P，R，P-R，F1曲线绘制

- 什么是PR曲线？为什么需要这一个评估指标？

PR（Precession–Recall curve）是一种评估二分类模型的指标。PR曲线反应了在不同的阈值下,准确率和召回率的关系

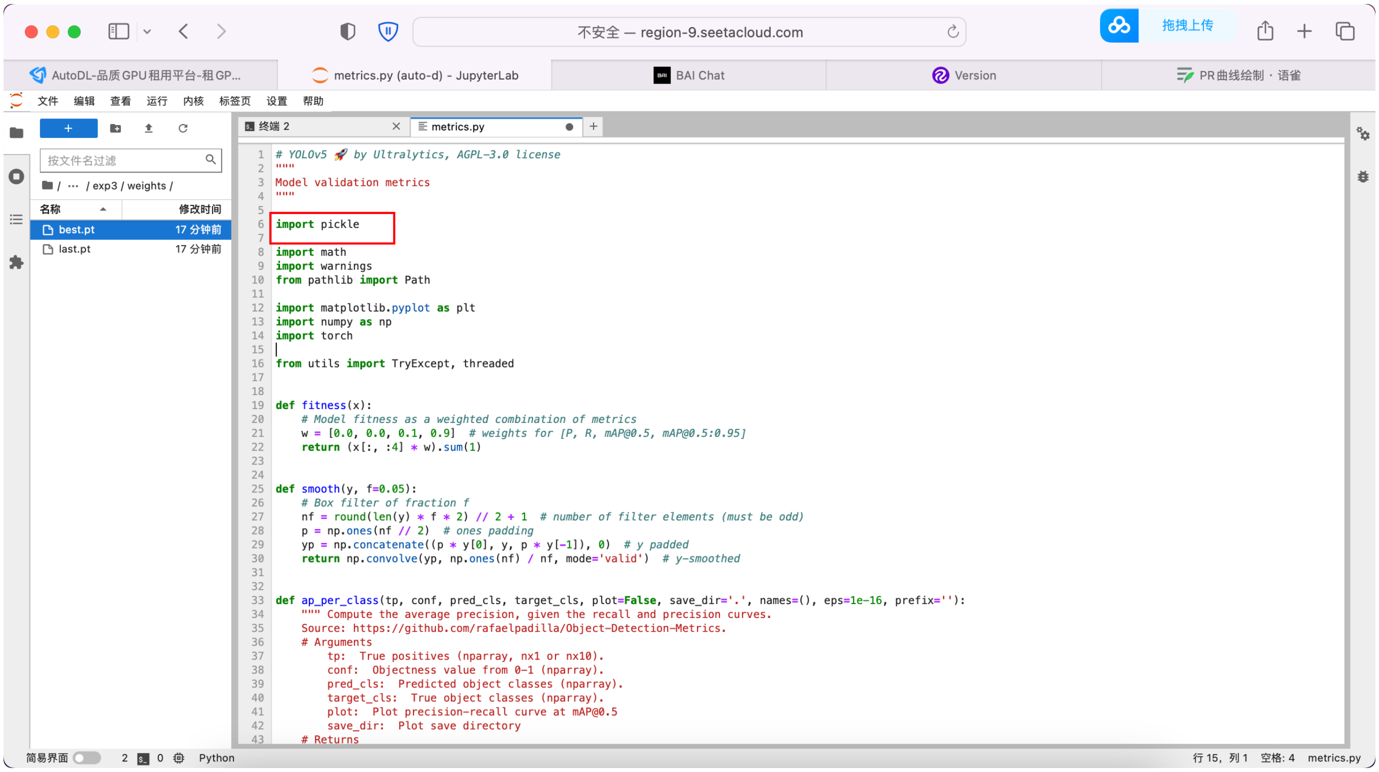
- 如何把多条PR，P，RF1等曲线绘制在一起？

原理：使用序列化将绘制的多条曲线的所需的数据分别保存在pkl文件中，再通过反序列从pkl文件中读取出多条曲线数据，最终在绘图函数中把它们绘制在一起。

步骤一：获取pkl

（1）在utills/metrics.py头部加上

```
import pickle
```



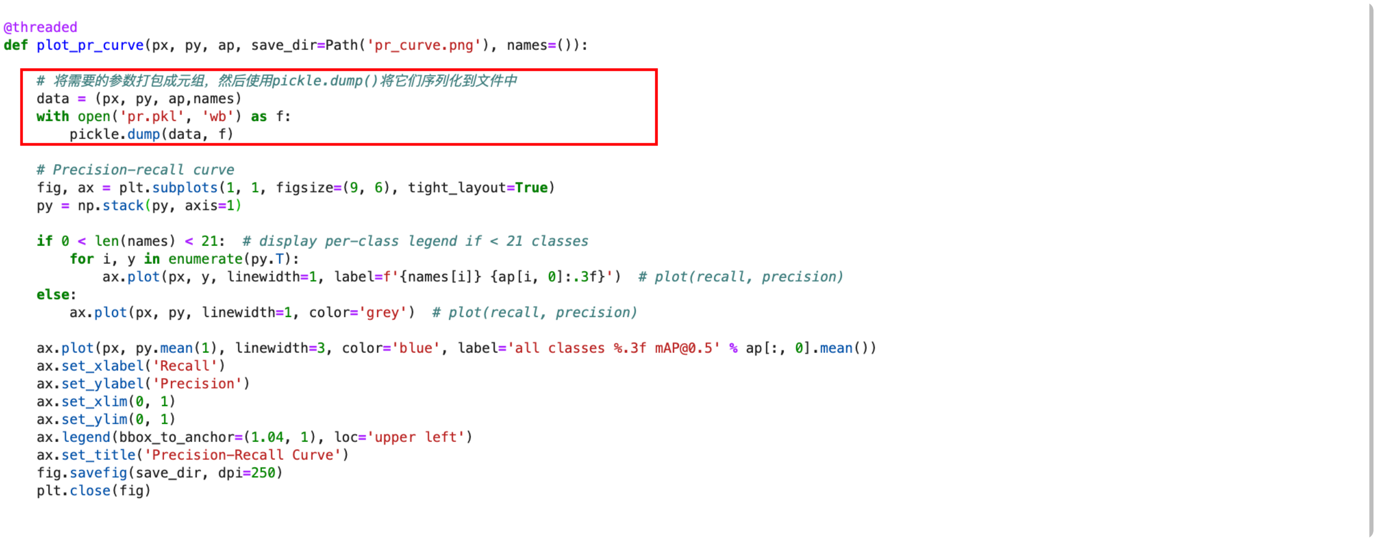
```python
# YOLOv5 🚀 by Ultralytics, AGPL-3.0 license
"""
Model validation metrics
"""

import pickle

import math
import warnings
from pathlib import Path

import matplotlib.pyplot as plt
import numpy as np
import torch

from utils import TryExcept, threaded


def fitness(x):
    # Model fitness as a weighted combination of metrics
    w = [0.0, 0.0, 0.1, 0.9]  # weights for [P, R, mAP@0.5, mAP@0.5:0.95]
    return (x[:, :4] * w).sum(1)


def smooth(y, f=0.05):
    # Box filter of fraction f
    nf = round(len(y) * f * 2) // 2 + 1  # number of filter elements (must be odd)
    p = np.ones(nf // 2)  # ones padding
    yp = np.concatenate((p * y[0], y, p * y[-1]), 0)  # y padded
    return np.convolve(yp, np.ones(nf) / nf, mode='valid')  # y-smoothed


def ap_per_class(tp, conf, pred_cls, target_cls, plot=False, save_dir='.', names=(), eps=1e-16, prefix=''):
    """ Compute the average precision, given the recall and precision curves.
    Source: https://github.com/rafaelpadilla/Object-Detection-Metrics.
    # Arguments
        tp:  True positives (nparray, nx1 or nx10).
        conf:  Objectness value from 0-1 (nparray).
        pred_cls:  Predicted object classes (nparray).
        target_cls:  True object classes (nparray).
        plot:  Plot precision-recall curve at mAP@0.5
        save_dir:  Plot save directory
    # Returns
```

（2）修改plot_pr_curve函数

添加以下内容

Plain Text　复制代码

```
    # 将需要的参数打包成元组，然后使用pickle.dump()将它们序列化到文件中
    data = (px, py, ap,names)
    with open('pr.pkl', 'wb') as f:
        pickle.dump(data, f)
```



```python
@threaded
def plot_pr_curve(px, py, ap, save_dir=Path('pr_curve.png'), names=()):

    # 将需要的参数打包成元组，然后使用pickle.dump()将它们序列化到文件中
    data = (px, py, ap,names)
    with open('pr.pkl', 'wb') as f:
        pickle.dump(data, f)

    # Precision-recall curve
    fig, ax = plt.subplots(1, 1, figsize=(9, 6), tight_layout=True)
    py = np.stack(py, axis=1)

    if 0 < len(names) < 21:  # display per-class legend if < 21 classes
        for i, y in enumerate(py.T):
            ax.plot(px, y, linewidth=1, label=f'{names[i]} {ap[i, 0]:.3f}')  # plot(recall, precision)
    else:
        ax.plot(px, py, linewidth=1, color='grey')  # plot(recall, precision)

    ax.plot(px, py.mean(1), linewidth=3, color='blue', label='all classes %.3f mAP@0.5' % ap[:, 0].mean())
    ax.set_xlabel('Recall')
    ax.set_ylabel('Precision')
    ax.set_xlim(0, 1)
    ax.set_ylim(0, 1)
    ax.legend(bbox_to_anchor=(1.04, 1), loc='upper left')
    ax.set_title('Precision-Recall Curve')
    fig.savefig(save_dir, dpi=250)
    plt.close(fig)
```

修改后完整代码如下：

Plain Text 复制代码

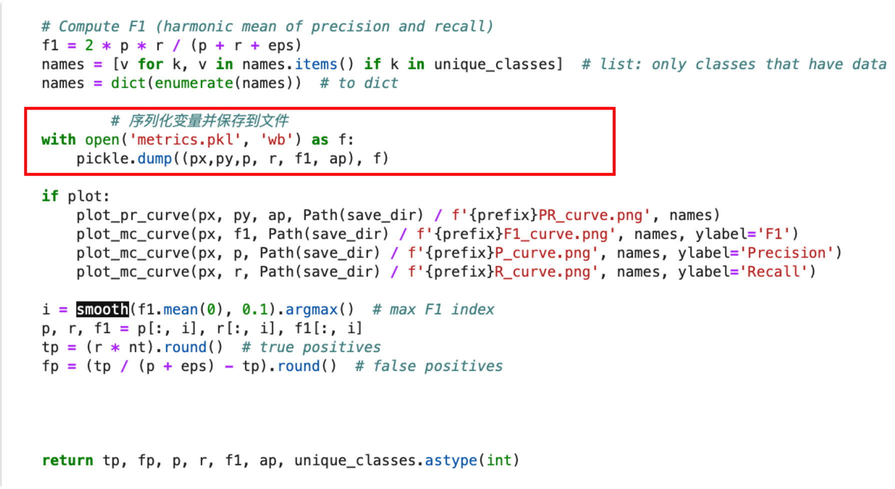
```
def plot_pr_curve(px, py, ap, save_dir=Path('pr_curve.png'), names=()):

    # 将需要的参数打包成元组，然后使用pickle.dump()将它们序列化到文件中
    data = (px, py, ap,names)
    with open('pr.pkl', 'wb') as f:
        pickle.dump(data, f)

    # Precision-recall curve
    fig, ax = plt.subplots(1, 1, figsize=(9, 6), tight_layout=True)
    py = np.stack(py, axis=1)

    if 0 < len(names) < 21:  # display per-class legend if < 21 classes
        for i, y in enumerate(py.T):
            ax.plot(px, y, linewidth=1, label=f'{names[i]} {ap[i, 0]:.3f}')  # plot(recall, precision)
    else:
        ax.plot(px, py, linewidth=1, color='grey')  # plot(recall, precision)

    ax.plot(px, py.mean(1), linewidth=3, color='blue', label='all classes %.3f mAP@0.5' % ap[:, 0].mean())
    ax.set_xlabel('Recall')
    ax.set_ylabel('Precision')
    ax.set_xlim(0, 1)
    ax.set_ylim(0, 1)
    ax.legend(bbox_to_anchor=(1.04, 1), loc='upper left')
    ax.set_title('Precision-Recall Curve')
    fig.savefig(save_dir, dpi=250)
    plt.close(fig)
```

（3）修改ap_per_class函数

添加以下内容：

Plain Text 复制代码

```
            # 序列化变量并保存到文件
    with open('metrics.pkl', 'wb') as f:
        pickle.dump((px,py,p, r, f1, ap), f)
```



```
# Compute F1 (harmonic mean of precision and recall)
f1 = 2 * p * r / (p + r + eps)
names = [v for k, v in names.items() if k in unique_classes]  # list: only classes that have data
names = dict(enumerate(names))  # to dict

        # 序列化变量并保存到文件
with open('metrics.pkl', 'wb') as f:
    pickle.dump((px,py,p, r, f1, ap), f)

if plot:
    plot_pr_curve(px, py, ap, Path(save_dir) / f'{prefix}PR_curve.png', names)
    plot_mc_curve(px, f1, Path(save_dir) / f'{prefix}F1_curve.png', names, ylabel='F1')
    plot_mc_curve(px, p, Path(save_dir) / f'{prefix}P_curve.png', names, ylabel='Precision')
    plot_mc_curve(px, r, Path(save_dir) / f'{prefix}R_curve.png', names, ylabel='Recall')

i = smooth(f1.mean(0), 0.1).argmax()  # max F1 index
p, r, f1 = p[:, i], r[:, i], f1[:, i]
tp = (r * nt).round()  # true positives
fp = (tp / (p + eps) - tp).round()  # false positives




return tp, fp, p, r, f1, ap, unique_classes.astype(int)
```

修改后完整代码如下：

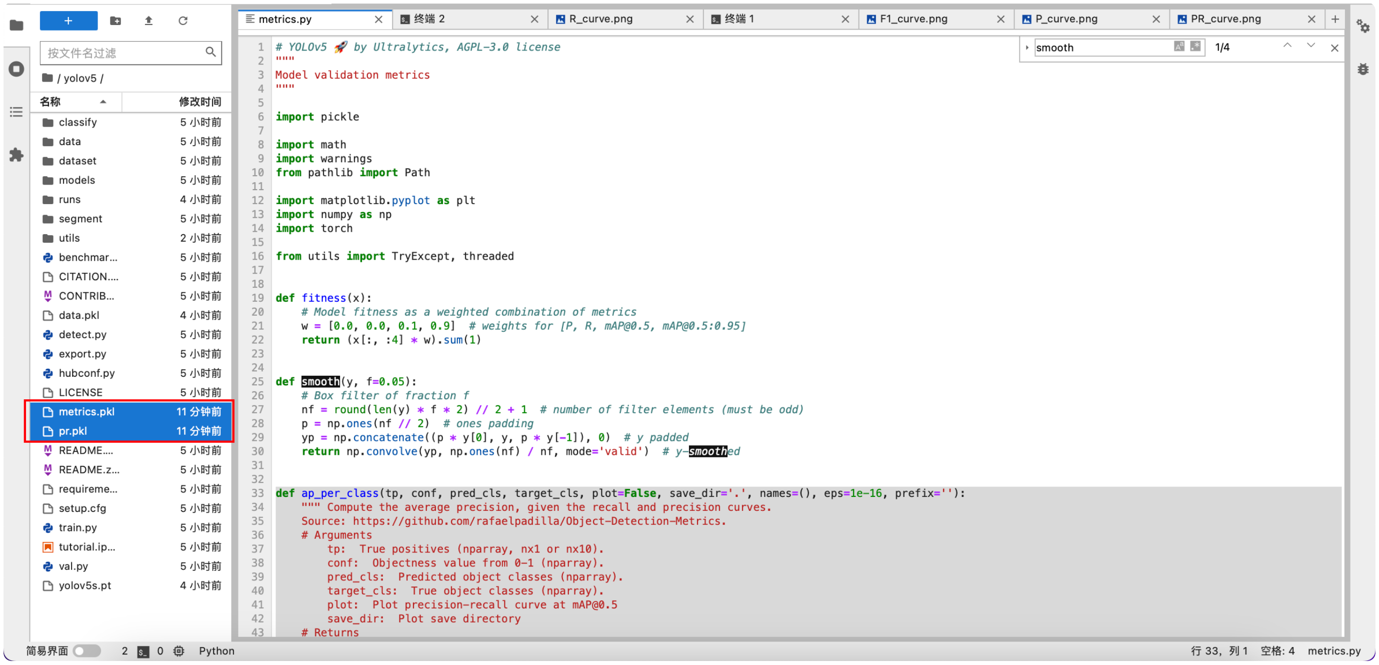
Plain Text 复制代码

```
def ap_per_class(tp, conf, pred_cls, target_cls, plot=False, save_dir='.', names=(), eps=1e-16, prefix=''):
    """ Compute the average precision, given the recall and precision curves.
    Source: https://github.com/rafaelpadilla/Object-Detection-Metrics.
    # Arguments
        tp:  True positives (nparray, nx1 or nx10).
        conf:  Objectness value from 0-1 (nparray).
        pred_cls:  Predicted object classes (nparray).
        target_cls:  True object classes (nparray).
        plot:  Plot precision-recall curve at mAP@0.5
        save_dir:  Plot save directory
    # Returns
        The average precision as computed in py-faster-rcnn.
    """

    # Sort by objectness
    i = np.argsort(-conf)
    tp, conf, pred_cls = tp[i], conf[i], pred_cls[i]

    # Find unique classes
    unique_classes, nt = np.unique(target_cls, return_counts=True)
    nc = unique_classes.shape[0]  # number of classes, number of detections

    # Create Precision-Recall curve and compute AP for each class
    px, py = np.linspace(0, 1, 1000), []  # for plotting
    ap, p, r = np.zeros((nc, tp.shape[1])), np.zeros((nc, 1000)), np.zeros((nc, 1000))
    for ci, c in enumerate(unique_classes):
        i = pred_cls == c
        n_l = nt[ci]  # number of labels
        n_p = i.sum()  # number of predictions
        if n_p == 0 or n_l == 0:
            continue

        # Accumulate FPs and TPs
        fpc = (1 - tp[i]).cumsum(0)
        tpc = tp[i].cumsum(0)

        # Recall
        recall = tpc / (n_l + eps)  # recall curve
        r[ci] = np.interp(-px, -conf[i], recall[:, 0], left=0)  # negative x, xp because xp decreases

        # Precision
        precision = tpc / (tpc + fpc)  # precision curve
        p[ci] = np.interp(-px, -conf[i], precision[:, 0], left=1)  # p at pr_score

        # AP from recall-precision curve
        for j in range(tp.shape[1]):
            ap[ci, j], mpre, mrec = compute_ap(recall[:, j], precision[:, j])
            if plot and j == 0:
                py.append(np.interp(px, mrec, mpre))  # precision at mAP@0.5

    # Compute F1 (harmonic mean of precision and recall)
    f1 = 2 * p * r / (p + r + eps)
    names = [v for k, v in names.items() if k in unique_classes]  # list: only classes that have data
    names = dict(enumerate(names))  # to dict

            # 序列化变量并保存到文件
    with open('metrics.pkl', 'wb') as f:
        pickle.dump((px,py,p, r, f1, ap), f)

    if plot:
        plot_pr_curve(px, py, ap, Path(save_dir) / f'{prefix}PR_curve.png', names)
        plot_mc_curve(px, f1, Path(save_dir) / f'{prefix}F1_curve.png', names, ylabel='F1')
        plot_mc_curve(px, p, Path(save_dir) / f'{prefix}P_curve.png', names, ylabel='Precision')
        plot_mc_curve(px, r, Path(save_dir) / f'{prefix}R_curve.png', names, ylabel='Recall')

    i = smooth(f1.mean(0), 0.1).argmax()  # max F1 index
    p, r, f1 = p[:, i], r[:, i], f1[:, i]
    tp = (r * nt).round()  # true positives
    fp = (tp / (p + eps) - tp).round()  # false positives




    return tp, fp, p, r, f1, ap, unique_classes.astype(int)
```

(3）运行val.py，配置好data.yaml和权重文件 weights两个参数

```
python val.py --data yolov5/dataset/data.yaml --weights yolov5/runs/train/exp3/weights/best.pt
```

运行后在文件目录，可以看到pr.pkl和metrcs.pkl这两个文件。



名称	修改时间
classify	5 小时前
data	5 小时前
dataset	5 小时前
models	5 小时前
runs	4 小时前
segment	5 小时前
utils	2 小时前
benchmar...	5 小时前
CITATION....	5 小时前
CONTRIB...	5 小时前
data.pkl	4 小时前
detect.py	5 小时前
export.py	5 小时前
hubconf.py	5 小时前
LICENSE	5 小时前
metrics.pkl	11 分钟前
pr.pkl	11 分钟前
README....	5 小时前
README.z...	5 小时前
requireme...	5 小时前
setup.cfg	5 小时前
train.py	5 小时前
tutorial.ip...	5 小时前
val.py	5 小时前
yolov5s.pt	4 小时前

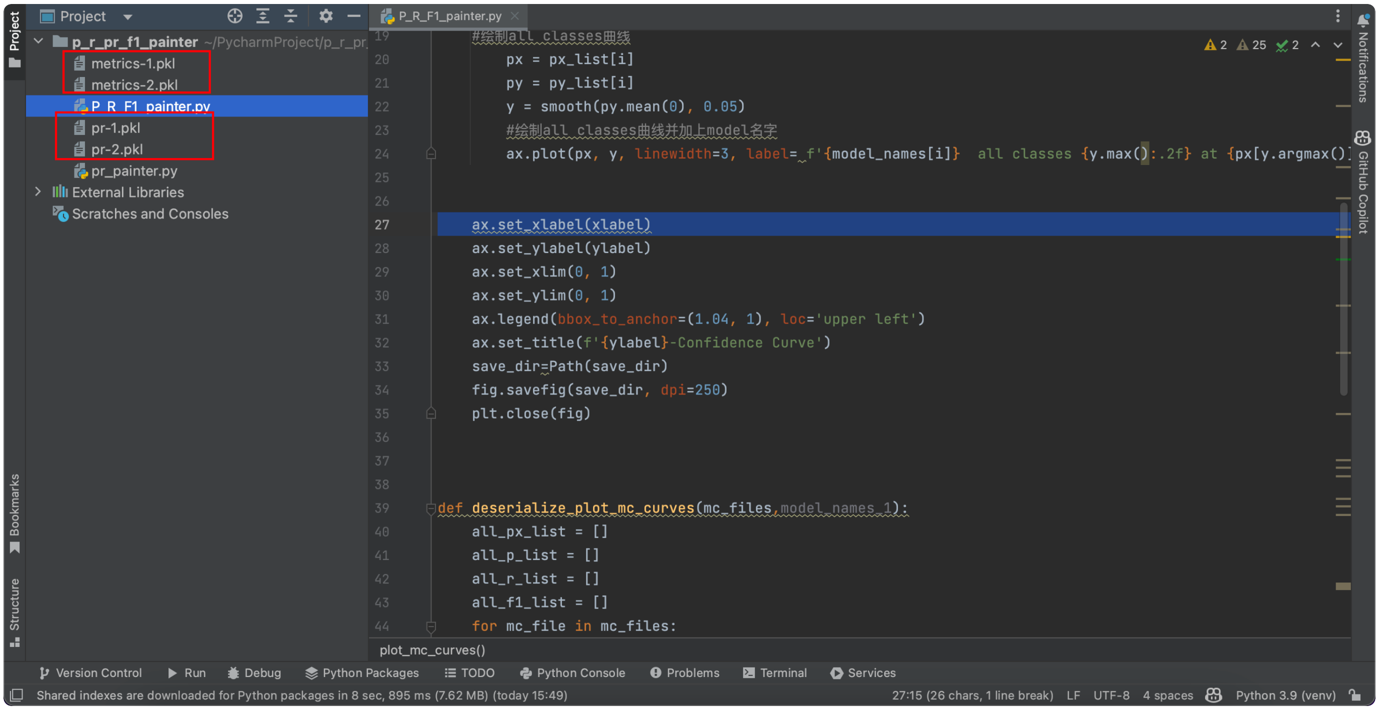
```python
# YOLOv5 🚀 by Ultralytics, AGPL-3.0 license
"""
Model validation metrics
"""

import pickle

import math
import warnings
from pathlib import Path

import matplotlib.pyplot as plt
import numpy as np
import torch

from utils import TryExcept, threaded


def fitness(x):
    # Model fitness as a weighted combination of metrics
    w = [0.0, 0.0, 0.1, 0.9]  # weights for [P, R, mAP@0.5, mAP@0.5:0.95]
    return (x[:, :4] * w).sum(1)


def smooth(y, f=0.05):
    # Box filter of fraction f
    nf = round(len(y) * f * 2) // 2 + 1  # number of filter elements (must be odd)
    p = np.ones(nf // 2)  # ones padding
    yp = np.concatenate((p * y[0], y, p * y[-1]), 0)  # y padded
    return np.convolve(yp, np.ones(nf) / nf, mode='valid')  # y-smoothed


def ap_per_class(tp, conf, pred_cls, target_cls, plot=False, save_dir='.', names=(), eps=1e-16, prefix=''):
    """ Compute the average precision, given the recall and precision curves.
    Source: https://github.com/rafaelpadilla/Object-Detection-Metrics.
    # Arguments
        tp:  True positives (nparray, nx1 or nx10).
        conf:  Objectness value from 0-1 (nparray).
        pred_cls:  Predicted object classes (nparray).
        target_cls:  True object classes (nparray).
        plot:  Plot precision-recall curve at mAP@0.5
        save_dir:  Plot save directory
    # Returns
```

这样第一步完成！

步骤二：使用pkl绘图

（1）重命名pkl文件

将得到的pkl文件放在项目目录下，并且重命名。如第一个模型得到的两个文件分别命名为pr-1.pkl,metrics-1.pkl.第二个数字就改成2，以此类推。



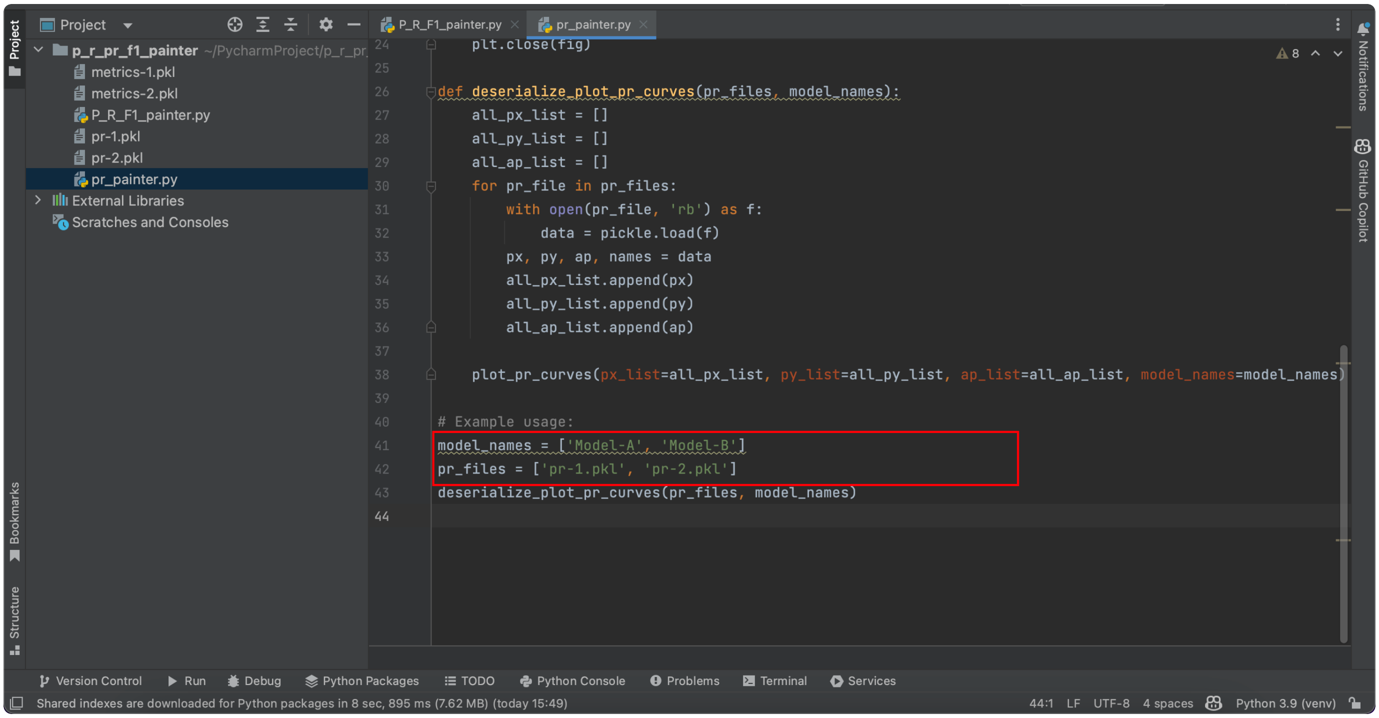
```
    #绘制all_classes曲线
        px = px_list[i]
        py = py_list[i]
        y = smooth(py.mean(0), 0.05)
        #绘制all_classes曲线并加上model名字
        ax.plot(px, y, linewidth=3, label= f'{model_names[i]}  all classes {y.max():.2f} at {px[y.argmax()]


    ax.set_xlabel(xlabel)
    ax.set_ylabel(ylabel)
    ax.set_xlim(0, 1)
    ax.set_ylim(0, 1)
    ax.legend(bbox_to_anchor=(1.04, 1), loc='upper left')
    ax.set_title(f'{ylabel}-Confidence Curve')
    save_dir=Path(save_dir)
    fig.savefig(save_dir, dpi=250)
    plt.close(fig)



def deserialize_plot_mc_curves(mc_files,model_names_1):
    all_px_list = []
    all_p_list = []
    all_r_list = []
    all_f1_list = []
    for mc_file in mc_files:
```

plot_mc_curves()

Shared indexes are downloaded for Python packages in 8 sec, 895 ms (7.62 MB) (today 15:49)

（2）绘制pr图

在在pr_painter.py中修改model_name为自己的模型名字和pkl的文件名。运行改py文件即可。

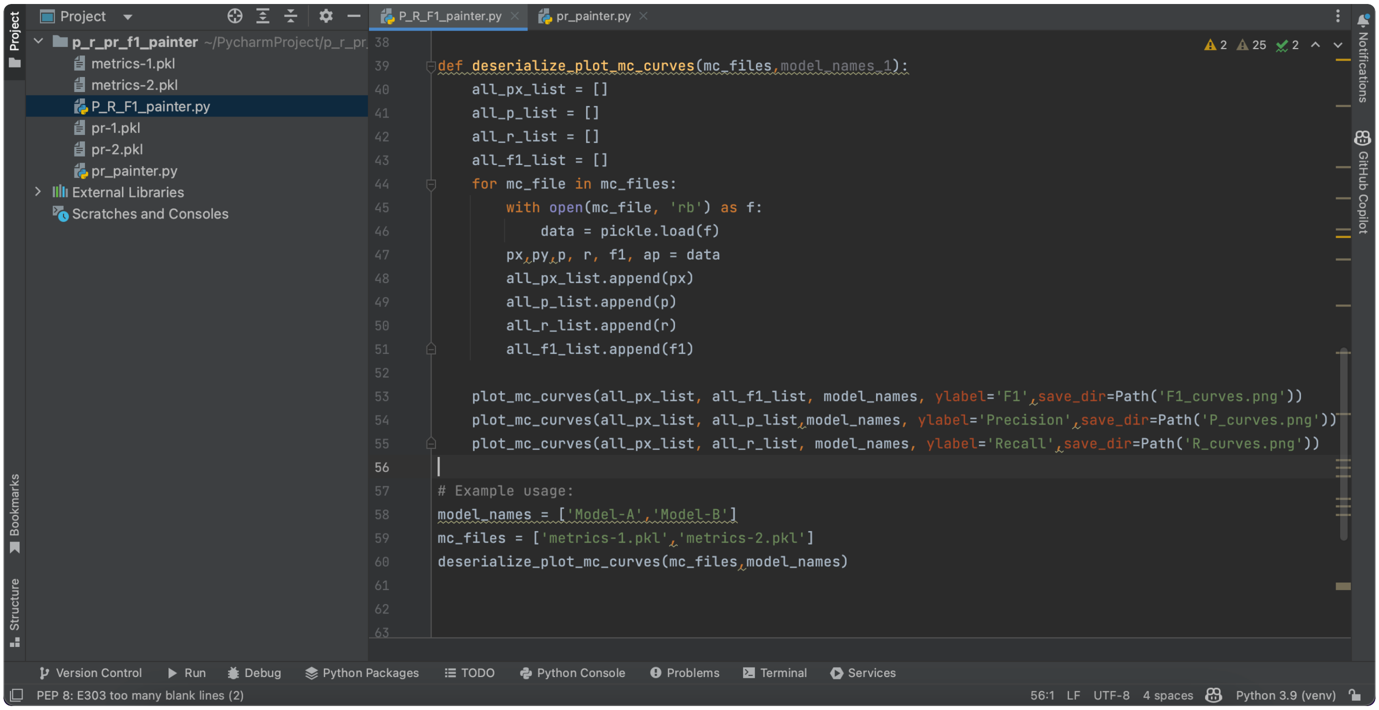


```
    plt.close(fig)

def deserialize_plot_pr_curves(pr_files, model_names):
    all_px_list = []
    all_py_list = []
    all_ap_list = []
    for pr_file in pr_files:
        with open(pr_file, 'rb') as f:
            data = pickle.load(f)
        px, py, ap, names = data
        all_px_list.append(px)
        all_py_list.append(py)
        all_ap_list.append(ap)

    plot_pr_curves(px_list=all_px_list, py_list=all_py_list, ap_list=all_ap_list, model_names=model_names)

# Example usage:
model_names = ['Model-A', 'Model-B']
pr_files = ['pr-1.pkl', 'pr-2.pkl']
deserialize_plot_pr_curves(pr_files, model_names)

```

Shared indexes are downloaded for Python packages in 8 sec, 895 ms (7.62 MB) (today 15:49)

（3）绘制P，R，F1图

同2，修改model_name为自己的模型名字和pkl的文件名。运行改py文件即可。



```python
def deserialize_plot_mc_curves(mc_files,model_names_1):
    all_px_list = []
    all_p_list = []
    all_r_list = []
    all_f1_list = []
    for mc_file in mc_files:
        with open(mc_file, 'rb') as f:
            data = pickle.load(f)
        px,py,p, r, f1, ap = data
        all_px_list.append(px)
        all_p_list.append(p)
        all_r_list.append(r)
        all_f1_list.append(f1)

    plot_mc_curves(all_px_list, all_f1_list, model_names, ylabel='F1',save_dir=Path('F1_curves.png'))
    plot_mc_curves(all_px_list, all_p_list,model_names, ylabel='Precision',save_dir=Path('P_curves.png'))
    plot_mc_curves(all_px_list, all_r_list, model_names, ylabel='Recall',save_dir=Path('R_curves.png'))

# Example usage:
model_names = ['Model-A','Model-B']
mc_files = ['metrics-1.pkl','metrics-2.pkl']
deserialize_plot_mc_curves(mc_files,model_names)
```

PEP 8: E303 too many blank lines (2)