import pandas as pd

from sklearn.impute import SimpleImputer

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Conv1D, Flatten

import ipywidgets as widgets

from IPython.display import display

# Read input data from Excel file

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

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

X = df.drop(columns=["UCS"])

y = df["UCS"]

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Impute missing values in input features with the mean

imputer = SimpleImputer(strategy='mean')

imputer.fit(X\_train)  # Fit the imputer on the training data

X\_train\_imputed = imputer.transform(X\_train)

X\_test\_imputed = imputer.transform(X\_test)

# Standardize features by removing the mean and scaling to unit variance

scaler = StandardScaler()

scaler.fit(X\_train\_imputed)

X\_train\_scaled = scaler.transform(X\_train\_imputed)

X\_test\_scaled = scaler.transform(X\_test\_imputed)

# Define CNN model

model = Sequential([

    Conv1D(filters=64, kernel\_size=3, activation='relu', input\_shape=(X\_train\_scaled.shape[1], 1)),

    Flatten(),

    Dense(64, activation='relu'),

    Dense(1)

])

# Compile the model

model.compile(optimizer='adam', loss='mean\_squared\_error')

# Train the CNN model

model.fit(X\_train\_scaled.reshape(X\_train\_scaled.shape[0], X\_train\_scaled.shape[1], 1), y\_train, epochs=50, batch\_size=32, verbose=0)

# 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 UCS using the CNN model

def predict\_ucs(btn):

    # Extract input values from the widgets

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

    # Transform input data using the fitted imputer and scaler

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

    inputs\_scaled = scaler.transform(inputs\_imputed)

    # Perform prediction using the CNN model

    prediction = model.predict(inputs\_scaled.reshape(1, -1, 1))[0][0]

    # Display the predicted UCS

    result\_label.value = f"<b>Predicted UCS:</b> <span style='color:blue'>{prediction:.2f}</span>"

# Create a Predict button

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

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

# Attach event listener to each input widget

for widget in input\_widgets.values():

    widget.observe(predict\_ucs, 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 = '60%'

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

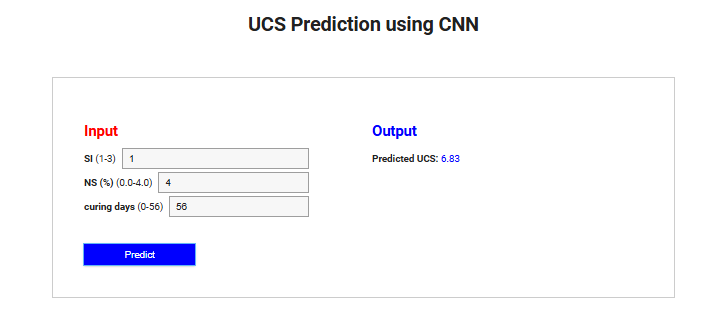
# Display the GUI

display(widgets.VBox([

    widgets.HTML("<h1 style='text-align:center;'> UCS Prediction using CNN</h1>"),

    input\_output\_box

]))



**GUI for prediction of UCS for NS fine-grained soil**