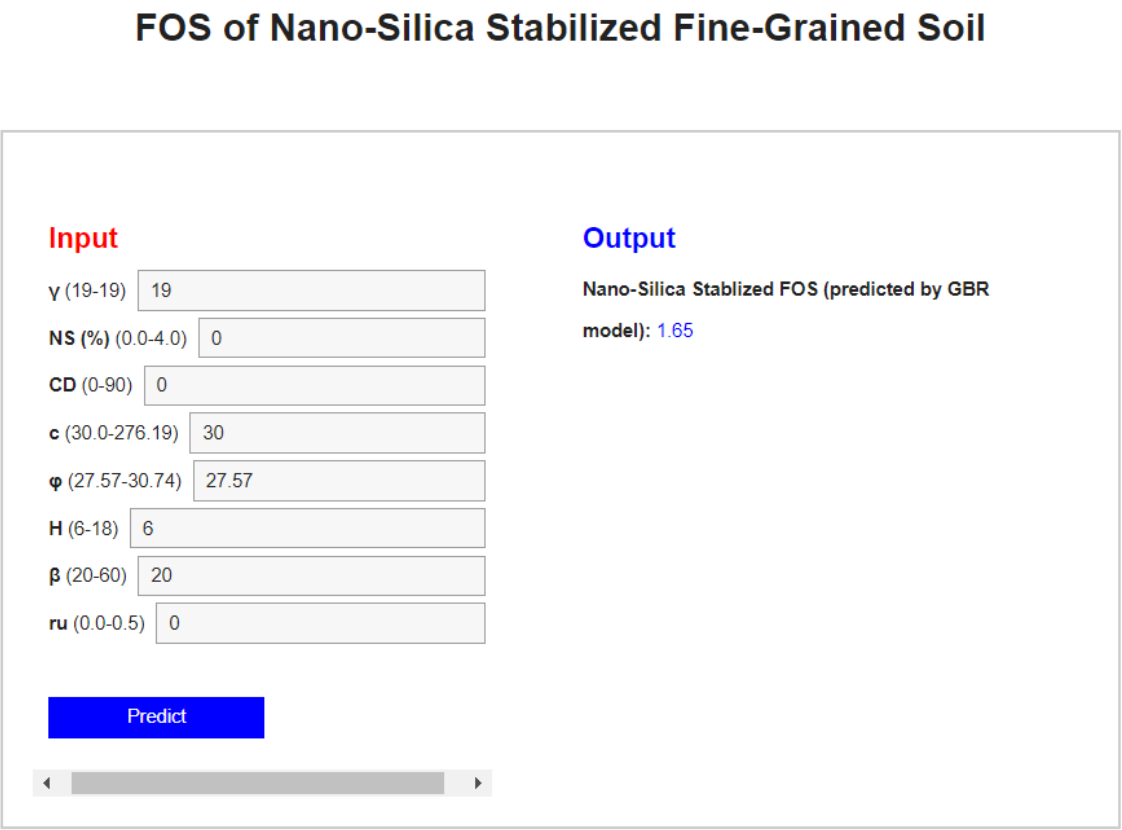
```
import pandas as pd
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.impute import SimpleImputer
import ipywidgets as widgets
from IPython.display import display

# Read input data from Excel file
df = pd.read_excel("IF.xlsx")  # Update with your Excel file path

# Separate input features (X) and target variable (y)
X = df.drop(columns=["FOS"])
y = df["FOS"]

# 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>Nano-Silica Stablized FOS (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 = '60%'
input_output_box.layout.box_shadow = '5px 5px 5px #888888'

# Display the GUI
display(widgets.VBox([
    widgets.HTML("<h1 style='text-align:center;'> FOS of Nano-Silica
Stabilized Fine-Grained Soil</h1>"),
    input_output_box
]))
```



FOS of Nano-Silica Stabilized Fine-Grained Soil

Input

γ (19-19) 19

NS (%) (0.0-4.0) 0

CD (0-90) 0

c (30.0-276.19) 30

φ (27.57-30.74) 27.57

H (6-18) 6

β (20-60) 20

ru (0.0-0.5) 0

Output

Nano-Silica Stablized FOS (predicted by GBR model): 1.65