# USER DOCUMENTATION FOR WEATHER DATA ANALYSIS AND FORECASTING CODE

---

## OVERVIEW

This code provides a comprehensive framework for analysing, visualizing, and forecasting weather data. It uses several machine learning models, including LSTM, ARIMA, and Prophet, to predict future weather trends based on historical data. The code is structured into various functions to handle data preprocessing, model training, evaluation, and visualization.

---

## 1. PREREQUISITES

- Python 3.x
- Required Libraries:
    - pandas
    - numpy
    - matplotlib
    - tensorflow
    - keras
    - statsmodels
    - sklearn
    - scipy
    - prophet
    - kerastuner

Install the required libraries using pip:

```
pip install pandas numpy matplotlib tensorflow keras
statsmodels sklearn scipy prophet kerastuner
```

---

## 2. DATA LOADING And PREPROCESSING

### 2.1 Data Loading

**Function:** load_dataset(filepath)

- **Purpose:** Loads weather data from a CSV file and sorts it by date.
- **Parameters:**
    - **filepath (str):** Path to the CSV file.

- **Returns**:
    - A pandas DataFrame sorted by date or None if an error occurs.

**2.2 Preprocessing Data**

**Function:** preprocess_data(df)

- **Purpose:** Fills missing values in the data with the column mean and sorts by date.
- **Parameters:**
    - df (pd.DataFrame): Input DataFrame.
- **Returns:**
    - Processed DataFrame with filled missing values and sorted by date.

---

## 3. EXPLORATORY DATA ANALYSIS (EDA)

**3.1 Aggregating Data**

**Function:** aggregate_station_data(df)

- **Purpose:** Aggregates data by station, calculating the mean for temperature and sum for precipitation and snow depth.
- **Parameters:**
    - df (pd.DataFrame): Input DataFrame with weather data.
- **Returns:**
    - Aggregated DataFrame by station.

**3.2 Finding Top Stations**

**Function:** find_top_stations(df, column, n=5, largest=True)

- **Purpose:** Finds the top or lowest n stations based on a specified parameter.
- **Parameters:**
    - df (pd.DataFrame): Input DataFrame.
    - column (str): Column name for ranking.
    - n (int): Number of stations to return (default: 5).
    - largest (bool): Return largest values if True (default: True).
- **Returns:**
    - DataFrame with top or lowest stations.

**3.3 Plotting Data**

**Function:** plot_top_stations(df, column, title, xlabel, ylabel)

- **Purpose:** Plots the top or lowest n stations based on a specified parameter.

- **Parameters:**
    - df (pd.DataFrame): Input DataFrame.
    - column (str): Column name to plot.
    - title (str): Plot title.
    - xlabel (str): X-axis label.
    - ylabel (str): Y-axis label.

---

## 4. STATISTICAL ANALYSIS

### 4.1 Displaying Summary Statistics

**Function:** display_summary_statistics(df, name, columns)

- **Purpose:** Displays summary statistics for specified columns of a DataFrame.
- **Parameters:**
    - df (pd.DataFrame): Input DataFrame.
    - name (str): Name for the data displayed.
    - columns (list): List of columns for which to display statistics.

### 4.2 Plotting Histograms and KDE

**Function:** plot_histograms_with_stats(df, name, cols, col_names, units, figsize, suptitle_y)

- **Purpose:** Plots histograms with statistics for a given dataset.
- **Parameters:**
    - df (pd.DataFrame): Input DataFrame.
    - name (str): Dataset name.
    - cols (list): Column names to plot.
    - col_names (list): Column display names.
    - units (list): Units for each column.
    - figsize (tuple): Figure size.
    - suptitle_y (float): Y position for subplot title.

---

## 5. TIME SERIES ANALYSIS

### 5.1 ARIMA Model

**Function:** run_arima_pipeline(filepath, column, order=(5, 1, 0), test_size=0.2)

- **Purpose:** Runs ARIMA model evaluation pipeline.
- **Parameters:**
    - filepath (str): Path to the CSV file.
    - column (str): Column to be forecasted.
    - order (tuple): (p, d, q) order of ARIMA model (default: (5, 1, 0)).
    - test_size (float): Proportion of data for test set (default: 0.2).
- **Returns:**
    - RMSE and MAE values for ARIMA model.

### 5.2 Prophet Model

**Function:** run_prophet_pipeline(filepath, column, test_size=0.2)

- **Purpose:** Runs Prophet model pipeline from data loading to evaluation.
- **Parameters:**
    - filepath (str): Path to the CSV file.
    - column (str): Column to be forecasted.
    - test_size (float): Proportion of data for test set (default: 0.2).
- **Returns:**
    - RMSE and MAE values for Prophet model.

---

## 6. LSTM MODEL FOR TIME SERIES FORECASTING

### 6.1 Building and Training LSTM

**Function:** build_and_train_lstm_model(X_train, y_train, input_shape, epochs=20, batch_size=32)

- **Purpose:** Builds and trains an LSTM model.
- **Parameters:**
    - X_train (np.array): Training input data.
    - y_train (np.array): Training target data.
    - input_shape (tuple): Shape of the input data.
    - epochs (int): Number of epochs for training (default: 20).
    - batch_size (int): Batch size for training (default: 32).
- **Returns:**
    - Trained LSTM model.

### 6.2 LSTM Cross-Validation

**Function:** time_series_cross_validation(data, seq_length=30, n_splits=5)

- **Purpose:** Performs time series cross-validation for LSTM model.
- **Parameters:**
    - data (np.array): Input data.
    - seq_length (int): Length of each sequence (default: 30).
    - n_splits (int): Number of cross-validation splits (default: 5).
- **Returns:**
    - Average RMSE and MAE scores across all splits.

---

## 7. ADVANCED FORECASTING WITH LSTM AND HYPERPARAMETER TUNING

### 7.1 Tuning and Training with Bayesian Optimization

**Function:** tune_and_train_model(X_train, y_train, input_shape)

- **Purpose:** Tunes and trains an LSTM model using Bayesian Optimization.
- **Parameters:**
    - X_train (np.array): Training input sequences.
    - y_train (np.array): Training target values.
    - input_shape (tuple): Shape of the input data.
- **Returns:**
    - Best trained LSTM model.

### 7.2 Future Forecasting

**Function:** predict_future(model, data, scaler, seq_length, num_features, steps=1095)

- **Purpose:** Predicts future values using the trained model.
- **Parameters:**
    - model (Sequential): Trained LSTM model.
    - data (np.ndarray): Scaled feature data.
    - scaler (MinMaxScaler): Scaler used to normalize the data.
    - seq_length (int): Length of each sequence.
    - num_features (int): Number of features in the data.
    - steps (int): Number of future steps to predict (default: 1095).

- **Returns:**
    - Predicted future values in original scale.

---

## 8. MODEL EVALUATION AND COMPARISON

### 8.1 Evaluating Models

**Function:** calculate_metrics(y_test, y_pred, columns)

- **Purpose:** Calculates and prints RMSE and MAE metrics for the test set.
- **Parameters:**
    - y_test (np.ndarray): Actual test set values.
    - y_pred (np.ndarray): Predicted test set values.
    - columns (list): Names of columns being predicted.

### 8.2 Plotting Model Comparisons

**Function:** plot_model_comparison(models, rmse_values, mae_values)

- **Purpose:** Plots a comparison of RMSE and MAE values across different models.
- **Parameters:**
    - models (list of str): Names of the models.
    - rmse_values (list of float): RMSE values for each model.
    - mae_values (list of float): MAE values for each model.

---

## 9. CONCLUSION

This code provides a robust framework for analysing and forecasting weather data using various machine learning models. By following the modular structure of functions, users can easily preprocess data, train models, evaluate performance, and visualize results. The flexibility of the code allows for integration with other datasets and the application of additional models or tuning techniques.