Clase15: Práctica Guiada para el Desarrollo de una Aplicación Web con Streamlit, Pandas y Requests con API REST

Sección 1: Repaso sobre el Desarrollo de una Aplicación Web con Streamlit, Pandas

Objetivos

1. Utilizar **Streamlit** para crear aplicaciones web interactivas.
2. Cargar y procesar datos desde un archivo **CSV** utilizando **Pandas**.
3. Visualizar los datos y generar gráficos interactivos para su análisis.

Requisitos Previos

- Conocimientos básicos de **Python**.
- Familiaridad con **Pandas** para la manipulación de datos.
- Tener instaladas las bibliotecas **Streamlit** y **Pandas**.

Instalar las bibliotecas necesarias si no las tienes:

```
pip install streamlit pandas
```

Parte 1: Introducción a Streamlit

Streamlit es una biblioteca que permite crear aplicaciones web con pocas líneas de código. Es ideal para visualizar datos de forma rápida e interactiva.

1. Estructura básica de una aplicación en Streamlit:

```
import streamlit as st

# Título de la aplicación
st.title('Mi Primera Aplicación Web con Streamlit')

# Texto simple
st.write('Esta es una aplicación web de ejemplo utilizando Streamlit.')
```

2. Ejecutar la aplicación forma local o remota

Para ejecutar la aplicación de forma local o remota primero guarda el archivo en formato “nombre_archivo.py” (notar que debe estar en formato .py)

- Para desplegar la aplicación web de forma local utilice el siguiente comando desde un terminal de Python :

  ```
  streamlit run nombre_del_archivo.py
  ```

- Para desplegar la aplicación web de forma remota con Streamlit Cloud de manera gratuita. Debes seguir los siguientes dos pasos para ello puede apoyarse del video tutorial del disponible en el blackboard.

  1. Sube tu proyecto a GitHub.
  2. Conecta tu repositorio a Streamlit Cloud para hacer el despliegue automático.

Parte 2: Cargar un Archivo CSV usando Pandas

La mayoría de las aplicaciones de análisis de datos comienzan cargando un archivo de datos. Usaremos **Pandas** para cargar un archivo CSV. Utilice para crear su aplicación el archivo “database_titanic.csv” disponible en blackboard. Copie el siguiente código base y ejecute la aplicación.

```
import streamlit as st
import pandas as pd

# Título de la aplicación
st.title('Aplicación de Análisis de Datos')

# Cargar el archivo CSV
uploaded_file = st.file_uploader('Sube tu archivo CSV', type=['csv'])

if uploaded_file is not None:
    # Leer el archivo CSV usando Pandas
    df = pd.read_csv(uploaded_file)

    # Mostrar las primeras filas del archivo
    st.write('Primeras 5 filas del archivo:')
    st.write(df.head())
```

Parte 3: Filtrar y Procesar Datos con Pandas

Podemos aplicar filtros interactivos sobre los datos y visualizarlos de manera dinámica. Para ello actualice el código base con las siguientes líneas de código, tenga en cuenta que esto se debe ejecutar dentro de la sentencia condicional.

```
if uploaded_file is not None:
    # Leer el archivo CSV
    df = pd.read_csv(uploaded_file)

    # Mostrar la estructura del DataFrame
    st.write('Estructura del DataFrame:')
    st.write(df.describe())

    # Seleccionar una columna numérica para analizar
    col = st.selectbox('Selecciona una columna para filtrar', df.columns)

    # Filtrar datos según el valor de la columna
    valor_min = st.slider('Selecciona un valor mínimo',
float(df[col].min()), float(df[col].max()))
    df_filtrado = df[df[col] >= valor_min]

    # Mostrar los datos filtrados
    st.write(f'Datos filtrados donde {col} >= {valor_min}:')
    st.write(df_filtrado)
```

Parte 4: Visualización de Datos con Streamlit

Streamlit es compatible con varias bibliotecas de visualización como **Matplotlib**, **Plotly**, y **Altair**. A continuación, te muestro cómo integrar visualizaciones directamente en la aplicación. Añade al código de la parte 3 las siguientes líneas de código y asegúrate de haber importado matplotlib

Ejemplo con Matplotlib:

```
    # Crear gráfico de histograma
    fig, ax = plt.subplots()
    ax.hist(df[col], bins=20)
    ax.set_title(f'Histograma de {col}')
    ax.set_xlabel(col)
    ax.set_ylabel('Frecuencia')

    # Mostrar el gráfico en Streamlit
    st.pyplot(fig)
```

Parte 5: Generación de Gráficos Interactivos

También podemos crear gráficos interactivos con **Plotly** o **Altair**. st.bar_chart, st.line_chart, st.scatter_chart

Ejemplo con Altair:

```
import altair as alt

if uploaded_file is not None:
    df = pd.read_csv(uploaded_file)

    # Seleccionar columnas numéricas para el gráfico
    x_axis = st.selectbox('Selecciona el eje X',
df.select_dtypes(include='number').columns)
    y_axis = st.selectbox('Selecciona el eje Y',
df.select_dtypes(include='number').columns)

    # Crear gráfico de dispersión con Altair
    chart = alt.Chart(df).mark_circle(size=60).encode(
        x=x_axis,
        y=y_axis,
        tooltip=[x_axis, y_axis]).interactive()

    # Mostrar el gráfico en Streamlit
    st.altair_chart(chart, use_container_width=True)
```

Resumen

Con esta clase guiada, hemos aprendido cómo usar **Streamlit** para crear aplicaciones web interactivas que cargan, procesan, y visualizan datos desde un archivo CSV utilizando **Pandas**. Este enfoque permite crear prototipos rápidos y fáciles de aplicaciones de análisis de datos que pueden ser compartidas fácilmente.

Esta guía puede servir de base para que los estudiantes desarrollen aplicaciones más complejas a medida que dominen las funcionalidades de **Pandas** y las opciones de visualización de **Streamlit**.

Sección 2: Integración de Streamlit, Pandas y Requests con API REST

Objetivos

1. Utilizar **Requests** para hacer una petición a una **API REST** que devuelve datos en formato **JSON**.
2. Procesar esos datos con **Pandas** para manipularlos y transformarlos.
3. Visualizar los datos de manera interactiva utilizando **Streamlit**.

Instalar las bibliotecas necesarias:

```
pip install streamlit pandas requests
```

Parte 1: Estructura Básica con Streamlit y Requests

En esta primera parte, haremos una petición a una API REST para obtener datos en formato JSON. Copie el siguiente código y genere una aplicación web.

```
import streamlit as st
import pandas as pd
import requests

# Título de la aplicación
st.title('Aplicación Web: Datos desde una API REST')

# URL de la API REST (puedes cambiarla por cualquier API pública que
devuelva JSON)
api_url = 'https://jsonplaceholder.typicode.com/posts'

# Realizar la petición a la API
response = requests.get(api_url)

# Verificar que la respuesta sea exitosa (código 200)
if response.status_code == 200:
    # Convertir los datos JSON en un DataFrame de Pandas
    data = response.json()
    df = pd.DataFrame(data)

    # Mostrar los primeros registros
    st.write('Datos obtenidos de la API:')
    st.write(df.head())
else:
    st.error('Error al obtener los datos de la API')
```

Parte 2: Manipulación de Datos con Pandas

Ahora que hemos obtenido los datos desde la API, podemos manipularlos y transformarlos con **Pandas**.

1. **Filtrar Datos**: Podemos seleccionar columnas específicas o aplicar filtros.
2. **Visualización**: Usamos Streamlit para mostrar los datos y realizar análisis simples.

Dentro del primer ciclo condicional añade el siguiente código y luego genere su aplicación web. Esto permite realizar el filtrado si los datos son cargados correctamente

```
if response.status_code == 200:
    # Convertir los datos JSON en un DataFrame de Pandas
    data = response.json()
    df = pd.DataFrame(data)

    # Mostrar los primeros registros
    st.write('Datos obtenidos de la API:')
    st.write(df.head())

    # Seleccionar una columna para mostrar en Streamlit
    columnas = st.multiselect('Selecciona las columnas a visualizar',
df.columns.tolist(), default=df.columns.tolist())
    df_seleccionado = df[columnas]

    # Mostrar el DataFrame con las columnas seleccionadas
    st.write('Datos seleccionados:')
    st.write(df_seleccionado)

    # Filtro por ID
    id_filtro = st.slider('Filtrar por ID (entre 1 y 100)', 1, 100, 50)
    df_filtrado = df[df['id'] <= id_filtro]

    st.write(f'Mostrando datos donde ID <= {id_filtro}:')
    st.write(df_filtrado)
```

Parte 3: Visualización Gráfica con Streamlit

Podemos integrar gráficos con **Matplotlib**, **Altair o el propio Streamlit** para visualizar los datos. Actualice el código anterior con las siguientes líneas, tenga en cuenta que esto debe ir dentro de la primera condición para generar el gráfico solo si se obtienen los resultados correctamente. Asegúrate de importar la librería matplotlib.

- Ejemplo de Gráfico de Barras con Matplotlib:

```
import matplotlib.pyplot as plt

    # Solo graficar si se obtienen los datos correctamente
    # Gráfico de barras basado en el conteo de IDs
    fig, ax = plt.subplots()
    df['userId'].value_counts().sort_index().plot(kind='bar', ax=ax)
    ax.set_title('Conteo de User ID')
    ax.set_xlabel('User ID')
    ax.set_ylabel('Frecuencia')

    # Mostrar el gráfico en Streamlit
    st.pyplot(fig)
```

Parte 4: Creación de Funciones para una Mejor Estructura

Es recomendable estructurar el código en funciones para mejorar la legibilidad y la organización.

```
def obtener_datos_api(api_url):
    """Función que realiza la petición a la API y devuelve un
DataFrame."""
    response = requests.get(api_url)
    if response.status_code == 200:
        data = response.json()
        return pd.DataFrame(data)
    else:
        st.error('Error al obtener los datos de la API')
        return None

# Llamar la función para obtener los datos
df = obtener_datos_api(api_url)

# Si hay datos, mostrar el DataFrame, mostrar dataframe con las columnas
seleccionadas, permitir filtrado y mostrar gráficos.

if df is not None:
    # Mostrar las primeras 5 filas del dataframe
    st.write(df.head())
```

Resumen

Esta práctica guiada te permite integrar **Streamlit**, **Pandas**, y **Requests** para crear una aplicación web que obtenga datos en formato JSON desde una **API REST**, los procese con Pandas, y los visualice de forma interactiva. Este enfoque es ideal para construir aplicaciones que permiten analizar datos externos en tiempo real y hacer un prototipo rápidamente.