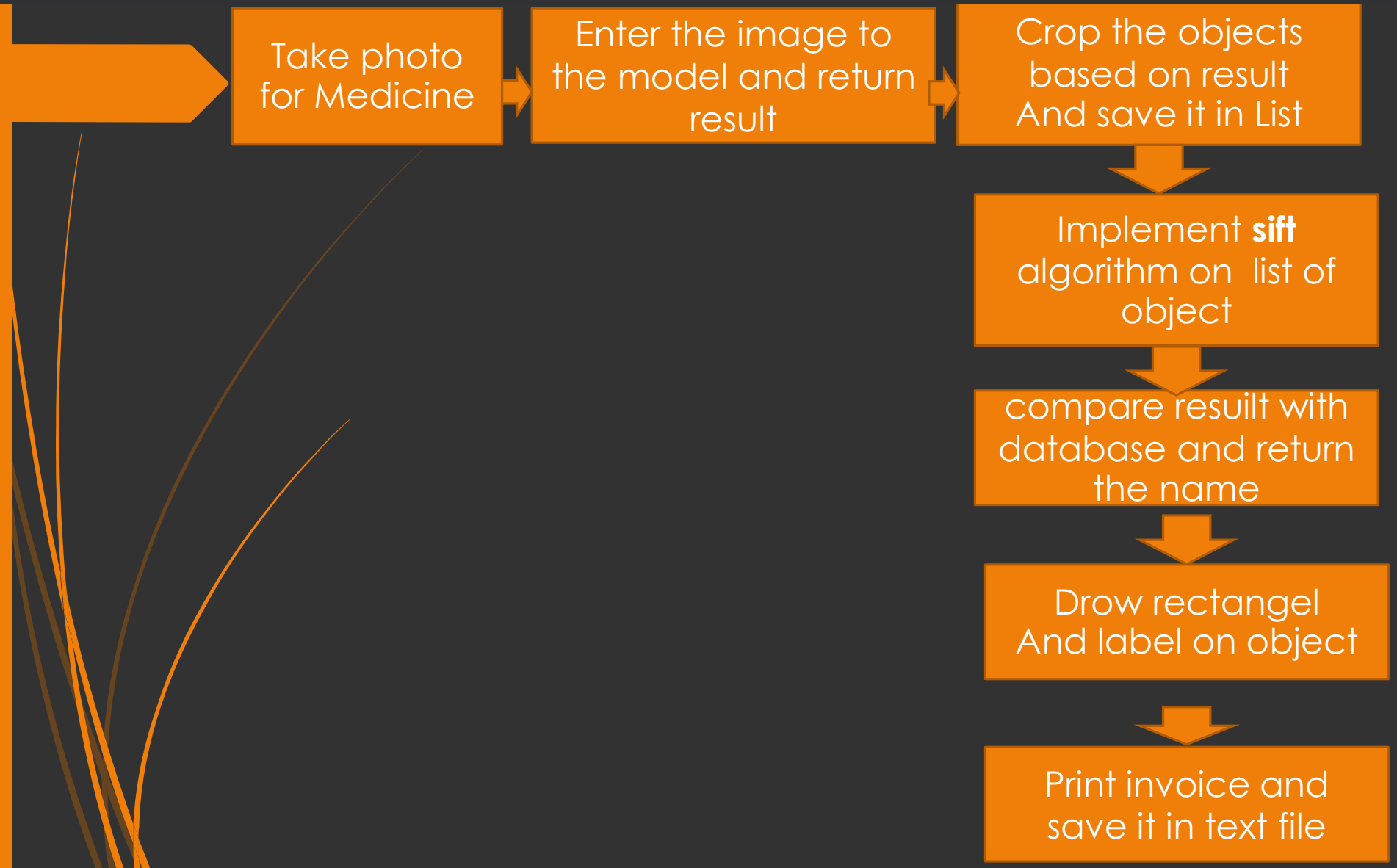
نظام محاسبة في صيدلية باستخدام نموذج yolo8 وخوارزمية ال sift

إعداد الطلاب :

- محمد قاسم الجمعات

Enter the image for the model

مخطط سير العمل:



لمحة عن المشروع :

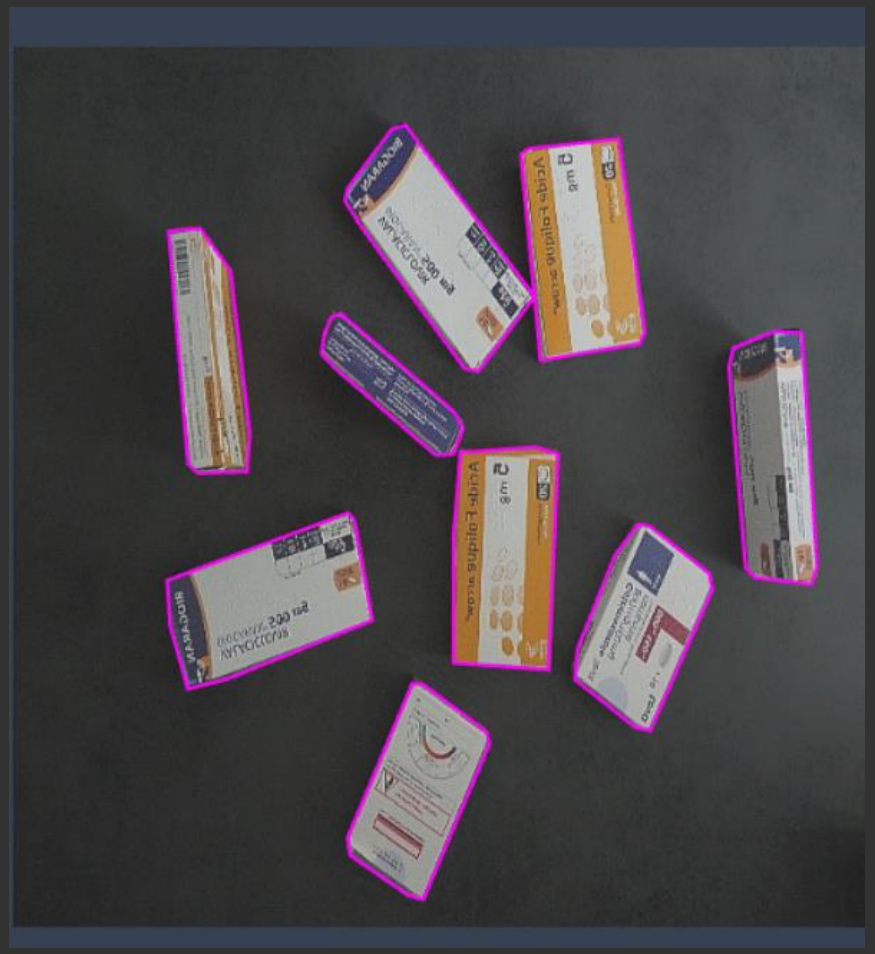
- يهدف المشروع الى تسهيل عملية المحاسبة checkoutفي صيدلية وذلك

باستخدام تقنيات الابصار الحاسوبي بدل من نظام baracode التقليدي

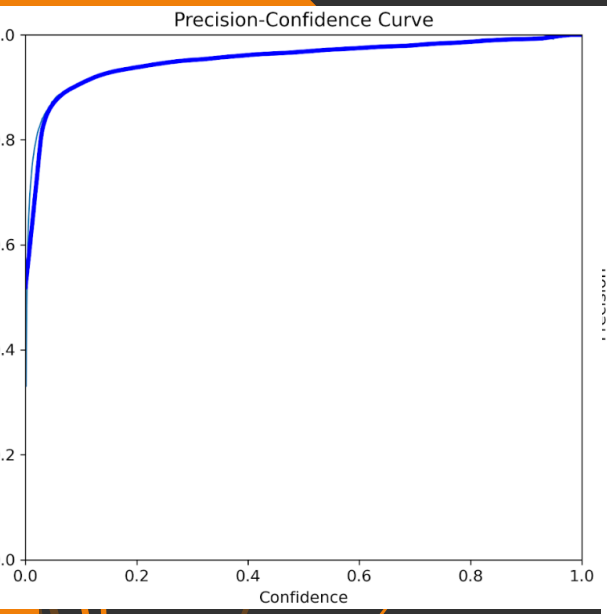
- حيث يسمح بحساب أسعار عدّة منتجات في نفس الوقت بدل من منتج واحد في كل مرة

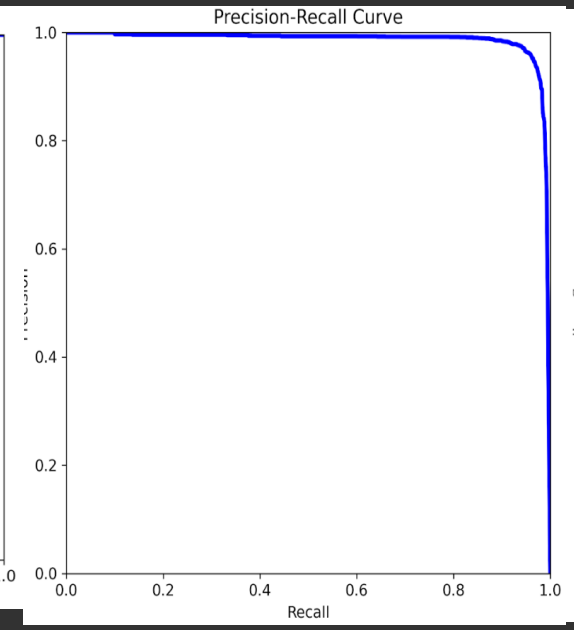
مراحل إنجاز المشروع: 1- <التدريب>

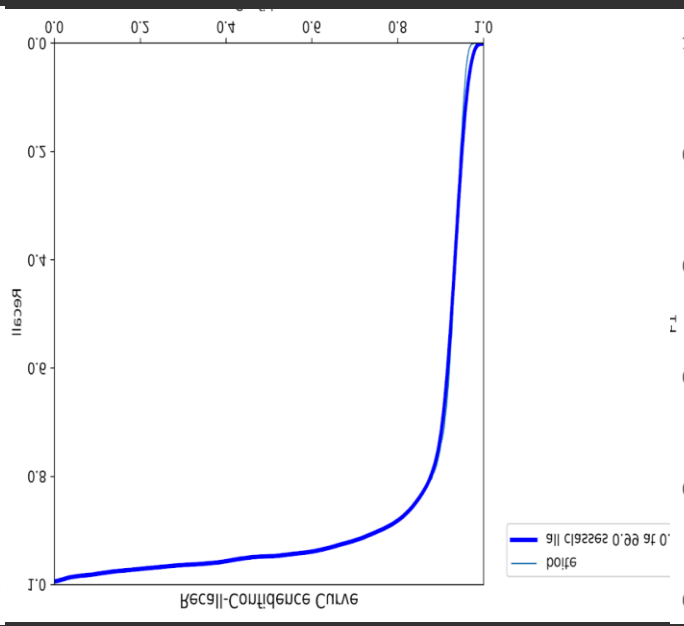
في البداية تم اختيار نموذج الابصار الحاسوبي yolov8 وتدريبه على dataset من موقع roboflow تتضمن 2000 صوره لعلب أدوية في مختلف الوضعيات.

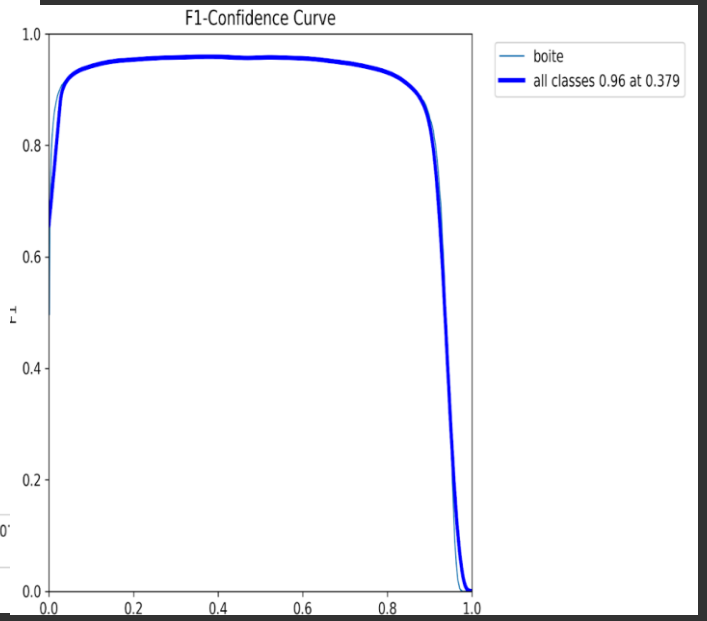


- التدريب تم على موقع google colab وتم الحصول على النتائج التالية









- وفي النهاية التدريب تم حفظ الأوزان بصيغة (.pt) لاجل استخدامه في المشروع

تنفيذ المشروع :

1 - في البداية لدينا التابع التى الذى يطبق خوارزمية ال sift على صورة :

```
def extract_sift_features_from_image(image):
    query_img =cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    sift = cv2.SIFT_create()
    query_keypoints, query_descriptors= sift.detectAndCompute(query_img, None)
    return query_keypoints, query_descriptors
```

2 -لدينا ايضا التابع التالي الذي يطبق خوارزمية siftعلى مجلد من الصور ويحفظ الناتج في قائمة مع اسم الدواء

```
def extract_sift_features_from_folder(folder_path):
    features_list = []
    image_files = [f for f in os.listdir(folder_path) if f.lower().endswith(('.p
    for index, image_file in enumerate(image_files):
        image_path = os.path.join(folder_path, image_file)
        img = cv2.imread(image_path)
        keypoints, descriptors=  extract_sift_features_from_image(img)
        features_list.append({
            'index': index,
            'image_path': str(image_path).split('/')[1].split('.')[0],
            'keypoints': keypoints,
            'descriptors': descriptors
        })

    return features_list
```

- في الخطوة الاولى عند تشغيل البرنامج يتم اختيار صورة من مجلد وذلك بمساعدة مكتبة **pickfile**
- يتم معالجة الصورة وذلك بتصغير حجمها من اجل ان تتناسب مع النموذج من خلال **cv2.resize**
- بعد ذلك يقرأ الصور النموذج من خلال **model.prdict** وتحفظ النتائج في المتغير **result**
- يحوي هذا المتغير على احداثيات الكائنات
- ندخل هذه الاحداثيات على التابع **crop_object** لاقتطاع الكائنات من الصورة وحفظها في list

```python
def resize_img(image, per):
    down_width = image.shape[0] - (image.shape[0] * per)
    down_height = image.shape[1] - (image.shape[1] * per)
    down_points = (int(down_height), int(down_width))
    resized_down = cv2.resize(image, down_points,
                              interpolation=cv2.INTER_LINEAR
    return resized_down
```

(تصغير الحجم)

```python
def prdict(model, img_recized):
    # pretrained YOLOv8n model
    results = model.predict(
        source=img_recized,
        show=False,
        save=False,
        conf=0.8,
        verbose=False,
        imgsz=640,
        show_labels=False,
        show_conf=False,
    )

    return results
```

(الكشف)

```python
def crop_object(results,img):
    list_of_object=[]
    boxes = results[0].boxes.xyxy.tolist()
    if len(boxes)==0 :
        list_of_object.append(img)
        return list_of_object
    for i, box in enumerate(boxes):
        x1, y1, x2, y2 = box
        cropped_image = img[int(y1):int(y2), int(x1):int(x2)]
        list_of_object.append(cropped_image)
    return list_of_object
```

(القص)

- يتم بعد ذلك تطبيق خوازمية ال siftعلى كل كائن مقتطع ومقارنة النتائج من قاعدة المعطيات التي تحوي على ناتج خوارزمية (key ,des) مع اسم الدواء وارجاع اسم الدواء الذي حصل على اعلى تطابق
- تتم حساب مقدار التطابق من خلال خوارزمية الجار الاقرب (k-nearest neighbors algorithm)
- وتكرر هذه العملية حتى انتهاء جميع الكائنات

```python
def match_sift_features(query_keypoints, query_descriptors,
                        sift_features_list, threshold=0.6):
    bf = cv2.BFMatcher()
    best_match = None
    max_matches = 0
    for features in sift_features_list:
        stored_keypoints = features['keypoints']
        stored_descriptors = features['descriptors']
        matches = bf.knnMatch(query_descriptors, stored_descriptors, k=2)
        good_matches = []
        for m, n in matches:
            if m.distance < threshold * n.distance:
                good_matches.append(m)
        if len(good_matches) > max_matches:
            max_matches = len(good_matches)
            best_match = features

    return best_match
```

(المطابقة)

```python
for object in list_of_object:
    key, des = extract_sift_features_from_image(object)
    best_match = match_sift_features(key, des, sift_features)
    name=best_match['image_path']
    price=prices[best_match["image_path"]]
    if best_match:
        tabel.add_row([name,f"{price}"+" SP"])

        total = total+price
    else:
        print("No match found.")
```

(طباعة ناتج المطابقة)

- في النهاية يتم طباعة الصورة مع العناصر المكتشفة من خلال تابع drow label

```
def drow_label(imge,results):
    text="Medicine box"
    image = imge
    boxes = results[0].boxes.xyxy.tolist()
    for i, box in enumerate(boxes):
        x1, y1, x2, y2 = box
        start_point = (int(x1), int(y2))
        start_point_text = (int(x1), int(y1))
        end_point = (int(x2), int(y1))
        new_end_point = (int(x2), int((y1 - 25)))
        color = (255, 0, 0)
        text_color = (255, 255, 1)
        thickness = 2
        fontFace = cv2.FONT_HERSHEY_DUPLEX
        fontScale = 1
        thickness = 2
        image = cv2.rectangle(image, start_point, end_point, color, thickness)
        image = cv2.rectangle(image, start_point_text, new_end_point, color, -1)
        image = cv2.putText(
```

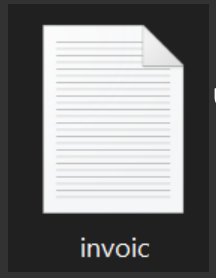
الخرج النهائي :

```
+----------+----------+
|   Item   |  price   |
+----------+----------+
|  Laxine  | 5500 SP  |
| Skelflex | 15500 SP |
| Deopflam | 14500 SP |
+----------+----------+

total is.............. = 35500 SP
```

(الفاتورة)

(الكائنات المكتشفة)



في النهاية يتم حفظ جميع الفواتير في ملف