**D3**

<https://www.cupoy.com/qa/kwassist/ai_tw/0000016A2E326291000001316375706F795F72656C656173655155455354>

順著 Highwind Chang 的回答，這邊補充下 df 的取值用法：

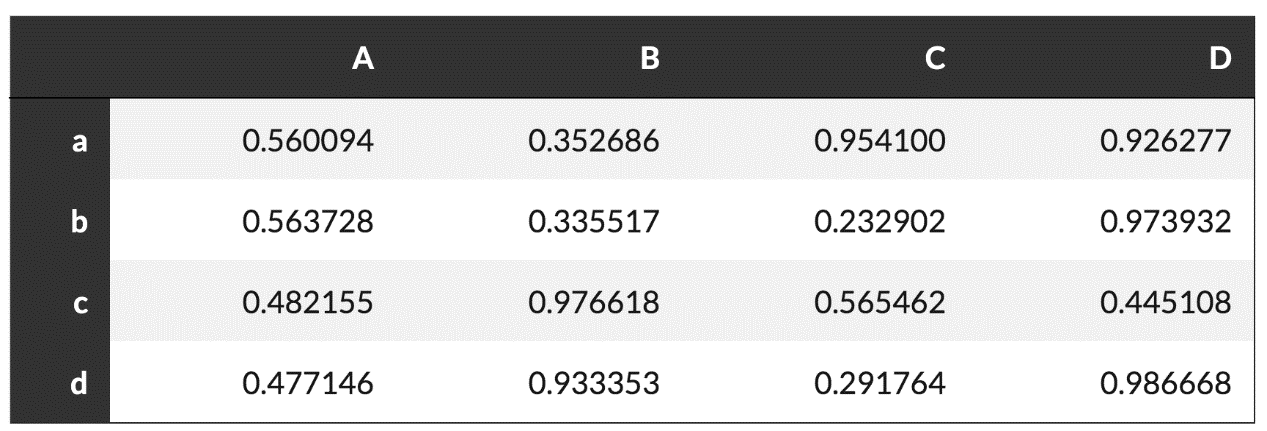
1. df['column']['row']

2. df.loc['row', 'column']、df.iloc['row index', 'column index']、df;ix[..., ...]

3. [**df.at**](http://df.at/)['row', 'column']、df.iat['row index', 'column index']

df.loc[ \_\_\_\_\_ , \_\_\_\_\_ ]、df.iloc[ \_\_\_\_\_ , \_\_\_\_\_ ] 的基本用法都是左邊放「row」右邊放「column」。loc 要用列名稱或欄位名稱，iloc 用列索引或是欄位的索引。

舉例來說，如果你有一個這樣的資料：



可以用 df.loc[ 'a' , 'A' ] 或 df.iloc[0, 0] 取到 0.560094 的值。

-----------------------------------------------------------------------------------

這邊寫了一個範例提供 Highwind Chang 參考：

```

import pandas as pd

d = {'col1': [1, 2], 'col2': [3, 4]}

df = pd.DataFrame(data=d)

print(df)

#    col1  col2

# 0     1     3

# 1     2     4

print(df['col1'][0]) # 1

print(df.loc[0, 'col1']) # 1

print(df.iloc[0, 0]) # 1

print(**[df.at](http://df.at/" \t "_blank)**[0, 'col1']) # 1

print(df.iat[0, 0]) # 1

```

Highwind Chang，這邊可以再補充一點：

df**['col']** 其實會抓出的是 df 中 col 這一個 column 的 series 型態

df['col']**['row']**等同於從 col 這個 series 中取值

但是你可以試試看 df[0:1] 會取出什麼？取出的是 第 0 個 row 的 dataframe 型態

也就是說，當對 df 取一層的 slice 或 [ ] 操作的行為，其實是很容易搞混的，有時候是 series、有時候又是 dataframe

因此才會有 loc、iloc、 這三個方法出現，這三個叫做 location，意思就是用「位置」去取資料，用的是 row、column 的座標位置去取！

那你可以想想看，有了 loc、iloc 這一類的不就夠了嗎？為什麼還要有 at、iat 的用法呢？

----------------------------------------------------------------------

剛剛試了一下, 用 df 真的容易搞混

df['col1'] 會傳回 Series

df[0:1] 的確會傳回 DataFrame

如果寫  df[0] 則會報錯, 對初學者來說有點不直覺, 因為 df[0:1] 明明可以啊?

但

df.loc[0] 傳回 Series

df.loc[0:1] 傳回 DataFrame

除了把 row 放前面, 比較符合一般使用習慣以外, 看起來跟 df[]  的行為很像啊?

對於 at, 我的想法是因為 [**df.at**](http://df.at/)[row, column] 需要兩個參數.

這可以確保傳回的都是一致的 type, 因為少參數會報錯, 比較不會像 df[] df[][] 這樣使用上容易混淆

**D6**

app\_train[numeric\_columns].columns[app\_train[numeric\_columns].apply(lambda x:len(x.unique())!=2 )]

前面藍色的部分您一定瞭解，就是列出所有篩選過(numeric\_columns)的columns

後面紅色的部分您要先瞭解三個部分

1. 瞭解 Dataframe裡的apply函數:

    apply函數在Dataframe裡非常好用，就是指 把所有的Series都套用這個公式上

    ex: app\_train[numeric\_columns].apply(lambda x:len(x.unique())!=2 )

           把橘色的公式套用在app\_train[numeric\_columns]裡

2. 瞭解 lambda函數:

    lambda其實就是簡易版的def fun(), 他適合用在一次性或是簡短的功能上

    lambda x:len(x.unique())!=2

    先定義一個變數x ，x 會等於len(x.unique()) != 2 ,也就是x唯一值的數量不等於二

    您可以試看看這兩段程式碼，這樣會更清楚我在講什麼

    app\_train['NAME\_CONTRACT\_TYPE'].value\_counts()      
   app\_train['NAME\_CONTRACT\_TYPE'].unique()

3. 瞭解unique()函數:

    unique英文意思就是唯一的，他就是用來刪除掉重複地值，只顯示不重複的值

    有點像是Set(集合) 一樣

dtypes.isin 當中要放的是是 dtype 的型態，而不是字串，所以必須這麼寫會比較好：

```

app\_train.dtypes.isin([np.dtype('int64'), np.dtype('float64')])

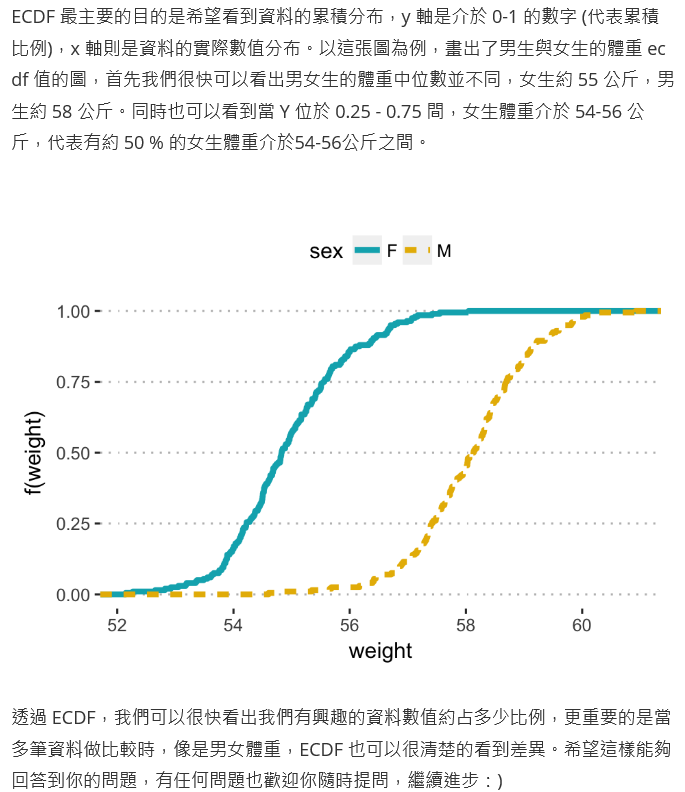
```

也可以改用對 df 直接做 select\_dtypes：

```

app\_train.select\_dtypes(include=['float64', 'int64'])

```



**作業抄tim的, 有時間要重做.**

**D7**

quantile\_all = [np.percentile(app\_train[~app\_train['AMT\_ANNUITY'].isnull()]['AMT\_ANNUITY'], q = i) for i in q\_all]

1."~"代表不是的意思，也就是邏輯裡的NOT。

所以app\_train["AMT\_ANNUITY"].isnull()是指說"AMT\_ANNUITY"這個column裡面空的值，而前面多一個"~"就代表不是空的值，也就是我們說的一般值啦!!

2.建議您可以練習拆解看看

step1. app\_train[~app\_train['AMT\_ANNUITY'].isnull()]

step2. app\_train[~app\_train['AMT\_ANNUITY'].isnull()]['AMT\_ANNUITY']

step3. 了解 np.percentile 裡的參數代表什麼意思

step4. 了解 List comprehension

ex: [ 2 \* i for i in range(10) ]

最後就會解出您第二個問題啦!!

這種複雜的寫法，建議從內層慢慢往外看，在看的同時可以搭配印出確認。

這邊補充一下中間這一段： app\_train[~app\_train['AMT\_ANNUITY'].isnull()]，你會發現 ~app\_train['AMT\_ANNUITY'] 會回傳一個 Boolean Series 如下：

```

0 False

1 False

2 True

...

Name: numeric, dtype: bool

```

當我們把上述結果放在 DF 的 index 中，稱為 Boolean filtering/indexing，會將等於 True 的資料留下來。畫句話就是用 True/False 來決定每一筆資料要不要篩選。

**D8**

當我們把上述結果放在 DF 的 index 中，稱為 Boolean filtering/indexing，會將等於 True 的資料留下來。畫句話就是用 True/False 來決定每一筆資料要不要篩選。

可以這樣:

sub\_df = app\_train.**loc**[app\_train['AMT\_INCOME\_TOTAL'] > app\_train['AMT\_INCOME\_TOTAL'].mean(), ['SK\_ID\_CURR', 'TARGET']]

但不能這樣

sub\_df = app\_train[app\_train['AMT\_INCOME\_TOTAL'] > app\_train['AMT\_INCOME\_TOTAL'].mean(), ['SK\_ID\_CURR', 'TARGET']]

但這樣可以

sub\_df = app\_train[app\_train['AMT\_INCOME\_TOTAL'] > app\_train['AMT\_INCOME\_TOTAL'].mean()]