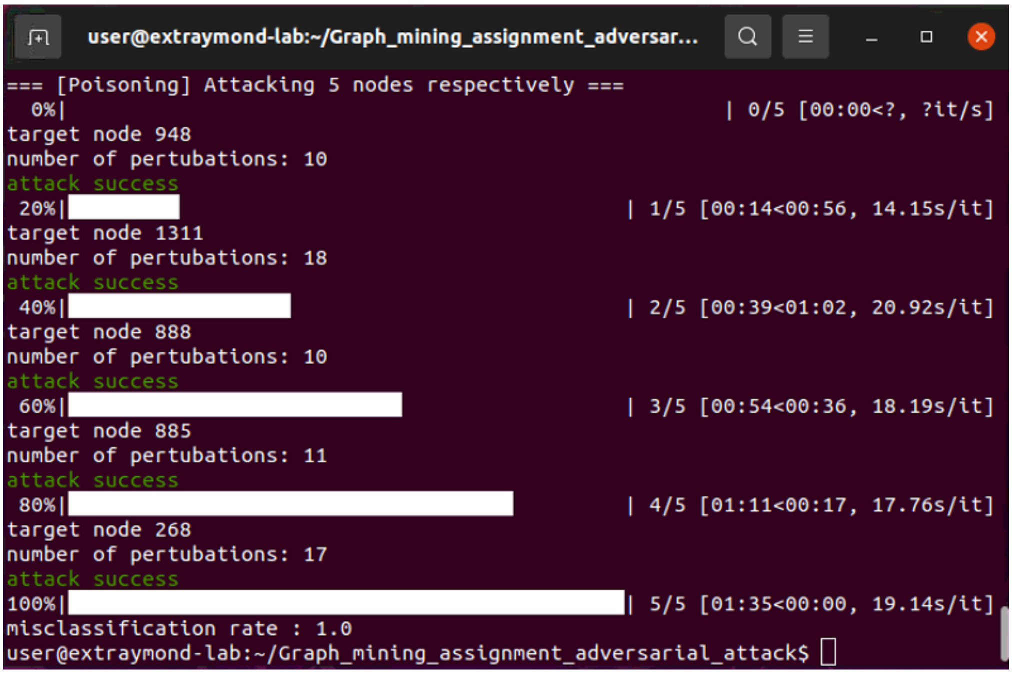
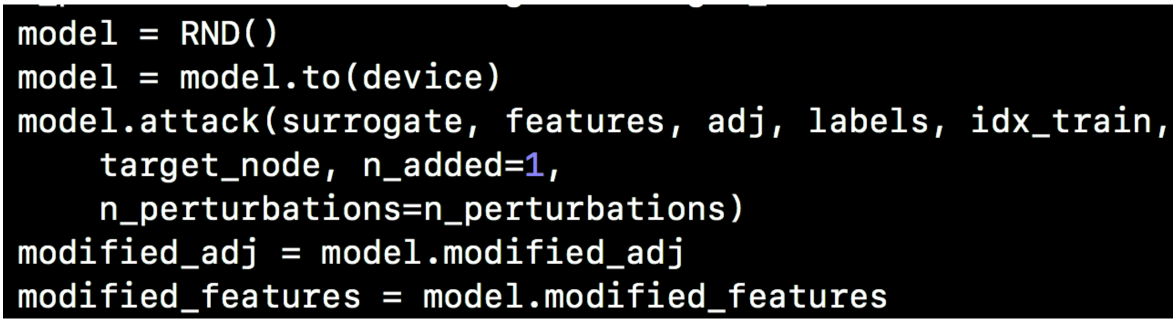
以下截圖為結果。



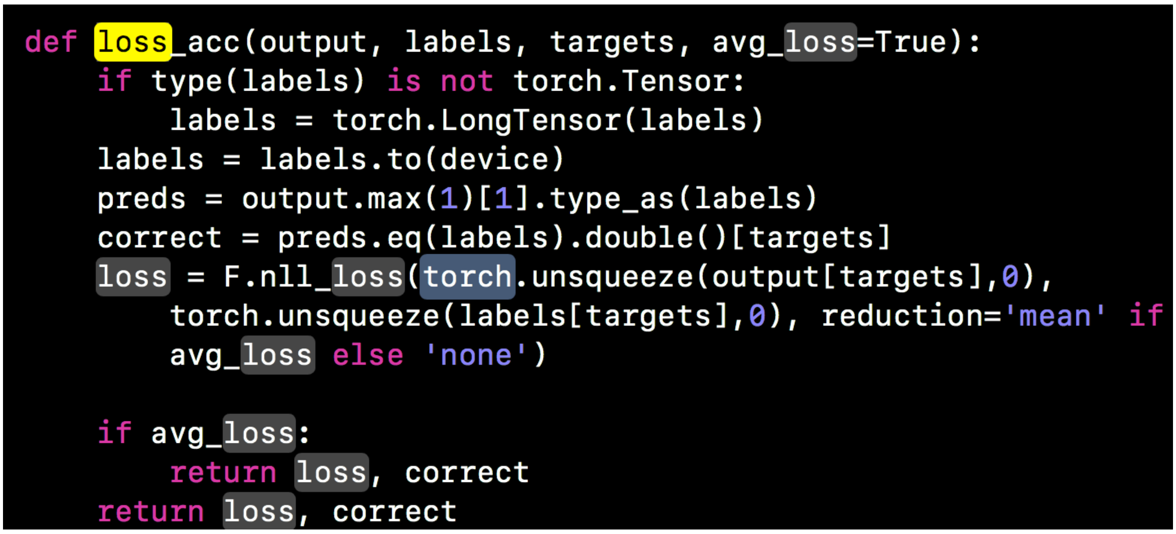
```
=== [Poisoning] Attacking 5 nodes respectively ===
  0%|                                                    | 0/5 [00:00<?, ?it/s]
target node 948
number of pertubations: 10
attack success
 20%|█████████                              | 1/5 [00:14<00:56, 14.15s/it]
target node 1311
number of pertubations: 18
attack success
 40%|██████████████████                     | 2/5 [00:39<01:02, 20.92s/it]
target node 888
number of pertubations: 10
attack success
 60%|███████████████████████████            | 3/5 [00:54<00:36, 18.19s/it]
target node 885
number of pertubations: 11
attack success
 80%|████████████████████████████████████   | 4/5 [01:11<00:17, 17.76s/it]
target node 268
number of pertubations: 17
attack success
100%|█████████████████████████████████████████████| 5/5 [01:35<00:00, 19.14s/it]
misclassification rate : 1.0
user@extraymond-lab:~/Graph_mining_assignment_adversarial_attack$ 
```

Main.py :首先將要被攻擊的model傳進attacker。



```
model = RND()
model = model.to(device)
model.attack(surrogate, features, adj, labels, idx_train,
    target_node, n_added=1,
    n_perturbations=n_perturbations)
modified_adj = model.modified_adj
modified_features = model.modified_features
```

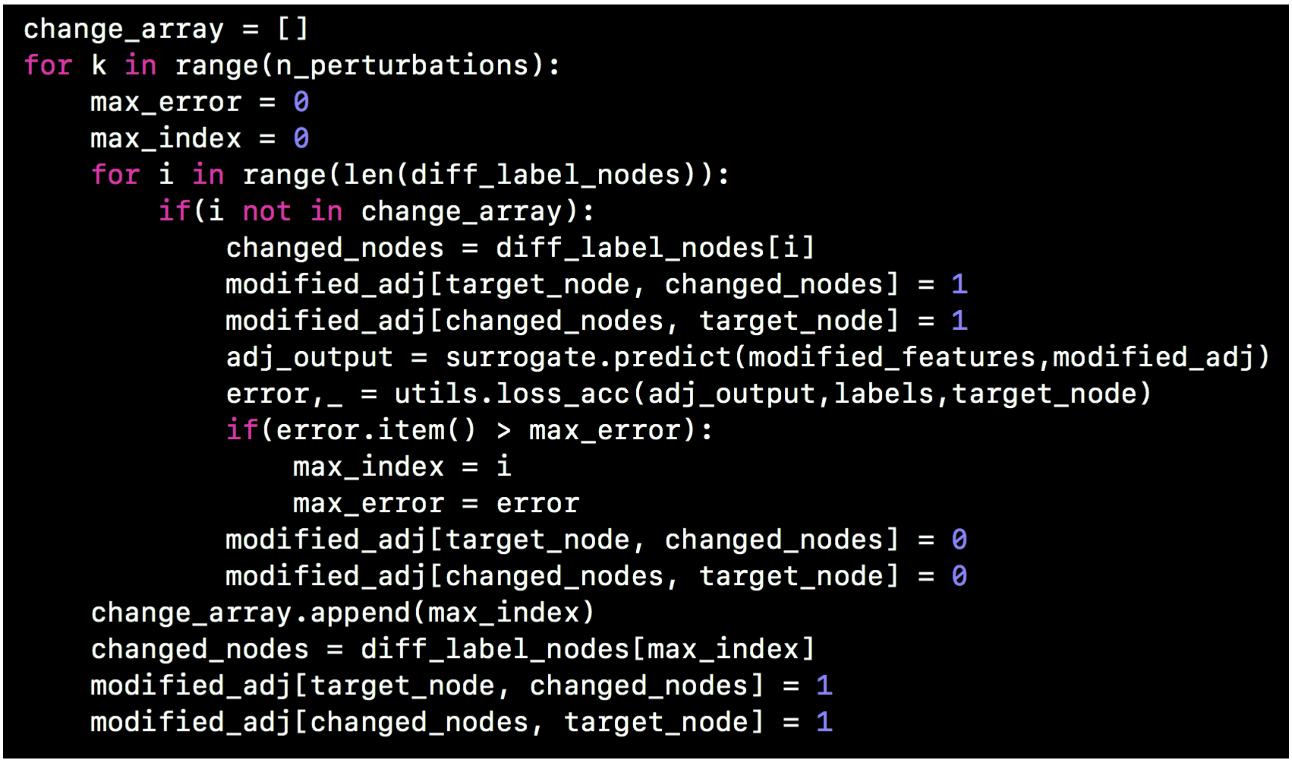
Utils.py :然後更改loss_acc一些形態上面的問題，以便計算loss。



```
def loss_acc(output, labels, targets, avg_loss=True):
    if type(labels) is not torch.Tensor:
        labels = torch.LongTensor(labels)
    labels = labels.to(device)
    preds = output.max(1)[1].type_as(labels)
    correct = preds.eq(labels).double()[targets]
    loss = F.nll_loss(torch.unsqueeze(output[targets],0),
        torch.unsqueeze(labels[targets],0), reduction='mean' if
        avg_loss else 'none')

    if avg_loss:
        return loss, correct
    return loss, correct
```

Attacker.py :演算法的地方主要是這樣，拿取那些沒有跟target node連接且label跟target node不一樣的node(跟助教sample code的diff_label_node一樣)，因為這樣不僅能夠省下一些時間，並且達到更有效率的攻擊。
再來是將這些node一個一個連接到target node，每連接一個就計算loss，並且記錄最大loss的node，然後將此node回復到原本的狀態，直到跑遍所有的diff_label_node，就可以將此造成最大loss的node接到target node上，並且記錄下來這是哪一個node，此步驟將花費一個budge。
接著就是重複執行上面的算法budgets次，就成功了。



```
change_array = []
for k in range(n_perturbations):
    max_error = 0
    max_index = 0
    for i in range(len(diff_label_nodes)):
        if(i not in change_array):
            changed_nodes = diff_label_nodes[i]
            modified_adj[target_node, changed_nodes] = 1
            modified_adj[changed_nodes, target_node] = 1
            adj_output = surrogate.predict(modified_features,modified_adj)
            error,_ = utils.loss_acc(adj_output,labels,target_node)
            if(error.item() > max_error):
                max_index = i
                max_error = error
            modified_adj[target_node, changed_nodes] = 0
            modified_adj[changed_nodes, target_node] = 0
    change_array.append(max_index)
    changed_nodes = diff_label_nodes[max_index]
    modified_adj[target_node, changed_nodes] = 1
    modified_adj[changed_nodes, target_node] = 1
```