```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

df=pd.read_csv("Energy Dataset.csv")

df.head()
```

	X1	X2	X3	X4	X5	X6	X7	X8	Heating Load
0	0.98	514.5	294.0	110.25	7.0	2	0.0	0	15.55
1	0.98	514.5	294.0	110.25	7.0	3	0.0	0	15.55
2	0.98	514.5	294.0	110.25	7.0	4	0.0	0	15.55
3	0.98	514.5	294.0	110.25	7.0	5	0.0	0	15.55
4	0.90	563.5	318.5	122.50	7.0	2	0.0	0	20.84

```
df.isnull().sum()
```

```
X1              0
X2              0
X3              0
X4              0
X5              0
X6              0
X7              0
X8              0
Heating Load    0
dtype: int64
```

```
# Step 3: Separate features (X) and target variable (y: heating load)
X = df.drop('Heating Load', axis=1)  # Features
y = df['Heating Load']  # Target variable

# Step 4: Split the dataset into training and testing sets (80:20
ratio)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Step 5: Perform multi-linear regression
# Fit a multi-linear regression model
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)
```

```
LinearRegression()
```

```
# Predict heating load for the testing data
y_pred_linear = linear_model.predict(X_test)

# Evaluate the performance of the model
mse_linear = mean_squared_error(y_test, y_pred_linear)
r2_linear = r2_score(y_test, y_pred_linear)

# Print MSE and R^2 values for multi-linear regression
print("Multi-Linear Regression:")
print(f"MSE: {mse_linear}")
print(f"R^2: {r2_linear}")
```

```
Multi-Linear Regression:
MSE: 9.139792788392425
R^2: 0.9123126077484475
```

```
# Step 6: Perform polynomial regression
# Use PolynomialFeatures to transform features into polynomial
features
degree = 2  # You can adjust the degree as needed
poly_features = PolynomialFeatures(degree=degree)
X_train_poly = poly_features.fit_transform(X_train)
X_test_poly = poly_features.transform(X_test)

# Fit a polynomial regression model
poly_model = make_pipeline(StandardScaler(), LinearRegression())
poly_model.fit(X_train_poly, y_train)
```

```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('linearregression', LinearRegression())])
```

```
# Predict heating load for the testing data
y_pred_poly = poly_model.predict(X_test_poly)

# Evaluate the performance of the model
mse_poly = mean_squared_error(y_test, y_pred_poly)
r2_poly = r2_score(y_test, y_pred_poly)

# Print MSE and R^2 values for polynomial regression
print("\nPolynomial Regression:")
print(f"MSE: {mse_poly}")
print(f"R^2: {r2_poly}")
```

```
Polynomial Regression:
MSE: 0.6453340037152746
R^2: 0.9938086500178743
```

```
# Step 7: Compare the performance of multi-linear and polynomial
regression
print("\nComparison:")
```

```
print(f"Multi-Linear Regression MSE: {mse_linear}, R^2: {r2_linear}")
print(f"Polynomial Regression MSE: {mse_poly}, R^2: {r2_poly}")
```

```
Comparison:
Multi-Linear Regression MSE: 9.139792788392425, R^2:
0.9123126077484475
Polynomial Regression MSE: 0.6453340037152746, R^2: 0.9938086500178743
```