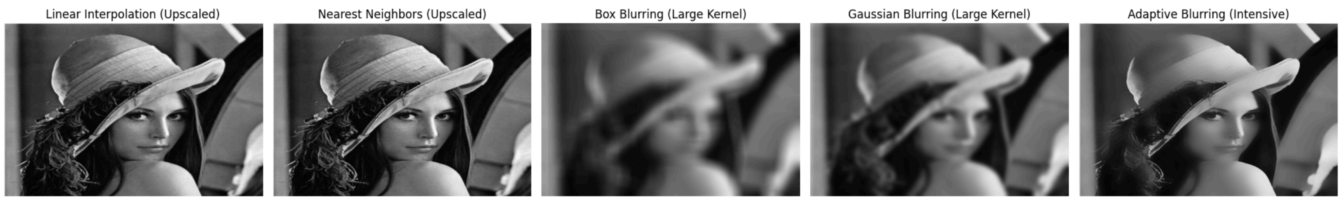
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Load an image
image = cv2.imread('/content/useimg.png')
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  # Convert from BGR to RGB
```

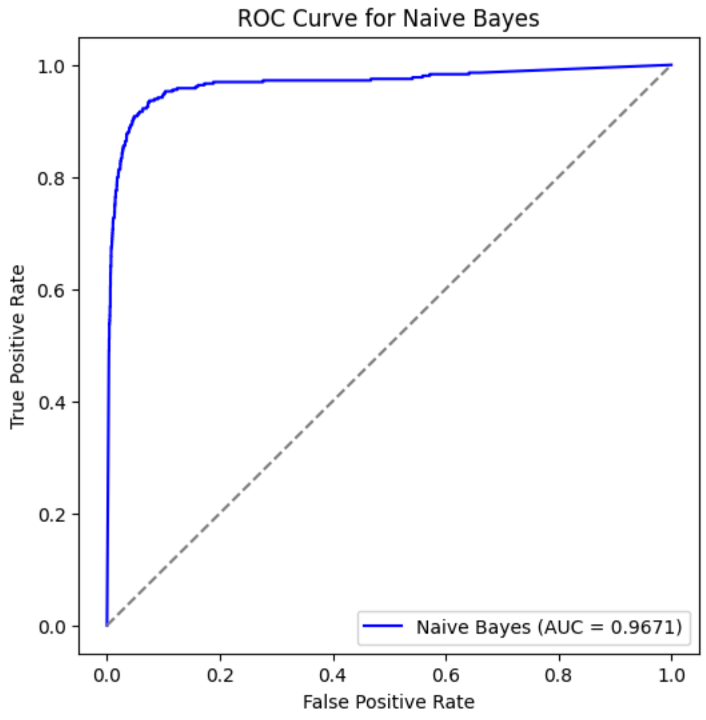
```
def display_images(images, titles, cmap=None):
    plt.figure(figsize=(20, 10))
    for i, (img, title) in enumerate(zip(images, titles), 1):
        plt.subplot(1, len(images), i)
        plt.imshow(img, cmap=cmap)
        plt.title(title)
        plt.axis('off')
    plt.tight_layout()
    plt.show()

linear_resized = cv2.resize(image, None, fx=2, fy=2, interpolation=cv2.INTER_LINEAR)
nearest_resized = cv2.resize(image, None, fx=2, fy=2, interpolation=cv2.INTER_NEAREST)

box_blurred = cv2.blur(image, (25, 25))  # Larger Box blur kernel
gaussian_blurred = cv2.GaussianBlur(image, (25, 25), 0)  # Larger Gaussian blur kernel
adaptive_blur = cv2.bilateralFilter(image, 25, 150, 150)  # More intensive adaptive blurring

# Collect images and their titles for display
resized_images = [linear_resized, nearest_resized]
blurred_images = [box_blurred, gaussian_blurred, adaptive_blur]
titles_resized = ['Linear Interpolation (Upscaled)', 'Nearest Neighbors (Upscaled)']
titles_blurred = ['Box Blurring (Large Kernel)', 'Gaussian Blurring (Large Kernel)', 'Adaptive Blurring (Intensive)']

# Display the results
display_images(resized_images + blurred_images, titles_resized + titles_blurred)
```


Linear Interpolation (Upscaled)
Nearest Neighbors (Upscaled)
Box Blurring (Large Kernel)
Gaussian Blurring (Large Kernel)
Adaptive Blurring (Intensive)

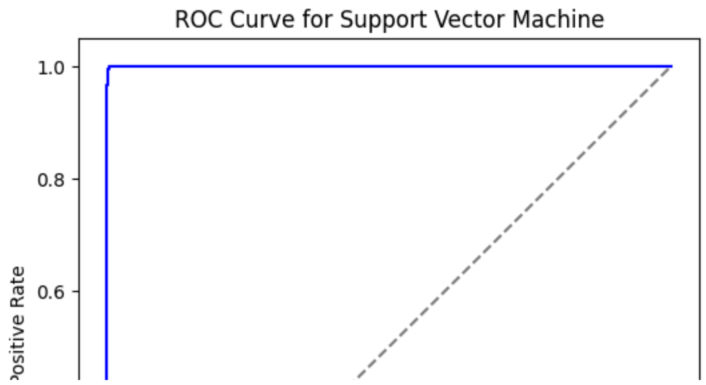
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, StratifiedKFold, cross_val_score
from sklearn.datasets import load_digits
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score,
    confusion_matrix, roc_curve, auc
)
from sklearn.preprocessing import label_binarize

# Load the dataset
digits = load_digits()
X = digits.data  # Features
y = digits.target  # Labels

# Normalize feature values
X = X / 16.0

# 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, stratify=y)

# Setup for K-fold cross-validation
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

def compute_metrics(y_test, y_pred, model_name, y_prob=None):
    acc = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, average='weighted')
    recall = recall_score(y_test, y_pred, average='weighted')
    f1 = f1_score(y_test, y_pred, average='weighted')
    conf_matrix = confusion_matrix(y_test, y_pred)
```

```
        print(f"\nMetrics for {model_name}:")
        print(f"Accuracy: {acc:.4f}")
        print(f"Precision: {precision:.4f}")
        print(f"Recall: {recall:.4f}")
        print(f"F1-Score: {f1:.4f}")
        print("Confusion Matrix:")
        print(conf_matrix)

        if y_prob is not None:
            y_bin = label_binarize(y_test, classes=np.unique(y_test))
            fpr, tpr, _ = roc_curve(y_bin.ravel(), y_prob.ravel())
            auc_score = auc(fpr, tpr)

            plt.figure(figsize=(6, 6))
            plt.plot(fpr, tpr, label=f'{model_name} (AUC = {auc_score:.4f})', color='blue')
            plt.plot([0, 1], [0, 1], 'k--', color='gray')
            plt.xlabel('False Positive Rate')
            plt.ylabel('True Positive Rate')
            plt.title(f'ROC Curve for {model_name}')
            plt.legend(loc='lower right')
            plt.show()

            return acc, precision, recall, f1, conf_matrix, auc_score

        return acc, precision, recall, f1, conf_matrix

    # Naive Bayes model
    nb_model = GaussianNB()
    nb_accuracies = cross_val_score(nb_model, X_train, y_train, cv=kfold, scoring='accuracy')
    print(f"\nNaive Bayes - Average K-Fold Accuracy: {np.mean(nb_accuracies):.4f}")
    nb_model.fit(X_train, y_train)
    y_pred_nb = nb_model.predict(X_test)
    y_prob_nb = nb_model.predict_proba(X_test)
    compute_metrics(y_test, y_pred_nb, "Naive Bayes", y_prob_nb)

    # Support Vector Machine model
    svm_model = SVC(kernel='rbf', probability=True, random_state=42)
    svm_accuracies = cross_val_score(svm_model, X_train, y_train, cv=kfold, scoring='accuracy')
    print(f"\nSVM - Average K-Fold Accuracy: {np.mean(svm_accuracies):.4f}")
    svm_model.fit(X_train, y_train)
    y_pred_svm = svm_model.predict(X_test)
    y_prob_svm = svm_model.predict_proba(X_test)
    compute_metrics(y_test, y_pred_svm, "Support Vector Machine", y_prob_svm)

    # Random Forest model
    rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
    rf_accuracies = cross_val_score(rf_model, X_train, y_train, cv=kfold, scoring='accuracy')
    print(f"\nRandom Forest - Average K-Fold Accuracy: {np.mean(rf_accuracies):.4f}")
    rf_model.fit(X_train, y_train)
    y_pred_rf = rf_model.predict(X_test)
    y_prob_rf = rf_model.predict_proba(X_test)
    compute_metrics(y_test, y_pred_rf, "Random Forest", y_prob_rf)
```

```
Naive Bayes - Average K-Fold Accuracy: 0.8226

Metrics for Naive Bayes:
Accuracy: 0.8111
Precision: 0.8480
Recall: 0.8111
F1-Score: 0.8151
Confusion Matrix:
[[33  0  0  0  1  1  0  0  1  0]
 [ 0 29  1  0  0  0  1  0  3  2]
 [ 0  3 19  0  0  0  0  0 13  0]
 [ 0  0  1 26  0  0  0  1  8  1]
 [ 0  2  0  0 26  0  1  6  1  0]
 [ 0  0  0  0  0 35  0  1  0  1]
 [ 0  1  0  0  0  0 35  0  0  0]
 [ 0  0  0  0  0  1  0 35  0  0]
 [ 0  4  1  0  0  0  0  1 29  0]
 [ 0  3  1  0  0  0  0  4  3 25]]
<ipython-input-4-c843449dcfcb>:50: UserWarning: color is redundantly defined by the 'color' keyword argument and the fmt string "
  plt.plot([0, 1], [0, 1], 'k--', color='gray')
```



```
SVM - Average K-Fold Accuracy: 0.9882

Metrics for Support Vector Machine:
Accuracy: 0.9917
Precision: 0.9920
Recall: 0.9917
F1-Score: 0.9917
Confusion Matrix:
[[36  0  0  0  0  0  0  0  0  0]
 [ 0 36  0  0  0  0  0  0  0  0]
 [ 0  0 35  0  0  0  0  0  0  0]
 [ 0  0  0 37  0  0  0  0  0  0]
 [ 0  0  0  0 36  0  0  0  0  0]
 [ 0  0  0  0  0 37  0  0  0  0]
 [ 0  0  0  0  0  0 36  0  0  0]
 [ 0  0  0  0  0  0  0 36  0  0]
 [ 0  2  0  0  0  0  0  0 33  0]
 [ 0  0  0  0  0  0  0  1  0 35]]
<ipython-input-4-c843449dcfcb>:50: UserWarning: color is redundantly defined by the 'color' keyword argument and the fmt string "
  plt.plot([0, 1], [0, 1], 'k--', color='gray')
```


ROC Curve for Support Vector Machine
Positive Rate