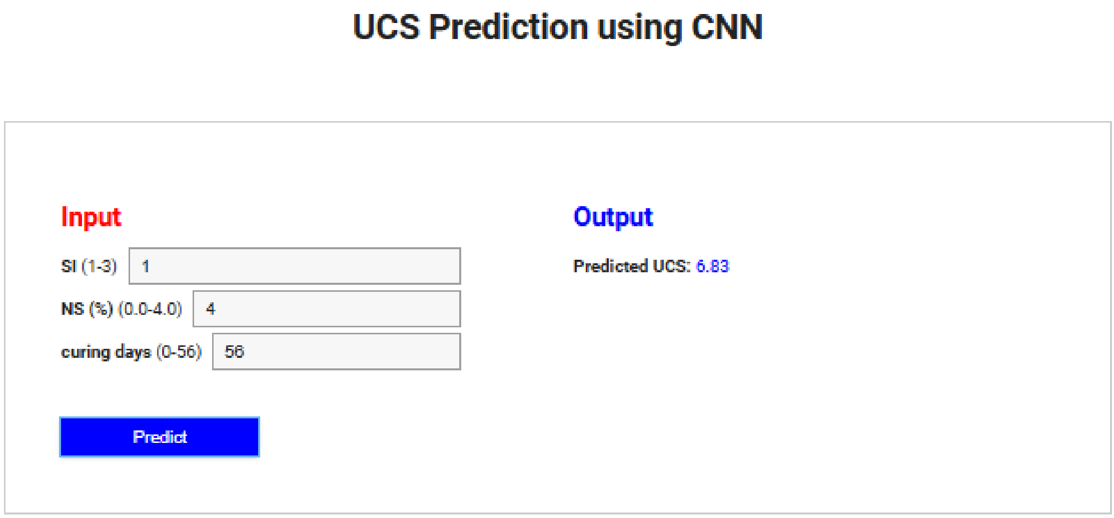
```
import pandas as pd
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv1D, Flatten
import ipywidgets as widgets
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
```

```python
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
]))
```



UCS Prediction using CNN

Input

SI (1-3) 1

NS (%) (0.0-4.0) 4

curing days (0-56) 56

Output

Predicted UCS: 6.83

GUI for prediction of UCS for NS fine-grained soil