Solar Farm Code Documentation

Mihret Akalu

Submission Date: December 8, 2024

Solar Farm Code Documentation

Project Overview

This script performs **Exploratory Data Analysis (EDA)** and **data preprocessing** on three solar farm datasets: **Benin, Sierra Leone, and Togo**. It analyzes statistical summaries, checks for anomalies, visualizes key patterns, and prepares cleaned datasets for further analysis.

Dependencies

Ensure the following libraries are installed:

Code:

```
pip install pandas matplotlib seaborn scipy windrose
```

Libraries Import

These libraries are used throughout the analysis:

Code:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import zscore
from windrose import WindroseAxes
```

Dataset Loading

Load datasets for Benin, Sierra Leone, and Togo. Analyze missing values and get general statistics.

Code:

```
# Define paths for datasets
datasets = {
    "Benin": "data/benin-malanville.csv",
    "Sierra Leone": "data/sierraleone-bumbuna.csv",
    "Togo": "data/togo-dapaong_qc.csv"
}
# Loop through each dataset
for name, path in datasets.items():
    print(f"\n--- {name} Dataset ---")
    try:
        # Load dataset
        data = pd.read_csv(path)

        # Display dataset information
        print("Dataset Overview:")
```

```
        print(data.info())
        # Display summary statistics
        print("\nSummary Statistics:")
        print(data.describe())
        # Check for missing values
        print("\nMissing Values:")
        print(data.isnull().sum())
    except Exception as e:
        print(f"Error loading {name}: {e}")
```

What This Does:

1. Loops through all three datasets.
2. Prints dataset summary, column types, and missing data.

Missing Value Analysis

Analyzes missing values and anomalies in the data (e.g., negative values).

Code:

```
columns_to_check = ['GHI', 'DNI', 'DHI']
for col in columns_to_check:
    if col in data.columns:
        print(f"{col}: {sum(data[col] < 0)} negative values")
```

What This Does:

- Checks for anomalies (negative values) in `GHI`, `DNI`, and `DHI`.

Correlation Heatmap

Visualizes correlations between numeric columns.

Code:

```
if data.select_dtypes(include='number').shape[1] > 1:
    corr_matrix = data.corr()
    sns.heatmap(corr_matrix, annot=True, cmap="coolwarm")
    plt.title(f"Correlation Heatmap for {name}")
    plt.show()
```

What This Does:

- Plots a heatmap to check relationships between numeric columns.

Histograms

Visualizes frequency distributions of numeric columns.

Code:

```
data.hist(bins=20, figsize=(12, 8))
plt.suptitle(f"Histograms for {name}")
plt.show()
```

What This Does:

- Plots histograms for numeric columns to observe patterns and distributions.

Time Series Trend: GHI over Time

Plots **GHI** trends if **Timestamp** data is available.

Code:

```
if 'Timestamp' in data.columns:
    data['Timestamp'] = pd.to_datetime(data['Timestamp'])
    data.set_index('Timestamp', inplace=True)
    # Plot GHI over time
    if 'GHI' in data.columns:
        data['GHI'].plot(title=f"GHI Over Time in {name}")
        plt.xlabel('Time')
        plt.ylabel('GHI')
        plt.show()
```

What This Does:

1. Converts string timestamps to datetime format.
2. Plots **GHI over time** to analyze trends.

Windrose Visualization

Visualizes wind speed and direction trends.

Code:

```
if {'WS', 'WD'}.issubset(data.columns):
    ax = WindroseAxes.from_ax()
    ax.bar(data['WD'], data['WS'], normed=True, opening=0.8,
edgecolor='white')
    ax.set_legend()
    plt.title(f"Wind Rose for {name}")
    plt.show()
```

What This Does:

- Visualizes wind speed and direction using a windrose chart.

Data Cleaning

Cleans anomalies and prepares data for modeling.

Steps Taken:

1. Handle missing GHI values by filling them with the mean.
2. Drop unnecessary columns (`Comments`).
3. Remove anomalies using z-score filtering.

Code:

```
# Handle missing values
data['GHI'].fillna(data['GHI'].mean(), inplace=True)
# Drop unnecessary column
data.drop('Comments', axis=1, inplace=True)
# Remove anomalies using z-score
z_scores = zscore(data['GHI'])
data_cleaned = data[(z_scores > -3) & (z_scores < 3)]
# Save cleaned data
data_cleaned.to_csv(f"data/{name}_cleaned.csv", index=False)
```

What This Does:

1. Replaces missing `GHI` values with their average.
2. Drops the **Comments** column entirely, as it is filled with null values.
3. Removes statistical outliers (based on Z-score thresholds) from **GHI** values.
4. Saves the cleaned datasets as CSV files for further modeling.

Final Output

1. **Cleaned Datasets Saved:**
    - data/Benin_cleaned.csv
    - data/Sierra_Leone_cleaned.csv
    - data/Togo_cleaned.csv
2. **Visualizations like:**
    - Correlation Heatmaps.
    - GHI Over Time plots.
    - Histograms & Windrose Charts.