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

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.

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
```

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 esta 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.

Descripcion 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', 'Pl
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', 'Trygliceride', '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
Trygliceride       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.