Source Code

Generation of dataset :*Operational Downtime Log*

```
# Generating a larger synthetic dataset for failure times to simulate more sample failures


# New parameters for generating a larger dataset

num_failures_large = 100  # Increased number of failures

failure_times_large = np.sort(np.cumsum(np.random.exponential(scale=15, size=num_failures_large)))


# Limit failure times within a 1000-hour test period

failure_times_large = failure_times_large[failure_times_large <= 1000]


# Create DataFrame for the larger generated failure data

failure_data_large = pd.DataFrame({'Failure_Number': range(1, len(failure_times_large) + 1),

                                   'Failure_Time': failure_times_large})


# Display the larger dataset to the user

tools.display_dataframe_to_user(name="Generated Larger Failure Data", dataframe=failure_data_large)
```

```
# Displaying the first few rows to verify
failure_data_large.head()
```

Various Models :

```
# Re-calculating the failure intensity, MTBF, reliability, and availability for each model based on the user-provided data

# JM Model (recalculating with adjusted parameters to avoid issues)
calculation_summary_user_data = pd.DataFrame({
    "Failure_Number": failure_data_user_provided["Failure_Number"],
    "Time": failure_data_user_provided["Time"],
    "JM_Failure_Intensity": [jelinski_moranda_failure_intensity(N_JM_demo, phi_JM_demo, k)
                             for k in range(1, len(failure_data_user_provided) + 1)],
    "JM_MTBF": [adjusted_jelinski_moranda_mtbf(N_JM_demo, phi_JM_demo, k)
                for k in range(1, len(failure_data_user_provided) + 1)]
})

# NHPP Model calculations
calculation_summary_user_data["NHPP_Failure_Intensity"] = nhpp_failure_intensity(
    a_NHPP_demo, b_NHPP_demo, calculation_summary_user_data["Time"]
```

```
)
calculation_summary_user_data["NHPP_MTBF"] = nhpp_mtbf(
    a_NHPP_demo, b_NHPP_demo, calculation_summary_user_data["Time"]
)

# Weibull Model calculations
calculation_summary_user_data["Weibull_Failure_Intensity"] =
weibull_failure_intensity(
    lambda_Weibull_demo, beta_Weibull_demo,
calculation_summary_user_data["Time"]
)
calculation_summary_user_data["Weibull_MTBF"] = weibull_mtbf(
    lambda_Weibull_demo, beta_Weibull_demo,
calculation_summary_user_data["Time"]
)

# Reliability over a 10-hour period
time_for_reliability = 10  # fixed time to compute reliability

calculation_summary_user_data["Reliability_JM"] = [
    reliability_jm(phi_JM_demo, N_JM_demo, k, time_for_reliability)
    for k in range(1, len(failure_data_user_provided) + 1)
]
calculation_summary_user_data["Reliability_NHPP"] = reliability_nhpp(
```

```
    a_NHPP_demo, b_NHPP_demo, time_for_reliability
)
calculation_summary_user_data["Reliability_Weibull"] = reliability_weibull(
    lambda_Weibull_demo, beta_Weibull_demo, time_for_reliability
)

# Availability calculation with an assumed MTTR
MTTR_demo = 2  # Mean Time to Repair

calculation_summary_user_data["Availability_JM"] =
calculation_summary_user_data["JM_MTBF"] / (
    calculation_summary_user_data["JM_MTBF"] + MTTR_demo
)
calculation_summary_user_data["Availability_NHPP"] =
calculation_summary_user_data["NHPP_MTBF"] / (
    calculation_summary_user_data["NHPP_MTBF"] + MTTR_demo
)
calculation_summary_user_data["Availability_Weibull"] =
calculation_summary_user_data["Weibull_MTBF"] / (
    calculation_summary_user_data["Weibull_MTBF"] + MTTR_demo
)

# Display the final results table with all calculations
```

```
tools.display_dataframe_to_user(name="Final Reliability Model Results",
dataframe=calculation_summary_user_data)


# Displaying the calculation summary to check the results

calculation_summary_user_data.head()
```

U-value calculations:

```
# Function to calculate U-values for prediction models based on previous failures

def calculate_predictions_and_u_values(failure_times, num_previous):

    predictions = []

    u_values = []


    # Loop through the data starting from the point where we have enough previous
data

    for i in range(num_previous, len(failure_times)):

        # Calculate mean of the specified number of previous failure times

        prediction = failure_times[i - num_previous:i].mean()

        predictions.append(prediction)


        # Calculate the u-value: |Predicted - Actual| / Actual

        actual = failure_times[i]

        u_value = abs(prediction - actual) / actual

        u_values.append(u_value)
```

```
    return predictions, u_values

# Extract the failure times from the dataset
failure_times = failure_data_updated["Failure_Time"]

# Calculate predictions and u-values for 2, 3, and 4 previous failures
predictions_2, u_values_2 = calculate_predictions_and_u_values(failure_times, 2)
predictions_3, u_values_3 = calculate_predictions_and_u_values(failure_times, 3)
predictions_4, u_values_4 = calculate_predictions_and_u_values(failure_times, 4)

# Create a DataFrame to tabulate the results
u_values_table = pd.DataFrame({
    "Failure_Number": failure_data_updated["Failure_Number"][4:],  # Align to longest series (4 previous failures)
    "Actual_Failure_Time": failure_data_updated["Failure_Time"][4:].values,  # Trim to match predictions
    "Prediction_2_Prev": [None, None] + predictions_2,  # Offset by 2 to align with actual data
    "U_Value_2_Prev": [None, None] + u_values_2,
    "Prediction_3_Prev": [None, None, None] + predictions_3,  # Offset by 3
    "U_Value_3_Prev": [None, None, None] + u_values_3,
    "Prediction_4_Prev": [None, None, None, None] + predictions_4,  # Offset by 4
    "U_Value_4_Prev": [None, None, None, None] + u_values_4
```

```
})

tools.display_dataframe_to_user(name="U-Values Prediction Model Table", dataframe=u_values_table)

# Displaying the first few rows to verify alignment and calculations
u_values_table.head(10)

# Adjusting alignment for consistent length by trimming the data to the minimum available points
min_length = len(failure_data_updated) - 4  # The minimum length after 4 previous failures

# Re-calculating the table to fit the trimmed length
u_values_table = pd.DataFrame({
    "Failure_Number": failure_data_updated["Failure_Number"][4:4 + min_length].reset_index(drop=True),
    "Actual_Failure_Time": failure_data_updated["Failure_Time"][4:4 + min_length].reset_index(drop=True),
    "Prediction_2_Prev": predictions_2[:min_length],
    "U_Value_2_Prev": u_values_2[:min_length],
    "Prediction_3_Prev": predictions_3[:min_length],
    "U_Value_3_Prev": u_values_3[:min_length],
    "Prediction_4_Prev": predictions_4[:min_length],
    "U_Value_4_Prev": u_values_4[:min_length]
```

```
})

# Display the final table with predictions and u-values
tools.display_dataframe_to_user(name="U-Values Prediction Model Table (Aligned)", dataframe=u_values_table)

# Displaying the first few rows to verify alignment and calculations
u_values_table.head()

# Adjusting lengths to match between actual and predictions for each model
# Align to minimum shared length for consistent comparison

# Minimum length to match predictions starting points
min_length_2 = min(len(actual_times_for_comparison), len(predictions_2))
min_length_3 = min(len(actual_times_for_comparison), len(predictions_3))
min_length_4 = min(len(actual_times_for_comparison), len(predictions_4))

# Truncate each series to match lengths for KS calculations
actual_for_2_prev = actual_times_for_comparison[:min_length_2]
pred_for_2_prev = predictions_2[:min_length_2]

actual_for_3_prev = actual_times_for_comparison[:min_length_3]
pred_for_3_prev = predictions_3[:min_length_3]
```

```
actual_for_4_prev = actual_times_for_comparison[:min_length_4]

pred_for_4_prev = predictions_4[:min_length_4]


# Re-run the detailed KS calculation steps for each model

ks_statistic_2_prev, ks_detail_2_prev = calculate_ks_details(actual_for_2_prev, pred_for_2_prev)

ks_statistic_3_prev, ks_detail_3_prev = calculate_ks_details(actual_for_3_prev, pred_for_3_prev)

ks_statistic_4_prev, ks_detail_4_prev = calculate_ks_details(actual_for_4_prev, pred_for_4_prev)


# Display detailed calculations for each model

tools.display_dataframe_to_user(name="KS Test Details - 2 Previous Failures Model (Adjusted)", dataframe=ks_detail_2_prev)

tools.display_dataframe_to_user(name="KS Test Details - 3 Previous Failures Model (Adjusted)", dataframe=ks_detail_3_prev)

tools.display_dataframe_to_user(name="KS Test Details - 4 Previous Failures Model (Adjusted)", dataframe=ks_detail_4_prev)


# Display KS statistics for each model

ks_statistic_2_prev, ks_statistic_3_prev, ks_statistic_4_prev
```