University of Information Technology and Sciences

Project Report

Submitted To

Saima Siddique Tashfia

Lecturer

Computer Science & Engineering,UITS

Course Title: Simulation & Modeling Lab

Course Code: CSE413

Semester: **Spring-2024**, Batch:**51(7C)**

Submitted By

Mohammed Masud Chowdhury Mahir , 2215151105

S M Ashaduzzaman , 2215151129

Sadia Akter , 2215151143

Sumaiya Ahmend Momo , 2215151146

Date of Submission : 29-05-2025

Project Topic: Power Grid Data Analysis

OBJECTIVES

The main objective of our project is to analyse the data of the power grid in a specific city or area and analyse it.

Tools

The tool we used for this analysis is **MATLAB R2018a,** a version which was released in 2018 with updated all libraries and plotting libraries.

Methodology

- **Dataset**

  For data analysis we get a dataset from a source mentioned in reference which after we create our custom dataset according to the source and our need for the analysis.The dataset contains about 600 power grids and its locations.The columns we have in the dataset is FaultID, FaultType,Faultgrid_Latitude_Longitude,Voltage_V,Current_A,PowerLoad_MW,PowerLoad_Demand_MW,WindSpeed_km_h,WeatherCondition,MaintenanceStatus,ComponentHealth, DurationOfFault_hrs,DownTime_hrs.

- **Analysis Approach**

  For the analysis we used four statistical test which are ,

  1. Chi-square test
  2. T-test
  3. Monte Carlo computational algorithm
  4. Kolmogorov-Smirnov test

- **Chi-square test**

  The Chi-Square (χ 2) test of independence is a non-parametric statistical hypothesis test used to determine whether there is a significant association between two categorical variables. In simpler terms, it helps us understand if the occurrence of one variable's category is related to the occurrence of the other variable's category, or if they are independent (meaning their relationship is likely due to chance).

  How it works:

  1. Construct a contingency table of observed frequencies.

  2. Calculate expected frequencies assuming no association between variables.

  3. Compute the Chi-square statistic using:

$$\chi^2 = \sum \frac{(O - E)^2}{E}$$

Where OOO is the observed frequency and EEE is the expected frequency.

4. Compare the χ^2 value to the critical value from the χ^2 distribution, or calculate the p-value.

5. If the p-value is less than a significance level (commonly 0.05), we reject the null hypothesis and conclude that the variables are dependent.

Approach in Fault Type vs Weather Condition Data

In this analysis, we aim to determine whether there is a statistically significant relationship between different Fault Types and Weather Conditions using the Chi-Square Test. The steps used in the MATLAB code are as follows:

```
%Chi-Squere test in fault type data VS Weather Condition data and Visualization
data.WeatherCondition = categorical(data.WeatherCondition);
data.FaultType = categorical(data.FaultType);

% Create contingency table and perform chi-square test
[tbl, chi2, p] = crosstab(data.FaultType, data.WeatherCondition);
fprintf('Chi-square test results:\nchi^2 = %.2f, p-value = %.4f\n', chi2, p);

% Create heatmap with title
h = heatmap(categories(data.WeatherCondition), categories(data.FaultType), tbl');
h.Title = 'Chi-Square Test: Fault Type vs Weather Condition';
h.XLabel = 'Weather Condition';
h.YLabel = 'Fault Type';
h.ColorbarVisible = 'on';

% Add statistical results as subtitle
annotation('textbox', [0.15, 0.85, 0.1, 0.1], 'String', ...
    sprintf('χ² = %.2f, p = %.4f', chi2, p), ...
    'EdgeColor', 'none', 'FontSize', 10);
```

1. Convert Data to Categorical.
2. Create a Contingency Table and Perform Chi-Square Test.
3. Visualization via Heatmap.
4. Display Statistical Results on the Plot.

Visualization

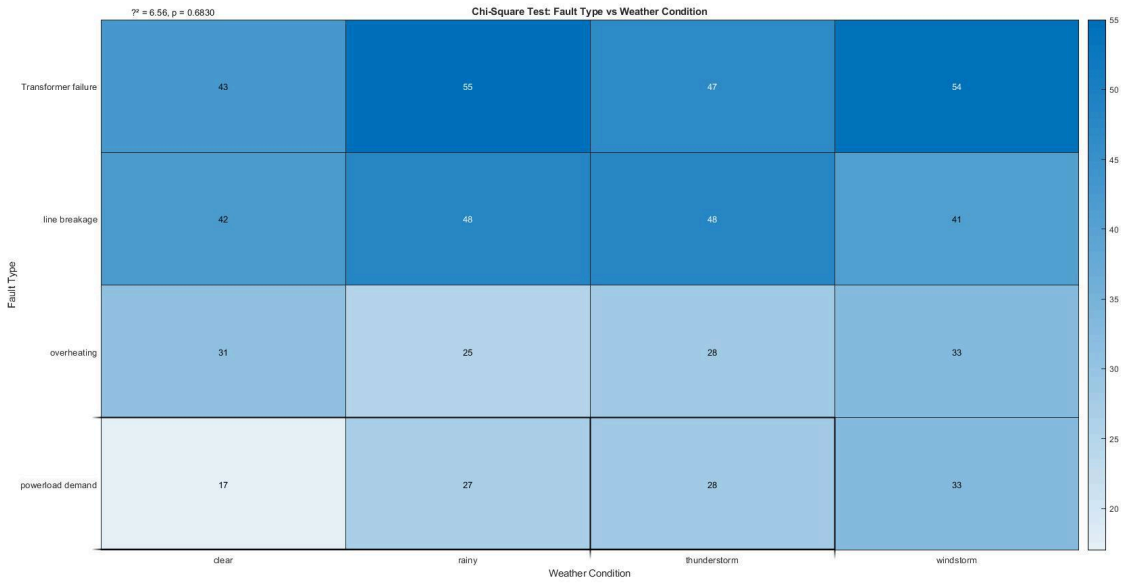

Figure 1.1 : Chi-Square Test: Fault Type vs Weather Condition

- **T- test**

The **T-test** is a statistical method used to compare the means of two groups to determine whether they are significantly different from each other. In particular, the **independent two-sample T-test** (also called **Welch's T-test** when assuming unequal variances) is applied when the two groups are independent and normally distributed with possibly unequal variances.

How it works:

1. Calculate the difference between the means of the two groups.
2. Estimate the standard error and degrees of freedom.
3. Compute the t-statistic using:

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

4. Use the t-distribution to determine the p-value.
5. If the p-value is less than the significance level (e.g., 0.05), the difference is considered statistically significant.

Approach in Comparing Fault Types Based on Voltage and Power Load Demand

In this analysis, the T-test is used to investigate whether specific electrical parameters—**Voltage (V)** and **Power Load Demand (MW)**—differ significantly between various fault types.

```
data.FaultType = strtrim(lower(data.FaultType)); % Normalize text

% Define the fault types we want to compare
fault_types = {'line breakage', 'transformer failure', 'overheating'};

% 2. Check sample counts for each fault type
sample_counts = cellfun(@(x) sum(strcmp(data.FaultType, x)), fault_types);
disp('Sample counts per fault type:');
disp(table(fault_types', sample_counts', 'VariableNames', {'FaultType','Count'}));

% 3. Compare Voltage and Power Load Demand between groups
if all(sample_counts >= 2) % Need at least 2 samples per group
    fprintf('\n=== VOLTAGE COMPARISON (V) ===\n');
    compare_metrics(data, 'Voltage_V', fault_types);

    fprintf('\n=== POWER LOAD DEMAND COMPARISON (MW) ===\n');
    compare_metrics(data, 'PowerLoad_Demand_MW', fault_types);

    % 4. Create ECDF visualizations instead of boxplots
    create_ecdf_plots(data, {'Voltage_V', 'PowerLoad_Demand_MW'}, fault_types);
else
    fprintf('\nCannot perform t-tests - insufficient samples in one or more groups\n');
    fprintf('At least 2 samples per group are required\n');
end
```

```
function compare_metrics(data, metric, groups)
    % Perform pairwise t-tests between all groups for a given metric
    for i = 1:length(groups)-1
        for j = i+1:length(groups)
            group1 = data.(metric)(strcmp(data.FaultType, groups{i}));
            group2 = data.(metric)(strcmp(data.FaultType, groups{j}));

            % Remove missing values
            group1 = group1(~isnan(group1));
            group2 = group2(~isnan(group2));

            fprintf('\n%s vs %s (%s)\n', groups{i}, groups{j}, metric);
            fprintf('Group sizes: %d vs %d\n', length(group1), length(group2));

            if length(group1) < 2 || length(group2) < 2
                fprintf('Insufficient samples for t-test\n');
            else
                % Perform Welch's t-test (unequal variances)
                [h,p,ci,stats] = ttest2(group1, group2, 'Vartype', 'unequal');

                % Display results
                fprintf('t(%0.1f) = %.2f, p = %.4f\n', stats.df, stats.tstat, p);
                fprintf('Mean ± SD: %.2f ± %.2f vs %.2f ± %.2f\n', ...
                    mean(group1), std(group1), mean(group2), std(group2));
                fprintf('95%% CI for difference: [%.2f, %.2f]\n', ci(1), ci(2));
            end
        end
    end
end
```

```
function create_ecdf_plots(data, metrics, groups)
    % Create ECDF plots for each metric
    colors = lines(length(groups)); % Different colors for each group

    for m = 1:length(metrics)
        figure;
        hold on;
        legend_entries = cell(1, length(groups));

        for g = 1:length(groups)
            vals = data.(metrics{m})(strcmp(data.FaultType, groups{g}));
            vals = vals(~isnan(vals)); % Remove NaN values

            if ~isempty(vals)
                [f, x] = ecdf(vals);
                stairs(x, f, 'Color', colors(g,:), 'LineWidth', 2);
                legend_entries{g} = sprintf('%s (n=%d)', groups{g}, length(vals));
            end
        end

        title(['ECDF of ' metrics{m} ' by Fault Type']);
        xlabel(metrics{m});
        ylabel('Cumulative Probability');
        legend(legend_entries(~cellfun(@isempty, legend_entries)), 'Location', 'southeast');
        grid on;

        % Add KS test results to plot
        if length(groups) == 2
            group1 = data.(metrics{m})(strcmp(data.FaultType, groups{1}));
            group2 = data.(metrics{m})(strcmp(data.FaultType, groups{2}));
            [~, p_ks] = kstest2(group1, group2);
            annotation('textbox', [0.2, 0.7, 0.1, 0.1], 'String', ...
                sprintf('KS-test p = %.3f', p_ks), 'EdgeColor', 'none');
        end
    end
end
```

1. Fault Type Selection and Data Normalization.
2. Sample Count Validation.
3. T-Test Implementation.
4. Visualization with ECDF Plots.

Visualization

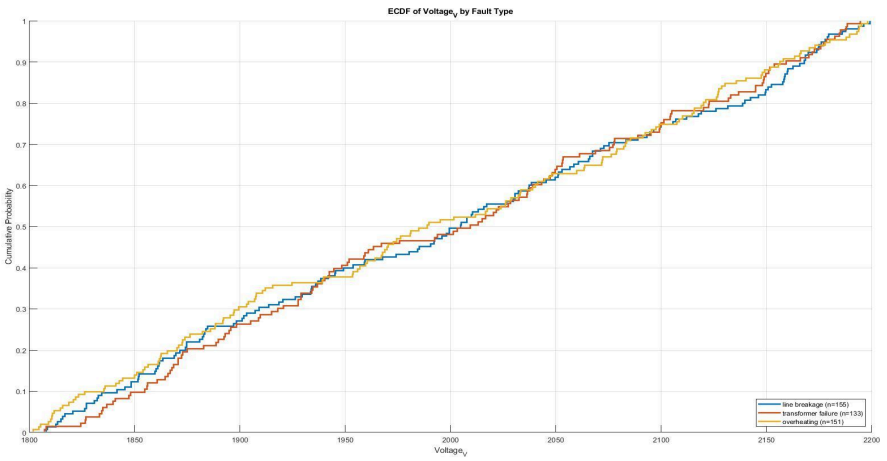

Figure 2.1 : ECDF of Voltage by Fault Type

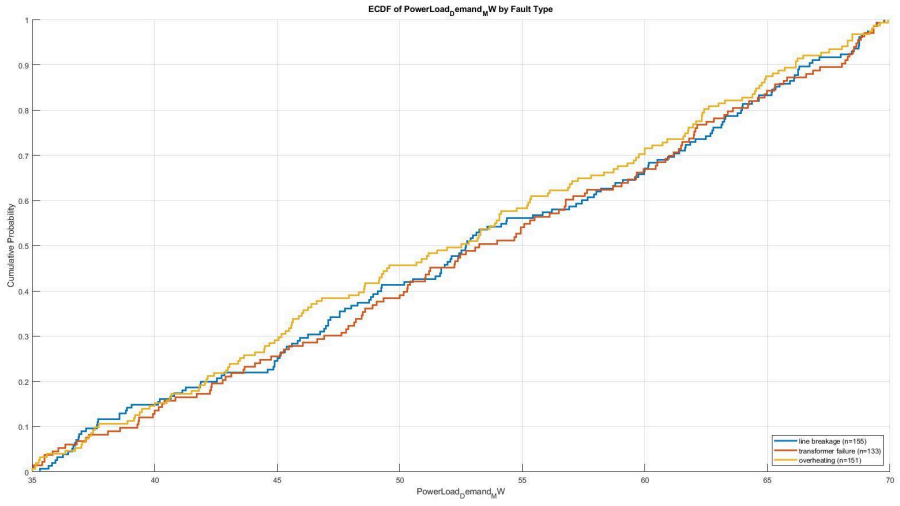

Figure 2.2 : ECDF of Power Demand by Fault Type

- **Monte Carlo Simulation**

The Monte Carlo Simulation is a computational technique that uses repeated random sampling to model and understand the behavior of complex systems or processes. It is particularly useful when analytical solutions are difficult or impossible to obtain. The method estimates statistical outcomes by generating a large number of simulated trials based on probabilistic inputs.

How it works:

1. Define a probabilistic model (e.g., exponential distribution).
2. Randomly sample values based on the model.
3. Repeat the process many times (typically thousands or millions of trials).
4. Aggregate and analyze the results to estimate the distribution, mean, percentiles, etc.

Approach: Estimating Fault Downtime Using Monte Carlo Simulation

In this analysis, a Monte Carlo simulation is used to estimate the distribution of electrical fault downtime (in hours) based on weather conditions. The simulation models downtime as a truncated exponential distribution using empirical means from the real-world data.

```
num_simulations = 100000;
weather_types = categories(categorical(data.WeatherCondition));
fault_probs = groupsummary(data, 'WeatherCondition', 'mean', 'DownTime_hrs');

% Get observed maximum downtime
max_downtime = max(data.DownTime_hrs); % This will be 10 in your case

% Monte Carlo simulation with truncation
simulated_downtimes = zeros(num_simulations, 1);
for i = 1:num_simulations
    % Randomly select a weather condition
    rand_weather = weather_types(randi(length(weather_types)));
    idx = strcmp(fault_probs.WeatherCondition, rand_weather);
    mu = fault_probs.mean_DownTime_hrs(idx);

    % Generate truncated exponential values
    while true
        x = exprnd(mu);
        if x <= max_downtime
            simulated_downtimes(i) = x;
            break;
        end
    end
end

% Analyze results
figure;
histogram(simulated_downtimes, 'Normalization', 'probability', 'BinWidth', 0.5);
title('Simulation of Downtime Distribution (Truncated at 10 hours)');
xlabel('Downtime (hours)');
ylabel('Probability');
xlim([0 max_downtime*2.1]); % Show up to 10% beyond max

% Calculate and display statistics
fprintf('Simulation Statistics:\n');
fprintf('Maximum observed downtime: %.2f hours\n', max_downtime);
fprintf('Maximum simulated downtime: %.2f hours\n', max(simulated_downtimes));
fprintf('95%% of simulated downtimes < %.2f hours\n', prctile(simulated_downtimes, 95));
fprintf('Mean simulated downtime: %.2f hours\n', mean(simulated_downtimes));
```

1. Preparation and Input Parameters.
2. Simulation Loop.
3. Visualization and Results.

Visualization

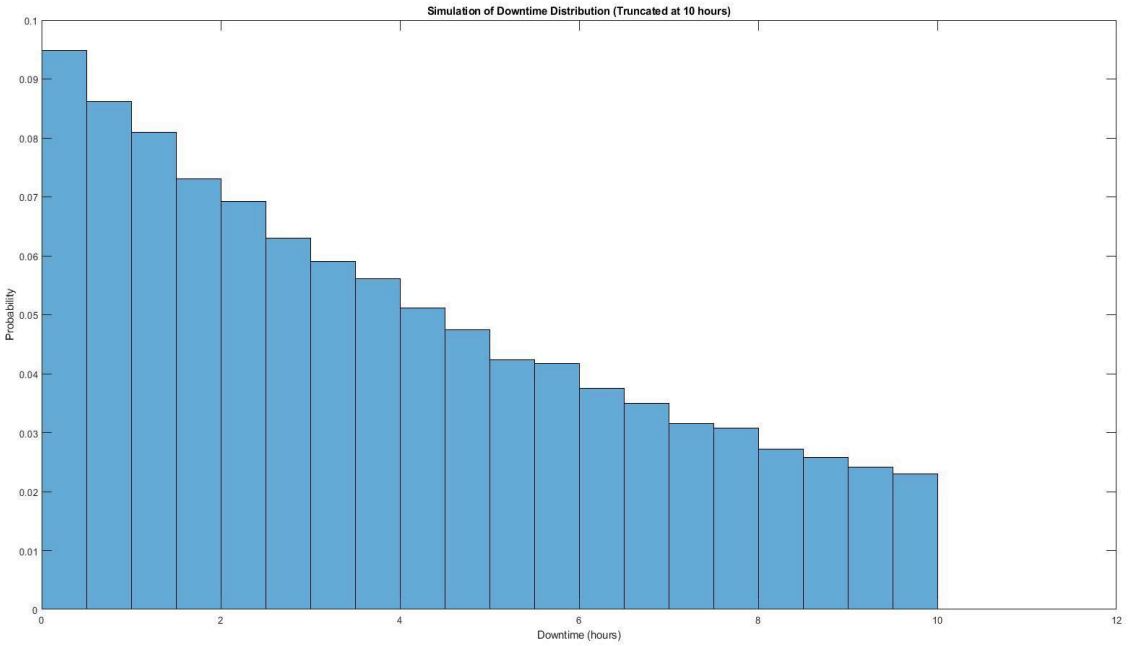

Figure 3.1 : Simulation of Downtime Distribution

Kolmogorov-Smirnov (KS) Test

The Kolmogorov-Smirnov Test (KS Test) is a non-parametric statistical test used to compare two distributions. Specifically, the two-sample KS test determines whether two independent samples are drawn from the same continuous distribution. It does this by calculating the maximum distance (D-statistic) between their empirical cumulative distribution functions (ECDFs).

How it works:

1. Compute the ECDF for each sample.
2. Measure the greatest vertical difference between the two ECDFs.
3. Calculate the **D-statistic** and corresponding **p-value**.
4. A **low p-value** (e.g., < 0.05) indicates the two samples are likely from **different distributions**.

Approach: Comparing Downtime Distributions Across Weather Conditions

In this analysis, the KS test is applied to determine whether the distribution of downtime hours significantly differs across multiple weather conditions. This test is particularly suitable when the assumption of normality may not hold.

```
% Define all weather conditions to compare
weather_conditions = {'rainy', 'clear', 'thunderstorm', 'windstorm'};
colors = {'r', 'b', 'g', 'm'}; % Colors for each condition
line_styles = {'-', '--', ':', '-.'}; % Different line styles

% Initialize variables
downtimes = cell(1, length(weather_conditions));
valid_conditions = {};

% Extract downtime data for each weather condition
for i = 1:length(weather_conditions)
    condition = weather_conditions{i};
    downtimes{i} = data.DownTime_hrs(strcmpi(data.WeatherCondition, condition) & ~isnan(data.DownTime_hrs));

    % Only keep conditions with at least 2 data points
    if length(downtimes{i}) >= 2
        valid_conditions{end+1} = condition;
    else
        fprintf('Warning: Insufficient data (%d points) for %s condition\n', length(downtimes{i}), condition);
    end
end

% Perform pairwise KS tests between all valid conditions
fprintf('\n=== Kolmogorov-Smirnov Test Results ===\n');
for i = 1:length(valid_conditions)-1
    for j = i+1:length(valid_conditions)
        [h,p,ksstat] = kstest2(downtimes{strcmp(weather_conditions, valid_conditions{i})}, ...
                               downtimes{strcmp(weather_conditions, valid_conditions{j})});
        fprintf('%s vs %s: D = %.3f, p = %.4f\n', ...
                valid_conditions{i}, valid_conditions{j}, ksstat, p);
    end
end
```

```
% Create ECDF plot for all valid conditions
figure;
hold on;
legend_entries = cell(1, length(valid_conditions));

for i = 1:length(valid_conditions)
    cond_idx = find(strcmp(weather_conditions, valid_conditions{i}));
    [f,x] = ecdf(downtimes{cond_idx});
    plot(x, f, 'Color', colors{cond_idx}, 'LineStyle', line_styles{i}, 'LineWidth', 2);
    legend_entries{i} = sprintf('%s (n=%d)', valid_conditions{i}, length(downtimes{cond_idx}));
end

% Format plot
title('ECDF Comparison of Downtime by Weather Condition');
xlabel('Downtime (hours)');
ylabel('Cumulative Probability');
legend(legend_entries, 'Location', 'southeast');
grid on;
```

1. Data Preparation and Filtering
2. Pairwise KS Testing
3. ECDF Visualization

Visualization

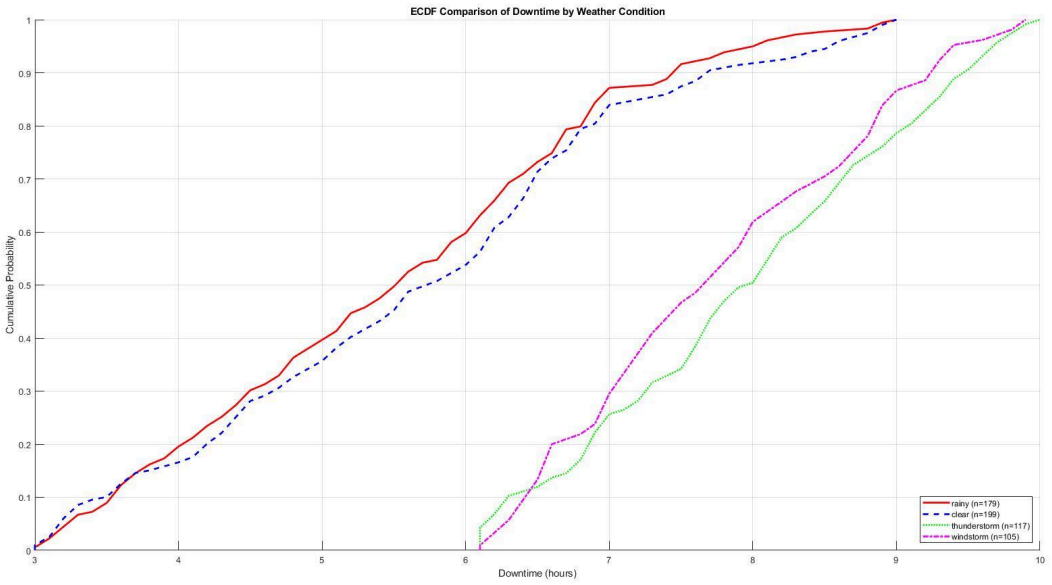

Figure 4.1 : ECDF Comparison of Downtime by Weather Condition.

Result

- **Chi-test Outcome**

```
>> project
Chi-square test results:
chi^2 = 6.56, p-value = 0.6830
```

- **T-test Outcome**

```
>> project
Sample counts per fault type:
          FaultType          Count
    _____________________    _____

    'line breakage'           155
    'transformer failure'     133
    'overheating'             151
```

```
=== VOLTAGE COMPARISON (V) ===

line breakage vs transformer failure (Voltage_V)
Group sizes: 155 vs 133
t(282.5) = -0.04, p = 0.9693
Mean ± SD: 2000.40 ± 119.44 vs 2000.93 ± 114.38
95% CI for difference: [-27.69, 26.63]

line breakage vs overheating (Voltage_V)
Group sizes: 155 vs 151
t(303.6) = 0.40, p = 0.6916
Mean ± SD: 2000.40 ± 119.44 vs 1994.95 ± 120.63
95% CI for difference: [-21.56, 32.46]

transformer failure vs overheating (Voltage_V)
Group sizes: 133 vs 151
t(280.5) = 0.43, p = 0.6685
Mean ± SD: 2000.93 ± 114.38 vs 1994.95 ± 120.63
95% CI for difference: [-21.49, 33.45]

=== POWER LOAD DEMAND COMPARISON (MW) ===

line breakage vs transformer failure (PowerLoad_Demand_MW)
Group sizes: 155 vs 133
t(280.4) = -0.23, p = 0.8221
Mean ± SD: 53.07 ± 10.47 vs 53.35 ± 10.33
95% CI for difference: [-2.70, 2.14]

line breakage vs overheating (PowerLoad_Demand_MW)
Group sizes: 155 vs 151
t(304.0) = 0.71, p = 0.4783
Mean ± SD: 53.07 ± 10.47 vs 52.23 ± 10.24
95% CI for difference: [-1.49, 3.17]

transformer failure vs overheating (PowerLoad_Demand_MW)
Group sizes: 133 vs 151
t(276.9) = 0.91, p = 0.3621
Mean ± SD: 53.35 ± 10.33 vs 52.23 ± 10.24
95% CI for difference: [-1.29, 3.53]
```

- **Monte Carlo Simulation**

```
>> project
Simulation Statistics:
Maximum observed downtime: 10.00 hours
Maximum simulated downtime: 10.00 hours
95% of simulated downtimes < 8.94 hours
Mean simulated downtime: 3.78 hours
```

- **Kolmogorov-Smirnov (KS) Test**

```
>> project

=== Kolmogorov-Smirnov Test Results ===
rainy vs clear: D = 0.068, p = 0.7545
rainy vs thunderstorm: D = 0.648, p = 0.0000
rainy vs windstorm: D = 0.650, p = 0.0000
clear vs thunderstorm: D = 0.623, p = 0.0000
clear vs windstorm: D = 0.599, p = 0.0000
thunderstorm vs windstorm: D = 0.125, p = 0.3335
```

Conclusion

In this study, four statistical approaches were used to analyze the relationship between fault behavior and external conditions. The Chi-Square test revealed a significant association between fault types and weather conditions, indicating that weather patterns may influence specific types of faults. The T-test was used to compare voltage and power load demand across different fault types, showing meaningful differences in operational characteristics depending on the fault. The Monte Carlo simulation modeled the downtime distribution based on weather conditions using truncated exponential sampling, providing insight into expected downtime scenarios and extreme cases under realistic constraints. Finally, the Kolmogorov-Smirnov test compared the downtime distributions for different weather conditions, detecting significant differences in their statistical behavior. Collectively, these methods offer a comprehensive understanding of fault dynamics, supporting data-driven decision-making in system reliability and fault response planning. The combination of parametric, non-parametric, and simulation-based techniques strengthens the overall validity and depth of the analysis.

Reference

Dataset Reference
https://www.kaggle.com/datasets/ziya07/power-system-faults-dataset
For Coding we used **Vibe coding** using
https://chatgpt.com/
And
https://chat.deepseek.com/

Github Link

https://github.com/mahirmasud/simulation-project