In [1]:
```
import numpy as np
import pandas as pd
import shap
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from scipy.optimize import differential_evolution
```

```
C:\Users\reshm\anaconda3\Lib\site-packages\pandas\core\arrays\masked.py:60: UserWarning: Pandas requires ver
sion '1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
```

In [2]:
```
# Load dataset (replace with actual file path)
df = pd.read_excel("D:/Capstone_2025/data/solardata_addis.xlsx")
df
```

Out[2]:

	Year	Month	Day	Hour	Minute	Clearsky DHI	Clearsky DNI	Temperature	Clearsky GHI	cloud fill flag	...	DNI	Fill Flag	GHI	Relative Humidity	Solar Zenith Angle	Surfa Albe
0	2006	1	1	0	30	0	0	6.9	0	0	...	0	0	0	80.78	137.73	0
1	2006	1	1	1	30	0	0	6.5	0	0	...	0	0	0	85.35	124.08	0
2	2006	1	1	2	30	0	0	6.2	0	0	...	0	0	0	89.38	110.30	0
3	2006	1	1	3	30	0	0	7.0	0	0	...	0	0	0	86.64	96.54	0
4	2006	1	1	4	30	36	484	10.8	96	1	...	0	1	39	68.50	82.85	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
17515	2022	1	31	19	30	0	0	13.7	0	0	...	0	0	0	64.84	144.22	0
17516	2022	1	31	20	30	0	0	13.6	0	0	...	0	0	0	63.55	158.14	0
17517	2022	1	31	21	30	0	0	13.3	0	0	...	0	0	0	64.07	170.03	0
17518	2022	1	31	22	30	0	0	12.8	0	0	...	0	0	0	63.95	167.90	0
17519	2022	1	31	23	30	0	0	12.1	0	0	...	0	0	0	65.48	155.12	0

17520 rows × 23 columns

In [3]:
```python
# Print column names to check if 'efficiency' exists
print(df.columns)

# Ensure correct feature names (match column names in your dataset)
features = ['Temperature', 'Relative Humidity', 'Wind Speed', 'Solar Zenith Angle']

# Check for variations in column names (e.g., spaces, capitalization)
df.columns = df.columns.str.strip()  # Remove any leading/trailing spaces

# Verify if 'efficiency' is present
if 'efficiency' not in df.columns:
    print("Column 'efficiency' not found! Check for typos or variations.")

# Correct target variable name if necessary
correct_target_name = [col for col in df.columns if 'efficiency' in col.lower()]
target = correct_target_name[0] if correct_target_name else None

if target:
    print(f"Using target column: {target}")
    X = df[features]
    y = df[target]
else:
    print("No suitable target column found. Please check your dataset.")
```

```
Index(['Year', 'Month', 'Day', 'Hour', 'Minute', 'Clearsky DHI',
       'Clearsky DNI', 'Temperature', 'Clearsky GHI', 'cloud fill flag',
       'Cloud Type', 'Dew Point', 'DHI', 'DNI', 'Fill Flag', 'GHI',
       'Relative Humidity', 'Solar Zenith Angle', 'Surface Albedo', 'Pressure',
       'Precipitable Water', 'Wind Direction', 'Wind Speed'],
      dtype='object')
Column 'efficiency' not found! Check for typos or variations.
No suitable target column found. Please check your dataset.
```

In [4]:
```python
df['efficiency'] = (df['GHI'] + df['DNI']) / (df['Clearsky GHI'] + df['Clearsky DNI'])
```

In [5]:
```python
# Define relevant features
features = ['Temperature', 'Relative Humidity', 'Wind Speed', 'Solar Zenith Angle']
target = 'efficiency'

# Split dataset
X = df[features]
y = df[target]
```

In [6]:
```python
# Check for NaN values in the target variable
print(f"Missing values in y: {y.isna().sum()}")

# Option 2: Fill missing values with the median (if removing data is not ideal)
df['efficiency'].fillna(df['efficiency'].median(), inplace=True)

# Verify again
print(f"Missing values after fixing: {df['efficiency'].isna().sum()}")

# Retry model training
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X, y)
```

```
Missing values in y: 8874
Missing values after fixing: 0
```

Out[6]:

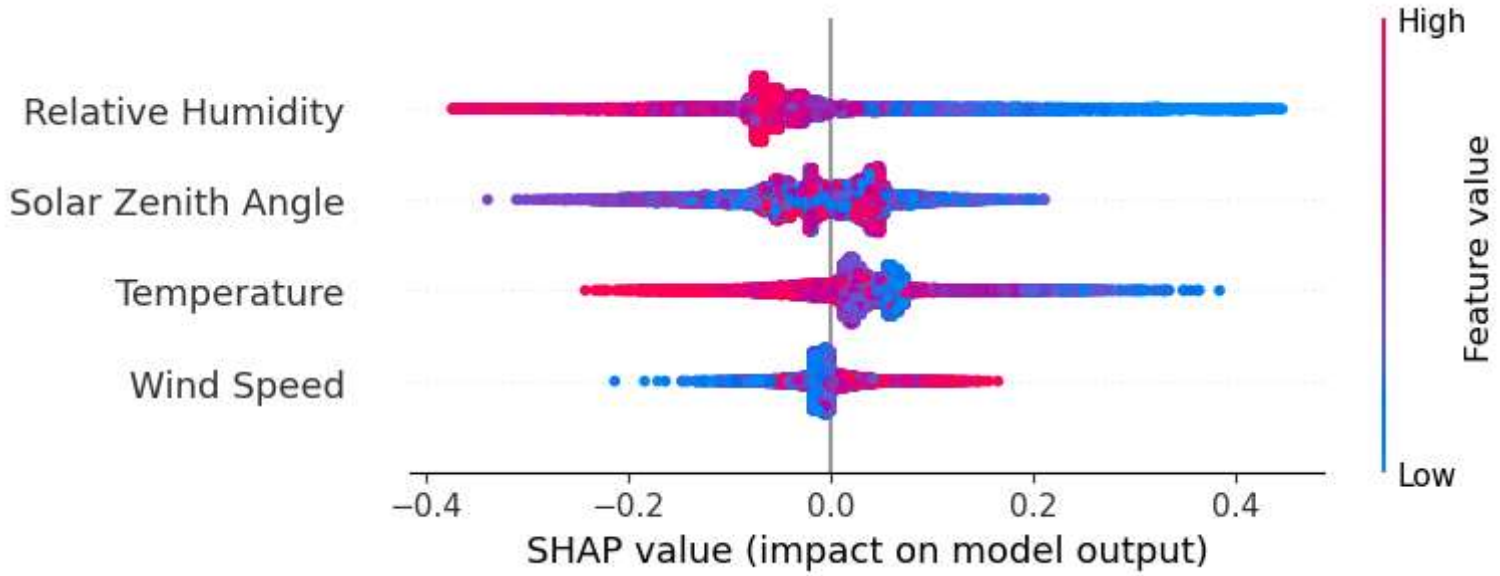
RandomForestRegressor (https://scikit-learn.org/1.5/modules/generated/sklearn.ensemble.RandomForestRegressor.html)

```
RandomForestRegressor(random_state=42)
```

Sensitivity Analysis using SHAP

In [7]:
```
# Train a Random Forest model for sensitivity analysis
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X, y)

# Perform SHAP analysis
explainer = shap.Explainer(model)
shap_values = explainer(X)
shap.summary_plot(shap_values, X)
```



Optimization of Solar Panel Tilt using Genetic Algorithm (GA)

In [9]:
```
# Define objective function for optimization
def panel_tilt_objective(angle):
    """Minimization function for tilt optimization."""
    df['adjusted_efficiency'] = df['efficiency'] * np.cos(np.radians(df['Solar Zenith Angle'] - angle))
    return -df['adjusted_efficiency'].mean()  # Negative because we minimize

# Perform Genetic Algorithm optimization
result = differential_evolution(panel_tilt_objective, bounds=[(0, 90)], strategy='best1bin', maxiter=100, pop
optimal_tilt = result.x[0]

print(f'Optimal Solar Panel Tilt Angle: {optimal_tilt:.2f} degrees')
```

```
Optimal Solar Panel Tilt Angle: 87.07 degrees
```

In [ ]:

In [ ]: