The problem is being resolved by including multiple machine learning models for classification and then comparing their performance. Below, I tried to include additional models like **Logistic Regression**, **Support Vector Machine (SVM)**, **K-Nearest Neighbors (KNN)**, and **Gradient Boosting**. After training these models, we evaluated them using a classification report and confusion matrix.

Step 1: Prepare the Data for Classification

Since we're focusing on classification, we'll binarize the target variable (energy consumption) into two classes: ***low and high consumption.***

Step 2: Train and Evaluate Multiple Machine Learning Models

Code to implement and evaluate the models:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

from sklearn.linear_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import classification_report, confusion_matrix

from sklearn.model_selection import train_test_split


# Simulate GIS Data

gis_data = pd.DataFrame({

    'Building_ID': range(1, 101),

    'Building_Footprint': np.random.uniform(50, 200, 100),  # in square meters

    'Building_Height': np.random.uniform(5, 50, 100),  # in meters

    'Roof_Type': np.random.choice(['Flat', 'Gabled', 'Hipped'], 100)

})


# Simulate Drone Data

drone_data = pd.DataFrame({

    'Building_ID': range(1, 101),
```

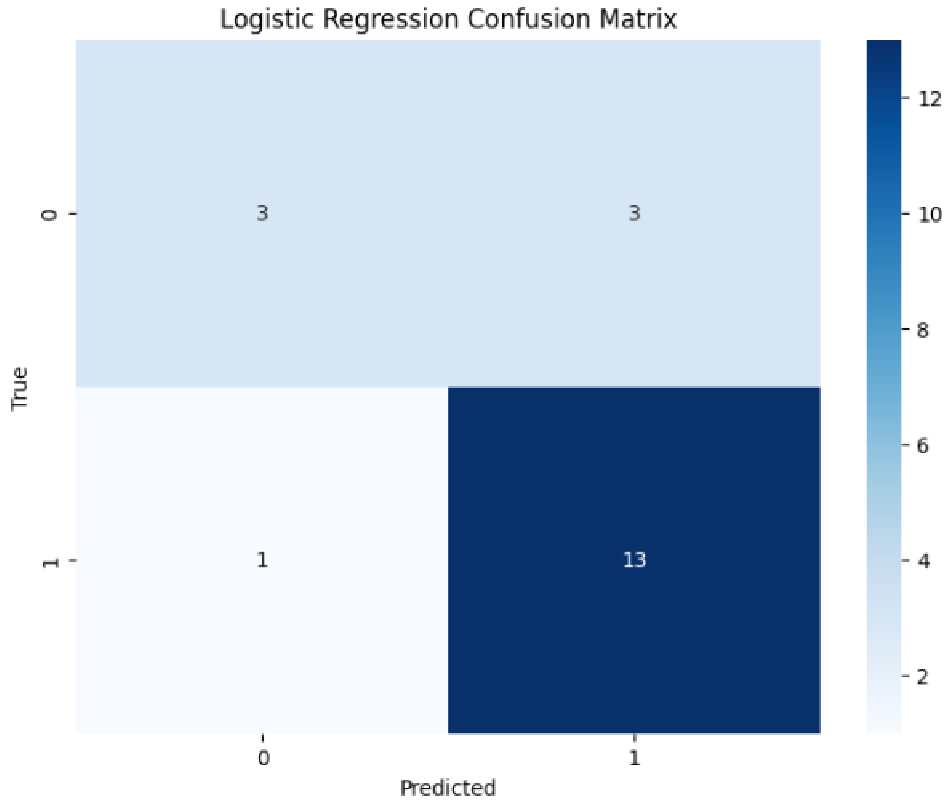
```python
    'Solar_Potential': np.random.uniform(100, 500, 100),  # kWh/year
    'Shape_Complexity': np.random.randint(1, 10, 100)  # Arbitrary complexity score
})

# Simulate Historical Energy Data
historical_data = pd.DataFrame({
    'Building_ID': range(1, 101),
    'Annual_Energy_Consumption': np.random.uniform(5000, 15000, 100),  # kWh/year
    'Peak_Demand': np.random.uniform(1, 5, 100)  # kW
})

# Merge datasets based on Building_ID
data = pd.merge(gis_data, drone_data, on='Building_ID')
data = pd.merge(data, historical_data, on='Building_ID')

# Feature engineering
data['Footprint_Height_Ratio'] = data['Building_Footprint'] / data['Building_Height']
data['Energy_Intensity'] = data['Annual_Energy_Consumption'] / data['Building_Footprint']

# Convert categorical variables to numerical
data = pd.get_dummies(data, columns=['Roof_Type'], drop_first=True)

# Features and target
X = data.drop(columns=['Building_ID', 'Annual_Energy_Consumption'])
y = pd.cut(data['Annual_Energy_Consumption'], bins=[0, 7500, 15000], labels=[0, 1])

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize models
models = {
```

```python
    'Logistic Regression': LogisticRegression(),
    'Random Forest': RandomForestClassifier(n_estimators=100, random_state=42),
    'Support Vector Machine': SVC(),
    'K-Nearest Neighbors': KNeighborsClassifier(n_neighbors=5),
    'Gradient Boosting': GradientBoostingClassifier(n_estimators=100, random_state=42)
}

# Train and evaluate each model
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # Print the classification report
    print(f"Classification Report for {name}:")
    print(classification_report(y_test, y_pred))

    # Confusion Matrix
    cm = confusion_matrix(y_test, y_pred)

    # Plot Confusion Matrix
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title(f'{name} Confusion Matrix')
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.show()
```

VISUALIZED RESULTS OF 5 MACHINE LEARNING CLASSIFIERS

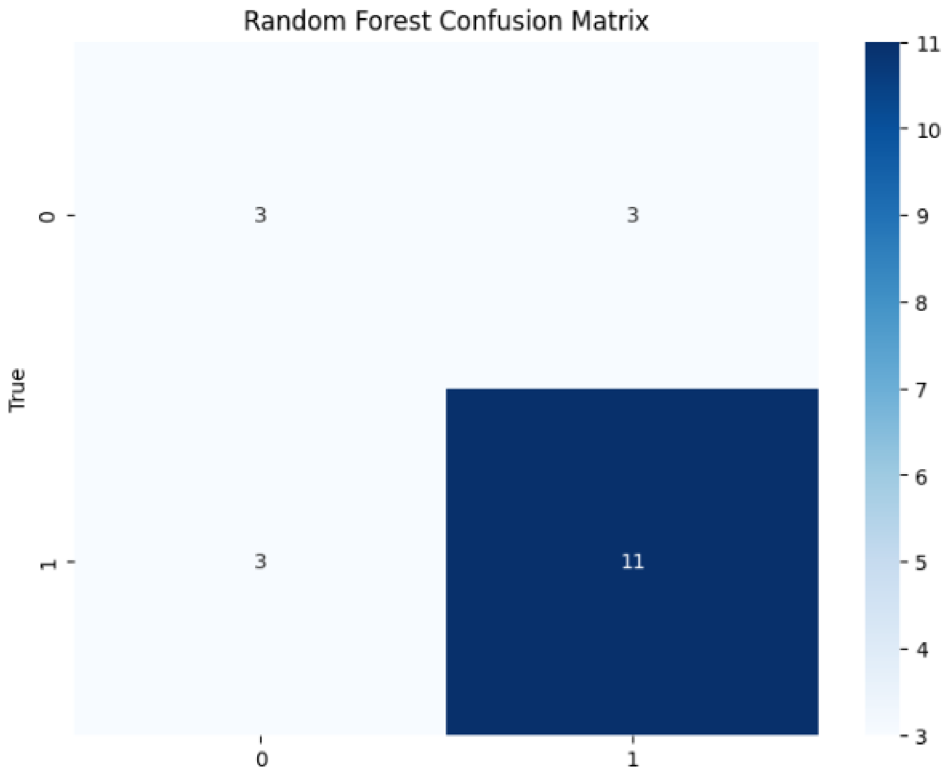
Classification Report for Logistic Regression:

	precision	recall	f1-score	support
0	0.75	0.50	0.60	6
1	0.81	0.93	0.87	14
accuracy			0.80	20
macro avg	0.78	0.71	0.73	20
weighted avg	0.79	0.80	0.79	20



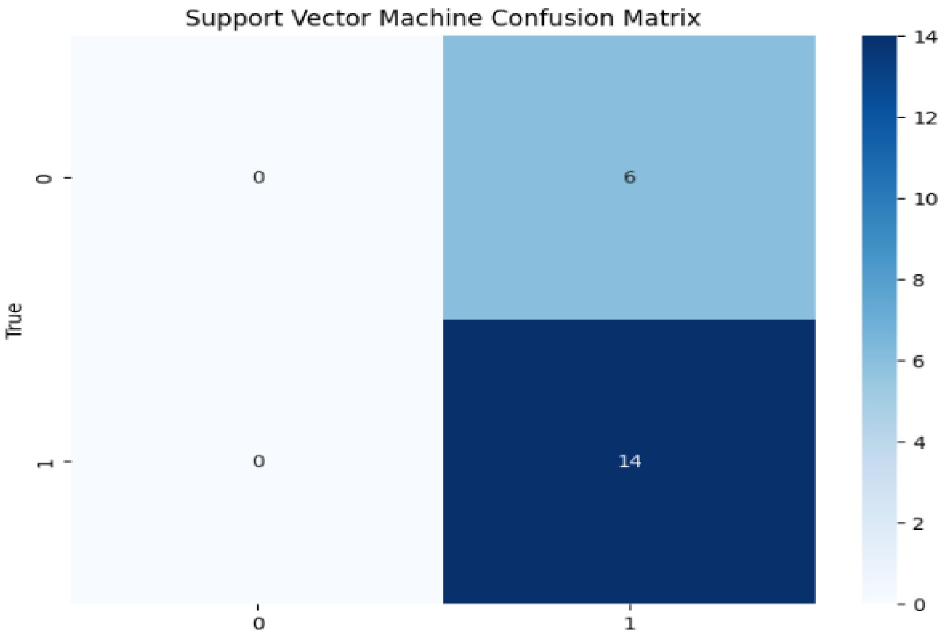
Classification Report for Random Forest:

	precision	recall	f1-score	support
0	0.50	0.50	0.50	6
1	0.79	0.79	0.79	14
accuracy			0.70	20
macro avg	0.64	0.64	0.64	20
weighted avg	0.70	0.70	0.70	20

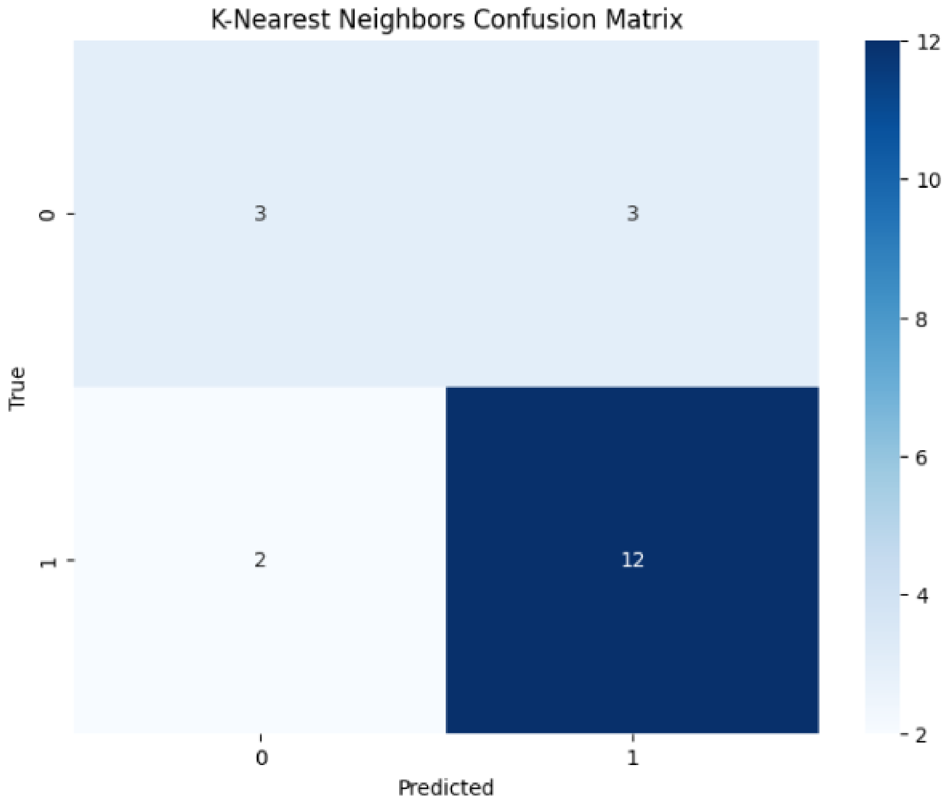


Classification Report for Support Vector Machine:

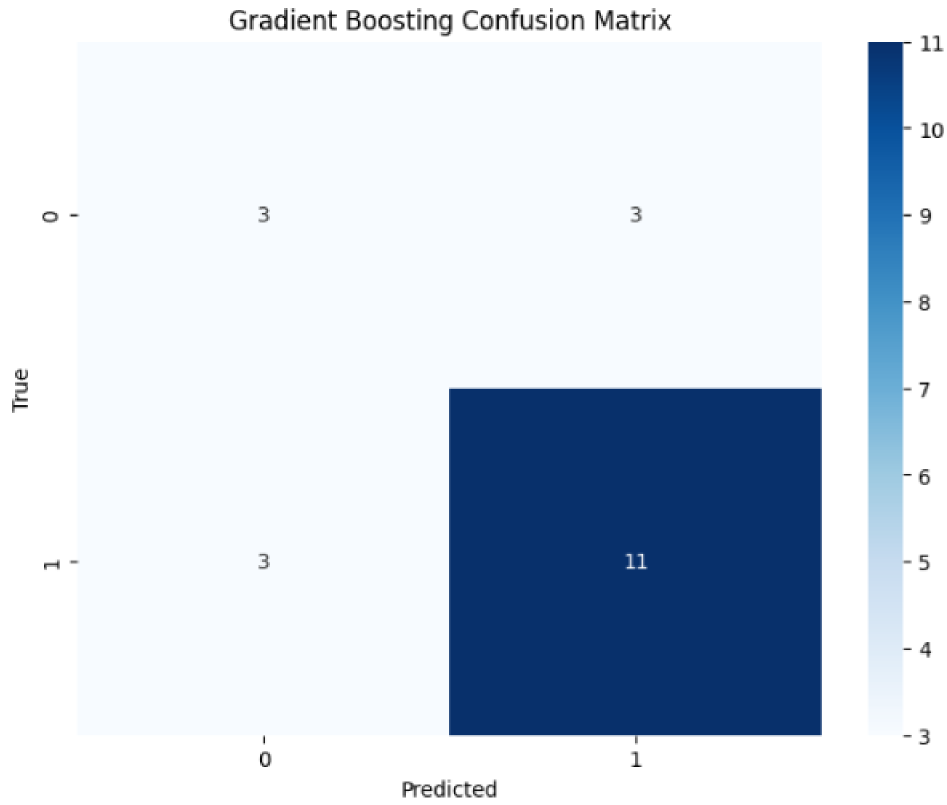
	precision	recall	f1-score	support
0	0.00	0.00	0.00	6
1	0.70	1.00	0.82	14
accuracy			0.70	20
macro avg	0.35	0.50	0.41	20
weighted avg	0.49	0.70	0.58	20



```
Classification Report for K-Nearest Neighbors:
              precision    recall  f1-score   support

           0       0.60      0.50      0.55         6
           1       0.80      0.86      0.83        14

    accuracy                           0.75        20
   macro avg       0.70      0.68      0.69        20
weighted avg       0.74      0.75      0.74        20
```




```
Classification Report for Gradient Boosting:
              precision    recall  f1-score   support

           0       0.50      0.50      0.50         6
           1       0.79      0.79      0.79        14

    accuracy                           0.70        20
   macro avg       0.64      0.64      0.64        20
weighted avg       0.70      0.70      0.70        20
```



Model Accuracy Comparison chart