

TASK 4 - DETECTION OF OBJECTS IN AN IMAGE USING COMPUTER VISION SDK

B PRIYANKAA

s224207694

priyayj2016@gmail.com

TASK OVERVIEW:

- Need to develop a model to detect different objects in images with the coordinates. It can be any image with different objects.
- Need to use the computer vision SDK to detect different objects on Azure.

TARGET GRADE : PASS(P)

What is Computer vision?

Computer vision means the usage of machines in order to understand both images and videos. It is a subfield to both Artificial Intelligence and Machine Learning as it uses both specialised methods and learning algorithms in order to have a better understanding of both the images and videos input files.

There are 8 capabilities of Computer Vision, They are:

1. Image Description: Analyse an image, evaluate the objects in the image and give a description of the image.
2. Feature Tagging: Based on objects that are recognised, tags are applied.
3. Image Categorization: Based on the images, that image is put into a category. For Example: Image of a cricket groud is put into **Sports** category.
4. Image Type Detection: This helps to identify if the image is a clip art or line drawing.
5. Color Scheme Detection: Find the dominant color in the background.
6. Content Moderation: To detect if there is any adult content in the image.
7. Object Detection: Find the common objects and return the coordinates of the bounding box.

8. Brand Detection: To find if there is a presence of any company logos in the image present.

Task Objective:

To develop a model using computer vision SDK to detect different objects in images with the coordinates. It can be any image with different objects.

1. IMPORTING THE NECESSARY LIBRARIES:

```
In [1]: # Importing the images using **IPython.display** Library modules.
```

```
from IPython.display import Image as img
```

```
In [2]: # CV Library - a package of modules that are designed to solve CV Problems.
```

```
# !pip install --upgrade azure-cognitiveservices-vision-computervision
```

```
# Pillow library: used for image processing.
```

```
#!pip install pillow
```

```
# dotenv library: This Library is installed in order to give the credentials needed to  
#!pip install python-dotenv
```

```
In [3]: # To run the Azure Computer Vision Service:
```

```
from azure.cognitiveservices.vision.computervision import ComputerVisionClient
```

```
# To extract specific features from the image:
```

```
from azure.cognitiveservices.vision.computervision.models import VisualFeatureTypes
```

```
# To authenticate the client:
```

```
from msrest.authentication import CognitiveServicesCredentials
```

```
# To draw bounding boxes in images:
```

```
from PIL import Image, ImageDraw  
import matplotlib.pyplot as plt
```

```
# To read the secret keys for Authentication:
```

```
import os  
from dotenv import load_dotenv
```

2. Creating a resource group on Azure portal:

1. Log in to the Azure portal using the credentials.

2. Here is the screenshot of the front page of the Azure portal. Azure for Students is the subscription with 100 credits developed by the university for the students to create the resource groups, workspace, develop and deploy the model and perform all the machine learning activities.

In [5]: `img('C:\\\\Users\\\\Priyankaa_B\\\\TASK_4\\\\AP1.jpg')`

Out[5]:

The screenshot shows the Microsoft Azure portal interface. At the top, there's a navigation bar with the 'Microsoft Azure' logo, a search bar, and various account and settings icons. Below the bar, the 'Azure services' section features a grid of icons for 'Create a resource', 'Resource groups', 'Virtual machines', 'Microsoft Entra ID', 'Subscriptions', 'Security', 'All resources', 'Quickstart Center', 'App Services', and 'More services'. The 'Virtual machines' icon is highlighted with a white border. Under the 'Resources' section, there are tabs for 'Recent' and 'Favorite'. A table lists three resources: 'my-first-cv-resource' (Computer vision, 4 hours ago), 'my-first-resource-group' (Resource group, 4 hours ago), and 'Azure for Students' (Subscription, 4 hours ago). At the bottom, there's a 'Navigate' section with links for 'Subscriptions', 'Resource groups', 'All resources', and 'Dashboard'.

1. Click on the Resource-group icon available as seen in the below image.

2. The resource group name is given as "my-first-resource-group" and the Region is set to "Central-India". Now "Review+Create" must be clicked.

In [6]: `img('C:\\\\Users\\\\Priyankaa_B\\\\TASK_4\\\\RG1.jpg')`

Out[6]:

Create a resource group ...

The screenshot shows the 'Create a resource group' wizard. The 'Basics' tab is selected. It includes fields for 'Subscription' (set to 'Azure for Students') and 'Resource group' (set to 'my-first-resource-group'). In the 'Resource details' section, the 'Region' is set to '(Asia Pacific) Central India'. At the bottom, there are buttons for 'Review + create', '< Previous', and 'Next : Tags >'.

1. So "Create" Button is clicked in order to choose "Computer Vision" from the MarketPlace.

```
In [7]: img('C:\\\\Users\\\\Priyankaa_B\\\\TASK_4\\\\RG2.jpg')
```

Out[7]:

The screenshot shows the Microsoft Azure portal interface. The left sidebar is titled 'my-first-resource-group' and includes sections for Overview, Activity log, Access control (IAM), Tags, Resource visualizer, Events, Settings (Deployments, Security, Deployment stacks, Policies, Properties, Locks), Cost Management (Cost analysis, Cost alerts (preview), Budgets), and Metrics. The main content area is titled 'Essentials' and has tabs for 'Resources' and 'Recommendations'. A search bar at the top says 'Search resources, services, and docs (G+/-)'. Below it, there are filters for 'Type equals all' and 'Location equals all'. The results table has columns for 'Name', 'Type', and 'Location'. A large central message says 'No resources match your filters' with a link to 'Try changing or clearing your filters.' Buttons for 'Create resources' and 'Clear filters' are also present.

1. "Computer Vision" option is chosen and create option is clicked.

```
In [8]: img('C:\\\\Users\\\\Priyankaa_B\\\\TASK_4\\\\CV1.jpg')
```

Out[8]:

The screenshot shows the Microsoft Azure Marketplace. The left sidebar has sections for Get Started, Service Providers, Management (Private Marketplace, Private Offer Management), My Marketplace (Favorites, My solutions, Recently created, Private plans), and Categories (AI + Machine Learning (77), Analytics (35), IT & Management Tools (15)). The main search bar shows 'computer vision' with filters for Pricing: All, Operating System: All, Publisher Type: All, Product Type: All, and Publisher name: All. Below the search bar, a message says 'New! Get AI-generated suggestions for your search. Ask AI to suggest products, articles, and solutions for what you need.' The results section shows 1 to 20 of 106 results for 'computer vision'. It includes cards for 'Computer Vision' (Microsoft, Azure Service, An AI service that analyzes content in images, Create button), 'Product Recognition Using Computer Vision' (Verofax Limited, SaaS, Real-time marketing and operational insights by turning products interactive, Starts at \$60,209.227/1 year, Subscribe button), 'Roboflow: End-to-end Computer Vision Platform' (Roboflow, Inc., SaaS, Simplify data collection & annotation, model training & deployment with our dev tools & APIs, Starts at \$35,909.329/1 year, Subscribe button), 'Reward Leakage Prevention Using Computer Vision' (Verofax Limited, SaaS, Verify reward claims with computer vision, Starts at \$42,967.851/1 year, Subscribe button), and 'Custom Vision' (Microsoft, Azure Service, An AI service and end-to-end platform for applying computer vision to your specific scenario, Create button). A 'Tile view' link is also visible.

1. Once **Create** button is clicked, then a name for CV workspace with a name "my-first-cv-resource" is given and the Region is set to "Central India".

```
In [9]: img('C:\\\\Users\\\\Priyankaa_B\\\\TASK_4\\\\CV2.jpg')
```

Out[9]:

The screenshot shows the 'Create Computer Vision' wizard in progress. Step 2, 'Instance Details', is displayed. The 'Region' dropdown is set to 'Central India'. The 'Name' field contains 'my-first-cv-resource'. A note at the top states: 'Azure AI services resource creation requires subscription registration, we detected that your selected subscription did not register Cognitive services resource type before, we will help you to register Cognitive services resource type when you select a subscription in subscription dropdown. Click to learn more how to check registration state for your selected subscription.' Below this, there's a 'Pricing tier' dropdown set to 'Free F0 (20 Calls per minute, 5K Calls per month)'. A link 'View full pricing details' is available. Under 'Responsible AI Notice', it says: 'Microsoft provides technical documentation regarding the appropriate operation applicable to this Azure AI service that is made available by Microsoft. Customer acknowledges and agrees that they have reviewed this documentation and will use this service in accordance with it. This Azure AI services is intended to process Customer Data that includes Biometric Data (as may be further described in product documentation) that Customer may incorporate into its own systems used for personal identification or other purposes. Customer acknowledges and agrees that it is responsible for complying with the Biometric Data obligations contained in the Online Services DPA.' At the bottom are 'Previous', 'Next', 'Review + create' (which is highlighted in blue), and 'Give feedback'.

1. The Pricing Tier is set to Free F0(20 calls per minute).
2. Once "Review+Create" is clicked and then "Create" is clicked, the deployment process begins and few minutes later, the process is over.

In [10]:

```
img('C:\\\\Users\\\\Priyankaa_B\\\\TASK_4\\\\CV3.jpg')
```

Out[10]:

The screenshot shows the 'Create Computer Vision' wizard in progress. Step 3, 'Review + Create', is displayed. It lists the configuration: 'Subscription' (Azure for Students), 'Resource group' (my-first-resource-group), 'Region' (Central India), 'Name' (my-first-cv-resource), and 'Pricing tier' (Free F0 (20 Calls per minute, 5K Calls per month)). Below this, under 'TERMS', it says: 'By clicking "Create", I (a) agree to the legal terms and privacy statement(s) associated with the Marketplace offering(s) listed above; (b) authorize Microsoft to bill my current payment method for the fees associated with the offering(s), with the same billing frequency as my Azure subscription; and (c) agree that Microsoft may share my contact, usage and transactional information with the provider(s) of the offering(s) for support, billing and other transactional activities. Microsoft does not provide rights for third-party offerings. See the [Azure Marketplace Terms](#) for additional details.' Under 'Basics', the same configuration is shown again. Under 'Network', 'Type' is listed as 'All networks, including the internet, can access this resource.'. Under 'Identity', 'Identity type' is listed as 'None'. At the bottom are 'Previous', 'Next', 'Create' (which is highlighted in blue), and 'Give feedback'.

In [11]:

```
img('C:\\\\Users\\\\Priyankaa_B\\\\TASK_4\\\\CV4.jpg')
```

Out[11]:

The screenshot shows the Microsoft Azure Deployment Overview page for a deployment named "Microsoft.CognitiveServicesComputerVision-20240406120336". The status is "Your deployment is complete". Deployment details include a start time of 4/6/2024, 12:09:22 PM, a Correlation ID of f539d4f4-b9f1-4bd1-bbcf-0a5ba41f81cd, and a Resource group of "my-first-resource-group". There are links for "Deployment details" and "Next steps", and a "Go to resource" button. On the right, there are promotional cards for "Cost management", "Microsoft Defender for Cloud", and "Free Microsoft tutorials".

1. Inside the "my-first-cv-resource", a key and endpoint is made after deployment.

In [12]:

```
img('C:\\\\Users\\\\Priyankaa_B\\\\TASK_4\\\\CV-Key-and-endpoint.jpg')
```

Out[12]:

The screenshot shows the Microsoft Azure Computer Vision resource overview for "my-first-cv-resource". It displays a "Keys and endpoint" section with two keys (KEY 1 and KEY 2) and their corresponding endpoints. A red arrow points to the endpoint URL "https://my-first-cv-resource.cognitiveservices.azure.com/". The left sidebar shows navigation options like Overview, Activity log, Access control (IAM), Tags, and Diagnose and solve problems.

1. Those subscription_key (Key1) is copied and endpoint is copied onto the notepad for future use.

In [13]:

```
img('C:\\\\Users\\\\Priyankaa_B\\\\TASK_4\\\\CV-Key-and-endpoint-2.jpg')
```

Out[13]:

File Edit View

```
subscription_key = '0b614e5025c44bf6b85644f178b4e523'
endpoint = 'https://my-first-cv-resource.cognitiveservices.azure.com/'
```

3. Creating the subscription_key and endpoint variables:

```
In [14]: subscription_key = os.environ.get("0b614e5025c44bf6b85644f178b4e523")
endpoint = os.environ.get("https://my-first-cv-resource.cognitiveservices.azure.com/")
```

A computervision client is made with the help of endpoint and subscription_key.

```
In [15]: computervision_client = ComputerVisionClient("https://my-first-cv-resource.cognitivese
```

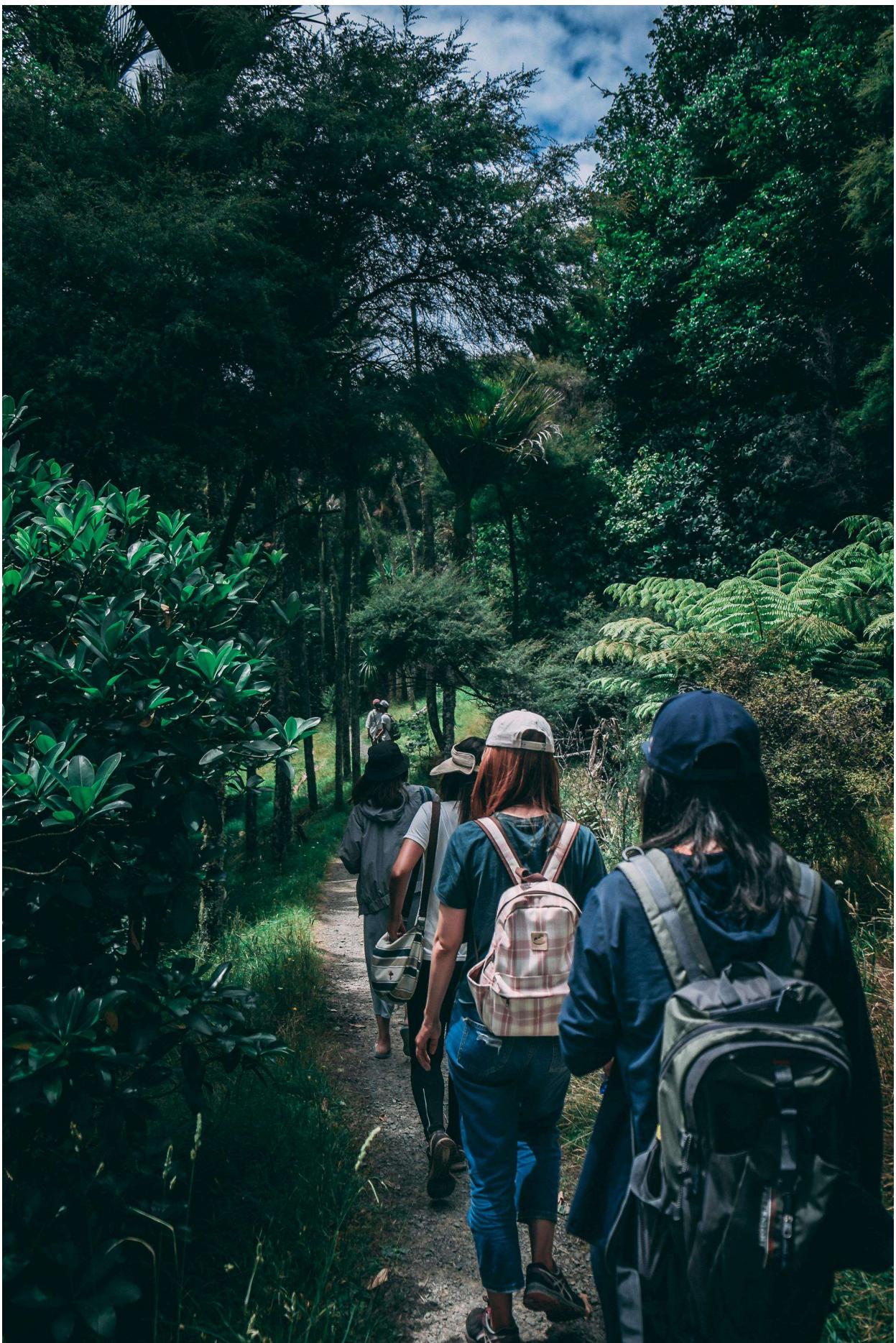
```
In [16]: # Features like description, tags, categories, brands, objects, adult to be retrieved:
features = [VisualFeatureTypes.description,
            VisualFeatureTypes.tags,
            VisualFeatureTypes.categories,
            VisualFeatureTypes.brands,
            VisualFeatureTypes.objects,
            VisualFeatureTypes.adult]
```

4. First Image: An image of a group of people going on a trekking is given as a first input.

```
In [18]: # Uploading Image to be analysed
image_file = 'C:\\\\Users\\\\Priyankaa B\\\\TASK_4\\\\TASK_4_INPUTS\\\\Trek.jpg'

# Viewing the image file
img(image_file)
```

Out[18]:



5. Reading the image as image data in order to retrieve the features like description, tags, categories, brands, objects, adult.

- The image is opened and mode is set as "rb" in order to read the image as image data and features are retrieved using the computervision_client.analyze_image_in_stream() function and it is stored into analysis variable.

```
In [19]: # Get image analysis
with open(image_file, mode="rb") as image_data:
    # call the API.
    # The first argument in the function is the image to be analysed. The second argument
    # We will run the analysis once per image. Then, we will display each feature in a
analysis = computervision_client.analyze_image_in_stream(image_data ,features)
```

Here `computervision_client()` is the API called with local image and features.

6. Image_Description:

- Based on reading the image and features collected, a description is made.
- For Example, this is an image of "**a group of people walking on a path through a forest**".

```
In [20]: # Get image description
if (len(analysis.description.captions) > 0):
    print("Describing the image: ")
    for caption in analysis.description.captions:
        print("Description: '{}' (confidence: {:.2f}%)".format(caption.text, caption.confidence))
else:
    print('No description found.')
```

```
Describing the image:
Description: 'a group of people walking on a path through a forest' (confidence: 49.92%)
```

7. Image_Tagging:

- Based on the features extracted from the image, there are several objects identified by this `analysis.tags` statement.

```
In [21]: # Get image tags
if (len(analysis.tags) > 0):
    print("These are the tags associated with the image:")
    for tag in analysis.tags:
        print(" - '{}' (confidence: {:.2f}%)".format(tag.name, tag.confidence * 100))
else:
    print("No image tags found")
```

These are the tags associated with the image:

```
- 'outdoor' (confidence: 99.89%)
- 'tree' (confidence: 98.86%)
- 'clothing' (confidence: 98.72%)
- 'hiking' (confidence: 98.31%)
- 'plant' (confidence: 96.66%)
- 'sky' (confidence: 95.23%)
- 'forest' (confidence: 95.13%)
- 'person' (confidence: 94.97%)
- 'backpack' (confidence: 89.37%)
- 'jungle' (confidence: 88.54%)
- 'people' (confidence: 85.07%)
- 'man' (confidence: 84.59%)
- 'nature' (confidence: 79.54%)
- 'grass' (confidence: 78.12%)
```

Inference Made:

- Here tags identified from the image are: Outdoor, tree, clothing, hiking, plant, sky, forest, person, backpack, jungle, people, man, nature, grass.

Here **confidence** means the degree of certainty of a Computer Vision Algorithm in its predictions.

- From the image, the algorithm has confidence of predicting girls as 'people' - 85.07%.

8. Image_Category:

- Based on the image, it is segregated as indoor or outdoor.
- With the help of **analysis.categories** feature obtained from the image_data, the confidence is also printed.
- Here since its a Trekking image, it is found to be outdoor category.

```
In [22]: # Get image categories
if (len(analysis.categories) > 0):
    print("Categories:")
    landmarks = []
    for category in analysis.categories:
        # Print the category
        print(" - '{}' (confidence: {:.2f}%)".format(category.name, category.score * 100))
        if category.detail:
            # Get Landmarks in this category
            if category.detail.landmarks:
                for landmark in category.detail.landmarks:
                    if landmark not in landmarks:
                        landmarks.append(landmark)

    # If there were Landmarks, list them
    if len(landmarks) > 0:
        print("Landmarks:")
        for landmark in landmarks:
```

```
        print(" -'{}' (confidence: {:.2f}%)".format(landmark.name, landmark.confidence))
    else:
        print("Unable to find image categories.")
```

```
Categories:
- 'outdoor_' (confidence: 0.78%)
```

9. Object_Detection:

- With the help of **analysis.objects** feature obtained from the image_data, the confidence is also printed.
- Image is given as input and drawn.
- Image is taken as a rectangle and Based on these x, y , w, h variables, a bounding box is drawn.
- The output file is stored as "objects_present.jpg" file.

```
In [23]: '''
Object Detection - This is done to locate objects, identifies them, and draws bounding
'''

print("===== Object Detection =====")
# Get objects in the image
if len(analysis.objects) > 0:
    print("Objects in image:")

    # Prepare image for drawing
    fig = plt.figure(figsize=(8, 8))
    plt.axis('off')
    image = Image.open(image_file)
    draw = ImageDraw.Draw(image)
    color = 'cyan'
    for detected_object in analysis.objects:
        # Print object name
        print(" -{} (confidence: {:.2f}%) at location {}, {}, {}, {}".format(detected_object.name, detected_object.confidence, detected_object.rectangle.x, detected_object.rectangle.y, detected_object.rectangle.x + detected_object.rectangle.w, detected_object.rectangle.y + detected_object.rectangle.h))

        # Draw object bounding box
        r = detected_object.rectangle
        bounding_box = ((r.x, r.y), (r.x + r.w, r.y + r.h))
        draw.rectangle(bounding_box, outline=color, width=3)
        plt.annotate(detected_object.object_property, (r.x, r.y), backgroundcolor=color)

    # Save annotated image
    plt.imshow(image)

    # Save the annotated image in a new file for future reference
    outputfile = 'objects_present.jpg'
    fig.savefig(outputfile)
    print(' Results saved in ', outputfile)
else:
    print('No objects found in the image.')
```

===== Object Detection =====

Objects in image:

- Luggage and bags (confidence: 51.80% at location 1053, 1311, 1903, 2302)
- person (confidence: 73.70% at location 768, 995, 1605, 2373)
- person (confidence: 86.40% at location 948, 1355, 1616, 2979)
- person (confidence: 58.20% at location 1261, 1963, 1656, 2940)

Results saved in objects.jpg



Inference Made:

1. The model draws a bounding box of a person with co-ordinates of 768, 995, 1605, 2373.
2. The model draws a bounding box of the Luggages and Bags with co-ordinates of 1053, 1311, 1903, 2302.

10. Color Detection: This gives an idea of the dominant colour present in the image.

- The image is taken as input with read mode.
- The color feature is taken and stored on to **local_image_features**.
- `dominant_color_background()` using this, the dominant color is found to be **BLACK**.

```
In [25]: # Detect Color: To detect the different aspects of its color scheme in a Local image.

print("===== Detect Color =====")
# Open Local image
local_image = open(image_file, "rb")
# Select visual feature(s) you want
local_image_features = ["color"]
# Call API with Local image and features
detect_color_results_local = computervision_client.analyze_image_in_stream(local_image)

# Print results of the color scheme detected
print("Getting color scheme of the local image: ")
print("Is black and white: {}".format(detect_color_results_local.color.is_bw_img))
print("Accent color: {}".format(detect_color_results_local.color.accent_color))
print("Dominant background color: {}".format(detect_color_results_local.color.dominant))
print("Dominant foreground color: {}".format(detect_color_results_local.color.dominant))
print("Dominant colors: {}".format(detect_color_results_local.color.dominant_colors))
print()

===== Detect Color =====
Getting color scheme of the local image:
Is black and white: False
Accent color: 0C2622
Dominant background color: Black
Dominant foreground color: Black
Dominant colors: ['Black']
```

12. Brand Detection:

- This feature "analysis.brands" is taken as input.
- Name of the brand and confidence of the brand is found.

```
In [27]: # Detect brands: example detects brands in a Local image.

print("===== Detect brands =====")

# Get brands in the image
if (len(analysis.brands) > 0):
    print("Brands: ")
    for brand in analysis.brands:
        print(" - '{}' (confidence: {:.2f}%)".format(brand.name, brand.confidence * 100))
```

```
else:  
    print("No brands detected in image.")
```

===== Detect brands =====

No brands detected in image.

Second Image: An image of a apple laptop kept on table is taken as a second input.

```
In [28]: image_file2 = 'C:\\\\Users\\\\Priyankaa B\\\\TASK_4\\\\TASK_4_INPUTS\\\\Apple_Products.jpg'  
  
# Viewing the image file  
img(image_file2)
```

Out[28]:



1. Reading the image as image data in order to retrieve the features like description, tags, categories, brands, objects, adult.

- The image is opened and mode is set as "rb" in order to read the image as image data and features are retrieved using the computervision_client.analyze_image_in_stream() function and it is stored into analysis variable.

```
In [29]: # Get image analysis  
with open(image_file2, mode="rb") as image_data:  
    # call the API.  
    # The first argument in the function is the image to be analysed. The second argum
```

```
# We will run the analysis once per image. Then, we will display each feature in a
analysis = computervision_client.analyze_image_in_stream(image_data ,features)
```

2. Image_Description:

- Based on reading the image and features collected, a description is made.
- For Example, this is an image of "**a laptop on a table**".

```
In [30]: # Get image description
if (len(analysis.description.captions) > 0):
    print("Describing the image: ")
    for caption in analysis.description.captions:
        print("Description: '{}' (confidence: {:.2f}%)".format(caption.text, caption.confidence))
else:
    print('No description found.')

Describing the image:
Description: 'a laptop on a table' (confidence: 61.17%)
```

3. Image_Tagging:

- Based on the features extracted from the image, there are several objects identified by this **analysis.tags** statement.

```
In [31]: # Get image tags
if (len(analysis.tags) > 0):
    print("These are the tags associated with the image:")
    for tag in analysis.tags:
        print(" - '{}' (confidence: {:.2f}%)".format(tag.name, tag.confidence * 100))
else:
    print("No image tags found")
```

These are the tags associated with the image:

- 'laptop' (confidence: 98.43%)
- 'indoor' (confidence: 97.69%)
- 'computer' (confidence: 93.59%)
- 'mac' (confidence: 91.40%)
- 'table' (confidence: 91.11%)
- 'gadget' (confidence: 89.58%)
- 'sitting' (confidence: 88.75%)
- 'furniture' (confidence: 87.21%)
- 'electronic device' (confidence: 86.05%)
- 'wall' (confidence: 79.74%)
- 'floor' (confidence: 73.24%)
- 'apple' (confidence: 68.75%)
- 'wooden' (confidence: 59.45%)

Inference Made:

- Here tags identified from the image are: Indoor, laptop, computer, mac, table, furniture, floor, wooden.

Here **confidence** means the degree of certainty of a Computer Vision Algorithm in its predictions.

- From the image, the algorithm has confidence of predicting **laptop** as 'electronic device' - 86.05%.

4. Image_Category:

- Based on the image, it is segregated as indoor or outdoor.
- With the help of **analysis.categories** feature obtained from the image_data, the confidence is also printed.
- Here since its a Laptop image, it is found to be abstract category.

```
In [32]: # Get image categories
if (len(analysis.categories) > 0):
    print("Categories:")
    landmarks = []
    for category in analysis.categories:
        # Print the category
        print(" - '{}' (confidence: {:.2f}%)".format(category.name, category.score * 100))
        if category.detail:
            # Get Landmarks in this category
            if category.detail.landmarks:
                for landmark in category.detail.landmarks:
                    if landmark not in landmarks:
                        landmarks.append(landmark)

    # If there were Landmarks, list them
    if len(landmarks) > 0:
        print("Landmarks:")
        for landmark in landmarks:
            print(" - '{}' (confidence: {:.2f}%)".format(landmark.name, landmark.confidence))
    else:
        print("Unable to find image categories.")
```

Categories:
- 'abstract_' (confidence: 0.78%)
- 'others_' (confidence: 7.81%)

5. Object_Detection:

- With the help of **analysis.objects** feature obtained from the image_data, the confidence is also printed.
- Image is given as input and drawn.
- Image is taken as a rectangle and Based on these x, y , w, h variables, a bounding box is drawn.
- The output file is stored as "brands_image.jpg" file.

Here computervision_client() is the API called with local image and features.

```
In [33]: ## Detect brands: This example detects brands in a Local image. If any brands are found
print("===== Detect brands =====")
# Open local image
local_image_type = open(image_file2, "rb")
# Select visual feature(s) you want
local_image_features = ["brands"]

# Call API with local image and features
detect_type_results_local = computervision_client.analyze_image_in_stream(local_image_
# Get brands in the image
if (len(detect_type_results_local.brands) > 0):
    print("Brands: ")

    fig = plt.figure(figsize=(8, 8))
    plt.axis('off')
    image = Image.open(image_file2)
    draw = ImageDraw.Draw(image)
    color = 'cyan'
    for brand in detect_type_results_local.brands:
        # Prepare image for drawing
        print(" -'{}' (confidence: {:.2f}% at location {}, {}, {}, {})".format(brand.r
        brand.confidence * 100, brand.rectangle.x, brand.rectangle.x + brand.rectangle.
        brand.rectangle.y, brand.rectangle.y + brand.rectangle.h))
        # Draw object bounding box
        r = brand.rectangle
        bounding_box = ((r.x, r.y), (r.x + r.w, r.y + r.h))
        draw.rectangle(bounding_box, outline=color, width=3)
        plt.annotate(brand.name,(r.x, r.y), backgroundcolor=color)
    # Save annotated image
    plt.imshow(image)

    # Save the annotated image in a new file for future reference
    outputfile = 'brands_image.jpg'
    fig.savefig(outputfile)
    print(' Results saved in', outputfile)
else:
    print("No brands detected.")

===== Detect brands =====
Brands:
-'Apple' (confidence: 83.50% at location 2807, 3146, 1582, 1961)
Results saved in brands_image.jpg
```



Inference Made:

- Image of a Apple laptop is printed with its bounding box values, (2807, 3146, 1582, 1961) and confidence of 83.50%.

6. Color Detection: This gives an idea of the dominant colour present in the image.

- The image is taken as input with read mode.
- The color feature is taken and stored on to **local_image_features**.
- dominant_color_background() using this, the dominant color is found to be **GREY**.

```
In [36]: # Detect Color: To detect the different aspects of its color scheme in a Local image.

print("===== Detect Color =====")
# Open Local image
local_image = open(image_file2, "rb")
# Select visual feature(s) you want
local_image_features = ["color"]
# Call API with Local image and features
detect_color_results_local = computervision_client.analyze_image_in_stream(local_image)

# Print results of the color scheme detected
print("Getting color scheme of the local image: ")
print("Is black and white: {}".format(detect_color_results_local.color.is_bw_img))
print("Accent color: {}".format(detect_color_results_local.color.accent_color))
```

```

print("Dominant background color: {}".format(detect_color_results_local.color.dominant))
print("Dominant foreground color: {}".format(detect_color_results_local.color.dominant))
print("Dominant colors: {}".format(detect_color_results_local.color.dominant_colors))
print()

===== Detect Color =====
Getting color scheme of the local image:
Is black and white: False
Accent color: A6252E
Dominant background color: Grey
Dominant foreground color: White
Dominant colors: ['White', 'Grey', 'Black']

```

7. Brand Detection:

- This feature "analysis.brands" is taken as input.
- Name of the brand and confidence of the brand is found.

```

In [37]: # Detect brands: example detects brands in a local image.

print("===== Detect brands =====")

# Get brands in the image
if (len(analysis.brands) > 0):
    print("Brands: ")
    for brand in analysis.brands:
        print(" - '{}' (confidence: {:.2f}%)".format(brand.name, brand.confidence * 100))
else:
    print("No brands detected in image.")

===== Detect brands =====
Brands:
 - 'Apple' (confidence: 83.50%)

```

Inference Made:

The brand: Apple is found from the Apple Laptop Image present with confidence 83.50%.