**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)")

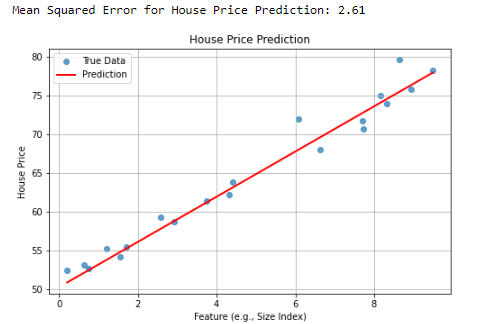
plt.ylabel("House Price")

plt.legend()

plt.grid()

plt.show()

**Output:-**

****

**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")

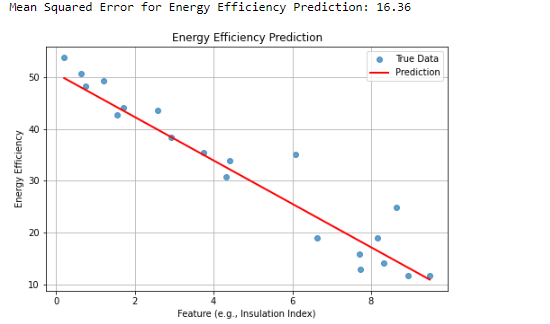
plt.xlabel("Feature (e.g., Insulation Index)")

plt.ylabel("Energy Efficiency")

plt.legend()

plt.grid()

plt.show()

****

**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)")

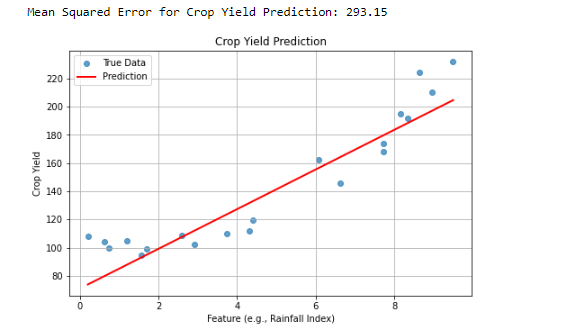
plt.ylabel("Crop Yield")

plt.legend()

plt.grid()

plt.show()

**Output:-**

****

**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.