Instructions:

- You need to code in this jupyter notebook only.
- Download this notebook and import in your jupyter lab.
- You need to write a partial code for step 0 to step 8 mentioned with prefix ##
- Fill the blanks where it is instructed in comments.
- Leave other codes, structure as it is.
- Follow all the instructions commented in the cells.

Answer the questions given at the end of this notebook within your report.

Upload this jupyter notebook after completion with your partial code and the report in one file in PDF format. Your file name should be yourname_lab4.pdf

Also upload the resulting image showing all the selected points and boundary line between them after LDA analysis.

Your submission should contain the pdf file and the output plot. Upload it on the LMS before the due time.

In [1]:
```
import numpy as np  ## import numpy
import cv2  ## import opencv
import matplotlib ## import matplotlib
import matplotlib.pyplot as plt  ## import matplotlib pyplot
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA ## from
matplotlib.use('TkAgg')

##------------------------------------------------------
## Step 0: Install all other dependencies that occur at run time if  any module not
##------------------------------------------------------
```

In [2]:
```
Number_of_points = 20  ## Number of points you want select from each strip. Recomme

img = cv2.imread("Indian_Flag.jpg") ## Read the given image

def select_points(img, title):
    fig, ax = plt.subplots()
    #----------------------------------------
    ## step 1: Convert the img from BGR to RGB using cv2 and display it using cv2.i
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    ax.imshow(img_rgb)

    ## step 2: Put title of the image
    ax.set_title(title)
    ##----------------------------------------

    # Set the cursor style to a plus sign
    fig.canvas.manager.set_window_title('Select Points')
    cursor = matplotlib.widgets.Cursor(ax, useblit=True, color='red', linewidth=1)
    plt.show(block=False)  # Show the image without blocking
```

```
    k = 0
    points = [] ## Create here an empty list to store points

    while k < Number_of_points:
        xy = plt.ginput(1, timeout=0)  # Non-blocking input
        if len(xy) > 0:
            col, row = map(int, xy[0])  # Convert to integer
            ##-------------------------------------------
            ## Step 3: Collect RGB values at the clicked positions (col, row) and p
            ##-------------------------------------------

            k += 1
            points.append([row, col, img[row, col]])  # Store RGB values in empty l

            # Display colored dot on the image
            plt.scatter(col, row, c='black', marker='o', s=10)

            # Redraw the image to include the dot
            plt.draw()

    plt.close()  # Close the window after all points are collected
    return points ## Fill this blank
```

In [3]:
```
##-----------------------------------------------------------------
## Step4: fill the blanks for Selected points from saffron strip
pts_saffron = select_points(img , "Select from saffron strip ")
## Step5: fill the blanks for Selected points from white strip)
pts_white = select_points (img , "Select from white strip")
## Step6: fill the blanks for Selected points from green strip
pts_green = select_points(img , "Select from greeb strip")
##-----------------------------------------------------------------
```

In [4]:
```
# Convert RGB values to Lab color space
def rgb_to_lab(rgb):
    return cv2.cvtColor(np.uint8([[rgb]]), cv2.COLOR_RGB2Lab)[0][0]

saffron_lab = np.array([rgb_to_lab(rgb) for _, _, rgb in pts_saffron])
white_lab = np.array([rgb_to_lab(rgb) for _, _, rgb in pts_white])
green_lab = np.array([rgb_to_lab(rgb) for _, _, rgb in pts_green])

## Step7: Extract a* and b* components from Lab color space
a_features = np.hstack((saffron_lab[:, 1], white_lab[:, 1], green_lab[:, 1]))
b_features = np.hstack((saffron_lab[:, 2], white_lab[:, 2], green_lab[:, 2]))
```

In [5]:
```
# Map class labels to numeric values
class_mapping = {'Saffron': 0, 'White': 1, 'Green': 2}
y = np.array([class_mapping[label] for label in ['Saffron'] * Number_of_points + ['

plt.figure()
plt.scatter(a_features[:Number_of_points], b_features[:Number_of_points], c='b', ma
plt.scatter(a_features[Number_of_points:2*Number_of_points], b_features[Number_of_p
plt.scatter(a_features[2*Number_of_points:], b_features[2*Number_of_points:], c='r'
plt.legend(['Saffron', 'White', 'Green'], loc='best')
plt.xlabel('a* component (Green-Red)')  ## Provide x label
```

```
plt.ylabel('b* component (Blue-Yellow)') ## Provide y label
plt.title('Distribution of Flag Colors in a*b* Space') ## Provide title
plt.grid()
plt.show()

##------------------------------------------------------------
# Step 8: Perform LDA analysis using LinearDiscriminantAnalysis() and lda.fit()

X = np.column_stack((a_features , b_features))
lda = LDA()
lda.fit(X , y)

##------------------------------------------------------------
```

Out[5]:

LinearDiscriminantAnalysis

▶ Parameters

In [6]:

```
# Plot LDA boundaries
plt.figure()
plt.scatter(a_features[:Number_of_points], b_features[:Number_of_points], c='b', ma
plt.scatter(a_features[Number_of_points:2*Number_of_points], b_features[Number_of_p
plt.scatter(a_features[2*Number_of_points:], b_features[2*Number_of_points:], c='r'

plt.xlabel('a* component (Green to Red)')  ## Provide x label
plt.ylabel('b* component (Blue to Yellow)') ## Provide y label
plt.title('LDA boundaries (linear model) for Colors of the Indian Flag')

# Plot the decision boundaries
ax = plt.gca()
xlim = ax.get_xlim()
ylim = ax.get_ylim()

xx, yy = np.meshgrid(np.linspace(xlim[0], xlim[1], 100), np.linspace(ylim[0], ylim[
Z = lda.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

plt.contour(xx, yy, Z, colors='k', linewidths=2, linestyles='solid')
plt.legend(loc='best')
plt.grid()
plt.show()
```

Report:

Answer the following questions within your report:

1. What are the key assumptions underlying LDA, and how do these assumptions influence the model's performance?

2. What are the hyperparameters in LDA, and how do they affect the outcome of the model?

3. What methods can be used to assess an LDA model's effectiveness?

4. What are some common challenges or limitations associated with LDA, and how can they be addressed or mitigated?

5. What practical applications does this assignment have in real-world situations, and what benefits does it offer in those specific scenarios?

MLPR LAB 4 REPORT

1. What are the key assumptions underlying LDA, and how do these assumptions influence the model's performance?

LDA assumes that each color class follows a Gaussian (normal) distribution and that all classes share the same covariance matrix. In this lab, if your selected points for Saffron, White, and Green are tightly clustered and have similar spreads in the a^*b^* space, the LDA will produce highly accurate, clean linear boundaries.

2. What are the hyperparameters in LDA, and how do they affect the outcome of the model?

The main hyperparameters are the solver (e.g., 'svd' or 'lsqr') and **shrinkage**. In this lab, since we have more points than features, the default 'svd' solver works perfectly; however, if you had very few points, adding shrinkage would help stabilize the boundary lines by improving the covariance estimation.

3. What methods can be used to assess an LDA model's effectiveness?

You can assess effectiveness visually by checking if the black contour lines correctly separate the colored dots in your plot without misclassifying any points. You could use a confusion matrix to see if the model correctly predicts the strip color based on the a^* and b^* values.

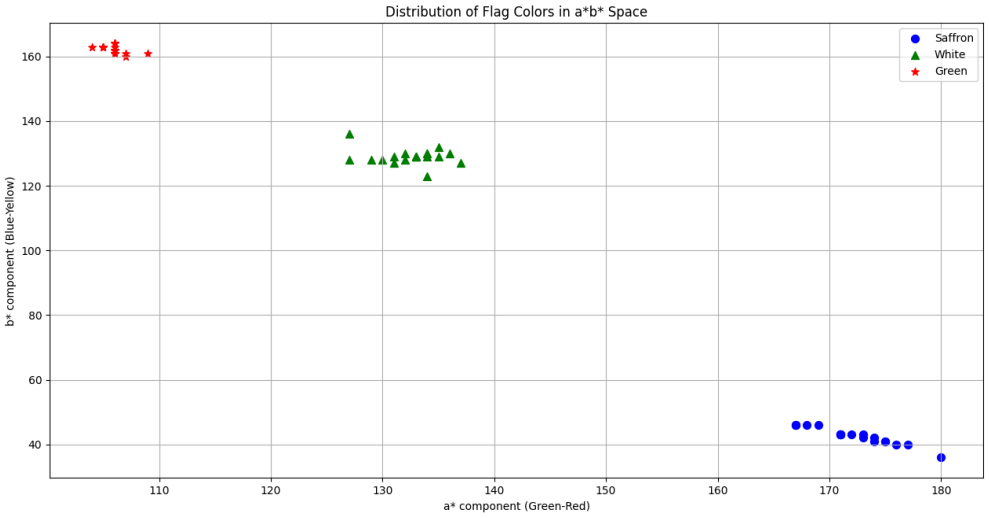
4. What are some common challenges or limitations associated with LDA, and how can they be addressed or mitigated?

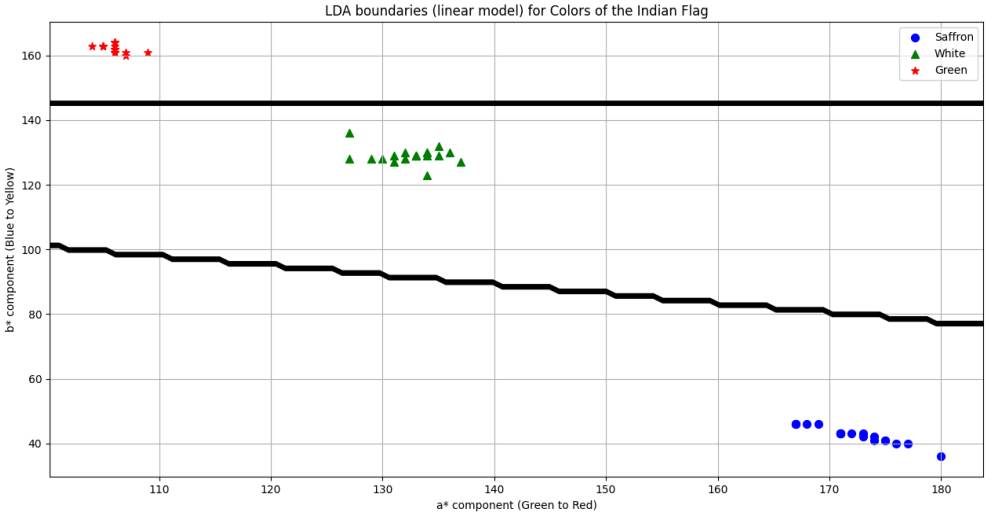
A major limitation is that LDA only creates linear boundaries; if the colors in an image overlapped significantly , a straight line might not separate them well. This effect is decreased in this lab by switching from RGB to Lab color space, which makes the classes more linearly separable .

5. What practical applications does this assignment have in real-world situations, and what benefits does it offer in those specific scenarios?

This logic is used in Spam Detection Disease Classification on Patients’ data . Using LDA provides a computationally "cheap" and fast way to automate color-based sorting or object detection compared to complex deep-learning models.

Outputs :


Distribution of Flag Colors in a*b* Space
Saffron
White
Green
b* component (Blue-Yellow)
a* component (Green-Red)
160
140
120
100
80
60
40
110
120
130
140
150
160
170
180


LDA boundaries (linear model) for Colors of the Indian Flag
Saffron
White
Green
b* component (Blue to Yellow)
a* component (Green to Red)
160
140
120
100
80
60
40
110
120
130
140
150
160
170
180