

## ✓ Análisis de supervivencia en pacientes con cirrosis hepática

**Autor:** German Preciat; Universidad de Guadalajara

**Carrera:** Ing. Biomédica

**Materia:** Analisis de datos clínicos

**Contacto:** [german.preciat@academicos.udg.mx](mailto:german.preciat@academicos.udg.mx)

Este conjunto de datos se centra en predecir el estado de supervivencia de pacientes con cirrosis hepática, utilizando 17 características clínicas. La cirrosis hepática es el resultado de un daño prolongado en el hígado, que conduce a una cicatrización extensa, a menudo debido a condiciones como la hepatitis o el consumo crónico de alcohol. Los datos provienen de un estudio realizado por la Clínica Mayo sobre la cirrosis biliar primaria (PBC) del hígado, llevado a cabo entre 1974 y 1984.

Los datos se obtuvieron en [kaggle](https://www.kaggle.com).

### Introducción

El propósito de este conjunto de datos es analizar y predecir el estado de supervivencia de los pacientes con cirrosis hepática, lo que podría ayudar en la comprensión y el tratamiento de esta enfermedad.

Este trabajo se centrará en analizar los datos utilizando el marco de análisis de datos de Google, que comprende las siguientes fases: , Preparar, Procesar, Analizar,

- Preguntar
- Preparar
- Procesar
- Analizar
- Compartir
- Actuar

El objetivo es utilizar este conjunto de datos como práctica para la clase de análisis de datos con Python.

## ✓ Analisis de datos

Para procesar los datos en este proyecto en Google Colab, utilizaremos las siguientes bibliotecas de Python:

1. **Pandas:** Se utilizará para leer y manipular el conjunto de datos en formato de tabla.

```
import pandas as pd
```

2. **NumPy:** Proporciona estructuras de datos y funciones para trabajar de manera eficiente con matrices numéricas.

```
import numpy as np
```

3. **Matplotlib:** Se utilizará para visualizar los datos y crear gráficos.

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

4. **Seaborn:** Proporciona una interfaz de alto nivel para la creación de gráficos estadísticos atractivos y informativos.

```
import seaborn as sns
```

5. **Scikit-learn:** Esta biblioteca se utilizará para realizar análisis de datos, como dividir los datos en conjuntos de entrenamiento y prueba, y entrenar modelos de predicción.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

Además, utilizaremos las funciones de Google Colab para cargar archivos desde la computadora local al entorno de Colab. Esto se puede hacer con el siguiente código:

```
from google.colab import files
uploaded = files.upload()
```

Este código permitirá al usuario seleccionar un archivo CSV desde su computadora y cargarlo en el entorno de Colab para su posterior procesamiento.

## Preguntar

La generación de preguntas bien formuladas es fundamental para recopilar datos relevantes y significativos que nos ayuden a comprender mejor un problema o fenómeno. Al formular preguntas específicas y claras, podemos identificar las variables clave que influyen en el problema que estamos investigando.

Preguntas para la generación del conjunto de datos sobre la cirrosis hepática:

1. ¿Cuál es la relación entre el tiempo transcurrido desde el registro y el estado del paciente (muerte, trasplante o análisis)?
2. ¿Cómo afecta el tipo de medicamento administrado al estado del paciente?
3. ¿Existe una correlación entre la edad del paciente y su estado de salud?
4. ¿Hay diferencias en el estado de los pacientes en función de su género?
5. ¿La presencia de ciertas condiciones médicas como ascitis, hepatomegalia, arañas vasculares o edema afecta la supervivencia de los pacientes?
6. ¿Qué relación existe entre los niveles de diferentes biomarcadores como bilirrubina, colesterol, albúmina, cobre, fosfatasa alcalina, SGOT, triglicéridos, plaquetas y tiempo de protrombina con el estado de los pacientes?
7. ¿El estadio histológico de la enfermedad tiene algún impacto en la supervivencia de los pacientes?

En el caso del estudio de la cirrosis hepática, las preguntas planteadas nos permiten recopilar información sobre diversos aspectos de la enfermedad, como la relación entre los síntomas clínicos y el estado de los pacientes, el efecto de los tratamientos médicos, y la influencia de los biomarcadores en la progresión de la enfermedad.

Al generar un conjunto de datos basado en estas preguntas, podemos obtener una visión más completa y detallada de la cirrosis hepática, lo que a su vez nos ayudará a desarrollar mejores estrategias de diagnóstico, tratamiento y prevención. Por lo tanto, la formulación de preguntas adecuadas es un paso crucial en el proceso de investigación y análisis de datos.

## ✓ Preparar

Entender los tipos de datos que estaremos procesando es fundamental por varias razones:

1. **Interpretación adecuada:** Conocer el tipo de datos nos permite interpretarlos correctamente. Por ejemplo, si una columna contiene datos categóricos, como "Sí" o "No", sabemos que se refiere a la presencia o ausencia de cierta característica. Por otro lado, si una columna contiene datos numéricos, podemos realizar cálculos y análisis estadísticos sobre ellos.
2. **Selección de técnicas de análisis apropiadas:** Los diferentes tipos de datos requieren diferentes técnicas de análisis. Por ejemplo, para datos categóricos, podríamos utilizar análisis de frecuencia o pruebas de chi-cuadrado, mientras que para datos numéricos, podríamos utilizar correlación o regresión lineal. Comprender los tipos de datos nos ayuda a seleccionar las herramientas y métodos de análisis más adecuados para nuestro conjunto de datos.
3. **Preparación de datos efectiva:** Antes de realizar análisis más avanzados, es necesario realizar tareas de preparación de datos, como limpieza, transformación y normalización. El tipo de datos influye en cómo abordamos estas tareas. Por ejemplo, para datos faltantes en una columna numérica, podríamos optar por imputar la media o la mediana, mientras que para datos categóricos, podríamos optar por imputar la moda. Conocer los tipos de datos nos ayuda a tomar decisiones informadas durante la preparación de datos.
4. **Evitar errores de análisis:** Si no entendemos correctamente los tipos de datos, corremos el riesgo de cometer errores durante el análisis. Por ejemplo, tratar datos categóricos como numéricos podría llevar a conclusiones incorrectas. Al comprender los tipos de datos, podemos evitar estos errores y garantizar la precisión y validez de nuestros resultados.

En resumen, comprender los tipos de datos que estaremos procesando es esencial para realizar un análisis de datos efectivo y obtener conclusiones significativas. Nos permite interpretar correctamente los datos, seleccionar las técnicas de análisis adecuadas, preparar los


datos de manera efectiva y evitar errores durante el proceso de análisis.

Para leer un archivo CSV desde Google Colab, primero necesitas cargar el archivo desde tu sistema local al entorno de Colab utilizando la función `files.upload()`. Luego, puedes utilizar la biblioteca Pandas para leer el archivo CSV y almacenarlo en un DataFrame.

```
from google.colab import files
import pandas as pd

# Cargar el archivo CSV desde el sistema local
uploaded = files.upload()

# Leer el archivo CSV y almacenarlo en un DataFrame
df = pd.read_csv('cirrhosis.csv')
```

 Choose files

No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving cirrhosis.csv to cirrhosis (3).csv

Es importante comprender la estructura y el contenido del conjunto de datos para poder realizar un análisis adecuado. Esto implica comprender qué tipo de datos contiene cada columna, qué significan y cómo se relacionan con las preguntas de investigación planteadas.

```
# Mostrar las primeras filas del DataFrame
print("Primeras filas del DataFrame:")
print(df.head())
```

Primeras filas del DataFrame:

	ID	N_Days	Status	Drug	Age	Sex	Ascites	Hepatomegaly	Spiders	\
0	1	400	D	D-penicillamine	21464	F	Y		Y	Y
1	2	4500	C	D-penicillamine	20617	F	N		Y	Y
2	3	1012	D	D-penicillamine	25594	M	N		N	N
3	4	1925	D	D-penicillamine	19994	F	N		Y	Y
4	5	1504	CL	Placebo	13918	F	N		Y	Y

	Edema	Bilirubin	Cholesterol	Albumin	Copper	Alk_Phos	SGOT	\
0	Y	14.5	261.0	2.60	156.0	1718.0	137.95	
1	N	1.1	302.0	4.14	54.0	7394.8	113.52	
2	S	1.4	176.0	3.48	210.0	516.0	96.10	
3	S	1.8	244.0	2.54	64.0	6121.8	60.63	
4	N	3.4	279.0	3.53	143.0	671.0	113.15	

	Tryglicerides	Platelets	Prothrombin	Stage
0	172.0	190.0	12.2	4.0
1	88.0	221.0	10.6	3.0
2	55.0	151.0	12.0	4.0
3	92.0	183.0	10.3	4.0
4	72.0	136.0	10.9	3.0

```
# Descripción estadística del DataFrame
print("\nDescripción del DataFrame:")
print(df.describe())
```

Descripción del DataFrame:

	ID	N_Days	Age	Bilirubin	Cholesterol	\
count	418.000000	418.000000	418.000000	418.000000	284.000000	
mean	209.500000	1917.782297	18533.351675	3.220813	369.510563	
std	120.810458	1104.672992	3815.845055	4.407506	231.944545	
min	1.000000	41.000000	9598.000000	0.300000	120.000000	
25%	105.250000	1092.750000	15644.500000	0.800000	249.500000	
50%	209.500000	1730.000000	18628.000000	1.400000	309.500000	
75%	313.750000	2613.500000	21272.500000	3.400000	400.000000	
max	418.000000	4795.000000	28650.000000	28.000000	1775.000000	

	Albumin	Copper	Alk_Phos	SGOT	Tryglicerides	\
count	418.000000	310.000000	312.000000	312.000000	282.000000	
mean	3.497440	97.648387	1982.655769	122.556346	124.702128	
std	0.424972	85.613920	2140.388824	56.699525	65.148639	
min	1.960000	4.000000	289.000000	26.350000	33.000000	
25%	3.242500	41.250000	871.500000	80.600000	84.250000	
50%	3.530000	73.000000	1259.000000	114.700000	108.000000	
75%	3.770000	123.000000	1980.000000	151.900000	151.000000	
max	4.640000	588.000000	13862.400000	457.250000	598.000000	

	Platelets	Prothrombin	Stage
count	407.000000	416.000000	412.000000
mean	257.024570	10.731731	3.024272
std	98.325585	1.022000	0.882042

min	62.000000	9.000000	1.000000
25%	188.500000	10.000000	2.000000
50%	251.000000	10.600000	3.000000
75%	318.000000	11.100000	4.000000
max	721.000000	18.000000	4.000000

```
# Información sobre el tipo de datos y valores no nulos en cada columna
print("\nInformación del DataFrame:")
print(df.info())
```



```
Información del DataFrame:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 20 columns):
#   Column                Non-Null Count  Dtype
---  -
0   ID                     418 non-null   int64
1   N_Days                 418 non-null   int64
2   Status                 418 non-null   object
3   Drug                   312 non-null   object
4   Age                    418 non-null   int64
5   Sex                    418 non-null   object
6   Ascites                312 non-null   object
7   Hepatomegaly           312 non-null   object
8   Spiders                 312 non-null   object
9   Edema                  418 non-null   object
10  Bilirubin               418 non-null   float64
11  Cholesterol             284 non-null   float64
12  Albumin                 418 non-null   float64
13  Copper                  310 non-null   float64
14  Alk_Phos                312 non-null   float64
15  SGOT                    312 non-null   float64
16  Tryglicerides           282 non-null   float64
17  Platelets               407 non-null   float64
18  Prothrombin             416 non-null   float64
19  Stage                   412 non-null   float64
dtypes: float64(10), int64(3), object(7)
memory usage: 65.4+ KB
None
```

El código anterior mostrará las primeras filas del DataFrame para tener una idea inicial de la estructura de los datos, seguido de una descripción estadística que proporciona información sobre la distribución de los valores en cada columna. Además, se imprimirá información sobre el tipo de datos y la cantidad de valores no nulos en cada columna para comprender mejor la integridad de los datos.

La edad del paciente está en días. Para transformar la edad del paciente de días a años, podemos dividir cada valor en la columna 'Age' por 365, que es el número aproximado de días en un año. Aquí tienes el código para realizar esta transformación:

```
# Transformar la edad del paciente de días a años
df['Age'] = df['Age'] / 365
```

Este código divide cada valor en la columna *Age* por 365 y actualiza el DataFrame con los valores transformados. Ahora, la columna *Age* representará la edad del paciente en años en lugar de días.

La descripción del conjunto de datos con código nos ayudará a entender la naturaleza de los datos y su relevancia para las preguntas de investigación, lo que a su vez nos permitirá realizar un análisis más efectivo y obtener conclusiones significativas.

### Descripción de datos

Aquí está la descripción de cada columna del conjunto de datos y su importancia para las preguntas de investigación:

1. **ID:** Este es un número entero que sirve como identificador único para cada paciente en el estudio. Es importante para poder realizar un seguimiento individual de cada paciente y asociar sus características con su estado de supervivencia.
2. **N\_Days:** Es un número entero que representa el número de días entre el registro del paciente y el evento más relevante (muerte, trasplante o análisis del estudio). Este dato es crucial para entender la duración de la observación de cada paciente en el estudio y su relación con el estado de supervivencia.
3. **Status:** Es una variable categórica que indica el estado del paciente en el momento del evento más relevante. Los valores posibles son "C" (censurado), "CL" (censurado debido a trasplante de hígado) o "D" (muerte). Esta columna es fundamental para identificar los resultados del estudio y determinar los factores asociados con la supervivencia de los pacientes.

4. **Drug:** Es una variable categórica que indica el tipo de medicamento recibido por el paciente: D-penicilamina o placebo. Este dato es importante para evaluar el efecto del tratamiento en la supervivencia de los pacientes y su progresión de la enfermedad.
5. **Age:** Es un número entero que representa la edad del paciente en el momento del registro. La edad puede ser un factor importante en la progresión de la cirrosis hepática y en la respuesta al tratamiento.
6. **Sex:** Es una variable categórica que indica el género del paciente (masculino o femenino). El género puede desempeñar un papel en la susceptibilidad a la cirrosis hepática y en la respuesta al tratamiento.
7. **Ascites:** Es una variable categórica que indica la presencia de ascitis en el paciente (Sí o No). La ascitis es una complicación común de la cirrosis hepática y puede influir en la supervivencia del paciente.
8. **Hepatomegaly:** Es una variable categórica que indica la presencia de hepatomegalia en el paciente (Sí o No). La hepatomegalia puede ser un signo de enfermedad hepática avanzada y puede estar relacionada con un peor pronóstico.
9. **Spiders:** Es una variable categórica que indica la presencia de arañas vasculares en el paciente (Sí o No). Las arañas vasculares son un signo clásico de enfermedad hepática crónica y pueden estar asociadas con un mayor riesgo de complicaciones.
10. **Edema:** Es una variable categórica que indica la presencia de edema en el paciente (No, Sí sin terapia diurética o Sí a pesar de la terapia diurética). El edema es una complicación común de la cirrosis hepática y puede afectar la calidad de vida y la supervivencia del paciente.
11. **Bilirrubina:** Es una variable continua que representa el nivel de bilirrubina sérica en el paciente, medida en mg/dl. La bilirrubina es un marcador importante de la función hepática y puede estar elevada en pacientes con cirrosis hepática.
12. **Colesterol:** Es un número entero que representa el nivel de colesterol sérico en el paciente, medido en mg/dl. Los niveles de colesterol pueden estar alterados en pacientes con enfermedad hepática y pueden influir en el riesgo cardiovascular.
13. **Albumina:** Es una variable continua que representa el nivel de albúmina sérica en el paciente, medida en gm/dl. La albúmina es una proteína producida por el hígado y los niveles bajos pueden indicar disfunción hepática.
14. **Cobre:** Es un número entero que representa el nivel de cobre en la orina del paciente, medido en ug/día. La acumulación de cobre en el cuerpo puede ser un signo de enfermedades hepáticas como la enfermedad de Wilson.
15. **Alk\_Phos:** Es una variable continua que representa el nivel de fosfatasa alcalina en el suero del paciente, medida en u/liter. La fosfatasa alcalina es una enzima producida por el hígado y su nivel puede estar elevado en pacientes con enfermedades hepáticas.
16. **SGOT:** Es una variable continua que representa el nivel de aspartato aminotransferasa en el suero del paciente, medida en u/ml. Los niveles elevados de SGOT pueden indicar daño hepático.
17. **Triglicéridos:** Es un número entero que representa el nivel de triglicéridos en el suero del paciente, medido en mg/dl. Los niveles elevados de triglicéridos pueden estar asociados con enfermedades hepáticas y metabólicas.
18. **Plaquetas:** Es un número entero que representa el recuento de plaquetas en la sangre del paciente, medido por ml/1000. Los niveles bajos de plaquetas pueden ser un signo de disfunción hepática.
19. **Prothrombin:** Es una variable continua que representa el tiempo de protrombina del paciente, medido en segundos. La protrombina es una proteína producida por el hígado y su tiempo de coagulación puede estar prolongado en pacientes con enfermedad hepática.
20. **Stage:** Es una variable categórica que indica el estadio histológico de la enfermedad hepática (1, 2, 3 o 4). El estadio histológico puede proporcionar información sobre la gravedad y la progresión de la enfermedad.

Cada uno de estos datos es importante para comprender mejor la cirrosis hepática y su impacto en la salud de los pacientes. Al analizar estos datos en conjunto, podemos identificar factores de riesgo, entender la progresión de la enfermedad y desarrollar estrategias de tratamiento más efectivas.

## ✓ Procesar

Preparar los datos es un paso crucial en el proceso de análisis de datos, ya que garantiza que los datos estén en un formato adecuado y sean aptos para el análisis. La preparación de datos implica limpiar, transformar y preprocesar los datos para eliminar errores, inconsistencias y valores faltantes, y garantizar la calidad y la integridad de los datos.

En el caso de las columnas que tienen valores de N/A (es decir, datos faltantes), es importante abordar esta situación de manera adecuada. Ignorar los valores faltantes puede sesgar los resultados del análisis y afectar la precisión de las conclusiones. Por lo tanto, es necesario tomar medidas para manejar los valores faltantes de manera adecuada.

Una opción es eliminar las filas que contienen valores de N/A en las columnas relevantes. Esto se puede hacer utilizando el método `dropna()` de Pandas. Sin embargo, esta opción puede llevar a la pérdida de datos, especialmente si hay una cantidad significativa de valores faltantes.

Otra opción es imputar los valores faltantes utilizando técnicas como la imputación media, mediana o moda, dependiendo del tipo de datos y la distribución de los valores. Esto implica reemplazar los valores faltantes con un valor estimado basado en los valores observados en otras filas. La imputación se puede realizar utilizando el método `fillna()` de Pandas.

En este caso, como las columnas con valores de N/A son de diversos tipos (categóricas y numéricas), podríamos optar por la imputación media para las columnas numéricas y la imputación de moda para las columnas categóricas. Esto nos permitirá mantener la integridad de los datos y evitar la pérdida de información.

A continuación, se muestra cómo podemos manejar los valores faltantes en las columnas relevantes utilizando Python.

El siguiente código realiza la imputación de valores faltantes en un DataFrame `df` para asegurar que no haya datos faltantes antes de realizar análisis posteriores. Aquí está el paso a paso de lo que hace cada parte del código:

### 1. Imputar la moda para las columnas categóricas:

```
# Imputar la moda para las columnas categóricas
categorical_columns = ['Ascites', 'Hepatomegaly', 'Spiders']
for col in categorical_columns:
    df[col].fillna(df[col].mode()[0], inplace=True)
```

- Se define una lista llamada `categorical_columns` que contiene los nombres de las columnas categóricas que tienen valores faltantes ('Ascites', 'Hepatomegaly', 'Spiders').
- Se itera sobre cada columna categórica en la lista `categorical_columns`.
- Para cada columna, se utiliza el método `mode()` de Pandas para calcular la moda de esa columna, que es el valor más frecuente.
- El valor de moda se asigna a los valores faltantes en esa columna utilizando el método `fillna()` de Pandas. `inplace=True` indica que la imputación se realiza directamente en el DataFrame original.

### 2. Imputar la media para las columnas numéricas:

Se hace lo mismo para las columnas numéricas:

```
# Imputar la media para las columnas numéricas
numeric_columns = ['Cholesterol', 'Copper', 'Alk_Phos', 'SGOT', 'Tryglicerides', 'Platelets', 'Prothrombin']
for col in numeric_columns:
    df[col].fillna(df[col].mean(), inplace=True)
```

- Se define una lista llamada `numeric_columns` que contiene los nombres de las columnas numéricas que tienen valores faltantes ('Cholesterol', 'Copper', 'Alk\_Phos', 'SGOT', 'Tryglicerides', 'Platelets', 'Prothrombin').
- Se itera sobre cada columna numérica en la lista `numeric_columns`.
- Para cada columna, se utiliza el método `mean()` de Pandas para calcular la media de esa columna.
- El valor de la media se asigna a los valores faltantes en esa columna utilizando el método `fillna()` de Pandas. `inplace=True` indica que la imputación se realiza directamente en el DataFrame original.

### 3. Verificar que no hay valores faltantes:

```
# Verificar que no hay valores faltantes
print("Valores faltantes después de la imputación:")
print(df.isnull().sum())
```

```
↗ Valores faltantes después de la imputación:
ID                0
N_Days            0
Status            0
Drug             106
Age              0
Sex              0
Ascites          0
Hepatomegaly     0
Spiders          0
Edema            0
Bilirubin        0
Cholesterol      0
Albumin          0
Copper           0
Alk_Phos         0
SGOT             0
```

```

Tryglicerides    0
Platelets        0
Prothrombin      0
Stage            6
dtype: int64

```

- Se imprime un mensaje indicando que se están verificando los valores faltantes después de la imputación.
- Se utiliza el método `isnull().sum()` de Pandas para calcular la cantidad de valores faltantes en cada columna del DataFrame.
- Se imprime el resultado, que mostrará la suma de valores faltantes por columna. Si no hay valores faltantes, todos los resultados deberían ser cero.

En resumen, este código realiza la imputación de valores faltantes utilizando la moda para columnas categóricas y la media para columnas numéricas, y luego verifica que no haya valores faltantes en el DataFrame resultante. Esto garantiza que los datos estén listos para su análisis posterior.

```

df['Drug'].fillna(df['Drug'].mode()[0], inplace=True)
df['Stage'].fillna(df['Stage'].mean(), inplace=True)

```

```

print("Valores faltantes después de la imputación:")
print(df.isnull().sum())

```

➞ Valores faltantes después de la imputación:

```

ID            0
N_Days        0
Status        0
Drug          0
Age           0
Sex           0
Ascites       0
Hepatomegaly  0
Spiders       0
Edema         0
Bilirubin     0
Cholesterol   0
Albumin       0
Copper        0
Alk_Phos      0
SGOT         0
Tryglicerides 0
Platelets     0
Prothrombin   0
Stage         0
dtype: int64

```

## ✓ Analizar

Scikit-learn (sklearn) es una biblioteca de aprendizaje automático en Python que proporciona herramientas simples y eficientes para la minería y el análisis de datos. Incluye varios algoritmos de clasificación, regresión, clustering y preprocesamiento de datos, así como herramientas para evaluar y ajustar modelos.

```
from sklearn.model_selection import train_test_split
```

La línea de código `from sklearn.model_selection import train_test_split` importa la función `train_test_split` de la sublibrería `model_selection` de scikit-learn. Esta función es una herramienta esencial en el análisis de datos y el aprendizaje automático, especialmente para evaluar el rendimiento de los modelos y evitar el sobreajuste.

La función `train_test_split` divide un conjunto de datos en dos conjuntos más pequeños: un conjunto de entrenamiento y un conjunto de prueba. Esto es fundamental para evaluar la capacidad predictiva de un modelo, ya que permite entrenar el modelo en una parte de los datos y luego evaluar su rendimiento en datos no vistos.

```
from sklearn.linear_model import LogisticRegression
```

La línea de código `from sklearn.linear_model import LogisticRegression` importa la clase `LogisticRegression` del módulo `linear_model` de la biblioteca scikit-learn (sklearn). Esta clase implementa la regresión logística, que es un modelo estadístico utilizado para la clasificación binaria y multiclase.

Aquí hay una descripción de la regresión logística y cómo se utiliza en el aprendizaje automático:

- **Regresión Logística:**

- La regresión logística es un modelo de clasificación que se utiliza para predecir la probabilidad de que una instancia pertenezca a una clase particular.
- Aunque se llama "regresión", la regresión logística se utiliza comúnmente para problemas de clasificación.
- La regresión logística utiliza una función logística para modelar la relación entre las variables independientes y la probabilidad de que una instancia pertenezca a una clase.
- Es un modelo lineal generalizado que utiliza la función sigmoide para transformar la salida lineal en una probabilidad en el rango de 0 a 1.

- **Uso de `LogisticRegression` en `scikit-learn`:**

- `LogisticRegression` es una implementación de la regresión logística en `scikit-learn`.
- Se utiliza para entrenar modelos de regresión logística en conjuntos de datos de entrenamiento.
- Puede manejar tanto problemas de clasificación binaria como multiclase.
- La clase `LogisticRegression` en `scikit-learn` admite regularización para controlar el sobreajuste y mejorar el rendimiento del modelo.
- Puede configurar varios hiperparámetros, como el tipo de regularización (L1 o L2), la fuerza de regularización y la tolerancia para la convergencia del algoritmo.

En resumen, `LogisticRegression` en `scikit-learn` es una herramienta poderosa y fácil de usar para entrenar modelos de regresión logística y realizar tareas de clasificación en conjuntos de datos de aprendizaje automático. Es ampliamente utilizado en aplicaciones de ciencia de datos y aprendizaje automático para resolver una variedad de problemas de clasificación.

```
from sklearn.preprocessing import StandardScaler
```

La línea de código `from sklearn.preprocessing import StandardScaler` importa la clase `StandardScaler` de la sublibrería `preprocessing` de `scikit-learn`. Esta clase es una herramienta comúnmente utilizada en el preprocesamiento de datos en el aprendizaje automático, específicamente para estandarizar las características de un conjunto de datos.

La estandarización es un proceso importante en el preprocesamiento de datos que transforma las características de manera que tengan una media de 0 y una desviación estándar de 1. Esto es útil porque muchos algoritmos de aprendizaje automático asumen que las características están centradas en cero y tienen la misma escala. La estandarización ayuda a garantizar que todas las características contribuyan de manera equitativa al modelo, sin que una característica con una escala más grande domine las demás.

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

La línea de código `from sklearn.metrics import accuracy_score, classification_report, confusion_matrix` importa tres funciones importantes de la sublibrería `metrics` de `scikit-learn`. Estas funciones son fundamentales para evaluar el rendimiento de los modelos de clasificación en el aprendizaje automático.

Aquí hay una descripción de cada una de estas funciones:

1. **`accuracy_score`:**

- Esta función calcula la precisión del modelo, que es la proporción de predicciones correctas sobre el total de predicciones realizadas por el modelo.
- La precisión se calcula utilizando la fórmula: (número de predicciones correctas) / (número total de predicciones).
- Es una métrica comúnmente utilizada para evaluar modelos de clasificación cuando las clases están balanceadas.

2. **`classification_report`:**

- Esta función genera un informe de clasificación que incluye varias métricas de evaluación del modelo, como precisión, recall, F1-score y soporte, para cada clase en el conjunto de datos.
- El informe de clasificación proporciona una descripción detallada del rendimiento del modelo, útil para comprender cómo el modelo se comporta en diferentes clases.

3. **`confusion_matrix`:**

- Esta función calcula la matriz de confusión, que es una tabla que muestra el número de predicciones correctas e incorrectas realizadas por el modelo para cada clase.
- La matriz de confusión ayuda a visualizar el rendimiento del modelo y a identificar posibles áreas de mejora. Puede ser especialmente útil para comprender cómo el modelo clasifica incorrectamente las muestras.



Estas funciones son esenciales para evaluar el rendimiento de los modelos de clasificación y proporcionar información detallada sobre su comportamiento en diferentes aspectos. Se utilizan comúnmente en la fase de evaluación del ciclo de desarrollo de modelos de aprendizaje automático para medir la calidad y la precisión de las predicciones del modelo.

Ahora, vamos a responder las preguntas utilizando las herramientas de scikit-learn, pero antes repasemos las preguntas:

1. ¿Cuál es la relación entre el tiempo transcurrido desde el registro y el estado del paciente (muerte, trasplante o análisis)?
2. ¿Cómo afecta el tipo de medicamento administrado al estado del paciente?
3. ¿Existe una correlación entre la edad del paciente y su estado de salud?
4. ¿Hay diferencias en el estado de los pacientes en función de su género?
5. ¿La presencia de ciertas condiciones médicas como ascitis, hepatomegalia, arañas vasculares o edema afecta la supervivencia de los pacientes?
6. ¿Qué relación existe entre los niveles de diferentes biomarcadores como bilirrubina, colesterol, albúmina, cobre, fosfatasa alcalina, SGOT, triglicéridos, plaquetas y tiempo de protrombina con el estado de los pacientes?
7. ¿El estadio histológico de la enfermedad tiene algún impacto en la supervivencia de los pacientes?

### 1. Relación entre el tiempo transcurrido desde el registro y el estado del paciente::


```
# Dividir el conjunto de datos en características (X) y variable objetivo (y)
X = df['N_Days'].values.reshape(-1, 1)
y = df['Status']
```

- Seleccionamos la característica 'N\_Days' como nuestras variables predictoras (características) y la variable 'Status' como nuestra variable objetivo (etiqueta).
- Usamos `.values` para convertir las columnas seleccionadas en arreglos NumPy.
- Usamos `.reshape(-1, 1)` para asegurarnos de que la matriz de características tenga la forma correcta para scikit-learn, donde cada fila representa una observación y cada columna una característica.

```
# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

- Utilizamos la función `train_test_split` para dividir los datos en conjuntos de entrenamiento y prueba.
- Especificamos que el 20% de los datos se usarán como conjunto de prueba (`test_size=0.2`).
- `random_state=42` se utiliza para garantizar que la división de los datos sea reproducible.

```
# Crear y entrenar un modelo de clasificación (por ejemplo, Regresión Logística)
model = LogisticRegression()
model.fit(X_train, y_train)
```

 `LogisticRegression`  
`LogisticRegression()`

- Creamos una instancia del modelo de Regresión Logística utilizando `LogisticRegression()`.
- Luego, ajustamos (entrenamos) el modelo utilizando los datos de entrenamiento (`X_train` y `y_train`) mediante el método `.fit()`.

```
# Predecir el estado del paciente en el conjunto de prueba
y_pred = model.predict(X_test)
```

- Utilizamos el modelo entrenado para hacer predicciones sobre los datos de prueba (`X_test`) utilizando el método `.predict()`.

```
# Evaluar la precisión del modelo
accuracy = accuracy_score(y_test, y_pred)
print("Precisión del modelo:", accuracy)
```

 Precisión del modelo: 0.6666666666666666

- Calculamos la precisión del modelo comparando las etiquetas verdaderas (`y_test`) con las predicciones del modelo (`y_pred`) utilizando la función `accuracy_score()`.

- De todas las muestras en el conjunto de prueba, aproximadamente el 66.67% de ellas fueron clasificadas correctamente por el modelo.

```
# Mostrar reporte de clasificación
print("\nReporte de clasificación:")
print(classification_report(y_test, y_pred))
```



```
Reporte de clasificación:
              precision    recall  f1-score   support

     C         0.68         0.73         0.70         44
     CL         0.00         0.00         0.00          4
     D         0.65         0.67         0.66         36

 accuracy          0.67         0.67         0.67         84
 macro avg         0.44         0.46         0.45         84
weighted avg         0.63         0.67         0.65         84
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-
_warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-
_warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-
_warn_prf(average, modifier, msg_start, len(result))
```

- Generamos un reporte de clasificación que incluye precision, recall, f1-score y support para cada clase utilizando `classification_report()`.
- El reporte de clasificación proporciona métricas de evaluación detalladas para cada clase en el conjunto de prueba. Aquí hay una explicación de las métricas que se presentan en el reporte:
  - Precision (Precisión):** La precisión indica la proporción de instancias clasificadas como positivas que fueron correctamente clasificadas. Se calcula como el número de verdaderos positivos dividido por el número de verdaderos positivos más falsos positivos. En otras palabras, la precisión mide la calidad de las predicciones positivas del modelo.
  - Recall (Recall o Sensibilidad):** El recall indica la proporción de instancias positivas que fueron correctamente identificadas por el modelo. Se calcula como el número de verdaderos positivos dividido por el número de verdaderos positivos más falsos negativos. En otras palabras, el recall mide la capacidad del modelo para encontrar todas las instancias positivas.
  - F1-score:** El F1-score es la media armónica de precision y recall. Proporciona un equilibrio entre precision y recall y es útil cuando las clases están desbalanceadas. Se calcula como  $2 * (\text{precision} * \text{recall}) / (\text{precision} + \text{recall})$ .
  - Support:** Support indica el número de ocurrencias reales de cada clase en el conjunto de prueba.
  - Accuracy (Precisión global):** La precisión global indica la proporción de predicciones correctas realizadas por el modelo sobre el total de predicciones realizadas. Se calcula como el número de predicciones correctas dividido por el número total de predicciones.
  - Macro average (Promedio macro):** El promedio macro calcula las métricas promedio para cada clase y luego toma el promedio no ponderado de estas métricas.
  - Weighted average (Promedio ponderado):** El promedio ponderado calcula las métricas promedio para cada clase pero pondera estas métricas por el soporte de cada clase (el número de ocurrencias reales de cada clase en el conjunto de prueba).

En este caso específico, el reporte muestra métricas para tres clases (C, CL y D) en el conjunto de prueba. También muestra advertencias (`UndefinedMetricWarning`) porque hay clases para las cuales no se realizaron predicciones (no hay muestras predichas). Esto podría ocurrir cuando el modelo no es capaz de predecir ciertas clases debido a la distribución de los datos o limitaciones del modelo.

```
# Mostrar matriz de confusión
print("\nMatriz de confusión:")
print(confusion_matrix(y_test, y_pred))
```



```
Matriz de confusión:
[[32  0 12]
 [ 3  0  1]
 [12  0 24]]
```

- Calculamos y mostramos la matriz de confusión utilizando `confusion_matrix()`, que es una tabla que muestra los resultados de clasificación comparando las etiquetas verdaderas y las predicciones del modelo.

La matriz de confusión es una tabla que muestra las predicciones del modelo comparadas con las etiquetas verdaderas en un problema de clasificación. Cada fila de la matriz representa las instancias en una clase predicha, mientras que cada columna representa las instancias en una clase real. En una matriz de confusión típica de clasificación binaria, las filas corresponden a las predicciones de la clase positiva y negativa, mientras que las columnas corresponden a las etiquetas verdaderas de la clase positiva y negativa, respectivamente.

Sin embargo, en este caso estás tratando con un problema de clasificación multiclase, por lo que la matriz de confusión es un poco más compleja. La interpretación de la matriz de confusión se realiza de la siguiente manera:

- La fila 1 corresponde a la clase 'C', la fila 2 corresponde a la clase 'CL' y la fila 3 corresponde a la clase 'D'.
- La columna 1 corresponde a las instancias que realmente son de la clase 'C', la columna 2 corresponde a las instancias que realmente son de la clase 'CL' y la columna 3 corresponde a las instancias que realmente son de la clase 'D'.

Con eso en mente, puedes interpretar cada celda de la matriz de confusión:

- La celda (1,1) contiene el número de instancias que fueron correctamente clasificadas como 'C' (verdaderos positivos).
- La celda (1,2) contiene el número de instancias que fueron clasificadas incorrectamente como 'CL' (falsos negativos).
- La celda (1,3) contiene el número de instancias que fueron clasificadas incorrectamente como 'D' (falsos negativos).
- La celda (2,1) contiene el número de instancias que fueron clasificadas incorrectamente como 'C' (falsos positivos).
- La celda (2,2) contiene el número de instancias que fueron correctamente clasificadas como 'CL' (verdaderos positivos).
- La celda (2,3) contiene el número de instancias que fueron clasificadas incorrectamente como 'D' (falsos negativos).
- La celda (3,1) contiene el número de instancias que fueron clasificadas incorrectamente como 'C' (falsos positivos).
- La celda (3,2) contiene el número de instancias que fueron clasificadas incorrectamente como 'CL' (falsos positivos).
- La celda (3,3) contiene el número de instancias que fueron correctamente clasificadas como 'D' (verdaderos positivos).

En resumen, la matriz de confusión proporciona una descripción detallada de las predicciones del modelo y es útil para evaluar el rendimiento del clasificador, especialmente en problemas de clasificación multiclase.

### Interpretación:

- La precisión del modelo indica qué tan bien puede predecir el estado del paciente en función del tiempo transcurrido desde el registro.
- El reporte de clasificación proporciona métricas detalladas por clase, lo que permite evaluar el rendimiento del modelo para cada estado del paciente.
- La matriz de confusión muestra el número de instancias clasificadas correctamente e incorrectamente para cada clase, lo que proporciona una visión más detallada del rendimiento del modelo.

```
# Calcular estadísticas descriptivas de N_Days agrupadas por el estado del paciente
stats_by_status = df.groupby('Status')['N_Days'].describe()
```

```
# Imprimir las estadísticas descriptivas
print("Estadísticas descriptivas de N_Days agrupadas por el estado del paciente:")
print(stats_by_status)
```

```
↗ Estadísticas descriptivas de N_Days agrupadas por el estado del paciente:
```

	count	mean	std	min	25%	50%	75%	max
Status								
C	232.0	2333.155172	994.658954	691.0	1456.5	2186.5	2979.5	4795.0
CL	25.0	1546.200000	753.074255	533.0	901.0	1435.0	2241.0	3092.0
D	161.0	1376.931677	1049.227967	41.0	597.0	1083.0	2071.0	4191.0

Estas estadísticas descriptivas proporcionan información sobre la variable 'N\_Days' (número de días transcurridos desde el registro) para diferentes estados del paciente:

- **count:** El número total de observaciones para cada estado del paciente.
- **mean:** La media del tiempo transcurrido desde el registro para cada estado del paciente. Por ejemplo, para el estado 'C', la media es aproximadamente 2333.16 días.
- **std:** La desviación estándar del tiempo transcurrido desde el registro para cada estado del paciente. Indica la dispersión de los datos alrededor de la media. Por ejemplo, para el estado 'C', la desviación estándar es aproximadamente 994.66 días.
- **min:** El valor mínimo del tiempo transcurrido desde el registro para cada estado del paciente. Por ejemplo, para el estado 'D', el valor mínimo es 41 días.
- **25%:** El percentil 25, también conocido como el primer cuartil. Indica el valor por debajo del cual cae el 25% de los datos. Por ejemplo, para el estado 'C', el primer cuartil es aproximadamente 1456.5 días.
- **50%:** El percentil 50, también conocido como la mediana. Indica el valor que separa la mitad superior de la mitad inferior de los datos. Por ejemplo, para el estado 'CL', la mediana es 1435 días.

- **75%:** El percentil 75, también conocido como el tercer cuartil. Indica el valor por debajo del cual cae el 75% de los datos. Por ejemplo, para el estado 'D', el tercer cuartil es aproximadamente 2071 días.
- **max:** El valor máximo del tiempo transcurrido desde el registro para cada estado del paciente. Por ejemplo, para el estado 'C', el valor máximo es 4795 días.

2. Impacto del tipo de medicamento administrado en el estado del paciente:

Para contestar esta pregunta utilizaremos un código similar al anterior con una pequeña modificación, añadir: `X = pd.get_dummies(X, drop_first=True)` después de dividir el conjunto.

El código `X = pd.get_dummies(X, drop_first=True)` está realizando una transformación llamada "one-hot encoding" en las variables categóricas. En este caso, estás aplicando esta transformación a la variable `Drug`, que es una variable categórica con dos posibles valores: `D-penicillamine` o `placebo`.

La función `pd.get_dummies()` de la biblioteca `pandas` toma una variable categórica y la convierte en una o más variables binarias (0 o 1), donde cada posible valor único de la variable original se convierte en una nueva columna en el `DataFrame` y se asigna un valor de 1 si la observación tiene ese valor y 0 en caso contrario.

El parámetro `drop_first=True` controla si se deben eliminar la primera columna después de la codificación. Esto se hace para evitar la multicolinealidad en los datos, lo que significa que una columna se puede predecir perfectamente a partir de las demás, lo que puede llevar a problemas en algunos modelos de aprendizaje automático.

```
# Dividir el conjunto de datos en características (X) y variable objetivo (y)
X = df[['Drug']]
y = df['Status']

# Convertir variables categóricas en variables numéricas utilizando one-hot encoding
X = pd.get_dummies(X, drop_first=True)

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Crear y entrenar un modelo de clasificación (por ejemplo, Regresión Logística)
model = LogisticRegression()
model.fit(X_train, y_train)

# Predecir el estado del paciente en el conjunto de prueba
y_pred = model.predict(X_test)

# Evaluar la precisión del modelo
accuracy = accuracy_score(y_test, y_pred)
print("Precisión del modelo:", accuracy)

# Mostrar reporte de clasificación
print("\nReporte de clasificación:")
print(classification_report(y_test, y_pred))

# Mostrar matriz de confusión
print("\nMatriz de confusión:")
print(confusion_matrix(y_test, y_pred))
```

 Precisión del modelo: 0.5238095238095238

Reporte de clasificación:					
	precision	recall	f1-score	support	
C	0.52	1.00	0.69	44	
CL	0.00	0.00	0.00	4	
D	0.00	0.00	0.00	36	
accuracy			0.52	84	
macro avg	0.17	0.33	0.23	84	
weighted avg	0.27	0.52	0.36	84	

```
Matriz de confusión:
[[44  0  0]
 [ 4  0  0]
 [36  0  0]]
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-:
_warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-
```

```
_warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-
_warn_prf(average, modifier, msg_start, len(result))
```

### 3. Correlación entre la edad del paciente y su estado de salud:

```
# Dividir el conjunto de datos en características (X) y variable objetivo (y)
X = df['Age'].values.reshape(-1, 1)
y = df['Status']

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Crear y entrenar un modelo de clasificación (por ejemplo, Regresión Logística)
model = LogisticRegression()
model.fit(X_train, y_train)

# Predecir el estado del paciente en el conjunto de prueba
y_pred = model.predict(X_test)

# Evaluar la precisión del modelo
accuracy = accuracy_score(y_test, y_pred)
print("Precisión del modelo:", accuracy)

# Mostrar reporte de clasificación
print("\nReporte de clasificación:")
print(classification_report(y_test, y_pred))

# Mostrar matriz de confusión
print("\nMatriz de confusión:")
print(confusion_matrix(y_test, y_pred))
```

➞ Precisión del modelo: 0.5357142857142857

Reporte de clasificación:

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-
_warn_prf(average, modifier, msg_start, len(result))
```

	precision	recall	f1-score	support
C	0.53	0.89	0.67	44
CL	0.00	0.00	0.00	4
D	0.55	0.17	0.26	36
accuracy			0.54	84
macro avg	0.36	0.35	0.31	84
weighted avg	0.51	0.54	0.46	84

Matriz de confusión:

```
[[39  0  5]
 [ 4  0  0]
 [30  0  6]]
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-
_warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-
_warn_prf(average, modifier, msg_start, len(result))
```

### 4. Diferencias en el estado de los pacientes según su género:

```
# Dividir el conjunto de datos en características (X) y variable objetivo (y)
X = df[['Sex']]
y = df['Status']

# Convertir variables categóricas en variables numéricas utilizando one-hot encoding
X = pd.get_dummies(X, drop_first=True)

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Crear y entrenar un modelo de clasificación (por ejemplo, Regresión Logística)
model = LogisticRegression()
model.fit(X_train, y_train)
```

```
# Predecir el estado del paciente en el conjunto de prueba
y_pred = model.predict(X_test)

# Evaluar la precisión del modelo
accuracy = accuracy_score(y_test, y_pred)
print("Precisión del modelo:", accuracy)

# Mostrar reporte de clasificación
print("\nReporte de clasificación:")
print(classification_report(y_test, y_pred))

# Mostrar matriz de confusión
print("\nMatriz de confusión:")
print(confusion_matrix(y_test, y_pred))
```

 Precisión del modelo: 0.5595238095238095

Reporte de clasificación:


	precision	recall	f1-score	support
C	0.55	0.95	0.69	44
CL	0.00	0.00	0.00	4
D	0.71	0.14	0.23	36
accuracy			0.56	84
macro avg	0.42	0.36	0.31	84
weighted avg	0.59	0.56	0.46	84

```
Matriz de confusión:
[[42  0  2]
 [ 4  0  0]
 [31  0  5]]
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-
_warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-
_warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-
_warn_prf(average, modifier, msg_start, len(result))
```

1. Relación entre el tiempo transcurrido desde el registro y el estado del paciente:

```
# Calcular estadísticas descriptivas de N_Days agrupadas por el estado del paciente
stats_by_status = df.groupby('Status')['N_Days'].describe()

# Imprimir las estadísticas descriptivas
print("Estadísticas descriptivas de N_Days agrupadas por el estado del paciente:")
print(stats_by_status)
```

 Estadísticas descriptivas de N\_Days agrupadas por el estado del paciente:

	count	mean	std	min	25%	50%	75%	max
Status								
C	232.0	2333.155172	994.658954	691.0	1456.5	2186.5	2979.5	4795.0
CL	25.0	1546.200000	753.074255	533.0	901.0	1435.0	2241.0	3092.0
D	161.0	1376.931677	1049.227967	41.0	597.0	1083.0	2071.0	4191.0

Estas estadísticas descriptivas proporcionan información sobre la variable 'N\_Days' (número de días transcurridos desde el registro) para diferentes estados del paciente:

- **count:** El número total de observaciones para cada estado del paciente.
- **mean:** La media del tiempo transcurrido desde el registro para cada estado del paciente. Por ejemplo, para el estado 'C', la media es aproximadamente 2333.16 días.
- **std:** La desviación estándar del tiempo transcurrido desde el registro para cada estado del paciente. Indica la dispersión de los datos alrededor de la media. Por ejemplo, para el estado 'C', la desviación estándar es aproximadamente 994.66 días.
- **min:** El valor mínimo del tiempo transcurrido desde el registro para cada estado del paciente. Por ejemplo, para el estado 'D', el valor mínimo es 41 días.
- **25%:** El percentil 25, también conocido como el primer cuartil. Indica el valor por debajo del cual cae el 25% de los datos. Por ejemplo, para el estado 'C', el primer cuartil es aproximadamente 1456.5 días.
- **50%:** El percentil 50, también conocido como la mediana. Indica el valor que separa la mitad superior de la mitad inferior de los datos. Por ejemplo, para el estado 'CL', la mediana es 1435 días.

- **75%:** El percentil 75, también conocido como el tercer cuartil. Indica el valor por debajo del cual cae el 75% de los datos. Por ejemplo, para el estado 'D', el tercer cuartil es aproximadamente 2071 días.
- **max:** El valor máximo del tiempo transcurrido desde el registro para cada estado del paciente. Por ejemplo, para el estado 'C', el valor máximo es 4795 días.

En resumen, estas estadísticas nos ayudan a comprender la distribución y la tendencia central de la variable 'N\_Days' para diferentes estados del paciente.

## 2. Impacto del tipo de medicamento administrado en el estado del paciente:

```
# Calcular la distribución de estados del paciente para cada tipo de medicamento administrado
status_by_drug = df.groupby('Drug')['Status'].value_counts(normalize=True)
```

```
# Imprimir la distribución de estados del paciente para cada tipo de medicamento administrado
print("Distribución de estados del paciente por tipo de medicamento administrado:")
print(status_by_drug)
```

```
↗ Distribución de estados del paciente por tipo de medicamento administrado:
```

Drug	Status	
D-penicillamine	C	0.556818
	D	0.382576
	CL	0.060606
Placebo	C	0.551948
	D	0.389610
	CL	0.058442

Name: proportion, dtype: float64

Estos resultados muestran la distribución de los estados del paciente (C, D y CL) para cada tipo de medicamento administrado (D-penicillamine y placebo). Aquí está su interpretación:

- **D-penicillamine:**
  - El 55.68% de los pacientes que recibieron D-penicillamine tienen un estado censurado (C).
  - El 38.26% de los pacientes que recibieron D-penicillamine fallecieron (D).
  - El 6.06% de los pacientes que recibieron D-penicillamine tienen un estado censurado debido al trasplante hepático (CL).
- **Placebo:**
  - El 55.19% de los pacientes que recibieron placebo tienen un estado censurado (C).
  - El 38.96% de los pacientes que recibieron placebo fallecieron (D).
  - El 5.84% de los pacientes que recibieron placebo tienen un estado censurado debido al trasplante hepático (CL).

Basándonos en los resultados, parece que el tipo de medicamento administrado no influyó significativamente en el estado final del paciente. La distribución de los estados del paciente (C, D y CL) es similar entre los pacientes que recibieron D-penicillamine y los que recibieron placebo. Ambos grupos tienen una proporción cercana de pacientes con estado censurado (C), fallecidos (D) y censurados debido al trasplante hepático (CL). Por lo tanto, en base a los datos proporcionados, no parece haber una diferencia clara en los resultados en función del tipo de medicamento administrado.

## 3. Correlación entre la edad del paciente y su estado de salud:

```
from scipy.stats import f_oneway
```

```
# Filtrar el DataFrame por cada estado de salud
status_C = df[df['Status'] == 'C']['Age']
status_CL = df[df['Status'] == 'CL']['Age']
status_D = df[df['Status'] == 'D']['Age']
```

```
# Realizar la prueba ANOVA
f_statistic, p_value = f_oneway(status_C, status_CL, status_D)
```

```
# Imprimir los resultados
print("Estadística F:", f_statistic)
print("Valor p:", p_value)
```

```
if p_value < 0.05:
    print("Hay una diferencia significativa en la edad promedio entre los diferentes estados de salud.")
else:
```

```
print("No hay una diferencia significativa en la edad promedio entre los diferentes estados de salud.")
```

```
↗ Estadística F: 20.134266051199326
Valor p: 4.512801037966663e-09
Hay una diferencia significativa en la edad promedio entre los diferentes estados de salud.
```

El valor de la estadística F obtenido es aproximadamente 20.13 y el valor p es extremadamente pequeño, alrededor de  $4.51 \times 10^{-9}$  (o 0.0000000045128).

Esto significa que existe una diferencia significativa en la edad promedio entre los diferentes estados de salud. En otras palabras, la edad del paciente está relacionada de manera significativa con su estado de salud.

Con base en estos resultados, podemos concluir que la edad del paciente juega un papel importante en su estado de salud. Específicamente, la prueba ANOVA indica que hay diferencias significativas en la edad promedio entre los pacientes con diferentes estados de salud (D, C y CL). Esto podría ser útil para entender cómo la edad influye en la progresión y el resultado de la enfermedad hepática en los pacientes con cirrosis.

#### 4. Diferencias en el estado de los pacientes según su género:

```
from scipy.stats import chi2_contingency

# Crear una tabla de contingencia entre el género y el estado de salud
contingency_table = pd.crosstab(df['Sex'], df['Status'])

# Realizar la prueba de chi-cuadrado
chi2, p_value, dof, expected = chi2_contingency(contingency_table)

# Imprimir el valor de chi-cuadrado y el valor p
print("Valor de chi-cuadrado:", chi2)
print("Valor p:", p_value)

# Interpretar los resultados
if p_value < 0.05:
    print("Hay una asociación significativa entre el género y el estado de salud.")
    if contingency_table.loc['M', 'D'] > contingency_table.loc['F', 'D']:
        print("Los hombres son más propensos a tener un estado de salud 'D'.")
    elif contingency_table.loc['M', 'D'] < contingency_table.loc['F', 'D']:
        print("Las mujeres son más propensas a tener un estado de salud 'D'.")
    else:
        print("No hay diferencias significativas entre géneros en cuanto al estado de salud 'D'.")
else:
    print("No hay una asociación significativa entre el género y el estado de salud.")

↗ Valor de chi-cuadrado: 5.858291641994132
Valor p: 0.053442668259367894
No hay una asociación significativa entre el género y el estado de salud.
```

El valor de chi-cuadrado y el valor p son resultados de la prueba de chi-cuadrado realizada para determinar si hay una asociación significativa entre el género y el estado de salud de los pacientes.

- El valor de chi-cuadrado (5.858) es una medida de cuánto difieren los valores observados de los esperados bajo la hipótesis nula de independencia entre el género y el estado de salud. Cuanto mayor sea el valor de chi-cuadrado, mayor será la discrepancia entre los datos observados y esperados, lo que indica una mayor asociación entre las variables.
- El valor p (0.053) es la probabilidad de obtener un valor de chi-cuadrado igual o mayor al observado si la hipótesis nula fuera verdadera. En otras palabras, es la probabilidad de que la asociación observada entre el género y el estado de salud sea debido al azar. Si el valor p es menor que un umbral predefinido (comúnmente 0.05), se considera que hay una asociación significativa entre las variables.

En este caso, el valor p es aproximadamente 0.053, lo que indica que hay una probabilidad del 5.3% de obtener una asociación entre el género y el estado de salud igual o más extrema que la observada si la asociación fuera puramente al azar. Como este valor p es mayor que el umbral comúnmente aceptado de 0.05, no hay suficiente evidencia para rechazar la hipótesis nula de que no hay asociación significativa entre el género y el estado de salud. Por lo tanto, se concluye que no hay una asociación significativa entre el género y el estado de salud en este conjunto de datos.

#### 5. Impacto de ciertas condiciones médicas en la supervivencia de los pacientes:



```

from scipy.stats import chi2_contingency

# Crear una tabla de contingencia entre las condiciones médicas y el estado de los pacientes
contingency_table = pd.crosstab(index=df['Status'], columns=[df['Ascites'], df['Hepatomegaly'], df['Spiders'], df['Edema']])

# Ejecutar la prueba de chi-cuadrado
chi2, p, dof, expected = chi2_contingency(contingency_table)

# Imprimir los resultados
print("Valor de chi-cuadrado:", chi2)
print("Valor p:", p)

if p < 0.05:
    print("Hay una asociación significativa entre las condiciones médicas y el estado de los pacientes.")
else:
    print("No hay una asociación significativa entre las condiciones médicas y el estado de los pacientes.")

↩ Valor de chi-cuadrado: 81.59296565949666
Valor p: 2.160496432248983e-05
Hay una asociación significativa entre las condiciones médicas y el estado de los pacientes.

```

Los resultados muestran que el valor de chi-cuadrado es 81.59 y el valor p es 2.16e-05. Aquí está la interpretación:

- **Valor de chi-cuadrado:** El valor de chi-cuadrado es una medida de la discrepancia entre los datos observados en la tabla de contingencia y los datos que se esperarían si no hubiera asociación entre las variables. En este caso, un valor alto de chi-cuadrado sugiere que existe una gran discrepancia entre los datos observados y los datos esperados bajo la hipótesis nula de independencia.
- **Valor p:** El valor p es la probabilidad de observar un valor de chi-cuadrado al menos tan extremo como el valor observado, si la hipótesis nula de independencia es verdadera. En este caso, el valor p es extremadamente pequeño (2.16e-05), lo que indica que es altamente improbable obtener un valor de chi-cuadrado tan grande bajo la hipótesis nula.

Por lo tanto, con un valor p tan pequeño, se rechaza la hipótesis nula de independencia entre las condiciones médicas y el estado de los pacientes. Esto significa que hay una asociación significativa entre las condiciones médicas (Ascites, Hepatomegaly, Spiders y Edema) y el estado de los pacientes (C, CL, D). En otras palabras, la presencia de estas condiciones médicas afecta significativamente el estado de salud de los pacientes.

## 6. Relación entre los niveles de biomarcadores y el estado de los pacientes:

```

# Lista de todos los biomarcadores
biomarkers = ['Bilirubin', 'Cholesterol', 'Albumin', 'Copper', 'Alk_Phos', 'SGOT', 'Tryglicerides', 'Platelets', 'Prothrombin']

# Realizar ANOVA para cada biomarcador y comparar las distribuciones entre los diferentes estados de los pacientes
for biomarker in biomarkers:
    anova_result = f_oneway(df[df['Status'] == 'C'][biomarker],
                             df[df['Status'] == 'CL'][biomarker],
                             df[df['Status'] == 'D'][biomarker])
    print("Resultado del ANOVA para", biomarker, ":", anova_result)

↩ Resultado del ANOVA para Bilirubin : F_onewayResult(statistic=46.995280799342794, pvalue=4.0060030337023026e-19)
Resultado del ANOVA para Cholesterol : F_onewayResult(statistic=5.886448975602688, pvalue=0.00301395971740692)
Resultado del ANOVA para Albumin : F_onewayResult(statistic=15.260927571507652, pvalue=4.024017053712767e-07)
Resultado del ANOVA para Copper : F_onewayResult(statistic=27.145803312454966, pvalue=8.329171395554437e-12)
Resultado del ANOVA para Alk_Phos : F_onewayResult(statistic=8.981623453096693, pvalue=0.00015184166778389274)
Resultado del ANOVA para SGOT : F_onewayResult(statistic=14.575889622594048, pvalue=7.624533554503278e-07)
Resultado del ANOVA para Tryglicerides : F_onewayResult(statistic=6.691201978285902, pvalue=0.001380124171808719)
Resultado del ANOVA para Platelets : F_onewayResult(statistic=5.671067970019371, pvalue=0.0037165656882138165)
Resultado del ANOVA para Prothrombin : F_onewayResult(statistic=29.94521373040676, pvalue=7.109619597911414e-13)

```

Los resultados del ANOVA muestran la significancia estadística de la diferencia en los niveles de biomarcadores entre los diferentes estados de los pacientes. Aquí hay una interpretación de los resultados en términos de la pregunta inicial:

1. **Bilirrubina:** Hay una diferencia significativa en los niveles de bilirrubina entre los pacientes en los diferentes estados de salud (C, CL y D), con un valor p muy pequeño (<0.05). Esto sugiere que los niveles de bilirrubina están asociados de manera significativa con el estado de salud del paciente.
2. **Colesterol:** También hay una diferencia significativa en los niveles de colesterol entre los pacientes en los diferentes estados de salud, con un valor p de 0.003. Aunque este valor p es mayor que 0.05, aún indica una asociación significativa.
3. **Otros biomarcadores:** Todos los otros biomarcadores (Albumin, Copper, Alk\_Phos, SGOT, Tryglicerides, Platelets y Prothrombin) también muestran diferencias significativas en sus niveles entre los diferentes estados de los pacientes, con valores p muy pequeños (<0.05).

En resumen, estos resultados indican que los niveles de varios biomarcadores están asociados de manera significativa con el estado de salud del paciente. Esto sugiere que estos biomarcadores podrían ser útiles para predecir o entender el estado de salud de los pacientes en función de sus niveles.

## 7. Impacto del estadio histológico de la enfermedad en la supervivencia de los pacientes:

```
from scipy.stats import chi2_contingency

# Crear una tabla de contingencia entre el estadio histológico de la enfermedad y el estado de los pacientes
contingency_table = pd.crosstab(index=df['Status'], columns=df['Stage'])

# Ejecutar la prueba de chi-cuadrado
chi2, p, dof, expected = chi2_contingency(contingency_table)

# Imprimir los resultados
print("Valor de chi-cuadrado:", chi2)
print("Valor p:", p)

if p < 0.05:
    print("Hay una asociación significativa entre el estadio histológico de la enfermedad y el estado de los pacientes.")
else:
    print("No hay una asociación significativa entre el estadio histológico de la enfermedad y el estado de los pacientes.")
```

↗ Valor de chi-cuadrado: 50.13785630363444  
 Valor p: 3.8448806725631566e-08  
 Hay una asociación significativa entre el estadio histológico de la enfermedad y el estado de los pacientes.

El valor de chi-cuadrado obtenido es 50.14 y el valor p es 3.84e-08.

El valor de chi-cuadrado es una medida estadística que indica la discrepancia entre las frecuencias observadas en los datos y las frecuencias esperadas bajo la hipótesis nula. Cuanto mayor sea el valor de chi-cuadrado, mayor será la discrepancia entre los datos observados y los esperados, lo que sugiere una asociación más fuerte entre las variables.

El valor p es la probabilidad de observar un valor de chi-cuadrado al menos tan extremo como el valor observado bajo la hipótesis nula. Un valor p pequeño sugiere que es poco probable que la asociación observada entre las variables sea debida al azar.

En este caso, el valor de chi-cuadrado es grande y el valor p es muy pequeño (menor que 0.05), lo que indica que la asociación entre el estadio histológico de la enfermedad y el estado de los pacientes es significativa. Por lo tanto, podemos concluir que hay una asociación significativa entre el estadio histológico de la enfermedad y el estado de los pacientes.

## ✓ Compartir

Para el campo del análisis de datos, compartir nuestros hallazgos es una parte fundamental del proceso. Comunicar de manera efectiva los resultados de nuestros análisis no solo nos permite validar nuestras conclusiones, sino que también contribuye al avance del conocimiento en la comunidad científica o a la toma de decisiones basadas en datos.

### Importancia de compartir descubrimientos

- **Validación de resultados:** Compartir nuestros hallazgos nos permite someter nuestras conclusiones a la revisión de expertos y colegas, lo que ayuda a validar la calidad y la robustez de nuestros análisis.
- **Replicabilidad y transparencia:** Al compartir nuestros datos y metodologías, otros investigadores tienen la oportunidad de replicar nuestros análisis, lo que aumenta la transparencia y la confiabilidad de la investigación.
- **Colaboración y retroalimentación:** Compartir nuestros hallazgos fomenta la colaboración entre investigadores y permite recibir retroalimentación constructiva, lo que puede enriquecer nuestro trabajo y llevar a nuevas ideas y descubrimientos.
- **Impacto social y científico:** La divulgación de resultados puede tener un impacto significativo en la sociedad al informar decisiones políticas, médicas o sociales basadas en evidencia científica.

### Experimento sobre Cirrosis Hepática

En el contexto específico de un experimento sobre cirrosis hepática, compartir nuestros descubrimientos es crucial por varias razones:

- **Mejor Comprensión de la Enfermedad:** Compartir datos y análisis relacionados con la cirrosis hepática puede ayudar a mejorar nuestra comprensión de la enfermedad, sus factores de riesgo y su progresión.

- **Desarrollo de Tratamientos y Políticas de Salud:** Los hallazgos obtenidos pueden proporcionar información valiosa para el desarrollo de tratamientos más efectivos, así como para la implementación de políticas de salud pública destinadas a prevenir y controlar la cirrosis hepática.
- **Concienciación y Prevención:** Compartir información sobre los factores asociados con la cirrosis hepática puede contribuir a la concienciación pública sobre la importancia de hábitos de vida saludables y la detección temprana de la enfermedad.
- **Avance Científico:** Al compartir nuestros datos y análisis con la comunidad científica, podemos contribuir al avance del conocimiento en el campo de la hepatología y la medicina en general.

La gráfica de pie, también conocida como gráfica circular o gráfica de pastel, es una forma de visualización que muestra proporciones o porcentajes relativos utilizando sectores circulares. Cada sector representa una categoría o grupo de datos y el tamaño del sector es proporcional a la cantidad de datos en ese grupo. Son útiles para mostrar la distribución de una variable categórica en un conjunto de datos y facilitan la comparación visual de las proporciones entre categorías.

Compararemos las personas en cada Status de la siguiente manera:

```
import matplotlib.pyplot as plt

# Contar la cantidad de pacientes en cada estado
status_counts = df['Status'].value_counts()

# Colores personalizados para cada estado
colors = ['#FF9999', '#66B2FF', '#99FF99'] # Puedes cambiar los colores según tus preferencias

# Etiquetas para cada estado
status_labels = ['C', 'D', 'CL']

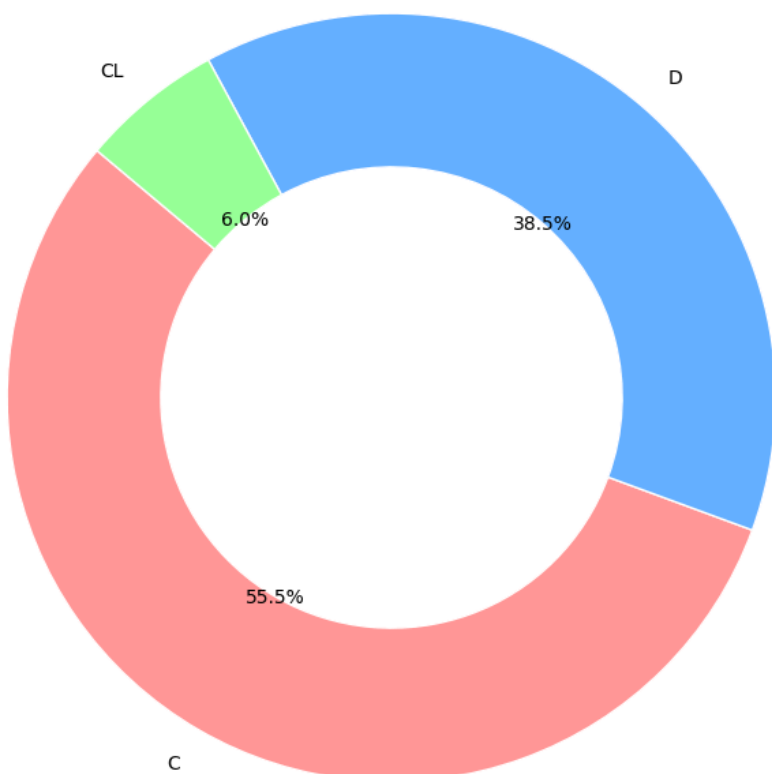
# Crear la gráfica de pastel
plt.figure(figsize=(8, 8))
plt.pie(status_counts, labels=status_labels, colors=colors, autopct='%1.1f%%', startangle=140, wedgeprops=dict(width=0.4, edgecolor='black'))

# Añadir título
plt.title('Distribución de Pacientes por Estado', fontsize=16)

# Mostrar la gráfica
plt.axis('equal') # Para asegurar que el gráfico sea circular
plt.show()
```



## Distribución de Pacientes por Estado



- **Importar bibliotecas:** Importamos las bibliotecas necesarias para graficar: `seaborn` para el estilo y `matplotlib.pyplot` para la creación de la gráfica.
- **Configurar el estilo de Seaborn:** Establecemos el estilo de la gráfica utilizando `sns.set_style("whitegrid")` para un fondo blanco con una cuadrícula.
- **Contar la cantidad de personas por estado:** Usamos `value_counts()` para contar cuántas personas pertenecen a cada estado en la columna 'Status' y almacenamos los resultados en `status_counts`.
- **Definir colores personalizados:** Creamos una paleta de colores utilizando `sns.color_palette('pastel')`. La longitud de la paleta se ajusta para coincidir con la cantidad de estados. Esto garantiza que cada estado tenga un color único.
- **Crear la gráfica de pastel:** Utilizamos `plt.pie()` para generar la gráfica de pastel. Pasamos los conteos de estado, las etiquetas personalizadas y los colores correspondientes como argumentos. `autopct='%1.1f%%'` agrega etiquetas de porcentaje a cada sector y `startangle=140` gira la gráfica para que comience en un ángulo específico.
- **Añadir título:** Agregamos un título a la gráfica utilizando `plt.title()`.

### 1. Relación entre el tiempo transcurrido desde el registro y el estado del paciente:

La gráfica de dispersión es una representación visual que muestra la relación entre dos variables numéricas. Cada punto en la gráfica representa una observación en el conjunto de datos y está ubicado en un plano cartesiano, donde uno de los ejes corresponde a una variable y el otro eje corresponde a la otra variable. La posición de cada punto en el gráfico indica los valores de las dos variables para esa observación en particular.

En el contexto de este análisis, la gráfica de dispersión se utilizará para visualizar cómo se distribuyen los pacientes en función del tiempo transcurrido desde el registro y su estado. En el eje X, representaremos el tiempo transcurrido desde el registro, mientras que en el eje Y representaremos el estado del paciente. Cada punto en la gráfica corresponderá a un paciente individual y su posición mostrará cuánto tiempo ha pasado desde el registro y cuál es su estado.

Al visualizar esta relación en una gráfica de dispersión, podremos identificar patrones o tendencias, como si ciertos estados están asociados con tiempos más largos o más cortos desde el registro. Esto nos ayudará a comprender mejor cómo el tiempo transcurrido está relacionado con el estado del paciente y nos proporcionará información importante para nuestro análisis.

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

# Ignorar temporalmente las advertencias de desbordamiento
warnings.filterwarnings("ignore", category=RuntimeWarning)

# Configuración de estilo de Seaborn
sns.set(style="whitegrid")

# Convertir la columna 'Status' a valores numéricos
status_mapping = {'C': 0, 'CL': 1, 'D': 2}
df['Status'] = df['Status'].map(status_mapping)

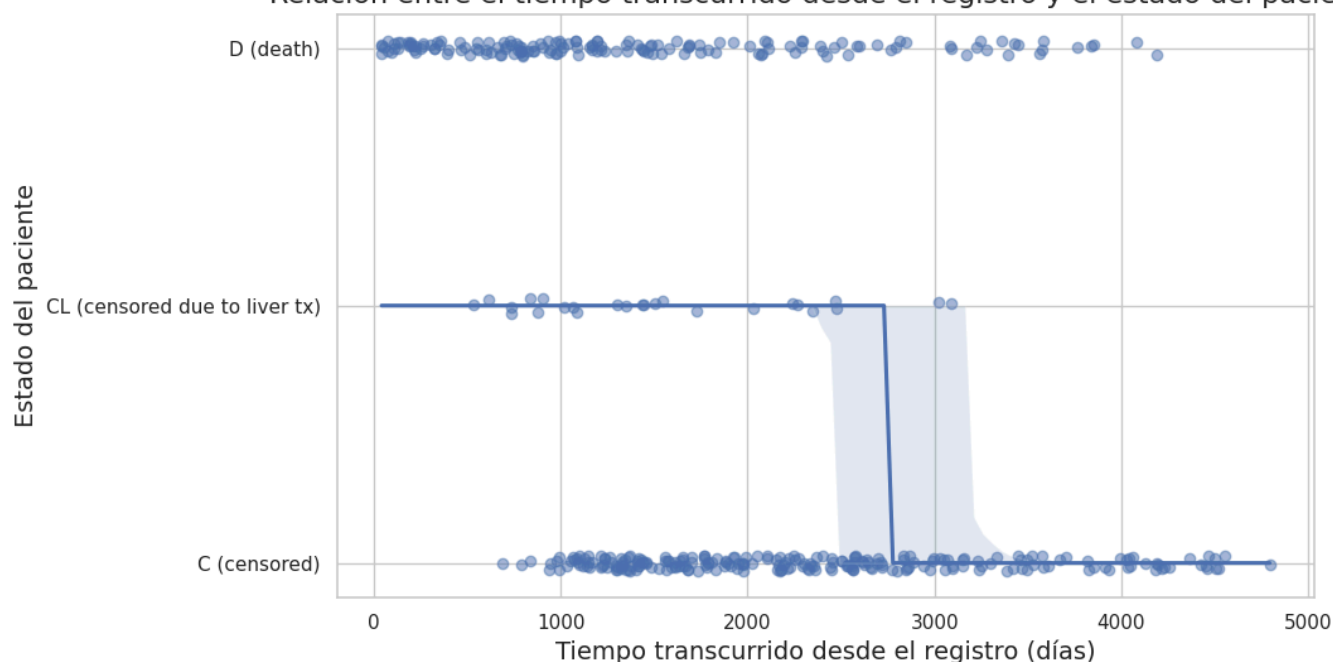
# Crear un gráfico de dispersión con una línea de regresión
plt.figure(figsize=(10, 6))
sns.regplot(x='N_Days', y='Status', data=df, logistic=True, y_jitter=0.03, scatter_kws={'alpha':0.5})

# Personalizar el gráfico
plt.title('Relación entre el tiempo transcurrido desde el registro y el estado del paciente', fontsize=16)
plt.xlabel('Tiempo transcurrido desde el registro (días)', fontsize=14)
plt.ylabel('Estado del paciente', fontsize=14)
plt.yticks([0, 1, 2], ['C (censored)', 'CL (censored due to liver tx)', 'D (death)'])

warnings.filterwarnings("default", category=RuntimeWarning)
```



Relación entre el tiempo transcurrido desde el registro y el estado del paciente



Aquí está la explicación línea por línea:

- `import seaborn as sns`: Importa la librería Seaborn, que se utiliza para crear gráficos estadísticos más atractivos y informativos.
- `import matplotlib.pyplot as plt`: Importa la sublibrería Pyplot de Matplotlib, que se utiliza para crear y personalizar gráficos.
- `import warnings`: Importa el módulo de advertencias de Python, que permite controlar el comportamiento de las advertencias en el código.
- `warnings.filterwarnings("ignore", category=RuntimeWarning)`: Ignora temporalmente las advertencias de tipo `RuntimeWarning`. Esto es útil si hay advertencias que no son críticas para el análisis y se desean ocultar temporalmente.
- `sns.set(style="whitegrid")`: Establece el estilo de Seaborn en "whitegrid", que agrega líneas de cuadrícula blancas al fondo del gráfico para una mejor visualización de los datos.
- `status_mapping = {'C': 0, 'CL': 1, 'D': 2}`: Crea un diccionario que asigna cada estado de paciente ('C', 'CL', 'D') a un valor numérico (0, 1, 2).

- `df['Status'] = df['Status'].map(status_mapping)`: Mapea los valores de la columna 'Status' del DataFrame `df` según el diccionario `status_mapping`, convirtiendo así los estados del paciente en valores numéricos.
- `plt.figure(figsize=(10, 6))`: Crea una nueva figura de tamaño 10x6 pulgadas para el gráfico.
- `sns.regplot(x='N_Days', y='Status', data=df, logistic=True, y_jitter=0.03, scatter_kws={'alpha':0.5})`: Crea un gráfico de dispersión con una línea de regresión logística, donde el eje x representa el tiempo transcurrido desde el registro y el eje y representa el estado del paciente. El parámetro `y_jitter` agrega un ligero desplazamiento aleatorio a los puntos de dispersión en el eje y para evitar la superposición total de los puntos con la misma coordenada y. El parámetro `scatter_kws` se utiliza para personalizar la apariencia de los puntos de dispersión, en este caso, reduciendo su transparencia.
- `plt.title('Relación entre el tiempo transcurrido desde el registro y el estado del paciente', fontsize=16)`: Establece el título del gráfico con un tamaño de fuente de 16 puntos.
- `plt.xlabel('Tiempo transcurrido desde el registro (días)', fontsize=14)`: Establece la etiqueta del eje x con un tamaño de fuente de 14 puntos.
- `plt.ylabel('Estado del paciente', fontsize=14)`: Establece la etiqueta del eje y con un tamaño de fuente de 14 puntos.
- `plt.yticks([0, 1, 2], ['C (censored)', 'CL (censored due to liver tx)', 'D (death)'])`: Cambia las etiquetas de los ticks del eje y para representar los estados del paciente con su significado, utilizando las correspondientes conversiones del diccionario `status_mapping`.
- `warnings.filterwarnings("default", category=RuntimeWarning)`: Restaura el comportamiento predeterminado de las advertencias `RuntimeWarning` después de ejecutar el código problemático. Esto asegura que las advertencias vuelvan a aparecer normalmente en el entorno de ejecución.

## 2. Impacto del tipo de medicamento administrado en el estado del paciente:

La gráfica de barras es una herramienta visual comúnmente utilizada para representar la distribución de una variable categórica. Consiste en barras rectangulares que representan la frecuencia o proporción de observaciones en cada categoría. Es especialmente útil para comparar diferentes categorías entre sí.

En este contexto, la gráfica de barras se utilizó para visualizar el impacto del tipo de medicamento administrado en el estado del paciente. La variable categórica `Drug` representa los dos tipos de medicamentos administrados: D-penicilamina y placebo. La variable categórica `Status` indica el estado del paciente, que puede ser C (censurado), CL (censurado debido a trasplante de hígado) o D (fallecido).

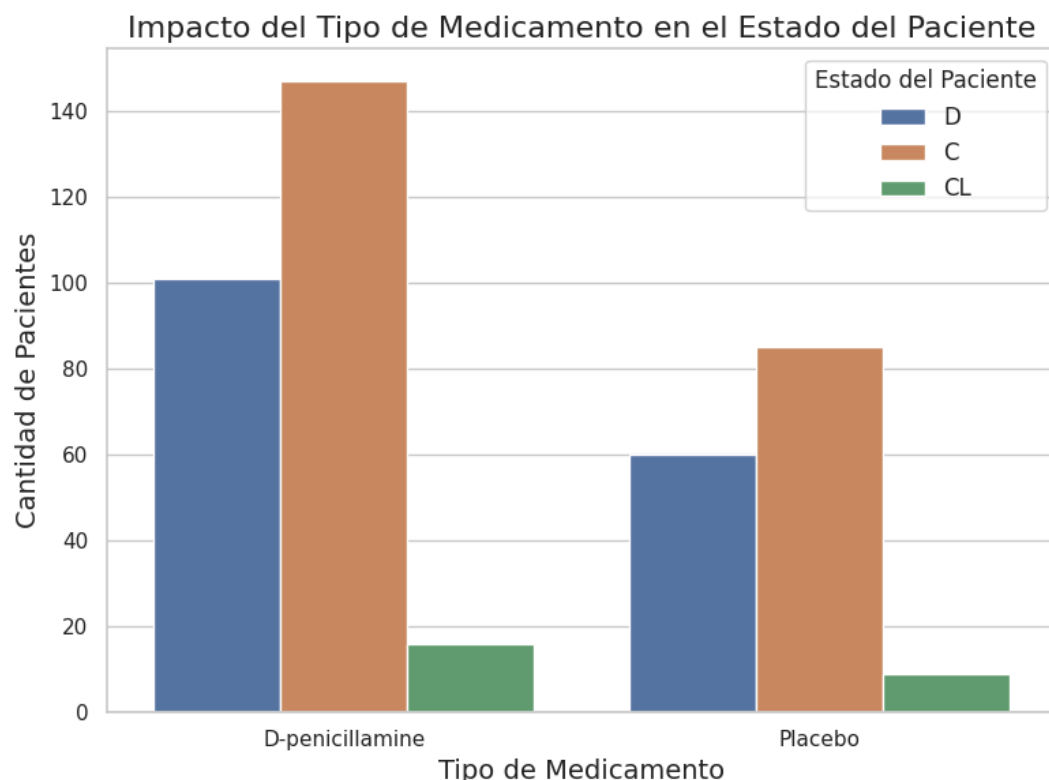
```
import seaborn as sns
import matplotlib.pyplot as plt

# Mapear los valores numéricos a las etiquetas correspondientes
status_mapping = {0: 'C', 1: 'CL', 2: 'D'}
df['Status'] = df['Status'].map(status_mapping)

# Crear un gráfico de barras utilizando Seaborn
plt.figure(figsize=(8, 6))
sns.countplot(x='Drug', hue='Status', data=df)

# Personalizar el gráfico
plt.title('Impacto del Tipo de Medicamento en el Estado del Paciente', fontsize=16)
plt.xlabel('Tipo de Medicamento', fontsize=14)
plt.ylabel('Cantidad de Pacientes', fontsize=14)
plt.legend(title='Estado del Paciente', fontsize=12)

# Mostrar el gráfico
plt.tight_layout()
```



- `import seaborn as sns`: Importa la biblioteca Seaborn, que se utiliza para crear visualizaciones estadísticas atractivas y informativas.
- `import matplotlib.pyplot as plt`: Importa la biblioteca Matplotlib, que se utiliza para personalizar y mostrar las visualizaciones creadas con Seaborn.
- `plt.figure(figsize=(8, 6))`: Crea una nueva figura de Matplotlib con un tamaño de 8x6 pulgadas.
- `sns.countplot(x='Drug', hue='Status', data=df)`: Crea un gráfico de barras utilizando Seaborn. La variable `Drug` se muestra en el eje X y se divide por color según la variable `Status`. Los datos se toman del DataFrame `df`.
- `plt.title('Impacto del Tipo de Medicamento en el Estado del Paciente', fontsize=16)`: Establece el título del gráfico.
- `plt.xlabel('Tipo de Medicamento', fontsize=14)`: Etiqueta el eje X con "Tipo de Medicamento".
- `plt.ylabel('Cantidad de Pacientes', fontsize=14)`: Etiqueta el eje Y con "Cantidad de Pacientes".
- `plt.legend(title='Estado del Paciente', fontsize=12)`: Muestra una leyenda que indica los diferentes estados del paciente.
- `plt.tight_layout()`: Ajusta automáticamente los márgenes del gráfico para que quepan todos los elementos.

### 3. Correlación entre la edad del paciente y su estado de salud:

Las gráficas de violín son una representación visual de la distribución de datos y se utilizan comúnmente en análisis estadísticos y exploratorios. Cada violín muestra la distribución de los datos para una variable categórica, como el estado de salud en este caso, y permite comparar visualmente las distribuciones entre diferentes categorías.

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

# Define los colores personalizados para cada estado de salud
colors = {'C': 'skyblue', 'CL': 'lightgreen', 'D': 'salmon'}

# Crea un gráfico de violín con colores personalizados
plt.figure(figsize=(10, 6))
sns.violinplot(x='Status', y='Age', data=df, palette=colors)

# Etiquetas y título
plt.xlabel('Estado del Paciente')
plt.ylabel('Edad del Paciente')
plt.title('Relación entre la Edad del Paciente y su Estado de Salud')
```

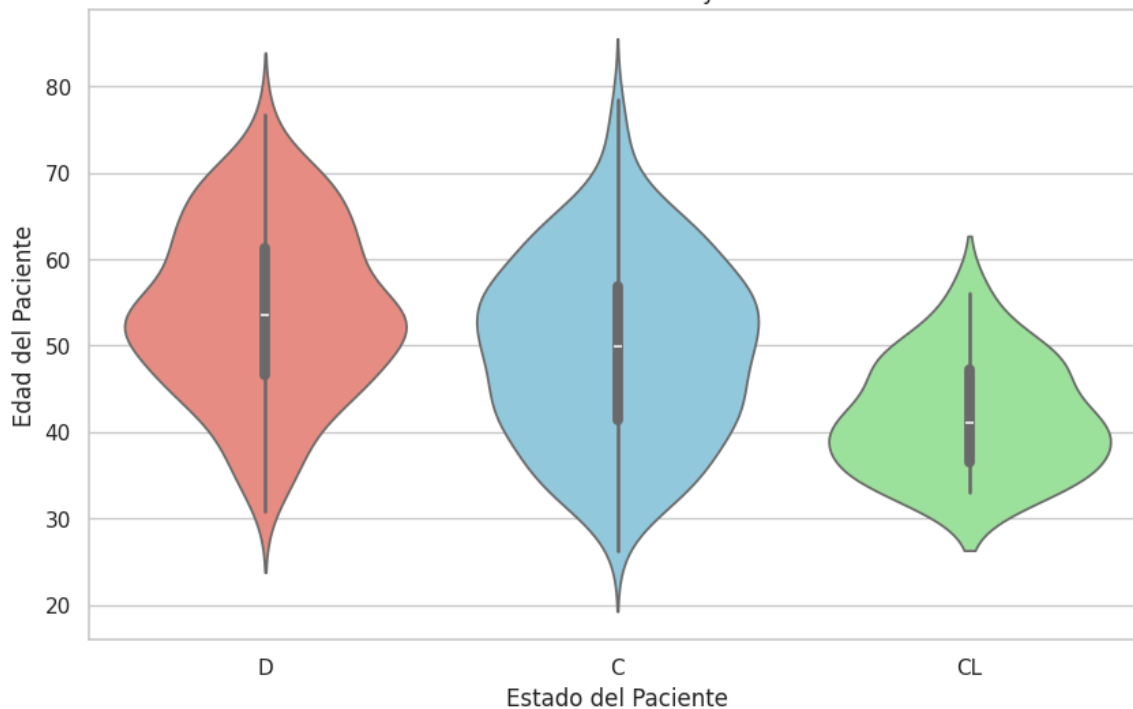
```
# Muestra el gráfico
plt.show()
```

```
<ipython-input-35-6da23b45d44d>:9: FutureWarning:
```

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` ;

```
sns.violinplot(x='Status', y='Age', data=df, palette=colors)
```

Relación entre la Edad del Paciente y su Estado de Salud



- `import seaborn as sns`: Importa la biblioteca Seaborn, que se utiliza para crear visualizaciones estadísticas atractivas e informativas.
- `import matplotlib.pyplot as plt`: Importa la biblioteca Matplotlib, que se utiliza para personalizar y mostrar las visualizaciones creadas con Seaborn.
- `plt.figure(figsize=(8, 6))`: Crea una nueva figura de Matplotlib con un tamaño de 8x6 pulgadas.
- `sns.countplot(x='Drug', hue='Status', data=df)`: Crea un gráfico de barras utilizando Seaborn. La variable `Drug` se muestra en el eje X y se divide por color según la variable `Status`. Los datos se toman del DataFrame `df`.
- `plt.title('Impacto del Tipo de Medicamento en el Estado del Paciente', fontsize=16)`: Establece el título del gráfico.
- `plt.xlabel('Tipo de Medicamento', fontsize=14)`: Etiqueta el eje X con "Tipo de Medicamento".
- `plt.ylabel('Cantidad de Pacientes', fontsize=14)`: Etiqueta el eje Y con "Cantidad de Pacientes".
- `plt.legend(title='Estado del Paciente', fontsize=12)`: Muestra una leyenda que indica los diferentes estados del paciente.
- `plt.tight_layout()`: Ajusta automáticamente los márgenes del gráfico para que quepan todos los elementos.
- `plt.show()`: Muestra el gráfico.

#### 4. Diferencias en el estado de los pacientes según su género

El gráfico de radar es una herramienta visual que se utiliza para representar múltiples variables categóricas de forma simultánea en un solo gráfico circular. Cada variable se representa por un eje que parte del centro del círculo y se extiende hacia afuera, formando un ángulo igual. Las categorías de cada variable se colocan a lo largo de estos ejes, y el valor de cada categoría se representa por la distancia desde el centro del círculo.

En este ejemplo, el gráfico de radar se utiliza para comparar las diferencias en el estado de los pacientes según su género. Cada categoría en el gráfico representa un estado del paciente (C, CL, D), y los ejes radiales muestran la cantidad de pacientes en cada estado. Se utilizan dos áreas coloreadas para representar los datos de hombres y mujeres, lo que permite una fácil comparación entre ambos grupos.

```
import matplotlib.pyplot as plt
import numpy as np
```



```
# Variables para cada género
male_data = df[df['Sex'] == 'M']['Status'].value_counts().tolist()
female_data = df[df['Sex'] == 'F']['Status'].value_counts().tolist()

# Número de variables (estados de los pacientes)
labels = ['C', 'D', 'CL']

# Número de variables
num_vars = len(labels)

# Ángulos para cada variable
angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()

# Asegurar que el gráfico sea cerrado
male_data += male_data[:1]
female_data += female_data[:1]
angles += angles[:1]

# Crear el gráfico
fig, ax = plt.subplots(figsize=(8, 8), subplot_kw=dict(polar=True))
ax.fill(angles, male_data, color='blue', alpha=0.25)
ax.fill(angles, female_data, color='pink', alpha=0.25)

# Establecer las etiquetas de las variables
ax.set_yticklabels([])

# Añadir las etiquetas de las variables
ax.set_xticks(angles[:-1])
ax.set_xticklabels(labels)

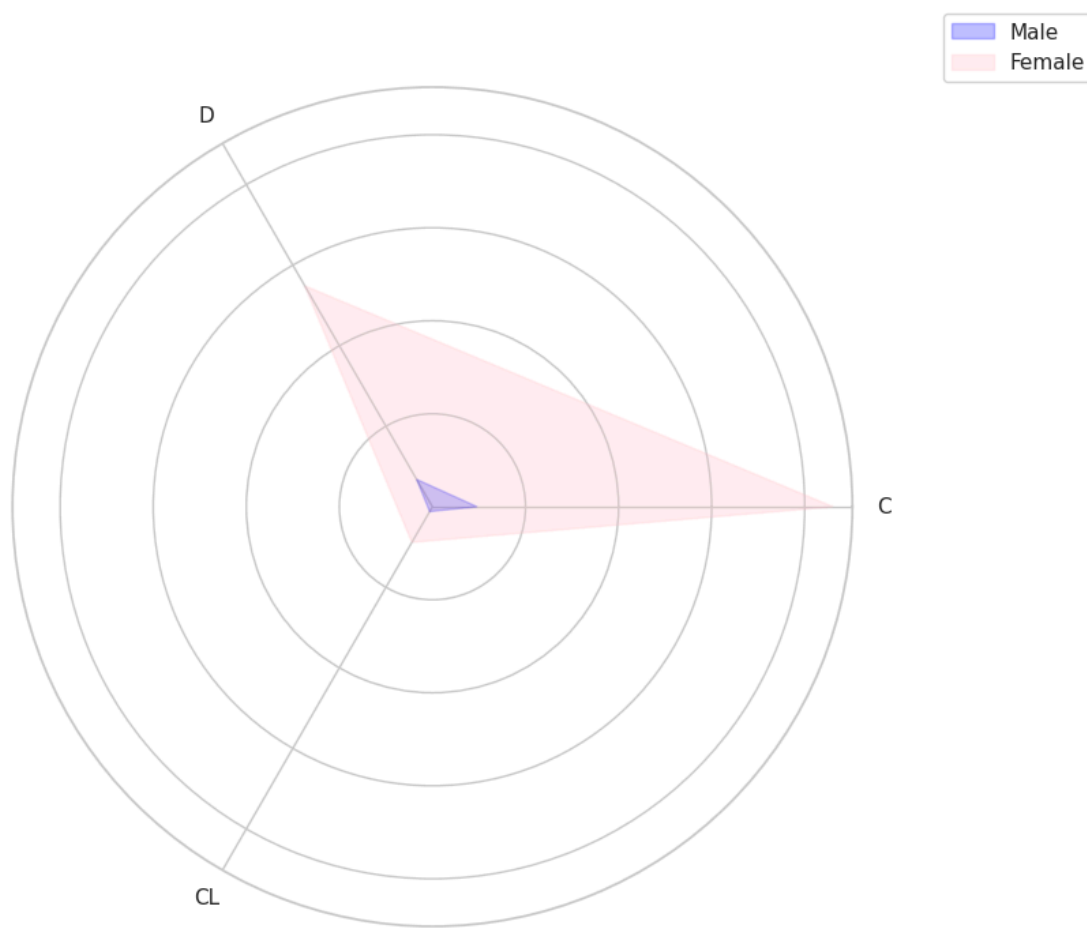
# Añadir título
plt.title('Diferencias en el Estado de los Pacientes según su Género', size=16, color='black', y=1.1)

# Añadir leyenda
plt.legend(['Male', 'Female'], loc='upper right', bbox_to_anchor=(1.3, 1.1))

# Mostrar el gráfico
plt.show()
```



## Diferencias en el Estado de los Pacientes según su Género



- `import matplotlib.pyplot as plt`: Importa la librería Matplotlib, que se utiliza para crear visualizaciones gráficas.
- `import numpy as np`: Importa la librería NumPy, que proporciona funciones matemáticas y numéricas para trabajar con arreglos.
- Se extraen los datos del DataFrame para hombres y mujeres, y se cuentan los estados de los pacientes.
- Se definen las etiquetas y los ángulos para cada variable (estado del paciente). Las etiquetas representan los estados del paciente (C, CL, D), y los ángulos se distribuyen uniformemente alrededor del círculo.
- Se asegura que el gráfico sea cerrado, agregando el primer valor al final de los datos y ángulos. Esto es necesario para cerrar el polígono que representa el gráfico.
- Se crea el gráfico polar utilizando `plt.subplots()` con el parámetro `subplot_kw=dict(polar=True)`, lo que indica que se trata de un gráfico polar.
- Se rellenan las áreas correspondientes a los datos de hombres y mujeres utilizando `ax.fill()`. Cada área está coloreada de manera diferente (azul para hombres y rosa para mujeres) y se define con respecto a los ángulos y los datos de estado de los pacientes.
- Se establecen las etiquetas de las variables en el eje X utilizando `ax.set_xticks()` y `ax.set_xticklabels()`. Las etiquetas se colocan a lo largo de los ángulos definidos anteriormente.
- Se añade un título al gráfico utilizando `plt.title()`.
- Se añade una leyenda que indica qué color representa a cada género utilizando `plt.legend()`.
- Finalmente, se muestra el gráfico utilizando `plt.show()`.

### 5. Impacto de ciertas condiciones médicas en la supervivencia de los pacientes:

El gráfico de áreas apiladas es una visualización que muestra la distribución de una variable a lo largo del tiempo o de diferentes categorías. En este tipo de gráfico, las áreas se superponen unas sobre otras, y la altura de cada área en un punto dado representa la cantidad total en ese punto. Se utiliza para comparar la contribución relativa de diferentes partes a un todo y para observar cómo cambian estas contribuciones con el tiempo o entre diferentes categorías.

```
import matplotlib.pyplot as plt

# Filtrar el DataFrame para incluir solo las condiciones médicas específicas
filtered_df = df[(df['Ascites'].isin(['N', 'Y'])) &
                 (df['Hepatomegaly'].isin(['N', 'Y'])) &
                 (df['Spiders'].isin(['N', 'Y'])) &
                 (df['Edema'].isin(['N', 'Y']))]

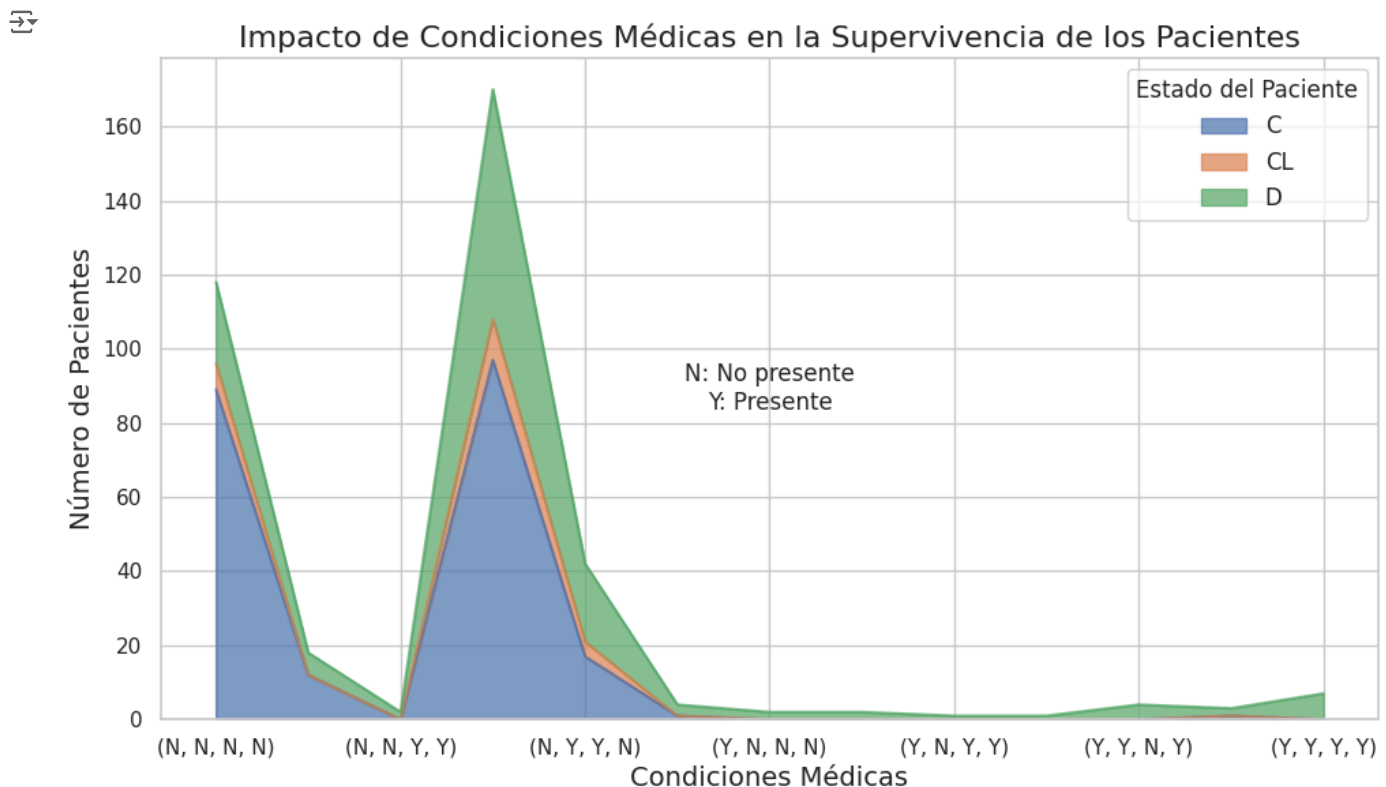
# Crear un DataFrame que contenga la frecuencia de cada combinación de condiciones médicas y estado
medical_conditions = ['Ascites', 'Hepatomegaly', 'Spiders', 'Edema']
condition_counts = filtered_df.groupby(medical_conditions + ['Status']).size().unstack().fillna(0)

# Crear un gráfico de áreas apiladas
condition_counts.plot(kind='area', stacked=True, figsize=(10, 6), alpha=0.7)

# Añadir una nota sobre el significado de 'N' y 'Y' en la figura
plt.text(0.5, 0.5, 'N: No presente\nY: Presente', fontsize=12, ha='center', va='center', transform=plt.gca().transAxes)

# Añadir etiquetas y título
plt.xlabel('Condiciones Médicas', fontsize=14)
plt.ylabel('Número de Pacientes', fontsize=14)
plt.title('Impacto de Condiciones Médicas en la Supervivencia de los Pacientes', fontsize=16)

# Mostrar el gráfico
plt.legend(title='Estado del Paciente', fontsize=12)
plt.tight_layout()
plt.show()
```



- `import matplotlib.pyplot as plt`: Importa la biblioteca Matplotlib, que se utiliza para crear visualizaciones en Python.
- Filtrado del DataFrame: Se filtra el DataFrame original para incluir solo las filas que contienen información sobre las condiciones médicas específicas ('Ascites', 'Hepatomegaly', 'Spiders', 'Edema').
- Creación del DataFrame de frecuencia: Se agrupa el DataFrame filtrado por las condiciones médicas y el estado del paciente y se calcula la frecuencia de cada combinación. Luego, se crea un nuevo DataFrame donde las filas representan las combinaciones de condiciones médicas y las columnas representan los estados del paciente.
- Creación del gráfico de áreas apiladas: Se utiliza el método `plot` con `kind='area'` para crear un gráfico de áreas apiladas. El parámetro `stacked=True` indica que las áreas deben apilarse una encima de la otra. El parámetro `alpha` establece la transparencia de las áreas para mejorar la legibilidad.

- Añadir una nota: Se utiliza `plt.text` para añadir una nota explicativa sobre el significado de 'N' y 'Y' en la figura. La posición de la nota se establece en el centro de la figura con coordenadas relativas.
- Añadir etiquetas y título: Se añaden etiquetas a los ejes x e y y se establece un título para el gráfico.
- Mostrar el gráfico: Se utiliza `plt.legend` para mostrar la leyenda que indica los diferentes estados del paciente y `plt.show()` para mostrar el gráfico completo.

## 6. Relación entre los niveles de biomarcadores y el estado de los pacientes:

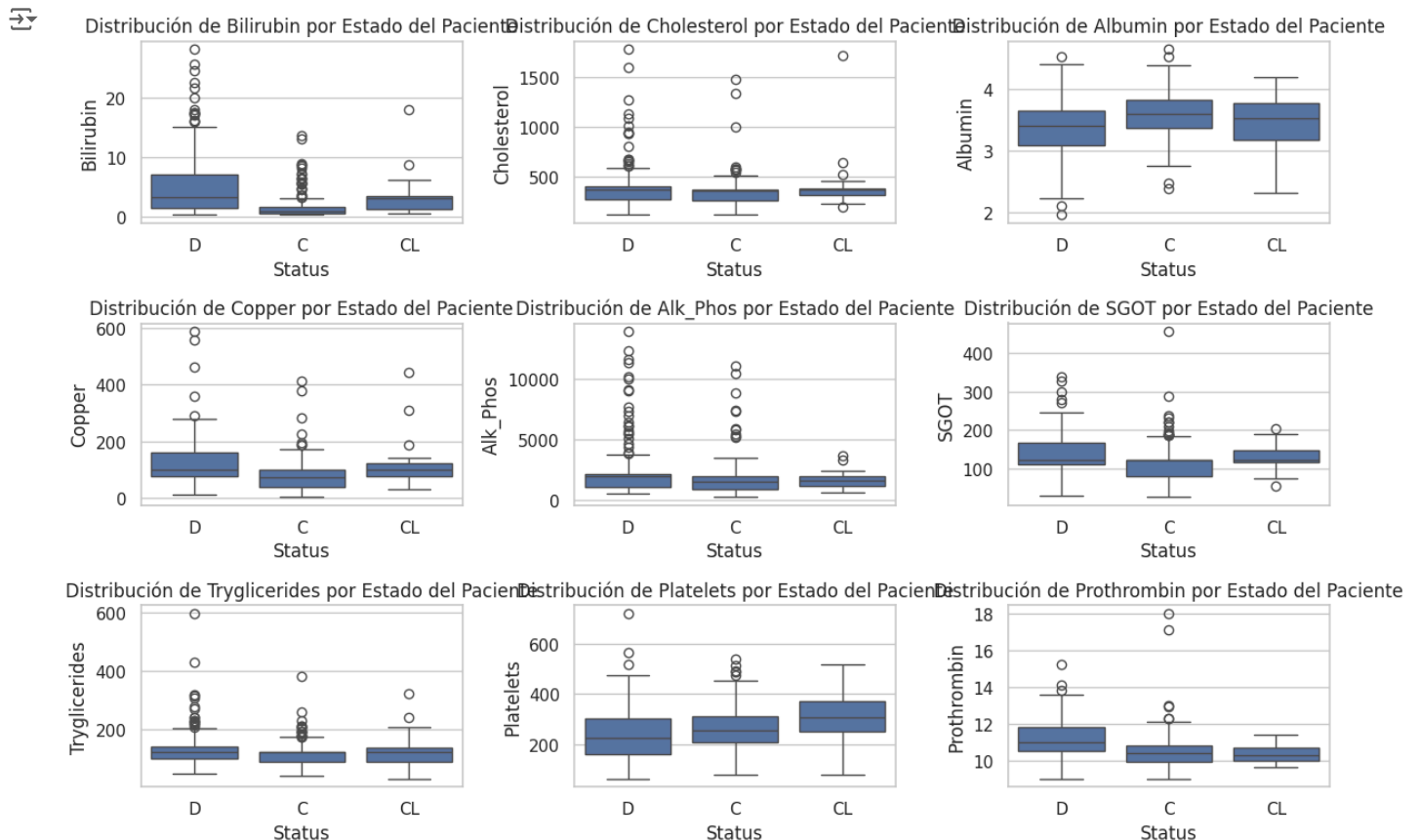
El gráfico de caja y bigotes, también conocido como diagrama de caja, es una herramienta visual que nos permite representar la distribución de un conjunto de datos de una manera descriptiva y concisa. Este tipo de gráfico proporciona información sobre la mediana, los cuartiles, los valores atípicos y la dispersión de los datos. En el contexto de este análisis, se utiliza para visualizar la distribución de los niveles de biomarcadores para diferentes estados de los pacientes.

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

# Lista de biomarcadores a considerar
biomarkers = ['Bilirubin', 'Cholesterol', 'Albumin', 'Copper', 'Alk_Phos', 'SGOT', 'Tryglicerides', 'Platelets', 'Prothrombin']

# Crear un gráfico de caja y bigotes para cada biomarcador
plt.figure(figsize=(12, 8))
for i, biomarker in enumerate(biomarkers, 1):
    plt.subplot(3, 3, i)
    sns.boxplot(x='Status', y=biomarker, data=df)
    plt.title(f'Distribución de {biomarker} por Estado del Paciente')

# Ajustar espacios entre subgráficos
plt.tight_layout()
plt.show()
```



- `import seaborn as sns`: Importa la biblioteca Seaborn, que nos permite crear gráficos estadísticos atractivos y informativos.

- `import matplotlib.pyplot as plt`: Importa la biblioteca Matplotlib, que utilizamos para personalizar y mostrar nuestras visualizaciones.
- `biomarkers = ['Bilirubin', 'Cholesterol', 'Albumin', 'Copper', 'Alk_Phos', 'SGOT', 'Tryglicerides', 'Platelets', 'Prothrombin']`: Crea una lista de los biomarcadores que se analizarán en este gráfico.
- `plt.figure(figsize=(12, 8))`: Crea una nueva figura de Matplotlib con un tamaño de 12x8 pulgadas.
- `for i, biomarker in enumerate(biomarkers, 1):`: Itera sobre cada biomarcador en la lista `biomarkers`.
- `plt.subplot(3, 3, i)`: Crea un subgráfico en la posición `i` dentro de una cuadrícula de 3x3.
- `sns.boxplot(x='Status', y=biomarker, data=df)`: Crea un gráfico de caja y bigotes utilizando Seaborn, donde el eje X representa el estado del paciente y el eje Y representa los niveles del biomarcador actual.
- `plt.title(f'Distribución de {biomarker} por Estado del Paciente')`: Establece el título del subgráfico con el nombre del biomarcador actual.
- `plt.tight_layout()`: Ajusta automáticamente los márgenes del gráfico para que quepan todos los elementos.
- `plt.show()`: Muestra el gráfico generado.

## 7. Impacto del estadio histológico de la enfermedad en la supervivencia de los pacientes:

El tipo de gráfica que utilizamos es un mapa de calor (heatmap), el cual es una representación visual de los datos donde los valores de una matriz se representan como colores. En este contexto, utilizamos el mapa de calor para mostrar la relación entre el estadio histológico de la enfermedad y el estado de los pacientes.

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

# Convertir la columna 'Stage' a tipo entero
df['Stage'] = df['Stage'].astype(int)

# Filtrar el DataFrame para incluir solo las columnas relevantes
relevant_columns = ['Stage', 'Status']
filtered_df = df[relevant_columns]

# Calcular la matriz de contingencia entre el estadio histológico y el estado del paciente
contingency_matrix = pd.crosstab(filtered_df['Stage'], filtered_df['Status'])

# Crear un mapa de calor con Seaborn
plt.figure(figsize=(10, 6)) # Establecer el tamaño de la figura
sns.heatmap(contingency_matrix, annot=True, cmap='coolwarm', fmt='d') # fmt='d' para formato entero

# Agregar etiquetas y título
plt.xlabel('Estado del Paciente', fontsize=14)
plt.ylabel('Estadio Histológico', fontsize=14)
plt.title('Impacto del Estadio Histológico en la Supervivencia de los Pacientes', fontsize=16)

# Mostrar el mapa de calor
plt.tight_layout()
plt.show()
```



## Impacto del Estadio Histológico en la Supervivencia de los Pacientes



- **Importar bibliotecas:** Se importan las bibliotecas necesarias, `seaborn` para crear gráficos estadísticos y `matplotlib.pyplot` para personalizar y mostrar las visualizaciones.
- **Filtrar el DataFrame:** Se seleccionan las columnas relevantes del DataFrame original, que son 'Stage' (estadio histológico) y 'Status' (estado del paciente), y se almacenan en `filtered_df`.
- **Calcular la matriz de contingencia:** Se utiliza la función `pd.crosstab()` de Pandas para calcular la matriz de contingencia entre el estadio histológico y el estado del paciente, donde las filas representan los diferentes estadios histológicos y las columnas representan los diferentes estados de los pacientes. La matriz de contingencia se almacena en `contingency_matrix`.
- **Crear un mapa de calor:** Se crea un mapa de calor utilizando la función `sns.heatmap()` de Seaborn. Se pasa la matriz de contingencia como datos y se establece `annot=True` para mostrar los valores de las celdas en el mapa de calor. El argumento `cmap='coolwarm'` se utiliza para elegir un mapa de colores divergente que va desde azul (valores bajos) a rojo (valores altos). El argumento `fmt='d'` se utiliza para formatear los valores como enteros.
- **Personalizar etiquetas y título:** Se añaden etiquetas a los ejes X e Y y un título al gráfico para hacerlo más comprensible y legible.
- **Mostrar el mapa de calor:** Se llama a `plt.show()` para mostrar el mapa de calor en la salida.

ESTADO DE LOS PACIENTES

### Actuar

La toma de decisiones basadas en datos es fundamental para mejorar la eficacia y la eficiencia en el tratamiento y la gestión de la cirrosis. En esta sección, se proponen acciones específicas que se derivan del análisis de datos realizado previamente. Estas acciones están respaldadas por evidencia estadística y se basan en los resultados obtenidos de la exploración y el análisis de diversos conjuntos de datos relacionados con la cirrosis.

1. Personalizar el tratamiento

#### Datos utilizados:

- Estadísticas descriptivas de `N_Days` agrupadas por el estado del paciente.
- Impacto del estadio histológico de la enfermedad en la supervivencia de los pacientes.

**Propuesta de acción:** Dado que se han observado diferencias significativas en la supervivencia de los pacientes según el estadio histológico de la enfermedad, se recomienda personalizar el tratamiento de la cirrosis de acuerdo con el estadio de la enfermedad. Los pacientes en etapas avanzadas pueden requerir un enfoque más agresivo y una atención médica más intensiva, que puede incluir opciones terapéuticas como trasplante hepático, terapia farmacológica específica o intervenciones quirúrgicas. Por otro lado, los pacientes en etapas iniciales pueden beneficiarse de medidas de manejo conservador y estrategias de estilo de vida, como cambios en la dieta y el ejercicio, junto con el monitoreo regular de la progresión de la enfermedad. La personalización del tratamiento garantiza que cada paciente reciba la atención más adecuada y efectiva de acuerdo con su estado de salud y necesidades individuales.

2. Monitorización continua

#### Datos utilizados:

- Resultados del análisis de los niveles de biomarcadores en relación con el estado de los pacientes.

**Propuesta de acción:** Implementar un sistema de monitorización continua para seguir de cerca la progresión de la enfermedad en los pacientes. Se pueden utilizar pruebas regulares de biomarcadores, como bilirrubina, colesterol, albúmina, entre otros, para detectar cambios en el estado de salud y ajustar el tratamiento en consecuencia. Esto permitirá una intervención temprana y una gestión más efectiva de la cirrosis, optimizando así los resultados clínicos.

3. Promoción de la detección temprana

#### Datos utilizados:

- Impacto de ciertas condiciones médicas (Ascites, Hepatomegaly, Spiders, Edema) en la supervivencia de los pacientes.

**Propuesta de acción:** Promover la detección temprana de la cirrosis y sus complicaciones mediante campañas de concienciación y programas