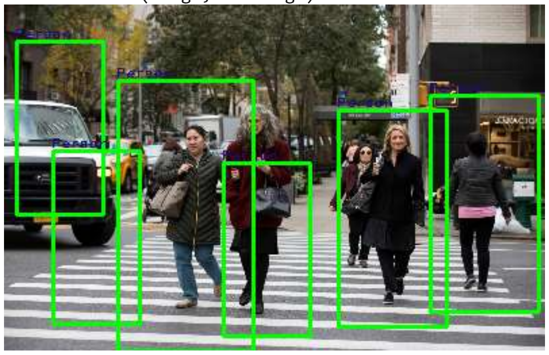
```
!git clone https://github.com/RashadGarayev/PersonDetection.git
%cd PersonDetection
```

```
Cloning into 'PersonDetection'...
remote: Enumerating objects: 6677, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 6677 (delta 3), reused 3 (delta 0), pack-reused 6668
Receiving objects: 100% (6677/6677), 65.69 MiB | 14.88 MiB/s, done.
Resolving deltas: 100% (190/190), done.
/content/PersonDetection/PersonDetection
```

```
import numpy as np
import cv2
from google.colab.patches import cv2_imshow
import cv2,joblib
import Sliding as sd
from imutils.object_detection import non_max_suppression
import imutils
from skimage.feature import hog
from skimage import color
from skimage.transform import pyramid_gaussian
```

```
image = cv2.imread('/content/00pedxing01-superJumbo.jpg')
image = cv2.resize(image,(400,256))
size = (64,128)
step_size = (9,9)
downscale = 1.25
#List to store the detections
detections = []
#The current scale of the image
scale = 0
model = joblib.load('models/models.dat')
for im_scaled in pyramid_gaussian(image, downscale = downscale):
    #The list contains detections at the current scale
    if im_scaled.shape[0] < size[1] or im_scaled.shape[1] < size[0]:
        break
    for (x, y, window) in sd.sliding_window(im_scaled, size, step_size):
        if window.shape[0] != size[1] or window.shape[1] != size[0]:
            continue
        window = color.rgb2gray(window)

        fd=hog(window, orientations=9,pixels_per_cell=(8,8),visualize=False,cells_per_block=(3,3))
        fd = fd.reshape(1, -1)
        pred = model.predict(fd)
        if pred == 1:

            if model.decision_function(fd) > 0.5:
                detections.append((int(x * (downscale**scale)), int(y * (downscale**scale)), model.decision_function(fd),
                int(size[0] * (downscale**scale)),
                int(size[1] * (downscale**scale))))

    scale += 1
clone = image.copy()
rects = np.array([[x, y, x + w, y + h] for (x, y, _, w, h) in detections])
sc = [score[0] for (x, y, score, w, h) in detections]
print ("sc: ", sc)
sc = np.array(sc)
pick = non_max_suppression(rects, probs = sc, overlapThresh = 0.3)
for(x1, y1, x2, y2) in pick:
    cv2.rectangle(clone, (x1, y1), (x2, y2), (0, 255, 0), 2)
    cv2.putText(clone,'Person',(x1-2,y1-2),1,0.75,(121,12,34),1)
cv2_imshow(clone)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:318: UserWarning: Trying to unpickle estimator LinearSVC from version 0.22.1 w
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/skimage/_shared/utils.py:348: RuntimeWarning: Images with dimensions (M, N, 3) are interpreted
  return func(*args, **kwargs)
/usr/local/lib/python3.10/dist-packages/skimage/_shared/utils.py:348: RuntimeWarning: Images with dimensions (M, N, 3) are interpreted
  return func(*args, **kwargs)
sc:  [0.8553524697658776, 1.3018006744394048, 0.8934450357206218, 2.3288157183432556, 0.8023539060191442, 1.5025955050097362, 1.977891
/usr/local/lib/python3.10/dist-packages/skimage/_shared/utils.py:348: RuntimeWarning: Images with dimensions (M, N, 3) are interpreted
  return func(*args, **kwargs)
/usr/local/lib/python3.10/dist-packages/skimage/_shared/utils.py:348: RuntimeWarning: Images with dimensions (M, N, 3) are interpreted
  return func(*args, **kwargs)
```

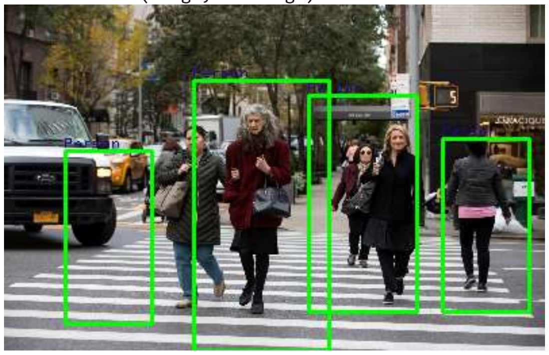


```
from skimage.feature import hog
#from skimage.io import imread
import joblib,glob,os,cv2

from sklearn.svm import LinearSVC
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn import svm, metrics
import numpy as np
from sklearn.preprocessing import LabelEncoder

train_data = []
train_labels = []
pos_im_path = 'DATAIMAGE/positive/'
neg_im_path = 'DATAIMAGE/negative/'
model_path = 'models/models.dat'
# Load the positive features
for filename in glob.glob(os.path.join(pos_im_path,"*.png")):
    fd = cv2.imread(filename,0)
    fd = cv2.resize(fd,(64,128))
    fd = hog(fd,orientations=9,pixels_per_cell=(12,12),visualize=False,cells_per_block=(5,5))
    train_data.append(fd)
    train_labels.append(1)

# Load the negative features
for filename in glob.glob(os.path.join(neg_im_path,"*.jpg")):
    fd = cv2.imread(filename,0)
    fd = cv2.resize(fd,(64,128))
    fd = hog(fd,orientations=9,pixels_per_cell=(12,12),visualize=False,cells_per_block=(5,5))
    train_data.append(fd)
    train_labels.append(0)
train_data = np.float32(train_data)
train_labels = np.array(train_labels)
print('Data Prepared........')
print('Train Data:',len(train_data))
print('Train Labels (1,0)',len(train_labels))
print("""
Classification with SVM

""")

model = LinearSVC()
print('Training...... Support Vector Machine')
model.fit(train_data,train_labels)
joblib.dump(model, 'models/models.dat')
print('Model saved : {}'.format('models/models.dat'))
```

```
Data Prepared........
Train Data: 5650
Train Labels (1,0) 5650

Classification with SVM


Training...... Support Vector Machine
Model saved : models/models.dat
```

```
import numpy as np
import cv2
from google.colab.patches import cv2_imshow
import cv2,joblib
import Sliding as sd
from imutils.object_detection import non_max_suppression
import imutils
from skimage.feature import hog
from skimage import color
from skimage.transform import pyramid_gaussian


image = cv2.imread('/content/00pedxing01-superJumbo.jpg')
image = cv2.resize(image,(400,256))
size = (64,128)
step_size = (9,9)
downscale = 1.25
#List to store the detections
detections = []
#The current scale of the image
scale = 0
model = joblib.load('models/models.dat')
for im_scaled in pyramid_gaussian(image, downscale = downscale):
    #The list contains detections at the current scale
    if im_scaled.shape[0] < size[1] or im_scaled.shape[1] < size[0]:
        break
    for (x, y, window) in sd.sliding_window(im_scaled, size, step_size):
        if window.shape[0] != size[1] or window.shape[1] != size[0]:
            continue
        window = color.rgb2gray(window)

        fd=hog(window, orientations=9,pixels_per_cell=(12,12),visualize=False,cells_per_block=(5,5))
        fd = fd.reshape(1, -1)
        pred = model.predict(fd)
        if pred == 1:

            if model.decision_function(fd) > 0.5:
                detections.append((int(x * (downscale**scale)), int(y * (downscale**scale)), model.decision_function(fd),
                int(size[0] * (downscale**scale)),
                int(size[1] * (downscale**scale))))

    scale += 1
clone = image.copy()
rects = np.array([[x, y, x + w, y + h] for (x, y, _, w, h) in detections])
sc = [score[0] for (x, y, score, w, h) in detections]
print ("sc: ", sc)
sc = np.array(sc)
pick = non_max_suppression(rects, probs = sc, overlapThresh = 0.3)
for(x1, y1, x2, y2) in pick:
    cv2.rectangle(clone, (x1, y1), (x2, y2), (0, 255, 0), 2)
    cv2.putText(clone,'Person',(x1-2,y1-2),1,0.75,(121,12,34),1)
cv2_imshow(clone)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
/usr/local/lib/python3.10/dist-packages/skimage/_shared/utils.py:348: RuntimeWarning: Images with dimensions (M, N, 3) are interpreted
  return func(*args, **kwargs)
/usr/local/lib/python3.10/dist-packages/skimage/_shared/utils.py:348: RuntimeWarning: Images with dimensions (M, N, 3) are interpreted
  return func(*args, **kwargs)
/usr/local/lib/python3.10/dist-packages/skimage/_shared/utils.py:348: RuntimeWarning: Images with dimensions (M, N, 3) are interpreted
  return func(*args, **kwargs)
sc:  [0.8578323880658427, 1.1936057357367575, 0.5021197023190718, 1.5210511775522244, 0.9096121186288295, 0.7927530547398076, 0.519087
/usr/local/lib/python3.10/dist-packages/skimage/_shared/utils.py:348: RuntimeWarning: Images with dimensions (M, N, 3) are interpreted
  return func(*args, **kwargs)
```



Double-click (or enter) to edit

Double-click (or enter) to edit