9. Develop a program to implement the Naive Bayesian classifier considering Olivetti Face Data set for training. Compute the accuracy of the classifier, considering a few test data sets.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Step 1: Manually Load the Olivetti Faces dataset
#dataset_path = "C:/Users/HP/datasets/olivetti_faces/"
X = np.load("input/olivetti_faces.npy")
y = np.load("input/olivetti_faces_target.npy")

# Step 2: Flatten the images (reshape from 3D to 2D)
X = X.reshape(X.shape[0], -1)

# Step 3: Split the dataset 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)

# Step 4: Train the Naive Bayesian Classifier
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# Step 5: Predict on the test set
y_pred = gnb.predict(X_test)

# Step 6: Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))

# Step 7: Visualize some test images with predictions
n_images = 5
plt.figure(figsize=(10, 4))
for i in range(n_images):
```

```
    plt.subplot(1, n_images, i + 1)
    plt.imshow(X_test[i].reshape(64, 64), cmap='gray')
    plt.title(f"Pred: {y_pred[i]}")
    plt.axis('off')
plt.show()
```

Output

Accuracy: 83.75%

Classification Report:

	precision	recall	f1-score	support
0	0.75	1.00	0.86	3
1	1.00	1.00	1.00	1
2	1.00	1.00	1.00	2
3	1.00	1.00	1.00	4
4	1.00	0.67	0.80	3
5	1.00	1.00	1.00	3
7	0.00	0.00	0.00	6
8	1.00	1.00	1.00	2
9	0.29	1.00	0.44	2
10	1.00	0.50	0.67	2
11	1.00	0.67	0.80	3
12	0.50	0.50	0.50	2
13	1.00	1.00	1.00	1
14	1.00	1.00	1.00	3
15	0.50	1.00	0.67	2
16	0.00	0.00	0.00	0
17	1.00	0.33	0.50	3
18	1.00	1.00	1.00	1
19	1.00	1.00	1.00	1
20	1.00	1.00	1.00	1
21	1.00	1.00	1.00	1
22	1.00	1.00	1.00	3
23	1.00	1.00	1.00	2
24	1.00	1.00	1.00	1
25	0.50	1.00	0.67	1
26	1.00	0.75	0.86	4
27	1.00	1.00	1.00	2
28	1.00	1.00	1.00	2
29	1.00	1.00	1.00	1
32	1.00	1.00	1.00	3

33	1.00	1.00	1.00	1
34	0.50	1.00	0.67	1
35	1.00	1.00	1.00	1
36	1.00	1.00	1.00	2
37	1.00	1.00	1.00	2
38	1.00	1.00	1.00	4
39	0.80	1.00	0.89	4
accuracy			0.84	80
macro avg	0.86	0.88	0.85	80
weighted avg	0.85	0.84	0.82	80

Confusion Matrix:

```
[[3 0 0 ... 0 0 0]
 [0 1 0 ... 0 0 0]
 [0 0 2 ... 0 0 0]
 ...
 [0 0 0 ... 2 0 0]
 [0 0 0 ... 0 4 0]
 [0 0 0 ... 0 0 4]]
```