Muhammad Nadeem

Documentation:

Overview

The st1.py file is a Streamlit application designed for data visualization. It provides users with the ability to upload data files in various formats (CSV, TXT, XLSX) and perform different types of visualizations. The application includes functionalities for data processing, statistical analysis, and generating plots, with options for customizing and downloading these plots.

Libraries Used

- **Streamlit (streamlit)**: The main framework used for building the web app.
- **Pandas (pandas)**: For handling and processing data files.
- **Matplotlib (matplotlib.pyplot)**: For creating visualizations.
- **Seaborn (seaborn)**: For advanced statistical data visualization.
- **BytesIO (io.BytesIO)**: For handling in-memory binary streams, used for downloading plots.

Key Functions

1. **load_file(file)**:
    - **Purpose**: Loads and processes the uploaded file based on its type.
    - **Parameters**:
        - file: The uploaded file object.
    - **Returns**: A Pandas DataFrame for CSV and TXT files, or a dictionary of DataFrames for XLSX files.
    - **Details**:
        - For CSV files: Uses pd.read_csv.
        - For TXT files: Uses pd.read_csv with a user-specified delimiter.
        - For XLSX files: Uses pd.read_excel to load all sheets.
2. **download_plot(fig, format='png')**:
    - **Purpose**: Saves the generated plot to a specified format and returns it as a downloadable object.
    - **Parameters**:
        - fig: The Matplotlib figure to be downloaded.

        - format: The format of the output file (default is PNG).
    - **Returns**: A BytesIO object containing the plot.
3. **chart_description (chart_type, analysis_type)**:
    - **Purpose**: Provides a detailed description of the selected chart type based on the user's choice.
    - **Parameters**:
        - chart_type: The type of chart selected by the user.
        - analysis_type: The type of analysis selected (Univariate, Bivariate, Multivariate).
    - **Returns**: A string containing a detailed description of the chart type.
4. **toggle_orientation (ax, axis, orientation)**:
    - **Purpose**: Toggles the orientation of the axis labels on the plot.
    - **Parameters**:
        - ax: The Matplotlib axes object.
        - axis: Specifies which axis labels to toggle ('x' or 'y').
        - orientation: The desired orientation angle (0 or 90 degrees).

User Interface Components

1. **Sidebar Navigation**:
    - **Buttons**: Home, About Us, Visualization.
    - **Features**:
        - Provides navigation through different sections of the app.
        - File uploader for data input.
        - Options for plot type selection, statistical analysis, and column selection.
2. **Home Page**:
    - **Content**: Introduction to the application, instructions on how to use it, and a brief description of its features.
    - **Visuals**: A welcoming image displayed using st.image.
3. **About Us Page**:
    - **Content**: Information about the project, team members, and contact details.

   - **Visuals**: Team image displayed using st.image.

4. **Visualization Page**:
   - **Content**:
     - Data file upload and preview.
     - Statistical analysis options (Describe, Correlation, Value Counts).
     - Plot type selection based on the chosen analysis type.
     - Download functionality for the generated plots.
   - **Interactivity**: Users can toggle the orientation of axis labels and select the format for downloading plots.
5. **Footer**:
   - **Content**: Copyright information, social media links, and a prompt to connect with the team.
   - **Styling**: Fixed at the bottom, with custom colors and icons.

Custom CSS

- Custom styles are applied to the sidebar, navigation buttons, footer, and other UI elements to enhance the visual appeal and user experience.
- FontAwesome icons are used for social media links in the footer.