# Install the required packages

!pip install catboost

!pip install ipywidgets

!pip install pandas

# Import necessary libraries

import pandas as pd

from catboost import CatBoostRegressor

import ipywidgets as widgets

from IPython.display import display

# Load your dataset from Excel

file\_path = 'PExp.xlsx'  # Make sure to replace with the actual path to your Excel file

train\_data = pd.read\_excel(file\_path, sheet\_name='TR')

# Preprocess data: assuming 'PExp (kN)' is the target variable

X\_train = train\_data.drop(columns=['PExp (kN)'])

y\_train = train\_data['PExp (kN)']

# Initialize and train the CatBoostRegressor model

model = CatBoostRegressor(

    iterations=2000,

    learning\_rate=0.1,

    depth=5,

    loss\_function='RMSE',

    verbose=100

)

model.fit(X\_train, y\_train)

# Define input widgets for each parameter

input\_widgets = {}

for column in X\_train.columns:

    min\_value = X\_train[column].min()

    max\_value = X\_train[column].max()

    input\_widgets[column] = widgets.FloatText(

        value=float(X\_train[column][0]),

        description=f"<b>{column}</b> ({min\_value}-{max\_value})",

        style={'description\_width': 'initial'},

        layout=widgets.Layout(width='70%', height='30px')  # Decreased size

    )

# Label to display result

result\_label = widgets.HTML(value="<b>Output:</b>", layout=widgets.Layout(width='100%', padding="10px"))

# Function to predict PExp (kN) using the CatBoost model

def predict\_cbr(btn):

    # Extract input values from the widgets

    inputs = [widget.value for widget in input\_widgets.values()]

    # Convert inputs to DataFrame

    inputs\_df = pd.DataFrame([inputs], columns=X\_train.columns)

    # Perform prediction using the CatBoost model

    cbr\_prediction = model.predict(inputs\_df)[0]

    # Display the predicted PExp (kN)

    result\_label.value = f"<b>Ultimate Axial Load (predicted):</b> <span style='color:blue; font-size: 24px;'>{cbr\_prediction:.2f} kN</span>"

# Create a Predict button

predict\_button = widgets.Button(

    description="Calculate",

    button\_style='primary',

    style={'button\_color': 'blue'},

    layout=widgets.Layout(width='50%', height='40px')  # Adjusted size to match input fields

)

predict\_button.on\_click(predict\_cbr)

# Creating a display for the input widgets (arranged like the image)

input\_parameters\_box = widgets.GridBox(

    children=[\*list(input\_widgets.values())],

    layout=widgets.Layout(

        grid\_template\_columns="repeat(2, 50%)",

        grid\_gap="20px 20px",

        width='100%'

    )

)

# Create a box for output parameter

output\_box = widgets.VBox([

    widgets.HTML("<h2 style='color:blue; font-size: 22px;'>Output Parameters</h2>"),

    result\_label

])

# Create a container for all elements in the GUI

main\_box = widgets.VBox([

    widgets.HTML("<h1 style='text-align:center; font-size: 28px;'>GUI model for Predicting Ultimate Axial Compressive Load Carrying Capacity of CFST Short Columns</h1>"),

    widgets.HTML("<h3 style='text-align:center; color:red;'>Developed by: Megha Gupta, Satya Prakash and Sufyan Ghani</h3>"),

    widgets.HTML("<hr>"),

    input\_parameters\_box,

    widgets.HTML("<br>"),

    predict\_button,

    output\_box

], layout=widgets.Layout(border='2px solid #ccc', padding='20px', border\_radius='10px', margin='auto', width='60%', box\_shadow='5px 5px 10px #888888'))

# Display the final layout

display(main\_box)