**Title:** Time Series Analysis with Pandas

**Description:** Perform time series analysis with pandas, including resampling, rolling statistics, and lagging using dataset(<https://www.kaggle.com/datasets/brendan45774/test-file>).

**Task 4 Explanation**

**Introduction**

Time series analysis is a method of analyzing data points collected or indexed over time. While the Titanic dataset is not inherently a time series dataset, this report demonstrates how time-based analysis can be performed by simulating a time column. This report explores:

* Simulating a time-based column.
* Applying resampling, rolling statistics, and lagging techniques.
* Deriving insights using Pandas and visualizations.

**Dataset Overview**

The Titanic dataset contains information about passengers, including:

* **Demographic data:** Age, sex, number of family members on board.
* **Travel details:** Ticket class (Pclass), ticket number, and fare paid.
* **Survival information:** Whether the passenger survived (Survived column).

Columns used in this analysis include:

* **Fare:** Analyzed to identify trends over time.
* **Survived:** Used to calculate survival rates.
* **Date (Simulated):** Introduced to enable time series analysis.

**Steps and Methodology**

1. **Data Loading and Preview** The dataset is loaded into a Pandas DataFrame, and the first few rows are displayed for initial inspection.
2. **Simulating a Time Column** A Date column is added by generating sequential dates starting from January 1, 2023. The dataset is indexed by this column to simulate time series data.
3. **Resampling** Resampling aggregates data at specified intervals. In this analysis:
   * Monthly average fares were calculated to observe fare trends over time.
   * Weekly survival rates were computed to analyze survival trends.
4. **Rolling Statistics** Rolling statistics are used to smooth data and observe trends:
   * A 7-day rolling average of fares was calculated to reduce short-term fluctuations and highlight longer-term trends.
5. **Lagging** Lagging shifts data by a specific number of time steps:
   * The fare data was lagged by one day to explore temporal relationships.
6. **Visualization** Visualizations were created using Matplotlib to represent trends and patterns:
   * Monthly average fare trends.
   * Rolling average fare vs. original fare.
   * Weekly survival rate trends.

**Results and Observations**

1. **Monthly Average Fare** The monthly resampling of fares showed fluctuations in the average fare paid. This could indicate variability in passenger demographics or travel patterns.
2. **Weekly Survival Rate** Weekly survival rates highlighted periodic variations, possibly reflecting passenger groups with different survival probabilities.
3. **Rolling Average** The 7-day rolling average of fares smoothed out daily fluctuations, making it easier to identify general trends over time.
4. **Lagging** The lagged fare data provided a reference for comparing daily fares with those of the previous day. This could help identify sudden spikes or drops.

**Visual Analysis**

1. **Monthly Average Fare** The line plot of monthly average fares demonstrated clear trends over time. Peaks and troughs may align with passenger demographics.
2. **Rolling Average of Fare** Comparing the original fare data with its 7-day rolling average highlighted how rolling statistics reduce noise and reveal underlying patterns.
3. **Weekly Survival Rate** The weekly survival rate plot showed distinct patterns, potentially reflecting survival probabilities across different passenger groups.

**Code Summary**

The code implements the above steps as follows:

* **Data Preprocessing:** Adds a Date column and sets it as the index.
* **Resampling:** Aggregates data using .resample() for monthly and weekly analysis.
* **Rolling Statistics:** Applies .rolling() to compute 7-day averages.
* **Lagging:** Uses .shift() to create a lagged version of the Fare column.
* **Visualization:** Uses Matplotlib for clear, interpretable plots.

**Conclusion**

This analysis demonstrates how a dataset lacking explicit time information can be adapted for time series analysis. By introducing a simulated time column, it was possible to resample, calculate rolling statistics, and analyze temporal relationships. The visualizations provided insights into passenger fares and survival trends, showcasing the versatility of Pandas in handling time series data.

**Future Scope**

* Apply more advanced time series techniques, such as exponential smoothing or ARIMA models, if a true temporal dataset is available.
* Perform segmentation based on demographic data (e.g., age groups or ticket classes) to uncover deeper insights.