Analizar

Autor: German Preciat; Universidad de Guadalajara

Carrera: Ing. Biomédica

Materia: Analisis de datos clínicos

Contacto: german.preciat@academicos.udg.mx

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_stat
```

- 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        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:
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
  _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 * (precision * recall) / (precision + 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 utlizaremos un codigo similar al anterior con una pequeña modificacion, añadir: `X = pd.get_dummies(X, drop_first=True)` despues 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_stat

# 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:
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
  _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_stat

# 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:
  _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:
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
  _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_stat

# 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:
```

```
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
  _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 adm
status_by_drug = df.groupby('Drug')['Status'].value_counts(normalize=True)

# Imprimir la distribución de estados del paciente para cada tipo de medicamento adm
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
else:
    print("No hay una diferencia significativa en la edad promedio entre los diferen
```

```
Estadística F: 20.134266051199326
Valor p: 4.512801037966663e-09
Hay una diferencia significativa en la edad promedio entre los diferentes estado
```

El valor de la estadística F obtenido es aproximadamente 20.13 y el valor p es extremadamente pequeño, alrededor de 4.51e-09 (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 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 p
contingency_table = pd.crosstab(index=df['Status'], columns=[df['Ascites'], df['Hepa

# 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 estad
else:
    print("No hay una asociación significativa entre las condiciones médicas y el es
```

```
Valor de chi-cuadrado: 81.59296565949666
Valor p: 2.160496432248983e-05
Hay una asociación significativa entre las condiciones médicas y el estado de lo
```

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', '

# Realizar ANOVA para cada biomarcador y comparar las distribuciones entre los difer
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
Resultado del ANOVA para Cholesterol : F_onewayResult(statistic=5.88644897560268
Resultado del ANOVA para Albumin : F_onewayResult(statistic=15.260927571507652,
Resultado del ANOVA para Copper : F_onewayResult(statistic=27.145803312454966, p
Resultado del ANOVA para Alk_Phos : F_onewayResult(statistic=8.981623453096693,
Resultado del ANOVA para SGOT : F_onewayResult(statistic=14.575889622594048, pva
Resultado del ANOVA para Tryglicerides : F_onewayResult(statistic=6.691201978285
Resultado del ANOVA para Platelets : F_onewayResult(statistic=5.671067970019371,
Resultado del ANOVA para Prothrombin : F_onewayResult(statistic=29.9452137304067
```

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
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 enfer
else:
    print("No hay una asociación significativa entre el estadio histológico de la en
```

```
Valor de chi-cuadrado: 50.13785630363444
Valor p: 3.8448806725631566e-08
Hay una asociación significativa entre el estadio histológico de la enfermedad y
```

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.