سوال دوم)

منبع: https://ricardodeazambuja.com/deep_learning/2020/04/09/colab-utils/

https://www.youtube.com/watch?v=fjynQ9P2C08

Annotation دستی تصویر با استفاده از labelImg و لینک قرار داده شده در فایل سوال:

نصب label-studio

```
>pip install label-studio
```

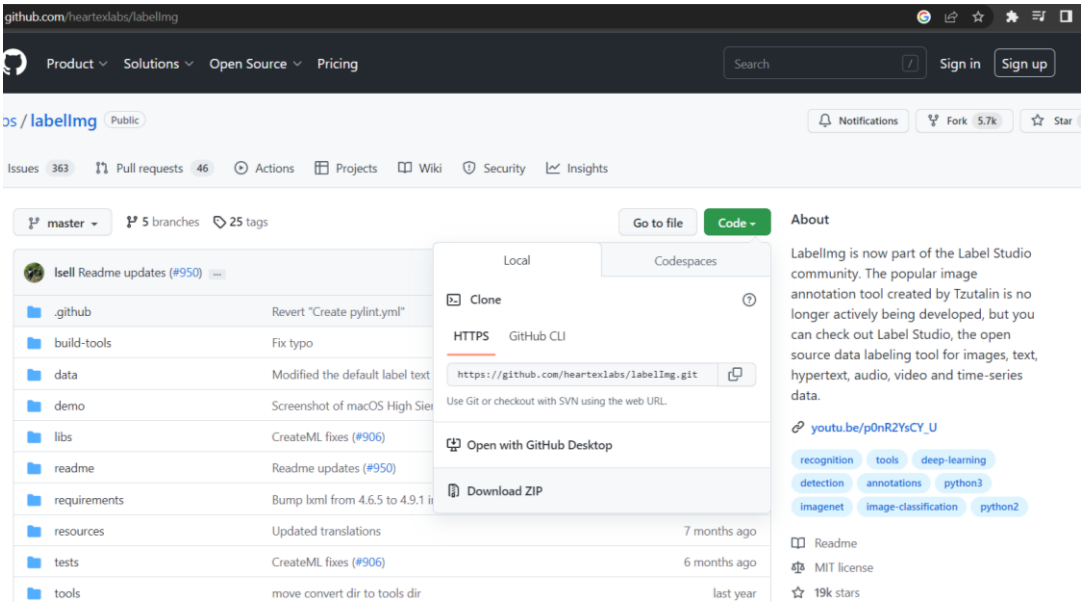
نصب PyQt:

```
>pip install pyqt5
```

نصب lxml:

```
>pip install lxml
```

دانلود فایل های ریپازیتوری گیت در منبع گفته شده:



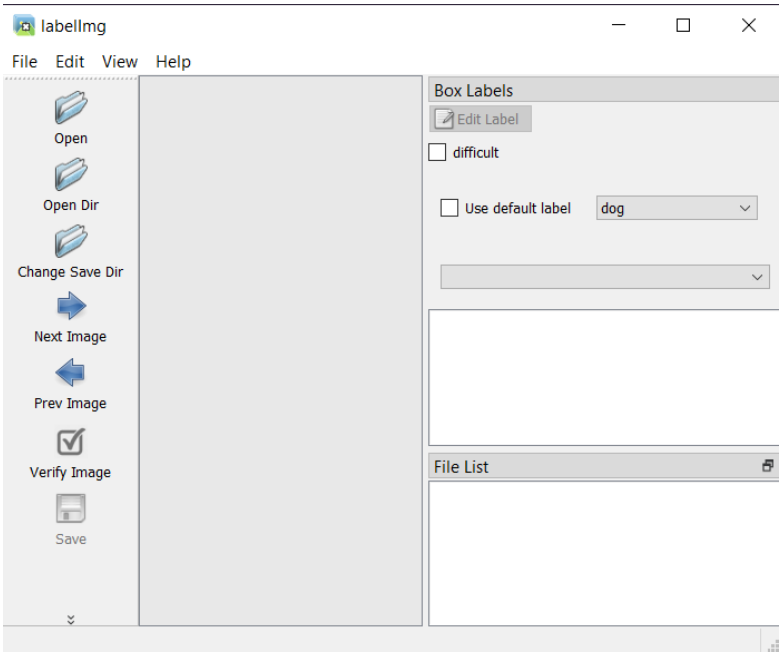
سپس از Anaconda Prompt استفاده میکنیم و یک envirement ساخته و آن را فعال میکنیم.

```
(base) C:\Users\Lenovo>conda create -n HW12
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: D:\Anaconda\envs\HW12



Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate HW12
#
# To deactivate an active environment, use
#
#     $ conda deactivate


(base) C:\Users\Lenovo>conda activate HW12
```

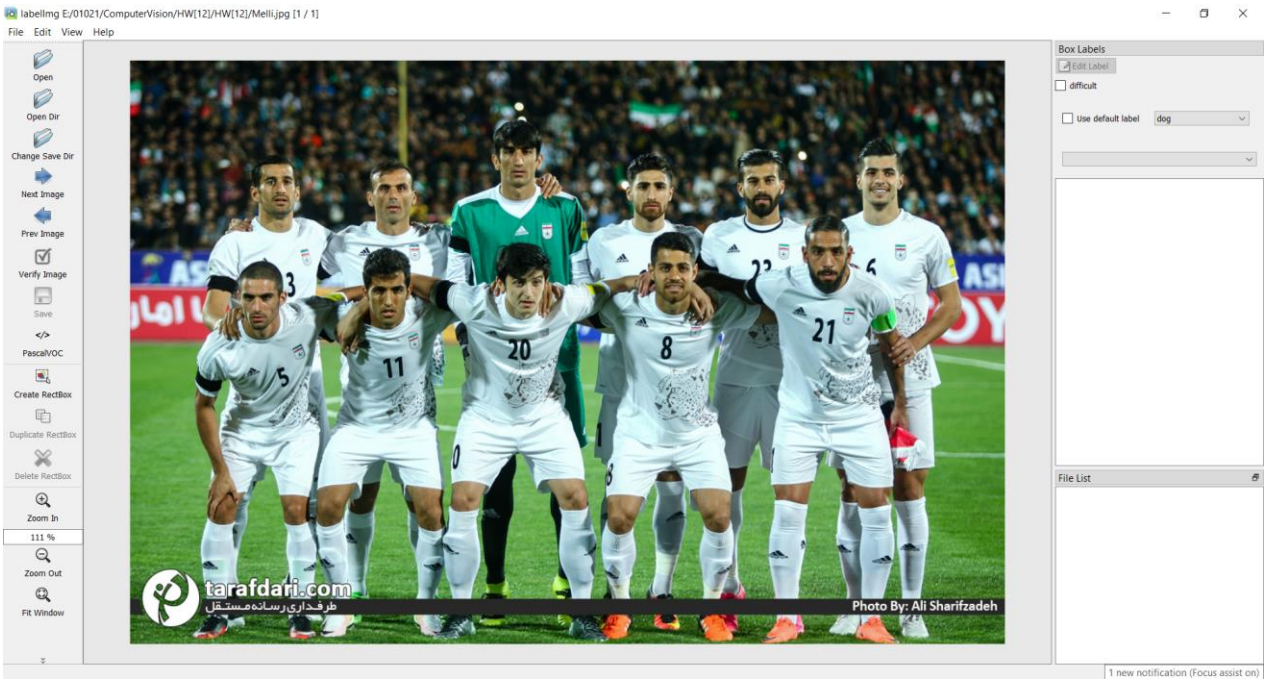
برای شروع کار دستورات زیر را هم در cmd اجرا میکنیم.

```
(HW12) C:\Users\Lenovo>conda install pyqt=5
(HW12) E:\01021\ComputerVision\HW[12]\HW[12]\Labelimg>conda install -c anaconda lxml
(HW12) E:\01021\ComputerVision\HW[12]\HW[12]\Labelimg>pyrcc5 -o libs/resources.py resources.qrc
(HW12) E:\01021\ComputerVision\HW[12]\HW[12]\Labelimg>python labelImg.py
```

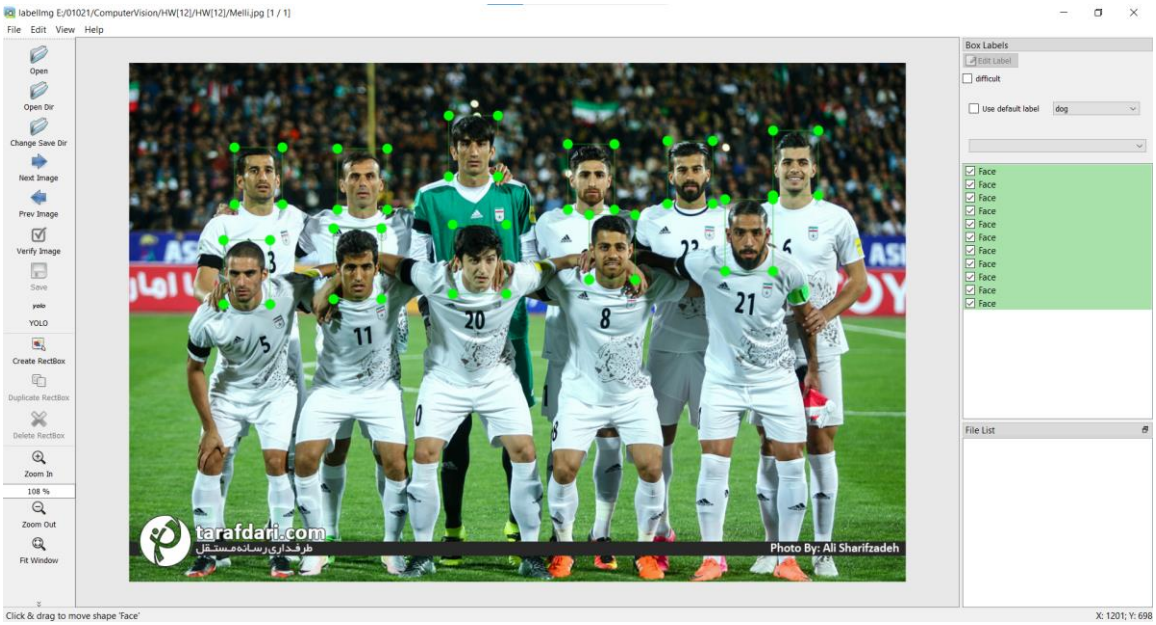
با اجرای این دستور یک پنجره با عنوان labelImg باز خواهد شد:



از طریق قسمت Open تصویر Melli.jpg را انتخاب میکنیم.



از Create RectBox استفاده کرده و طبق گفته ی سوال چهره ی بازیکن ها را به صورت دستی علامت گذاری میکنیم. من نام هر بازیکن را به عنوان label درنظرر گرفتم و سپسبه صورت فایل xml، save کردم.



دو فایل Melli.xmlو Melli.txt به صورت زیر تولید میشود.

Melli - Notepad

File Edit Format View Help

```
16 0.150833 0.403125 0.060000 0.123750
16 0.165833 0.217500 0.061667 0.110000
16 0.299583 0.223750 0.064167 0.117500
16 0.442500 0.160000 0.063333 0.117500
16 0.594167 0.219375 0.060000 0.126250
16 0.721667 0.210625 0.058333 0.121250
16 0.857500 0.192500 0.058333 0.125000
16 0.293333 0.386875 0.061667 0.136250
16 0.450417 0.378125 0.069167 0.133750
16 0.621250 0.355000 0.059167 0.125000
16 0.797500 0.331875 0.061667 0.138750
```

بخشی از فایل Melli.xml برای یکی از آبجکت ها:

```xml
<annotation>
    <folder>HW[12]</folder>
    <filename>Melli.jpg</filename>
    <path>E:\01021\ComputerVision\HW[12]\HW[12]\Melli.jpg</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>1200</width>
        <height>800</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>Face</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>145</xmin>
            <ymin>273</ymin>
            <xmax>217</xmax>
            <ymax>372</ymax>
        </bndbox>
    </object>
```

حالا نوبت اجرای عملیات پنجره ی لغزان است.

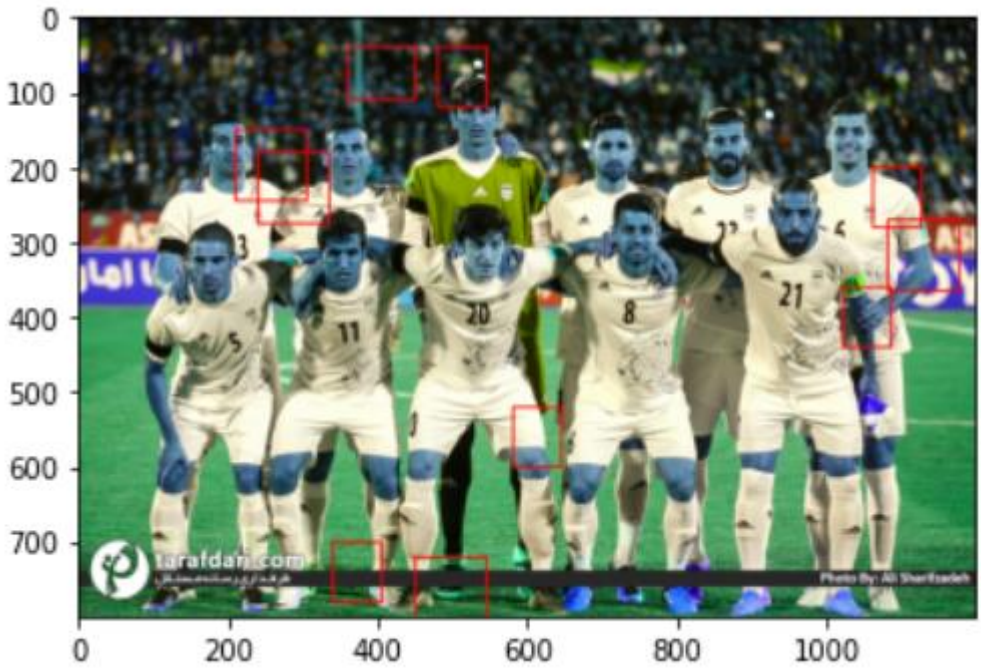
به طور میانگین طول و عرض مستطیل هایی که برچسب گذاری کردیم ۷۰ در ۹۰ بودند. من سه نوع پنجره با در نظر گرفتن این ابعاد انتخاب با step size مناسب انتخاب کردم و نواحی پیشنهادی را بدست آوردم. پنجره ی اول مستطیل با عرض بزرگتر از طول و پنجره ی دوم مستطیلی با طول بزرگتر از عرض و پنجره ی سوم مربعی است.

```
melli_img = cv2.imread("Melli.jpg")
def sliding_window(image, stepSize, windowSize):
  for y in range(0, image.shape[0], stepSize):
    for x in range(0, image.shape[1], stepSize):
      yield [x, y, x + windowSize[0], y + windowSize[1]]

windows1 = list(sliding_window(melli_img, 20, (65, 80)))
windows2 = list(sliding_window(melli_img, 40, (90, 70)))
windows3 = list(sliding_window(melli_img, 30, (95, 95)))
windows = windows1.copy() + windows2.copy() + windows3.copy()
print(len(windows))
```

```
4080
```

هر پنجره ی پیشنهادی از جنس لیست و شامل مختصات دو نقطه روی مستطیل های روی تصویر است.

۱۰ تا ازا این ناحیه ها را به صورت رندوم روی تصویر نشان دادم:



فایل xml تولید شده توسط LabelImg را با کمک کتابخانه ی xml.etree.ElementTree خواندم ومختصات نواحی علامت گذاری شده را به صورت لیست ذخیره کردم. برای نمایش بهتر از pandas استفاده کردم.

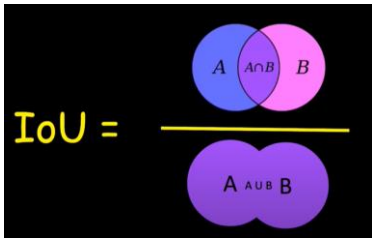
```
import xml.etree.ElementTree as ET
tree = ET.parse('Melli.xml')
root = tree.getroot()
data = []
boxes = []
for i in range(6, 17):
  data.append([root[i][0].text, root[i][4][0].text, root[i][4][1].text, root[i][4][2].text, root[i][4][3].text].copy())
  boxes.append([int(root[i][4][0].text), int(root[i][4][1].text), int(root[i][4][2].text), int(root[i][4][3].text)].copy())
df = pd.DataFrame(data)
df.columns = ["Label", "xmin", "ymin", "xmax", "ymax"]
```

خروجی:

	Label	xmin	ymin	xmax	ymax
0	Face	145	273	217	372
1	Face	162	130	236	218
2	Face	321	132	398	226
3	Face	493	81	569	175
4	Face	677	125	749	226
5	Face	831	120	901	217
6	Face	994	104	1064	204
7	Face	315	255	389	364
8	Face	499	249	582	356
9	Face	710	234	781	334
10	Face	920	210	994	321

بعد از آن ناحیه های پیشنهادی را از نظر IOU با نواحی برچسب گذاری شده مقایسه میکنیم. من حد آستانه را۰٫۵گرفتم.

تعریف: نسبت اشتراک به اجتماع مجموعه ها که برای معیاری برای مقایسه ی دو ناحیه و ارزیابی آنها است.

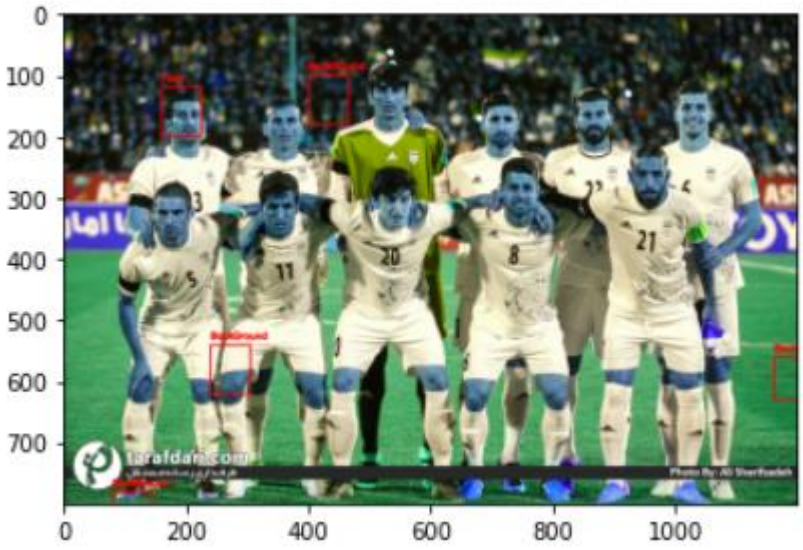



```python
def IOU(box, window):
    # مختصات ناحیه ی مشترک
    xmin = max(box[0], window[0])
    ymin = max(box[1], window[1])
    xmax = min(box[2], window[2])
    ymax = min(box[3], window[3])
    #مساحت ناحیه
    x = max(xmax - xmin, 0)
    y = max((ymax - ymin), 0)
    i = abs(x * y)
    if i == 0:
        return 0
    #اجتماع ناحیه ها
    #اجتماع = جمع دو مجموعه -اشتراک دو مجموعه
    x1 = (box[2] - box[0])
    y1 = (box[3] - box[1])
    s1 = abs(x1 * y1)
    x2 = window[2] - window[0]
    y2 = window[3] - window[1]
    s2 = abs(x2 * y2)
    u = s1 + s2 - i
    iou = i / float(u)
    return iou
```

باید هر پنجره ی پیشنهادی را با ۱۱ ناحیه ای که داریم مقایسه کنیم اگر IoU بیشتر از ۰٫۵ شد برچسب Face و اگر کمتر از ۰٫۳ شد Background برای آن درنظر بگیریم. برخی پنجره ها که مقدار IOU آنها با باکس ها بین این دو مقدار است ایگنور میشوند و یا به عبارتی دورریخته خواهند شد.

```
Hightreshold = 0.5
Lowtreshold = 0.3
Labels = []
for window in windows:
  max_iou = 0
  for box in boxes:
    h = IOU(box, window)
    if h > max_iou:
      max_iou = h
  if max_iou > Hightreshold:
    Labels.append(["Face", window].copy())
  elif max_iou < Lowtreshold:
    Labels.append(["BackGround", window].copy())

```

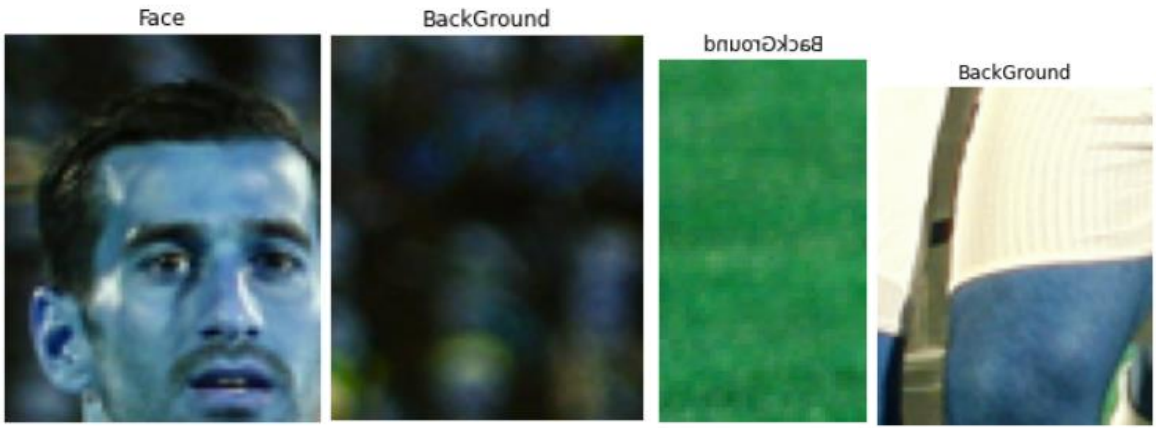
طبق خواسته ی سوال ۵ تا از آنها را به صورت رندوم همراه Label روی تصویر مشخص میکنیم.



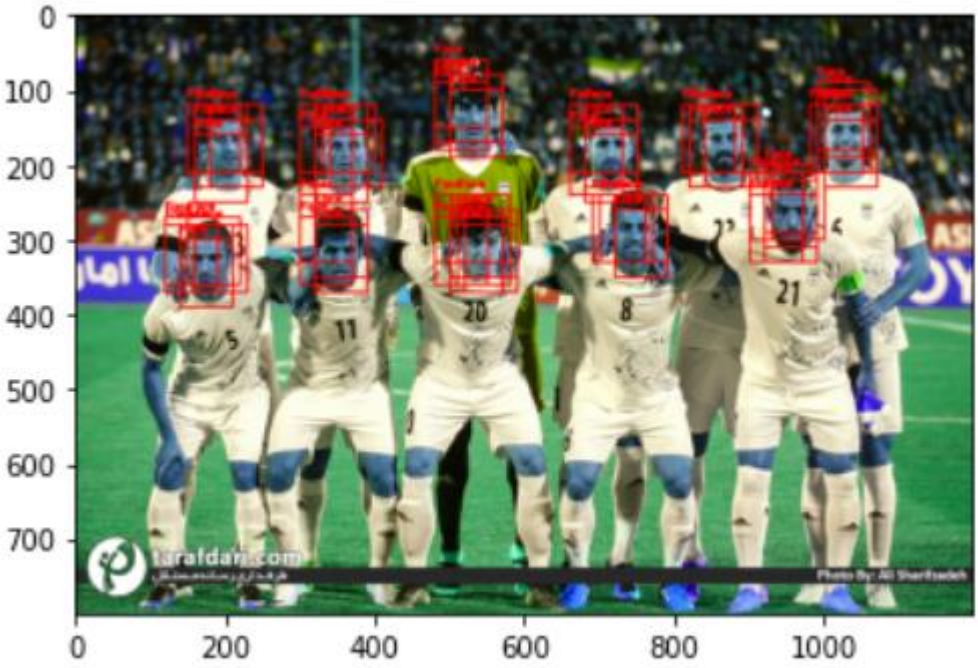
همانطور که مشخص است دو تا از ناحیه ها که در سمت چپ تصویر هستند به عنوان Face مشخص شدند و سه ناحیه ی دیگر که شامل صورت بازیکن ها نبودند به عنوان بکگراند مشخص شدند.

طبق خواسته ی سوال به طور مجزا هم رسمشان کردم:

```
for x in indexes2:
  i = Labels[x][1]
  # print(i)
  # print(np.array(list(i).copy(), dtype = object)[2].shape)
  im = np.array(melli_img[i[1]:i[3], i[0]:i[2]])
  # print(melli_img.shape)
  plt.imshow(im)
  plt.title(Labels[x][0])
  plt.axis("off")
  plt.show()

plt.show()
```



برای اینکه مطمئن شوم درست کار میکند پنجره های کلاس Face را هم جدا رسم کردم چون شاید در حالت رندوم بین موارد انتخابی نباشد.



برای اجرای درست سلول خای کد باید فایل تصویر و xml را در محیط google colab آپلود کنید.