**CODE FOR PROJECT**

Problem Statement:Understanding weather patterns and trends is essential for various applications

such as agriculture, disaster management, and climate research. This project will

involve analyzing and visualizing weather data to uncover insights and trends. By

leveraging MATLAB’s powerful data analysis and visualization capabilities, you

will develop a comprehensive understanding of how to handle real-world data.

% Specify the file name

filename = 'weather.xlsx';

% Import the data

data = readmatrix(filename);

% Display the first few rows to understand the structure

disp('Preview of imported data:');

disp(data(1:5, :));

**Inspect the data**

% Check for the size of the dataset

disp(['Data size: ', num2str(size(data))]);

% Use `summary` or manually check the range of values (if applicable)

disp('Summary statistics:');

disp(['Min: ', num2str(min(data, [], 'all'))]);

disp(['Max: ', num2str(max(data, [], 'all'))]);

% Check for NaN values (missing values)

numMissing = sum(isnan(data), 'all');

disp(['Number of missing values: ', num2str(numMissing)]);

**Handle Missing Values**

% Replace NaN values with the mean of their respective columns

for i = 1:size(data, 2)

column = data(:, i);

if any(isnan(column))

columnMean = mean(column, 'omitnan');

column(isnan(column)) = columnMean;

data(:, i) = column;

end

end

disp('Missing values replaced with column means.');

% Remove rows containing NaN values

data = data(~any(isnan(data), 2), :);

disp('Rows with missing values removed.');

**Saving the cleaned data**

% Save to a new CSV file

cleanedFilename = 'cleaned\_weather\_data.csv';

writematrix(data, cleanedFilename);

disp(['Cleaned data saved to ', cleanedFilename]);

Organizing data in matlab structure

% Use a matrix (already done in Task 1.1)

weatherMatrix = readmatrix(filename);

% Display the first few rows

disp('Preview of the matrix:');

disp(weatherMatrix(:, :));

%%---------------------------------------------%%

Extracting relevant columns for analysis

% Example: Assuming columns 1, 2, and 3 correspond to temperature, humidity, and wind speed

temperature = weatherMatrix(:, 11);

humidity\_prec = weatherMatrix(:, 1);

windSpeed = weatherMatrix(:, 14);

% Combine into a new matrix for analysis

analysisMatrix = weatherMatrix(:, [1, 2, 3]);

% Display the extracted data

disp('Extracted columns for analysis (as a matrix):');

disp(analysisMatrix(1:5, :));

%%-----------------------------------------%%

Calculate basic statistical measure

% Assuming 'weatherMatrix' contains the data

% Extract columns

% Time (e.g., serial dates or indices)

temperature = weatherMatrix(:, 11); % Temperature

% Humidity

precipitation = weatherMatrix(:, 1);% Precipitation

windSpeed = weatherMatrix(:, 14);

% Calculate basic statistics

meanTemp = mean(temperature);

medianTemp = median(temperature);

stdTemp = std(temperature);

meanwindspeed = mean(windSpeed);

medianwindspeed = median(windSpeed);

stdwindspeed = std(windSpeed);

%meanHum = mean(humidity);

%medianHum = median(humidity);

%stdHum = std(humidity);

meanPrec = mean(precipitation);

medianPrec = median(precipitation);

stdPrec = std(precipitation);

% Display results

disp('Basic Statistical Measures:');

fprintf('Temperature: Mean = %.2f, Median = %.2f, StdDev = %.2f\n', meanTemp, medianTemp, stdTemp);

%fprintf('Humidity: Mean = %.2f, Median = %.2f, StdDev = %.2f\n', meanHum, medianHum, stdHum);

fprintf('Precipitation: Mean = %.2f, Median = %.2f, StdDev = %.2f\n', meanPrec, medianPrec, stdPrec);

fprintf('windSpeed: Mean = %.2f, Median = %.2f, StdDev = %.2f\n', meanwindspeed, medianwindspeed, stdwindspeed);

**Plot**

% Plot each parameter

figure;

% Temperature

subplot(3,1,1);

plot(temperature, '-b', 'LineWidth', 1.5);

title('Temperature Over Time');

xlabel('Time');

ylabel('Temperature (°C)');

% Humidity

subplot(3,1,2);

plot(humidity\_prec, '-g', 'LineWidth', 1.5);

title('Humidity Over Time');

xlabel('Time');

ylabel('Humidity (%)');

% Precipitation

subplot(3,1,3);

plot(windSpeed, '-c', 'LineWidth', 1.5);

title('Precipitation Over Time');

xlabel('Time');

ylabel('windSpeed (m/s)');

**Using moving average**

% Define window size for moving average (e.g., 7 for weekly averages)

windowSize = 7;

% Apply moving averages

temp\_smooth = movmean(temperature, windowSize);

hum\_smooth = movmean(humidity\_prec, windowSize);

wind\_smooth = movmean(windSpeed, windowSize);

% Plot smoothed data

figure;

% Smoothed Temperature

subplot(3,1,1);

plot(temp\_smooth, '-r', 'LineWidth', 1.5);

title('Smoothed Temperature (7-day Moving Average)');

xlabel('Time');

ylabel('Temperature (°C)');

% Smoothed Humidity

subplot(3,1,2);

plot(hum\_smooth, '-m', 'LineWidth', 1.5);

title('Smoothed Humidity (7-day Moving Average)');

xlabel('Time');

ylabel('Humidity\_prec (%)');

% Smoothed Precipitation

subplot(3,1,3);

plot(wind\_smooth, '-k', 'LineWidth', 1.5);

title('Smoothed windSpeed (7-day Moving Average)');

xlabel('Time');

ylabel('Precipitation (m/s)');

**Co-relation**

% Assuming 'weatherMatrix' contains the data

% Extract columns for analysis

temperature = weatherMatrix(:, 11);

humidity\_prec = weatherMatrix(:, 1);

windSpeed = weatherMatrix(:, 14);

% Combine data into a matrix

dataMatrix = [temperature, humidity\_prec, windSpeed];

% Compute correlation matrix

correlationMatrix = corrcoef(dataMatrix);

% Display the correlation matrix

disp('Correlation Coefficients:');

disp(correlationMatrix);

**Scatter plot**

% Create scatter plots

figure;

% Temperature vs Humidity

subplot(2, 2, 1);

scatter(temperature, humidity\_prec, 'filled');

title('Temperature vs Humidity\_prec');

xlabel('Temperature (°C)');

ylabel('Humidity (%)');

**Linear Regression using Polyfit & Polyval**

% Assuming 'weatherMatrix' is the data matrix

% Columns: [Time, Temperature, Humidity, Precipitation]

% Extract relevant columns

temperature = weatherMatrix(:, 11);

humidity\_prec = weatherMatrix(:, 1);

windSpeed = weatherMatrix(:, 14);

% Linear regression: Temperature vs. Humidity

coeff\_humidity\_prec = polyfit(temperature, humidity\_prec, 1); % Fit a linear model

predicted\_humidity\_prec = polyval(coeff\_humidity\_prec, temperature); % Evaluate the model

% Linear regression: Temperature vs. Precipitation

coeff\_windSpeed = polyfit(temperature, windSpeed, 1); % Fit a linear model

predicted\_windSpeed = polyval(coeff\_windSpeed, temperature); % Evaluate the model

**Plot Data and Regression Lines**

figure;

% Plot: Temperature vs Humidity\_prec

subplot(2, 1, 1);

scatter(temperature, humidity\_prec, 'filled'); % Original data points

hold on;

plot(temperature, predicted\_humidity\_prec, '-r', 'LineWidth', 1.5); % Regression line

hold off;

title('Linear Regression: Temperature vs Humidity');

xlabel('Temperature (°C)');

ylabel('Humidity\_prec (%)');

legend('Data', 'Regression Line');

% Plot: Temperature vs windSpeed

subplot(2, 1, 2);

scatter(temperature, windSpeed, 'filled'); % Original data points

hold on;

plot(temperature, predicted\_windSpeed, '-b', 'LineWidth', 1.5); % Regression line

hold off;

title('Linear Regression: Temperature vs windSpeed');

xlabel('Temperature (°C)');

ylabel('windSpeed (m/s)');

legend('Data', 'Regression Line');

**3D surface plot for Temparature variation**

% Simulated data for spatial variation

[x, y] = meshgrid(1:10, 1:10); % Create a 10x10 grid

z = rand(10, 10) \* 10 + 20; % Simulated temperature data

% Create 3D surface plot

figure;

surf(x, y, z);

title('Temperature Variation Over Space');

xlabel('X Coordinate');

ylabel('Y Coordinate');

zlabel('Temperature (°C)');

colormap('jet'); % Use a color map

colorbar;

**3D Mesh plot for temparature variation**

figure;

mesh(x, y, z);

title('Temperature Variation Over Space (Mesh Plot)');

xlabel('X Coordinate');

ylabel('Y Coordinate');

zlabel('Temperature (°C)');

colormap('hot'); % Use a different color map

colorbar;