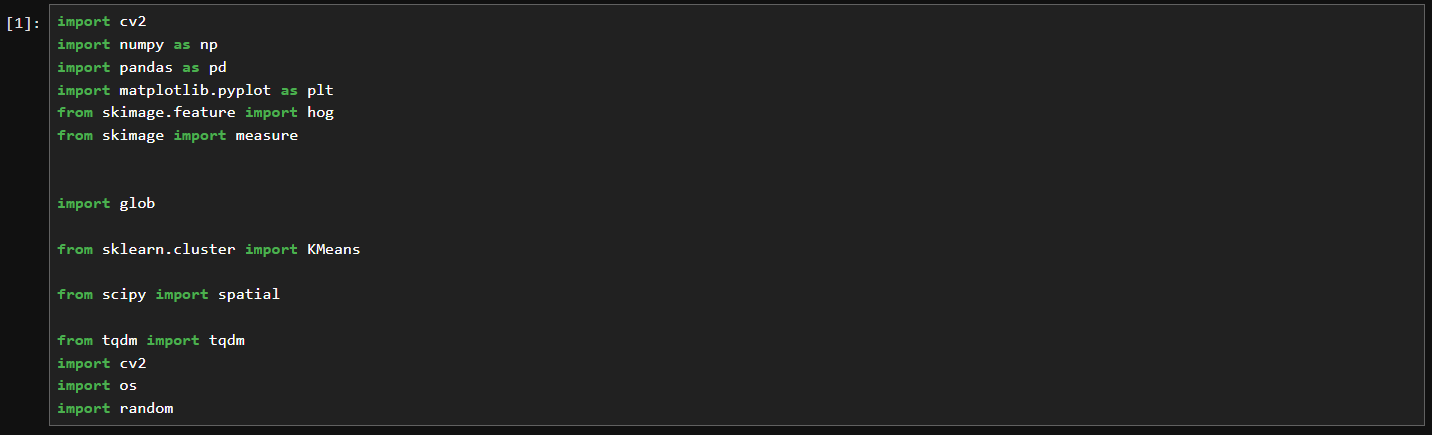
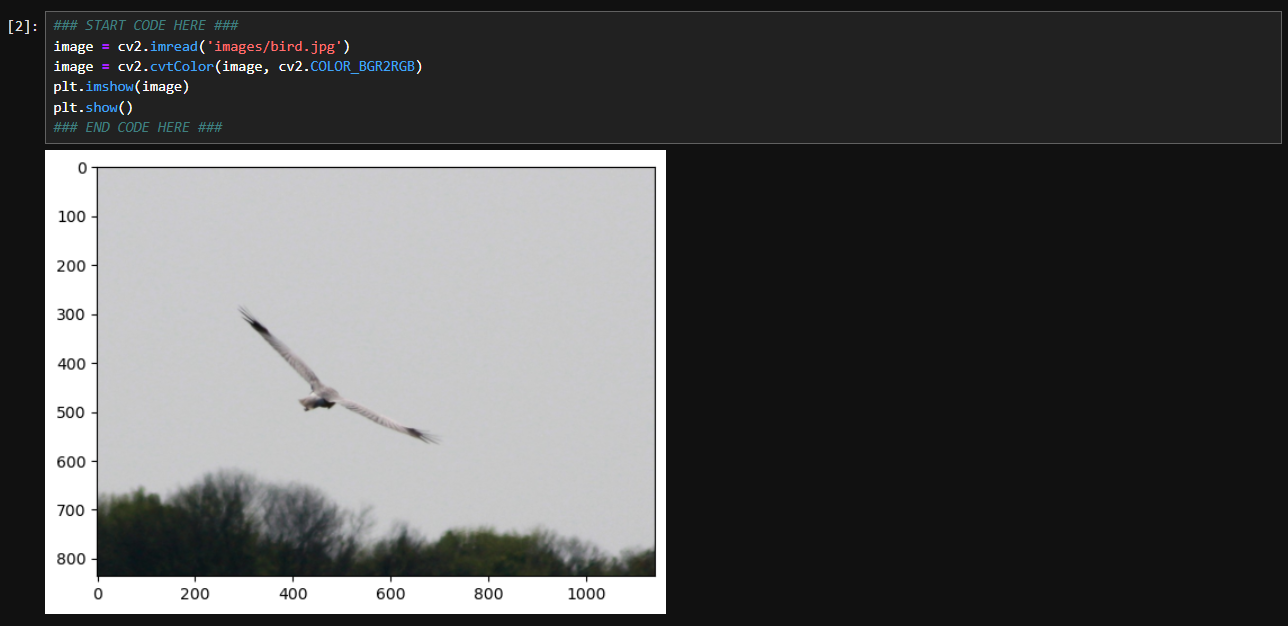
**Lab 4**

**Image Segmentation**



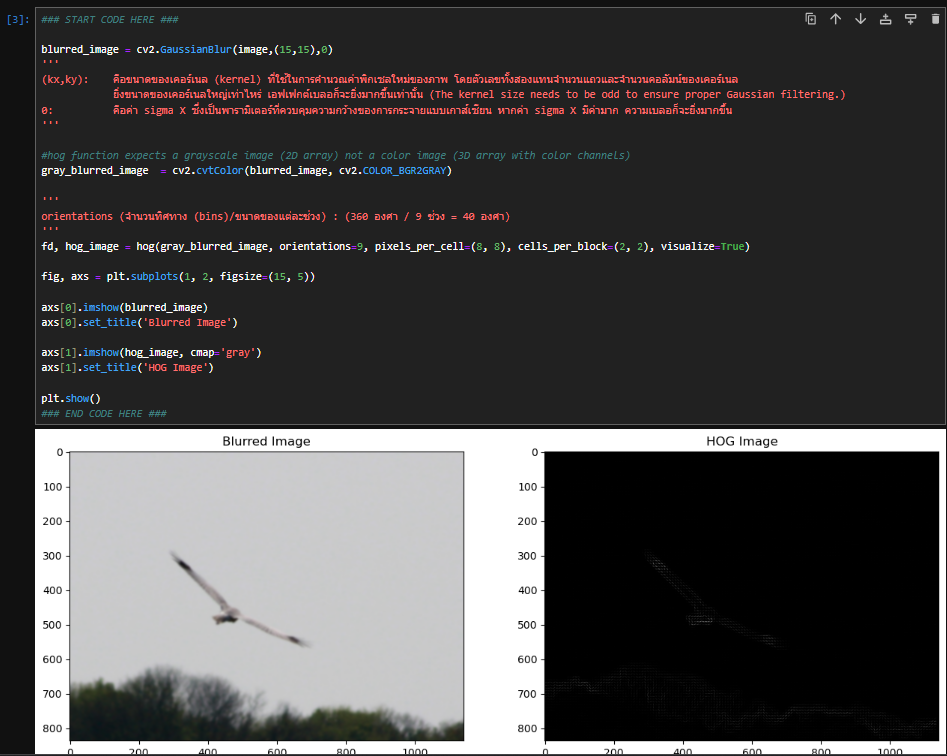
[1] จะเป็นการ Import Library ที่จำเป็นที่ต้องใช้ใน Lab 4 นี้



[2] - จะเป็นการโหลดรูปมาใช้งาน โดยใช้การ Read bird.jpg using cv2.imread

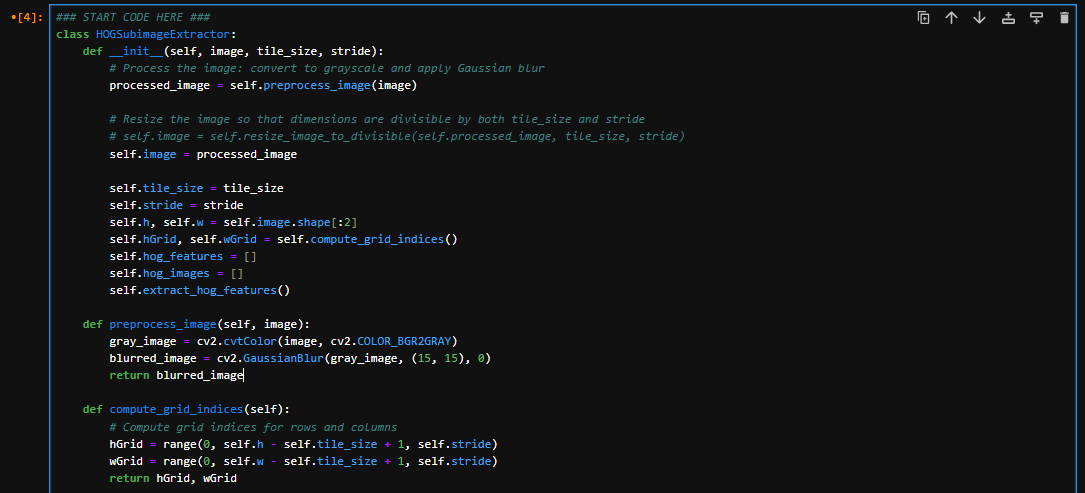
- Convert BGR to RGB using cv2.cvtColor

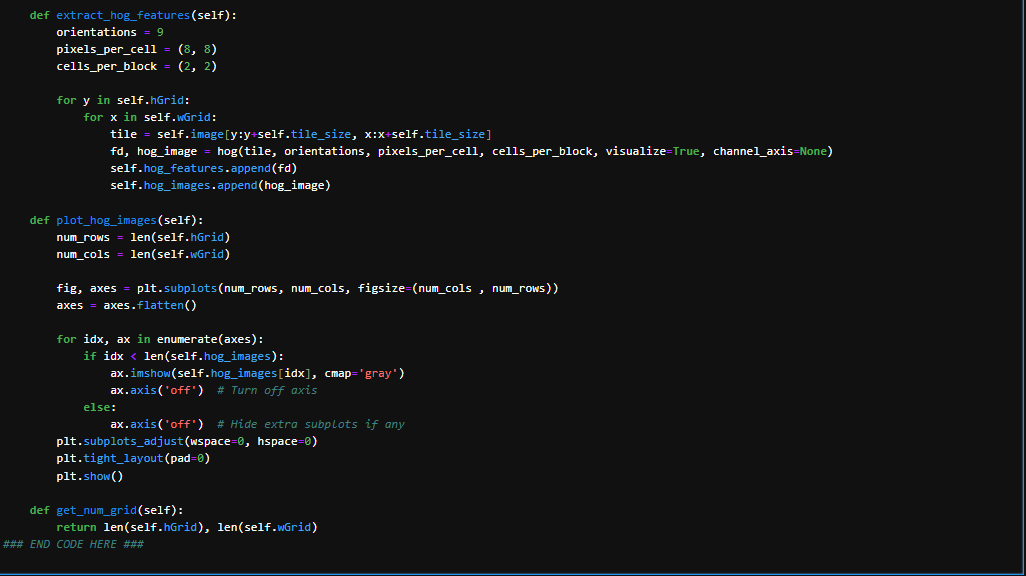
- และสุดท้ายเป็นการแสดงรูปภาพ show image using plt ที่ได้หลังจากการ convert



[3]

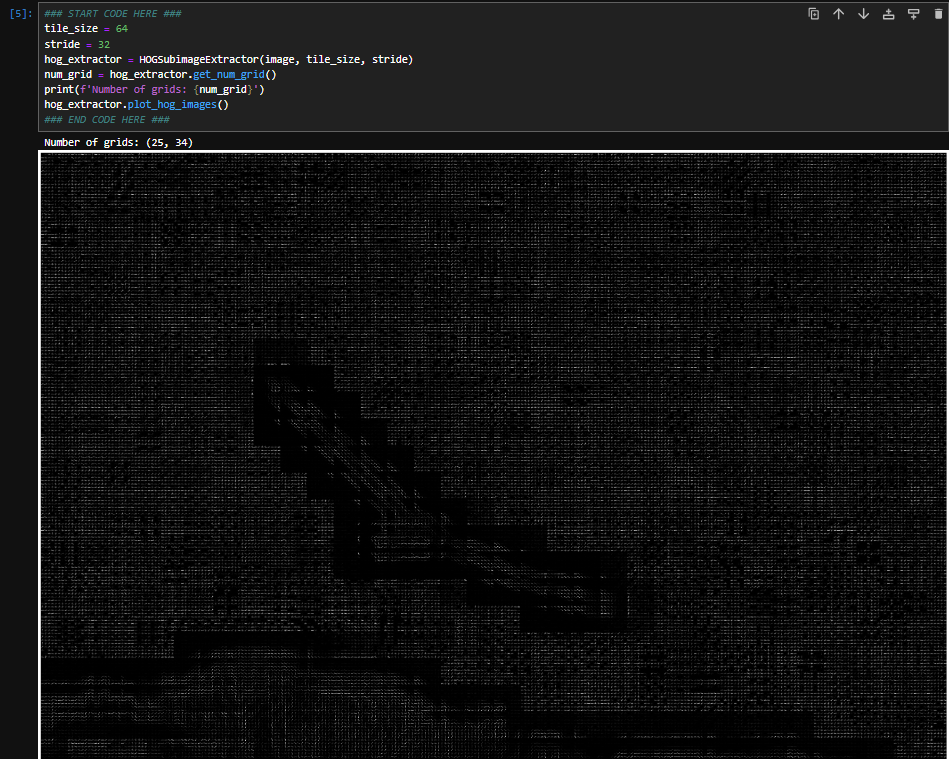
* blurred\_image = cv2.GaussianBlur(image, (15, 15), 0)
  + apply Gaussian blur to the input image
    - (kx,ky): คือขนาดของเคอร์เนล (kernel) ที่ใช้ในการคำนวณค่าพิกเซลใหม่ของภาพ โดยตัวเลขทั้งสองแทนจำนวนแถวและจำนวนคอลัมน์ของเคอร์เนล ยิ่งขนาดของเคอร์เนลใหญ่เท่าไหร่ เอฟเฟกต์เบลอก็จะยิ่งมากขึ้นเท่านั้น (The kernel size needs to be odd to ensure proper Gaussian filtering.)
    - 0: คือค่า sigma X ซึ่งเป็นพารามิเตอร์ที่ควบคุมความกว้างของการกระจายแบบเกาส์เซียน หากค่า sigma X มีค่ามาก ความเบลอก็จะยิ่งมากขึ้น
* gray\_blurred\_image = cv2.cvtColor(blurred\_image, cv2.COLOR\_BGR2GRAY)
  + เป็นการ convert blurred image from BGR to grayscale
    - โดยที่ HOG feature extraction จะทำงานกับ grayscale images, which are 2D arrays, rather than 3D color images.
* fd, hog\_image = hog(gray\_blurred\_image, orientations=9, pixels\_per\_cell=(8, 8), cells\_per\_block=(2, 2), visualize=True)
  + เป็น commard ให้ : compute HOG feature เพื่อสำหรับภาพ grayscale ที่ blurred
    - orientations=9 : (จำนวนทิศทาง (bins)/ขนาดของแต่ละช่วง) : (360 องศา / 9 ช่วง = 40 องศา)
    - pixels\_per\_cell=(8, 8) : size of cell that the histogram is computed.
    - cells\_per\_block=(2, 2) : The number of cells in each block used to normalize the histograms.
    - visualize=True : return image that visualizes the HOG features.
* โดยที่ Histogram of Oriented Gradients (HOG): เป็นเทคนิคที่ใช้ในการสกัดลักษณะของวัตถุในภาพ โดยอาศัยการวิเคราะห์การไล่ระดับสี (gradient) ของภาพ ซึ่งการไล่ระดับสีจะบอกถึงทิศทางและความเข้มของการเปลี่ยนแปลงของความสว่างในภาพ ซึ่งผลลัพธ์ที่ออกมาจาก command นี้คือ :
  + **fd (feature descriptor):** คือ เวกเตอร์ลักษณะที่ได้จากการประมวลผล HOG ซึ่งเป็นตัวแทนของภาพนั้นๆ โดยเวกเตอร์นี้จะเก็บข้อมูลเกี่ยวกับทิศทางและความเข้มของการไล่ระดับสีของภาพ
  + **hog\_image**: คือ ภาพที่แสดงผลการคำนวณ HOG โดยจะแสดงทิศทางของการไล่ระดับสีในแต่ละเซลล์ของภาพ ซึ่งจะช่วยให้เห็นภาพรวมของการเปลี่ยนแปลงของความสว่างในภาพ
* ซึ่งต่อมาก็จะเป็นการแสดงภาพต้นฉบับ คู่กับภาพที่ได้หลังจากการทำ hog เพื่อดูการเปลี่ยนแปลควมสว่างของภาพที่ได้





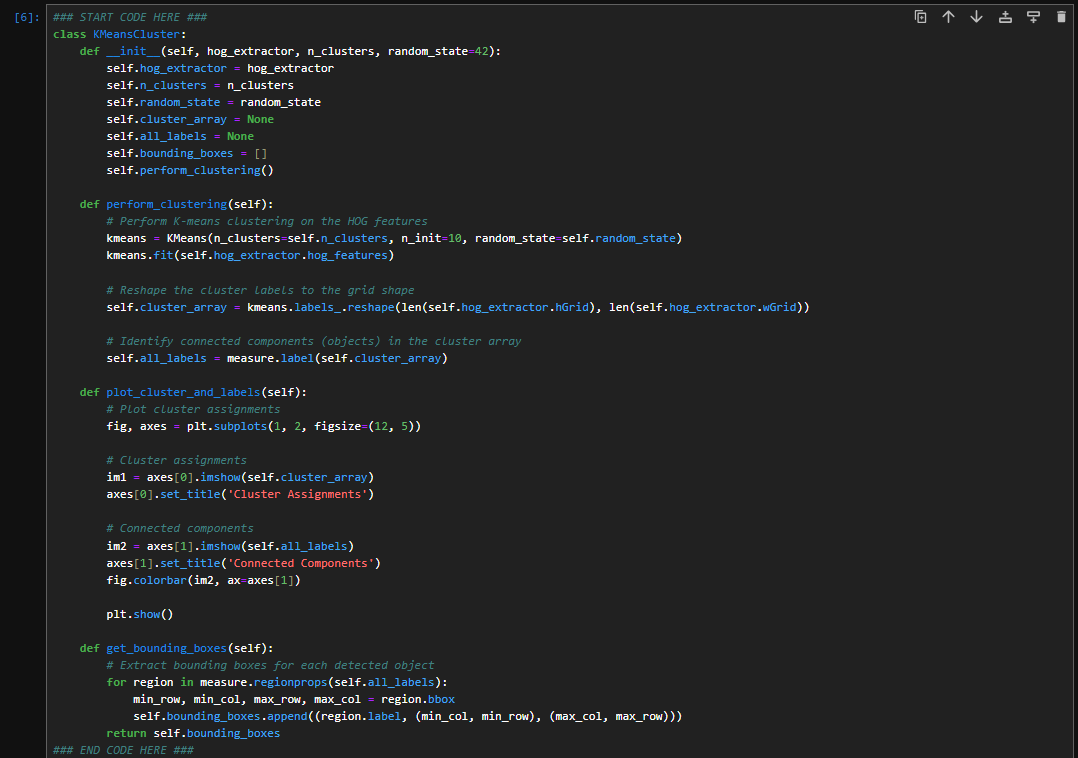
[4] ต่อมาก็จะเป็นการสร้าง **class HOGSubimageExtractor:** เพื่อเป็นการแบ่งภาพออกเป็นส่วนย่อยๆ และคำนวณ HOG สำหรับแต่ละส่วน ซึ่งจะนำไปประยุกต์กับการใช้กับงาน การตรวจจับวัตถุในภาพในลำดับถัดไป

* **def preprocess\_image(self, image):**
  + Converts the image to grayscale and applies a Gaussian blur. This prepares the image for HOG feature extraction. (ฟังก์ชันนี้จะทำการเตรียมภาพก่อนการคำนวณ HOG)
* **def compute\_grid\_indices(self):** 
  + Calculates the grid for tiling the image (ฟังก์ชันนี้คำนวณตำแหน่งที่จะแบ่งภาพ ซึ่งแนวตั้ง: hGrid และ แนวนอน: wGrid โดยเว้นระยะห่างตาม stride)
* **def extract\_hog\_features(self):**
  + ฟังก์ชันคำนวณ HOG สำหรับแต่ละส่วนของภาพ
  + Iterates over each tile in the image and extracts the HOG features
  + วน loop ไปตามตำแหน่งที่ได้จาก compute\_grid\_indices
  + ตัดส่วนของภาพ (subimage) ตามขนาด tile\_size
  + ใช้ฟังก์ชัน hog เพื่อคำนวณ HOG ของแต่ละส่วน
  + เก็บผลลัพธ์ซึ่งเป็น feature vector (fd) และ HOG image ไว้ในลิสต์ hog\_features และ hog\_images ตามลำดับ
* **def plot\_hog\_images(self):**
  + Visualizes the HOG images for all tiles in a grid layout
* **def get\_num\_grid(self):**
  + ฟังก์ชันนี้จะแสดงผล HOG image ในรูปแบบของ plot โดยจัดเรียง HOG image เป็นตารางตามจำนวน row และ column ที่แบ่งภาพ
  + Returns the number of tiles in the vertical and horizontal directions.



[5] แบ่งภาพออกเป็นส่วนย่อยขนาด 64x64 pixels โดยมีระยะห่างระหว่างส่วนย่อย 32 pixels คำนวณ HOG สำหรับแต่ละส่วนย่อย และแสดงภาพ HOG ทั้งหมดออกมา

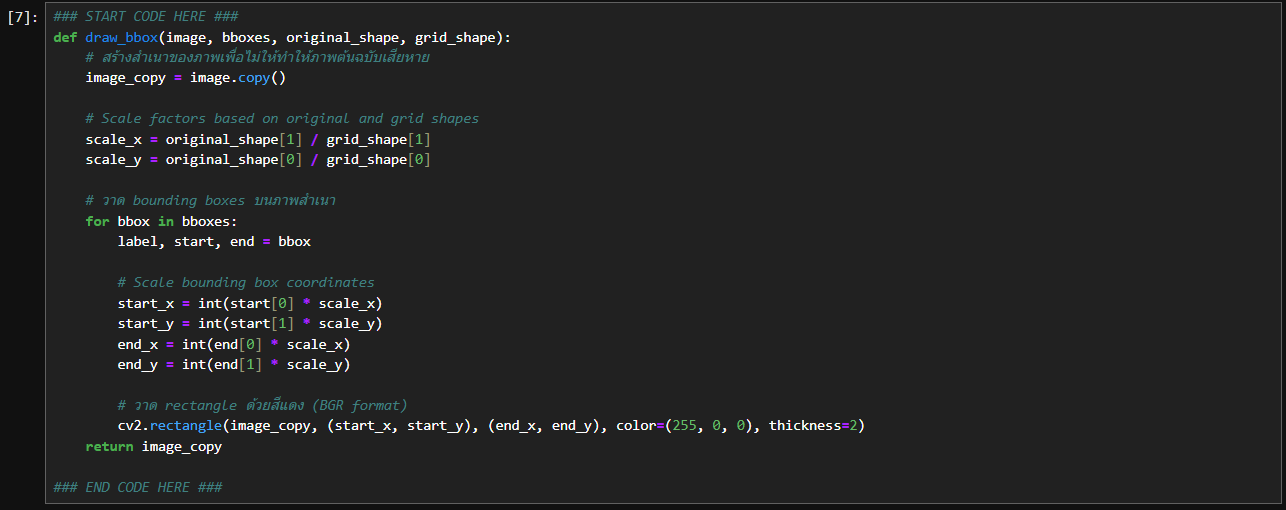
* Create HOGSubimageExtractor
* Get the Number of Grids : ขนาดที่ได้จากการแบ่งภาพด้วย tile และ stride บนภาพต้นฉบับ
* Plot HOG Image



[6] **class KMeansCluster:**

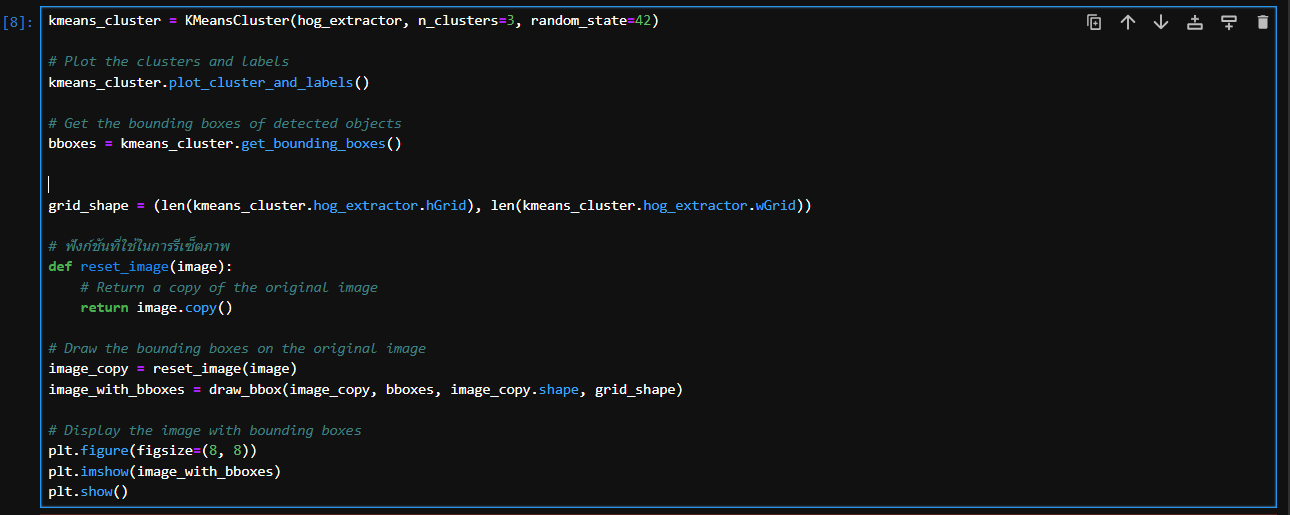
class: KMeansCluster ใช้สำหรับการแบ่งกลุ่มข้อมูล HOG features ด้วย K-means clustering เพื่อหา bounding boxes ของกลุ่มที่เชื่อมต่อกัน คล้ายการตรวจจับวัตถุในภาพ

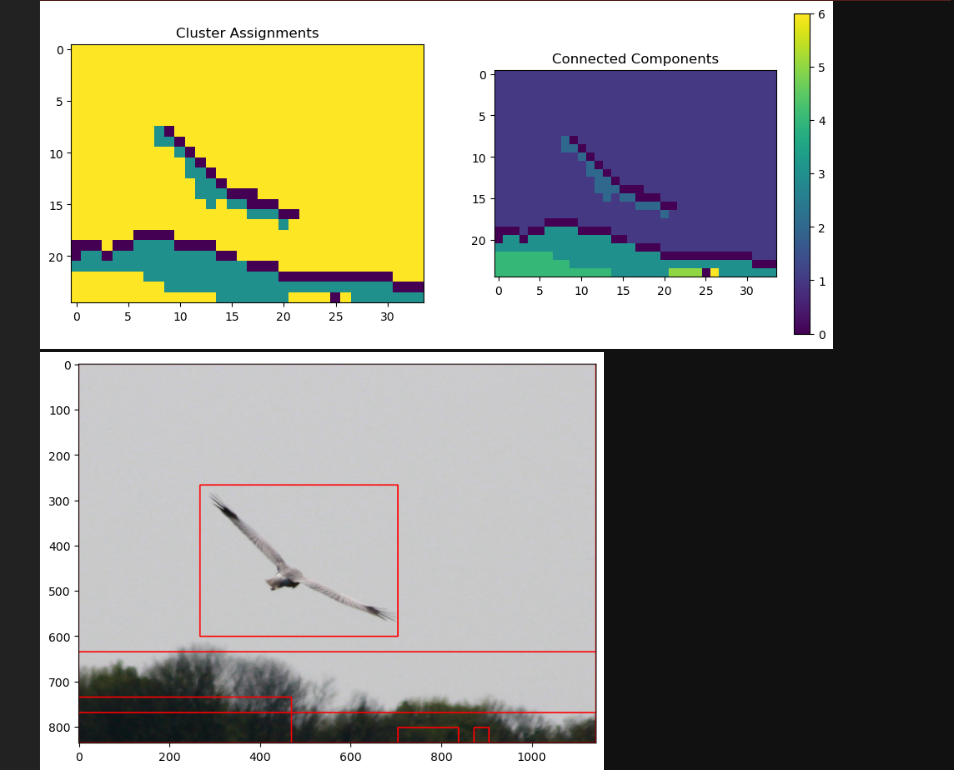
* **def perform\_clustering(self):**
  + สร้างวัตถุ kmeans - Initializes a K-means clustering model
  + Fit model to the HOG features extracted from the image tiles โดยแบ่งเป็น n\_clusters กลุ่ม
  + reshaped to match the grid layout of the image ซึ่งคือการจัดเรียงผลลัพธ์ (cluster labels) ให้ตรงกับโครงสร้างของการแบ่งภาพ (grid) เก็บไว้ใน self.cluster\_array
  + Identify connected components in the cluster array - เป็นการใช้ฟังก์ชัน label จากไลบรารี scikit-image เพื่อหากลุ่มของจุดที่เชื่อมต่อกันใน self.cluster\_array เก็บผลลัพธ์ไว้ใน self.all\_labels
* **def plot\_cluster\_and\_labels(self):**
  + visualizes the results of clustering ซึ่ง
    - Plot อันแรกจะแสดงผลลัพธ์การแบ่งกลุ่ม (cluster assignments)
    - Plot ที่สองขะแสดงผลกลุ่มของจุดที่เชื่อมต่อกัน (connected components)
* **def get\_bounding\_boxes(self):**
  + ใช้ฟังก์ชัน regionprops จากไลบรารี scikit-image เพื่อหา bounding boxes ของแต่ละกลุ่มที่เชื่อมต่อกันใน self.all\_labels
  + เก็บผลลัพธ์ (bounding box และ label) ไว้ในลิสต์ self.bounding\_boxes
  + Extracts the bounding boxes for each detected object in the image.
    - Returns the bounding box of the region as a tuple



[7] **def draw\_bbox(image, bboxes, original\_shape, grid\_shape):**

* Creating a Copy of the Image
* Calculating Scale Factors
  + scales the bounding box coordinates from the grid (tile) size to the original image size (คำนวณค่า scale factor เพื่อปรับขนาดของ bounding box ให้ตรงกับขนาดของภาพเดิม เนื่องจาก bounding box ที่ได้จากการทำ clustering จะอยู่ในระบบพิกัดของ grid ไม่ใช่พิกัดของภาพเดิม)
* Drawing Bounding Boxes
* Returning the Modified Image



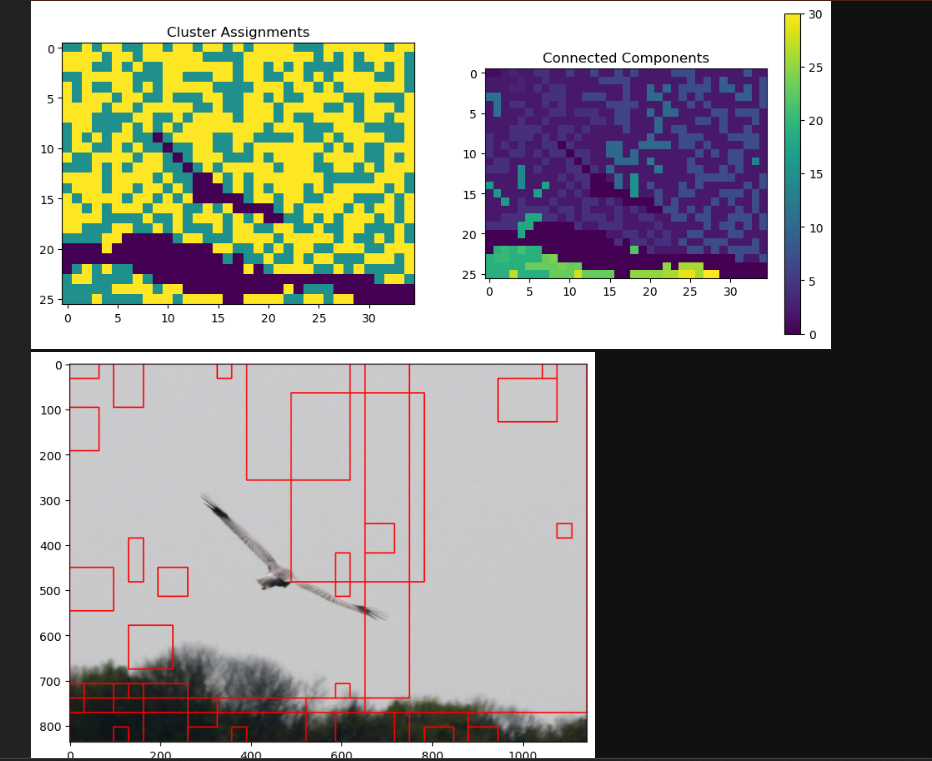


[8] performs K-means clustering on the HOG features extracted from an image, identifies distinct objects within the image, and then draws bounding boxes around these objects

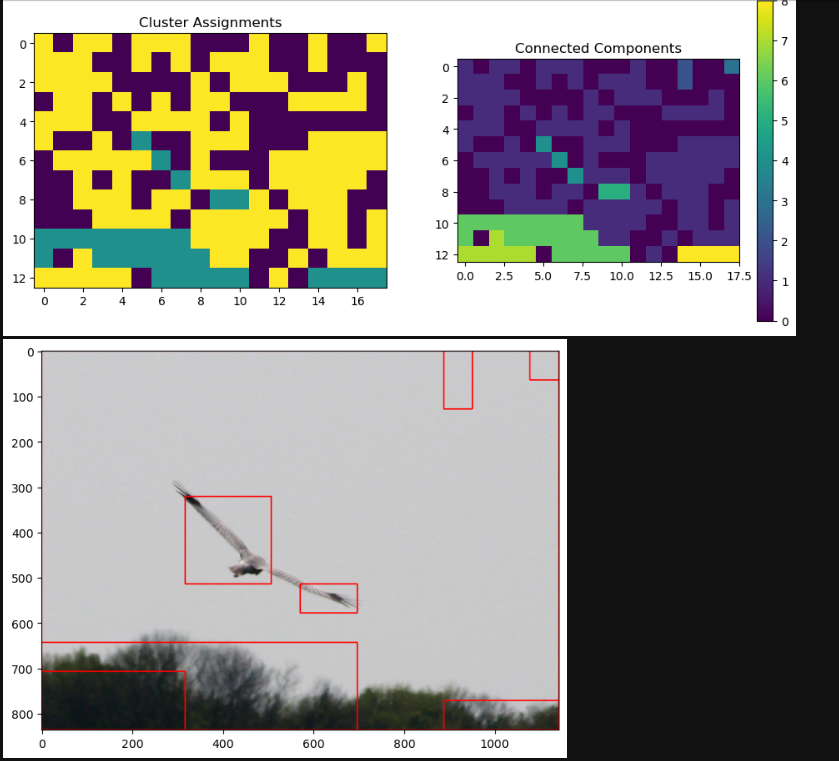
**Experiment**

* Do the experiment to identify 3 best parameters that produce the perfectly fitting bounding box of the object of interest in the image.

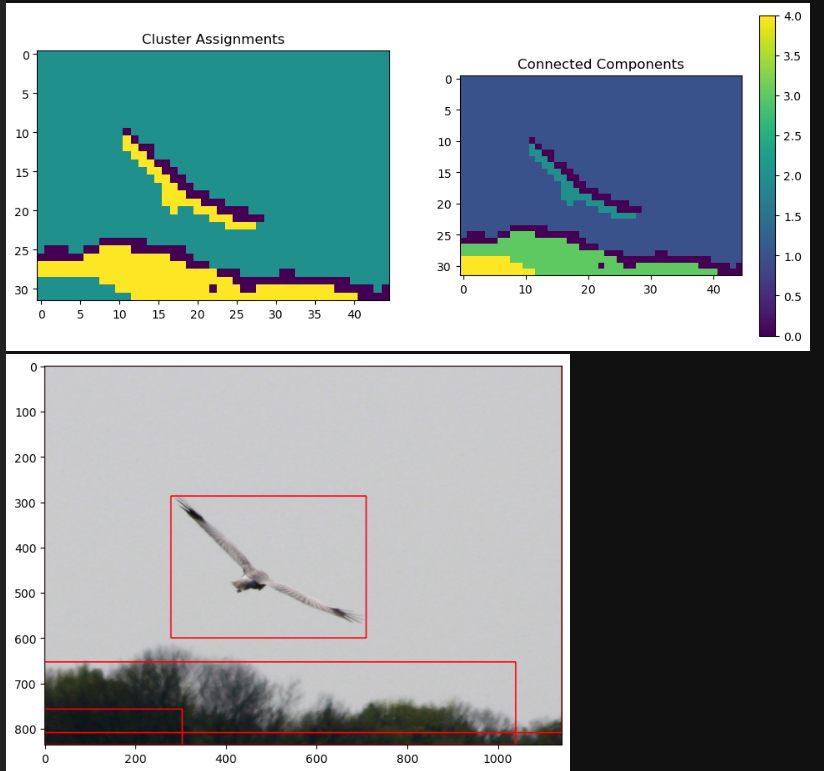
1. **tile\_size = 32, stride = 32**



1. **tile\_size = 32, stride = 64**



1. **tile\_size = 72, stride = 24**



โดยโค้ดทั้ง 3 ส่วนการทดลแงก็จะมีส่วนที่คล้ายกันยกเว้นแค่ **tile\_size** และ **stride** แต่มีใจความสำคัญดังนี้:

**1. การเตรียมข้อมูล:**

- กำหนดขนาดของส่วนย่อย (tile\_size) และระยะห่างระหว่างส่วนย่อย (stride)

- สร้างวัตถุ HOGSubimageExtractor

- คำนวณ Number of grids ที่ได้จาก tile\_size และ stride

- แสดงผล HOG images

- hog\_extractor.plot\_hog\_images()

**2. การแบ่งกลุ่มข้อมูล:**

- สร้างวัตถุ KMeansCluster

- แสดงผลการแบ่งกลุ่ม

- kmeans\_cluster.plot\_cluster\_and\_labels()

**3. การวาด bounding box:**

- ดึงข้อมูล bounding boxes

- bboxes = kmeans\_cluster.get\_bounding\_boxes()

- คำนวณขนาดของ grid (grid\_shape)

- สร้างสำเนาของภาพต้นฉบับ (image\_copy) เพื่อจะได้ไม่กระทบกับภาพต้นฉบับในการวาดสี่เหลี่ยมกรอบทับภาพนั้นๆไป

- เรียกฟังก์ชัน draw\_bbox เพื่อวาด bounding box ลงบนสำเนาของภาพ

- แสดงผลภาพต้นฉบับที่มี bounding box