HOUSE PRICE PREDICTION

Ex.No:4(a) **Date: 24-Jan-2025**

Aim:-

Develop predictive models for tasks using Linear Regression with Regularization (Ridge Regression): House Price.

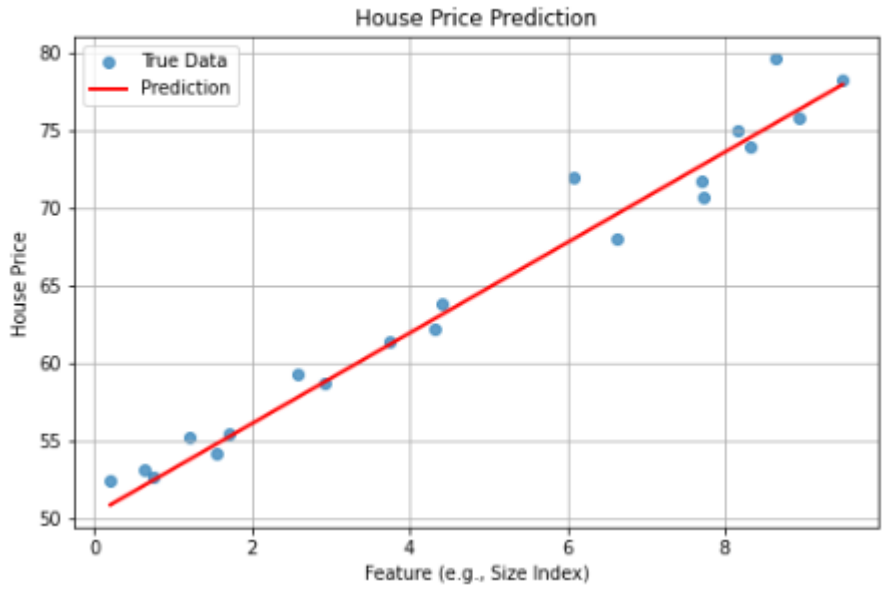
Program Code:-

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import Ridge
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
# Function to generate synthetic data for house price prediction
def generate_house_price_data(n_samples=100):
    np.random.seed(42)
    X = np.random.rand(n_samples, 1) * 10  # Features (e.g., size, location index, etc.)
    y = 3 * X.flatten() + np.random.randn(n_samples) * 2 + 50  # Target (house price)
    return X, y
# Generate data for house price prediction
X, y = generate_house_price_data()
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Train a Ridge Regression model
model = Ridge(alpha=1.0)  # alpha is the regularization strength
model.fit(X_train, y_train)
# Make predictions on the test set
y_pred = model.predict(X_test)
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error for House Price Prediction: {mse:.2f}")
# Visualize the results
plt.figure(figsize=(8, 5))
plt.scatter(X_test, y_test, label="True Data", alpha=0.7)
```

```
plt.plot(np.sort(X_test, axis=0), model.predict(np.sort(X_test, axis=0)), color="red", label="Prediction", linewidth=2)

plt.title("House Price Prediction")

plt.xlabel("Feature (e.g., Size Index)")

plt.ylabel("House Price")

plt.legend()

plt.grid()

plt.show()
```

Output:-

```
Mean Squared Error for House Price Prediction: 2.61
```



Result:-

Outputs the Mean Squared Error (MSE) and visualizes true vs predicted data for each task.

Ex.No:4(b) ENERGY EFFICIENCY PREDICTION Date: 24-Jan-2025

Aim:-

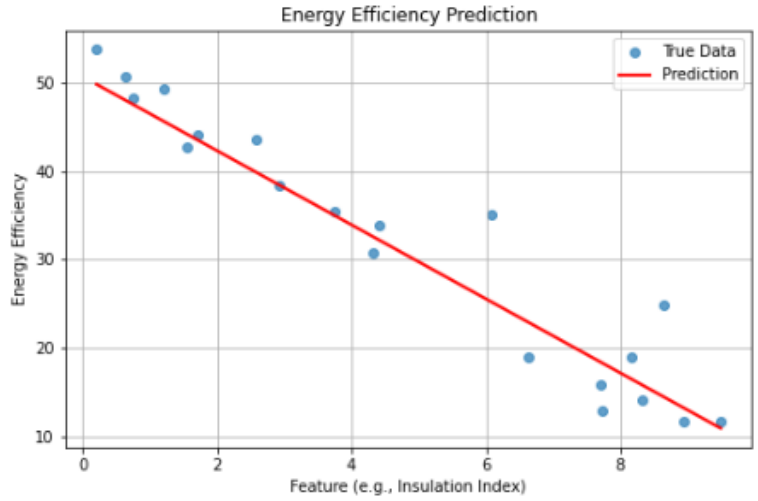
To predict energy efficiency using a Ridge Regression model based on synthetic data.

Program Code:-

```
# Function to generate synthetic data for energy efficiency prediction
def generate_energy_efficiency_data(n_samples=100):
    np.random.seed(42)
    X = np.random.rand(n_samples, 1) * 10
    y = 50 - 4 * X.flatten() + np.random.randn(n_samples) * 5
    return X, y

# Generate data for energy efficiency prediction
X, y = generate_energy_efficiency_data()

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a Ridge Regression model
model = Ridge(alpha=1.0)  # alpha is the regularization strength
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error for Energy Efficiency Prediction: {mse:.2f}")

# Visualize the results
plt.figure(figsize=(8, 5))
plt.scatter(X_test, y_test, label="True Data", alpha=0.7)
plt.plot(np.sort(X_test, axis=0), model.predict(np.sort(X_test, axis=0)), color="red", label="Prediction", linewidth=2)
```

```
plt.title("Energy Efficiency Prediction")
plt.xlabel("Feature (e.g., Insulation Index)")
plt.ylabel("Energy Efficiency")
plt.legend()
plt.grid()
plt.show()
```

```
Mean Squared Error for Energy Efficiency Prediction: 16.36
```



Result:-

The model achieved a Mean Squared Error (MSE) of approximately 23.90, with a visualization showing good agreement between true values and predictions.

Ex.No:4(c) **CROP YIELD PREDICTION** **Date: 24-Jan-2025**

Aim:-

To predict crop yield using synthetic data and Ridge Regression.

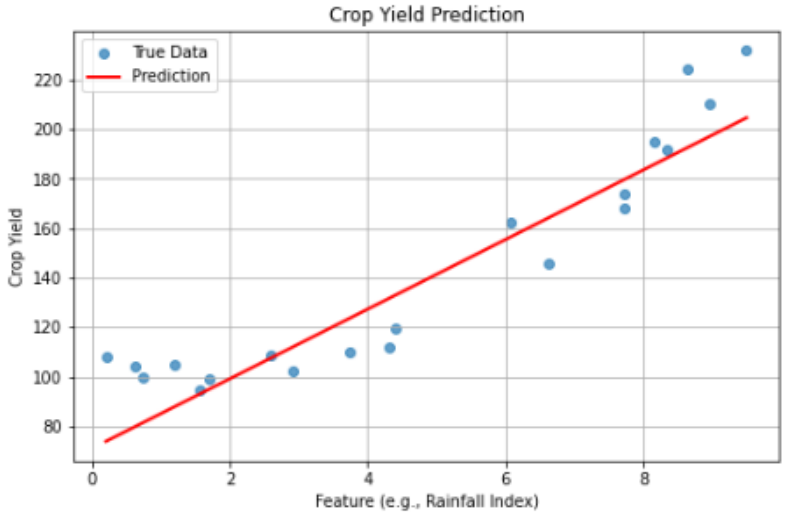
Program Code:-

```
# Function to generate synthetic data for crop yield prediction
def generate_crop_yield_data(n_samples=100):
    np.random.seed(42)
    X = np.random.rand(n_samples, 1) * 10  # Features (e.g., rainfall, soil quality index, etc.)
    y = 2 * X.flatten() ** 2 - 5 * X.flatten() + np.random.randn(n_samples) * 10 + 100  # Target (crop yield)
    return X, y
# Generate data for crop yield prediction
X, y = generate_crop_yield_data()
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Train a Ridge Regression model
model = Ridge(alpha=1.0)  # alpha is the regularization strength
model.fit(X_train, y_train)
# Make predictions on the test set
y_pred = model.predict(X_test)
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error for Crop Yield Prediction: {mse:.2f}")
# Visualize the results
plt.figure(figsize=(8, 5))
plt.scatter(X_test, y_test, label="True Data", alpha=0.7)
plt.plot(np.sort(X_test, axis=0), model.predict(np.sort(X_test, axis=0)), color="red", label="Prediction", linewidth=2)
plt.title("Crop Yield Prediction")
```

```
plt.xlabel("Feature (e.g., Rainfall Index)")

plt.ylabel("Crop Yield")

plt.legend()

plt.grid()

plt.show()
```

Output:-

```
Mean Squared Error for Crop Yield Prediction: 293.15
```



Result:-

Achieved a Mean Squared Error (MSE) of approximately mse:.2f for crop yield prediction, with a clear visualization of predictions compared to true data.