
    heart_dataset[maximum_heart_rate],  
    "Diagrama de cajas de la frecuencia cardiaca máxima alcanzada",  
    "Frecuencia cardiaca máxima alcanzada",  
    "Frecuencia cardiaca")
```



Tras observar las gráficas, podemos concluir que mientras que la presión arterial en reposo y el colesterol tienen valores extremos por ser muy superiores a la media, la frecuencia máxima alcanzada es al revés, se trata de un valor muy por debajo de la media, lo que se puede intuir que este valor si se puede tratar de un error, lo cual es más difícil afirmar en el resto de valores extremos. Por lo tanto únicamente se eliminará este valor mínimo extremo de frecuencia cardíaca máxima.

```
In [ ]: ## heart_dataset = heart_dataset.loc[maximum_heart_rate_outliers]
```

## 4. Análisis de los datos

### 4.1. Selección de los grupos de datos que se van a comparar

Se comparará los valores de:

- Presión arterial en reposo con la alta o baja probabilidad de infarto.
- Colesterol con la alta o baja probabilidad de infarto.
- Frecuencia cardíaca máxima con la alta o baja probabilidad de infarto.
- El tipo de dolor torácico con la alta o baja probabilidad de infarto.
- Glucemia en ayunas alta con alta o baja probabilidad de infarto.
- Resultados electrocardiográficos con alta o baja probabilidad de infarto.

## 4.2. Comparación de la normalidad y homogeneidad de la varianza

Para la comparación de normalidad y homogeneidad de la varianza de las variables, vamos a utilizar la prueba de normalidad de Shapiro-Wilk a las variables que vamos a emplear. Para ello definimos la siguiente función en Python.

```
In [ ]: def shapiro_test(data_set, column_name):
    statistic, p_value = shapiro(data_set[column_name])
    print("Prueba de normalidad de Shapiro-Wilk en la columna ", column_name, " :")
    print("Estadístico de prueba:", statistic)
    print("Valor p:", p_value)
```

Una vez definida, ejecutamos el test para cada variable:

```
In [ ]: shapiro_test(
    heart_dataset,
    cholesterol)
```

```
Prueba de normalidad de Shapiro-Wilk en la columna chol :
Estadístico de prueba: 0.9468811750411987
Valor p: 5.364368060867264e-09
```

```
In [ ]: shapiro_test(
    heart_dataset,
    resting_blood_pressure)
```

```
Prueba de normalidad de Shapiro-Wilk en la columna trtbps :
Estadístico de prueba: 0.9659166932106018
Valor p: 1.4575286968465662e-06
```

```
In [ ]: shapiro_test(
    heart_dataset,
    fasting_blood_sugar)
```

```
Prueba de normalidad de Shapiro-Wilk en la columna fbs :
Estadístico de prueba: 0.4239869713783264
Valor p: 5.4308542423809215e-30
```

```
In [ ]: shapiro_test(
    heart_dataset,
    resting_electrocardiographic_results)
```

```
Prueba de normalidad de Shapiro-Wilk en la columna restecg :
Estadístico de prueba: 0.6793190240859985
Valor p: 1.3784006410641926e-23
```

```
In [ ]: shapiro_test(
    heart_dataset,
    maximum_heart_rate)
```

```
Prueba de normalidad de Shapiro-Wilk en la columna thalachh :
Estadístico de prueba: 0.9763153195381165
Valor p: 6.620732165174559e-05
```

Viendo los resultados, se puede concluir que como todos los p valores son muy inferiores al nivel de significancia (0,05), podemos concluir que no siguen una distribución normal.

### 4.3. Aplicación de pruebas estadísticas para comparar los grupos de datos

Como las diferentes datos del dataset no presentan una distribución normal, las pruebas estadísticas que se utilizarán para analizar y comparar los diferentes grupos serán paramétricas. Para las variables dicotómicas, utilizaremos el test de Wilcoxon.

```
In [ ]: def wilcoxon_test(data_set, column_name_first, column_name_second):
    statistic, p_value = wilcoxon(data_set[column_name_first], data_set[column_name_second])
    print("Prueba de Wilcoxon en las columnas ", column_name_first, " y ", column_name_second)
    print("Estadístico de prueba:", statistic)
    print("Valor p:", p_value)
```

Comparamos la glucemia en ayunas mayor a 120 mg/dl con la probabilidad de infarto.

```
In [ ]: wilcoxon_test(
    heart_dataset,
    fasting_blood_sugar,
    chance_heart_attack)
```

```
Prueba de Wilcoxon en las columnas  fbs  y  output
Estadístico de prueba: 1815.0
Valor p: 7.224055883238502e-21
```

en el test se muestra que las diferencias entre los grupos de datos son estadísticamente significativas ( $p < 0.05$ ). Es decir, la glucemia es estadísticamente diferente entre personas con más probabilidad de infartos y con menos probabilidad.

El test equivalente cuando se tienen 3 o más grupos de datos, como es el caso del tipo de dolor torácico y el resultado electrocardiográfico, es el test de Kruskal-Wallis:

```
In [ ]: def kruskal_test(data_set, column_name_first, column_name_second):
    statistic, p_value = kruskal(data_set[column_name_first], data_set[column_name_second])
    print("Prueba de Kruskal-Wallis en las columnas ", column_name_first, " y ", column_name_second)
    print("Estadístico de prueba:", statistic)
    print("Valor p:", p_value)
```

Comparamos el tipo de dolor torácico con la probabilidad de sufrir un infarto.

```
In [ ]: kruskal_test(
    heart_dataset,
    chest_pain,
    chance_heart_attack)
```

```
Prueba de Kruskal-Wallis en las columnas  cp  y  output
Estadístico de prueba: 17.47159524838154
Valor p: 2.9163252122459847e-05
```

Comparamos el tipo de resultado de electrocardiograma en reposo con la probabilidad de sufrir un infarto.

```
In [ ]: krushal_test(  
            heart_dataset,  
            resting_electrocardiographic_results,  
            chance_heart_attack)
```

Prueba de Kruskal-Wallis en las columnas restecg y output  
Estadístico de prueba: 0.3055092250978719  
Valor p: 0.5804490681583108

En este caso, observamos que mientras el dolor torácico si tiene una cierta diferencia estadística entre grupos por probabilidad de infarto, pero no podemos asegurar lo mismo para el tipo e electrocardiograma en reposo ya que para este caso si hay un p value significativo al ser mayor que 0,05.

A continuación, para las variables numéricas, vamos a estudiar sus correlaciones utilizando la correlación de Spearman ya que no podemos confirmar la normalidad y homocedasticidad.

```
In [ ]: def student_test(data_set, column_name, column_name_target):  
    group1 = data_set[column_name][data_set[column_name_target] == 0]  
    group2 = data_set[column_name][data_set[column_name_target] == 1]  
    statistic, p_value = ttest_ind(group1, group2)  
    print("Test de student en las columnas ", column_name, " y ", column_name_target)  
    print("Estadístico de prueba:", statistic)  
    print("Valor p:", p_value)
```

Comparamos el tipo de resultado del colesterol con la probabilidad de sufrir un infarto.

```
In [ ]: student_test(  
            heart_dataset,  
            cholestorol,  
            chance_heart_attack  
)
```

Test de student en las columnas chol y output  
Estadístico de prueba: 1.4842450762526977  
Valor p: 0.1387903269560064

Se observa que tiene un p significativo ( $p > 0.05$ ) y por lo tanto se puede concluir que no hay una diferencia significativa entre las medias de los dos grupos.

Ahora comparamos con la frecuencia máxima alcanzada.

```
In [ ]: student_test(  
            heart_dataset,  
            maximum_heart_rate,  
            chance_heart_attack  
)
```

```
Test de student en las columnas thalachh y output  
Estadístico de prueba: -8.069702869452568  
Valor p: 1.697337638656049e-14
```

Se observa que tiene un p significativo no ( $p < 0.05$ ) y por lo tanto se puede concluir que hay una diferencia significativa entre las medias de los dos grupos y que por lo tanto si es un factor a tener en cuenta para un infarto.

Por último comparamos con la presión arterial en reposo:

```
In [ ]: student_ttest(  
        heart_dataset,  
        resting_blood_pressure,  
        chance_heart_attack  
)
```

```
Test de student en las columnas trtbps y output  
Estadístico de prueba: 2.5412927171039  
Valor p: 0.011546059200233312
```

Y se observa que tiene, al igual que el nivel de colesterol, un p significativo ( $p > 0.05$ ) y por lo tanto se puede concluir que no hay una diferencia significativa entre las medias de los dos grupos y que por lo tanto no se intuye correlación.

## 6. Resolución del problema

A partir del análisis anterior, hemos podido profundizar en qué variables médicas tienen una mayor correlación con probabilidad de infarto. Con esta información se puede intuir en un paciente las probabilidades de que sufra un infarto. Se observan que variables como la frecuencia cardíaca máxima alcanzada, el tipo de dolor torácico y la glucemia en ayunas mayor a 120 mg/dl pueden ser indicadores interesantes al tener una cierta correlación. Por otro lado el resto de variables no encuentran diferencias estadísticas.

Por lo tanto, podemos concluir que de las variables analizadas, hay 3 que permiten darnos indicadores médicos que puedan aproximar en cierta medida la probabilidad de infarto.