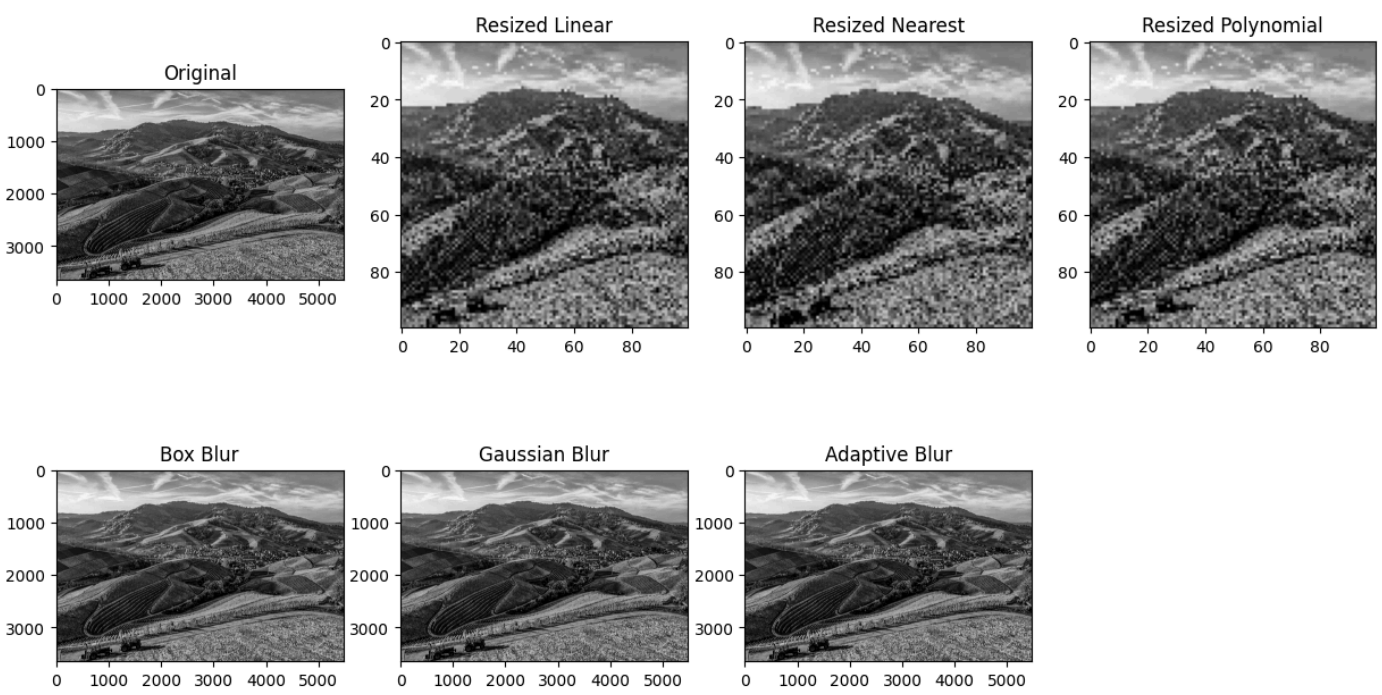
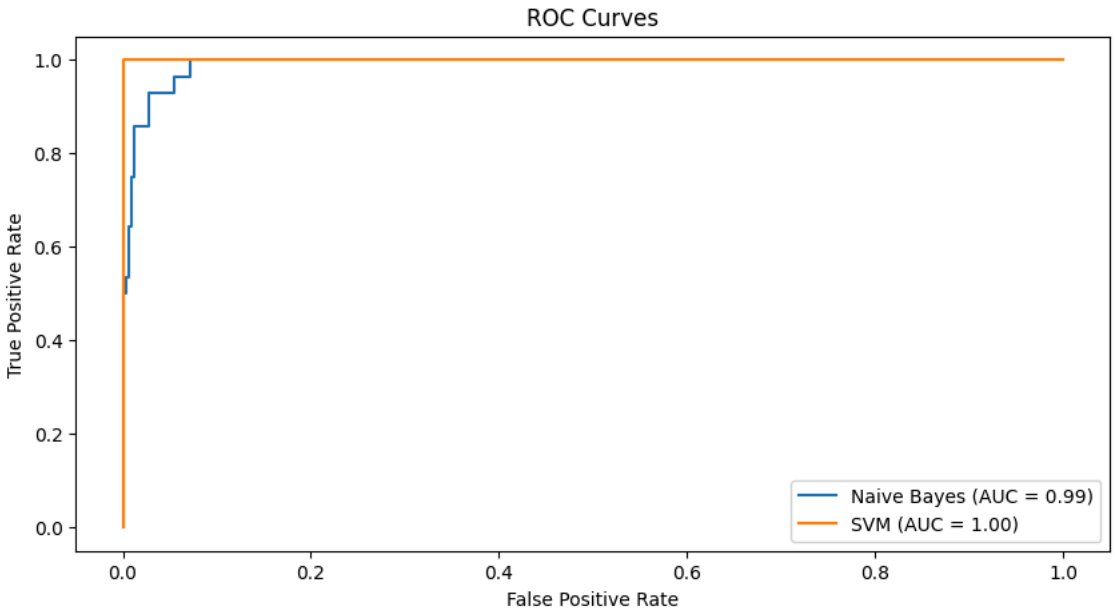
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, roc_curve, auc

# Task 1: Image Processing

def process_image(image_path):
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

    # Resizing with different interpolation methods
    resized_linear = cv2.resize(image, (100, 100), interpolation=cv2.INTER_LINEAR)
    resized_nn = cv2.resize(image, (100, 100), interpolation=cv2.INTER_NEAREST)
    resized_poly = cv2.resize(image, (100, 100), interpolation=cv2.INTER_CUBIC)

    # Blurring techniques
    box_blur = cv2.blur(image, (5, 5))
    gaussian_blur = cv2.GaussianBlur(image, (5, 5), 0)
    adaptive_blur = cv2.medianBlur(image, 5)

    # Displaying images
    fig, axs = plt.subplots(2, 4, figsize=(15, 8))
    axs[0, 0].imshow(image, cmap='gray'); axs[0, 0].set_title("Original")
    axs[0, 1].imshow(resized_linear, cmap='gray'); axs[0, 1].set_title("Resized Linear")
    axs[0, 2].imshow(resized_nn, cmap='gray'); axs[0, 2].set_title("Resized Nearest")
    axs[0, 3].imshow(resized_poly, cmap='gray'); axs[0, 3].set_title("Resized Polynomial")

    axs[1, 0].imshow(box_blur, cmap='gray'); axs[1, 0].set_title("Box Blur")
    axs[1, 1].imshow(gaussian_blur, cmap='gray'); axs[1, 1].set_title("Gaussian Blur")
    axs[1, 2].imshow(adaptive_blur, cmap='gray'); axs[1, 2].set_title("Adaptive Blur")
    axs[1, 3].axis('off')

    plt.show()


def train_ml_models():
    digits = load_digits()
    X, y = digits.data, digits.target
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    models = {
        "Naive Bayes": GaussianNB(),
        "SVM": SVC(probability=True)
    }

    results = {}
    plt.figure(figsize=(10, 5))
    for name, model in models.items():
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        y_prob = model.predict_proba(X_test)[:, 1] if hasattr(model, "predict_proba") else None

        acc = accuracy_score(y_test, y_pred)
        prec = precision_score(y_test, y_pred, average='macro')
        recall = recall_score(y_test, y_pred, average='macro')
        f1 = f1_score(y_test, y_pred, average='macro')
        cm = confusion_matrix(y_test, y_pred)

        results[name] = (acc, prec, recall, f1, cm)

        if y_prob is not None:
            fpr, tpr, _ = roc_curve(y_test, y_prob, pos_label=1)
            plt.plot(fpr, tpr, label=f"{name} (AUC = {auc(fpr, tpr):.2f})")

    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.title("ROC Curves")
    plt.legend()
    plt.show()

    return results
```

```
process_image('image.jpg')  # Provide an actual image path
results = train_ml_models()
print("Model Performance:")
for model, metrics in results.items():
    print(f"{model}: Accuracy={metrics[0]:.2f}, Precision={metrics[1]:.2f}, Recall={metrics[2]:.2f}, F1-score={metrics[3]:.2f}")
```






```
Model Performance:
Naive Bayes: Accuracy=0.85, Precision=0.86, Recall=0.85, F1-score=0.84
SVM: Accuracy=0.99, Precision=0.99, Recall=0.99, F1-score=0.99
```

Start coding or generate with AI.