Aim

Implement **Naïve Bayes** on the **Iris** and **California Housing** datasets and evaluate its performance. The objective is to apply **Gaussian Naïve Bayes** to both datasets, process the data, perform model evaluation, and compare results using accuracy and confusion matrix for classification, and Mean Squared Error (MSE) for regression (after binning).

Algorithm

1. **Data Loading**: Load the **Iris** dataset (classification) and **California Housing** dataset (regression).
2. **Preprocessing**:
   - Standardize the features of the **California Housing** dataset.
   - For the **Iris** dataset, no scaling is necessary.
3. **Model Training**:
   - Apply **Gaussian Naïve Bayes** to the **Iris** and **California Housing** datasets.
   - For **California Housing**, bin the continuous target variable for classification.
4. **Model Evaluation**:
   - For the **Iris dataset**, evaluate using **accuracy** and **confusion matrix**.
   - For the **California Housing dataset**, evaluate using **Mean Squared Error (MSE)** after converting the regression task to a classification one using binning.

Algorithm Description

- **Naïve Bayes** works on the assumption that features are conditionally independent given the class label. For **Gaussian Naïve Bayes**, it assumes that each feature follows a Gaussian (normal) distribution.
- **For the Iris dataset**, it classifies iris plant species based on features like petal and sepal length.
- **For the California Housing dataset**, it bins the continuous target variable (house prices) into discrete categories and performs classification, mapping the bin predictions back to continuous values for MSE evaluation.

Result

- **Iris Dataset**:
  - **Accuracy**: 97.78%
  - **Confusion Matrix**: Displays the distribution of correct and incorrect predictions for each class.
- **California Housing Dataset**:
  - **Predictions** (first 10 values): [0.68888111 0.68888111 0.68888111 2.30555444 2.30555444 2.30555444 2.30555444 2.30555444 2.30555444 4.46111889]
  - **Mean Squared Error (MSE)**: 8.261613

```python
from sklearn.datasets import load_iris
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.preprocessing import StandardScaler
import numpy as np
from sklearn.metrics import mean_squared_error

iris = load_iris()
X = iris.data
y = iris.target

y
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

```python
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

nb_classifier = GaussianNB()

nb_classifier.fit(X_train, y_train)
```

```
GaussianNB()
```

```python
y_pred = nb_classifier.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print(f"Accuracy: {accuracy * 100:.2f}%")
print("Confusion Matrix:")
print(conf_matrix)
```

```
Accuracy: 97.78%
Confusion Matrix:
[[19  0  0]
```

```
 [ 0 12  1]
 [ 0  0 13]]
```

```
data = fetch_california_housing()

X = data.data
y = data.target

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

num_bins = 10
y_binned = np.digitize(y, bins=np.linspace(y.min(), y.max(),
num_bins))

X_train, X_test, y_train, y_test = train_test_split(X_scaled,
y_binned, test_size=0.3, random_state=42)

gnb = GaussianNB()
gnb.fit(X_train, y_train)
```

```
GaussianNB()
```

```
y_pred_binned = gnb.predict(X_test)

bin_centers = np.linspace(y.min(), y.max(), num_bins)
y_pred_continuous = bin_centers[y_pred_binned - 1]

mse = mean_squared_error(y_test, y_pred_continuous)

print("Predictions (first 10):", y_pred_continuous[:10])
print("Mean Squared Error:", mse)
```

```
Predictions (first 10): [0.68888111 0.68888111 0.68888111 2.30555444
2.30555444 2.30555444
 2.30555444 2.30555444 2.30555444 4.46111889]
Mean Squared Error: 8.261613893860186
```