In [ ]:
```
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
from tkinter import filedialog
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import pandas as pd
from sklearn.ensemble import IsolationForest
import numpy as np

def main():
    # Create the main window
    root = tk.Tk()

    # Create a label and entry widget for the file path
    file_label = ttk.Label(root, text="Enter the dyn.out file path:")
    file_label.pack(padx=10, pady=10, fill=tk.X)
    file_entry = ttk.Entry(root, width=50)
    file_entry.pack(padx=10, pady=10, fill=tk.X)

    # Create a button to open the file dialog
    open_button = ttk.Button(root, text="Browse...", command=lambda: open_file_dialog(file_entry))
    open_button.pack(padx=10, pady=10, fill=tk.X)

    # Create a button to start processing the data
    process_button = ttk.Button(root, text="Process Data", command=lambda: process_data(file_entry.get()))
    process_button.pack(padx=10, pady=10, fill=tk.X)

    # Start the main loop
    root.mainloop()

def open_file_dialog(entry_widget):
    file_path = filedialog.askopenfilename(filetypes=[("Text Files", "*.txt")])
    entry_widget.insert(0, file_path)

def process_data(file_path):
    # Read the data from the file
    data = pd.read_csv(file_path, sep=" ", header=None)

    # Extract the time column
    time = data.iloc[:, 0]
```

```
    # Extract the remaining columns as features
    features = data.iloc[:, 1:]

    # Perform anomaly detection using Isolation Forest
    model = IsolationForest(contamination=0.05)
    model.fit(features)
    anomalies = model.predict(features)

    # Plot the selected features with anomaly detection
    selected_features = ['V1', 'V2', 'V3']  # Set the features to plot here
    plot_selected_features(time, features, selected_features, anomalies)

def plot_selected_features(time, features, selected_features, anomalies):
    # Create a new window for the plot
    plot_window = tk.Toplevel()
    plot_window.title("Feature Plot")

    # Create a figure and canvas for the plot
    fig = Figure(figsize=(10, 5))
    canvas = FigureCanvasTkAgg(fig, master=plot_window)
    canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)

    # Plot the selected features
    for feature in selected_features:
        ax = fig.add_subplot(1, len(selected_features), selected_features.index(feature) + 1)
        ax.set_title(feature)
        ax.plot(time, features[feature], label="Normal")

        # Plot the anomalies
        anomaly_indices = np.where(anomalies == -1)[0]
        ax.scatter(time.iloc[anomaly_indices], features.iloc[anomaly_indices, selected_features.index(feature)], color

        ax.legend()

if __name__ == "__main__":
    main()
```

In [ ]:

```
import tkinter as tk
from tkinter import filedialog
import pandas as pd
import numpy as np
from sklearn.ensemble import IsolationForest
import matplotlib.pyplot as plt

def open_file_dialog():
    file_path = filedialog.askopenfilename(filetypes=[("Text Files", "*.txt")], title="Select dyn.out files")
    return file_path

def load_data(file_paths):
    data_list = []
    for file_path in file_paths:
        data = pd.read_csv(file_path, sep=" ", header=None)
        data_list.append(data)
    return pd.concat(data_list, ignore_index=True)

def select_columns():
    global column_names
    column_names = column_listbox.get(0, tk.END)

def detect_anomalies(data, column_names):
    clustering = IsolationForest(random_state=0).fit(data[column_names])
    labels = clustering.predict(data[column_names])
    return labels

def plot_anomalies(data, column_names, labels):
    colors = ['blue' if label == 1 else 'red' for label in labels]
    plt.figure(figsize=(10, 5))
    for column in column_names:
        plt.plot(data[column], color=colors, alpha=0.5)
    plt.title("Anomalies in Selected Columns")
    plt.xlabel("Index")
    plt.ylabel("Value")
    plt.legend(column_names)
    plt.show()

if __name__ == "__main__":
    file_paths = [open_file_dialog() for _ in range(2)]  # Select 2 dyn.out files
    data = load_data(file_paths)
```

```
column_window = tk.Toplevel()
column_window.title("Select Columns")

column_listbox = tk.Listbox(column_window)
column_listbox.pack(side="left", fill="both", expand=True)

scrollbar = tk.Scrollbar(column_window)
scrollbar.pack(side="right", fill="y")

column_listbox.config(yscrollcommand=scrollbar.set)
scrollbar.config(command=column_listbox.yview)

for column in data.columns:
    column_listbox.insert(tk.END, column)

tk.Button(column_window, text="Select Columns", command=select_columns).pack(side="bottom")

column_names = []
labels = detect_anomalies(data, column_names)
plot_anomalies(data, column_names, labels)

column_window.mainloop()
```

In [ ]:

```
import tkinter as tk
from tkinter import ttk, filedialog, messagebox
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import pandas as pd

# Function to load data from provided file paths
def load_data(file_paths):
    try:
        dfs = []
        for file_path in file_paths:
            df = pd.read_csv('Book1.csv', delim_whitespace=True)
            dfs.append(df)
        return pd.concat(dfs)
    except Exception as e:
        messagebox.showerror("Error", f"Error loading data: {e}")
        return None

# Function to plot selected parameter across all files
def plot_parameter_across_files():
    # Get selected parameter from the combobox
    selected_param = parameter_combobox.get()

    # Check if a parameter is selected
    if not selected_param:
        messagebox.showerror("Error", "Please select a parameter to plot.")
        return

    # Clear previous plots
    plot_frame.clear()

    # Load data
    data = load_data(file_paths)
    if data is None:
        return

    # Create figure and subplot
    fig = Figure(figsize=(8, 5))
    ax = fig.add_subplot(111)

    # Plot data for each file
```

```
    for file_path in file_paths:
        file_data = pd.read_csv(file_path, delim_whitespace=True)
        ax.plot(file_data['Timestamp'], file_data[selected_param], label=file_path)

    ax.set_title(f'Parameter: {selected_param}')
    ax.set_xlabel('Timestamp')
    ax.set_ylabel(selected_param)
    ax.legend()

    # Embed the plot in the Tkinter window
    canvas = FigureCanvasTkAgg(fig, master=plot_frame)
    canvas.draw()
    canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)

# Create Tkinter window
root = tk.Tk()
root.title("Plot Parameters Across Files")

# Function to select files
def select_files():
    files = filedialog.askopenfilenames(title="Select Files", filetypes=[("Text files", "*.txt")])
    if files:
        global file_paths
        file_paths = files

# Create parameter selection frame
param_frame = ttk.Frame(root)
param_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)

label = ttk.Label(param_frame, text="Select Parameter:")
label.pack(side=tk.TOP, padx=10, pady=5)

parameter_combobox = ttk.Combobox(param_frame, state="readonly")
parameter_combobox.pack(side=tk.TOP, padx=10, pady=5)

# Create plot button
plot_button = ttk.Button(param_frame, text="Plot Parameter Across Files", command=plot_parameter_across_files)
plot_button.pack(side=tk.TOP, padx=10, pady=5)

# Create plot frame
plot_frame = ttk.Frame(root)
plot_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)
```

```
plot_frame.clear = lambda: plot_frame.winfo_children()  # Clear plot frame

# Button to select files
select_files_button = ttk.Button(root, text="Select Files", command=select_files)
select_files_button.pack(padx=10, pady=5)

# List to store file paths
file_paths = []

# Start the Tkinter event loop
root.mainloop()
```

In [ ]:

```
import tkinter as tk
from tkinter import ttk, filedialog, messagebox
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import pandas as pd
from sklearn.ensemble import IsolationForest
import numpy as np

# Function to load data from provided file paths
def load_data(file_paths):
    try:
        dfs = []
        for file_path in file_paths:
            df = pd.read_csv('file_path', delim_whitespace=True)
            dfs.append(df)
        return pd.concat(dfs)
    except Exception as e:
        messagebox.showerror("Error", f"Error loading data: {e}")
        return None

# Function to plot selected parameter across all dyn.out files
def plot_selected_parameter():
    # Get selected parameter from the combobox
    selected_param = parameter_combobox.get()

    # Check if a parameter is selected
    if not selected_param:
        messagebox.showerror("Error", "Please select a parameter to plot.")
        return

    # Clear previous plots
    plot_frame.clear()

    # Load data
    data = load_data(file_paths)
    if data is None:
        return

    fig = Figure(figsize=(8, 5))
    ax = fig.add_subplot(111)
```

```
    # Plot data for selected parameter across all test cases
    for test_case in data['Test_Case'].unique():
        test_case_data = data[data['Test_Case'] == test_case]
        ax.plot(test_case_data['Timestamp'], test_case_data[selected_param], label=f'Test Case {test_case}')

    ax.set_title(f'Parameter: {selected_param}')
    ax.set_xlabel('Timestamp')
    ax.set_ylabel(selected_param)
    ax.legend()

    # Embed the plot in the Tkinter window
    canvas = FigureCanvasTkAgg(fig, master=plot_frame)
    canvas.draw()
    canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)

# Create Tkinter window
root = tk.Tk()
root.title("Parameter Plotter across dyn.out Files")

# Function to select files
def select_files():
    files = filedialog.askopenfilenames(title="Select Files", filetypes=[("Text files", "*.txt")])
    if files:
        global file_paths
        file_paths = files

# Create frame for file selection
file_frame = ttk.Frame(root)
file_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)

select_files_button = ttk.Button(file_frame, text="Select dyn.out Files", command=select_files)
select_files_button.pack(side=tk.TOP, padx=10, pady=5)

# Create parameter selection frame
param_frame = ttk.Frame(root)
param_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)

label = ttk.Label(param_frame, text="Select Parameter to Plot:")
label.pack(side=tk.TOP, padx=10, pady=5)

parameter_combobox = ttk.Combobox(param_frame, values=[], state="readonly")
parameter_combobox.pack(side=tk.TOP, padx=10, pady=5)
```

```
# Create plot button
plot_button = ttk.Button(param_frame, text="Plot Selected Parameter", command=plot_selected_parameter)
plot_button.pack(side=tk.TOP, padx=10, pady=5)

# Create plot frame
plot_frame = ttk.Frame(root)
plot_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)
plot_frame.clear = lambda: plot_frame.winfo_children()  # Clear plot frame

# List to store file paths
file_paths = []

# Start the Tkinter event loop
root.mainloop()
```

In [ ]:

```
import tkinter as tk
from tkinter import ttk, messagebox
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import pandas as pd
import os

# File paths for dyn.out files (modify as needed)
file_paths = [r"C:\Users\DEEPAK PRASAD\OneDrive\Desktop\Book1.csv"]


# Function to load data from provided file paths
def load_data(file_paths):
    try:
        dfs = []
        for file_path in file_paths:
            df = pd.read_csv(file_path, delim_whitespace=True)
            dfs.append(df)
        return pd.concat(dfs)
    except Exception as e:
        messagebox.showerror("Error", f"Error loading data: {e}")
        return None

# Function to plot selected parameter across all dyn.out files
def plot_selected_parameter():
    # Get selected parameter from the combobox
    selected_param = parameter_combobox.get()

    # Check if a parameter is selected
    if not selected_param:
        messagebox.showerror("Error", "Please select a parameter to plot.")
        return

    # Clear previous plots
    plot_frame.clear()

    # Load data
    data = load_data(file_paths)
    if data is None:
        return
```

```
    fig = Figure(figsize=(10, 6))
    ax = fig.add_subplot(111)

    # Plot data for selected parameter across all dyn.out files
    for file_path in file_paths:
        file_name = os.path.basename(file_path)
        file_data = pd.read_csv(file_path, delim_whitespace=True)
        ax.plot(file_data.iloc[:, 0], file_data[selected_param], label=file_name)

    ax.set_title(f'Parameter: {selected_param}')
    ax.set_xlabel('Time')
    ax.set_ylabel(selected_param)
    ax.legend()

    # Embed the plot in the Tkinter window
    canvas = FigureCanvasTkAgg(fig, master=plot_frame)
    canvas.draw()
    canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)

# Create Tkinter window
root = tk.Tk()
root.title("Parameter Plotter across dyn.out Files")

# Create frame for parameter selection
param_frame = ttk.Frame(root)
param_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)

label = ttk.Label(param_frame, text="Select Parameter to Plot:")
label.pack(side=tk.TOP, padx=10, pady=5)

# Get all parameters from the first dyn.out file
first_file_data = pd.read_csv(file_paths[0], delim_whitespace=True)
parameters = list(first_file_data.columns)[1:]  # Exclude the first column (time)

parameter_combobox = ttk.Combobox(param_frame, values=parameters, state="readonly")
parameter_combobox.pack(side=tk.TOP, padx=10, pady=5)

# Create plot button
plot_button = ttk.Button(param_frame, text="Plot Selected Parameter", command=plot_selected_parameter)
plot_button.pack(side=tk.TOP, padx=10, pady=5)

# Create plot frame
```

```
plot_frame = ttk.Frame(root)
plot_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)
plot_frame.clear = lambda: plot_frame.winfo_children()  # Clear plot frame

# Start the Tkinter event loop
root.mainloop()
```

In [ ]:

```
import tkinter as tk
from tkinter import ttk, messagebox, filedialog
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import pandas as pd

# Function to load data from a CSV file
def load_data():
    file_path = filedialog.askopenfilename(filetypes=[("CSV files", "*.csv")])
    if not file_path:
        return None

    try:
        return pd.read_csv(file_path)
    except Exception as e:
        messagebox.showerror("Error", f"Error loading data: {e}")
        return None

# Function to plot selected parameter
def plot_selected_parameter():
    # Get selected parameter from the combobox
    selected_param = parameter_combobox.get()

    # Check if a parameter is selected
    if not selected_param:
        messagebox.showerror("Error", "Please select a parameter to plot.")
        return

    # Clear previous plots
    plot_frame.clear()

    # Load data
    data = load_data()
    if data is None:
        return

    fig = Figure(figsize=(10, 6))
    ax = fig.add_subplot(111)

    # Plot selected parameter against time
    ax.plot(data['Time'], data[selected_param])
```

```
    ax.set_title(f'Parameter: {selected_param}')
    ax.set_xlabel('Time')
    ax.set_ylabel(selected_param)

    # Embed the plot in the Tkinter window
    canvas = FigureCanvasTkAgg(fig, master=plot_frame)
    canvas.draw()
    canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)

# Create Tkinter window
root = tk.Tk()
root.title("Parameter Plotter")

# Create frame for parameter selection
param_frame = ttk.Frame(root)
param_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)

label = ttk.Label(param_frame, text="Select Parameter to Plot:")
label.pack(side=tk.TOP, padx=10, pady=5)

# Create a Combobox for parameter selection
parameter_combobox = ttk.Combobox(param_frame, state="readonly")
parameter_combobox.pack(side=tk.TOP, padx=10, pady=5)

# Create plot button
plot_button = ttk.Button(param_frame, text="Plot Selected Parameter", command=plot_selected_parameter)
plot_button.pack(side=tk.TOP, padx=10, pady=5)

# Create plot frame
plot_frame = ttk.Frame(root)
plot_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)
plot_frame.clear = lambda: plot_frame.winfo_children()  # Clear plot frame

# Start the Tkinter event loop
root.mainloop()
```

In [ ]:

```
import tkinter as tk
from tkinter import ttk, messagebox
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import pandas as pd
import os

# File paths for dyn.out files (modify as needed)
file_paths = [r"C:\Users\DEEPAK PRASAD\OneDrive\Desktop\dyn.out.txt"]

# Function to load data from provided file paths
def load_data(file_paths):
    try:
        dfs = []
        for file_path in file_paths:
            df = pd.read_csv(file_path, delim_whitespace=True)
            dfs.append(df)
        return pd.concat(dfs)
    except Exception as e:
        messagebox.showerror("Error", f"Error loading data: {e}")
        return None

# Function to plot selected parameter across all dyn.out files
def plot_selected_parameter():
    # Get selected parameter from the combobox
    selected_param = parameter_combobox.get()

    # Get column number to plot
    try:
        column_number = int(column_entry.get()) - 1  # Convert to zero-based index
    except ValueError:
        messagebox.showerror("Error", "Please enter a valid column number.")
        return

    # Check if a parameter is selected
    if not selected_param:
        messagebox.showerror("Error", "Please select a parameter to plot.")
        return

    # Clear previous plots
    plot_frame.clear()
```

```
    # Load data
    data = load_data(file_paths)
    if data is None:
        return

    fig = Figure(figsize=(12, 8))
    ax = fig.add_subplot(111)

    # Plot data for selected parameter across all dyn.out files
    for file_path in file_paths:
        file_name = os.path.basename(file_path)
        file_data = pd.read_csv(file_path, delim_whitespace=True)
        ax.plot(file_data.iloc[:, 0], file_data.iloc[:, column_number], label=f"{file_name} - Column {column_number+1}

    ax.set_title(f'Parameter: {selected_param}, Column Number: {column_number+1}')
    ax.set_xlabel('Time')
    ax.set_ylabel('Value')
    ax.legend()

    # Embed the plot in the Tkinter window
    canvas = FigureCanvasTkAgg(fig, master=plot_frame)
    canvas.draw()
    canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)

# Create Tkinter window
root = tk.Tk()
root.title("Parameter Plotter across dyn.out Files")

# Create frame for parameter selection
param_frame = ttk.Frame(root)
param_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)

label_param = ttk.Label(param_frame, text="Select Parameter to Plot:")
label_param.pack(side=tk.TOP, padx=10, pady=5)

# Get all parameters from the first dyn.out file
first_file_data = pd.read_csv(file_paths[0], delim_whitespace=True)
parameters = list(first_file_data.columns)[1:]  # Exclude the first column (time)

parameter_combobox = ttk.Combobox(param_frame, values=parameters, state="readonly", width=40)
parameter_combobox.pack(side=tk.TOP, padx=10, pady=5)
```

```
label_column = ttk.Label(param_frame, text="Enter Column Number to Plot:")
label_column.pack(side=tk.TOP, padx=10, pady=5)

column_entry = ttk.Entry(param_frame, width=40)
column_entry.pack(side=tk.TOP, padx=10, pady=5)

# Create plot button
plot_button = ttk.Button(param_frame, text="Plot Selected Parameter", command=plot_selected_parameter)
plot_button.pack(side=tk.TOP, padx=10, pady=5)

# Create plot frame
plot_frame = ttk.Frame(root)
plot_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)
plot_frame.clear = lambda: plot_frame.winfo_children()  # Clear plot frame

# Start the Tkinter event loop
root.mainloop()
```

In [ ]:

```
import tkinter as tk
from tkinter import ttk, messagebox, filedialog
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import pandas as pd
import os

# File paths for dyn.out files (modify as needed)
file_paths = [r"C:\Users\DEEPAK PRASAD\OneDrive\Desktop\dyn.out.txt"]

# Function to load data from provided file paths
def load_data(file_paths):
    try:
        dfs = []
        for file_path in file_paths:
            df = pd.read_csv(file_path, delim_whitespace=True)
            dfs.append(df)
        return pd.concat(dfs)
    except Exception as e:
        messagebox.showerror("Error", f"Error loading data: {e}")
        return None

# Function to plot selected parameter across all dyn.out files
def plot_selected_parameter():
    # Get selected parameter from the combobox
    selected_param = parameter_combobox.get()

    # Get column number to plot
    try:
        column_number = int(column_entry.get()) - 1  # Convert to zero-based index
    except ValueError:
        messagebox.showerror("Error", "Please enter a valid column number.")
        return

    # Check if a parameter is selected
    if not selected_param:
        messagebox.showerror("Error", "Please select a parameter to plot.")
        return

    # Clear previous plots
    plot_frame.clear()
```

```
    # Load data
    data = load_data(file_paths)
    if data is None:
        return

    fig = Figure(figsize=(12, 8))
    ax = fig.add_subplot(111)

    # Plot data for selected parameter across all dyn.out files
    for file_path in file_paths:
        file_name = os.path.basename(file_path)
        file_data = pd.read_csv(file_path, delim_whitespace=True)
        ax.plot(file_data.iloc[:, 0], file_data.iloc[:, column_number], label=f"{file_name} - Column {column_number+1}

    ax.set_title(f'Parameter: {selected_param}, Column Number: {column_number+1}')
    ax.set_xlabel('Time')
    ax.set_ylabel('Value')
    ax.legend()

    # Embed the plot in the Tkinter window
    canvas = FigureCanvasTkAgg(fig, master=plot_frame)
    canvas.draw()
    canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)

# Create Tkinter window
root = tk.Tk()
root.title("Parameter Plotter across dyn.out Files")

# Create frame for parameter selection
param_frame = ttk.Frame(root)
param_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)

label_param = ttk.Label(param_frame, text="Select Parameter to Plot:")
label_param.pack(side=tk.TOP, padx=10, pady=5)

# Get all parameters from the first dyn.out file
first_file_data = pd.read_csv(file_paths[0], delim_whitespace=True)
parameters = list(first_file_data.columns)[1:]  # Exclude the first column (time)

parameter_combobox = ttk.Combobox(param_frame, values=parameters, state="readonly", width=40)
parameter_combobox.pack(side=tk.TOP, padx=10, pady=5)
```

```
label_column = ttk.Label(param_frame, text="Enter Column Number to Plot:")
label_column.pack(side=tk.TOP, padx=10, pady=5)

column_entry = ttk.Entry(param_frame, width=40)
column_entry.pack(side=tk.TOP, padx=10, pady=5)

# Create plot button
plot_button = ttk.Button(param_frame, text="Plot Selected Parameter", command=plot_selected_parameter)
plot_button.pack(side=tk.TOP, padx=10, pady=5)

# Create plot frame
plot_frame = ttk.Frame(root)
plot_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)
plot_frame.clear = lambda: plot_frame.winfo_children()  # Clear plot frame

# Start the Tkinter event loop
root.mainloop()
```

In [ ]:

```
import tkinter as tk
from tkinter import ttk, messagebox
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import pandas as pd

# File path for the CSV file (modify as needed)
csv_file_path = "C:/Users/DEEPAK PRASAD/OneDrive/Desktop/dyn.out.txt"

# Function to load data from the specified CSV file
def load_data():
    try:
        return pd.read_csv(csv_file_path)
    except Exception as e:
        messagebox.showerror("Error", f"Error loading data: {e}")
        return None

# Function to plot selected parameter
def plot_selected_parameter():
    # Get selected parameter from the combobox
    selected_param = parameter_combobox.get()

    # Get column number to plot
    try:
        column_number = int(column_entry.get()) - 1  # Convert to zero-based index
    except ValueError:
        messagebox.showerror("Error", "Please enter a valid column number.")
        return

    # Check if a parameter is selected
    if not selected_param:
        messagebox.showerror("Error", "Please select a parameter to plot.")
        return

    # Clear previous plots
    plot_frame.clear()

    # Load data
    data = load_data()
    if data is None:
        return
```

```
    fig = Figure(figsize=(12, 8))
    ax = fig.add_subplot(111)

    # Plot data for selected parameter
    ax.plot(data.iloc[:, 0], data.iloc[:, column_number])

    ax.set_title(f'Parameter: {selected_param}, Column Number: {column_number+1}')
    ax.set_xlabel('Time')
    ax.set_ylabel('Value')

    # Embed the plot in the Tkinter window
    canvas = FigureCanvasTkAgg(fig, master=plot_frame)
    canvas.draw()
    canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)

# Create Tkinter window
root = tk.Tk()
root.title("Parameter Plotter")

# Create frame for parameter selection
param_frame = ttk.Frame(root)
param_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)

label_param = ttk.Label(param_frame, text="Select Parameter to Plot:")
label_param.pack(side=tk.TOP, padx=10, pady=5)

# Load data from the specified CSV file
data = load_data()
if data is not None:
    parameters = list(data.columns)
else:
    parameters = []

parameter_combobox = ttk.Combobox(param_frame, values=parameters, state="readonly", width=40)
parameter_combobox.pack(side=tk.TOP, padx=10, pady=5)

label_column = ttk.Label(param_frame, text="Enter Column Number to Plot:")
label_column.pack(side=tk.TOP, padx=10, pady=5)

column_entry = ttk.Entry(param_frame, width=40)
column_entry.pack(side=tk.TOP, padx=10, pady=5)
```

```
# Create plot button
plot_button = ttk.Button(param_frame, text="Plot Selected Parameter", command=plot_selected_parameter)
plot_button.pack(side=tk.TOP, padx=10, pady=5)

# Create plot frame
plot_frame = ttk.Frame(root)
plot_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)
plot_frame.clear = lambda: plot_frame.winfo_children()  # Clear plot frame

# Start the Tkinter event loop
root.mainloop()
```

In [ ]:

```
import tkinter as tk
from tkinter import ttk, messagebox
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import pandas as pd

# File path for the CSV file (modify as needed)
csv_file_path = "C:/Users/DEEPAK PRASAD/OneDrive/Desktop/dyn.out.txt"

# Function to load data from the specified CSV file
def load_data():
    try:
        return pd.read_csv(csv_file_path)
    except Exception as e:
        messagebox.showerror("Error", f"Error loading data: {e}")
        return None

# Function to plot selected parameter
def plot_selected_parameter():
    # Get selected parameter from the combobox
    selected_param = parameter_combobox.get()

    # Check if a parameter is selected
    if not selected_param:
        messagebox.showerror("Error", "Please select a parameter to plot.")
        return

    # Clear previous plots
    plot_frame.clear()

    # Load data
    data = load_data()
    if data is None:
        return

    fig = Figure(figsize=(12, 8))
    ax = fig.add_subplot(111)

    # Plot data for selected parameter
    ax.plot(data.iloc[:, 0], data[selected_param])
```

```
    ax.set_title(f'Parameter: {selected_param}')
    ax.set_xlabel('Time')
    ax.set_ylabel('Value')

    # Embed the plot in the Tkinter window
    canvas = FigureCanvasTkAgg(fig, master=plot_frame)
    canvas.draw()
    canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)

# Create Tkinter window
root = tk.Tk()
root.title("Parameter Plotter")

# Create frame for parameter selection
param_frame = ttk.Frame(root)
param_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)

label_param = ttk.Label(param_frame, text="Select Parameter to Plot:")
label_param.pack(side=tk.TOP, padx=10, pady=5)

# Load data from the specified CSV file
data = load_data()
if data is not None:
    parameters = list(data.columns)
else:
    parameters = []

parameter_combobox = ttk.Combobox(param_frame, values=parameters, state="readonly", width=40)
parameter_combobox.pack(side=tk.TOP, padx=10, pady=5)

# Create plot button
plot_button = ttk.Button(param_frame, text="Plot Selected Parameter", command=plot_selected_parameter)
plot_button.pack(side=tk.TOP, padx=10, pady=5)

# Create plot frame
plot_frame = ttk.Frame(root)
plot_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)
plot_frame.clear = lambda: plot_frame.winfo_children()  # Clear plot frame

# Start the Tkinter event loop
root.mainloop()
```

In [ ]:

```
import tkinter as tk
from tkinter import ttk, filedialog, messagebox
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import pandas as pd

# Function to load data from the specified CSV file
def load_data(file_path):
    try:
        return pd.read_csv("C:/Users/DEEPAK PRASAD/OneDrive/Desktop/dyn.out.txt")
    except Exception as e:
        messagebox.showerror("Error", f"Error loading data: {e}")
        return None

# Function to plot selected column
def plot_selected_column():
    # Get selected column from the combobox
    selected_column = column_combobox.get()

    # Check if a column is selected
    if not selected_column:
        messagebox.showerror("Error", "Please select a column to plot.")
        return

    # Clear previous plots
    plot_frame.clear()

    # Load data
    data = load_data(csv_file_path)
    if data is None:
        return

    fig = Figure(figsize=(12, 8))
    ax = fig.add_subplot(111)

    # Plot data for selected column
    ax.plot(data.index, data[selected_column])

    ax.set_title(f'Column: {selected_column}')
    ax.set_xlabel('Index')
    ax.set_ylabel('Value')
```

```
    # Embed the plot in the Tkinter window
    canvas = FigureCanvasTkAgg(fig, master=plot_frame)
    canvas.draw()
    canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)

# Create Tkinter window
root = tk.Tk()
root.title("Column Plotter")

# Specify the path to the CSV file (modify as needed)
csv_file_path = "path/to/your/data.csv"

# Load data from the specified CSV file
data = load_data(csv_file_path)
if data is not None:
    columns = list(data.columns)
else:
    columns = []

# Create frame for column selection
column_frame = ttk.Frame(root)
column_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)

label_column = ttk.Label(column_frame, text="Select Column to Plot:")
label_column.pack(side=tk.TOP, padx=10, pady=5)

column_combobox = ttk.Combobox(column_frame, values=columns, state="readonly", width=40)
column_combobox.pack(side=tk.TOP, padx=10, pady=5)

# Create plot button
plot_button = ttk.Button(column_frame, text="Plot Selected Column", command=plot_selected_column)
plot_button.pack(side=tk.TOP, padx=10, pady=5)

# Create plot frame
plot_frame = ttk.Frame(root)
plot_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)
plot_frame.clear = lambda: plot_frame.winfo_children()  # Clear plot frame

# Start the Tkinter event loop
root.mainloop()
```

In [ ]:

```
import tkinter as tk
from tkinter import ttk, messagebox
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import pandas as pd

# Specify the path to the Excel file (modify as needed)
excel_file_path = "D:/Book12.xlsx"

# Function to load data from the specified Excel file
def load_data():
    try:
        return pd.read_excel(excel_file_path)
    except Exception as e:
        messagebox.showerror("Error", f"Error loading data: {e}")
        return None

# Function to plot selected column
def plot_selected_column():
    # Get selected column from the combobox
    selected_column = column_combobox.get()

    # Check if a column is selected
    if not selected_column:
        messagebox.showerror("Error", "Please select a column to plot.")
        return

    # Clear previous plots
    plot_frame.clear()

    # Load data
    data = load_data()
    if data is None:
        return

    fig = Figure(figsize=(12, 8))
    ax = fig.add_subplot(111)

    # Plot data for selected column
    ax.plot(data[selected_column])
```

```
    ax.set_title(f'Column: {selected_column}')
    ax.set_xlabel('Index')
    ax.set_ylabel('Value')

    # Embed the plot in the Tkinter window
    canvas = FigureCanvasTkAgg(fig, master=plot_frame)
    canvas.draw()
    canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)

# Create Tkinter window
root = tk.Tk()
root.title("Column Plotter")

# Load data from the specified Excel file
data = load_data()
if data is not None:
    columns = list(data.columns)
else:
    columns = []

# Create frame for column selection
column_frame = ttk.Frame(root)
column_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)

label_column = ttk.Label(column_frame, text="Select Column to Plot:")
label_column.pack(side=tk.TOP, padx=10, pady=5)

column_combobox = ttk.Combobox(column_frame, values=columns, state="readonly", width=40)
column_combobox.pack(side=tk.TOP, padx=10, pady=5)

# Create plot button
plot_button = ttk.Button(column_frame, text="Plot Selected Column", command=plot_selected_column)
plot_button.pack(side=tk.TOP, padx=10, pady=5)

# Create plot frame
plot_frame = ttk.Frame(root)
plot_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)
plot_frame.clear = lambda: plot_frame.winfo_children()  # Clear plot frame

# Start the Tkinter event loop
root.mainloop()
```

In [ ]:

```
import tkinter as tk
from tkinter import ttk, messagebox
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import pandas as pd

# Specify the path to the Excel file (modify as needed)
excel_file_path = "D:/Book12.xlsx"

# Function to load data from the specified Excel file
def load_data():
    try:
        return pd.read_excel(excel_file_path)
    except Exception as e:
        messagebox.showerror("Error", f"Error loading data: {e}")
        return None

# Function to plot selected column
def plot_selected_column():
    # Get selected column from the combobox
    selected_column = column_combobox.get()

    # Check if a column is selected
    if not selected_column:
        messagebox.showerror("Error", "Please select a column to plot.")
        return

    # Clear previous plots
    plot_frame.clear()

    # Load data
    data = load_data()
    if data is None:
        return

    fig = Figure(figsize=(12, 8))
    ax = fig.add_subplot(111)

    # Plot data for selected column against the first column (time values)
    ax.plot(data.iloc[:, 0], data[selected_column])
```

```
    ax.set_title(f'Column: {selected_column}')
    ax.set_xlabel('Time')
    ax.set_ylabel('Value')

    # Embed the plot in the Tkinter window
    canvas = FigureCanvasTkAgg(fig, master=plot_frame)
    canvas.draw()
    canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)

# Create Tkinter window
root = tk.Tk()
root.title("Column Plotter")

# Load data from the specified Excel file
data = load_data()
if data is not None:
    columns = list(data.columns)
else:
    columns = []

# Create frame for column selection
column_frame = ttk.Frame(root)
column_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)

label_column = ttk.Label(column_frame, text="Select Column to Plot:")
label_column.pack(side=tk.TOP, padx=10, pady=5)

column_combobox = ttk.Combobox(column_frame, values=columns, state="readonly", width=40)
column_combobox.pack(side=tk.TOP, padx=10, pady=5)

# Create plot button
plot_button = ttk.Button(column_frame, text="Plot Selected Column", command=plot_selected_column)
plot_button.pack(side=tk.TOP, padx=10, pady=5)

# Create plot frame
plot_frame = ttk.Frame(root)
plot_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)
plot_frame.clear = lambda: plot_frame.winfo_children()  # Clear plot frame

# Start the Tkinter event loop
root.mainloop()
```

In [ ]:

```
import tkinter as tk
from tkinter import ttk, messagebox
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import pandas as pd

# Specify the path to the Excel file (modify as needed)
excel_file_path = "D:/Book12.xlsx"

# Function to load data from the specified Excel file
def load_data():
    try:
        return pd.read_excel(excel_file_path)
    except Exception as e:
        messagebox.showerror("Error", f"Error loading data: {e}")
        return None

# Function to plot selected column
def plot_selected_column():
    # Get selected column from the combobox
    selected_column = column_combobox.get()

    # Check if a column is selected
    if not selected_column:
        messagebox.showerror("Error", "Please select a column to plot.")
        return

    # Clear previous plots
    plot_frame.clear()

    # Load data
    data = load_data()
    if data is None:
        return

    fig = Figure(figsize=(12, 8))
    ax = fig.add_subplot(111)

    # Plot data for selected column against the first column (time values)
    ax.plot(data.iloc[:, 0], data[selected_column], marker='o', linestyle='-')
```

```
    ax.set_title(f'Column: {selected_column}')
    ax.set_xlabel('Time')
    ax.set_ylabel('Value')

    # Embed the plot in the Tkinter window
    canvas = FigureCanvasTkAgg(fig, master=plot_frame)
    canvas.draw()
    canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)

# Create Tkinter window
root = tk.Tk()
root.title("Column Plotter")

# Load data from the specified Excel file
data = load_data()
if data is not None:
    columns = list(data.columns)
else:
    columns = []

# Create frame for column selection
column_frame = ttk.Frame(root)
column_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)

label_column = ttk.Label(column_frame, text="Select Column to Plot:")
label_column.pack(side=tk.TOP, padx=10, pady=5)

column_combobox = ttk.Combobox(column_frame, values=columns, state="readonly", width=40)
column_combobox.pack(side=tk.TOP, padx=10, pady=5)

# Create plot button
plot_button = ttk.Button(column_frame, text="Plot Selected Column", command=plot_selected_column)
plot_button.pack(side=tk.TOP, padx=10, pady=5)

# Create plot frame
plot_frame = ttk.Frame(root)
plot_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)
plot_frame.clear = lambda: plot_frame.winfo_children()  # Clear plot frame

# Start the Tkinter event loop
root.mainloop()
```

In [ ]:

```
import tkinter as tk
from tkinter import ttk, messagebox
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import pandas as pd

# Specify the paths to the Excel files (modify as needed)
excel_file_paths = [
    "D:/Data/Book12.xlsx",
    "D:/Data/Book13.xlsx",
    "D:/Data/Book14.xlsx",
    # Add more paths as needed
]

# Function to load data from multiple Excel files
def load_data():
    try:
        # Load data from each Excel file and concatenate into a single DataFrame
        data_frames = [pd.read_excel(file) for file in excel_file_paths]
        return pd.concat(data_frames)
    except Exception as e:
        messagebox.showerror("Error", f"Error loading data: {e}")
        return None

# Function to plot selected column
def plot_selected_column():
    # Get selected column from the combobox
    selected_column = column_combobox.get()

    # Check if a column is selected
    if not selected_column:
        messagebox.showerror("Error", "Please select a column to plot.")
        return

    # Clear previous plots
    plot_frame.clear()

    # Load data
    data = load_data()
    if data is None:
        return
```

```
    fig = Figure(figsize=(12, 8))
    ax = fig.add_subplot(111)

    # Plot data for selected column against the first column (time values)
    for i in range(len(data)):
        ax.plot(data.iloc[i, 0], data[selected_column].iloc[i], marker='o', linestyle='-')

    ax.set_title(f'Column: {selected_column}')
    ax.set_xlabel('Time')
    ax.set_ylabel('Value')

    # Set x-axis format to datetime if applicable
    if pd.api.types.is_datetime64_any_dtype(data.iloc[:, 0]):
        ax.xaxis_date()

    # Embed the plot in the Tkinter window
    canvas = FigureCanvasTkAgg(fig, master=plot_frame)
    canvas.draw()
    canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)

# Create Tkinter window
root = tk.Tk()
root.title("Column Plotter")

# Load data from multiple Excel files
data = load_data()
if data is not None:
    columns = list(data.columns)
else:
    columns = []

# Create frame for column selection
column_frame = ttk.Frame(root)
column_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)

label_column = ttk.Label(column_frame, text="Select Column to Plot:")
label_column.pack(side=tk.TOP, padx=10, pady=5)

column_combobox = ttk.Combobox(column_frame, values=columns, state="readonly", width=40)
column_combobox.pack(side=tk.TOP, padx=10, pady=5)
```

```
# Create plot button
plot_button = ttk.Button(column_frame, text="Plot Selected Column", command=plot_selected_column)
plot_button.pack(side=tk.TOP, padx=10, pady=5)

# Create plot frame
plot_frame = ttk.Frame(root)
plot_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)
plot_frame.clear = lambda: plot_frame.winfo_children()  # Clear plot frame

# Start the Tkinter event loop
root.mainloop()
```

In [ ]:

```
import tkinter as tk
from tkinter import ttk, messagebox
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import pandas as pd

# Specify the paths to the Excel files (modify as needed)
excel_file_paths = [
    "D:/Data/Book12.xlsx",
    "D:/Data/Book13.xlsx",
    "D:/Data/Book14.xlsx",
    # Add more paths as needed
]

# Function to load data from multiple Excel files
def load_data():
    try:
        # Load data from each Excel file and concatenate into a single DataFrame
        data_frames = [pd.read_excel(file) for file in excel_file_paths]
        return pd.concat(data_frames)
    except Exception as e:
        messagebox.showerror("Error", f"Error loading data: {e}")
        return None

# Function to plot selected column
def plot_selected_column():
    # Get selected column from the combobox
    selected_column = column_combobox.get()

    # Check if a column is selected
    if not selected_column:
        messagebox.showerror("Error", "Please select a column to plot.")
        return

    # Clear previous plots
    plot_frame.clear()

    # Load data
    data = load_data()
    if data is None:
        return
```

```
    fig = Figure(figsize=(12, 8))
    ax = fig.add_subplot(111)

    # Plot data for selected column against the first column (time values)
    for i in range(len(data)):
        ax.plot(data.iloc[i, 0], data[selected_column].iloc[i], marker='o', linestyle='-')

    ax.set_title(f'Column: {selected_column}')
    ax.set_xlabel('Time')
    ax.set_ylabel('Value')

    # Set x-axis format to datetime if applicable
    if pd.api.types.is_datetime64_any_dtype(data.iloc[:, 0]):
        ax.xaxis_date()

    # Embed the plot in the Tkinter window
    canvas = FigureCanvasTkAgg(fig, master=plot_frame)
    canvas.draw()
    canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)

# Create Tkinter window
root = tk.Tk()
root.title("Column Plotter")

# Load data from multiple Excel files
data = load_data()
if data is not None:
    columns = list(data.columns)
else:
    columns = []

# Create frame for column selection
column_frame = ttk.Frame(root)
column_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)

label_column = ttk.Label(column_frame, text="Select Column to Plot:")
label_column.pack(side=tk.TOP, padx=10, pady=5)

column_combobox = ttk.Combobox(column_frame, values=columns, state="readonly", width=40)
column_combobox.pack(side=tk.TOP, padx=10, pady=5)
```

```
# Create plot button
plot_button = ttk.Button(column_frame, text="Plot Selected Column", command=plot_selected_column)
plot_button.pack(side=tk.TOP, padx=10, pady=5)

# Create plot frame
plot_frame = ttk.Frame(root)
plot_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)
plot_frame.clear = lambda: plot_frame.winfo_children()  # Clear plot frame

# Start the Tkinter event loop
root.mainloop()
```

In [1]:
```
import tkinter as tk
from tkinter import ttk, messagebox
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import pandas as pd

# Specify the paths to the Excel files (modify as needed)
excel_file_paths = [
    "D:/Data/Book12.xlsx",
    "D:/Data/Book13.xlsx",
    "D:/Data/Book14.xlsx",
    # Add more paths as needed
]

# Function to load data from multiple Excel files
def load_data():
    try:
        # Load data from each Excel file and concatenate into a single DataFrame
        data_frames = [pd.read_excel(file) for file in excel_file_paths]
        return pd.concat(data_frames)
    except Exception as e:
        messagebox.showerror("Error", f"Error loading data: {e}")
        return None

# Function to plot selected column
def plot_selected_column():
    # Get selected column from the combobox
    selected_column = column_combobox.get()

    # Check if a column is selected
    if not selected_column:
        messagebox.showerror("Error", "Please select a column to plot.")
        return

    # Clear previous plots
    plot_frame.clear()

    # Load data
    data = load_data()
    if data is None:
        return
```

```
    fig = Figure(figsize=(12, 8))
    ax = fig.add_subplot(111)

    # Plot data for selected column across all files
    for file_path in excel_file_paths:
        # Load data from current file
        df = pd.read_excel(file_path)
        # Plot selected column against the first column (time values)
        ax.plot(df.iloc[:, 0], df[selected_column], marker='o', linestyle='-', label=file_path)

    ax.set_title(f'Column: {selected_column}')
    ax.set_xlabel('Time')
    ax.set_ylabel('Value')
    ax.legend()

    # Set x-axis format to datetime if applicable
    if pd.api.types.is_datetime64_any_dtype(data.iloc[:, 0]):
        ax.xaxis_date()

    # Embed the plot in the Tkinter window
    canvas = FigureCanvasTkAgg(fig, master=plot_frame)
    canvas.draw()
    canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)

# Create Tkinter window
root = tk.Tk()
root.title("Column Plotter")

# Load data from multiple Excel files
data = load_data()
if data is not None:
    columns = list(data.columns)
else:
    columns = []

# Create frame for column selection
column_frame = ttk.Frame(root)
column_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)

label_column = ttk.Label(column_frame, text="Select Column to Plot:")
label_column.pack(side=tk.TOP, padx=10, pady=5)
```

```
column_combobox = ttk.Combobox(column_frame, values=columns, state="readonly", width=40)
column_combobox.pack(side=tk.TOP, padx=10, pady=5)

# Create plot button
plot_button = ttk.Button(column_frame, text="Plot Selected Column", command=plot_selected_column)
plot_button.pack(side=tk.TOP, padx=10, pady=5)

# Create plot frame
plot_frame = ttk.Frame(root)
plot_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)
plot_frame.clear = lambda: plot_frame.winfo_children()  # Clear plot frame

# Start the Tkinter event loop
root.mainloop()
```

In [ ]: