In [64]:
```
# --- 1. Import Libraries ---
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
```

In [65]:
```
# --- 2. Load and Prepare Data ---
df = pd.read_csv('global_energy_consumption.csv')
df_model = df.drop(['Year'], axis=1)
```

In [66]:
```
# --- 3. Simulate Missing Features ---
df_model['Simulated_Population'] = df_model['Total Energy Consumption (TWh)'] / df_
df_model['Simulated_GDP'] = df_model['Simulated_Population'] * df_model['Energy Pri
```

In [67]:
```
# --- 4. Feature Engineering ---
df_model['Fossil_Energy_Used'] = df_model['Total Energy Consumption (TWh)'] * (df_m
df_model['Industrial × Fossil'] = df_model['Industrial Energy Use (%)'] * df_model[
df_model['Household × Price'] = df_model['Household Energy Use (%)'] * df_model['En
df_model['Energy_Intensity'] = df_model['Total Energy Consumption (TWh)'] / df_mode
df_model['Fossil_Intensity'] = df_model['Fossil_Energy_Used'] / df_model['Per Capit
```

In [68]:
```
# --- 5. Define New Target: Carbon Intensity ---
df_model['Carbon_Intensity'] = df_model['Carbon Emissions (Million Tons)'] / df_mod
y = np.log1p(df_model['Carbon_Intensity'])
```

In [69]:
```
# --- 6. Define Features ---
X = df_model.drop(columns=['Country', 'Carbon Emissions (Million Tons)', 'Carbon_In
```

In [70]:
```
# --- 7. Scale Features ---
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [71]:
```
# --- 8. Train-Test Split ---
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, ran
```

In [72]:
```
# --- 9. Train Gradient Boosting Model ---
gb = GradientBoostingRegressor(n_estimators=300, learning_rate=0.05, max_depth=6, r
gb.fit(X_train, y_train)
```

Out[72]:
```
▾ GradientBoostingRegressor

GradientBoostingRegressor(learning_rate=0.05, max_depth=6, n_estimators=30
0,
                          random_state=42)
```

In [73]:
```
# --- 10. Predict and Reverse Log Transform ---
y_pred_log = gb.predict(X_test)
```

```
y_test_exp = np.expm1(y_test)
y_pred_exp = np.expm1(y_pred_log)
```

In [74]:
```
# --- 11. Evaluate Model ---
def evaluate(y_true, y_pred, model_name):
    print(f"\n{model_name} Results:")
    print(f"R² Score: {r2_score(y_true, y_pred):.4f}")
    print(f"RMSE: {np.sqrt(mean_squared_error(y_true, y_pred)):.4f}")
    print("-" * 30)
```

In [75]:
```
evaluate(y_test_exp, y_pred_exp, "Gradient Boosting (Carbon Intensity)")
```

```
Gradient Boosting (Carbon Intensity) Results:
R² Score: 0.6484
RMSE: 1.3719
------------------------------
```

In [ ]: