import pandas as pd

from sklearn.ensemble import GradientBoostingRegressor

from sklearn.impute import SimpleImputer

import ipywidgets as widgets

from IPython.display import display

# Read input data from Excel file

df = pd.read\_excel("data3.xlsx")  # Update with your Excel file path

# Separate input features (X) and target variable (y)

X = df.drop(columns=["Probability of Liquefaction Failure"])

y = df["Probability of Liquefaction Failure"]

# Impute missing values in input features with the mean

imputer = SimpleImputer(strategy='mean')

imputer.fit(X)  # Fit the imputer on the data

# Train the GradientBoostingRegressor model with imputed data

X\_imputed = imputer.transform(X)

gbr\_model = GradientBoostingRegressor()

gbr\_model.fit(X\_imputed, y)

# Define input widgets for each parameter

input\_widgets = {}

for i, column in enumerate(X.columns):

    min\_value = df[column].min()

    max\_value = df[column].max()

    input\_widgets[column] = widgets.FloatText(value=float(df[column][0]), description=f"<b>{column}</b> ({min\_value}-{max\_value})", style={'description\_width': 'initial', 'color': 'red'})

# Label to display result

result\_label = widgets.HTML(value="")

# Function to predict liquefaction probability using the GBR model

def predict\_probability(btn):

    # Extract input values from the widgets

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

    # Transform input data using the fitted imputer

    inputs\_imputed = imputer.transform([inputs])

    # Perform prediction using the GBR model

    probability = gbr\_model.predict(inputs\_imputed)[0]

    # Display the predicted probability

    result\_label.value = f"<b>Probability of Liquefaction Failure (predicted by GBR model):</b> <span style='color:blue'>{probability:.2f}</span>"

# Create a Predict button

predict\_button = widgets.Button(description="Predict", button\_style='primary', style={'button\_color': 'blue'})

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

# Attach event listener to each input widget

for widget in input\_widgets.values():

    widget.observe(predict\_probability, names='value')

# Create a box for input parameters

input\_parameters\_box = widgets.VBox([

    widgets.HTML("<h2 style='color:red;'>Input</h2>"),

    \*list(input\_widgets.values()),

    widgets.HTML("<br>"),

    predict\_button

])

# Create a box for output parameter

output\_box = widgets.VBox([

    widgets.HTML("<h2 style='color:blue;'>Output</h2>"),

    result\_label

])

# Arrange input and output boxes horizontally

input\_output\_box = widgets.HBox([input\_parameters\_box, output\_box])

# Style the input and output boxes

input\_parameters\_box.layout.margin = '20px'

input\_parameters\_box.layout.padding = '20px'

output\_box.layout.margin = '20px'

output\_box.layout.padding = '20px'

input\_output\_box.layout.border = '2px solid #ccc'

input\_output\_box.layout.border\_radius = '10px'

input\_output\_box.layout.margin = '50px auto'

input\_output\_box.layout.width = '80%'

input\_output\_box.layout.box\_shadow = '5px 5px 5px #888888'

# Display the GUI

display(widgets.VBox([

    widgets.HTML("<h1 style='text-align:center;'>Probability of Liquefaction Failure</h1>"),

    input\_output\_box

]))