Actividad práctica 5.1: Análisis de Datos Tabulares en el Sector Biosanitario

Análisis de Datos Biosanitarios: Explicación Detallada

1. Importación de Bibliotecas y Carga de Datos

Primero, necesitamos importar las bibliotecas necesarias para nuestro análisis:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from typing import Dict
import numpy as np
```

Para cargar los datos desde el CSV, creamos una función específica que incluye manejo de errores:

```
def cargar_datos(ruta_archivo: str) -> pd.DataFrame:
    """
    Carga los datos desde un archivo CSV y maneja posibles errores
    """
    try:
        df = pd.read_csv(ruta_archivo)
        print("Datos cargados exitosamente")
        return df
    except Exception as e:
        print(f"Error al cargar el archivo: {e}")
        return None
```

2. Análisis Exploratorio Inicial

Esta función realiza un primer vistazo a nuestros datos para entender su estructura y características básicas:

```
def realizar_analisis_exploratorio(df: pd.DataFrame) -> None:
    """
    Realiza y muestra un análisis exploratorio básico del dataset
    """
    print("\n=== ANÁLISIS EXPLORATORIO INICIAL ===")
    print("\nInformación del Dataset:")
    print(df.info())   # Muestra tipos de datos y valores no nulos

    print("\nEstadísticas Descriptivas:")
    print(df.describe())  # Estadísticas numéricas básicas

    print("\nValores Nulos por Columna:")
    print(df.isnull().sum())  # Identifica datos faltantes
```

Este análisis inicial nos permite:
- Verificar los tipos de datos de cada columna
- Identificar valores faltantes
- Obtener estadísticas básicas de las variables numéricas

3. Análisis Demográfico

Para entender mejor la población de estudio:

```
def analizar_demograficos(df: pd.DataFrame) -> None:
    """
    Analiza y visualiza la distribución demográfica de los pacientes
    """
    print("\n=== ANÁLISIS DEMOGRÁFICO ===")

    # Distribución por sexo
    print("\nDistribución por Sexo:")
    print(df['Sex'].value_counts())
```

```
    # Distribución por edad
    print("\nEstadísticas de Edad:")
    print(df['Age'].describe())

    # Visualización de la distribución de edad
    plt.figure(figsize=(10, 6))
    sns.histplot(data=df, x='Age', bins=30)
    plt.title('Distribución de Edad de los Pacientes')
    plt.xlabel('Edad')
    plt.ylabel('Frecuencia')
    plt.show()
```

Este análisis nos permite:
- Visualizar la distribución por género
- Entender la estructura de edad de los pacientes
- Identificar posibles sesgos demográficos

4. Análisis de Condiciones Clínicas

Para examinar los aspectos médicos:

```
def analizar_condiciones_clinicas(df: pd.DataFrame) -> None:
    """
    Analiza la prevalencia de síntomas y marcadores bioquímicos
    """
    print("\n=== ANÁLISIS DE CONDICIONES CLÍNICAS ===")

    # Análisis de síntomas
    sintomas = ['Ascites', 'Hepatomegaly', 'Spiders', 'Edema']
    for sintoma in sintomas:
        print(f"\nDistribución de {sintoma}:")
        print(df[sintoma].value_counts())

    # Análisis de marcadores bioquímicos
    marcadores = ['Bilirubin', 'Cholesterol', 'Albumin', 'Copper',
                  'Alk_Phos', 'SGOT']
    print("\nEstadísticas de Marcadores Bioquímicos:")
    print(df[marcadores].describe())
```

Este análisis nos permite:
- Identificar la prevalencia de cada síntoma

- Analizar la distribución de marcadores bioquímicos
- Detectar valores anormales en los parámetros clínicos

5. Análisis de Tratamientos y Resultados

Para evaluar la efectividad de los tratamientos:

```
def analizar_tratamientos(df: pd.DataFrame) -> None:
    """
    Analiza los tratamientos aplicados y sus resultados
    """
    print("\n=== ANÁLISIS DE TRATAMIENTOS Y RESULTADOS ===")

    # Distribución de medicamentos
    print("\nDistribución de Medicamentos:")
    print(df['Drug'].value_counts())

    # Análisis de estado (Status)
    print("\nDistribución de Estado:")
    print(df['Status'].value_counts())

    # Análisis por etapa
    print("\nDistribución por Etapa:")
    print(df['Stage'].value_counts())
```

Este análisis nos permite:
- Identificar los tratamientos más comunes
- Evaluar los resultados de los tratamientos
- Analizar la distribución de etapas de la enfermedad

6. Visualizaciones Avanzadas

Para una comprensión más profunda de las relaciones entre variables:

```
def crear_visualizaciones(df: pd.DataFrame) -> None:
    """
    Crea visualizaciones avanzadas para análisis detallado
    """
    # Matriz de correlación
```

```
    variables_numericas = ['Age', 'Bilirubin', 'Cholesterol', 'Albumin',
                           'Copper', 'Alk_Phos', 'SGOT', 'Trygliceride',
                           'Platelets', 'Prothrombin']

    plt.figure(figsize=(12, 8))
    sns.heatmap(df[variables_numericas].corr(), annot=True,
                cmap='coolwarm', center=0)
    plt.title('Matriz de Correlación de Variables Numéricas')
    plt.xticks(rotation=45)
    plt.yticks(rotation=45)
    plt.tight_layout()
    plt.show()

    # Distribución de etapas por medicamento
    plt.figure(figsize=(10, 6))
    sns.boxplot(data=df, x='Drug', y='Stage')
    plt.title('Distribución de Etapas por Medicamento')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

7. Función Principal

Para ejecutar todo el análisis de manera organizada:

```
def main():
    Cargar los datos
    df = cargar_datos('ruta_del_archivo.csv')

    if df is not None:
        realizar_analisis_exploratorio(df)
        analizar_demograficos(df)
        analizar_condiciones_clinicas(df)
        analizar_tratamientos(df)
        crear_visualizaciones(df)

if __name__ == "__main__":
    main()
```

Este código proporciona un análisis completo y detallado de los datos biosanitarios, incluyendo:

- Análisis exploratorio inicial
- Estudio demográfico
- Análisis de condiciones clínicas
- Evaluación de tratamientos
- Visualizaciones avanzadas

Para usar este código:
1. Asegúrate de tener todas las bibliotecas instaladas
2. Reemplaza '**ruta_del_archivo.csv**' con la ruta real de tu archivo
3. Ejecuta el script

Las visualizaciones te ayudarán a:
- Identificar patrones en los datos
- Descubrir correlaciones entre variables
- Evaluar la efectividad de los tratamientos
- Entender la distribución de las etapas de la enfermedad

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from typing import Dict
import numpy as np

def cargar_datos(ruta_archivo: str) -> pd.DataFrame:
    """
    Carga los datos desde un archivo CSV
    """
    try:
        df = pd.read_csv(ruta_archivo)
        print("Datos cargados exitosamente")
        return df
    except Exception as e:
        print(f"Error al cargar el archivo: {e}")
        return None

def realizar_analisis_exploratorio(df: pd.DataFrame) -> None:
    """
    Realiza y muestra un análisis exploratorio básico del dataset
    """
    print("\n=== ANÁLISIS EXPLORATORIO INICIAL ===")
    print("\nInformación del Dataset:")
    print(df.info())
```

```
    print("\nEstadísticas Descriptivas:")
    print(df.describe())

    print("\nValores Nulos por Columna:")
    print(df.isnull().sum())

def analizar_demograficos(df: pd.DataFrame) -> None:
    """
    Analiza y muestra la distribución demográfica
    """
    print("\n=== ANÁLISIS DEMOGRÁFICO ===")

    # Distribución por sexo
    print("\nDistribución por Sexo:")
    print(df['Sex'].value_counts())

    # Distribución por edad
    print("\nEstadísticas de Edad:")
    print(df['Age'].describe())

    # Crear gráfico de distribución de edad
    plt.figure(figsize=(10, 6))
    sns.histplot(data=df, x='Age', bins=30)
    plt.title('Distribución de Edad de los Pacientes')
    plt.xlabel('Edad')
    plt.ylabel('Frecuencia')
    plt.show()

def analizar_condiciones_clinicas(df: pd.DataFrame) -> None:
    """
    Analiza y muestra la distribución de condiciones clínicas
    """
    print("\n=== ANÁLISIS DE CONDICIONES CLÍNICAS ===")

    # Análisis de síntomas
    sintomas = ['Ascites', 'Hepatomegaly', 'Spiders', 'Edema']
    for sintoma in sintomas:
        print(f"\nDistribución de {sintoma}:")
        print(df[sintoma].value_counts())

    # Análisis de marcadores bioquímicos
    marcadores = ['Bilirubin', 'Cholesterol', 'Albumin', 'Copper', 'Alk_Phos',
'SGOT']
    print("\nEstadísticas de Marcadores Bioquímicos:")
    print(df[marcadores].describe())
```

```
def analizar_tratamientos(df: pd.DataFrame) -> None:
    """
    Analiza y muestra información sobre tratamientos y resultados
    """
    print("\n=== ANÁLISIS DE TRATAMIENTOS Y RESULTADOS ===")

    # Distribución de medicamentos
    print("\nDistribución de Medicamentos:")
    print(df['Drug'].value_counts())

    # Análisis de estado (Status)
    print("\nDistribución de Estado:")
    print(df['Status'].value_counts())

    # Análisis por etapa
    print("\nDistribución por Etapa:")
    print(df['Stage'].value_counts())

def crear_visualizaciones(df: pd.DataFrame) -> None:
    """
    Crea visualizaciones importantes del dataset
    """
    # Correlación entre variables numéricas
    variables_numericas = ['Age', 'Bilirubin', 'Cholesterol', 'Albumin', 'Copper',
                           'Alk_Phos', 'SGOT', 'Tryglicerides', 'Platelets',
'Prothrombin']

    plt.figure(figsize=(12, 8))
    sns.heatmap(df[variables_numericas].corr(), annot=True, cmap='coolwarm',
center=0)
    plt.title('Matriz de Correlación de Variables Numéricas')
    plt.xticks(rotation=45)
    plt.yticks(rotation=45)
    plt.tight_layout()
    plt.show()

    # Distribución de etapas por medicamento
    plt.figure(figsize=(10, 6))
    sns.boxplot(data=df, x='Drug', y='Stage')
    plt.title('Distribución de Etapas por Medicamento')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()

def main():
    # Reemplazar 'ruta_del_archivo.csv' con la ruta real del archivo
    df = cargar_datos('ruta_del_archivo.csv')
```

```
    if df is not None:
        realizar_analisis_exploratorio(df)
        analizar_demograficos(df)
        analizar_condiciones_clinicas(df)
        analizar_tratamientos(df)
        crear_visualizaciones(df)

if __name__ == "__main__":
    main()
```

Análisis de Datos Biosanitarios en Jupyter Notebook

1. Importación de Bibliotecas

Primero, importamos todas las bibliotecas necesarias:

```
# Importación de bibliotecas necesarias
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Configuración para mostrar las gráficas en el notebook
%matplotlib inline
# Configuración para mejorar la visualización
plt.style.use('seaborn')
# Configuración para mostrar todas las columnas
pd.set_option('display.max_columns', None)
```

2. Carga de Datos

Cargamos el dataset y realizamos una primera inspección:

```
# Carga del dataset
# Reemplaza 'ruta_del_archivo.csv' con la ruta de tu archivo
df = pd.read_csv('ruta_del_archivo.csv')

# Mostrar las primeras filas del dataset
print("Primeras filas del dataset:")
display(df.head())

# Información general del dataset
print("\nInformación del dataset:")
display(df.info())
```

3. Análisis Exploratorio Inicial

```
# Estadísticas descriptivas básicas
print("Estadísticas descriptivas:")
display(df.describe())

# Verificar valores nulos
print("\nValores nulos por columna:")
display(df.isnull().sum())
```

4. Análisis Demográfico

```
# Análisis de distribución por sexo
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='Sex')
plt.title('Distribución por Sexo')
plt.show()

# Distribución de edad
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='Age', bins=30)
plt.title('Distribución de Edad de los Pacientes')
```

```
plt.xlabel('Edad')
plt.ylabel('Frecuencia')
plt.show()

# Estadísticas de edad por sexo
print("\nEstadísticas de edad por sexo:")
display(df.groupby('Sex')['Age'].describe())
```

5. Análisis de Condiciones Clínicas

```
# Análisis de síntomas
sintomas = ['Ascites', 'Hepatomegaly', 'Spiders', 'Edema']

# Crear subplots para cada síntoma
fig, axes = plt.subplots(2, 2, figsize=(12, 10))
axes = axes.ravel()

for idx, sintoma in enumerate(sintomas):
    sns.countplot(data=df, x=sintoma, ax=axes[idx])
    axes[idx].set_title(f'Distribución de {sintoma}')

plt.tight_layout()
plt.show()

# Análisis de marcadores bioquímicos
marcadores = ['Bilirubin', 'Cholesterol', 'Albumin', 'Copper', 'Alk_Phos', 'SGOT']

# Crear boxplots para marcadores bioquímicos
plt.figure(figsize=(15, 6))
df[marcadores].boxplot()
plt.xticks(rotation=45)
plt.title('Distribución de Marcadores Bioquímicos')
plt.show()

# Estadísticas descriptivas de marcadores
print("\nEstadísticas de marcadores bioquímicos:")
display(df[marcadores].describe())
```

6. Análisis de Tratamientos y Resultados

```
# Distribución de medicamentos
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='Drug')
plt.title('Distribución de Medicamentos')
plt.xticks(rotation=45)
plt.show()

# Análisis de estado (Status)
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='Status')
plt.title('Distribución de Estado')
plt.show()

# Análisis por etapa
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='Stage')
plt.title('Distribución por Etapa')
plt.show()

# Relación entre medicamento y etapa
plt.figure(figsize=(12, 6))
sns.boxplot(data=df, x='Drug', y='Stage')
plt.title('Distribución de Etapas por Medicamento')
plt.xticks(rotation=45)
plt.show()
```

7. Análisis de Correlaciones

```
# Seleccionar variables numéricas para la correlación
variables_numericas = ['Age', 'Bilirubin', 'Cholesterol', 'Albumin',
                       'Copper', 'Alk_Phos', 'SGOT', 'Trygliceride',
                       'Platelets', 'Prothrombin']

# Crear matriz de correlación
plt.figure(figsize=(12, 10))
sns.heatmap(df[variables_numericas].corr(),
            annot=True,
            cmap='coolwarm',
            center=0,
            fmt='.2f')
```

```
plt.title('Matriz de Correlación de Variables Numéricas')
plt.xticks(rotation=45)
plt.yticks(rotation=45)
plt.tight_layout()
plt.show()
```

8. Análisis Estadístico Avanzado

```
# Comparación de marcadores bioquímicos por etapa
for marcador in marcadores:
    plt.figure(figsize=(10, 6))
    sns.boxplot(data=df, x='Stage', y=marcador)
    plt.title(f'{marcador} por Etapa')
    plt.show()

# Análisis de supervivencia básico
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='Status', y='N_Days')
plt.title('Días de Supervivencia por Estado')
plt.show()
```

Este código está adaptado específicamente para Jupyter Notebook y ofrece varias ventajas:

1. Visualización inmediata de resultados
2. Capacidad de modificar y re-ejecutar celdas individuales
3. Mejor documentación con texto explicativo entre celdas
4. Facilidad para exportar resultados

Para usar este código:
1. Copia cada sección en una celda separada de tu Jupyter Notebook
2. Modifica la ruta del archivo en la sección de carga de datos
3. Ejecuta las celdas en orden

Recomendaciones adicionales:
- Puedes agregar celdas de markdown entre el código para documentar tus hallazgos
- Ajusta los tamaños de las figuras según tus necesidades
- Modifica los colores y estilos de las visualizaciones según prefieras
- Agrega análisis estadísticos adicionales según tus objetivos específicos