Move to base directory

In [6]:
```
cd ../
```

```
C:\Users\luisr\Desktop\Repositories\Data Science Projects\Hackaton COR IV - Centro de Operações do RJ\INCUBAÇÃO\Cameras
```


In [1]:
```
#@title Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# https://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

Object Detection with MediaPipe Tasks

This notebook shows you how to use MediaPipe Tasks Python API to detect objects in images.

Preparation

Let's start with installing MediaPipe.

In [ ]:
```
# !pip install -q mediapipe==0.10.0 --user
```

Then download an off-the-shelf model. Check out the MediaPipe documentation for more image classification models that you can use.

In [5]:
```
# !wget -q -O efficientdet.tflite -q https://storage.googleapis.com/mediapipe-models/object_detector/efficientdet_lite0/int
```

```
'wget' nÆo , reconhecido como um comando interno
ou externo, um programa oper vel ou um arquivo em lotes.
```

Visualization utilities

In [1]:
```
#@markdown We implemented some functions to visualize the object detection results. <br/> Run the following cell to activat
import cv2
import numpy as np

MARGIN = 10  # pixels
ROW_SIZE = 10  # pixels
FONT_SIZE = 1
FONT_THICKNESS = 1
TEXT_COLOR = (255, 0, 0)  # red


def visualize(image, detection_result) -> np.ndarray:
    """Draws bounding boxes on the input image and return it.
    Args:
    image: The input RGB image.
    detection_result: The list of all "Detection" entities to be visualize.
    Returns:
    Image with bounding boxes.
    """
    for detection in detection_result.detections:
        # Draw bounding_box
        bbox = detection.bounding_box
        start_point = bbox.origin_x, bbox.origin_y
        end_point = bbox.origin_x + bbox.width, bbox.origin_y + bbox.height
        cv2.rectangle(image, start_point, end_point, TEXT_COLOR, 3)

        # Draw label and score
        category = detection.categories[0]
        category_name = category.category_name
        probability = round(category.score, 2)
        result_text = category_name + ' (' + str(probability) + ')'
        text_location = (MARGIN + bbox.origin_x,
                         MARGIN + ROW_SIZE + bbox.origin_y)
        cv2.putText(image, result_text, text_location, cv2.FONT_HERSHEY_PLAIN,
                    FONT_SIZE, TEXT_COLOR, FONT_THICKNESS)

    return image
```

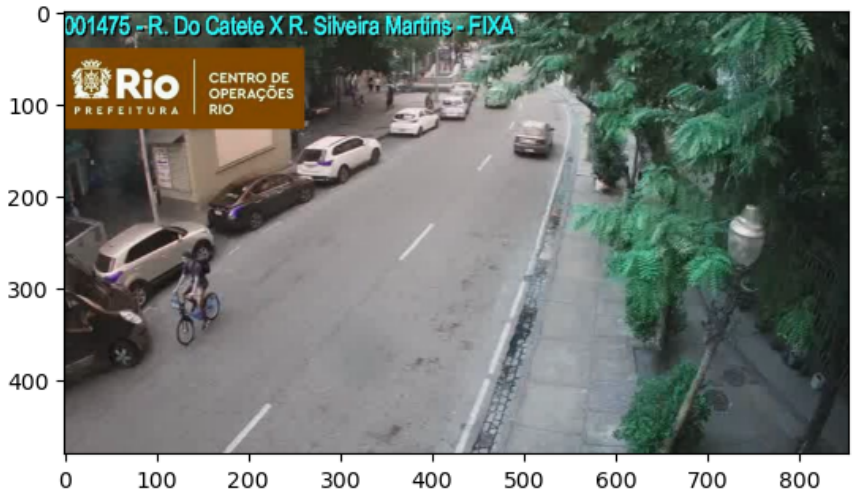
Download test image

Let's grab a test image that we'll use later. This image comes from Pixabay.

In [7]:
```
IMAGE_FILE = 'Dados/images/1475/reference/day/CODE1475_20230329_16-40-54.jpg'

import cv2
import matplotlib.pyplot as plt
# from google.colab.patches import cv2_imshow

img = cv2.imread(IMAGE_FILE)
ax = plt.imshow(img)
```



Optionally, you can upload your own image. If you want to do so, uncomment and run the cell below.

In [4]:
```
# from google.colab import files
# uploaded = files.upload()

# for filename in uploaded:
#   content = uploaded[filename]
#   with open(filename, 'wb') as f:
#     f.write(content)

# if len(uploaded.keys()):
#   IMAGE_FILE = next(iter(uploaded))
#   print('Uploaded file:', IMAGE_FILE)
```

Running inference and visualizing the results

Here are the steps to run object detection using MediaPipe.

Check out the MediaPipe documentation to learn more about configuration options that this solution supports.

In [8]:
```
# STEP 1: Import the necessary modules.
import numpy as np
import mediapipe as mp
from mediapipe.tasks import python
from mediapipe.tasks.python import vision

model_asset_path = 'models/mediapipe/efficientdet_lite0.tflite'

# STEP 2: Create an ObjectDetector object.
base_options = python.BaseOptions(model_asset_path=model_asset_path)
options = vision.ObjectDetectorOptions(base_options=base_options,
                                       score_threshold=0.5)
detector = vision.ObjectDetector.create_from_options(options)

# STEP 3: Load the input image.
image = mp.Image.create_from_file(IMAGE_FILE)

# STEP 4: Detect objects in the input image.
detection_result = detector.detect(image)

# STEP 5: Process the detection result. In this case, visualize it.
image_copy = np.copy(image.numpy_view())
annotated_image = visualize(image_copy, detection_result)
rgb_annotated_image = cv2.cvtColor(annotated_image, cv2.COLOR_BGR2RGB)
plt.imshow(rgb_annotated_image)
```

```
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
Cell In[8], line 13
     10 base_options = python.BaseOptions(model_asset_path=model_asset_path)
     11 options = vision.ObjectDetectorOptions(base_options=base_options,
     12                                        score_threshold=0.5)
---> 13 detector = vision.ObjectDetector.create_from_options(options)
     15 # STEP 3: Load the input image.
     16 image = mp.Image.create_from_file(IMAGE_FILE)

File ~\AppData\Roaming\Python\Python310\site-packages\mediapipe\tasks\python\vision\object_detector.py:234, in ObjectDetector.create_from_options(cls, options)
    220   options.result_callback(detection_result, image, timestamp)
    222 task_info = _TaskInfo(
    223     task_graph=_TASK_GRAPH_NAME,
    224     input_streams=[
   (...)
    232     task_options=options,
    233 )
--> 234 return cls(
    235     task_info.generate_graph_config(
    236         enable_flow_limiting=options.running_mode
    237         == _RunningMode.LIVE_STREAM
    238     ),
    239     options.running_mode,
    240     packets_callback if options.result_callback else None,
    241 )

File ~\AppData\Roaming\Python\Python310\site-packages\mediapipe\tasks\python\vision\core\base_vision_task_api.py:70, in BaseVisionTaskApi.__init__(self, graph_config, running_mode, packet_callback)
     65 elif packet_callback:
     66   raise ValueError(
     67       'The vision task is in image or video mode, a user-defined result '
     68       'callback should not be provided.'
     69   )
---> 70 self._runner = _TaskRunner.create(graph_config, packet_callback)
     71 self._running_mode = running_mode

RuntimeError: Unable to open file at C:\Users\luisr\Desktop\Repositories\Data Science Projects\Hackaton COR IV - Centro de Operações do RJ\INCUBAÇÃO\Cameras\models\mediapipe\efficientdet_lite0.tflite
```