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 a cells.
- Upload this jupyter notebook after completion with your partial code.
- Also upload the resulting image showing all the selected points and boundary line between them after LDA analysis.

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

import matplotlib.pyplot as plt                                       ## import matplotlib pyplot

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

In [2]:
```
Number_of_points = 22  ## Number of points you want select from each strip. Recommended >= 20

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 plt.imshow
    #_______________________________________
    image_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)    # Convert the img from BGR to RGB
    ax.imshow(image_rgb)                                # Display image using matplotlib
    ax.axis('off')



    ## Step 2: Put title of the image
    ##_____________________________________
    ax.set_title(f'Select points on the Indian flag from {title} strip', fontsize = 16)
    #plt.show()
    ##---------------------------------------

    # 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 print it.
            ##-----------------------------------------------

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

            # 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, 'Saffron')
## Step5: fill the blanks for Selected points from white strip)
pts_white = select_points(img, 'White')
## Step6: fill the blanks for Selected points from green strip
pts_green = select_points(img, 'Green')
##-------------------------------------------------------------------
```

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

# Splits the LAB image into 3 channles L, a* and b* and extract values for a* and b* channels
a_saffron, b_saffron = saffron_lab[:, 1], saffron_lab[:, 2]
a_white, b_white = white_lab[:, 1], white_lab[:, 2]
a_green, b_green = green_lab[:, 1], green_lab[:, 2]

# Concatenate a* channels for saffron, white and green
a_features = np.hstack((a_saffron, a_white, a_green))

# Concatenate b* channels for saffron, white and green
b_features = np.hstack((b_saffron, b_white, b_green))
```

In [9]:
```
# 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 + ['White'] * Number_of_pc

plt.figure()
plt.scatter(a_features[:Number_of_points], b_features[:Number_of_points], c='b', marker='o', s=50, label=
plt.scatter(a_features[Number_of_points : 2 * Number_of_points], b_features[Number_of_points : 2 * Number
            c='g', marker='^', s=50, label='White')
plt.scatter(a_features[2 * Number_of_points:], b_features[2 * Number_of_points:], c='r', marker='*', s=50

plt.legend(['Saffron', 'White', 'Green'], loc='best')
plt.xlabel('a* feature axis')                    ## Provide x label
plt.ylabel('b* feature axis')                    ## Provide y label
plt.title('CIELAB Color Space - a* V/s b*')      ## Provide title
plt.grid()
plt.show()

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

X = np.c_[a_features, b_features]

lda = LinearDiscriminantAnalysis()
lda.fit(X, y)

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

Out[9]:
```
LinearDiscriminantAnalysis
LinearDiscriminantAnalysis()
```

In [8]:
```
# Plot LDA boundaries
plt.figure()
plt.scatter(a_features[:Number_of_points], b_features[:Number_of_points], c='b', marker='o', s=50, label=
plt.scatter(a_features[Number_of_points : 2 * Number_of_points], b_features[Number_of_points : 2 * Number
            c='g', marker='^', s=50, label='White')
plt.scatter(a_features[2 * Number_of_points:], b_features[2 * Number_of_points:], c='r', marker='*', s=50

plt.xlabel('a* feature axis')                    ## Provide x label
plt.ylabel('b* feature axis')                    ## 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[1], 100))
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()
```

In [ ]: