```
import tensorflow as tf
import numpy as np
import win32com.client
import random

# Location of Excel workbook
EXCEL_FILE = "C:\\Users\\avahu\\OneDrive\\JADE\\model\\JADE model - Revised v3.xlsx"
SHEET_NAME = "Dashboard"
MINIMUM_Y_THRESHOLD = 500

def open_workbook():
    try:
        excel = win32com.client.Dispatch("Excel.Application")
        excel.Visible = True  # Keep Excel on top
        workbook = excel.Workbooks.Open(EXCEL_FILE)
        sheet = workbook.Sheets(SHEET_NAME)
        return excel, workbook, sheet
    except Exception as e:
        print(f"Error opening Excel file: {e}")
        return None, None, None

def write_excel(sheet, cell, value):
    try:
        sheet.Range(cell).Value = value
        print(f"Written {value} to cell {cell}")
    except Exception as e:
        print(f"Error writing to Excel file at cell {cell}: {e}")

def read_excel(sheet, cell):
    try:
        value = sheet.Range(cell).Value
        print(f"Read {value} from cell {cell}")
        return value
    except Exception as e:
        print(f"Error reading from Excel file: {e}")
        return None

def objective_function(sheet, x1, x2, x3, x4, x5, x6):
```

```
    # Cell location in Excel for six input parameters
    cell_input_1 = "I2"
    cell_input_2 = "I3"
    cell_input_3 = "I4"
    cell_input_4 = "Q3"
    cell_input_5 = "Q4"
    cell_input_6 = "Q5"
    cell_output = "D44"

    # Input parameter bounds
    x1 = max(min(x1, 2), 0)
    x2 = max(min(x2, 1), 0)
    x3 = max(min(x3, 1), 0)
    x4 = max(min(x4, 1), 0)
    x5 = max(min(x5, 1), 0)
    x6 = int(round(max(min(x6, 20), 0)))  # x6 as integer between 0 and 20

    # Write input values to Excel
    write_excel(sheet, cell_input_1, x1)
    write_excel(sheet, cell_input_2, x2)
    write_excel(sheet, cell_input_3, x3)
    write_excel(sheet, cell_input_4, x4)
    write_excel(sheet, cell_input_5, x5)
    write_excel(sheet, cell_input_6, x6)

    # Read output value from Excel
    output = read_excel(sheet, cell_output)

    if output is None:
        print(f"Warning: Read None from Excel for cell {cell_output}. Check the cell references and
the Excel file.")
        return float('-inf')

    try:
        return float(output)
    except ValueError as e:
        print(f"Error converting output to float: {e}")
        return float('-inf')
```

```
if __name__ == '__main__':
    excel, workbook, sheet = open_workbook()
    if sheet:
        try:
            # TensorFlow neural network model with six inputs
            model = tf.keras.Sequential([
                tf.keras.layers.Input(shape=(6,)),
                tf.keras.layers.Dense(64, activation='relu'),
                tf.keras.layers.Dense(64, activation='relu'),
                tf.keras.layers.Dense(32, activation='relu'),
                tf.keras.layers.Dense(1, activation='linear')
            ])
            model.compile(optimizer='adam', loss='mse')
            print("TensorFlow model initialized successfully.")

            # Initial training data
            x_train = np.array([
                [1, 0.5, 0.5, 0.5, 0.5, 10],
                [2, 1.0, 0.8, 0.9, 0.6, 15],
                [1.5, 0.75, 0.3, 0.4, 0.7, 5]
            ])
            y_train = np.array([objective_function(sheet, *x) for x in x_train]).reshape(-1, 1)

            # Train model on the initial data
            model.fit(x_train, y_train, epochs=20, verbose=1)
            print("Initial training completed successfully.")

            # Optimization loop with increased iterations
            best_values = [0, 0, 0, 0, 0, 0]
            best_y = float('-inf')
            num_iterations = 500  # Change number of iterations here - may change to prompt later

            for i in range(num_iterations):
                # Generate random values within the allowed range for each input
                x1_next = random.uniform(0, 2)
                x2_next = random.uniform(0, 1)
                x3_next = random.uniform(0, 1)
```

```
            x4_next = random.uniform(0, 1)
            x5_next = random.uniform(0, 1)
            x6_next = random.randint(0, 20)  # Integer for switch year

            # Evaluate objective function
            y_next = objective_function(sheet, x1_next, x2_next, x3_next, x4_next, x5_next,
x6_next)
            if y_next < MINIMUM_Y_THRESHOLD:
                continue

            if y_next > best_y:
                best_y = y_next
                best_values = [x1_next, x2_next, x3_next, x4_next, x5_next, x6_next]

            # Append data and re-train
            x_train = np.vstack([x_train, [x1_next, x2_next, x3_next, x4_next, x5_next, x6_next]])
            y_train = np.append(y_train, y_next).reshape(-1, 1)
            model.fit(x_train, y_train, epochs=10, verbose=0)

            # Put updates on console
            if (i + 1) % 100 == 0:  # Print status every 100 iterations
                print(f"Iteration {i + 1}: Best values = {best_values}, Output = {best_y}")
        except Exception as e:
            print(f"An error occurred during optimization: {e}")
        finally:
            # Print final results
            print(f"Final Best Values: {best_values}")
            print(f"Final Best Output: {best_y}")

    # Keep the workbook open with current values
```