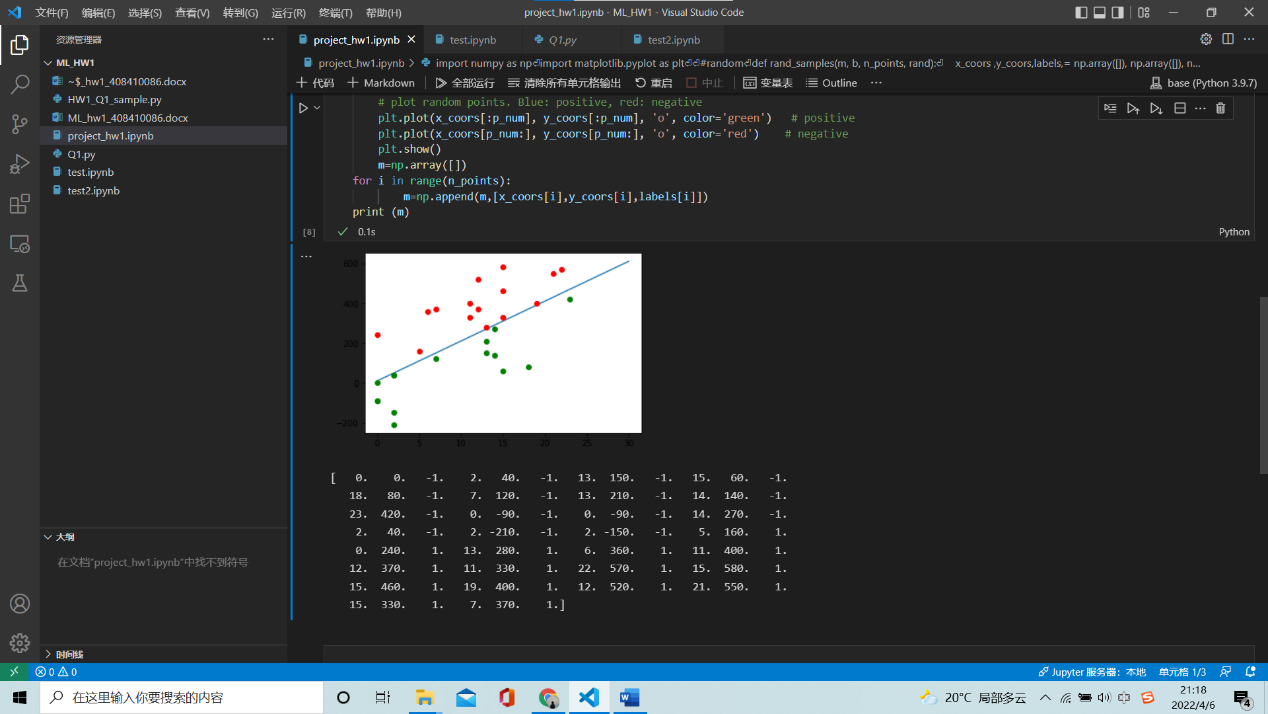
Report

常博愛 408410086 資工三

1. 執行描述：先設置random sample（），然後畫出已知的直線y=mx+b，之後再根據直線y的參數代入random函式，最後output每個點的坐標與對應的label值。

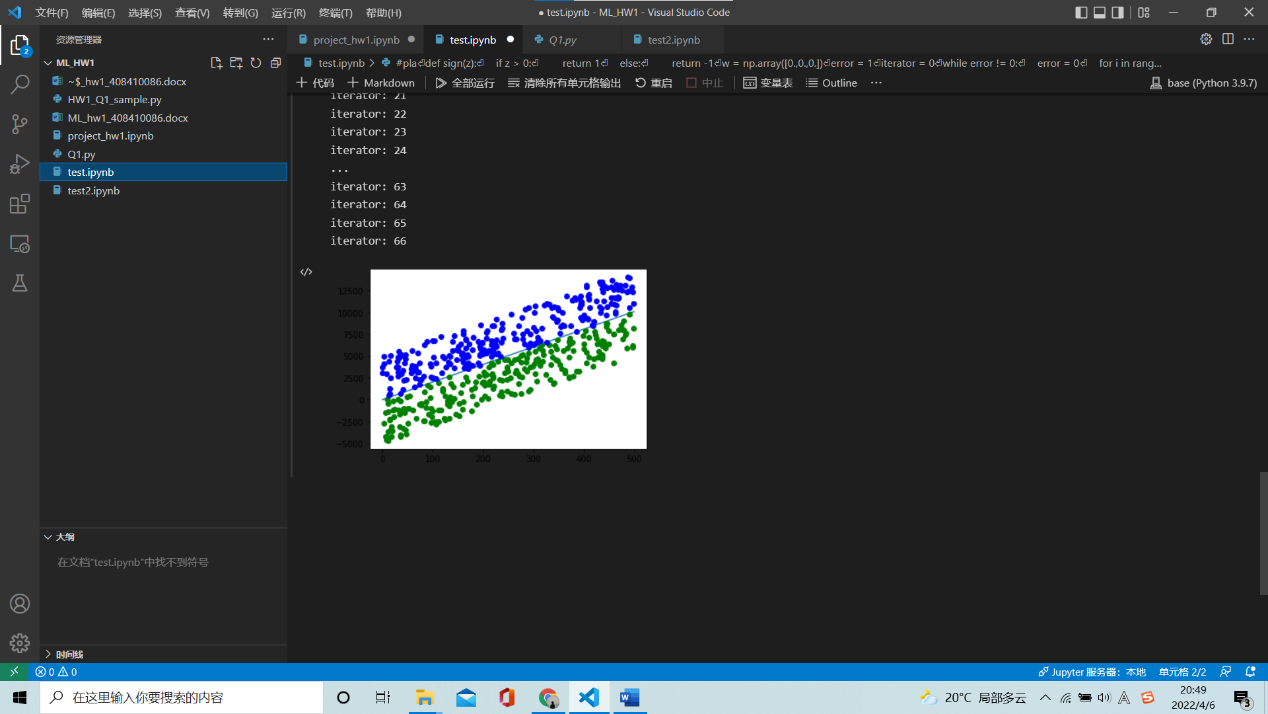
執行結果：如圖

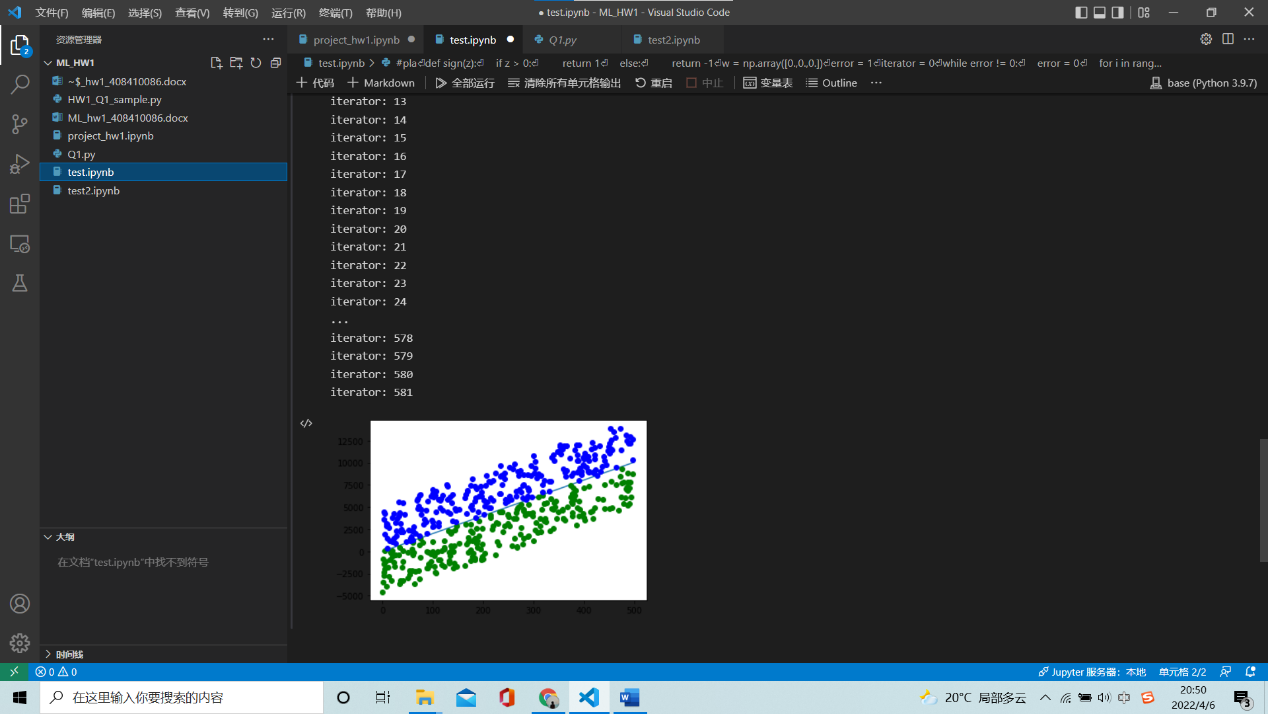


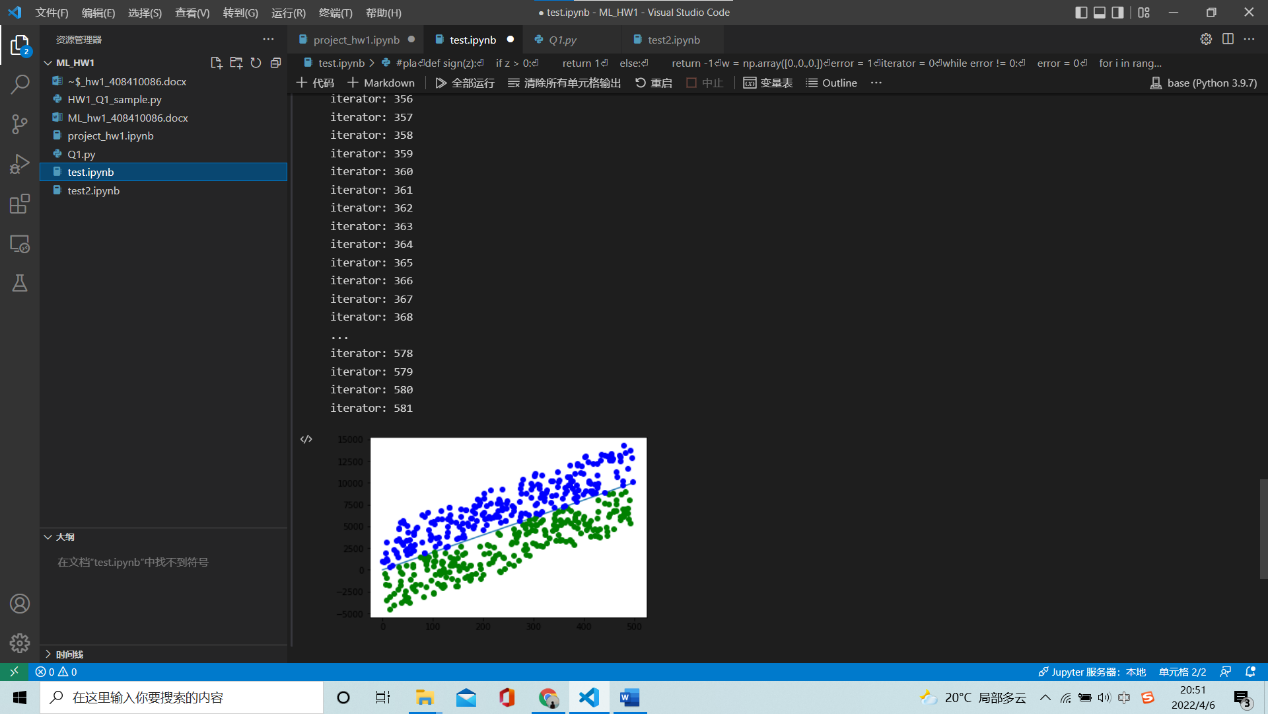
1. **執行描述**：

首先產生隨機亂數點共500（pos250,neg250）,然後撰寫激勵函數sign（），如果sign（z）>0;return 1,else -1.之後開始迴圈由w0=[0,0,0]開始遍歷每個點x計算w0\*x的內積，如果y是positive就相加，否則就相減。如果sign的結果與y相同，跳出迴圈，結束程式，畫出圖像。

**執行結果**：以500sample為例：average=（66+581+581）/3=409次

第一次：

第二次：

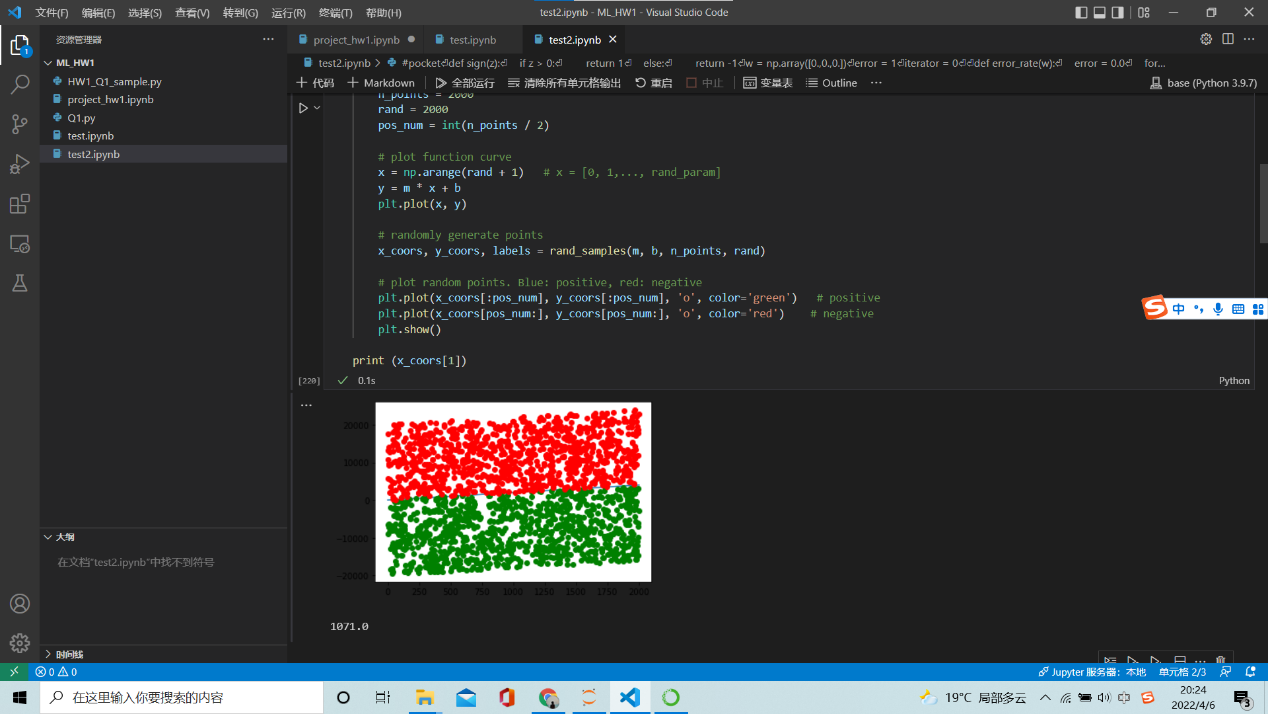
第三次：

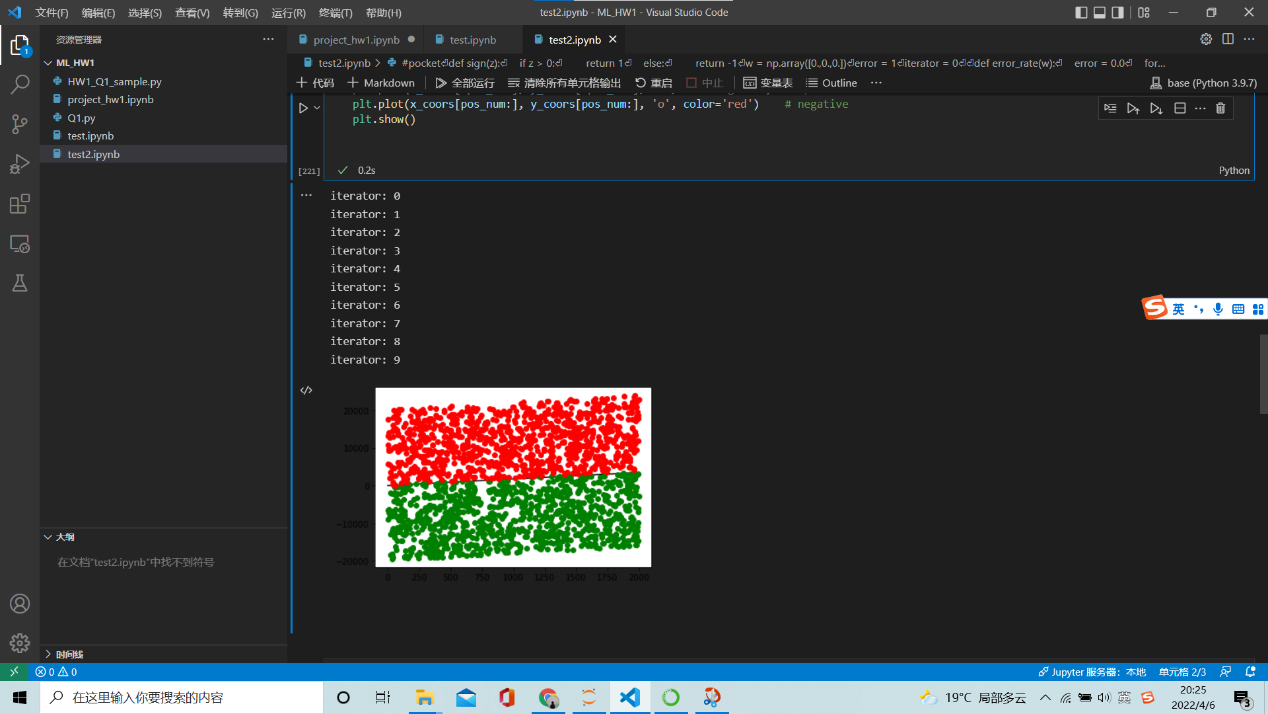
**總結與討論**：PLA每次迭代結果都不會對後面產生影響，即train結果不是收斂的，可能需要多次迭代才能實現分類，當data量過大時，不適用。

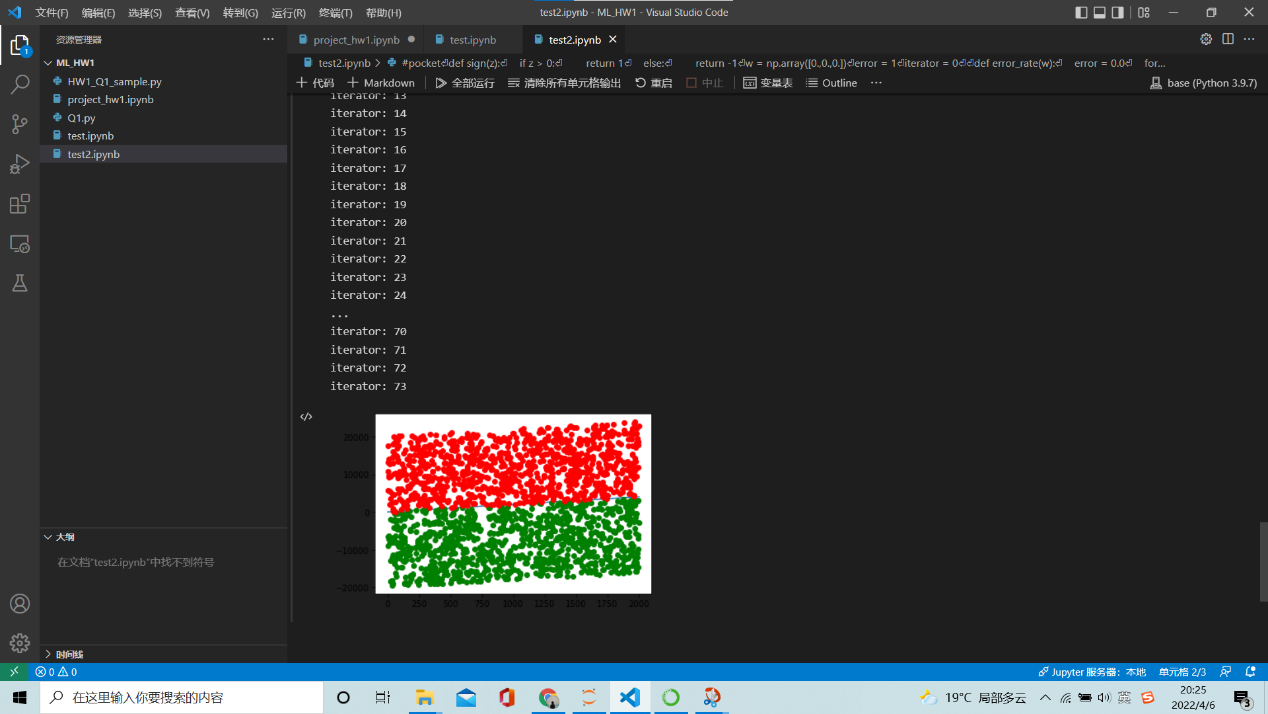
1. 執行描述：

框架基本與PLA一致，只是多撰寫了一個error（）function，如果weight（new）的error小於weight（old），就更新weight，並跳出function。

執行結果：如圖可知，該次實驗 生成隨機點時間為0.1s；pocket執行時間為0.2s,迭代9次；PLA執行時間為0.3s,迭代73次。







總結與討論：由於我本次實驗只顯示出最後一次分類的圖像，所以計算量要簡化許多，時間也減少很多，因此在共2000個sample的情況下時間依舊控制在很小的範圍；兩種演算法的時間差異也很小，但是可以根據迭代次數推斷，pocket的效率是高於PLA的，如果把每次迭代的結果都顯示出來，勢必PLA要花費更多時間，但無奈電腦算力不夠，無法執行題述的資料量。

Difficulty：公式的實現，可視化的處理，形態格式的轉換，套件不熟悉

Note：code之前為ipynb檔，後改為.py