HW10

姓名：李懿恩 學號：107403510 系級：資管 3A

Colab 連結: https://colab.research.google.com/drive/1KNYn7PB0-JdvbimlilSkmIKINEB4jfag?usp=sharing

以下是做二元分類(target label :’ Malicious’ , true:惡意 apk, false:正常 apk)

資料集 csv 檔 : apk_dataset.csv

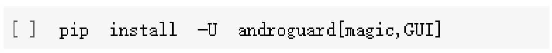
1. 從原始 APK 取特徵

先將 apk 壓縮檔存到自己的雲端，在 colab 中掛接自己雲端以取得 apk 壓縮檔，然後利用 Androguard 的 get_permissions 方法取得 apk 的 permission。不過此方法無法取得 shell 檔的 apk 檔案，所以 SMS, Adware, Banking 的檔案皆各有一個 apk 無法取得 permission。取得全部 permssion 後，SMS, Adware, Banking, Benign，共 13 個，各自存成 csv 檔，方便後面前處理不用再重新解壓縮、取 permission 一次。

(取 permission 方法參考：https://www.jianshu.com/p/670023af50f6)

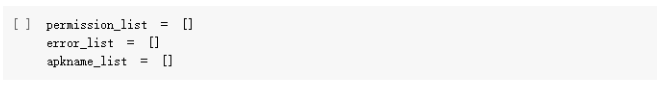
程式碼：

Step1:安裝 Androguard

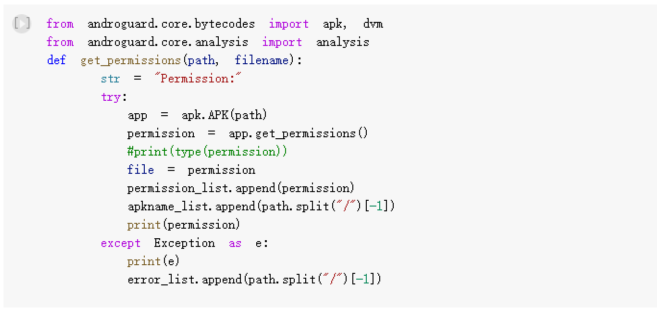


```
[ ]  pip install -U androguard[magic,GUI]
```

Step2:自訂 get_permissions 函數取 permission



```
[ ]  permission_list = []
     error_list = []
     apkname_list = []
```



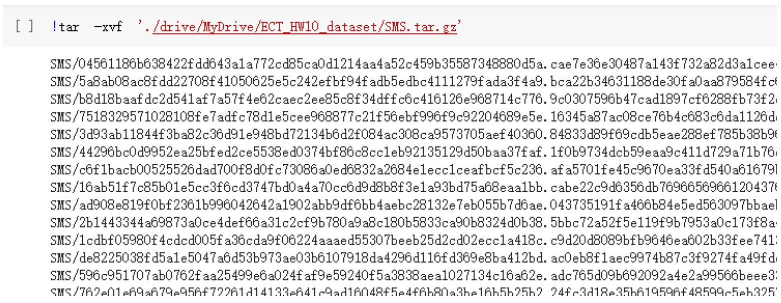
```
[ ]  from androguard.core.bytecodes import apk, dvm
     from androguard.core.analysis import analysis
     def get_permissions(path, filename):
             str = "Permission:"
             try:
                 app = apk.APK(path)
                 permission = app.get_permissions()
                 #print(type(permission))
                 file = permission
                 permission_list.append(permission)
                 apkname_list.append(path.split("/")[-1])
                 print(permission)
             except Exception as e:
                 print(e)
                 error_list.append(path.split("/")[-1])
```

Step3:解壓縮全部 13 個檔案(SMS, Adware, Banking, Benign10 個)，並各自用 get_permissions 函數取得 permission，最後存成 csv 檔

(以下取特徵的部分可以直接看 ipynb 檔或 colab，pdf 檔的這部分可以跳過直接從第 2 題前處理看)

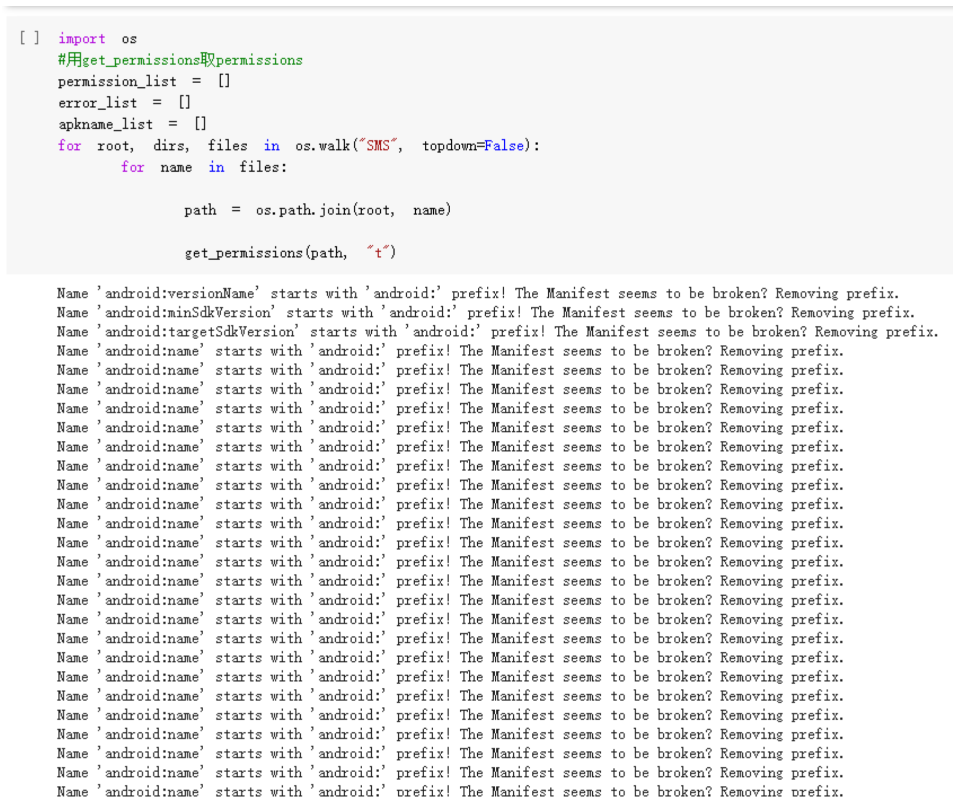
- SMS

  解壓縮



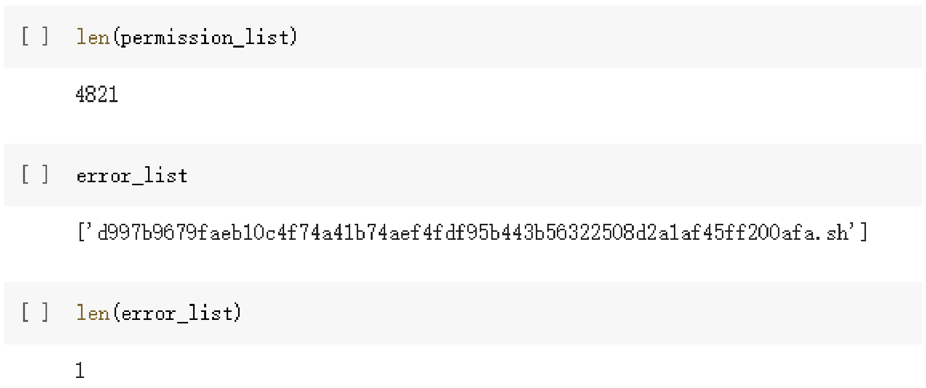
```
[ ]  !tar -xvf './drive/MyDrive/ECT_HW10_dataset/SMS.tar.gz'

     SMS/04561186b638422fdd643a1a772cd85ca0d1214aa4a52c459b35587348880d5a.cae7e36e30487a143f732a82d3a1cee
     SMS/5a8ab08ac8fdd22708f41050625e5c242efbf94fadb5edbc4111279fada3f4a9.bca22b34631188de30fa0aa879584fc
     SMS/b8d18baafdc2d541af7a57f4e62caec2ee85c8f34dffc6c416126e968714c776.9c0307596b47cad1897cf6288fb73f2
     SMS/7518329571028108fe7adfc78d1e5cee968877c21f56ebf996f9c92204689e5e.16345a87ac08ce76b4c683c6da1126d
     SMS/3d93ab11844f3ba82c36d91e948bd72134b6d2f084ac308ca9573705aef40360.84833d89f69cdb5eae288ef785b38b9
     SMS/44296bc0d9952ea25bfed2ce5538ed0374bf86c8cc1eb92135129d50baa37faf.1f0b9734dcb59eaa9c411d729a71b76
     SMS/c6f1bacb00525526dad700f8d0fc73086a0ed6832a2684e1ecc1ceafbcf5c236.afa5701fe45c9670ea33fd540a61679
     SMS/16ab51f7c85b01e5cc3f6cd3747bd0a4a70cc6d9d8b8f3e1a93bd75a68eaa1bb.cabe22c9d6356db769665696612043
     SMS/ad908e819f0bf2361b996042642a1902abb9df6bb4aebc28132e7eb055b7d6ae.043735191fa466b84e5ed563097bbae
     SMS/2b1443344a69873a0ce4def66a31c2cf9b780a9a8c180b5833ca90b8324d0b38.5bbc72a52f5e119f9b7953a0c173f8a
     SMS/1cdbf05980f4cdcd005fa36cda9f06224aaaed55307beeb25d2cd02ecc1a418c.c9d20d8089bfb9646ea602b33fee741
     SMS/de8225038fd5a1e5047a6d53b973ae03b6107918da4296d116fd369e8ba412bd.ac0eb8f1aec9974b87c3f9274fa49fd
     SMS/596c951707ab0762faa25499e6a024faf9e59240f5a3838aea1027134c16a62e.adc765d09b692092a4e2a99566beee3
     SMS/762e01e69a679e956f72261d14133e641c9ad16048f5e4f6b80a3be16b5b25b2.24fc3d18e35b619596f48599c5eb325
```

  用自訂函數取 permissions



```
[ ] import os
    #用get_permissions取permissions
    permission_list = []
    error_list = []
    apkname_list = []
    for root, dirs, files in os.walk("SMS", topdown=False):
        for name in files:

            path = os.path.join(root, name)

            get_permissions(path, "t")
```

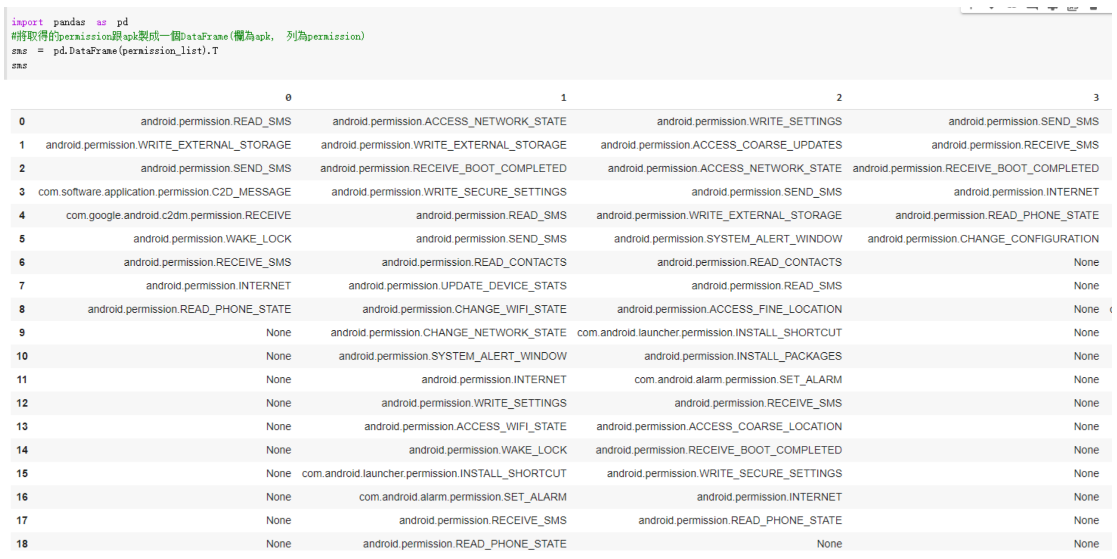
```
Name 'android:versionName' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:minSdkVersion' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:targetSdkVersion' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:name' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:name' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:name' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:name' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:name' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:name' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:name' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:name' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:name' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:name' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:name' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:name' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:name' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:name' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:name' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:name' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:name' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:name' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:name' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:name' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:name' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:name' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:name' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:name' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
```



```
[ ] len(permission_list)

    4821

[ ] error_list

    ['d997b9679faeb10c4f74a41b74aef4fdf95b443b56322508d2a1af45ff200afa.sh']

[ ] len(error_list)

    1
```

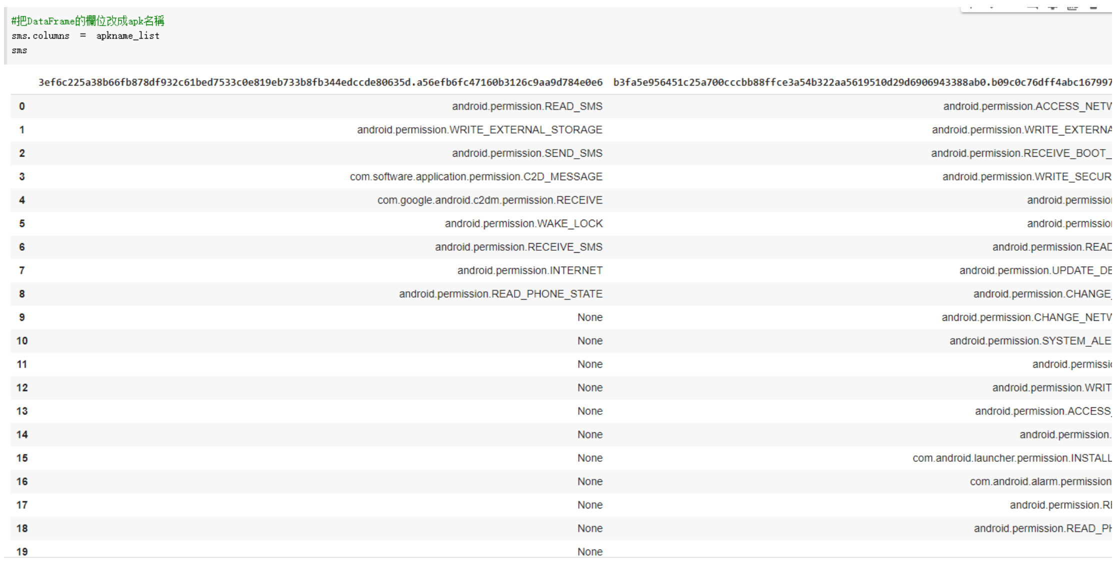
- 總共從 4821 個 apk 取 permissions，其中這一個 d997b9679faeb10c4f74a41b74aef4fdf95b443b56322508d2a1af45ff200afa.sh 是 shell 檔，所以在取 permission 時有錯誤產生

將 permissions 和 apk 製成 DataFrame



```
import pandas as pd
#將取得的permission跟apk製成一個DataFrame(欄為apk, 列為permission)
sms = pd.DataFrame(permission_list).T
sms
```

	0	1	2	3
0	android.permission.READ_SMS	android.permission.ACCESS_NETWORK_STATE	android.permission.WRITE_SETTINGS	android.permission.SEND_SMS
1	android.permission.WRITE_EXTERNAL_STORAGE	android.permission.WRITE_EXTERNAL_STORAGE	android.permission.ACCESS_COARSE_UPDATES	android.permission.RECEIVE_SMS
2	android.permission.SEND_SMS	android.permission.RECEIVE_BOOT_COMPLETED	android.permission.ACCESS_NETWORK_STATE	android.permission.RECEIVE_BOOT_COMPLETED
3	com.software.application.permission.C2D_MESSAGE	android.permission.WRITE_SECURE_SETTINGS	android.permission.SEND_SMS	android.permission.INTERNET
4	com.google.android.c2dm.permission.RECEIVE	android.permission.READ_SMS	android.permission.WRITE_EXTERNAL_STORAGE	android.permission.READ_PHONE_STATE
5	android.permission.WAKE_LOCK	android.permission.SEND_SMS	android.permission.SYSTEM_ALERT_WINDOW	android.permission.CHANGE_CONFIGURATION
6	android.permission.RECEIVE_SMS	android.permission.READ_CONTACTS	android.permission.READ_CONTACTS	None
7	android.permission.INTERNET	android.permission.UPDATE_DEVICE_STATS	android.permission.READ_SMS	None
8	android.permission.READ_PHONE_STATE	android.permission.CHANGE_WIFI_STATE	android.permission.ACCESS_FINE_LOCATION	None
9	None	android.permission.CHANGE_NETWORK_STATE	com.android.launcher.permission.INSTALL_SHORTCUT	None
10	None	android.permission.SYSTEM_ALERT_WINDOW	android.permission.INSTALL_PACKAGES	None
11	None	android.permission.INTERNET	com.android.alarm.permission.SET_ALARM	None
12	None	android.permission.WRITE_SETTINGS	android.permission.RECEIVE_SMS	None
13	None	android.permission.ACCESS_WIFI_STATE	android.permission.ACCESS_COARSE_LOCATION	None
14	None	android.permission.WAKE_LOCK	android.permission.RECEIVE_BOOT_COMPLETED	None
15	None	com.android.launcher.permission.INSTALL_SHORTCUT	android.permission.WRITE_SECURE_SETTINGS	None
16	None	com.android.alarm.permission.SET_ALARM	android.permission.INTERNET	None
17	None	android.permission.RECEIVE_SMS	android.permission.READ_PHONE_STATE	None
18	None	android.permission.READ_PHONE_STATE	None	None



```
#把DataFrame的欄位改成apk名稱
sms.columns = apkname_list
sms
```

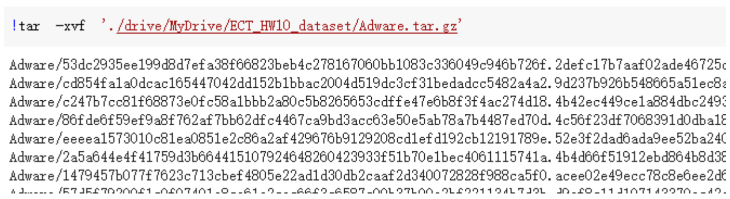
	3ef6c225a38b66fb878df932c61bed7533c0e819eb733b8fb344edccde80635d.a56efb6fc47160b3126c9aa9d784e0e6	b3fa5e956451c25a700cccbb88ffce3a54b322aa5619510d29d6906943388ab0.b09c0c76dff4abc167997
0	android.permission.READ_SMS	android.permission.ACCESS_NETV
1	android.permission.WRITE_EXTERNAL_STORAGE	android.permission.WRITE_EXTERNA
2	android.permission.SEND_SMS	android.permission.RECEIVE_BOOT_
3	com.software.application.permission.C2D_MESSAGE	android.permission.WRITE_SECUR
4	com.google.android.c2dm.permission.RECEIVE	android.permissio
5	android.permission.WAKE_LOCK	android.permissio
6	android.permission.RECEIVE_SMS	android.permission.READ
7	android.permission.INTERNET	android.permission.UPDATE_DE
8	android.permission.READ_PHONE_STATE	android.permission.CHANGE
9	None	android.permission.CHANGE_NETV
10	None	android.permission.SYSTEM_ALE
11	None	android.permissi
12	None	android.permission.WRIT
13	None	android.permission.ACCESS
14	None	android.permission.
15	None	com.android.launcher.permission.INSTALL
16	None	com.android.alarm.permission
17	None	android.permission.RI
18	None	android.permission.READ_PH
19	None	

存成 csv 檔

```
#存成csv檔
sms.to_csv("sms_permission.csv")
```

- Adware

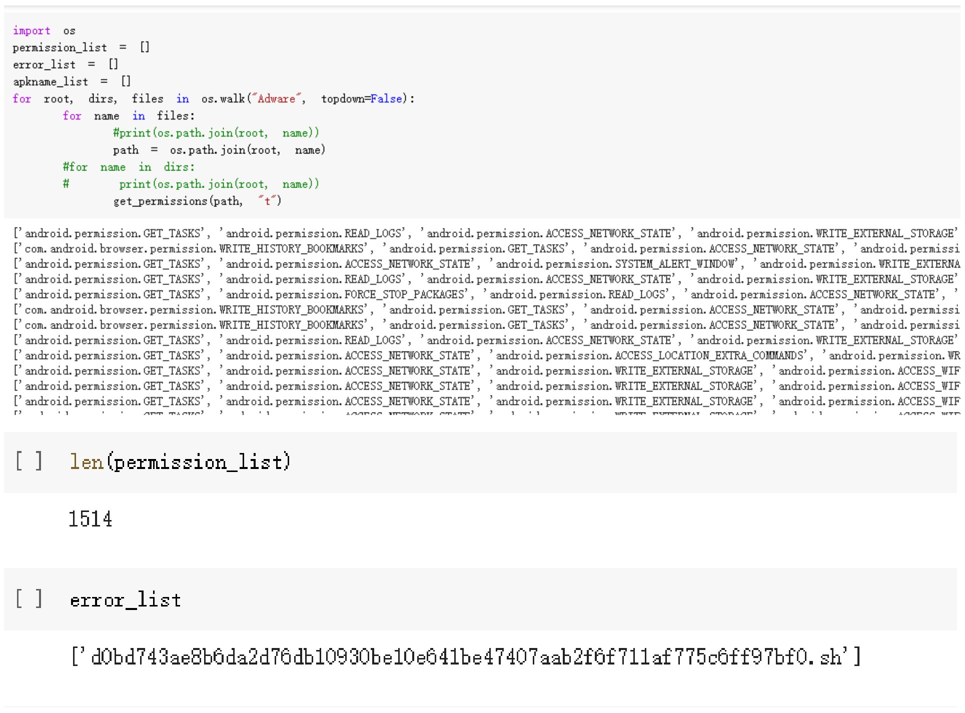
解壓縮



```
!tar -xvf './drive/MyDrive/ECT_HW10_dataset/Adware.tar.gz'
```

```
Adware/53dc2935ee199d8d7efa38f66823beb4c278167060bb1083c336049c946b726f.2defc17b7aaf02ade46725c
Adware/cd854fa1a0dcac165447042dd152b1bbac2004d519dc3cf31bedadcc5482a4a2.9d237b926b548665a51ec8a
Adware/c247b7cc81f68873e0fc58a1bbb2a80c5b8265653cdffe47e6b8f3f4ac274d18.4b42ec449ce1a884dbc2493
Adware/86fde6f59ef9a8f762af7bb62dfc4467ca9bd3acc63e50e5ab78a7b4487ed70d.4c56f23df7068391d0dba18
Adware/eeeea1573010c81ea0851e2c86a2af429676b9129208cd1efd192cb12191789e.52e3f2dad6ada9ee52ba240
Adware/2a5a644e4f41759d3b6644151079246482604239 33f51b70e1bec4061115741a.4b4d66f51912ebd864b8d38
Adware/1479457b077f7623c713cbef4805e22ad1d30db2caaf2d340072828f988ca5f0.acee02e49ecc78c8e6ee2d6
[illegible]
```

用自訂函數取 permissions



```
import os
permission_list = []
error_list = []
apkname_list = []
for root, dirs, files in os.walk("Adware", topdown=False):
        for name in files:
                #print(os.path.join(root, name))
                path = os.path.join(root, name)
        #for name in dirs:
        #        print(os.path.join(root, name))
                get_permissions(path, "t")
```

```
['android.permission.GET_TASKS', 'android.permission.READ_LOGS', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.WRITE_EXTERNAL_STORAGE'
['com.android.browser.permission.WRITE_HISTORY_BOOKMARKS', 'android.permission.GET_TASKS', 'android.permission.ACCESS_NETWORK_STATE', 'android.permissi
['android.permission.GET_TASKS', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.SYSTEM_ALERT_WINDOW', 'android.permission.WRITE_EXTERNA
['android.permission.GET_TASKS', 'android.permission.READ_LOGS', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.WRITE_EXTERNAL_STORAGE'
['android.permission.GET_TASKS', 'android.permission.FORCE_STOP_PACKAGES', 'android.permission.READ_LOGS', 'android.permission.ACCESS_NETWORK_STATE', '
['com.android.browser.permission.WRITE_HISTORY_BOOKMARKS', 'android.permission.GET_TASKS', 'android.permission.ACCESS_NETWORK_STATE', 'android.permissi
['com.android.browser.permission.WRITE_HISTORY_BOOKMARKS', 'android.permission.GET_TASKS', 'android.permission.ACCESS_NETWORK_STATE', 'android.permissi
['android.permission.GET_TASKS', 'android.permission.READ_LOGS', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.WRITE_EXTERNAL_STORAGE'
['android.permission.GET_TASKS', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.ACCESS_LOCATION_EXTRA_COMMANDS', 'android.permission.WR
['android.permission.GET_TASKS', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.ACCESS_WIF
['android.permission.GET_TASKS', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.ACCESS_WIF
['android.permission.GET_TASKS', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.ACCESS_WIF
```

```
[ ]  len(permission_list)
```

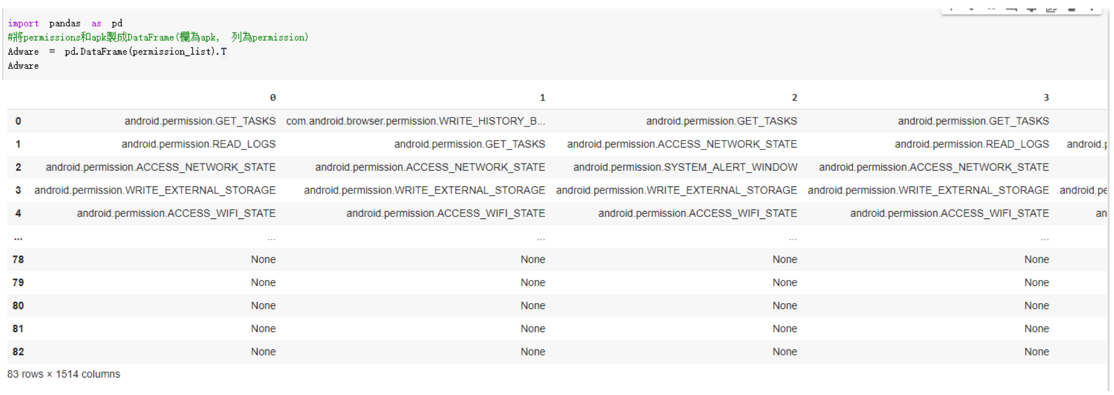
```
1514
```

```
[ ]  error_list
```

```
['d0bd743ae8b6da2d76db10930be10e641be47407aab2f6f711af775c6ff97bf0.sh']
```

- 總共從 1514 個 apk 取 permissions，其中這一個 d0bd743ae8b6da2d76db10930be10e641be47407aab2f6f711af775c6ff97bf0.sh 是 shell 檔，所以在取 permission 時有錯誤產生

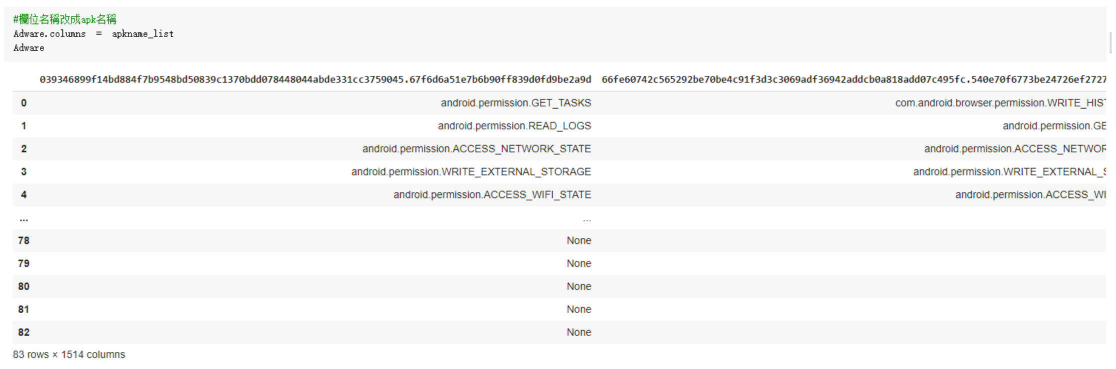
將 permissions 和 apk 製成 DataFrame



```
import pandas as pd
#將permissions和apk製成DataFrame(欄為apk, 列為permission)
Adware = pd.DataFrame(permission_list).T
Adware
```

	0	1	2	3	
0	android.permission.GET_TASKS	com.android.browser.permission.WRITE_HISTORY_B...	android.permission.GET_TASKS	android.permission.GET_TASKS	
1	android.permission.READ_LOGS	android.permission.GET_TASKS	android.permission.ACCESS_NETWORK_STATE	android.permission.READ_LOGS	android.p
2	android.permission.ACCESS_NETWORK_STATE	android.permission.ACCESS_NETWORK_STATE	android.permission.SYSTEM_ALERT_WINDOW	android.permission.ACCESS_NETWORK_STATE	
3	android.permission.WRITE_EXTERNAL_STORAGE	android.permission.WRITE_EXTERNAL_STORAGE	android.permission.WRITE_EXTERNAL_STORAGE	android.permission.WRITE_EXTERNAL_STORAGE	android.pe
4	android.permission.ACCESS_WIFI_STATE	android.permission.ACCESS_WIFI_STATE	android.permission.ACCESS_WIFI_STATE	android.permission.ACCESS_WIFI_STATE	an
...	...	...	...	...	
78	None	None	None	None	
79	None	None	None	None	
80	None	None	None	None	
81	None	None	None	None	
82	None	None	None	None	

83 rows × 1514 columns

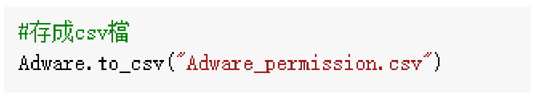


```python
#欄位名稱改成apk名稱
Adware.columns = apkname_list
Adware
```

	039346899f14bd884f7b9548bd50839c1370bdd078448044abde331cc3759045.67f6d6a51e7b6b90ff839d0fd9be2a9d	66fe60742c565292be70be4c91f3d3c3069adf36942addcb0a818add07c495fc.540e70f6773be24726ef2727
0	android.permission.GET_TASKS	com.android.browser.permission.WRITE_HIS
1	android.permission.READ_LOGS	android.permission.GE
2	android.permission.ACCESS_NETWORK_STATE	android.permission.ACCESS_NETWOR
3	android.permission.WRITE_EXTERNAL_STORAGE	android.permission.WRITE_EXTERNAL_
4	android.permission.ACCESS_WIFI_STATE	android.permission.ACCESS_WI
...	...	
78	None	
79	None	
80	None	
81	None	
82	None	

83 rows × 1514 columns

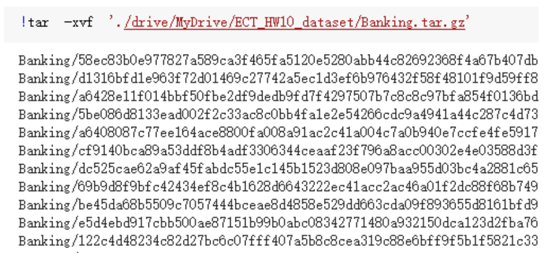
存成 csv 檔



```python
#存成csv檔
Adware.to_csv("Adware_permission.csv")
```

- ## Banking

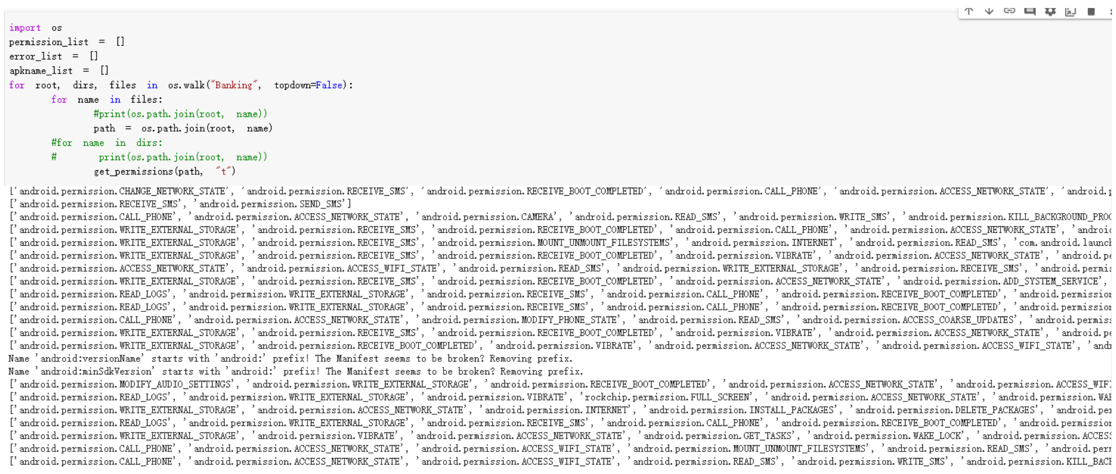
解壓縮



```
!tar -xvf './drive/MyDrive/ECT_HW10_dataset/Banking.tar.gz'
```

```
Banking/58ec83b0e977827a589ca3f465fa5120e5280abb44c82692368f4a67b407db
Banking/d1316bfd1e963f72d01469c27742a5ec1d3ef6b976432f58f48101f9d59ff8
Banking/a6428e11f014bbf50fbe2df9dedb9fd7f4297507b7c8c8c97bfa854f0136bd
Banking/5be086d8133ead002f2c33ac8c0bb4fa1e2e54266cdc9a4941a44c287c4d73
Banking/a6408087c77ee164ace8800fa008a91ac2c41a004c7a0b940e7ccfe4fe5917
Banking/cf9140bca89a53ddf8b4adf3306344ceaaf23f796a8acc00302e4e03588d3f
Banking/dc525cae62a9af45fabdc55e1c145b1523d808e097baa955d03bc4a2881c65
Banking/69b9d8f9bfc42434ef8c4b1628d6643222ec41acc2ac46a01f2dc88f68b749
Banking/be45da68b5509c7057444bceae8d4858e529dd663cda09f893655d8161bfd9
Banking/e5d4ebd917cbb500ae87151b99b0abc08342771480a932150dca123d2fba76
Banking/122c4d48234c82d27bc6c07fff407a5b8c8cea319c88e6bff9f5b1f5821c33
```

用自訂函數取 permissions



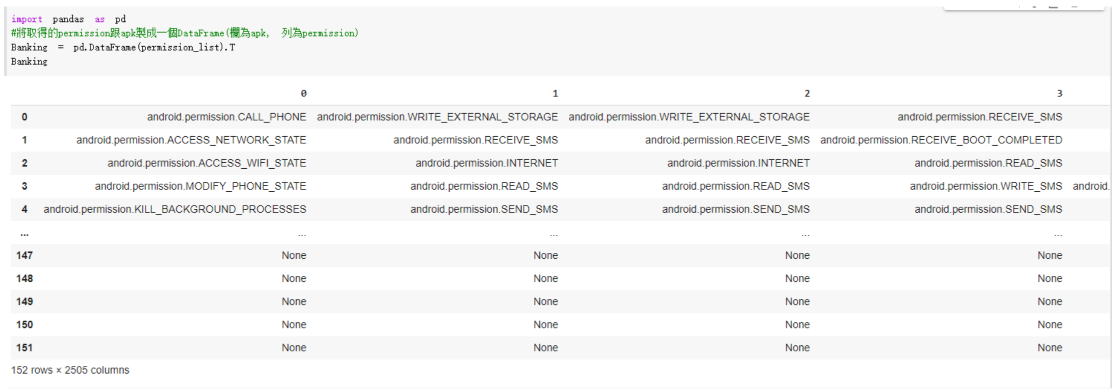
```python
import os
permission_list = []
error_list = []
apkname_list = []
for root, dirs, files in os.walk("Banking", topdown=False):
    for name in files:
        #print(os.path.join(root, name))
        path = os.path.join(root, name)
    #for name in dirs:
    #    print(os.path.join(root, name))
        get_permissions(path, "t")
```

```
['android.permission.CHANGE_NETWORK_STATE', 'android.permission.RECEIVE_SMS', 'android.permission.RECEIVE_BOOT_COMPLETED', 'android.permission.CALL_PHONE', 'android.permission.ACCESS_NETWORK_STATE', 'android.p
['android.permission.RECEIVE_SMS', 'android.permission.SEND_SMS']
['android.permission.CALL_PHONE', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.CAMERA', 'android.permission.READ_SMS', 'android.permission.WRITE_SMS', 'android.permission.KILL_BACKGROUND_PROC
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.RECEIVE_SMS', 'android.permission.RECEIVE_BOOT_COMPLETED', 'android.permission.CALL_PHONE', 'android.permission.ACCESS_NETWORK_STATE', 'androic
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.RECEIVE_SMS', 'android.permission.MOUNT_UNMOUNT_FILESYSTEMS', 'android.permission.INTERNET', 'android.permission.READ_SMS', 'com.android.launch
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.RECEIVE_SMS', 'android.permission.RECEIVE_BOOT_COMPLETED', 'android.permission.VIBRATE', 'android.permission.ACCESS_NETWORK_STATE', 'android.pe
['android.permission.ACCESS_NETWORK_STATE', 'android.permission.ACCESS_WIFI_STATE', 'android.permission.READ_SMS', 'android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.RECEIVE_SMS', 'android.permis
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.RECEIVE_SMS', 'android.permission.RECEIVE_BOOT_COMPLETED', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.ADD_SYSTEM_SERVICE',
['android.permission.READ_LOGS', 'android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.RECEIVE_SMS', 'android.permission.CALL_PHONE', 'android.permission.RECEIVE_BOOT_COMPLETED', 'android.permissio
['android.permission.READ_LOGS', 'android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.RECEIVE_SMS', 'android.permission.CALL_PHONE', 'android.permission.RECEIVE_BOOT_COMPLETED', 'android.permissio
['android.permission.CALL_PHONE', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.MODIFY_PHONE_STATE', 'android.permission.READ_SMS', 'android.permission.ACCESS_COARSE_UPDATES', 'android.permis
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.RECEIVE_SMS', 'android.permission.RECEIVE_BOOT_COMPLETED', 'android.permission.VIBRATE', 'android.permission.ACCESS_NETWORK_STATE', 'android.pe
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.RECEIVE_BOOT_COMPLETED', 'android.permission.VIBRATE', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.ACCESS_WIFI_STATE', 'andr
Name 'android:versionName' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:minSdkVersion' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
['android.permission.MODIFY_AUDIO_SETTINGS', 'android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.RECEIVE_BOOT_COMPLETED', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.ACCESS_WIFI
['android.permission.READ_LOGS', 'android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.VIBRATE', 'rockchip.permission.FULL_SCREEN', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.WAK
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.INTERNET', 'android.permission.INSTALL_PACKAGES', 'android.permission.DELETE_PACKAGES', 'android.per
['android.permission.READ_LOGS', 'android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.RECEIVE_SMS', 'android.permission.CALL_PHONE', 'android.permission.RECEIVE_BOOT_COMPLETED', 'android.permissio
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.VIBRATE', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.GET_TASKS', 'android.permission.WAKE_LOCK', 'android.permission.ACCESS
['android.permission.CALL_PHONE', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.ACCESS_WIFI_STATE', 'android.permission.MOUNT_UNMOUNT_FILESYSTEMS', 'android.permission.READ_SMS', 'android.perm
['android.permission.CALL_PHONE', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.ACCESS_WIFI_STATE', 'android.permission.READ_SMS', 'android.permission.WRITE_SMS', 'android.permission.KILL_BACK
```

```
[ ]  len(permission_list)

     2505

[ ]  error_list

     ['9cc47edb7378b27858632805a6e992454bc0ced64f3c057933d98053ffe17171.sh']

[ ]  len(error_list)

     1
```

- 總共從 2505 個 apk 取 permissions，其中這一個 9cc47edb7378b27858632805a6e992454bc0ced64f3c057933d98053ffe17171.sh 是 shell 檔，所以在取 permission 時有錯誤產生

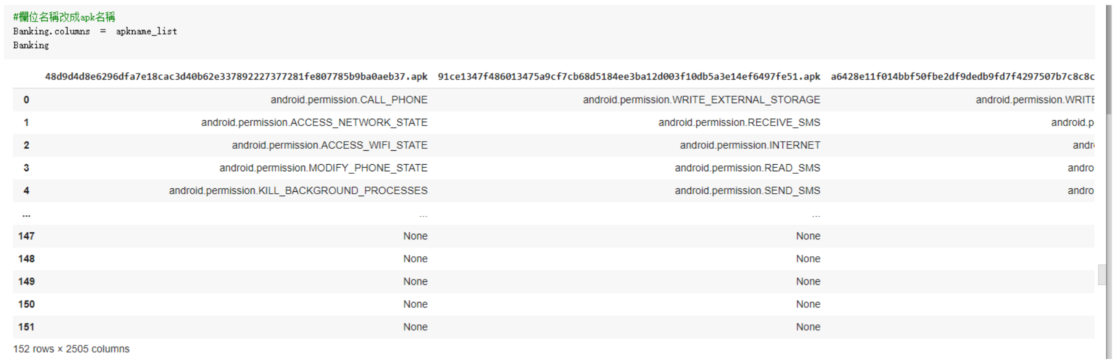
將 permissions 和 apk 製成 DataFrame



```
import  pandas  as  pd
#將取得的permission跟apk製成一個DataFrame(欄為apk,  列為permission)
Banking  =  pd.DataFrame(permission_list).T
Banking
```

	0	1	2	3	
0	android.permission.CALL_PHONE	android.permission.WRITE_EXTERNAL_STORAGE	android.permission.WRITE_EXTERNAL_STORAGE	android.permission.RECEIVE_SMS	
1	android.permission.ACCESS_NETWORK_STATE	android.permission.RECEIVE_SMS	android.permission.RECEIVE_SMS	android.permission.RECEIVE_BOOT_COMPLETED	
2	android.permission.ACCESS_WIFI_STATE	android.permission.INTERNET	android.permission.INTERNET	android.permission.READ_SMS	
3	android.permission.MODIFY_PHONE_STATE	android.permission.READ_SMS	android.permission.READ_SMS	android.permission.WRITE_SMS	android.
4	android.permission.KILL_BACKGROUND_PROCESSES	android.permission.SEND_SMS	android.permission.SEND_SMS	android.permission.SEND_SMS	
...	...	...	...	...	
147	None	None	None	None	
148	None	None	None	None	
149	None	None	None	None	
150	None	None	None	None	
151	None	None	None	None	

152 rows × 2505 columns



```
#欄位名稱改成apk名稱
Banking.columns  =  apkname_list
Banking
```

	48d9d4d8e6296dfa7e18cac3d40b62e337892227377281fe807785b9ba0aeb37.apk	91ce1347f486013475a9cf7cb68d5184ee3ba12d003f10db5a3e14ef6497fe51.apk	a6428e11f014bbf50fbe2df9dedb9fd7f4297507b7c8c8c
0	android.permission.CALL_PHONE	android.permission.WRITE_EXTERNAL_STORAGE	android.permission.WRITE
1	android.permission.ACCESS_NETWORK_STATE	android.permission.RECEIVE_SMS	android.p
2	android.permission.ACCESS_WIFI_STATE	android.permission.INTERNET	andr
3	android.permission.MODIFY_PHONE_STATE	android.permission.READ_SMS	andro
4	android.permission.KILL_BACKGROUND_PROCESSES	android.permission.SEND_SMS	andro
...	...	...	
147	None	None	
148	None	None	
149	None	None	
150	None	None	
151	None	None	

152 rows × 2505 columns

存成 csv 檔

```
Banking.to_csv("Banking_permission.csv")
```

- Benign1

解壓縮

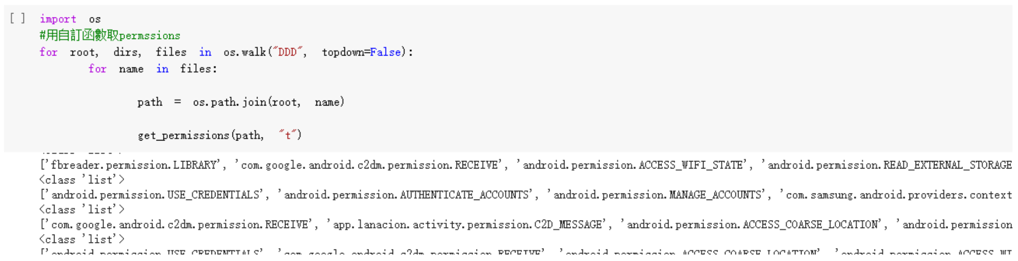
```
!tar  -xvf  './drive/MyDrive/ECT_HW10_dataset/Benign/Benign1.tar'
```

```
DDD.zip
```

```
!unzip  "DDD.zip"
```

```
Archive:  DDD.zip
   creating: DDD/
  extracting: DDD/000aa1f313ddc02b9e365d5dcca92841eac4f40cb88f595fad0a872d147
  extracting: DDD/003a38e8033e9860d10f441490ef1a15c15fc58e5a3d9485e17d4705357
  extracting: DDD/00b2154d1a843117dd3441a5b869f58a9667f31761aec7558e88b934807
  extracting: DDD/00be2e75ca91895675879a869c24547f3f71edc7bdd6cb873a7c8d421a9
  extracting: DDD/00c3aae6b765da3ea08b4cd61edeb76ed9824573b7f65a1aaba6b69db42
  extracting: DDD/00eac9c75cdb38f0158589d233d22cc78f69f0648159c9e8880e52e855b
  extracting: DDD/01ba83392dc6ff8867f9313b98843fabfdc1e1c9ff6a8ba15c824e0e377
  extracting: DDD/01c02229d591af45ceec8928c8d0827b164b061b1b676a88fe2b4cb82b2
  extracting: DDD/01c8eefb6202a7afbe3b965a47d4812a9a2c88edaf39f2a4c9fedc3000e
```

用自訂函數取 permissions



```
[ ]  import  os
     #用自訂函數取permssions
     for  root,  dirs,  files  in  os.walk("DDD",  topdown=False):
             for  name  in  files:

                     path  =  os.path.join(root,  name)

                     get_permissions(path,  "t")
```

```
['fbreader.permission.LIBRARY', 'com.google.android.c2dm.permission.RECEIVE', 'android.permission.ACCESS_WIFI_STATE', 'android.permission.READ_EXTERNAL_STORAGE
<class 'list'>
['android.permission.USE_CREDENTIALS', 'android.permission.AUTHENTICATE_ACCOUNTS', 'android.permission.MANAGE_ACCOUNTS', 'com.samsung.android.providers.context
<class 'list'>
['com.google.android.c2dm.permission.RECEIVE', 'app.lanacion.activity.permission.C2D_MESSAGE', 'android.permission.ACCESS_COARSE_LOCATION', 'android.permission
<class 'list'>
```

將 permissions 和 apk 製成 DataFrame

```
#將permissions和apk製成DataFrame
b1  =  pd.DataFrame(permission_list).T
```

```
apknames  =  []
for  root,  dirs,  files  in  os.walk("DDD",  topdown=False):
        for  name  in  files:
                apknames.append(name)
```

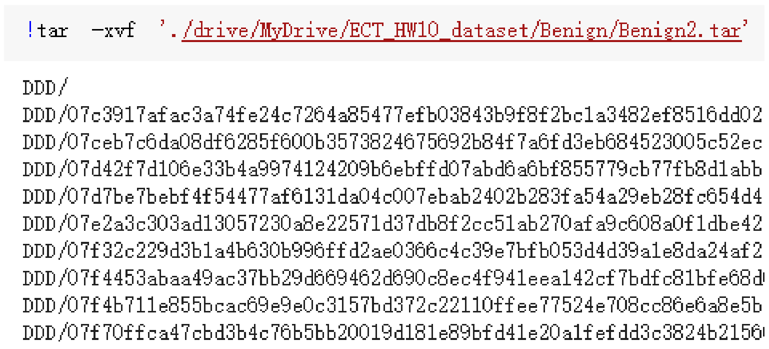
```
#DataFrame欄位名稱改成apk名稱
b1.columns  =  apknames
```

存成 csv 檔

```
b1.to_csv("benign1_permission.csv")
```

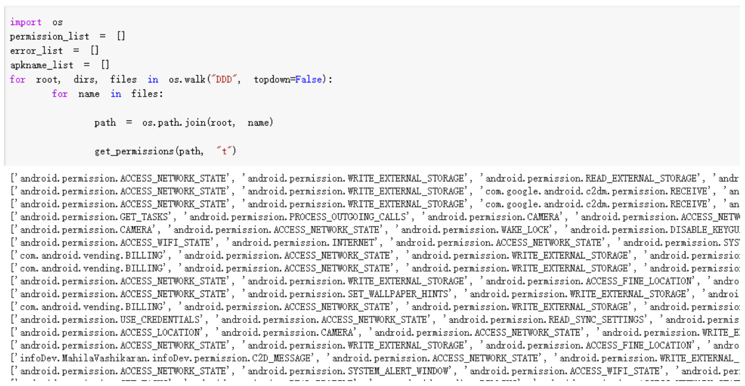
(以下同 Benign1 做法，所以僅截圖呈現)

- Benign2



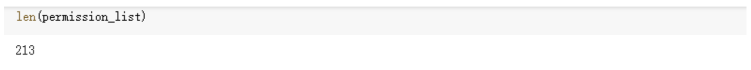
```
!tar  -xvf  './drive/MyDrive/ECT_HW10_dataset/Benign/Benign2.tar'
```

```
DDD/
DDD/07c3917afac3a74fe24c7264a85477efb03843b9f8f2bc1a3482ef8516dd02
DDD/07ceb7c6da08df6285f600b3573824675692b84f7a6fd3eb684523005c52ec
DDD/07d42f7d106e33b4a9974124209b6ebffd07abd6a6bf855779cb77fb8d1abb
DDD/07d7be7bebf4f54477af6131da04c007ebab2402b283fa54a29eb28fc654d4
DDD/07e2a3c303ad13057230a8e22571d37db8f2cc51ab270afa9c608a0f1dbe42
DDD/07f32c229d3b1a4b630b996ffd2ae0366c4c39e7bfb053d4d39a1e8da24af2
DDD/07f4453abaa49ac37bb29d669462d690c8ec4f941eea142cf7bdfc81bfe68d
DDD/07f4b711e855bcac69e9e0c3157bd372c22110ffee77524e708cc86e6a8e5b
DDD/07f70ffca47cbd3b4c76b5bb20019d181e89bfd41e20a1fefdd3c3824b2156
```



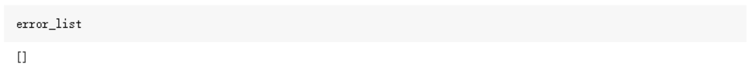
```python
import  os
permission_list  =  []
error_list  =  []
apkname_list  =  []
for  root,  dirs,  files  in  os.walk("DDD",  topdown=False):
      for  name  in  files:

            path  =  os.path.join(root,  name)

            get_permissions(path,  "t")
```

```
['android.permission.ACCESS_NETWORK_STATE', 'android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.READ_EXTERNAL_STORAGE', 'andr
['android.permission.ACCESS_NETWORK_STATE', 'android.permission.WRITE_EXTERNAL_STORAGE', 'com.google.android.c2dm.permission.RECEIVE', 'an
['android.permission.ACCESS_NETWORK_STATE', 'android.permission.WRITE_EXTERNAL_STORAGE', 'com.google.android.c2dm.permission.RECEIVE', 'an
['android.permission.GET_TASKS', 'android.permission.PROCESS_OUTGOING_CALLS', 'android.permission.CAMERA', 'android.permission.ACCESS_NETW
['android.permission.CAMERA', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.WAKE_LOCK', 'android.permission.DISABLE_KEYGU
['android.permission.ACCESS_WIFI_STATE', 'android.permission.INTERNET', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.SYS
['com.android.vending.BILLING', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.WRITE_EXTERNAL_STORAGE', 'android.permissio
['com.android.vending.BILLING', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.WRITE_EXTERNAL_STORAGE', 'android.permissio
['android.permission.ACCESS_NETWORK_STATE', 'android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.ACCESS_FINE_LOCATION', 'andro
['android.permission.ACCESS_NETWORK_STATE', 'android.permission.SET_WALLPAPER_HINTS', 'android.permission.WRITE_EXTERNAL_STORAGE', 'androi
['com.android.vending.BILLING', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.WRITE_EXTERNAL_STORAGE', 'android.permissio
['android.permission.USE_CREDENTIALS', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.READ_SYNC_SETTINGS', 'android.permis
['android.permission.ACCESS_LOCATION', 'android.permission.CAMERA', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.WRITE_E
['android.permission.ACCESS_NETWORK_STATE', 'android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.ACCESS_FINE_LOCATION', 'andro
['infoDev.MahilaVashikaran.infoDev.permission.C2D_MESSAGE', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.WRITE_EXTERNAL_
['android.permission.ACCESS_NETWORK_STATE', 'android.permission.SYSTEM_ALERT_WINDOW', 'android.permission.ACCESS_WIFI_STATE', 'android.per
```



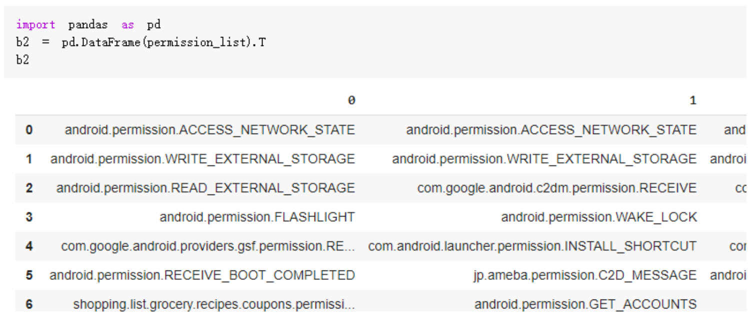
```python
len(permission_list)
```

```
213
```



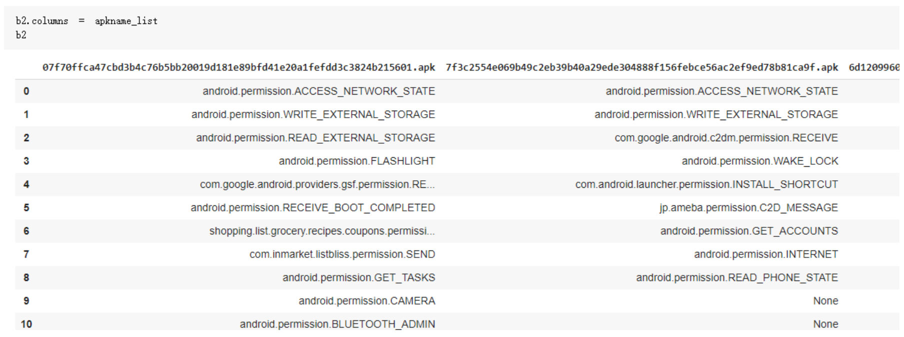
```python
error_list
```

```
[]
```



```python
import  pandas  as  pd
b2  =  pd.DataFrame(permission_list).T
b2
```

	0	1	
0	android.permission.ACCESS_NETWORK_STATE	android.permission.ACCESS_NETWORK_STATE	and
1	android.permission.WRITE_EXTERNAL_STORAGE	android.permission.WRITE_EXTERNAL_STORAGE	androi
2	android.permission.READ_EXTERNAL_STORAGE	com.google.android.c2dm.permission.RECEIVE	cc
3	android.permission.FLASHLIGHT	android.permission.WAKE_LOCK	
4	com.google.android.providers.gsf.permission.RE...	com.android.launcher.permission.INSTALL_SHORTCUT	coı
5	android.permission.RECEIVE_BOOT_COMPLETED	jp.ameba.permission.C2D_MESSAGE	androi
6	shopping.list.grocery.recipes.coupons.permissi...	android.permission.GET_ACCOUNTS	

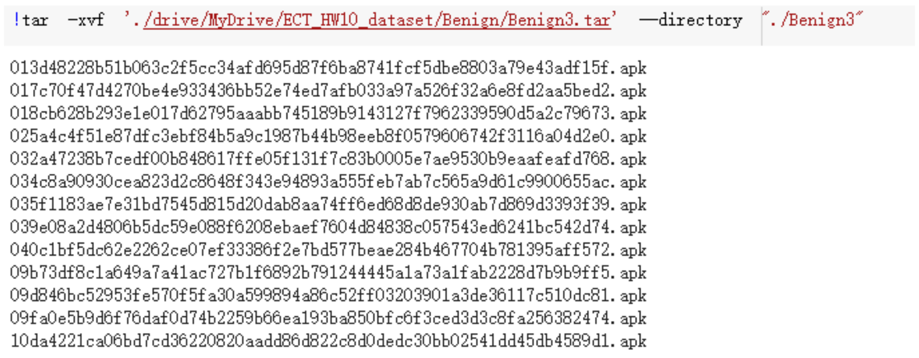


```python
b2.columns  =  apkname_list
b2
```

	07f70ffca47cbd3b4c76b5bb20019d181e89bfd41e20a1fefdd3c3824b215601.apk	7f3c2554e069b49c2eb39b40a29ede304888f156febce56ac2ef9ed78b81ca9f.apk	6d1209960
0	android.permission.ACCESS_NETWORK_STATE	android.permission.ACCESS_NETWORK_STATE	
1	android.permission.WRITE_EXTERNAL_STORAGE	android.permission.WRITE_EXTERNAL_STORAGE	
2	android.permission.READ_EXTERNAL_STORAGE	com.google.android.c2dm.permission.RECEIVE	
3	android.permission.FLASHLIGHT	android.permission.WAKE_LOCK	
4	com.google.android.providers.gsf.permission.RE...	com.android.launcher.permission.INSTALL_SHORTCUT	
5	android.permission.RECEIVE_BOOT_COMPLETED	jp.ameba.permission.C2D_MESSAGE	
6	shopping.list.grocery.recipes.coupons.permissi...	android.permission.GET_ACCOUNTS	
7	com.inmarket.listbliss.permission.SEND	android.permission.INTERNET	
8	android.permission.GET_TASKS	android.permission.READ_PHONE_STATE	
9	android.permission.CAMERA	None	
10	android.permission.BLUETOOTH_ADMIN	None	

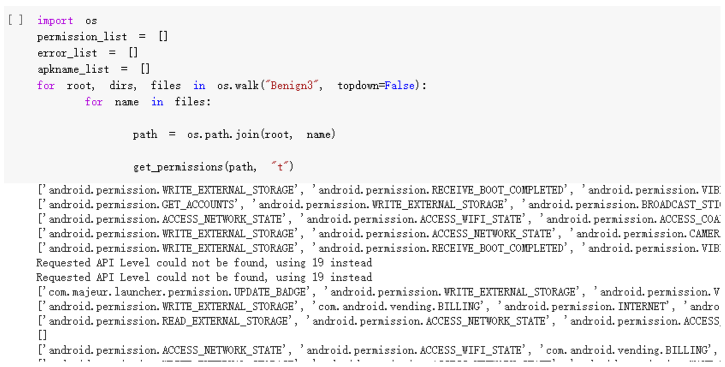
```python
b2.to_csv("benign2_permission.csv")
```

- Benign3



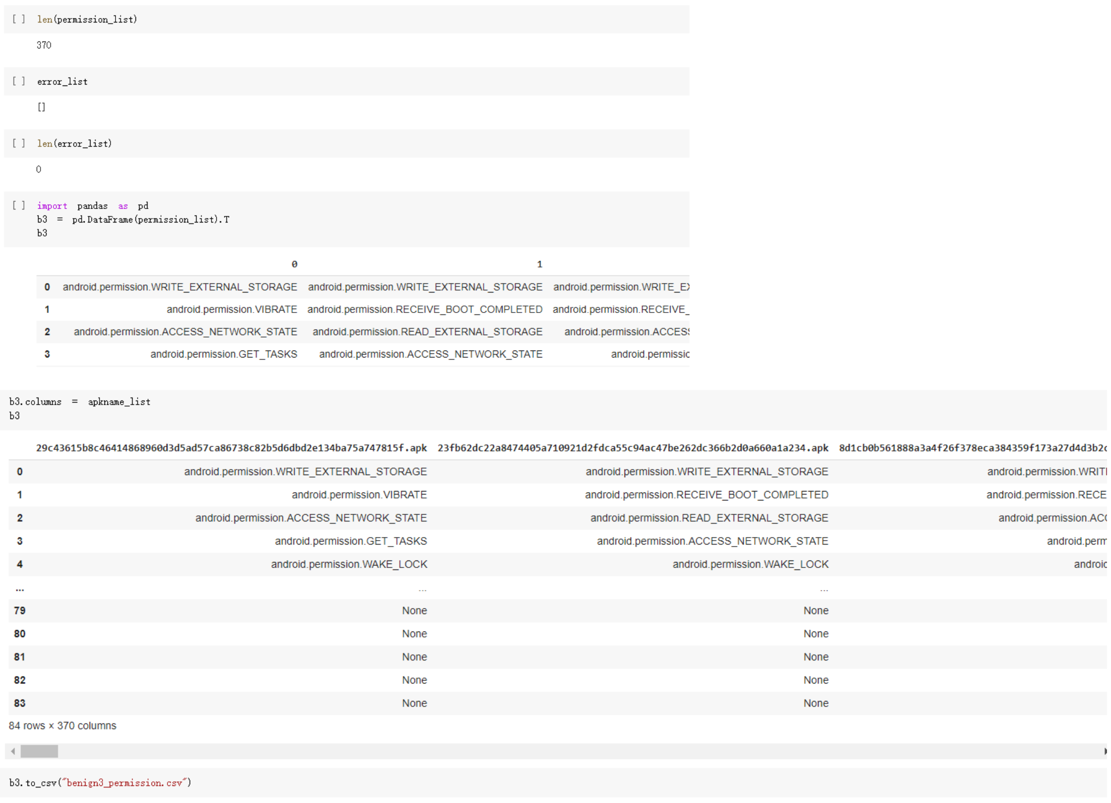
```
!tar -xvf './drive/MyDrive/ECT_HW10_dataset/Benign/Benign3.tar' --directory "./Benign3"
```

```
013d48228b51b063c2f5cc34afd695d87f6ba8741fcf5dbe8803a79e43adf15f.apk
017c70f47d4270be4e933436bb52e74ed7afb033a97a526f32a6e8fd2aa5bed2.apk
018cb628b293e1e017d62795aaabb745189b9143127f7962339590d5a2c79673.apk
025a4c4f51e87dfc3ebf84b5a9c1987b44b98eeb8f0579606742f3116a04d2e0.apk
032a47238b7cedf00b848617ffe05f131f7c83b0005e7ae9530b9eaafeafd768.apk
034c8a90930cea823d2c8648f343e94893a555febf7ab7c565a9d61c9900655ac.apk
035f1183ae7e31bd7545d815d20dab8aa74ff6ed68d8de930ab7d869d3393f39.apk
039e08a2d4806b5dc59e088f6208ebaef7604d84838c057543ed6241bc542d74.apk
040c1bf5dc62e2262ce07ef33386f2e7bd577beae284b467704b781395aff572.apk
09b73df8c1a649a7a41ac727b1f6892b791244445a1a73a1fab2228d7b9b9ff5.apk
09d846bc52953fe570f5fa30a599894a86c52ff03203901a3de36117c510dc81.apk
09fa0e5b9d6f76daf0d74b2259b66ea193ba850bfc6f3ced3d3c8fa256382474.apk
10da4221ca06bd7cd36220820aadd86d822c8d0dedc30bb02541dd45db4589d1.apk
```



```
[ ] import os
    permission_list = []
    error_list = []
    apkname_list = []
    for root, dirs, files in os.walk("Benign3", topdown=False):
          for name in files:

                  path = os.path.join(root, name)

                  get_permissions(path, "t")
```

```
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.RECEIVE_BOOT_COMPLETED', 'android.permission.VIB
['android.permission.GET_ACCOUNTS', 'android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.BROADCAST_STI
['android.permission.ACCESS_NETWORK_STATE', 'android.permission.ACCESS_WIFI_STATE', 'android.permission.ACCESS_COA
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.CAMER
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.RECEIVE_BOOT_COMPLETED', 'android.permission.VIB
Requested API Level could not be found, using 19 instead
Requested API Level could not be found, using 19 instead
['com.majeur.launcher.permission.UPDATE_BADGE', 'android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.V
['android.permission.WRITE_EXTERNAL_STORAGE', 'com.android.vending.BILLING', 'android.permission.INTERNET', 'andro
['android.permission.READ_EXTERNAL_STORAGE', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.ACCESS
[]
['android.permission.ACCESS_NETWORK_STATE', 'android.permission.ACCESS_WIFI_STATE', 'com.android.vending.BILLING',
```



```
[ ] len(permission_list)
```

```
370
```

```
[ ] error_list
```

```
[]
```

```
[ ] len(error_list)
```

```
0
```

```
[ ] import pandas as pd
    b3 = pd.DataFrame(permission_list).T
    b3
```

	0	1	
0	android.permission.WRITE_EXTERNAL_STORAGE	android.permission.WRITE_EXTERNAL_STORAGE	android.permission.WRITE_E)
1	android.permission.VIBRATE	android.permission.RECEIVE_BOOT_COMPLETED	android.permission.RECEIVE_
2	android.permission.ACCESS_NETWORK_STATE	android.permission.READ_EXTERNAL_STORAGE	android.permission.ACCES!
3	android.permission.GET_TASKS	android.permission.ACCESS_NETWORK_STATE	android.permissic

```
b3.columns = apkname_list
b3
```

	29c43615b8c46414868960d3d5ad57ca86738c82b5d6dbd2e134ba75a747815f.apk	23fb62dc22a8474405a710921d2fdca55c94ac47be262dc366b2d0a660a1a234.apk	8d1cb0b561888a3a4f26f378eca384359f173a27d4d3b2c
0	android.permission.WRITE_EXTERNAL_STORAGE	android.permission.WRITE_EXTERNAL_STORAGE	android.permission.WRITI
1	android.permission.VIBRATE	android.permission.RECEIVE_BOOT_COMPLETED	android.permission.RECE
2	android.permission.ACCESS_NETWORK_STATE	android.permission.READ_EXTERNAL_STORAGE	android.permission.ACC
3	android.permission.GET_TASKS	android.permission.ACCESS_NETWORK_STATE	android.perm
4	android.permission.WAKE_LOCK	android.permission.WAKE_LOCK	androic
...	...	...	
79	None	None	
80	None	None	
81	None	None	
82	None	None	
83	None	None	

84 rows × 370 columns

```
b3.to_csv("benign3_permission.csv")
```

- Benign4

```
!tar -xvf './drive/MyDrive/ECT_HW10_dataset/Benign/Benign4.tar' —directory "./Benign4"
```

```
89c64b7025270d14e76e07bf2a4ac61949e54c4915c71aefd2e83c84d4b3f39b.apk
89c665e17c5a22eb9ee50c782671eb0f46215a4e82bbfc5d40a3023c9c1cf541.apk
90c2ac2425ee09df9690ea541ae0da67c1ba27567b1905b275c3983cf1c34bf3.apk
90c6cca507db572a586d1325d7addadb336d0f58499b6ef0614fa5bc0ab1b34a.apk
90ca90fac4b02c6f258c9c5a676d87a77d664020f4d2684065ff0f4df383cdd5.apk
90cf5b827601596971d9012cc76549a57442603c93e7e012de5fe14627c6731a.apk
90d4f2816a7f90bc9b56ef51fe770d024d16fc584117200fd2362172e60e4234.apk
90e2f3d82693318c69cc6a1cd13b2aea6f04ff3fc056fdaf7d316aefbd9b0a49.apk
90ecd877273fd8ac1996491e8b961912da7ebf0c886263c716d86f0834874390.apk
91a89d58869809924706eec98b5c561758f8459d5971baaf640455282cea3455.apk
91b3656101152l626bc111e070e662a9bd5a5162963cd0fa07e0d7428fed8f04.apk
91ca6acedc015b215bffb62811399aec67b4666312fd6c039118fb18076b38a2.apk
91cf5bb3cbf359cf20be97f53c601af05eb38f18a747b81f6ee1c462eac2d70e.apk
91ecf697df41ac2f532a08d8133ec4ae3da9efff556311181188bc06163a967b.apk
```

```python
import os
permission_list = []
error_list = []
apkname_list = []
for root, dirs, files in os.walk("Benign4", topdown=False):
      for name in files:

            path = os.path.join(root, name)

            get_permissions(path, "t")
```

```
['android.permission.ACCESS_NETWORK_STATE', 'com.android.vending.BILLING', 'android.permission.ACCESS_WIFI_STATE', 'android.permission.ACCESS_COARSE_LOC
['android.permission.READ_PHONE_STATE', 'android.permission.INTERNET', 'android.permission.ACCESS_NETWORK_STATE']
Requested API Level could not be found, using 19 instead
['android.permission.FLASHLIGHT', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.ACCESS_WIFI_STATE', 'android.permission.CAMERA', 'andrc
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.INTERNET', 'android.permission.ACCESS_NETWORK_STATE']
['android.permission.CHANGE_NETWORK_STATE', 'com.lenovo.launcher.permission.READ_SETTINGS', 'android.permission.READ_PROFILE', 'com.fede.launcher.permis
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.READ_EXTERNAL_STORAGE', 'android.permission.ACCESS_NETWORK_STATE', 'android.permissior
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.REAL_GET_TASKS', 'android.permission.PACKAGE_USAGE_STATS', 'android.permission.RECEIVE
['android.permission.FLASHLIGHT', 'android.permission.CALL_PHONE', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.CAMERA', 'android.perm
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.VIBRATE', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.WAKE_LOCK', '
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.VIBRATE', 'android.permission.ACCESS_NETWORK_STATE', 'com.android.vending.BILLING', 'a
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.RECEIVE_BOOT_COMPLETED', 'android.permission.ACCESS_NETWORK_STATE', 'android.permissic
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.RECEIVE_BOOT_COMPLETED', 'android.permission.ACCESS_NETWORK_STATE', 'android.permissic
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.RECEIVE_SMS', 'android.permission.READ_EXTERNAL_STORAGE', 'android.permission.ACCESS_N
['android.permission.LOCATION', 'android.permission.VIBRATE', 'android.permission.CALL_PHONE', 'android.permission.ACCESS_NETWORK_STATE', 'android.permi
```

```python
len(permission_list)
```

```
430
```

```python
error_list
```

```
[]
```

```python
import pandas as pd
b4 = pd.DataFrame(permission_list).T
b4
```

	0	1	2	3	
0	android.permission.VIBRATE	android.permission.CHANGE_NETWORK_STATE	android.permission.WRITE_EXTERNAL_STORAGE	android.permission.WRITE_EXTERNAL_STORAGE	android.p
1	android.permission.ACCESS_NETWORK_STATE	android.permission.ACCESS_NETWORK_STATE	android.permission.GET_ACCOUNTS	android.permission.READ_EXTERNAL_STORAGE	androi
2	com.android.vending.BILLING	android.permission.ACCESS_WIFI_STATE	com.narcade.farm_bubbles_bubble_shooter.permis...	android.permission.ACCESS_NETWORK_STATE	
3	android.permission.INTERNET	android.permission.CAMERA	android.permission.VIBRATE	com.android.vending.BILLING	
4	com.samsung.android.providers.context.permissi...	android.permission.MOUNT_UNMOUNT_FILESYSTEMS	android.permission.ACCESS_NETWORK_STATE	android.permission.WAKE_LOCK	android.p
...	...	...	...	...	
76	None	None	None	None	
77	None	None	None	None	

```python
b4.columns = apkname_list
b4
```

	41dff80790a4ebb39f139f9bf9a0509775ea0ab9ee6a192b2a58213ac1b9c0fd.apk	46bde1e79c323a15474e60ae7aa7444e51746ba57d9499b7aa51cdc4ac7c7cc0.apk	47d92d4b9b0ce4333b
0	android.permission.VIBRATE	android.permission.CHANGE_NETWORK_STATE	
1	android.permission.ACCESS_NETWORK_STATE	android.permission.ACCESS_NETWORK_STATE	
2	com.android.vending.BILLING	android.permission.ACCESS_WIFI_STATE	
3	android.permission.INTERNET	android.permission.CAMERA	
4	com.samsung.android.providers.context.permissi...	android.permission.MOUNT_UNMOUNT_FILESYSTEMS	
...	...	...	
76	None	None	
77	None	None	
78	None	None	
79	None	None	

- Benign5

```
!tar  -xvf  './drive/MyDrive/ECT_HW10_dataset/Benign/Benign5.tar'  --directory  "./Benign5"
```

```
0234adcfa485181cb848efdb3de4590c331220c256503888e42413ac2f98aa6b.apk
0305f3b1b1f39fe397309e86f3ec5682335484bf31ad720f8037baf9c7567e05.apk
0349ca08aebf9c352759981b07aff8e4866a8ee0b8931d5394ba38caefac55ff.apk
0380c2fc780f4a8b0a9febf23b86c0e1c5a0bbc3e4bcfe0717e6a15f726b9d23.apk
0484cbb0907264a701422f902de923bf55bcbb8b4d57243ab5f5d56ebdef53d7.apk
0518a70366db5786a5207037c416661f36f9058cff83ab88862b81ab291b3d95.apk
0551a5f592ec78d318389fc8177f93130a6ef1b5fd4d981254a5bf0dcdcee3bb.apk
0563ff99f825cd244c73555b96b6447cdaa2416f4e1de14e83c2aeb79821ce80.apk
0577b202493b9e692dfb586a5adff8dbc32c9ae08bca08daaef733974d97e306.apk
0630e8f3dc2c546cda82caac7774bb021ad91951b5af239aeda8b231a1b42a41.apk
0661abb74f5c1edcb3931699aa0a9079556b850599bed70ad3ea1d8b998dd2f7.apk
0725f00dd2fee594f005cc2141c16794a41f489050b84284f3626d803390c00d.apk
0750f5c2a6436060a33631bbb3ccb0aa900fddd7027617b38338806f2a0c8449.apk
0772adc7636169870743cbd8abbb5ba1d6a3fb29919513c77e7cc7e577f693a6.apk
```

```python
import  os
permission_list  =  []
error_list  =  []
apkname_list  =  []
for  root,  dirs,  files  in  os.walk("Benign5",  topdown=False):
        for  name  in  files:

                path  =  os.path.join(root,  name)

                get_permissions(path,  "t")
```

```
['android.permission.ACCESS_NETWORK_STATE', 'android.permission.CAMERA', 'android.permission.ACCESS_WIFI_STATE', 'com.mobilemotion.dubsmash..permissi
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.INTERACT_ACROSS_USERS_FULL', 'android.permission.GET_ACCOUNTS', 'android.permission.
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.GET_ACCOUNTS', 'android.permission.VIBRATE', 'com.bizzabo.client.permission.MAPS_RE
['android.permission.READ_LOGS', 'android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.WAKE_LOC
['android.permission.ACCESS_MOCK_LOCATION', 'android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.BROADCAST_STICKY', 'android.permission.V
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.GET_ACCOUNTS', 'cdff.mobileapp.permission.C2D_MESSAGE', 'android.permission.VIBRATE
['android.permission.INTERNET', 'android.permission.ACCESS_NETWORK_STATE']
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.ACCESS_WIFI_STATE', 'android.permission.
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.GET_ACCOUNTS', 'android.permission.RECORD_AUDIO', 'android.permission.CALL_PHONE',
['android.permission.GET_ACCOUNTS', 'android.permission.VIBRATE', 'android.permission.ACCESS_NETWORK_STATE', 'hu.teambusiness.tipster365.permission.C
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.INTERNET', 'android.permission.WRITE_SET
['android.permission.WRITE_EXTERNAL_STORAGE', 'com.athan.android.permission.C2D_MESSAGE', 'android.permission.RECEIVE_BOOT_COMPLETED', 'android.permi
['android.permission.CHANGE_NETWORK_STATE', 'android.permission.CALL_PHONE', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.ACCESS_WI
```

```python
len(permission_list)
```

```
321
```

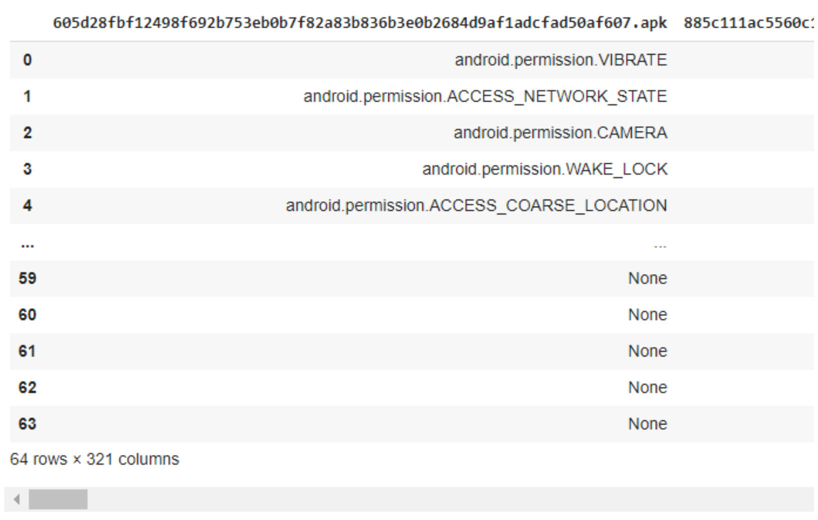
```python
error_list
```

```
[]
```

```python
import  pandas  as  pd
b5  =  pd.DataFrame(permission_list).T
b5
```

	0	1	
0	android.permission.VIBRATE	android.permission.WRITE_EXTERNAL_STORAGE	android.permission.WRITE_E
1	android.permission.ACCESS_NETWORK_STATE	android.permission.ACCESS_NETWORK_STATE	android.permis
2	android.permission.CAMERA	android.permission.ACCESS_WIFI_STATE	pimpmyphoto.lviv.com.pimpm
3	android.permission.WAKE_LOCK	android.permission.WAKE_LOCK	android.permission.READ_E
4	android.permission.ACCESS_COARSE_LOCATION	android.permission.INTERNET	android.permission.ACCES
...	...	...	
59	None	None	
60	None	None	
61	None	None	

```
b5.columns  =  apkname_list
b5
```



```
     605d28fbf12498f692b753eb0b7f82a83b836b3e0b2684d9af1adcfad50af607.apk  885c111ac5560c
0                                   android.permission.VIBRATE
1                      android.permission.ACCESS_NETWORK_STATE
2                                    android.permission.CAMERA
3                                 android.permission.WAKE_LOCK
4                    android.permission.ACCESS_COARSE_LOCATION
...                                                        ...
59                                                        None
60                                                        None
61                                                        None
62                                                        None
63                                                        None
64 rows × 321 columns
```

```
b5.to_csv("benign5_permission.csv")
```

- Benign6

```
!tar  -xvf  './drive/MyDrive/ECT_HW10_dataset/Benign/Benign6.tar'  --directory  "./Benign6"
```

```
00070891aea5767d858be0ff82253e6637e274cd5e7315dd9d8639e8465ed717.apk
001248a66e178bebdfbdca691b841832d4b2c04ef295a17fbe9b35f726d10a21.apk
00216736d5b7a5f18e87769195d69e95da8da267f06114755df2dcce9ed03981.apk
0034344c31f7b8812bc980d67ebd8dfec951d98d5786eb201f3ccfeca427cc54.apk
011022cf8052f1170b0c55a8e9bb45826e6ee1a39f88ff3284e6d10b8cbcb6a7.apk
01115bb5d043b775b1a9b90e4001035b6612c1030002c160219654738634a4b9.apk
02829b99485744dc489b07b8c4edf3470f14d429fb38599fd1a6cb1f3695a048.apk
03505a52fbb783a8f9d53108282865ec7151e26e182ce1bf964d0ff4a2b58a9b.apk
03929b2c7bc1cac90cc559ec4c689e968aa041d4e87387690400a441304537bd.apk
04461b4d48720a9f63164c9df945b3cbaae77e7bbb6c3c5f69fd54d6e9c866b8.apk
04672047ef3694b2ca0d3c483e8f402fd0de2f73148b4423619e778ca19e2c79.apk
0560204a3cd9845adabcdec10dfbd870d4206fc8b972164a57b0b07ea3f2bcda.apk
05704fc60eb8c341d5e6749f4e17b70ab39006e3e86033a58ba3d1d900b36f76.apk
058120bc9f391e930bbe7dfa188cc7e2b9bf3ca349a6583ad9438cd82faf7ff6.apk
05954c3b5b148aaf414a57e063417ed591ad08e593ed64cee2e58423982a38f5.apk
06028bf5045a0c795723b296147196d238408dc0e34de2b089c3d61991caaabf.apk
```

```python
import  os
permission_list  =  []
error_list  =  []
apkname_list  =  []
for  root,  dirs,  files  in  os.walk("Benign6",  topdown=False):
        for  name  in  files:
                #print(os.path.join(root,  name))
                path  =  os.path.join(root,  name)
        #for  name  in  dirs:
        #        print(os.path.join(root,  name))
                get_permissions(path,  "t")
```

```
Name 'android:noHistory' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefi
Name 'android:label' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:name' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:noHistory' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefi
Name 'android:label' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:name' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:configChanges' starts with 'android:' prefix! The Manifest seems to be broken? Removing p
Name 'android:label' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:name' starts with 'android:' prefix! The Manifest seems to be broken? Removing prefix.
Name 'android:configChanges' starts with 'android:' prefix! The Manifest seems to be broken? Removing p
```

```python
len(permission_list)
```

```
479
```

```python
error_list
```

```
[]
```

```python
import  pandas  as  pd
b6  =  pd.DataFrame(permission_list).T
b6
```

	0	1	
0	android.permission.ACCESS_NETWORK_STATE	android.permission.WRITE_EXTERNAL_STORAGE	an
1	android.permission.WAKE_LOCK	android.permission.INTERNET	
2	android.permission.ACCESS_WIFI_STATE	android.permission.ACCESS_NETWORK_STATE	
3	android.permission.INTERNET	None	soundc
4	com.google.android.c2dm.permission.RECEIVE	None	c
...	...	...	
60	None	None	
61	None	None	
62	None	None	
63	None	None	
64	None	None	

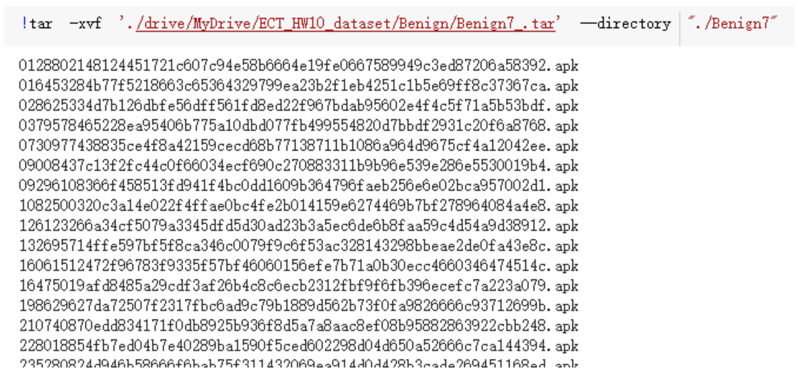
```python
b6.columns  =  apkname_list
b6
```

	5251ef0aa45ea6dbcdb03c7cb8bd19ba5edc135efc011ed1371049aa4cc89e94.apk	537600f6f61bb44f22af938c7
0	android.permission.ACCESS_NETWORK_STATE	
1	android.permission.WAKE_LOCK	
2	android.permission.ACCESS_WIFI_STATE	
3	android.permission.INTERNET	
4	com.google.android.c2dm.permission.RECEIVE	
...	...	
60	None	
61	None	
62	None	
63	None	
64	None	

65 rows × 479 columns

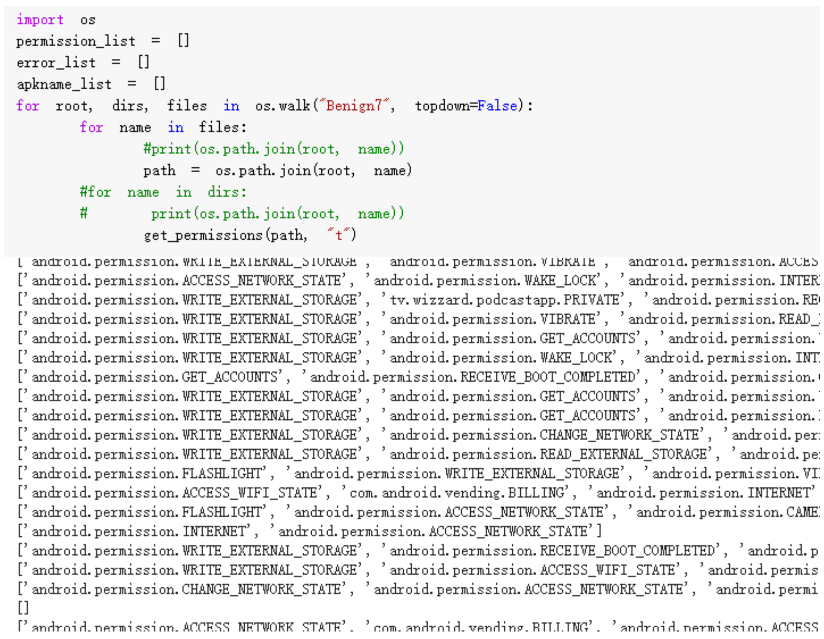
```python
b6.to_csv("benign6_permission.csv")
```

- Benign7



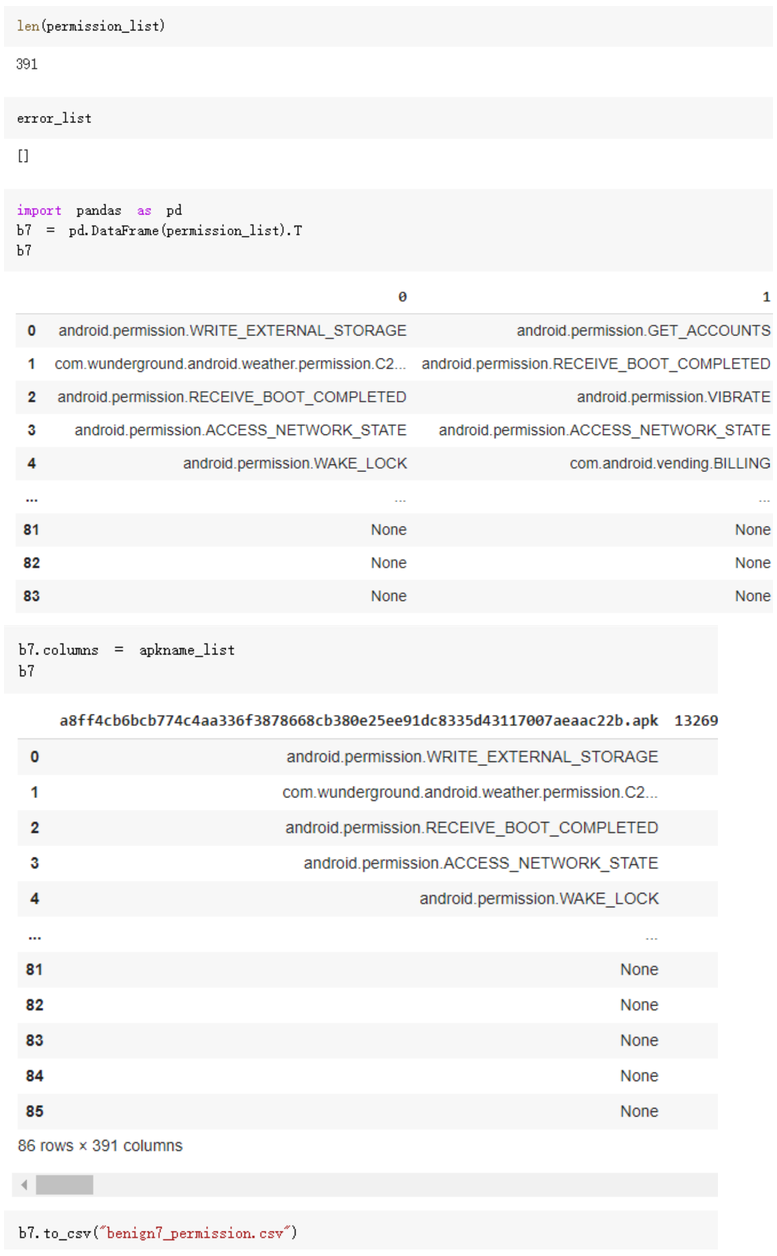
```
!tar -xvf './drive/MyDrive/ECT_HW10_dataset/Benign/Benign7_.tar' --directory "./Benign7"
```

```
0128802148124451721c607c94e58b6664e19fe0667589949c3ed87206a58392.apk
016453284b77f5218663c65364329799ea23b2f1eb4251c1b5e69ff8c37367ca.apk
028625334d7b126dbfe56dff561fd8ed22f967bdab95602e4f4c5f71a5b53bdf.apk
0379578465228ea95406b775a10dbd077fb499554820d7bbdf2931c20f6a8768.apk
0730977438835ce4f8a42159cecd68b77138711b1086a964d9675cf4a12042ee.apk
09008437c13f2fc44c0f66034ecf690c270883311b9b96e539e286e5530019b4.apk
09296108366f458513fd941f4bc0dd1609b364796faeb256e6e02bca957002d1.apk
1082500320c3a14e022f4ffae0bc4fe2b014159e6274469b7bf27896408 4a4e8.apk
126123266a34cf5079a3345dfd5d30ad23b3a5ec6de6b8faa59c4d54a9d38912.apk
132695714ffe597bf5f8ca346c0079f9c6f53ac328143298bbeae2de0fa43e8c.apk
16061512472f96783f9335f57bf46060156efe7b71a0b30ecc466034647 4514c.apk
16475019afd8485a29cdf3af26b4c8c6ecb2312fbf9f6fb396ecefc7a223a079.apk
198629627da72507f2317fbc6ad9c79b1889d562b73f0fa9826666c93712699b.apk
210740870edd834171f0db8925b936f8d5a7a8aac8ef08b95882863922cbb248.apk
228018854fb7ed04b7e40289ba1590f5ced602298d04d650a52666c7ca144394.apk
235280824d946b58666f6bab75f311432069ea914d0d428b3cade269451168ed.apk
```



```
import os
permission_list = []
error_list = []
apkname_list = []
for root, dirs, files in os.walk("Benign7", topdown=False):
        for name in files:
                #print(os.path.join(root, name))
                path = os.path.join(root, name)
        #for name in dirs:
        #        print(os.path.join(root, name))
                get_permissions(path, "t")
```

```
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.VIBRATE', 'android.permission.ACCES
['android.permission.ACCESS_NETWORK_STATE', 'android.permission.WAKE_LOCK', 'android.permission.INTER
['android.permission.WRITE_EXTERNAL_STORAGE', 'tv.wizzard.podcastapp.PRIVATE', 'android.permission.RE
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.VIBRATE', 'android.permission.READ_
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.GET_ACCOUNTS', 'android.permission.
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.WAKE_LOCK', 'android.permission.INT
['android.permission.GET_ACCOUNTS', 'android.permission.RECEIVE_BOOT_COMPLETED', 'android.permission.
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.GET_ACCOUNTS', 'android.permission.
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.GET_ACCOUNTS', 'android.permission.
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.CHANGE_NETWORK_STATE', 'android.per
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.READ_EXTERNAL_STORAGE', 'android.pe
['android.permission.FLASHLIGHT', 'android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.VI
['android.permission.ACCESS_WIFI_STATE', 'com.android.vending.BILLING', 'android.permission.INTERNET'
['android.permission.FLASHLIGHT', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.CAME
['android.permission.INTERNET', 'android.permission.ACCESS_NETWORK_STATE']
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.RECEIVE_BOOT_COMPLETED', 'android.p
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.ACCESS_WIFI_STATE', 'android.permis
['android.permission.CHANGE_NETWORK_STATE', 'android.permission.ACCESS_NETWORK_STATE', 'android.permi
[]
['android.permission.ACCESS_NETWORK_STATE', 'com.android.vending.BILLING', 'android.permission.ACCESS
```



```
len(permission_list)
```

```
391
```

```
error_list
```

```
[]
```

```
import pandas as pd
b7 = pd.DataFrame(permission_list).T
b7
```

	0	1
0	android.permission.WRITE_EXTERNAL_STORAGE	android.permission.GET_ACCOUNTS
1	com.wunderground.android.weather.permission.C2...	android.permission.RECEIVE_BOOT_COMPLETED
2	android.permission.RECEIVE_BOOT_COMPLETED	android.permission.VIBRATE
3	android.permission.ACCESS_NETWORK_STATE	android.permission.ACCESS_NETWORK_STATE
4	android.permission.WAKE_LOCK	com.android.vending.BILLING
...	...	...
81	None	None
82	None	None
83	None	None

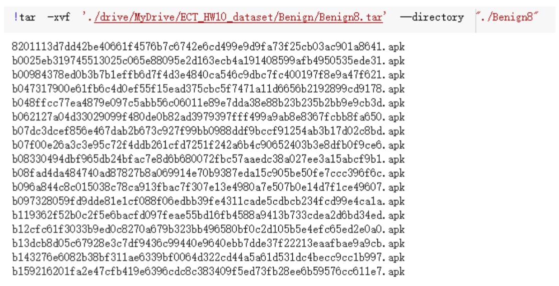
```
b7.columns = apkname_list
b7
```

	a8ff4cb6bcb774c4aa336f3878668cb380e25ee91dc8335d43117007aeaac22b.apk	13269
0	android.permission.WRITE_EXTERNAL_STORAGE	
1	com.wunderground.android.weather.permission.C2...	
2	android.permission.RECEIVE_BOOT_COMPLETED	
3	android.permission.ACCESS_NETWORK_STATE	
4	android.permission.WAKE_LOCK	
...	...	
81	None	
82	None	
83	None	
84	None	
85	None	

86 rows × 391 columns

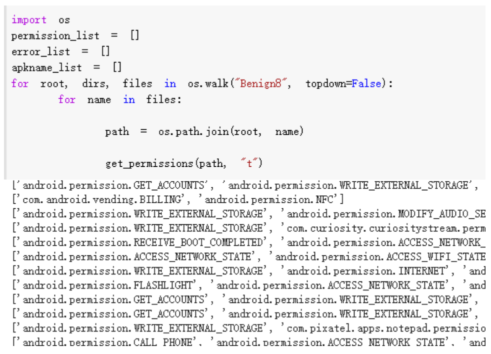
```
b7.to_csv("benign7_permission.csv")
```

- Benign8

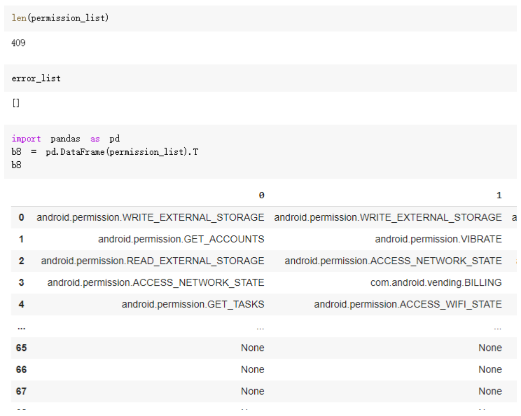


```
!tar -xvf './drive/MyDrive/ECT_HW10_dataset/Benign/Benign8.tar' --directory "./Benign8"
```

```
8201113d7dd42be40661f4576b7c6742e6cd499e9d9fa73f25cb03ac901a8641.apk
b0025eb319745513025c065e88095e2d163ecb4a191408599afb4950535ede31.apk
b00984378ed0b3b7b1effb6d7f4d3e4840ca546c9dbc7fc400197f8e9a47f621.apk
b047317900e61fb6c4d0ef55f15ead375cbc5f7471a11d6656b2192899cd9178.apk
b048ffcc77ea4879e097c5abb56c06011e89e7dda38e88b23b235b2bb9e9cb3d.apk
b062127a04d33029099f480de0b82ad3979397fff499a9ab8e8367fcbb8fa650.apk
b07dc3dcef856e467dab2b673c927f99bb0988ddf9bccf91254ab3b17d02c8bd.apk
b07f00e26a3c3e95c72f4ddb261cfd7251f242a6b4c90652403b3e8dfb0f9ce6.apk
b08330494dbf965db24bfac7e8d6b680072fbc57aaedc38a027ee3a15abcf9b1.apk
b08fad4da484740ad87827b8a069914e70b9387eda15c905be50fe7ccc396f6c.apk
b096a844c8c015038c78ca913fbac7f307e13e4980a7e507b0e14d7f1ce49607.apk
b097328059fd9dde81e1cf088f06edbb39fe4311cade5cdbcb234fcd99e4ca1a.apk
b119362f52b0c2f5e6bacfd097feae55bd16fb4588a9413b733cdea2d6bd34ed.apk
b12cfc61f3033b9ed0c8270a679b323bb496580bf0c2d105b5e4efc65ed2e0a0.apk
b13dcb8d05c67928e3c7df9436c99440e9640ebb7dde37f22213eaafbae9a9cb.apk
b143276e6082b38bf311ae6339bf0064d322cd44a5a61d531dc4becc9cc1b997.apk
b159216201fa2e47cfb419e6396cdc8c383409f5ed73fb28ee6b59576cc611e7.apk
```



```
import os
permission_list = []
error_list = []
apkname_list = []
for root, dirs, files in os.walk("Benign8", topdown=False):
        for name in files:

                path = os.path.join(root, name)

                get_permissions(path, "t")
['android.permission.GET_ACCOUNTS', 'android.permission.WRITE_EXTERNAL_STORAGE',
['com.android.vending.BILLING', 'android.permission.NFC']
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.MODIFY_AUDIO_SE
['android.permission.WRITE_EXTERNAL_STORAGE', 'com.curiosity.curiositystream.perm
['android.permission.RECEIVE_BOOT_COMPLETED', 'android.permission.ACCESS_NETWORK_
['android.permission.ACCESS_NETWORK_STATE', 'android.permission.ACCESS_WIFI_STATE
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.INTERNET', 'and
['android.permission.FLASHLIGHT', 'android.permission.ACCESS_NETWORK_STATE', 'and
['android.permission.GET_ACCOUNTS', 'android.permission.WRITE_EXTERNAL_STORAGE',
['android.permission.GET_ACCOUNTS', 'android.permission.WRITE_EXTERNAL_STORAGE',
['android.permission.WRITE_EXTERNAL_STORAGE', 'com.pixatel.apps.notepad.permissio
['android.permission.CALL_PHONE', 'android.permission.ACCESS_NETWORK_STATE', 'and
```



```
len(permission_list)
```

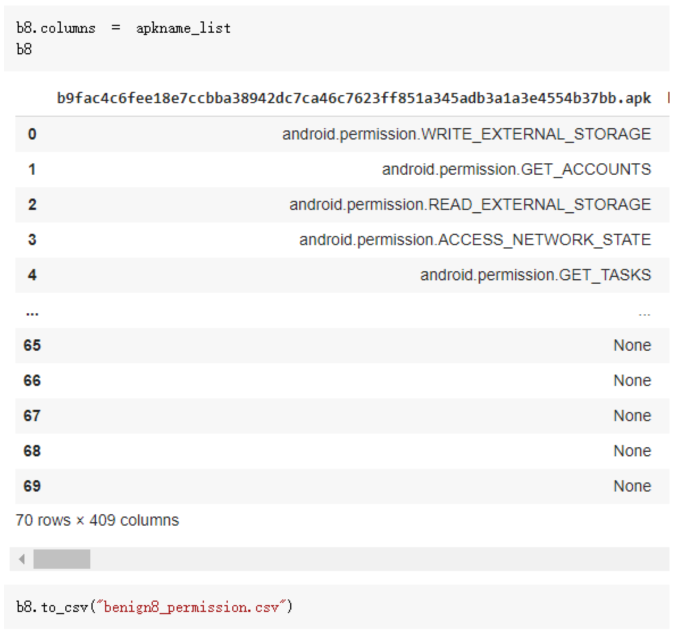
```
409
```

```
error_list
```

```
[]
```

```
import pandas as pd
b8 = pd.DataFrame(permission_list).T
b8
```

	0	1
0	android.permission.WRITE_EXTERNAL_STORAGE	android.permission.WRITE_EXTERNAL_STORAGE
1	android.permission.GET_ACCOUNTS	android.permission.VIBRATE
2	android.permission.READ_EXTERNAL_STORAGE	android.permission.ACCESS_NETWORK_STATE
3	android.permission.ACCESS_NETWORK_STATE	com.android.vending.BILLING
4	android.permission.GET_TASKS	android.permission.ACCESS_WIFI_STATE
...	...	...
65	None	None
66	None	None
67	None	None



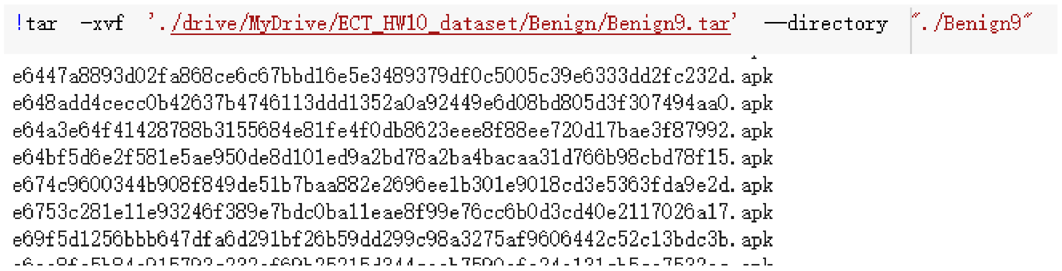
```
b8.columns = apkname_list
b8
```

	b9fac4c6fee18e7ccbba38942dc7ca46c7623ff851a345adb3a1a3e4554b37bb.apk
0	android.permission.WRITE_EXTERNAL_STORAGE
1	android.permission.GET_ACCOUNTS
2	android.permission.READ_EXTERNAL_STORAGE
3	android.permission.ACCESS_NETWORK_STATE
4	android.permission.GET_TASKS
...	...
65	None
66	None
67	None
68	None
69	None

70 rows × 409 columns

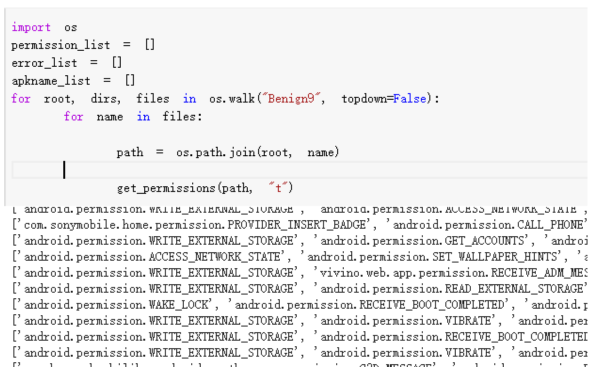
```
b8.to_csv("benign8_permission.csv")
```

- Benign9



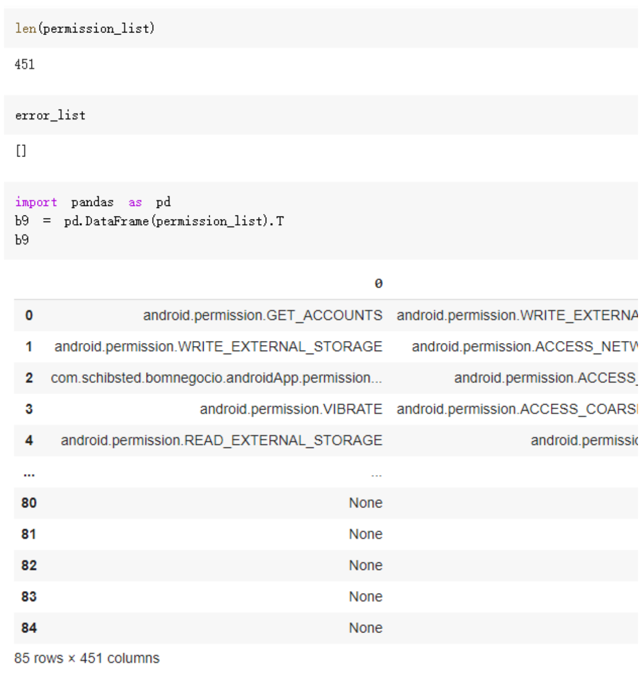
```
!tar -xvf './drive/MyDrive/ECT_HW10_dataset/Benign/Benign9.tar' —directory "./Benign9"
```

```
e6447a8893d02fa868ce6c67bbd16e5e3489379df0c5005c39e6333dd2fc232d.apk
e648add4cecc0b42637b4746113ddd1352a0a92449e6d08bd805d3f307494aa0.apk
e64a3e64f41428788b3155684e81fe4f0db8623eee8f88ee720d17bae3f87992.apk
e64bf5d6e2f581e5ae950de8d101ed9a2bd78a2ba4bacaa31d766b98cbd78f15.apk
e674c9600344b908f849de51b7baa882e2696ee1b301e9018cd3e5363fda9e2d.apk
e6753c281e11e93246f389e7bdc0ba11eae8f99e76cc6b0d3cd40e2117026a17.apk
e69f5d1256bbb647dfa6d291bf26b59dd299c98a3275af9606442c52c13bdc3b.apk
```



```
import os
permission_list = []
error_list = []
apkname_list = []
for root, dirs, files in os.walk("Benign9", topdown=False):
        for name in files:

                path = os.path.join(root, name)

                get_permissions(path, "t")
```

```
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.ACCESS_NETWORK_STATE',
['com.sonymobile.home.permission.PROVIDER_INSERT_BADGE', 'android.permission.CALL_PHONE'
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.GET_ACCOUNTS', 'androi
['android.permission.ACCESS_NETWORK_STATE', 'android.permission.SET_WALLPAPER_HINTS', 'a
['android.permission.WRITE_EXTERNAL_STORAGE', 'vivino.web.app.permission.RECEIVE_ADM_MES
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.READ_EXTERNAL_STORAGE'
['android.permission.WAKE_LOCK', 'android.permission.RECEIVE_BOOT_COMPLETED', 'android.p
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.VIBRATE', 'android.per
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.RECEIVE_BOOT_COMPLETEI
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.VIBRATE', 'android.per
```



```
len(permission_list)
```

```
451
```

```
error_list
```

```
[]
```

```
import pandas as pd
b9 = pd.DataFrame(permission_list).T
b9
```

	0	
0	android.permission.GET_ACCOUNTS	android.permission.WRITE_EXTERNA
1	android.permission.WRITE_EXTERNAL_STORAGE	android.permission.ACCESS_NETW
2	com.schibsted.bomnegocio.androidApp.permission...	android.permission.ACCESS_
3	android.permission.VIBRATE	android.permission.ACCESS_COARS
4	android.permission.READ_EXTERNAL_STORAGE	android.permissio
...	...	
80	None	
81	None	
82	None	
83	None	
84	None	

85 rows × 451 columns

```
b9.columns = apkname_list
b9
```

	e267ad48bba6bc71b2c404ddfa089f4d1eb7d6b65ba71a568a417d37f70d13ce.apk
0	android.permission.GET_ACCOUNTS
1	android.permission.WRITE_EXTERNAL_STORAGE
2	com.schibsted.bomnegocio.androidApp.permission...
3	android.permission.VIBRATE
4	android.permission.READ_EXTERNAL_STORAGE
...	...
80	None
81	None
82	None
83	None
84	None

85 rows × 451 columns

```
b9.to_csv("benign9_permission.csv")
```

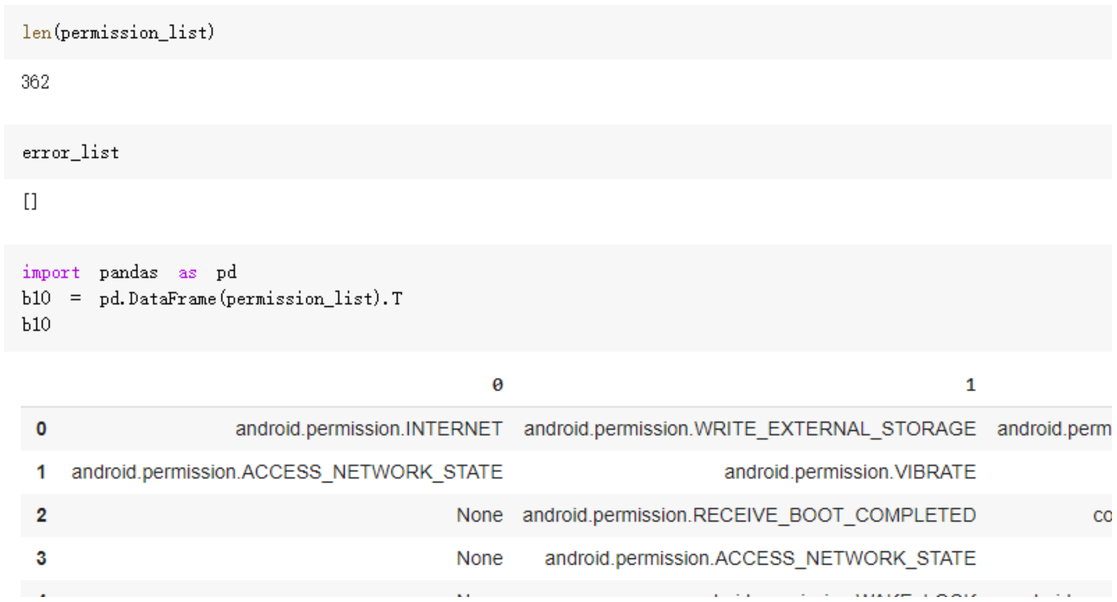
- Benign10

```
!tar -xvf './drive/MyDrive/ECT_HW10_dataset/Benign/Benign10.tar' —directory "./Benign10"
```

```
ea84d2edca4fabd32b822ee09c33b26a284508d316db54951d6e42fd08cfcdcf.apk
ea8c71fe01969abc650ffebc61e0ab6c42a2ea0e67e4afb2d27ec311b535a0cd.apk
ea9b9f84b9dcb6da922ea8488a54fc94c0145f1593c994e008eff3dda96a8308.apk
eae10814167cc03e20cb09d0262a7b25d52097c492167435c8509ae0cc2c6d46.apk
```

```
import os
permission_list = []
error_list = []
apkname_list = []
for root, dirs, files in os.walk("Benign10", topdown=False):
        for name in files:

                path = os.path.join(root, name)

                get_permissions(path, "t")
```

```
['android.permission.READ_EXTERNAL_STORAGE', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.ACCESS_W
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.RECORD_AUDIO', 'android.permission.VIBRATE', 'andr
['android.permission.CHANGE_NETWORK_STATE', 'android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.GET_ACC
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.READ_EXTERNAL_STORAGE', 'android.permission.ACCESS
['android.permission.CHANGE_WIFI_MULTICAST_STATE', 'android.permission.CHANGE_NETWORK_STATE', 'com.dianxinos.optimiz
['com.moovechat.permission.C2D_MESSAGE', 'android.permission.ACCESS_NETWORK_STATE', 'com.android.vending.BILLING', '
['android.permission.FLASHLIGHT', 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.CAMERA', 'android.p
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.GET_ACCOUNTS', 'android.permission.VIBRATE', 'andr
['android.permission.READ_EXTERNAL_STORAGE', 'android.permission.ACCESS_NETWORK_STATE', 'com.appswiz.papersplease.pe
['android.permission.GET_ACCOUNTS', 'android.permission.INTERACT_ACROSS_USERS_FULL', 'com.yoox.permission.C2D_MESSAG
['android.permission.CHANGE_NETWORK_STATE', 'android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.READ_EX
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.VIBRATE', 'android.permission.READ_EXTERNAL_STORAG
['android.permission.ACCESS_NETWORK_STATE', 'android.permission.ACCESS_WIFI_STATE', 'com.android.vending.BILLING', '
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.GET_ACCOUNTS', 'android.permission.INTERNET', 'and
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.READ_EXTERNAL_STORAGE', 'android.permission.ACCESS
['android.permission.WRITE_EXTERNAL_STORAGE', 'android.permission.VIBRATE', 'android.permission.ACCESS_NETWORK_STATE
```



```
len(permission_list)
```

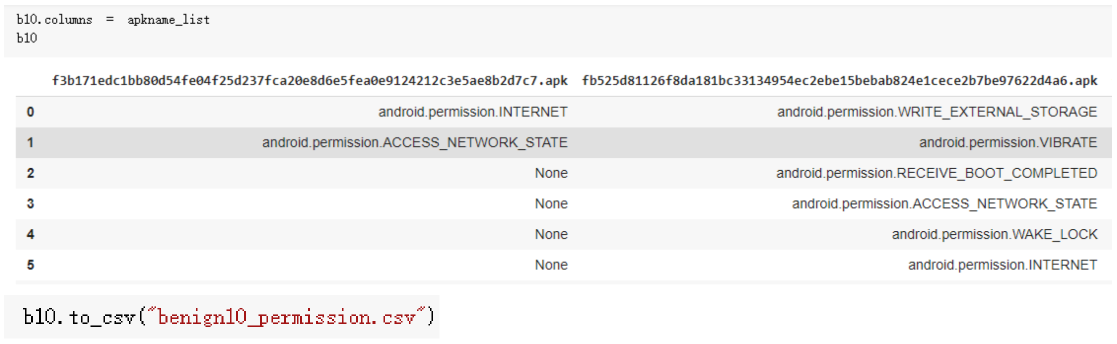
```
362
```

```
error_list
```

```
[]
```

```
import pandas as pd
b10 = pd.DataFrame(permission_list).T
b10
```

	0	1	
0	android.permission.INTERNET	android.permission.WRITE_EXTERNAL_STORAGE	android.perm
1	android.permission.ACCESS_NETWORK_STATE	android.permission.VIBRATE	
2	None	android.permission.RECEIVE_BOOT_COMPLETED	co
3	None	android.permission.ACCESS_NETWORK_STATE	



```
b10.columns = apkname_list
b10
```

	f3b171edc1bb80d54fe04f25d237fca20e8d6e5fea0e9124212c3e5ae8b2d7c7.apk	fb525d81126f8da181bc33134954ec2ebe15bebab824e1cece2b7be97622d4a6.apk
0	android.permission.INTERNET	android.permission.WRITE_EXTERNAL_STORAGE
1	android.permission.ACCESS_NETWORK_STATE	android.permission.VIBRATE
2	None	android.permission.RECEIVE_BOOT_COMPLETED
3	None	android.permission.ACCESS_NETWORK_STATE
4	None	android.permission.WAKE_LOCK
5	None	android.permission.INTERNET

```
b10.to_csv("benign10_permission.csv")
```

2. 前處理

Step1:讀入前面存的 13 個 csv 檔，對每個欄位的值做處理，只留 permission 的字

Step2:找出 SMS,Adware,Banking malware 排名前 50 的 permission

Step3:找出 Benign 的排名前 50 的 permission

Step4:用找出的 malware 和 Benign 的 permission，把全部的 apk 用成放入模型的 dataframe 資料集(apk_dataset.csv)

Step5:資料切成 train 和 test 的 data

程式碼：

- SMS

```
import pandas as pd
#sms
sms = pd.read_csv("sms_permission.csv")
sms
```

```
malware_total_permission = []
```

```
#刪除android.permission
import numpy as np

for column in sms.columns:
    l = []
    for v in sms[column]:
        #因為第一欄是索引且各欄位有NaN，所以先判斷是String的才要處理
        if isinstance(v, str):
            v = v.split(".")[-1] #只留最後一部分的字
            malware_total_permission.append(v)
        else:
            v = "non"
        #print(v)
        l.append(v)
    sms[column] = l
        #print(type(v))
```

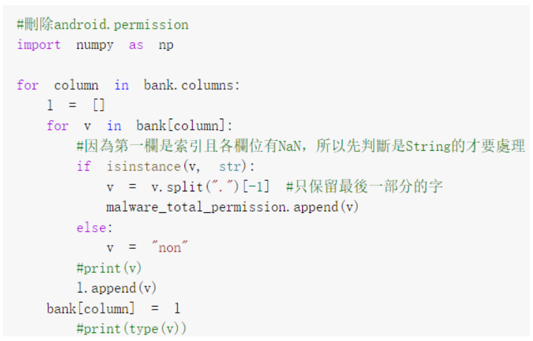
- Adware

```
#adware
adware = pd.read_csv("Adware_permission.csv")
adware
```

```
#刪除android.permission
import numpy as np

for column in adware.columns:
    l = []
    for v in adware[column]:
        #因為第一欄是索引且各欄位有NaN，所以先判斷是String的才要處理
        if isinstance(v, str):
            v = v.split(".")[-1] #只保留最後一部分的字
            malware_total_permission.append(v)
        else:
            v = "non"
        #print(v)
        l.append(v)
    adware[column] = l
        #print(type(v))
```

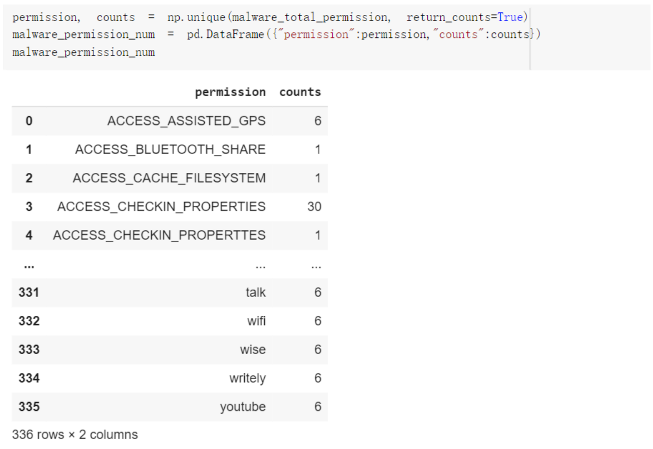
- Banking

```
#banking
bank = pd.read_csv("Banking_permission.csv")
bank
```



```python
#刪除android.permission
import numpy as np

for column in bank.columns:
    l = []
    for v in bank[column]:
        #因為第一欄是索引且各欄位有NaN，所以先判斷是String的才要處理
        if isinstance(v, str):
            v = v.split(".")[-1]  #只保留最後一部分的字
            malware_total_permission.append(v)
        else:
            v = "non"
        #print(v)
        l.append(v)
    bank[column] = l
        #print(type(v))
```

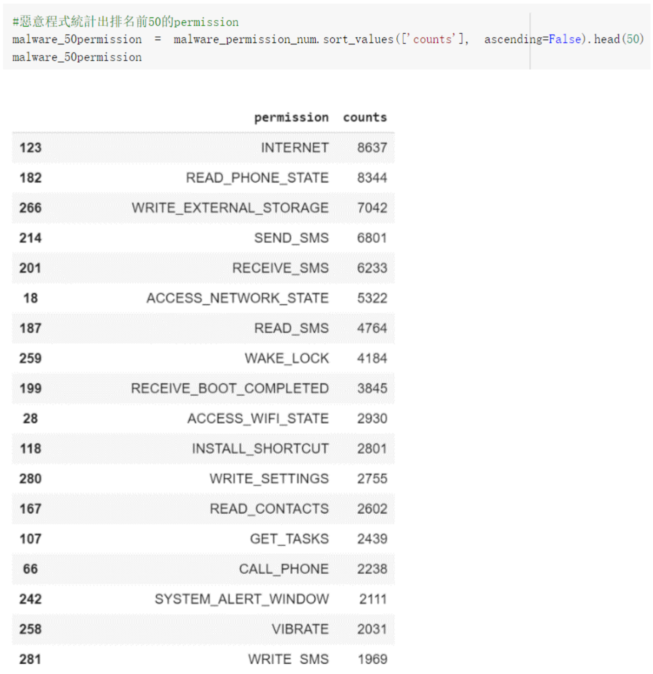
惡意程式排名前 50 個 permission



```python
permission, counts = np.unique(malware_total_permission, return_counts=True)
malware_permission_num = pd.DataFrame({"permission":permission,"counts":counts})
malware_permission_num
```

	permission	counts
0	ACCESS_ASSISTED_GPS	6
1	ACCESS_BLUETOOTH_SHARE	1
2	ACCESS_CACHE_FILESYSTEM	1
3	ACCESS_CHECKIN_PROPERTIES	30
4	ACCESS_CHECKIN_PROPERTTES	1
...	...	...
331	talk	6
332	wifi	6
333	wise	6
334	writely	6
335	youtube	6

336 rows × 2 columns



```python
#惡意程式統計出排名前50的permission
malware_50permission = malware_permission_num.sort_values(['counts'], ascending=False).head(50)
malware_50permission
```

	permission	counts
123	INTERNET	8637
182	READ_PHONE_STATE	8344
266	WRITE_EXTERNAL_STORAGE	7042
214	SEND_SMS	6801
201	RECEIVE_SMS	6233
18	ACCESS_NETWORK_STATE	5322
187	READ_SMS	4764
259	WAKE_LOCK	4184
199	RECEIVE_BOOT_COMPLETED	3845
28	ACCESS_WIFI_STATE	2930
118	INSTALL_SHORTCUT	2801
280	WRITE_SETTINGS	2755
167	READ_CONTACTS	2602
107	GET_TASKS	2439
66	CALL_PHONE	2238
242	SYSTEM_ALERT_WINDOW	2111
258	VIBRATE	2031
281	WRITE_SMS	1969

- Benign

(10 個 benign DataFrame 合併成一個 Dataframe)

```python
#benign
b1 = pd.read_csv("benign1_permission.csv")
b2 = pd.read_csv("benign2_permission.csv")
b3 = pd.read_csv("benign3_permission.csv")
b4 = pd.read_csv("benign4_permission.csv")
b5 = pd.read_csv("benign5_permission.csv")
b6 = pd.read_csv("benign6_permission.csv")
b7 = pd.read_csv("benign7_permission.csv")
b8 = pd.read_csv("benign8_permission.csv")
b9 = pd.read_csv("benign9_permission.csv")
b10 = pd.read_csv("benign10_permission.csv")
benign_column_name = []
b1_columnName = [c for c in b1.columns[1:]]
b2_columnName = [c for c in b2.columns[1:]]
b3_columnName = [c for c in b3.columns[1:]]
b4_columnName = [c for c in b4.columns[1:]]
b5_columnName = [c for c in b5.columns[1:]]
b6_columnName = [c for c in b6.columns[1:]]
b7_columnName = [c for c in b7.columns[1:]]
b8_columnName = [c for c in b8.columns[1:]]
b9_columnName = [c for c in b9.columns[1:]]
b10_columnName = [c for c in b10.columns[1:]]
benign_column_name = b1_columnName+b2_columnName+b3_columnName+b4_columnName+b5_columnName+b6_columnName+b7_columnName+b8_columnName+b9_columnName+b10_columnName
benign = pd.concat([b1[b1.columns[1:]],b2[b2.columns[1:]],b3[b3.columns[1:]],b4[b4.columns[1:]],
                    b5[b5.columns[1:]],b6[b6.columns[1:]],b7[b7.columns[1:]],
                    b8[b8.columns[1:]],b9[b9.columns[1:]],b10[b10.columns[1:]]], axis = 1, ignore_index=True)
benign.columns = benign_column_name
benign
```

```python
benign_total_permission = []
```

```python
#刪除android.permission
import numpy as np

for column in benign.columns:
    l = []
    for v in benign[column]:
        #因為第一欄是索引且各欄位有NaN，所以先判斷是String的才要處理
        if isinstance(v, str):
            v = v.split(".")[-1]  #只保留最後一部分的字
            benign_total_permission.append(v)
        else:
            v = "non"
        #print(v)
        l.append(v)
    benign[column] = l
        #print(type(v))
```

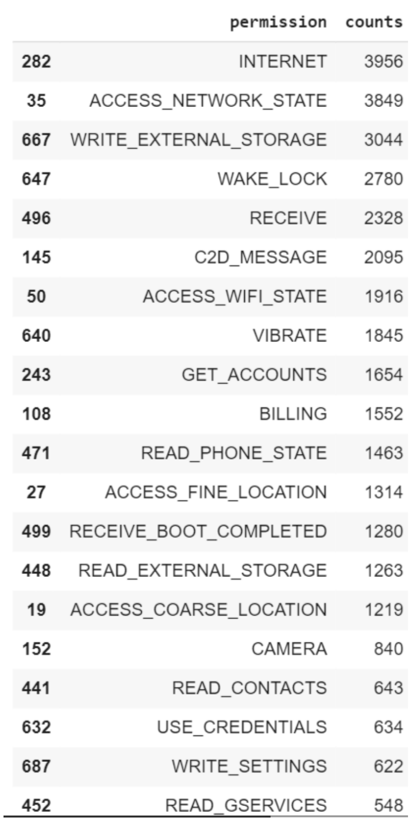
正常程式前 50 個 permissions

```python
permission, counts = np.unique(benign_total_permission, return_counts=True)
benign_permission_num = pd.DataFrame({"permission":permission,"counts":counts})
benign_permission_num
```

	permission	counts
0		1
1	A4S_SEND	5
2	AAM2	1
3	ACCCESS_NETWORK_STATE	1
4	ACCESS	21
...	...	...
746	videomeetings	1
747	wake_lock	1
748	wise	5
749	write	2
750	writely	5

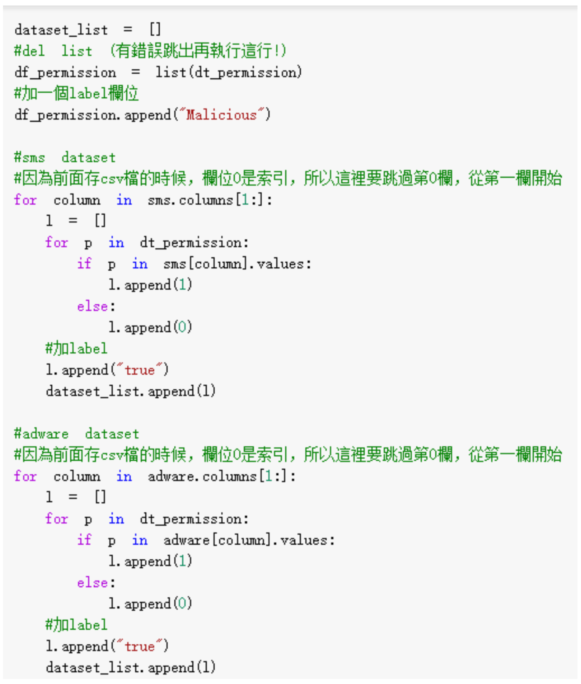
751 rows × 2 columns

```python
#正常程式統計出排名前50的permission
benign_50permission = benign_permission_num.sort_values(['counts'], ascending=False).head(50)
benign_50permission
```



	permission	counts
282	INTERNET	3956
35	ACCESS_NETWORK_STATE	3849
667	WRITE_EXTERNAL_STORAGE	3044
647	WAKE_LOCK	2780
496	RECEIVE	2328
145	C2D_MESSAGE	2095
50	ACCESS_WIFI_STATE	1916
640	VIBRATE	1845
243	GET_ACCOUNTS	1654
108	BILLING	1552
471	READ_PHONE_STATE	1463
27	ACCESS_FINE_LOCATION	1314
499	RECEIVE_BOOT_COMPLETED	1280
448	READ_EXTERNAL_STORAGE	1263
19	ACCESS_COARSE_LOCATION	1219
152	CAMERA	840
441	READ_CONTACTS	643
632	USE_CREDENTIALS	634
687	WRITE_SETTINGS	622
452	READ_GSERVICES	548

用找出的惡意程式前 50 個 permissions 和正常程式前 50 個 permissions 建立成要丟到模型的資料集



```
dataset_list = []
#del list (有錯誤跳出再執行這行!)
df_permission = list(dt_permission)
#加一個label欄位
df_permission.append("Malicious")

#sms dataset
#因為前面存csv檔的時候，欄位0是索引，所以這裡要跳過第0欄，從第一欄開始
for column in sms.columns[1:]:
    l = []
    for p in dt_permission:
        if p in sms[column].values:
            l.append(1)
        else:
            l.append(0)
    #加label
    l.append("true")
    dataset_list.append(l)

#adware dataset
#因為前面存csv檔的時候，欄位0是索引，所以這裡要跳過第0欄，從第一欄開始
for column in adware.columns[1:]:
    l = []
    for p in dt_permission:
        if p in adware[column].values:
            l.append(1)
        else:
            l.append(0)
    #加label
    l.append("true")
    dataset_list.append(l)
```

```
#banking dataset
#因為前面存csv檔的時候，欄位0是索引，所以這裡要跳過第0欄，從第一欄開始
for column in bank.columns[1:]:
    l = []
    for p in dt_permission:
        if p in bank[column].values:
            l.append(1)
        else:
            l.append(0)
    #加label
    l.append("true")
    dataset_list.append(l)

#benign dataset
#因為前面存csv檔的時候，欄位0是索引，所以這裡要跳過第0欄，從第一欄開始
for column in benign.columns[1:]:
    l = []
    for p in dt_permission:
        if p in benign[column].values:
            l.append(1)
        else:
            l.append(0)
    #加label
    l.append("false")
    dataset_list.append(l)
```

```
dataset = pd.DataFrame(dataset_list, columns=df_permission)
dataset
```

因為現在資料集太整齊，要把它打散一點

```
#把資料集打亂
from sklearn.utils import shuffle
dataset = shuffle(dataset,)
dataset
```

最後將資料集儲存成 csc 檔(此為跑模型的資料集)

```
#儲存資料集成csv檔
dataset.to_csv("apk_dataset.csv",index=False)
```

接下來要將 Malicious 欄位轉成 0,1 形式，因為資料集是用 true, false 方便

觀看所以放入模型前要轉換

```
#把target label轉成0,1形式
dataset['Malicious'] = dataset['Malicious'].astype('category').cat.codes
dataset.head(10)
```

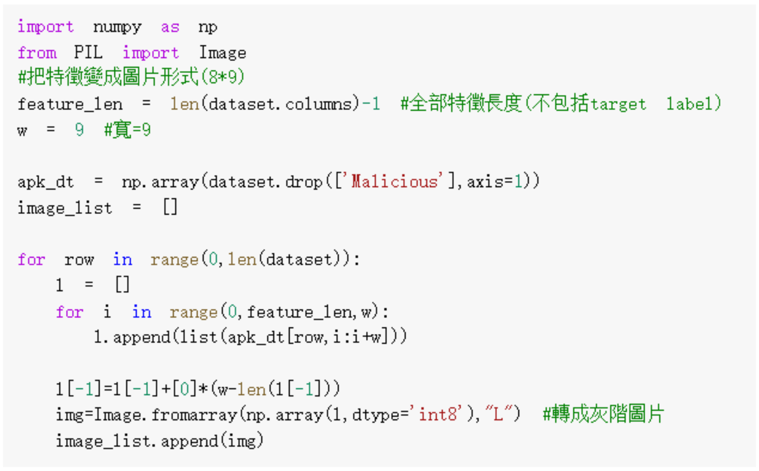
前處理的最後一步是將資料集切成訓練和測試的(SVM 和 RandomForest 都

是訓練資料:80%,測試資料:20%)

```
#資料集切割(train:80%, test:20%)
from sklearn.model_selection import train_test_split
X = dataset.drop(["Malicious"],axis = 1)
y= dataset['Malicious']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

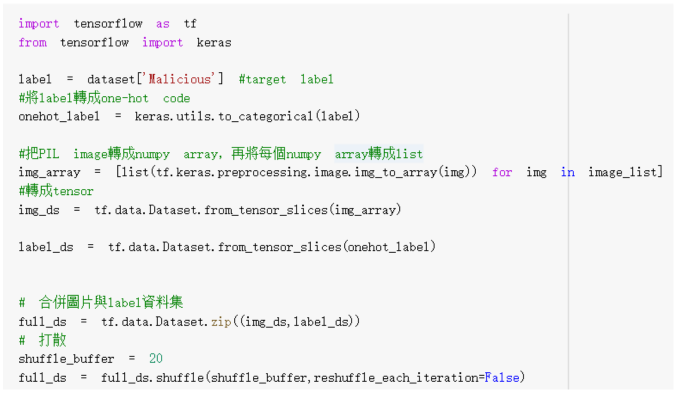
CNN 前處理：

Step1:將 permission 的資料集轉成 8*9 圖片形式(PIL image)



```
import numpy as np
from PIL import Image
#把特徵變成圖片形式(8*9)
feature_len = len(dataset.columns)-1  #全部特徵長度(不包括target label)
w = 9  #寬=9

apk_dt = np.array(dataset.drop(['Malicious'],axis=1))
image_list = []

for row in range(0,len(dataset)):
    l = []
    for i in range(0,feature_len,w):
        l.append(list(apk_dt[row,i:i+w]))

    l[-1]=l[-1]+[0]*(w-len(l[-1]))
    img=Image.fromarray(np.array(l,dtype='int8'),"L")  #轉成灰階圖片
    image_list.append(img)
```

Step2:將 PIL image 轉成 tensor



```
import tensorflow as tf
from tensorflow import keras

label = dataset['Malicious']  #target label
#將label轉成one-hot code
onehot_label = keras.utils.to_categorical(label)

#把PIL image轉成numpy array，再將每個numpy array轉成list
img_array = [list(tf.keras.preprocessing.image.img_to_array(img)) for img in image_list]
#轉成tensor
img_ds = tf.data.Dataset.from_tensor_slices(img_array)

label_ds = tf.data.Dataset.from_tensor_slices(onehot_label)


# 合併圖片與label資料集
full_ds = tf.data.Dataset.zip((img_ds,label_ds))
# 打散
shuffle_buffer = 20
full_ds = full_ds.shuffle(shuffle_buffer,reshuffle_each_iteration=False)
```

Step3:將 tensorflow dataset 切成 train, validation, test 的 dataset，並添加

batch(batch_size=100)

```
# 切割成training data與validation data和testing data
#train & test data
total_len = len(dataset)
train_len = int(0.8*total_len)
test_len = total_len - train_len
train_ds = full_ds.take(train_len)
test_ds = full_ds.take(test_len)
#train & validation dat
train_total_len = train_len
train_len = int(0.8*train_len)
val_len = train_total_len - train_len
val_ds = train_ds.skip(train_len)
train_ds = train_ds.take(train_len)


print("train size : ",train_len," val size : ",val_len)

# 添加batch
batch_size = 100

train_ds = train_ds.batch(batch_size)
val_ds = val_ds.batch(batch_size)
```

```
train size :  8224  val size :  2057
```

```
# 查看添加batch後的維度
trainiter = iter(train_ds)
x,y = trainiter.next()
print("training image batch shape : ",x.shape)
print("training label batch shape : ",y.shape)
```

```
training image batch shape :  (100, 8, 9, 1)
training label batch shape :  (100, 2)
```

```
# 查看添加batch後的維度
trainiter = iter(val_ds)
x,y = trainiter.next()
print("training image batch shape : ",x.shape)
print("training label batch shape : ",y.shape)
```

```
training image batch shape :  (100, 8, 9, 1)
training label batch shape :  (100, 2)
```

```
# 查看添加batch後的維度
trainiter = iter(test_ds)
x,y = trainiter.next()
print("training image batch shape : ",x.shape)
print("training label batch shape : ",y.shape)
```

```
training image batch shape :  (8, 9, 1)
training label batch shape :  (2,)
```

3. 建立 SVM, RandomForest 和 CNN 模型

SVM:

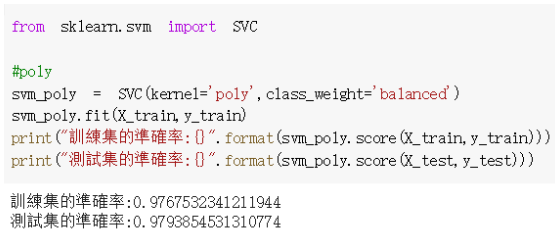
- Svm 用兩種 kernel function：polynomial 和 RBF

⇨ RBF 表現較好

- 資料訓練方式:Hold-out(train:80%, test:20%,截圖在上面前處理的部分)

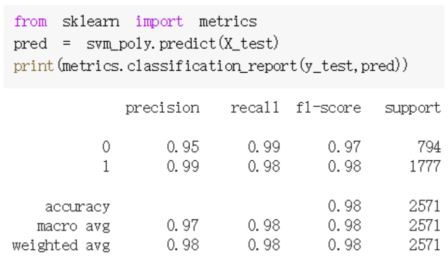
- 因為資料集的惡意 apk 數量和正常 apk 數量有落差，所以參數有加 class_weight='　balanced'

Polynomial



```
from sklearn.svm import SVC

#poly
svm_poly = SVC(kernel='poly',class_weight='balanced')
svm_poly.fit(X_train,y_train)
print("訓練集的準確率:{}".format(svm_poly.score(X_train,y_train)))
print("測試集的準確率:{}".format(svm_poly.score(X_test,y_test)))
```

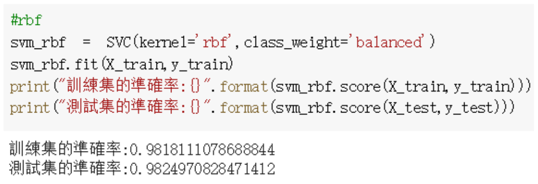
```
訓練集的準確率:0.9767532341211944
測試集的準確率:0.9793854531310774
```



```
from sklearn import metrics
pred = svm_poly.predict(X_test)
print(metrics.classification_report(y_test,pred))
```

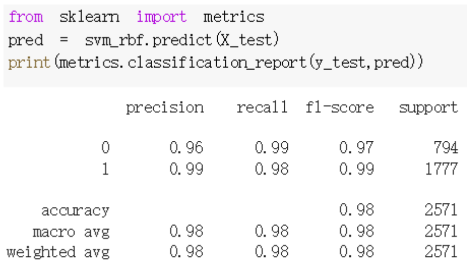
```
              precision    recall  f1-score   support

           0       0.95      0.99      0.97       794
           1       0.99      0.98      0.98      1777

    accuracy                           0.98      2571
   macro avg       0.97      0.98      0.98      2571
weighted avg       0.98      0.98      0.98      2571
```

RBF



```
#rbf
svm_rbf = SVC(kernel='rbf',class_weight='balanced')
svm_rbf.fit(X_train,y_train)
print("訓練集的準確率:{}".format(svm_rbf.score(X_train,y_train)))
print("測試集的準確率:{}".format(svm_rbf.score(X_test,y_test)))
```

```
訓練集的準確率:0.9818111078688844
測試集的準確率:0.9824970828471412
```

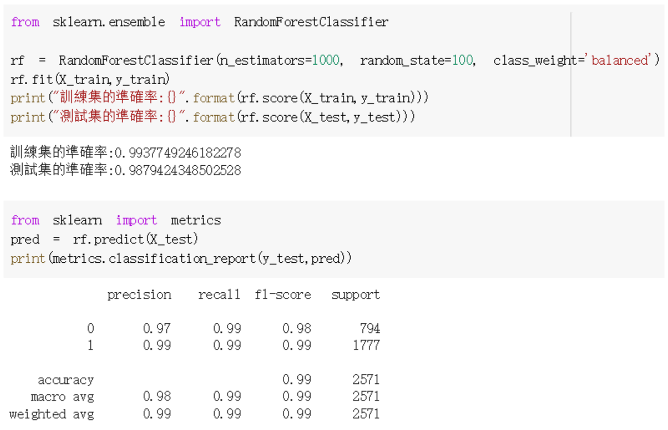


```
from sklearn import metrics
pred = svm_rbf.predict(X_test)
print(metrics.classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.96      0.99      0.97       794
           1       0.99      0.98      0.99      1777

    accuracy                           0.98      2571
   macro avg       0.98      0.98      0.98      2571
weighted avg       0.98      0.98      0.98      2571
```

RandomForest:

- 資料訓練方式:Hold-out(train:80%, test:20%,截圖在上面前處理的部分)
- 因為資料集的惡意 apk 數量和正常 apk 數量有落差，所以參數有加 class_weight='balanced'



```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=1000, random_state=100, class_weight='balanced')
rf.fit(X_train,y_train)
print("訓練集的準確率:{}".format(rf.score(X_train,y_train)))
print("測試集的準確率:{}".format(rf.score(X_test,y_test)))
```

```
訓練集的準確率:0.9937749246182278
測試集的準確率:0.9879424348502528
```

```
from sklearn import metrics
pred = rf.predict(X_test)
print(metrics.classification_report(y_test,pred))
```

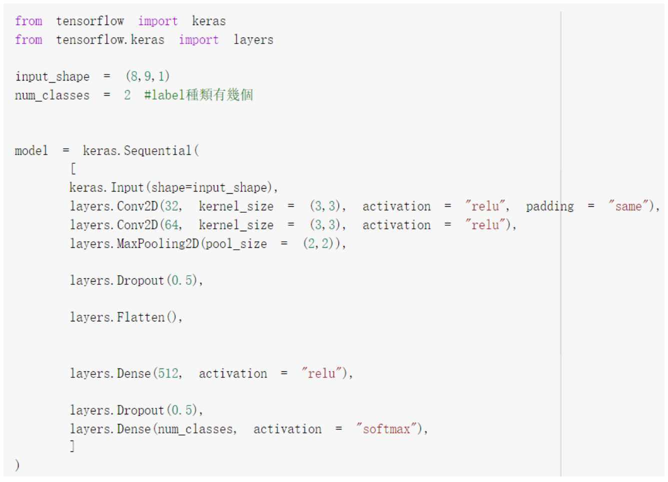
```
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       794
           1       0.99      0.99      0.99      1777

    accuracy                           0.99      2571
   macro avg       0.98      0.99      0.99      2571
weighted avg       0.99      0.99      0.99      2571
```

CNN:

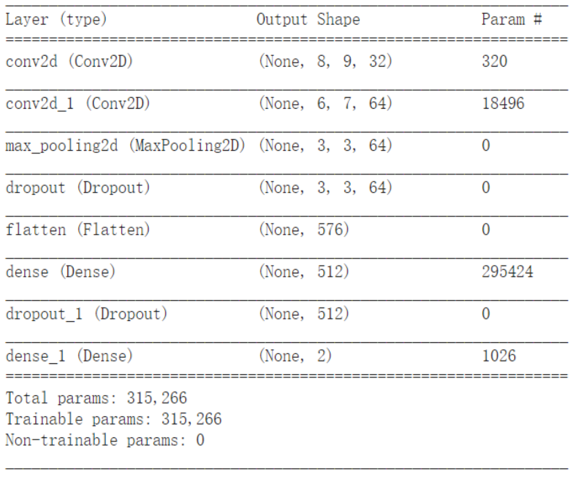
Input : 將 PIL image 轉成 tensor，shape=(8,9,1)，batch=100(截圖在上面前處理的地方)

Output : 2 維度，因為最後一層用 softmax 所以 output 可以看成機率

模型架構 1&參數:



```
from tensorflow import keras
from tensorflow.keras import layers

input_shape = (8,9,1)
num_classes = 2 #label種類有幾個


model = keras.Sequential(
        [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size = (3,3), activation = "relu", padding = "same"),
        layers.Conv2D(64, kernel_size = (3,3), activation = "relu"),
        layers.MaxPooling2D(pool_size = (2,2)),

        layers.Dropout(0.5),

        layers.Flatten(),


        layers.Dense(512, activation = "relu"),

        layers.Dropout(0.5),
        layers.Dense(num_classes, activation = "softmax"),
        ]
)
```



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 8, 9, 32)          320
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 6, 7, 64)          18496
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 3, 3, 64)          0
_________________________________________________________________
dropout (Dropout)            (None, 3, 3, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 576)               0
_________________________________________________________________
dense (Dense)                (None, 512)               295424
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 1026
=================================================================
Total params: 315,266
Trainable params: 315,266
Non-trainable params: 0
_________________________________________________________________
```

模型架構說明：

第一個模型做兩次卷積，第一次卷積有 32 個 3*3 的 kernel。參數有加 padding=same，因為圖片格式為 8*9，size 較小，所以第一次卷積讓 padding=same 讓大小不變。第二次卷積有 64 個 3*3 的 kernel。再來做一次 pooling，flatten 前先做一次 dropout。後面 NN 只有兩層，一

層有 512 個神經元，然後 dropout，最後一層有 2 個神經元，並用 softmax 當作 activation function 讓結果加起來=1。

訓練計畫制定:

```
epochs = 100

model.compile(loss = "categorical_crossentropy", optimizer = "adam", metrics = ["accuracy"])
#設置earlt stopping
earlystop_callback = tf.keras.callbacks.EarlyStopping(monitor='val_accuracy', min_delta=0.0001, patience=15, mode = 'max')

history = model.fit(train_ds, epochs=epochs, validation_data=val_ds, callbacks=[earlystop_callback])
```

Loss function : categorical_crossentropy

Optimizer : Adam

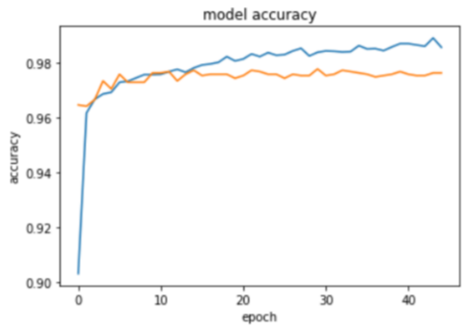
Metrics : accuracy

參數有用 early stopping，讓 validation 的準確率沒有變好的時候就停止訓練

模型評估

```
print(history.history.keys())

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.show()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```



藍色線為 train 的準確率，黃色線為 validation 的準確率，可以看出兩個準確率沒有落差太大，所以此模型架構訓練出來的模型沒有 overfitting，表現還行。

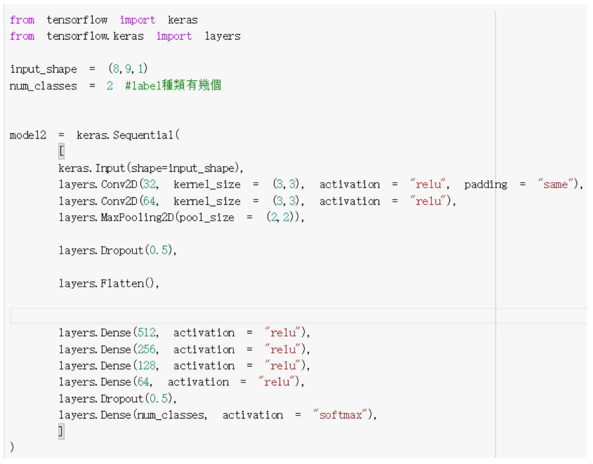
Test Accuracy :

```
# 讀入測試資料並評估模型

#test dataset添加batch
test_ds = test_ds.batch(batch_size)
score = model.evaluate(test_ds)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

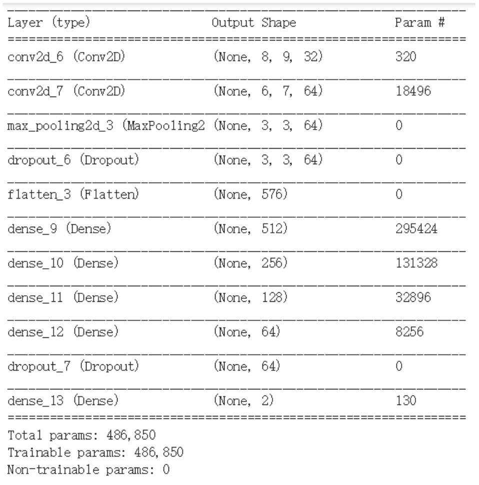
```
26/26 [==============================] - 0s 8ms/step - loss: 0.0247 - accuracy: 0.9922
Test loss: 0.02474989742040634
Test accuracy: 0.9922209382057190
```

模型架構 2&參數:

因為第一個模型沒有 overfitting，所以第二個模型把 NN 層數變多，試試看準確率有無提高



```
from  tensorflow  import  keras
from  tensorflow.keras  import  layers

input_shape  =  (8,9,1)
num_classes  =  2  #label種類有幾個


model2  =  keras.Sequential(
        [
        keras.Input(shape=input_shape),
        layers.Conv2D(32,  kernel_size  =  (3,3),  activation  =  "relu",  padding  =  "same"),
        layers.Conv2D(64,  kernel_size  =  (3,3),  activation  =  "relu"),
        layers.MaxPooling2D(pool_size  =  (2,2)),

        layers.Dropout(0.5),

        layers.Flatten(),


        layers.Dense(512,  activation  =  "relu"),
        layers.Dense(256,  activation  =  "relu"),
        layers.Dense(128,  activation  =  "relu"),
        layers.Dense(64,  activation  =  "relu"),
        layers.Dropout(0.5),
        layers.Dense(num_classes,  activation  =  "softmax"),
        ]
)
```

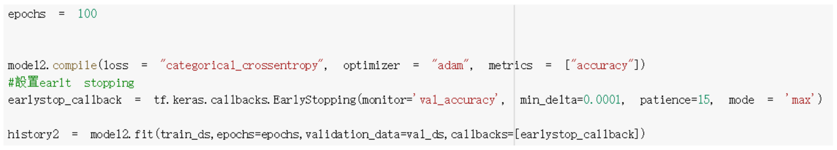


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 8, 9, 32)          320
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 6, 7, 64)          18496
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
dropout_6 (Dropout)          (None, 3, 3, 64)          0
_________________________________________________________________
flatten_3 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_9 (Dense)              (None, 512)               295424
_________________________________________________________________
dense_10 (Dense)             (None, 256)               131328
_________________________________________________________________
dense_11 (Dense)             (None, 128)               32896
_________________________________________________________________
dense_12 (Dense)             (None, 64)                8256
_________________________________________________________________
dropout_7 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_13 (Dense)             (None, 2)                 130
=================================================================
Total params: 486,850
Trainable params: 486,850
Non-trainable params: 0
```

模型架構說明：

第二個模型一樣做兩次卷積，第一次卷積有 32 個 3*3 的 kernel。參數有加 padding=same，因為圖片格式為 8*9，size 較小，所以第一次卷積讓 padding=same 讓大小不變。第二次卷積有 64 個 3*3 的 kernel。再來做一次 pooling，flatten 前先做一次 dropout，以上跟第一個模型一樣。後面 NN 變五層，分別是 512、256、128、64 個神經元，最後一層有 2 個神經元，並用 softmax 讓結果加起來=1。

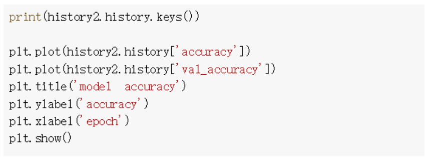
訓練計畫制定:



```
epochs  =  100


model2.compile(loss  =  "categorical_crossentropy",  optimizer  =  "adam",  metrics  =  ["accuracy"])
#設置earlt  stopping
earlystop_callback  =  tf.keras.callbacks.EarlyStopping(monitor='val_accuracy',  min_delta=0.0001,  patience=15,  mode  =  'max')

history2  =  model2.fit(train_ds,epochs=epochs,validation_data=val_ds,callbacks=[earlystop_callback])
```

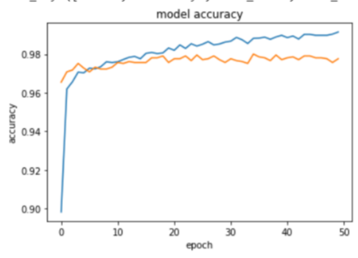
跟第一個模型一樣

模型評估:



```
print(history2.history.keys())

plt.plot(history2.history['accuracy'])
plt.plot(history2.history['val_accuracy'])
plt.title('model  accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.show()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```



藍色線為 train 的準確率，黃色線為 validation 的準確率，可以看出跟第一個模型一樣，兩個準確率沒有落差太大，所以此模型架構訓練出來的模型也沒有 overfitting。

Test Accuracy :

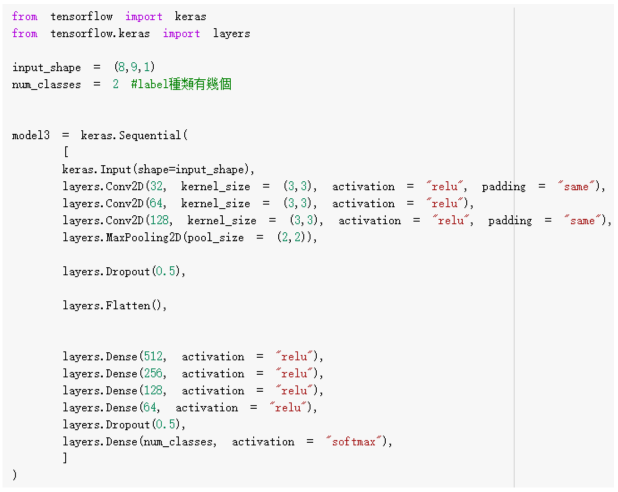
```
#  讀入測試資料並評估模型
test_ds  =  test_ds.batch(batch_size)
score  =  model2.evaluate(test_ds)
print("Test  loss:",  score[0])
print("Test  accuracy:",  score[1])
```

```
26/26 [==============================] - 0s 10ms/step - loss: 0.0174 - accuracy: 0.9949
Test loss: 0.017420515418052673
Test accuracy: 0.9949436187744141
```

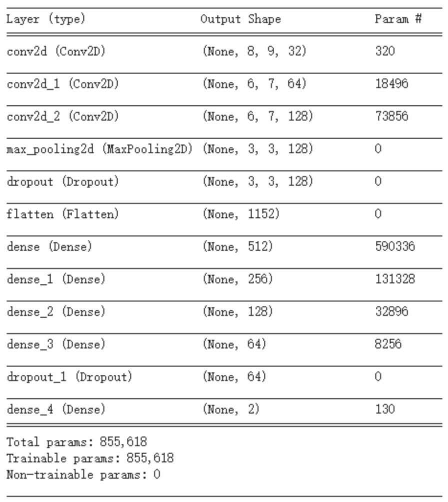
準確率比起第一個模型，稍微提升了一點

模型架構 3&參數:

第三個模型多加一層卷積層，試試看準確率有無提高



```
from  tensorflow  import  keras
from  tensorflow.keras  import  layers

input_shape  =  (8,9,1)
num_classes  =  2  #label種類有幾個


model3  =  keras.Sequential(
        [
        keras.Input(shape=input_shape),
        layers.Conv2D(32,  kernel_size  =  (3,3),  activation  =  "relu",  padding  =  "same"),
        layers.Conv2D(64,  kernel_size  =  (3,3),  activation  =  "relu"),
        layers.Conv2D(128,  kernel_size  =  (3,3),  activation  =  "relu",  padding  =  "same"),
        layers.MaxPooling2D(pool_size  =  (2,2)),

        layers.Dropout(0.5),

        layers.Flatten(),


        layers.Dense(512,  activation  =  "relu"),
        layers.Dense(256,  activation  =  "relu"),
        layers.Dense(128,  activation  =  "relu"),
        layers.Dense(64,  activation  =  "relu"),
        layers.Dropout(0.5),
        layers.Dense(num_classes,  activation  =  "softmax"),
        ]
)
```



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 8, 9, 32)          320
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 6, 7, 64)          18496
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 6, 7, 128)         73856
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 3, 3, 128)         0
_________________________________________________________________
dropout (Dropout)            (None, 3, 3, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 1152)              0
_________________________________________________________________
dense (Dense)                (None, 512)               590336
_________________________________________________________________
dense_1 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_2 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_3 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 130
=================================================================
Total params: 855,618
Trainable params: 855,618
Non-trainable params: 0
_________________________________________________________________
```

模型架構說明：

第三個模型做三次卷積，前兩次卷積跟前兩個模型架構一樣，第三次的卷積有 128 個 3*3 的 kernel。參數有加 padding=same，讓 feature map 的大小不變。再來做一次 pooling，flatten 前先做一次 dropout，後面架構跟第二個模型一樣。

訓練計畫制定:

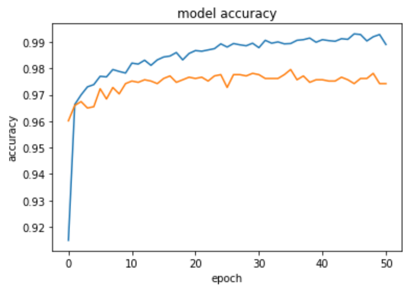
```
epochs  =  100


model3.compile(loss  =  "categorical_crossentropy",  optimizer  =  "adam",  metrics  =  ["accuracy"])
#設置earlt  stopping
earlystop_callback  =  tf.keras.callbacks.EarlyStopping(monitor='val_accuracy',  min_delta=0.0001,  patience=15,  mode  =  'max')

history3  =  model3.fit(train_ds,epochs=epochs,validation_data=val_ds,callbacks=[earlystop_callback])
```

跟前面的模型一樣

模型評估:

```
print(history3.history.keys())

plt.plot(history3.history['accuracy'])
plt.plot(history3.history['val_accuracy'])
plt.title('model  accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.show()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```



藍色線為 train 的準確率，黃色線為 validation 的準確率，兩個的準確率落差也沒有太大，所以此模型架構訓練出來的模型也沒有 overfitting。

Test Accuracy :

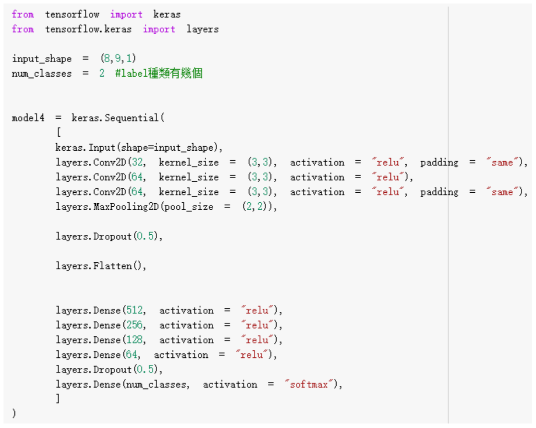
```
#  讀入測試資料並評估模型
test_ds  =  test_ds.unbatch()
test_ds  =  test_ds.batch(batch_size)
score  =  model3.evaluate(test_ds)
print("Test  loss:",   score[0])
print("Test  accuracy:",   score[1])
```

```
26/26 [==============================] - 0s 13ms/step - loss: 0.0204 - accuracy: 0.9930
Test loss: 0.020411128178238870
Test accuracy: 0.9929988384246826
```

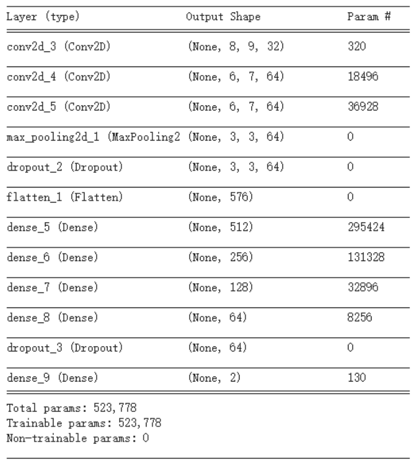
跟第一個模型的準確率差不多，不過比第二個模型的準確率低一點點

模型架構 4&參數:

對第三個模型做一點調整，將第三層卷積的 kernel 數量減少成 64 個，試試看準確率有無提高



```
from  tensorflow  import  keras
from  tensorflow.keras  import  layers

input_shape  =  (8,9,1)
num_classes  =  2  #label種類有幾個


model4  =  keras.Sequential(
        [
        keras.Input(shape=input_shape),
        layers.Conv2D(32,  kernel_size  =  (3,3),  activation  =  "relu",  padding  =  "same"),
        layers.Conv2D(64,  kernel_size  =  (3,3),  activation  =  "relu"),
        layers.Conv2D(64,  kernel_size  =  (3,3),  activation  =  "relu",  padding  =  "same"),
        layers.MaxPooling2D(pool_size  =  (2,2)),

        layers.Dropout(0.5),

        layers.Flatten(),


        layers.Dense(512,  activation  =  "relu"),
        layers.Dense(256,  activation  =  "relu"),
        layers.Dense(128,  activation  =  "relu"),
        layers.Dense(64,  activation  =  "relu"),
        layers.Dropout(0.5),
        layers.Dense(num_classes,  activation  =  "softmax"),
        ]
)
```



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 8, 9, 32)          320
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 6, 7, 64)          18496
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 6, 7, 64)          36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 3, 3, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_5 (Dense)              (None, 512)               295424
_________________________________________________________________
dense_6 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_7 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_8 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_9 (Dense)              (None, 2)                 130
=================================================================
Total params: 523,778
Trainable params: 523,778
Non-trainable params: 0
_________________________________________________________________
```

模型架構說明：

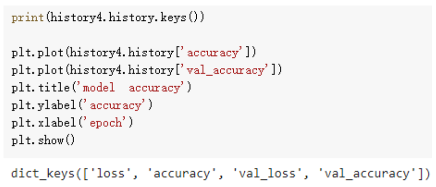
第四個模型做三次卷積，前兩次卷積跟前三個模型架構一樣，第三次的卷積調成 64 個 3*3 的

kernel。參數有加 padding=same，讓 feature map 的大小不變。再來做一次 pooling，flatten 前先做一次 dropout，後面架構跟第三個模型一樣。

訓練計畫制定:

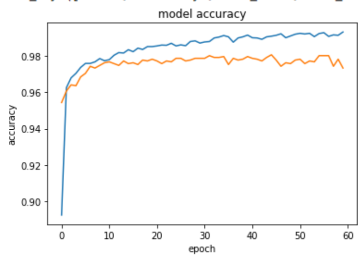
與前三個模型一樣

模型評估:



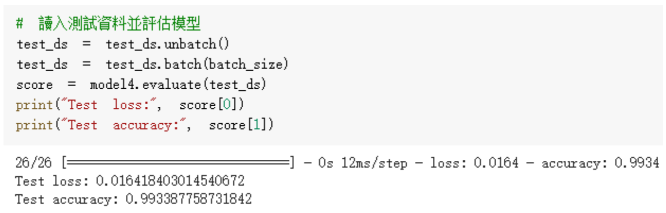
```
print(history4.history.keys())

plt.plot(history4.history['accuracy'])
plt.plot(history4.history['val_accuracy'])
plt.title('model  accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.show()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```



藍色線一樣為 train 的準確率，黃色線為 validation 的準確率，兩個的準確率落差沒有太大，所以此模型架構訓練出來的模型也沒有 overfitting。

Test Accuracy :



```
#  讀入測試資料並評估模型
test_ds  =  test_ds.unbatch()
test_ds  =  test_ds.batch(batch_size)
score  =  model4.evaluate(test_ds)
print("Test  loss:",  score[0])
print("Test  accuracy:",  score[1])
```

```
26/26 [==============================] - 0s 12ms/step - loss: 0.0164 - accuracy: 0.9934
Test loss: 0.016418403014540672
Test accuracy: 0.993387758731842
```

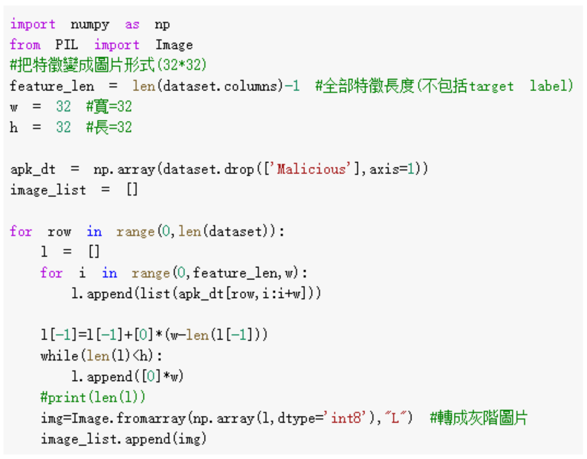
比第三個模型的準確率高一點，不過還是比第二個模型的準確率低一點點

因為四個模型中，第二個模型的準確率最高(0.994)，所以卷積層做兩次，一次 pooling，然後接五層 fully-connected layers，做一次 dropout 的表現最好。

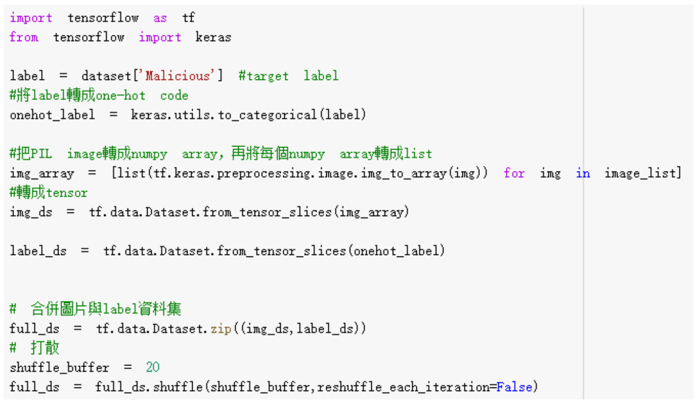
4. 【加分題】使用現有 CNN 架構訓練

使用 vgg16，因為 cnn 的卷積層數很多，所以圖片大小改成 32*32，因此前處理再做一次

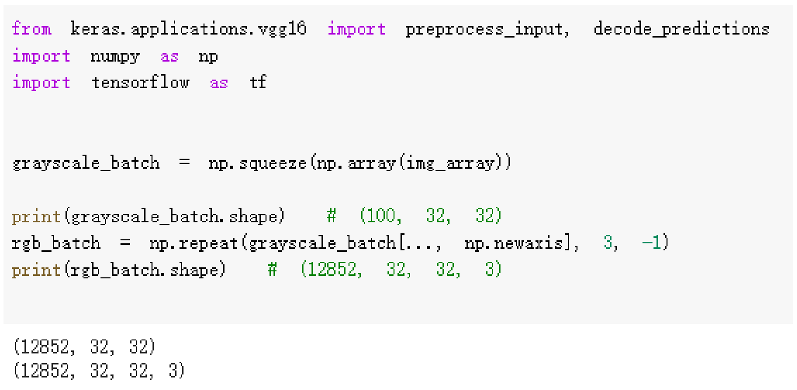
前處理



```
import numpy as np
from PIL import Image
#把特徵變成圖片形式(32*32)
feature_len = len(dataset.columns)-1 #全部特徵長度(不包括target label)
w = 32 #寬=32
h = 32 #長=32

apk_dt = np.array(dataset.drop(['Malicious'],axis=1))
image_list = []

for row in range(0,len(dataset)):
    l = []
    for i in range(0,feature_len,w):
        l.append(list(apk_dt[row,i:i+w]))

    l[-1]=l[-1]+[0]*(w-len(l[-1]))
    while(len(l)<h):
        l.append([0]*w)
    #print(len(l))
    img=Image.fromarray(np.array(l,dtype='int8'),"L") #轉成灰階圖片
    image_list.append(img)
```



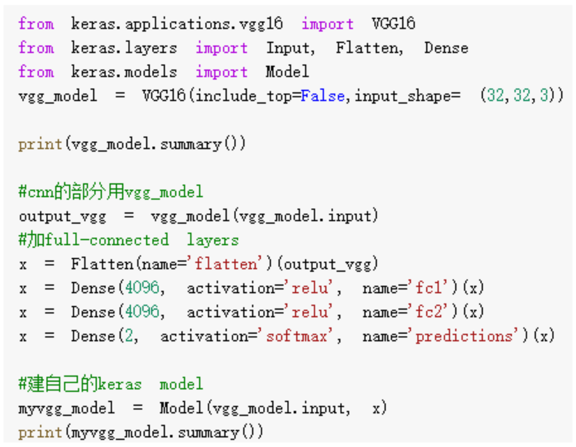
```
import tensorflow as tf
from tensorflow import keras

label = dataset['Malicious'] #target label
#將label轉成one-hot code
onehot_label = keras.utils.to_categorical(label)

#把PIL image轉成numpy array，再將每個numpy array轉成list
img_array = [list(tf.keras.preprocessing.image.img_to_array(img)) for img in image_list]
#轉成tensor
img_ds = tf.data.Dataset.from_tensor_slices(img_array)

label_ds = tf.data.Dataset.from_tensor_slices(onehot_label)


# 合併圖片與label資料集
full_ds = tf.data.Dataset.zip((img_ds,label_ds))
# 打散
shuffle_buffer = 20
full_ds = full_ds.shuffle(shuffle_buffer,reshuffle_each_iteration=False)
```



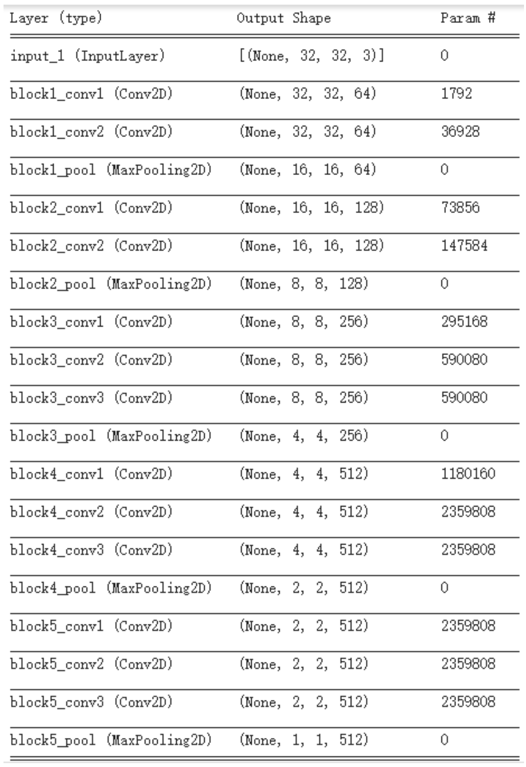
```
from keras.applications.vgg16 import preprocess_input, decode_predictions
import numpy as np
import tensorflow as tf


grayscale_batch = np.squeeze(np.array(img_array))

print(grayscale_batch.shape)  # (100, 32, 32)
rgb_batch = np.repeat(grayscale_batch[..., np.newaxis], 3, -1)
print(rgb_batch.shape)  # (12852, 32, 32, 3)
```

```
(12852, 32, 32)
(12852, 32, 32, 3)
```

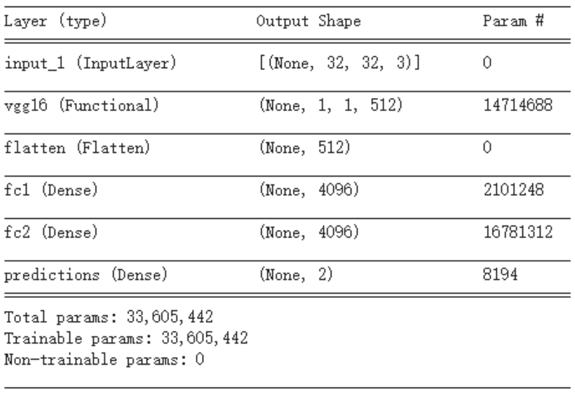
建立 Vgg16 模型



```
from keras.applications.vgg16 import VGG16
from keras.layers import Input, Flatten, Dense
from keras.models import Model
vgg_model = VGG16(include_top=False,input_shape= (32,32,3))

print(vgg_model.summary())

#cnn的部分用vgg_model
output_vgg = vgg_model(vgg_model.input)
#加full-connected layers
x = Flatten(name='flatten')(output_vgg)
x = Dense(4096, activation='relu', name='fc1')(x)
x = Dense(4096, activation='relu', name='fc2')(x)
x = Dense(2, activation='softmax', name='predictions')(x)

#建自己的keras model
myvgg_model = Model(vgg_model.input, x)
print(myvgg_model.summary())
```

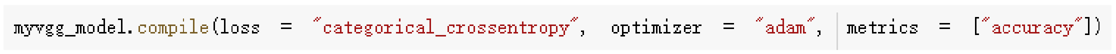


```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 32, 32, 3)]       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 32, 32, 64)        1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 32, 32, 64)        36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 16, 16, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 16, 16, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 16, 16, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 8, 8, 128)         0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 8, 8, 256)         295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 8, 8, 256)         590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 8, 8, 256)         590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 4, 4, 256)         0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 4, 4, 512)         1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 2, 2, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 2, 2, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 2, 2, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 2, 2, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 1, 1, 512)         0
=================================================================
```



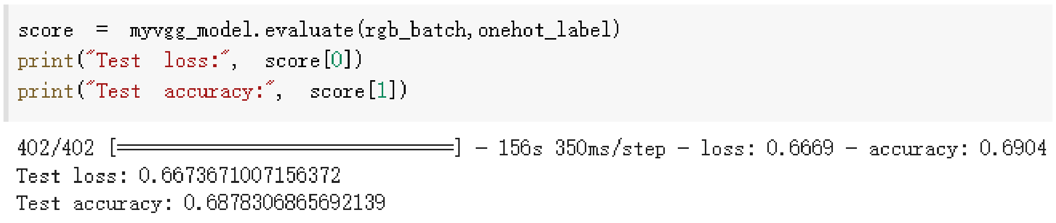
```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 32, 32, 3)]       0
_________________________________________________________________
vgg16 (Functional)           (None, 1, 1, 512)         14714688
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              2101248
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
predictions (Dense)          (None, 2)                 8194
=================================================================
Total params: 33,605,442
Trainable params: 33,605,442
Non-trainable params: 0
_________________________________________________________________
```

訓練計畫制定:



```
myvgg_model.compile(loss = "categorical_crossentropy", optimizer = "adam", metrics = ["accuracy"])
```

Test Accuracy :



```
score = myvgg_model.evaluate(rgb_batch,onehot_label)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
402/402 [==============================] - 156s 350ms/step - loss: 0.6669 - accuracy: 0.6904
Test loss: 0.6673671007156372
Test accuracy: 0.6878306865692139
```

5. 評估報告

- Svm 不同 kernel function 表現

Model	Test Accuracy
Polynomial	0.9794
RBF	0.9825

- CNN 不同超參數表現

Model	Test Accuracy

CNN1
2 層卷積(32,64)
1 次 pooling
Dropout(0.5)
Flatten
Dense(512, relu)
Dropout(0.5)
Dense(2, softmax)
	0.9922

CNN2
2 層卷積(32,64)
1 次 pooling
Dropout(0.5)
Flatten
Dense(512, relu)
Dense(256, relu)
Dense(128, relu)
Dense(64, relu)
Dropout(0.5)
Dense(2, softmax)
	0.9949

CNN3 3 層卷積(32,64,128) 1 次 pooling Dropout(0.5) Flatten Dense(512, relu) Dense(256, relu)	0.9930

Dense(128, relu)
Dense(64, relu)
Dropout(0.5)
Dense(2, softmax)	

CNN4
3 層卷積(32,64,64)
1 次 pooling
Dropout(0.5)
Flatten
Dense(512, relu)
Dense(256, relu)
Dense(128, relu)
Dense(64, relu)
Dropout(0.5)
Dense(2, softmax)
	0.9934

- 各模型表現(同模型若有用不同參數，取準確率最高的代表)：

Model	Test Accuracy
SVM	0.9825
RandomForest	0.9879
CNN2	0.9949
Vgg16	0.6878

- 使用的方法的優點或缺點：

  用 svm 或 randomforest 模型的優點是會跑的比深度學習模型快，不過深度學習模型準確率通常較高，所以用 CNN 準確率會比較高。另外 CNN 前處理要轉成圖片格式，所以會比較麻煩一點。

- 可改進之處

  用 Vgg16 的準確率太低，理論上不會那麼低，可能是因為我的 permission 取太少個當特徵，總共只有 67 個，但是 Vgg16 本來的 input shape 是

(224,224,3)。因為 Vgg16 的層數設很多，處理的圖片大小比較大，所以我把只有 67 個特徵的資料集硬是轉成 Vgg16 可以做的大小(32*32)，可能就會影響到參數已經訓練好的 Vgg16 的表現。所以可以改進的地方我覺得是把特徵數量變多，也就是 permission 在前處理挑多一點，或者是換個特徵，例如直接拿 apk 的二進位碼轉成圖片格式當作 Vgg16 的 input。

- 最後會選擇哪一個模型(可自行假設情境)

  如果要比較簡單快速的建好模型，就是選 SVM 或 RandomForest，RandomForest 的準確率又比 SVM 高一點，所以想要快速又準確的模型就選 RandomForest，如果想要更精準的分類，不在乎耗不耗時的話，可以用 CNN，準確率可以達到 0.99。