勞動部產業新尖兵計畫

人工智慧金融應用與實務培訓班

課程模組： AI 金融科技課程 - AI 程式設計

3.MLP 實務

葉建華 (Yeh, Jian-hua)

tdi.jhyeh@tdi.edu.tw
au4290@gmail.com

講次內容

- MLP 案例 1： 汽車評估資料預測
- MLP 案例 2： 手寫辨識 MNIST

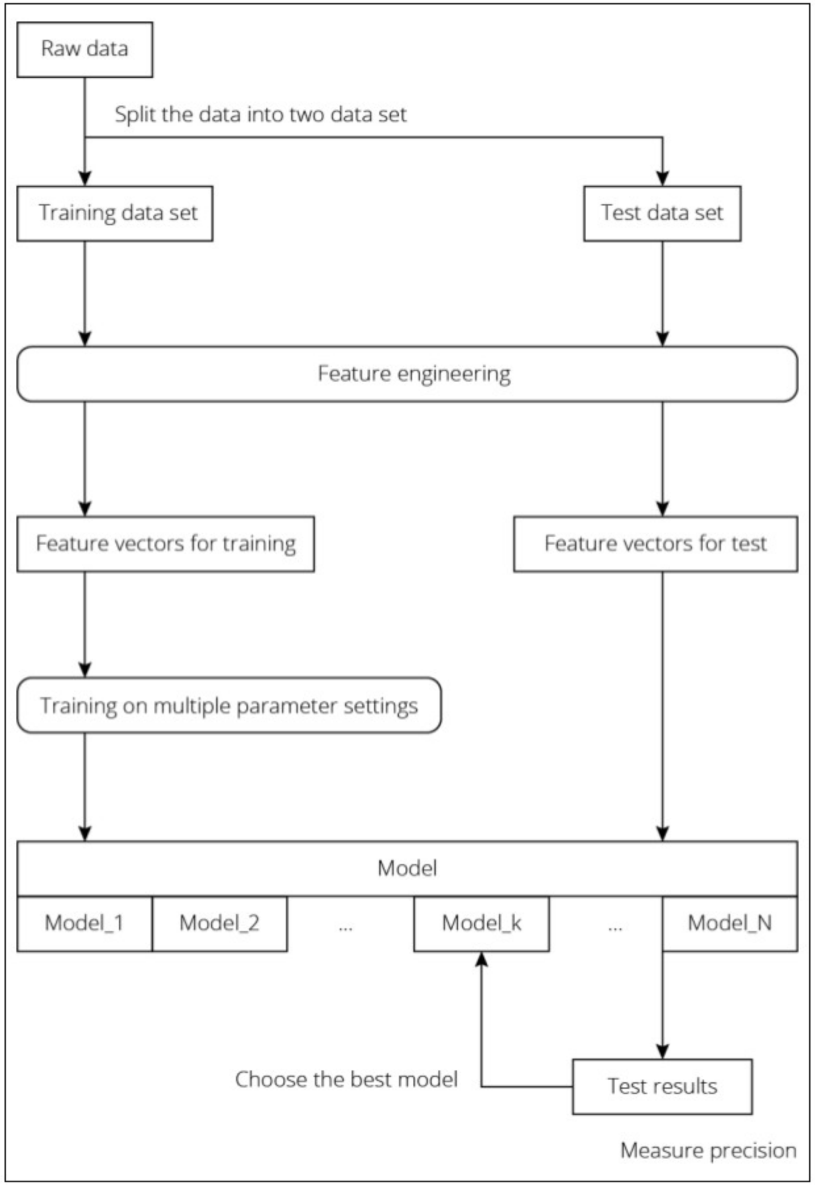
基本的邏輯流程

- 監督式的機器學習的解題方式基本流程都雷同!
  - 讀入資料檔
  - 分成 train 和 test 兩部分
  - 資料前處理
  - 根據資料維度，建立模型
  - 訓練 train 資料集
  - 用 test 資料集做評估

基本的邏

- 監督式的機器學習的解題方式基本
  - 讀入資料檔
  - 分成 train 和 test 兩部分
  - 資料前處理
  - 根據資料維度，建立模型
  - 訓練 train 資料集
  - 用 test 資料集做評估

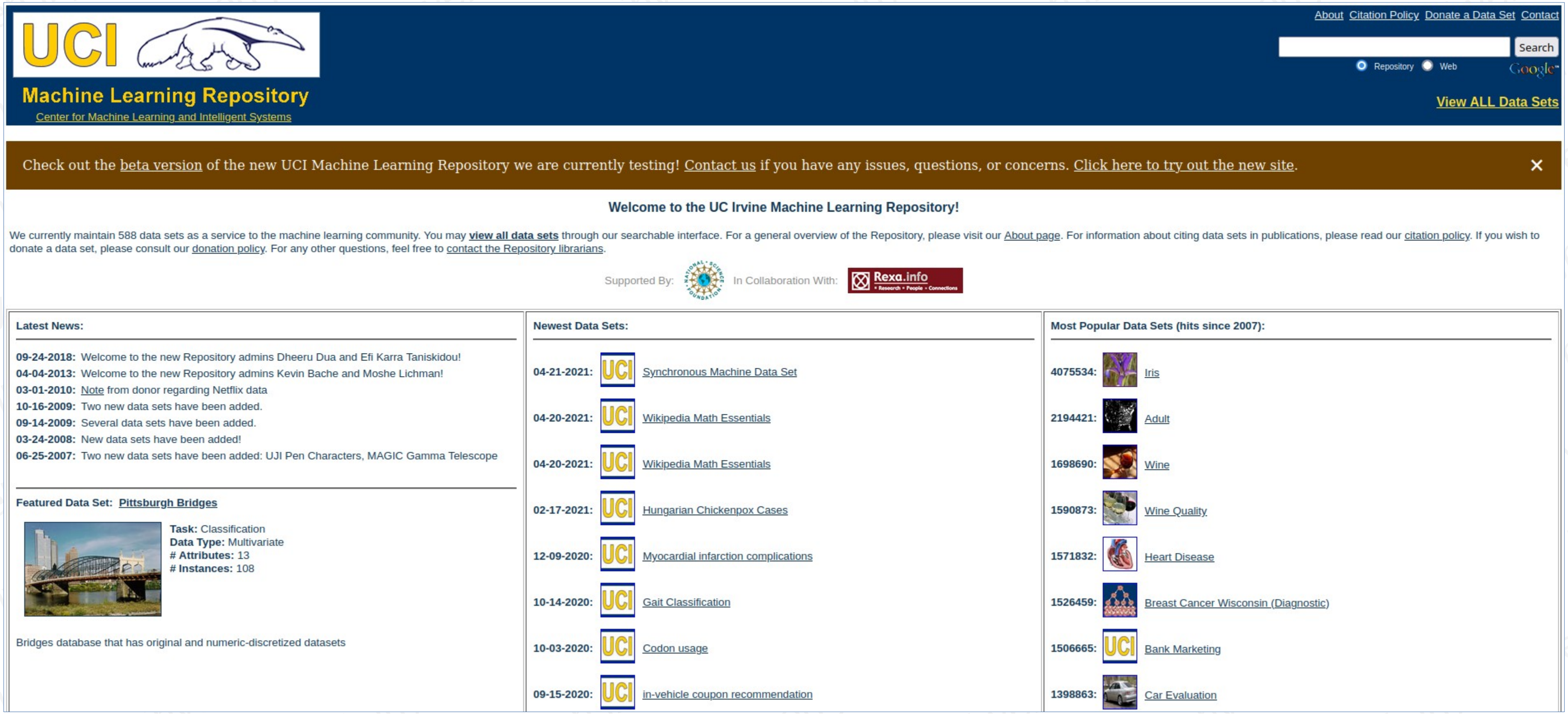
還記得這個嗎?



MLP 案例 1：汽車評估資料預測

- 買車要看好多條件！
  - 售價、維護費用
  - 車門數、載人數
  - 後車廂大小
  - 安全性
  - …

Car Evaluation 資料集



UCI Machine Learning Repository
Center for Machine Learning and Intelligent Systems

Check out the beta version of the new UCI Machine Learning Repository we are currently testing! Contact us if you have any issues, questions, or concerns. Click here to try out the new site. ✕

Welcome to the UC Irvine Machine Learning Repository!

We currently maintain 588 data sets as a service to the machine learning community. You may **view all data sets** through our searchable interface. For a general overview of the Repository, please visit our About page. For information about citing data sets in publications, please read our citation policy. If you wish to donate a data set, please consult our donation policy. For any other questions, feel free to contact the Repository librarians.

Supported By: In Collaboration With: Rexa.info • Research • People • Connections

Latest News:

09-24-2018: Welcome to the new Repository admins Dheeru Dua and Efi Karra Taniskidou!
04-04-2013: Welcome to the new Repository admins Kevin Bache and Moshe Lichman!
03-01-2010: Note from donor regarding Netflix data
10-16-2009: Two new data sets have been added.
09-14-2009: Several data sets have been added.
03-24-2008: New data sets have been added!
06-25-2007: Two new data sets have been added: UJI Pen Characters, MAGIC Gamma Telescope

Featured Data Set: **Pittsburgh Bridges**

Task: Classification
Data Type: Multivariate
# Attributes: 13
# Instances: 108

Bridges database that has original and numeric-discretized datasets

Newest Data Sets:

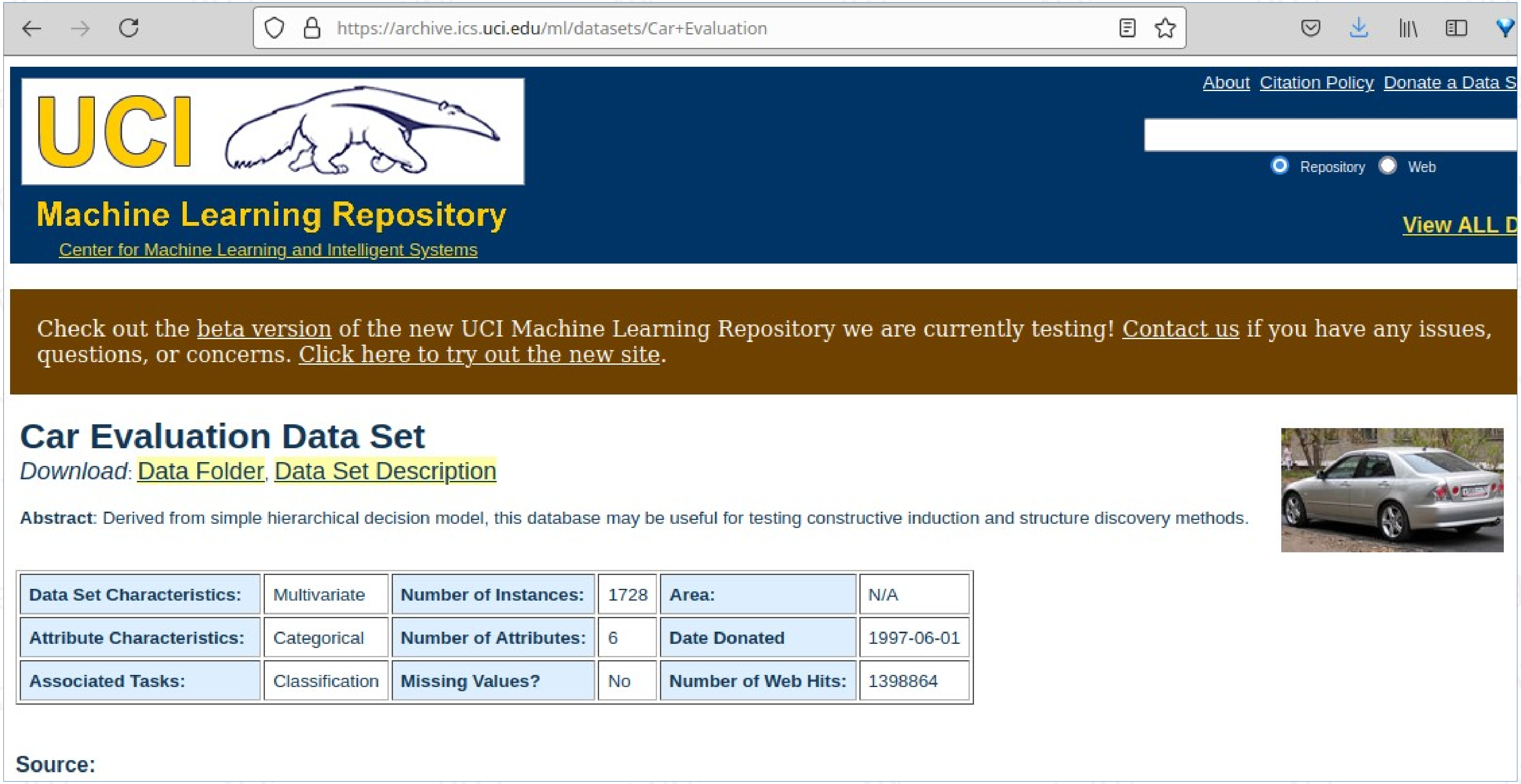
04-21-2021:	Synchronous Machine Data Set
04-20-2021:	Wikipedia Math Essentials
04-20-2021:	Wikipedia Math Essentials
02-17-2021:	Hungarian Chickenpox Cases
12-09-2020:	Myocardial infarction complications
10-14-2020:	Gait Classification
10-03-2020:	Codon usage
09-15-2020:	in-vehicle coupon recommendation

Most Popular Data Sets (hits since 2007):

4075534:	Iris
2194421:	Adult
1698690:	Wine
1590873:	Wine Quality
1571832:	Heart Disease
1526459:	Breast Cancer Wisconsin (Diagnostic)
1506665:	Bank Marketing
1398863:	Car Evaluation

第 8 名， 139 萬次下載

下載 Car Evaluation 資料集



https://archive.ics.uci.edu/ml/datasets/Car+Evaluation
UCI
Machine Learning Repository
Center for Machine Learning and Intelligent Systems
About Citation Policy Donate a Data S
● Repository ○ Web
View ALL D

Check out the beta version of the new UCI Machine Learning Repository we are currently testing! Contact us if you have any issues, questions, or concerns. Click here to try out the new site.

Car Evaluation Data Set

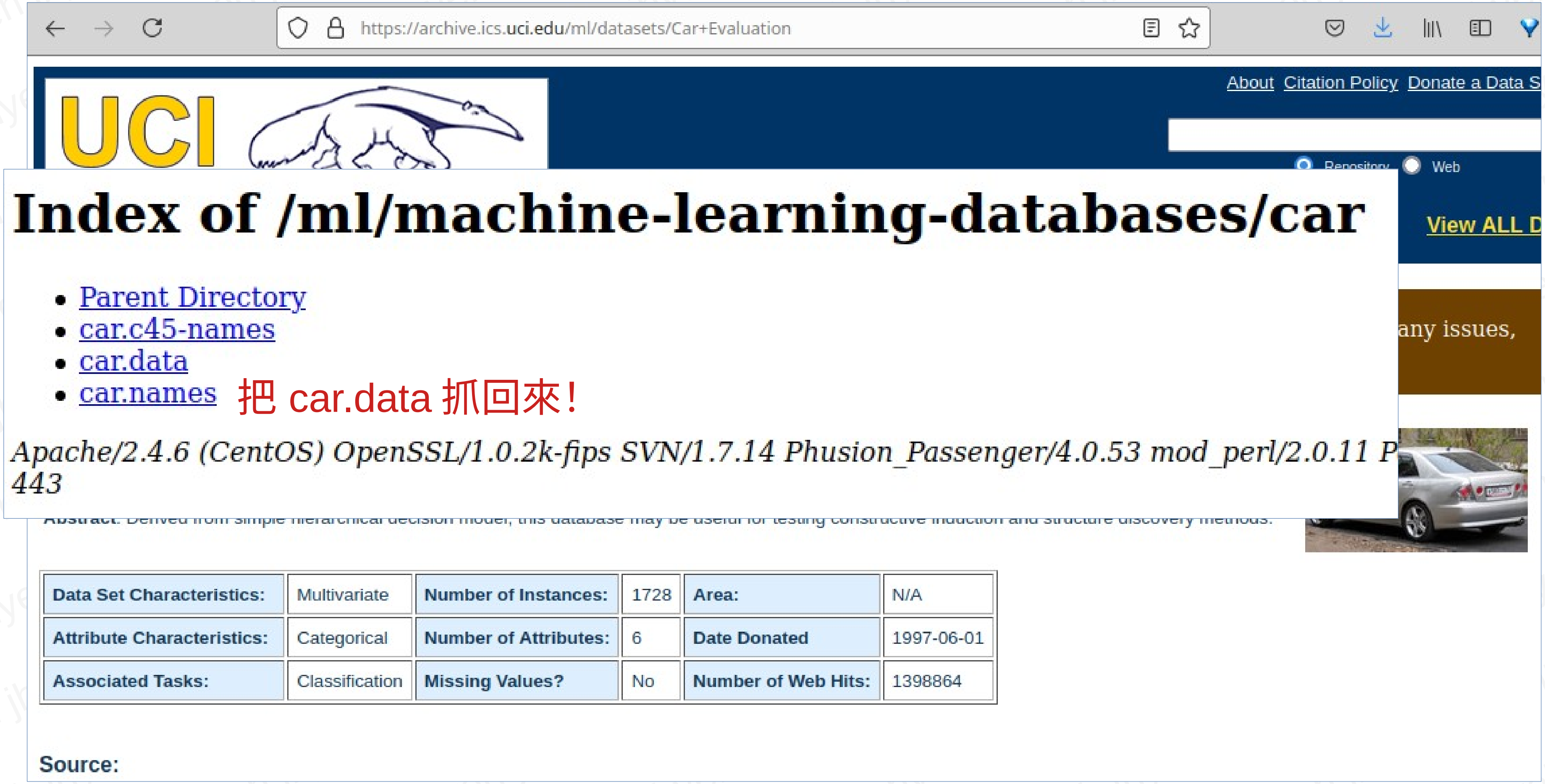
Download: Data Folder, Data Set Description

Abstract: Derived from simple hierarchical decision model, this database may be useful for testing constructive induction and structure discovery methods.

Data Set Characteristics:	Multivariate	**Number of Instances:**	1728	**Area:**	N/A
Attribute Characteristics:	Categorical	**Number of Attributes:**	6	**Date Donated**	1997-06-01
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	1398864

Source:

下載 Car Evaluation 資料集



https://archive.ics.uci.edu/ml/datasets/Car+Evaluation
About Citation Policy Donate a Data S
Repository Web
View ALL D
any issues,

Index of /ml/machine-learning-databases/car

- Parent Directory
- car.c45-names
- car.data
- car.names

把 car.data 抓回來！

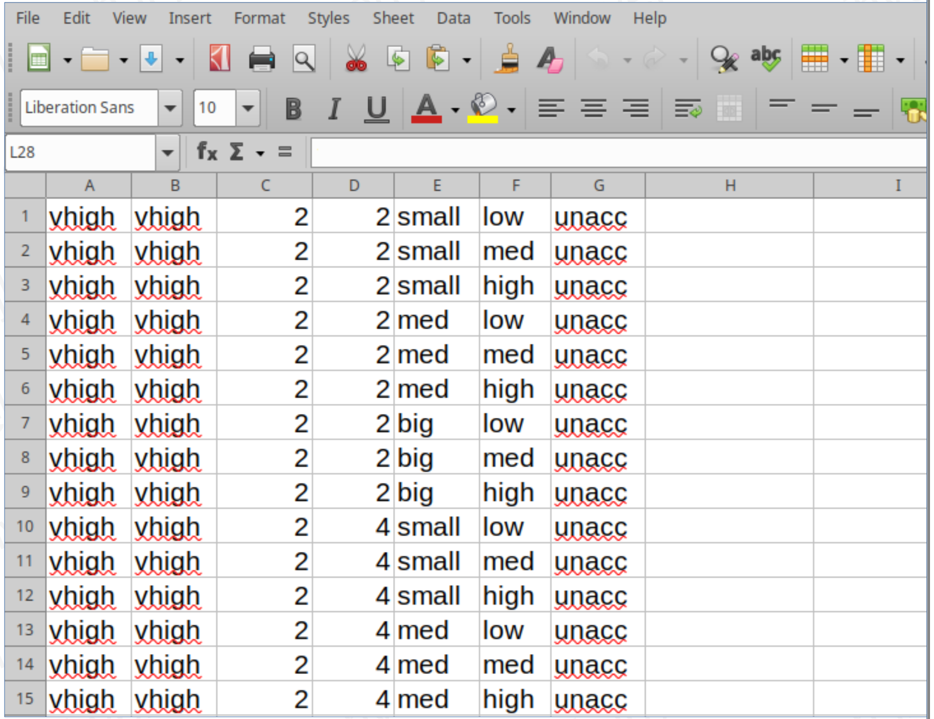
Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips SVN/1.7.14 Phusion_Passenger/4.0.53 mod_perl/2.0.11 P
443

Abstract: Derived from simple hierarchical decision model, this database may be useful for testing constructive induction and structure discovery methods.

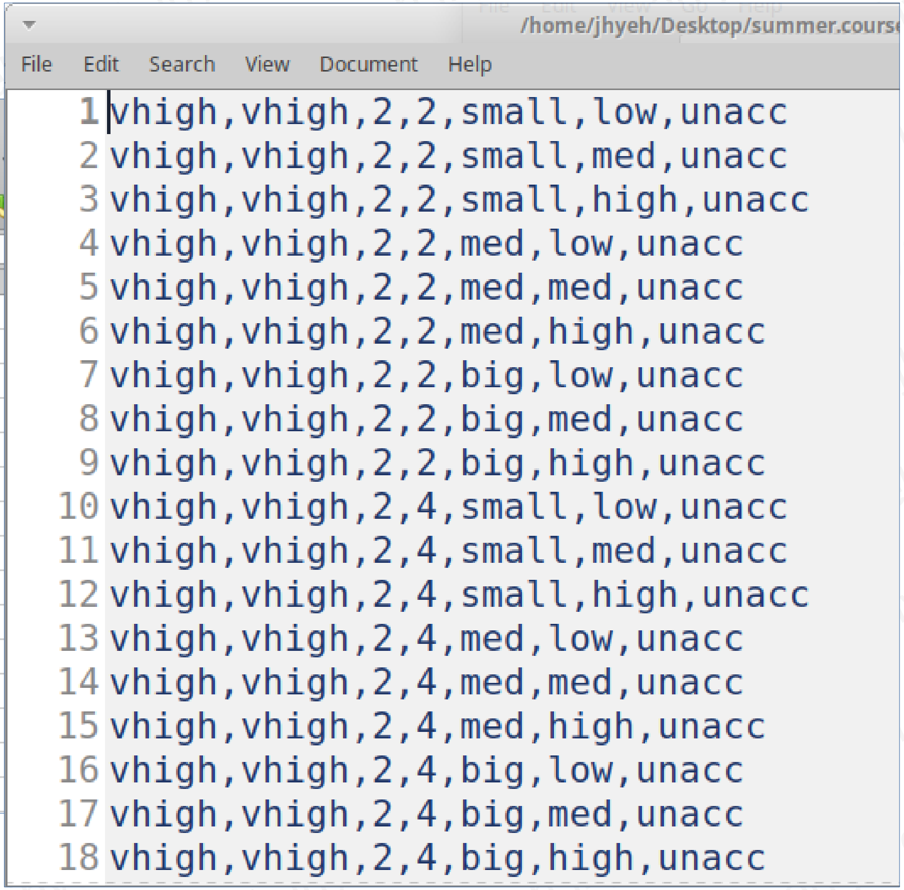
Data Set Characteristics:	Multivariate	**Number of Instances:**	1728	**Area:**	N/A
Attribute Characteristics:	Categorical	**Number of Attributes:**	6	**Date Donated**	1997-06-01
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	1398864

Source:

Car Evaluation 資料集

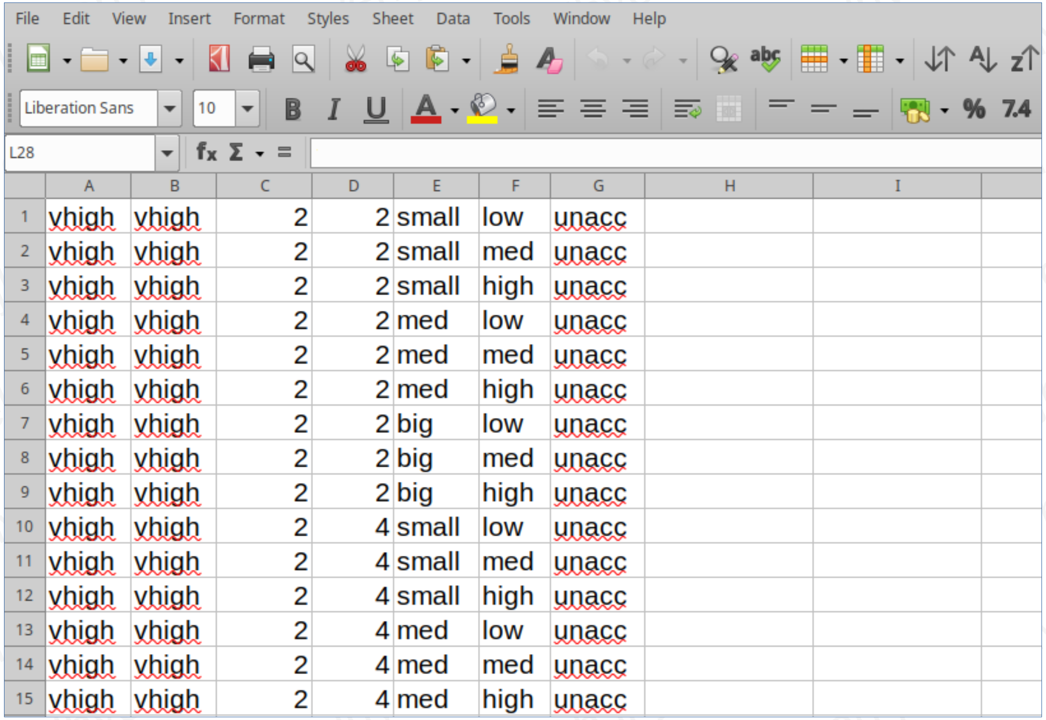


	A	B	C	D	E	F	G	H	I
1	vhigh	vhigh	2	2	small	low	unacc		
2	vhigh	vhigh	2	2	small	med	unacc		
3	vhigh	vhigh	2	2	small	high	unacc		
4	vhigh	vhigh	2	2	med	low	unacc		
5	vhigh	vhigh	2	2	med	med	unacc		
6	vhigh	vhigh	2	2	med	high	unacc		
7	vhigh	vhigh	2	2	big	low	unacc		
8	vhigh	vhigh	2	2	big	med	unacc		
9	vhigh	vhigh	2	2	big	high	unacc		
10	vhigh	vhigh	2	4	small	low	unacc		
11	vhigh	vhigh	2	4	small	med	unacc		
12	vhigh	vhigh	2	4	small	high	unacc		
13	vhigh	vhigh	2	4	med	low	unacc		
14	vhigh	vhigh	2	4	med	med	unacc		
15	vhigh	vhigh	2	4	med	high	unacc		



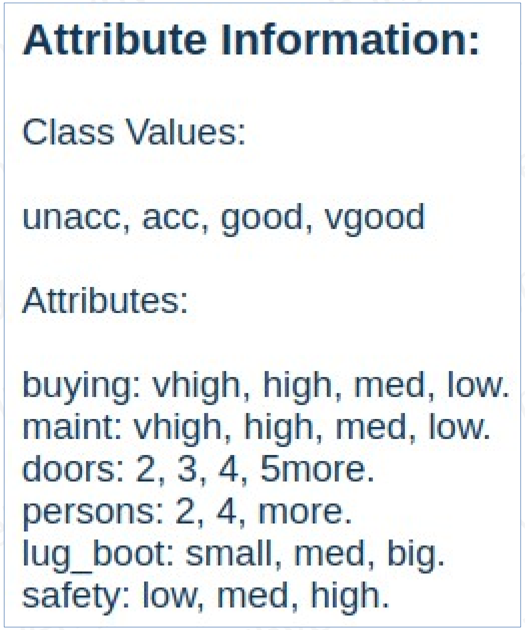
```
vhigh,vhigh,2,2,small,low,unacc
vhigh,vhigh,2,2,small,med,unacc
vhigh,vhigh,2,2,small,high,unacc
vhigh,vhigh,2,2,med,low,unacc
vhigh,vhigh,2,2,med,med,unacc
vhigh,vhigh,2,2,med,high,unacc
vhigh,vhigh,2,2,big,low,unacc
vhigh,vhigh,2,2,big,med,unacc
vhigh,vhigh,2,2,big,high,unacc
vhigh,vhigh,2,4,small,low,unacc
vhigh,vhigh,2,4,small,med,unacc
vhigh,vhigh,2,4,small,high,unacc
vhigh,vhigh,2,4,med,low,unacc
vhigh,vhigh,2,4,med,med,unacc
vhigh,vhigh,2,4,med,high,unacc
vhigh,vhigh,2,4,big,low,unacc
vhigh,vhigh,2,4,big,med,unacc
vhigh,vhigh,2,4,big,high,unacc
```

Car Evaluation 資料集


File Edit View Insert Format Styles Sheet Data Tools Window Help
Liberation Sans
10
L28

	A	B	C	D	E	F	G	H	I
1	vhigh	vhigh	2	2	small	low	unacc		
2	vhigh	vhigh	2	2	small	med	unacc		
3	vhigh	vhigh	2	2	small	high	unacc		
4	vhigh	vhigh	2	2	med	low	unacc		
5	vhigh	vhigh	2	2	med	med	unacc		
6	vhigh	vhigh	2	2	med	high	unacc		
7	vhigh	vhigh	2	2	big	low	unacc		
8	vhigh	vhigh	2	2	big	med	unacc		
9	vhigh	vhigh	2	2	big	high	unacc		
10	vhigh	vhigh	2	4	small	low	unacc		
11	vhigh	vhigh	2	4	small	med	unacc		
12	vhigh	vhigh	2	4	small	high	unacc		
13	vhigh	vhigh	2	4	med	low	unacc		
14	vhigh	vhigh	2	4	med	med	unacc		
15	vhigh	vhigh	2	4	med	high	unacc		

請留意網頁上還有這個訊息 ...



Attribute Information:

Class Values:

unacc, acc, good, vgood

Attributes:

buying: vhigh, high, med, low.
maint: vhigh, high, med, low.
doors: 2, 3, 4, 5more.
persons: 2, 4, more.
lug_boot: small, med, big.
safety: low, med, high.

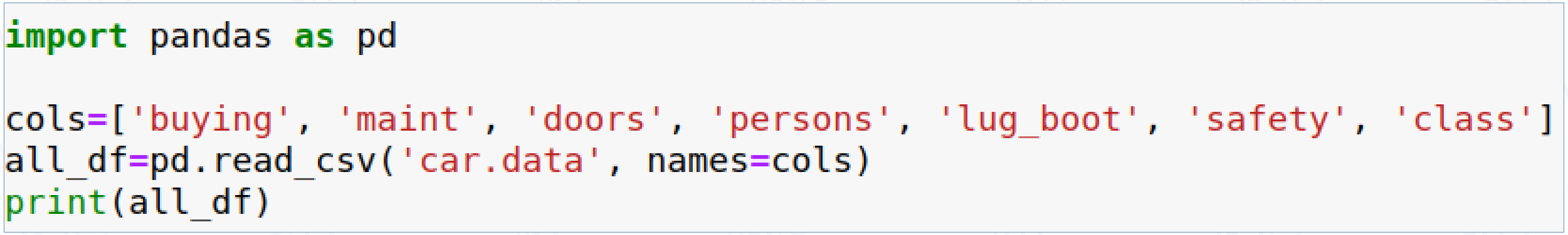
Car Evaluation 使用 MLP 進行學習

- 邏輯流程
  - 讀入資料檔 car.data (Pandas)
  - 分成 train 和 test 兩部分 (List processing)
  - 資料前處理 (有點 tricky...)
  - 根據資料維度，建立 Keras MLP 模型 (Keras)
  - 訓練 train 資料集 (Keras)
  - 用 test 資料集做評估 (Keras)

被你發現了！

Car Evaluation 使用 MLP 進行學習

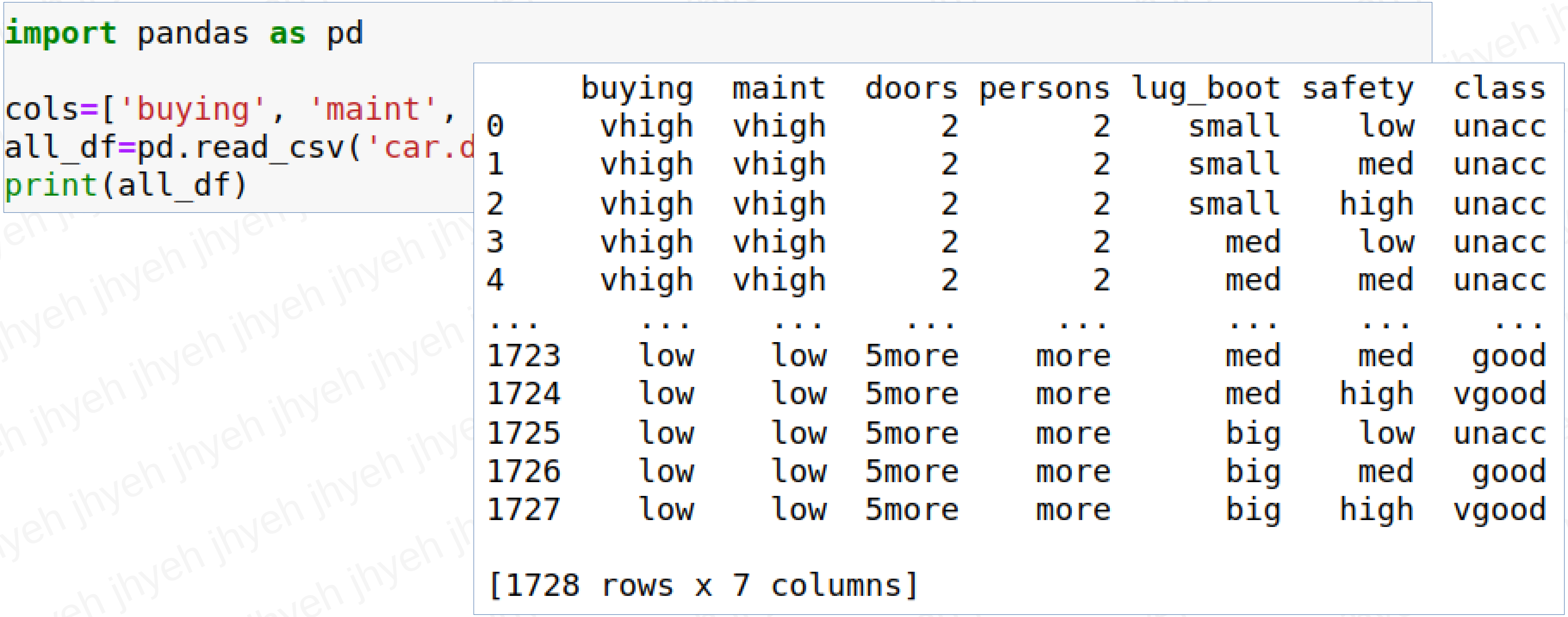
- 讀入資料檔 car.data (Pandas)



```
import pandas as pd

cols=['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']
all_df=pd.read_csv('car.data', names=cols)
print(all_df)
```

Car Evaluation 使用 MLP 進行學習

- 讀入資料檔 car.data (Pandas)

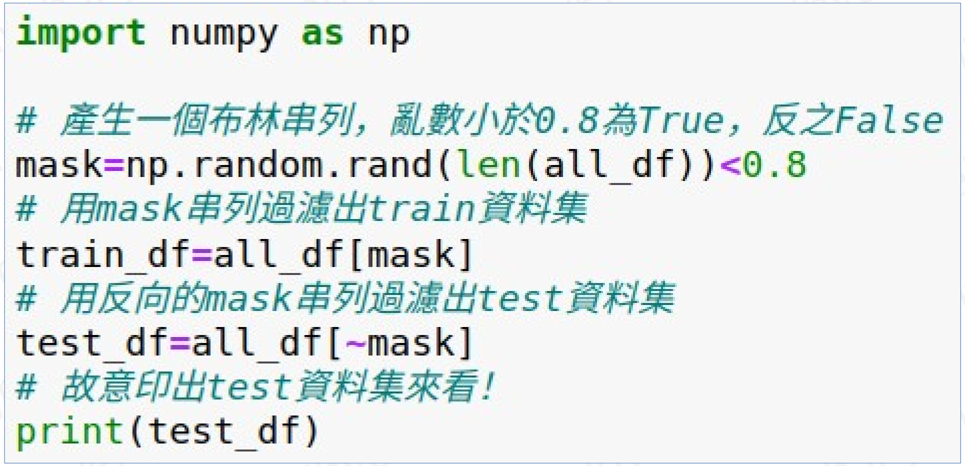


```
import pandas as pd

cols=['buying', 'maint', 
all_df=pd.read_csv('car.d
print(all_df)
```

```
      buying  maint  doors persons lug_boot safety  class
0      vhigh  vhigh      2       2    small    low  unacc
1      vhigh  vhigh      2       2    small    med  unacc
2      vhigh  vhigh      2       2    small   high  unacc
3      vhigh  vhigh      2       2      med    low  unacc
4      vhigh  vhigh      2       2      med    med  unacc
...      ...    ...    ...     ...      ...    ...    ...
1723     low    low  5more    more      med    med   good
1724     low    low  5more    more      med   high  vgood
1725     low    low  5more    more      big    low  unacc
1726     low    low  5more    more      big    med   good
1727     low    low  5more    more      big   high  vgood

[1728 rows x 7 columns]
```

Car Evaluation 使用 MLP 進行學習

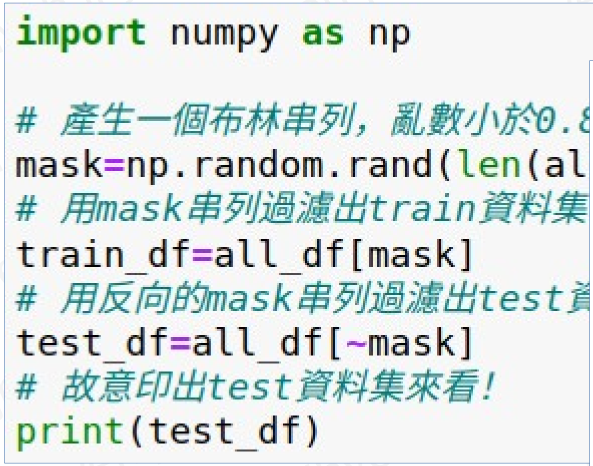
- 分成 train 和 test 兩部分 (List processing)



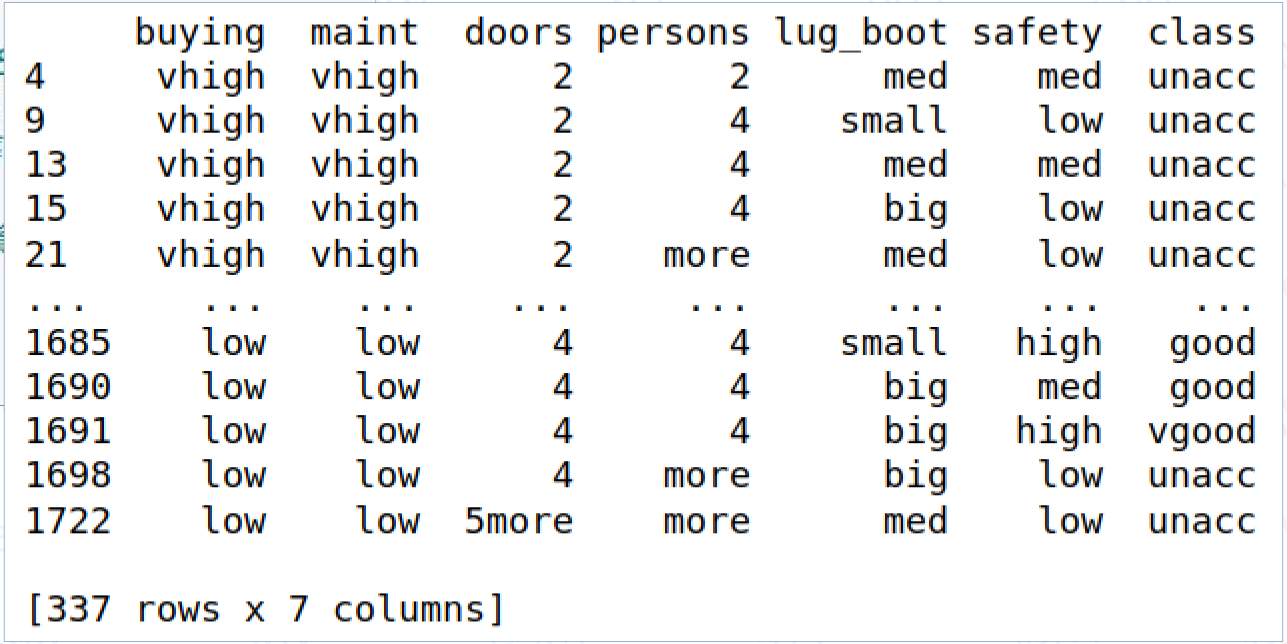
```
import numpy as np

# 產生一個布林串列，亂數小於0.8為True，反之False
mask=np.random.rand(len(all_df))<0.8
# 用mask串列過濾出train資料集
train_df=all_df[mask]
# 用反向的mask串列過濾出test資料集
test_df=all_df[~mask]
# 故意印出test資料集來看!
print(test_df)
```

Car Evaluation 使用 MLP 進行學習

- 分成 train 和 test 兩部分 (List processing)



```
import numpy as np

# 產生一個布林串列，亂數小於0.
mask=np.random.rand(len(al
# 用mask串列過濾出train資料集
train_df=all_df[mask]
# 用反向的mask串列過濾出test資
test_df=all_df[~mask]
# 故意印出test資料集來看!
print(test_df)
```



```
     buying  maint  doors persons lug_boot safety  class
4     vhigh  vhigh      2       2      med    med  unacc
9     vhigh  vhigh      2       4    small    low  unacc
13    vhigh  vhigh      2       4      med    med  unacc
15    vhigh  vhigh      2       4      big    low  unacc
21    vhigh  vhigh      2    more      med    low  unacc
...     ...    ...    ...     ...      ...    ...    ...
1685    low    low      4       4    small   high   good
1690    low    low      4       4      big    med   good
1691    low    low      4       4      big   high  vgood
1698    low    low      4    more      big    low  unacc
1722    low    low  5more    more      med    low  unacc

[337 rows x 7 columns]
```

隨機的，你不一定看到這樣的結果

Car Evaluation 使用 MLP 進行學習

- 資料前處理 （有點 tricky...)
  - 使用 scikit-learn 套件，Anaconda 已裝了
  - 將分類標籤 unacc, acc, good, vgood 轉成數字標籤（神經網路輸出要用）
    - 用 dict {'unacc':0, 'acc':1, 'good':2, 'vgood':3} 代入 DataFrame 的 map() 函數
    - 你將會發現所有的屬性都要對應!
  - 使用 scikit-learn 的前處理套件 preprocessing 中的 MinMaxScaler 物件
    - 做 DataFrame 內容值的縮放，對應到 (0, 1) 之間

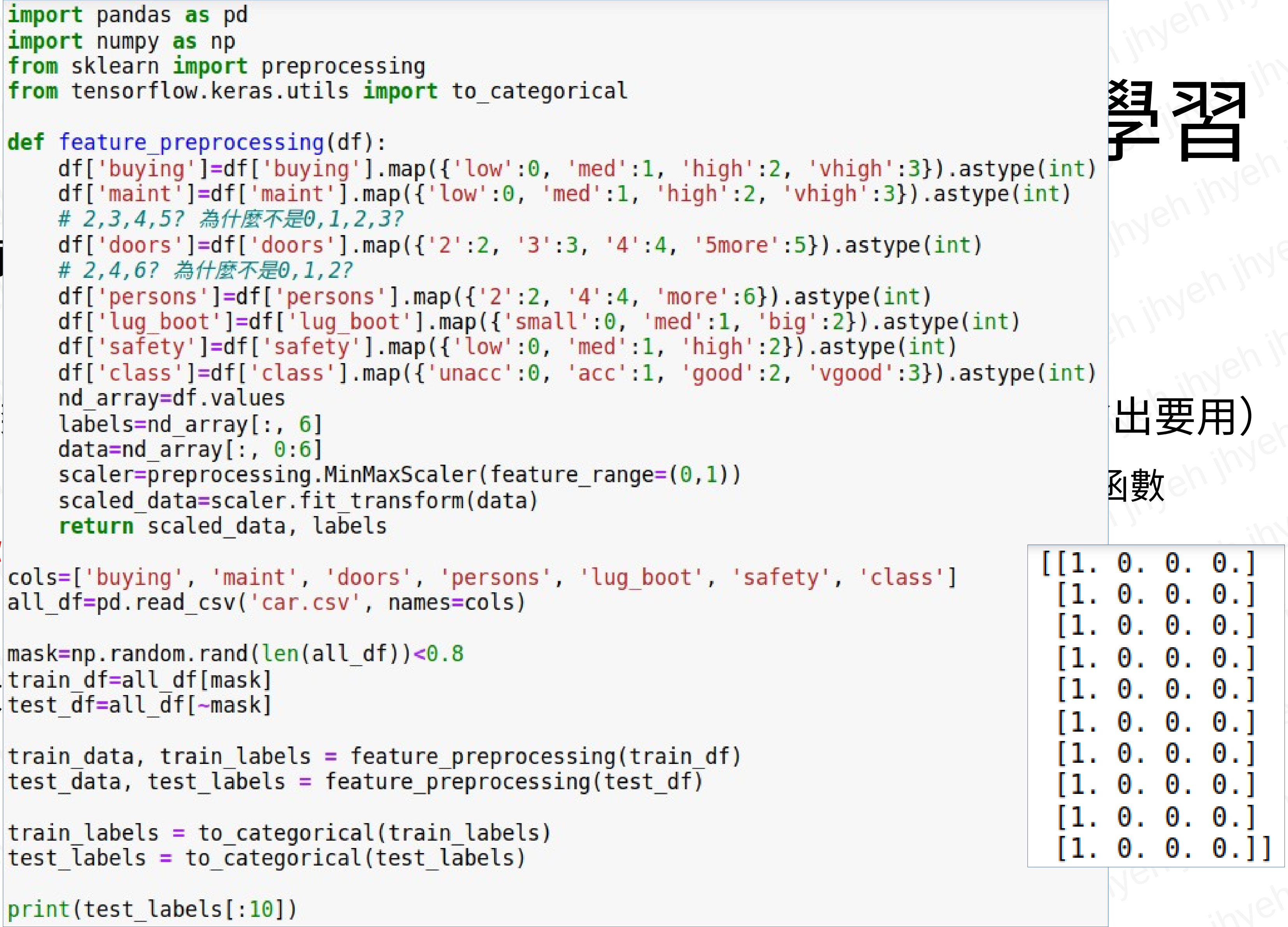
Car Evaluation 使用 MLP 進行學習

- 屬性對應，資料網頁上這麼說 ...
  - buying: vhigh, high, med, low
  - maint: vhigh, high, med, low
  - doors: 2, 3, 4, 5more
  - persons: 2, 4, more
  - lug_boot: small, med, big
  - safety: low, med, high

這兩個！千萬不要以為不用對應！

Car 學習

- 資料前
  - 使用
  - 將分 出要用)
    - 用 函數
    - 你
  - 使用
    - 做

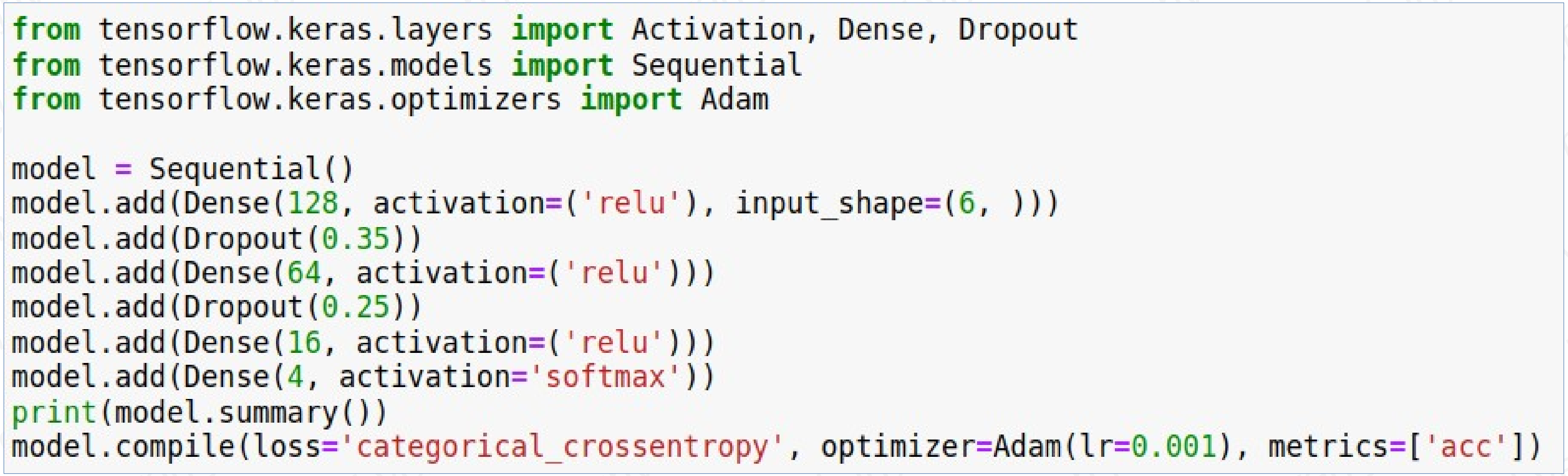


```
import pandas as pd
import numpy as np
from sklearn import preprocessing
from tensorflow.keras.utils import to_categorical

def feature_preprocessing(df):
    df['buying']=df['buying'].map({'low':0, 'med':1, 'high':2, 'vhigh':3}).astype(int)
    df['maint']=df['maint'].map({'low':0, 'med':1, 'high':2, 'vhigh':3}).astype(int)
    # 2,3,4,5? 為什麼不是0,1,2,3?
    df['doors']=df['doors'].map({'2':2, '3':3, '4':4, '5more':5}).astype(int)
    # 2,4,6? 為什麼不是0,1,2?
    df['persons']=df['persons'].map({'2':2, '4':4, 'more':6}).astype(int)
    df['lug_boot']=df['lug_boot'].map({'small':0, 'med':1, 'big':2}).astype(int)
    df['safety']=df['safety'].map({'low':0, 'med':1, 'high':2}).astype(int)
    df['class']=df['class'].map({'unacc':0, 'acc':1, 'good':2, 'vgood':3}).astype(int)
    nd_array=df.values
    labels=nd_array[:, 6]
    data=nd_array[:, 0:6]
    scaler=preprocessing.MinMaxScaler(feature_range=(0,1))
    scaled_data=scaler.fit_transform(data)
    return scaled_data, labels

cols=['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']
all_df=pd.read_csv('car.csv', names=cols)

mask=np.random.rand(len(all_df))<0.8
train_df=all_df[mask]
test_df=all_df[~mask]

train_data, train_labels = feature_preprocessing(train_df)
test_data, test_labels = feature_preprocessing(test_df)

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)

print(test_labels[:10])
```

```
[[1. 0. 0. 0.]
 [1. 0. 0. 0.]
 [1. 0. 0. 0.]
 [1. 0. 0. 0.]
 [1. 0. 0. 0.]
 [1. 0. 0. 0.]
 [1. 0. 0. 0.]
 [1. 0. 0. 0.]
 [1. 0. 0. 0.]
 [1. 0. 0. 0.]]
```

Car Evaluation 使用 MLP 進行學習

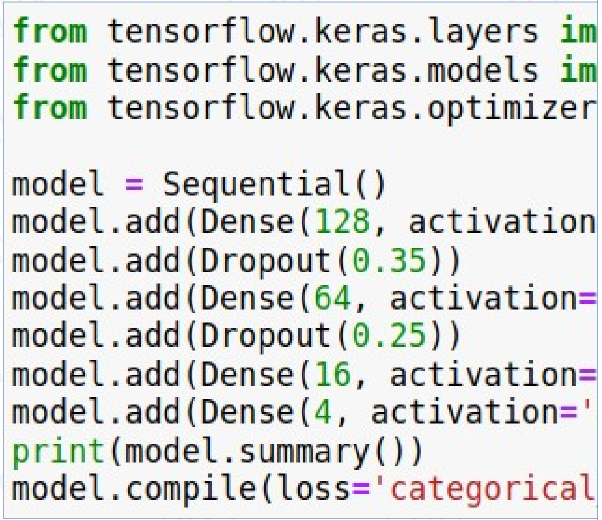
- 根據資料維度，建立 Keras MLP 模型（Keras)
  - 留意 Adam() 和 activation...



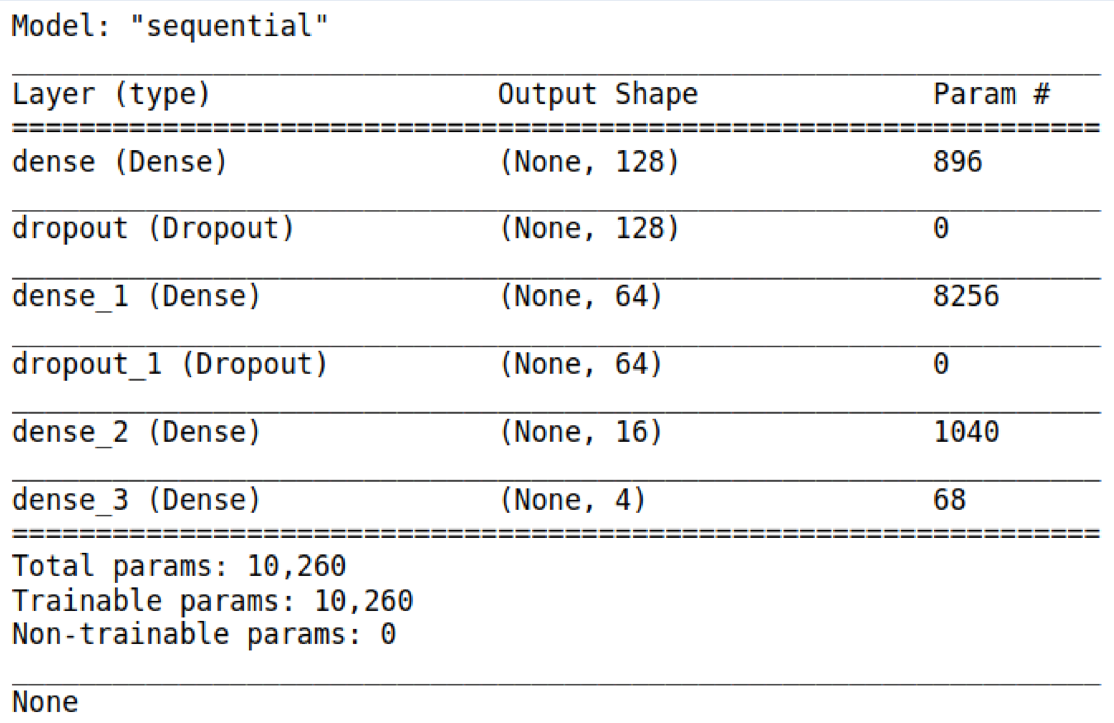
```
from tensorflow.keras.layers import Activation, Dense, Dropout
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam

model = Sequential()
model.add(Dense(128, activation=('relu'), input_shape=(6, )))
model.add(Dropout(0.35))
model.add(Dense(64, activation=('relu')))
model.add(Dropout(0.25))
model.add(Dense(16, activation=('relu')))
model.add(Dense(4, activation='softmax'))
print(model.summary())
model.compile(loss='categorical_crossentropy', optimizer=Adam(lr=0.001), metrics=['acc'])
```

Car Evaluation 使用 MLP 進行學習

- 根據資料維度，建立 Keras MLP 模型 (Keras)
  - 留意 Adam() 和



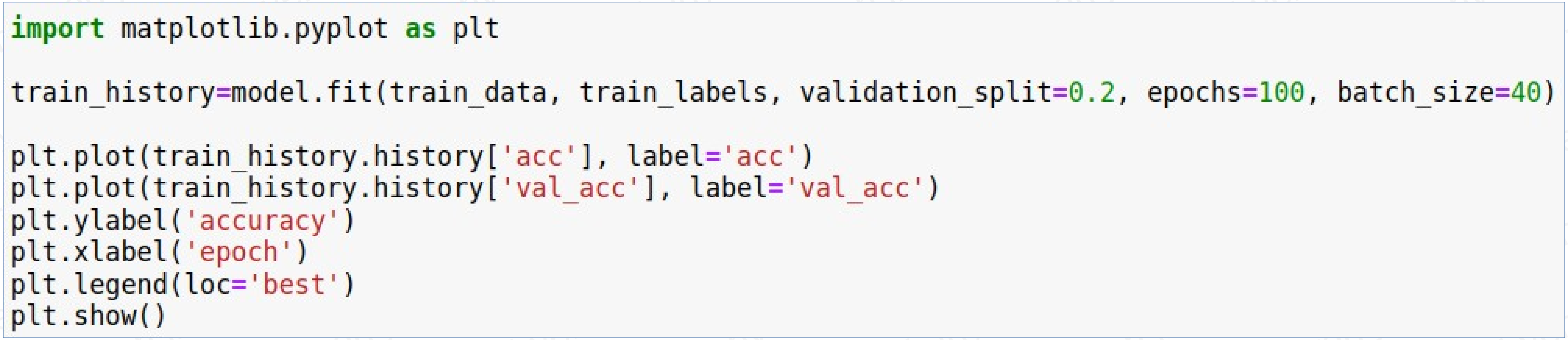
```
from tensorflow.keras.layers im
from tensorflow.keras.models im
from tensorflow.keras.optimizer

model = Sequential()
model.add(Dense(128, activation
model.add(Dropout(0.35))
model.add(Dense(64, activation=
model.add(Dropout(0.25))
model.add(Dense(16, activation=
model.add(Dense(4, activation='
print(model.summary())
model.compile(loss='categorical
```



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 128)               896
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 16)                1040
_________________________________________________________________
dense_3 (Dense)              (None, 4)                 68
=================================================================
Total params: 10,260
Trainable params: 10,260
Non-trainable params: 0
_________________________________________________________________
None
```

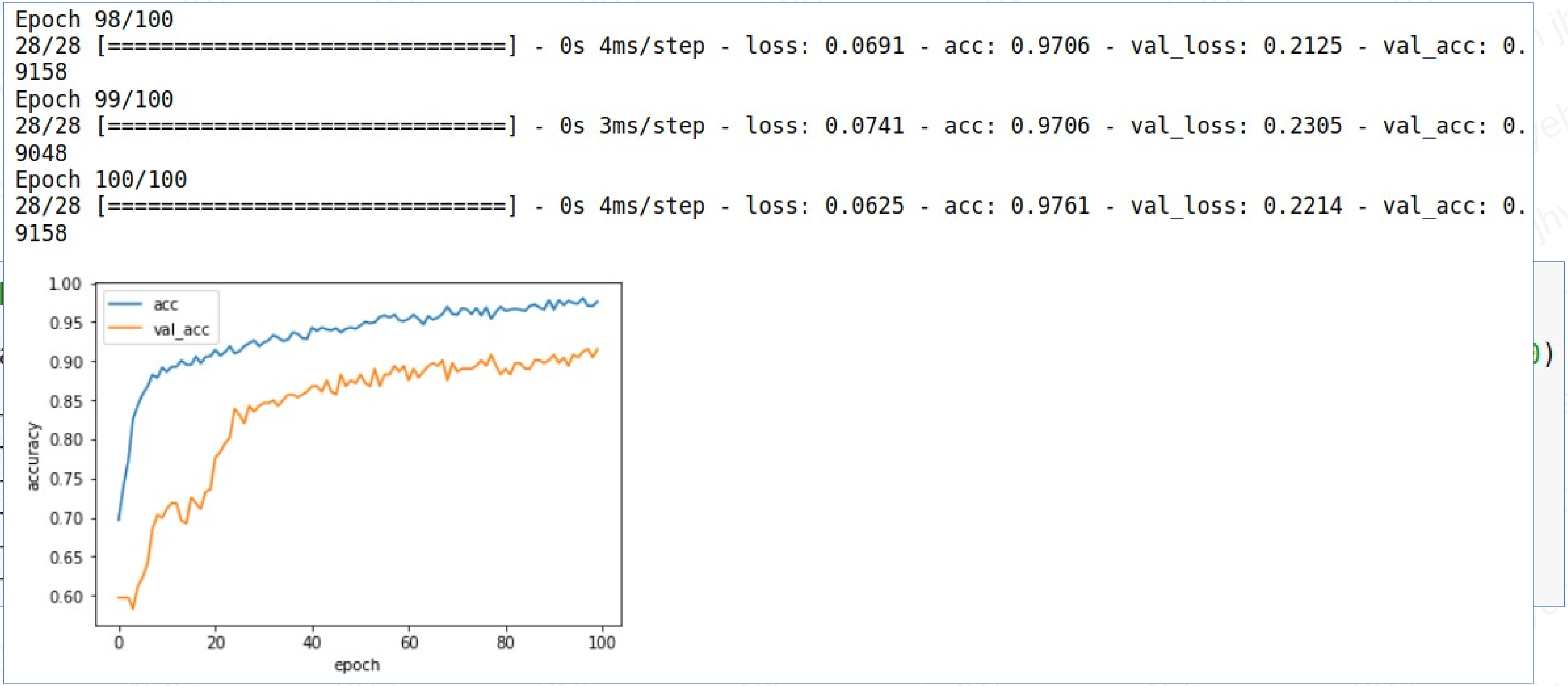
Car Evaluation 使用 MLP 進行學習

- 訓練 train 資料集 (Keras)
  - model.fit() 又出現了!



```
import matplotlib.pyplot as plt

train_history=model.fit(train_data, train_labels, validation_split=0.2, epochs=100, batch_size=40)

plt.plot(train_history.history['acc'], label='acc')
plt.plot(train_history.history['val_acc'], label='val_acc')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(loc='best')
plt.show()
```

Car Evaluation 使用 MLP 進行學習

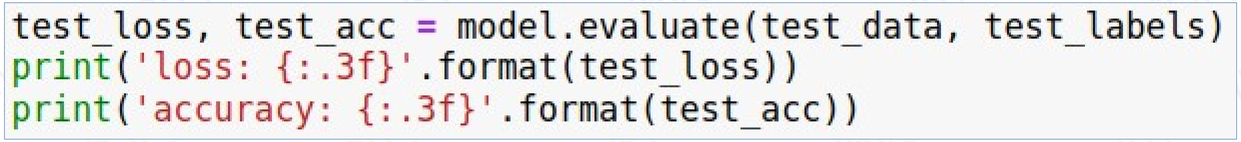


```
Epoch 98/100
28/28 [==============================] - 0s 4ms/step - loss: 0.0691 - acc: 0.9706 - val_loss: 0.2125 - val_acc: 0.
9158
Epoch 99/100
28/28 [==============================] - 0s 3ms/step - loss: 0.0741 - acc: 0.9706 - val_loss: 0.2305 - val_acc: 0.
9048
Epoch 100/100
28/28 [==============================] - 0s 4ms/step - loss: 0.0625 - acc: 0.9761 - val_loss: 0.2214 - val_acc: 0.
9158
```

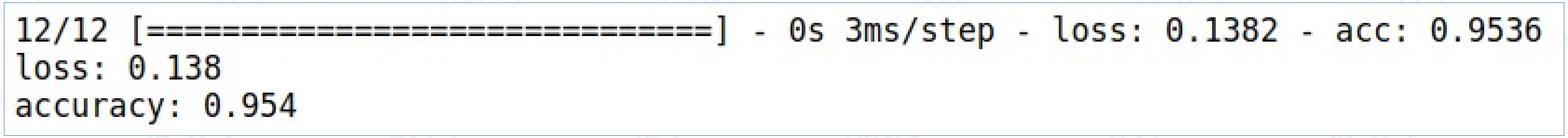
```
imp

tra                                                                                    ))

pl
pl
pl
pl
pl
pl
```

Car Evaluation 使用 MLP 進行學習

- 用 test 資料集做評估 (Keras)
  - 使用 model.evaluate()



```
test_loss, test_acc = model.evaluate(test_data, test_labels)
print('loss: {:.3f}'.format(test_loss))
print('accuracy: {:.3f}'.format(test_acc))
```



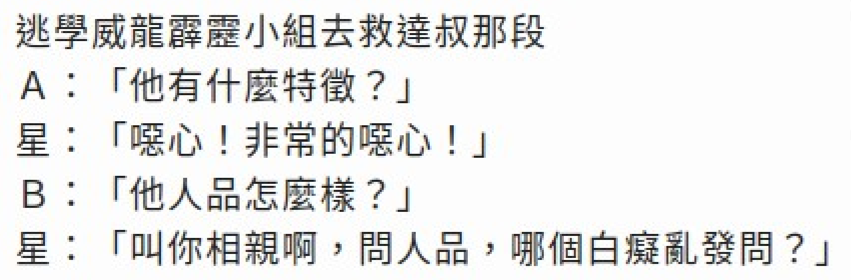
```
12/12 [==============================] - 0s 3ms/step - loss: 0.1382 - acc: 0.9536
loss: 0.138
accuracy: 0.954
```

所以這次的成果就是 95.4% 正確性!

Car Evaluation 使用 MLP 進行學習

- 回顧檢討一下
  - 這次特徵處理和 iris 資料集有什麼不同?
  - 這次建模曲線比較平滑，為什麼?
  - 換了 Adam 不用 SGD，有比較好嗎?
  - activation 用 ReLu?!
  - model.fit() 的參數 ...

是人品問題嗎?



逃學威龍霹靂小組去救達叔那段

A：「他有什麼特徵？」

星：「噁心！非常的噁心！」

B：「他人品怎麼樣？」

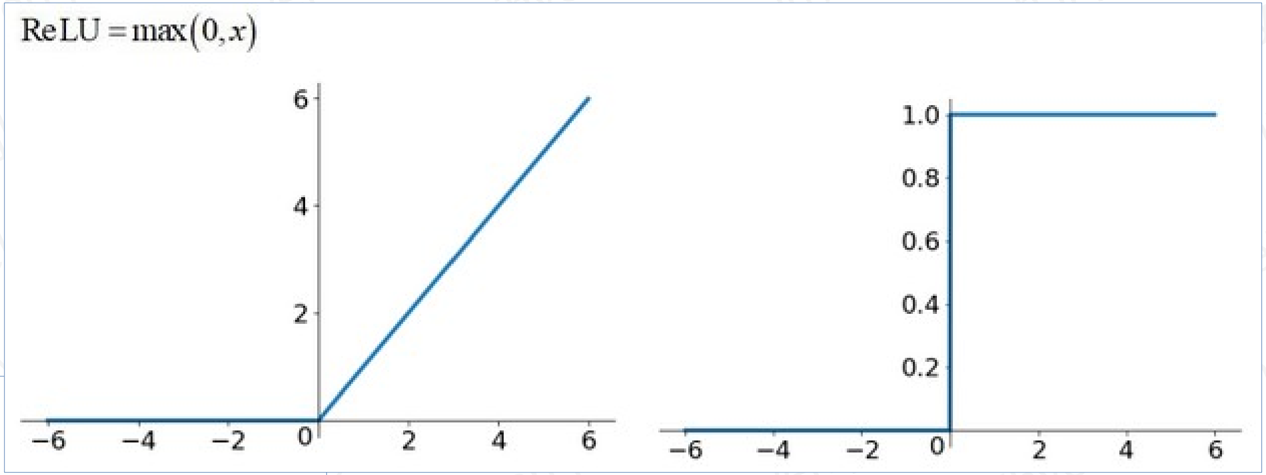
星：「叫你相親啊，問人品，哪個白癡亂發問？」

Sigmoid v.s. ReLu

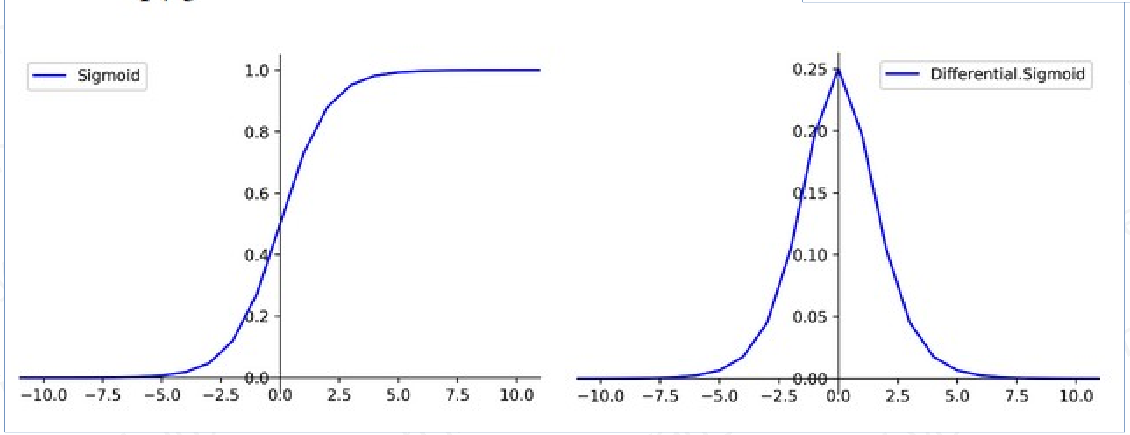
ReLu 可以避免梯度消失問題

（又好算！ ）

$\mathrm{ReLU} = \max(0, x)$



$$\sigma(x) = \frac{1}{1+e^{-x}}$$



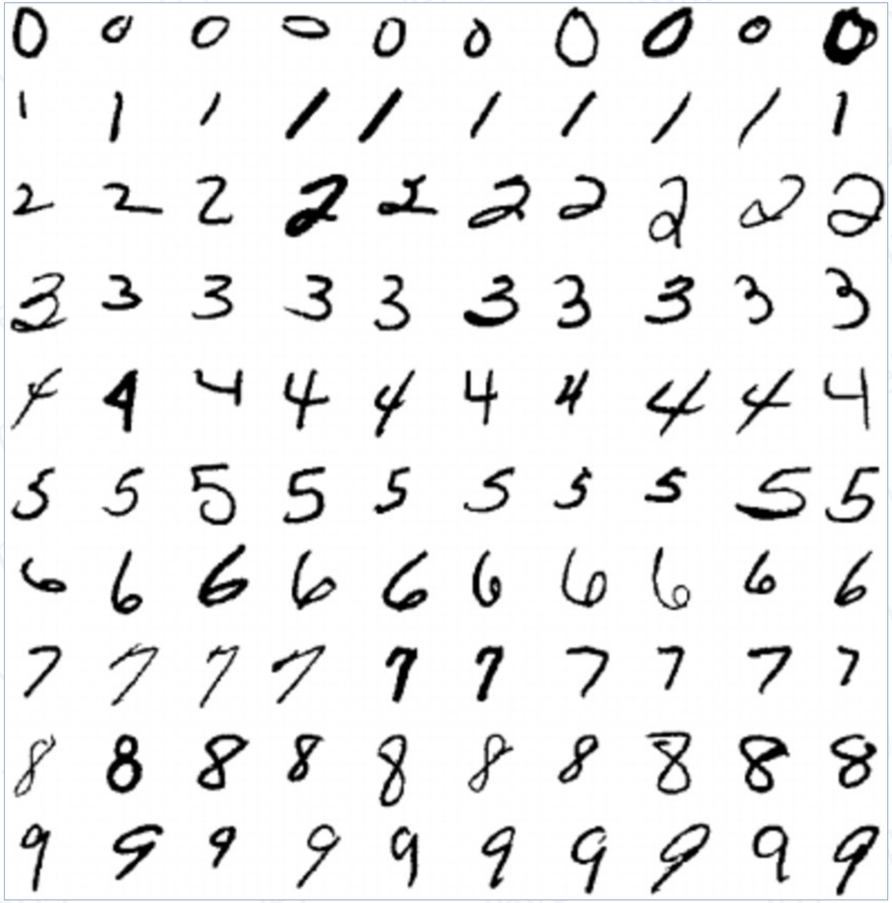
講次內容

- MLP 案例 1： 汽車評估資料預測
- MLP 案例 2： 手寫辨識 MNIST

MLP 案例 2： 手寫辨識 MNIST

- MNIST 資料集
  號稱深度學習的 Hello World!

print('Hello World!') ??



MNIST 資料集

- 由 AI 三大家 Yann LeCun 所建立
  - 2018 Turing Award 獲獎人
  - 卷積神經網路發明人
  - 神經網路影像處理大師

- 60000 筆訓練資料、 10000 筆測試資料
- 手寫數字的影像 (28x28 點灰階影像) 配合標籤

2018 Turing Award

2018 Turing Award

- 計算機工程領域的諾貝爾獎
- Hinton
  - 倒傳遞神經網路、波茲曼機器、改良卷積神經網路
- Bengio
  - 神經網路結合隱馬可夫模型、注意力機制、對抗生成網路
- LeCun
  - 卷積神經網路、改良倒傳遞神經網路、提昇影像處理

MNIST 使用 MLP 進行學習

- 邏輯流程
  - 載入 MNIST 資料集並分成 train 和 test 兩部分 (Keras)
  - 資料前處理 (一點點 tricky...)
  - 根據資料維度，建立 Keras MLP 模型 (Keras)
  - 訓練 train 資料集 (Keras)
  - 用 test 資料集做評估 (Keras)

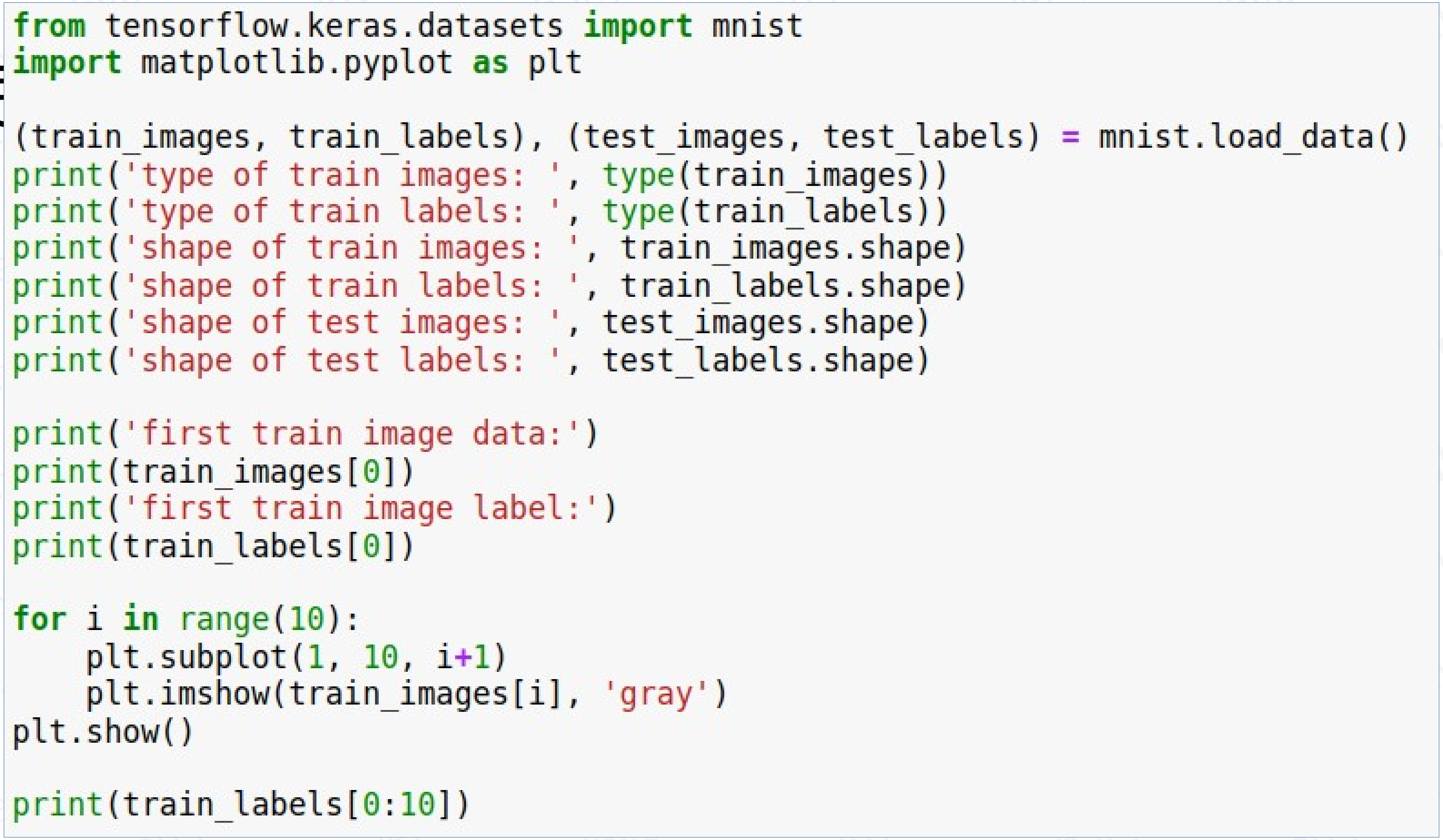
被你發現了！

下載 MNIST 資料集

- 使用 Keras API 下載
  - from tensorflow.keras.datasets import mnist
  - (train_images, train_labels), (test_images, test_labels) = mnist.load_data()
  - Train 和 Test 都幫你分好了！！

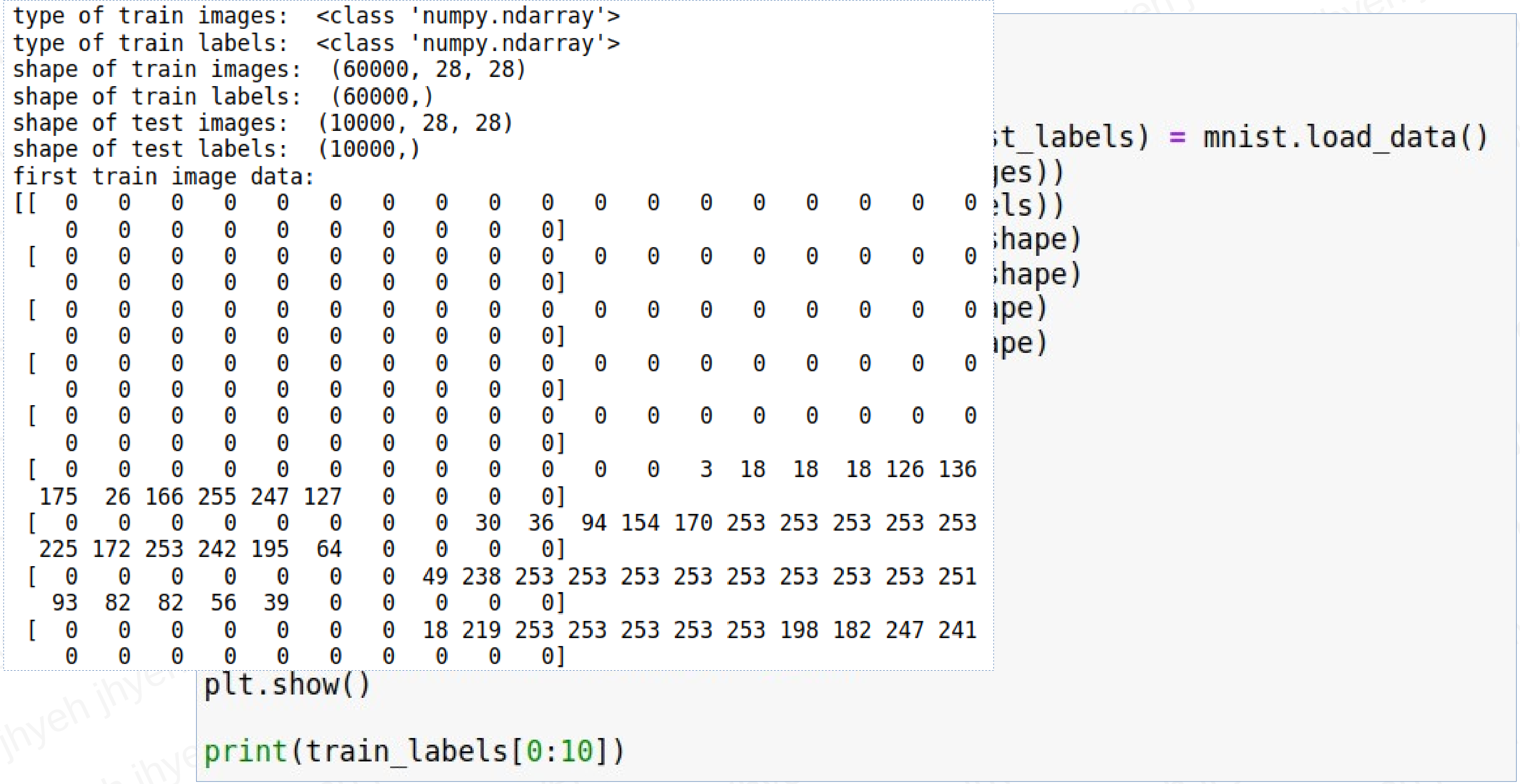
下載 MNIST 資料集

- 使
  –
  –
  –



```
from tensorflow.keras.datasets import mnist
import matplotlib.pyplot as plt

(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
print('type of train images: ', type(train_images))
print('type of train labels: ', type(train_labels))
print('shape of train images: ', train_images.shape)
print('shape of train labels: ', train_labels.shape)
print('shape of test images: ', test_images.shape)
print('shape of test labels: ', test_labels.shape)

print('first train image data:')
print(train_images[0])
print('first train image label:')
print(train_labels[0])

for i in range(10):
    plt.subplot(1, 10, i+1)
    plt.imshow(train_images[i], 'gray')
plt.show()

print(train_labels[0:10])
```

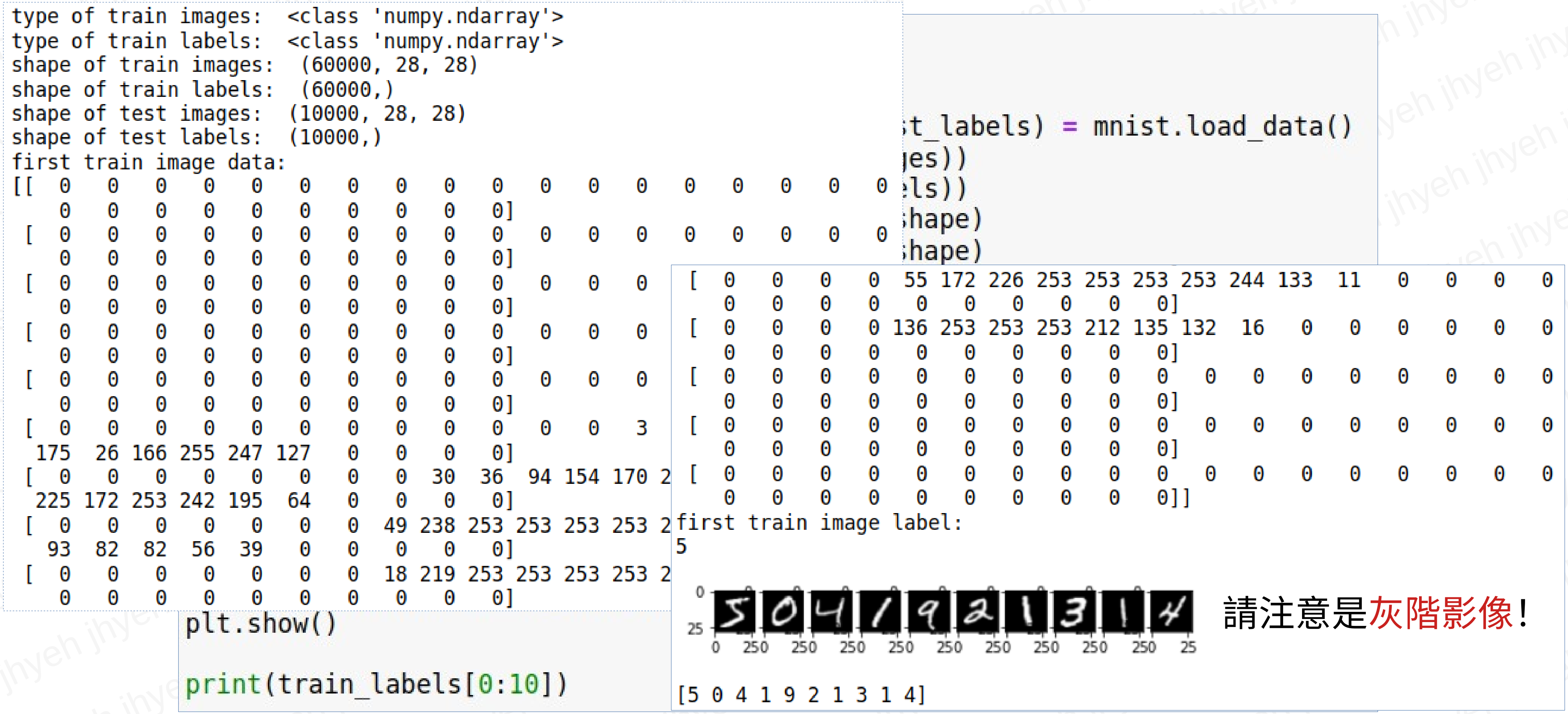
下載 MNIST 資料集



```
type of train images:  <class 'numpy.ndarray'>
type of train labels:  <class 'numpy.ndarray'>
shape of train images:  (60000, 28, 28)
shape of train labels:  (60000,)
shape of test images:  (10000, 28, 28)
shape of test labels:  (10000,)
first train image data:
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   3  18  18  18 126 136
  175  26 166 255 247 127   0   0   0   0]
 [  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253 253
  225 172 253 242 195  64   0   0   0   0]
 [  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251
   93  82  82  56  39   0   0   0   0   0]
 [  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241
    0   0   0   0   0   0   0   0   0   0]
```

```
t_labels) = mnist.load_data()
es))
ls))
hape)
hape)
pe)
pe)
plt.show()

print(train_labels[0:10])
```

下載 MNIST 資料集



```
type of train images:  <class 'numpy.ndarray'>
type of train labels:  <class 'numpy.ndarray'>
shape of train images:  (60000, 28, 28)
shape of train labels:  (60000,)
shape of test images:  (10000, 28, 28)
shape of test labels:  (10000,)
first train image data:
[[   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
     0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
     0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0
     0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0
     0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0
     0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   3
   175  26 166 255 247 127   0   0   0   0]
 [   0   0   0   0   0   0   0   0  30  36  94 154 170 2
   225 172 253 242 195  64   0   0   0   0]
 [   0   0   0   0   0   0   0  49 238 253 253 253 253 2
    93  82  82  56  39   0   0   0   0   0]
 [   0   0   0   0   0   0   0  18 219 253 253 253 253 2
     0   0   0   0   0   0   0   0   0   0]
```

```
t_labels) = mnist.load_data()
ges))
els))
shape)
shape)
```

```
plt.show()

print(train_labels[0:10])
```

```
[   0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
[   0   0   0   0 136 253 253 253 212 135 132  16   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
[   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
[   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
[   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]]
first train image label:
5
```

```
[5 0 4 1 9 2 1 3 1 4]
```

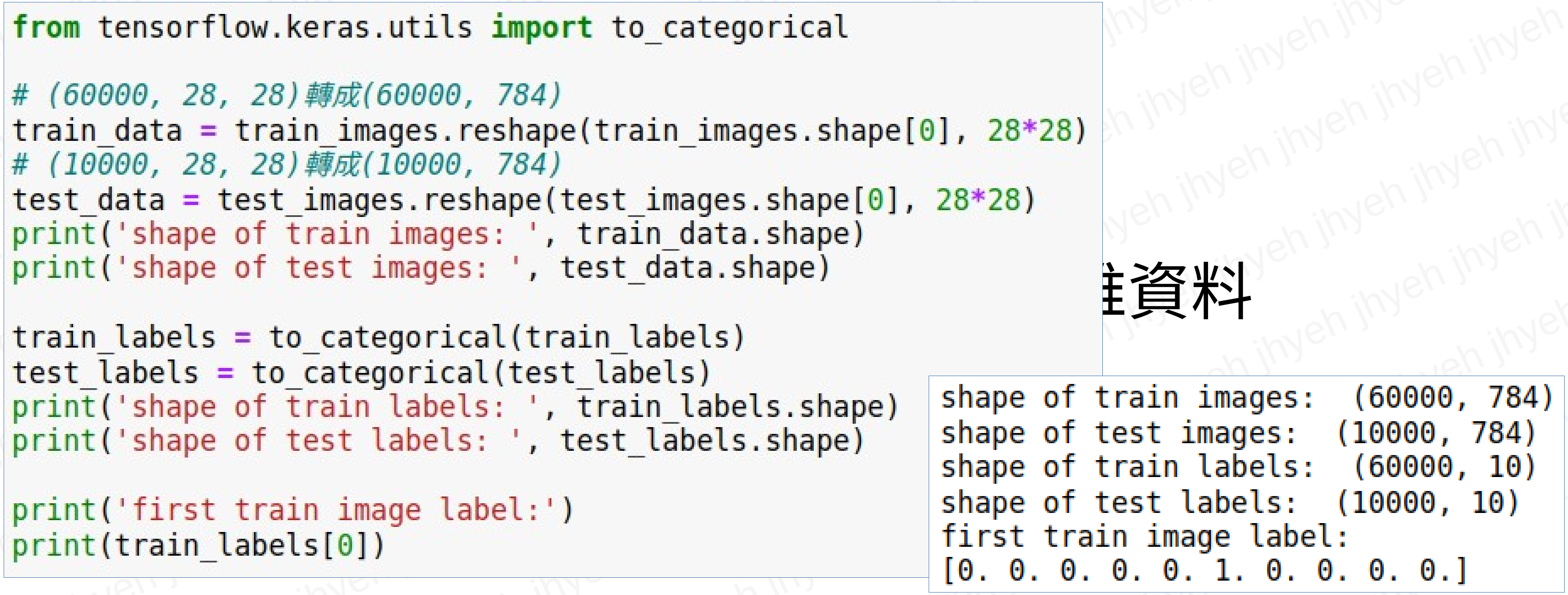
請注意是灰階影像!

MNIST 使用 MLP 進行學習

- 資料前處理 （一點點 tricky...)
  - 訓練資料維度： 60000x28x28
  - 代表 60000 張圖，每張 28x28 ，二維資料
  - 所以為了餵進神經網路，我們要「拉平」成一維 784
  - 標籤本來是 0 到 9 ，我們也將它 categorical 化

MNIST 使用 MLP 進行學習

- 資料

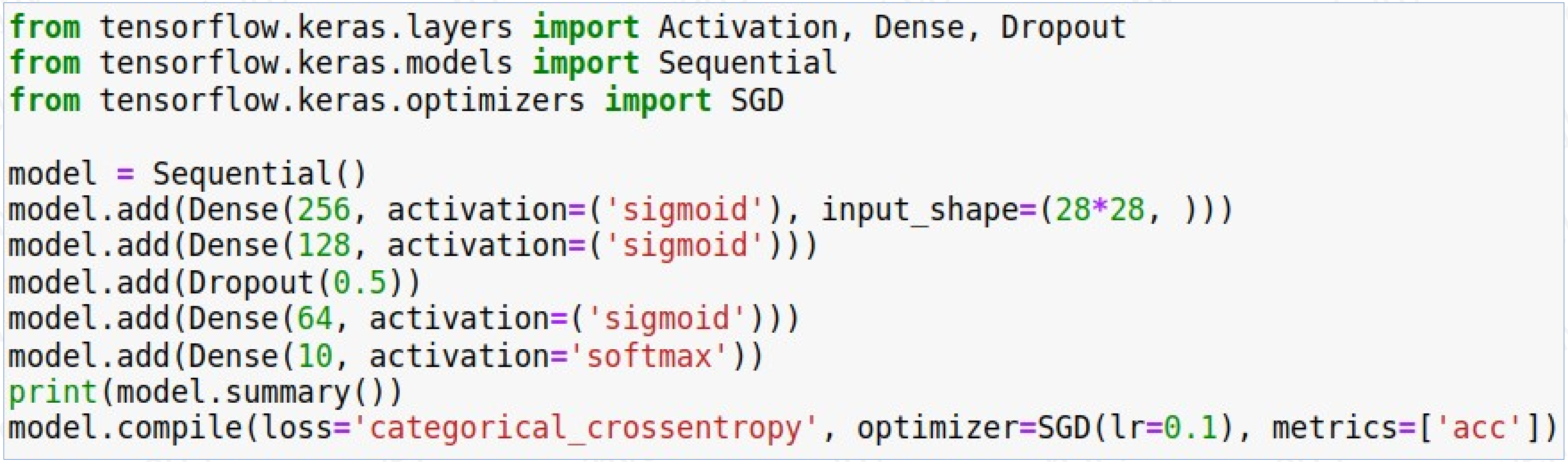


```
from tensorflow.keras.utils import to_categorical

# (60000, 28, 28)轉成(60000, 784)
train_data = train_images.reshape(train_images.shape[0], 28*28)
# (10000, 28, 28)轉成(10000, 784)
test_data = test_images.reshape(test_images.shape[0], 28*28)
print('shape of train images: ', train_data.shape)
print('shape of test images: ', test_data.shape)

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
print('shape of train labels: ', train_labels.shape)
print('shape of test labels: ', test_labels.shape)

print('first train image label:')
print(train_labels[0])
```

```
shape of train images:  (60000, 784)
shape of test images:  (10000, 784)
shape of train labels:  (60000, 10)
shape of test labels:  (10000, 10)
first train image label:
[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
```

MNIST 使用 MLP 進行學習

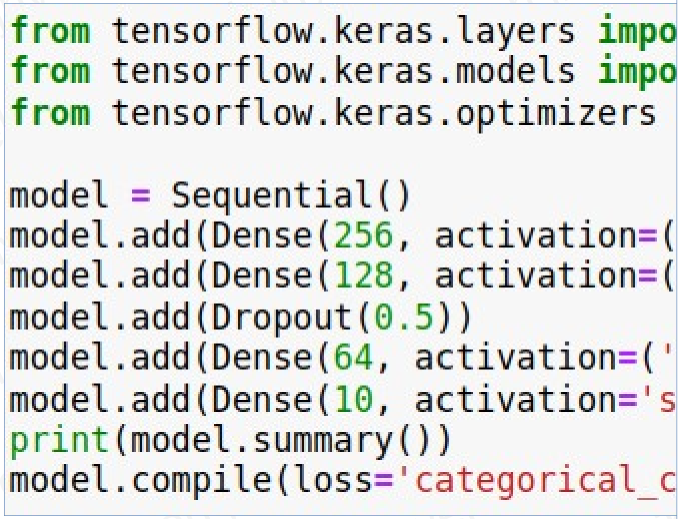
- 根據資料維度，建立 Keras MLP 模型 (Keras)
  - 留意 Dropout()



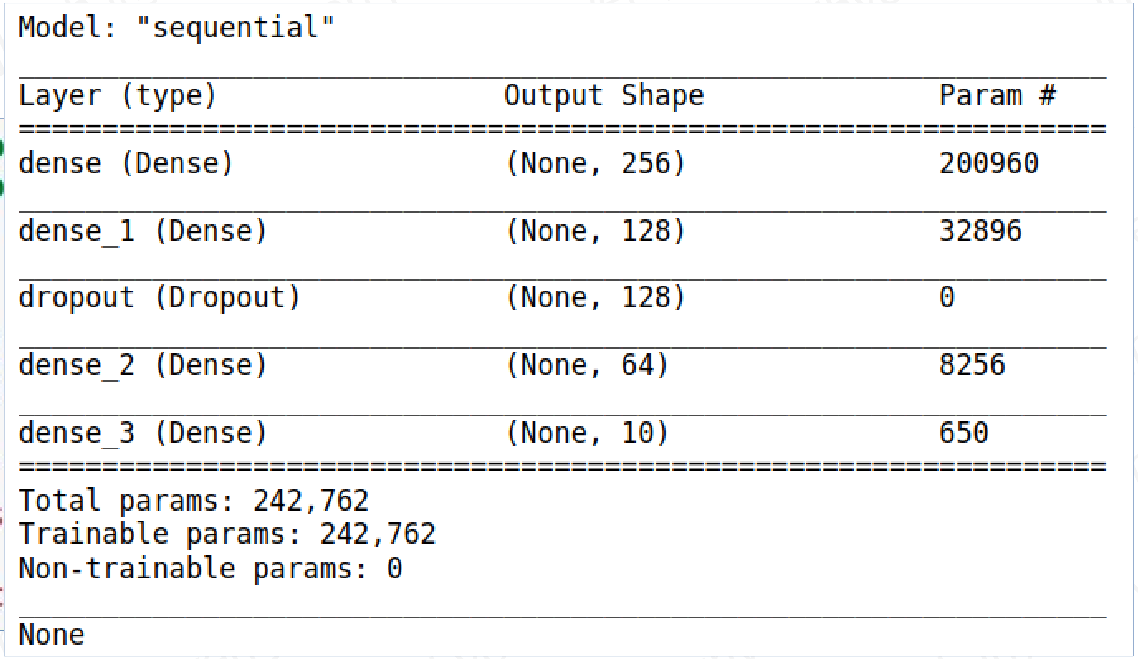
```
from tensorflow.keras.layers import Activation, Dense, Dropout
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import SGD

model = Sequential()
model.add(Dense(256, activation=('sigmoid'), input_shape=(28*28, )))
model.add(Dense(128, activation=('sigmoid')))
model.add(Dropout(0.5))
model.add(Dense(64, activation=('sigmoid')))
model.add(Dense(10, activation='softmax'))
print(model.summary())
model.compile(loss='categorical_crossentropy', optimizer=SGD(lr=0.1), metrics=['acc'])
```

MNIST 使用 MLP 進行學習

- 根據資料維度，建立 Keras MLP 模型 (Keras)
  - 留意 Dropout()



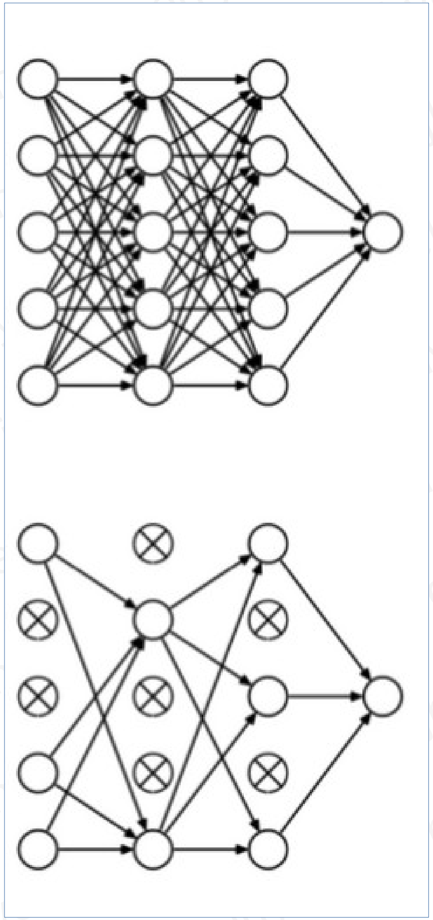
```
from tensorflow.keras.layers impo
from tensorflow.keras.models impo
from tensorflow.keras.optimizers 

model = Sequential()
model.add(Dense(256, activation=(
model.add(Dense(128, activation=(
model.add(Dropout(0.5))
model.add(Dense(64, activation=('
model.add(Dense(10, activation='s
print(model.summary())
model.compile(loss='categorical_c
```



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense (Dense)                (None, 256)               200960    
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896     
_________________________________________________________________
dropout (Dropout)            (None, 128)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256      
_________________________________________________________________
dense_3 (Dense)              (None, 10)                650       
=================================================================
Total params: 242,762
Trainable params: 242,762
Non-trainable params: 0
_________________________________________________________________
None
```

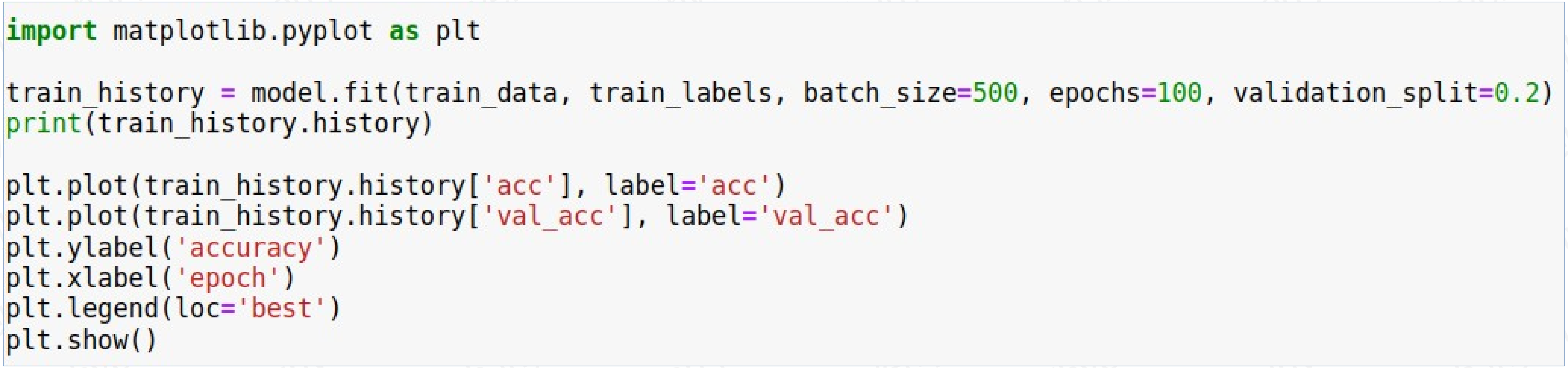
終於要談 Dropout 了！

- 2014 年被提出
- 模型訓練階段，隨機將一些神經元關閉
  - 避免神經元之間過度依賴
- 大幅降低模型過度適配 overfitting 的可能
  - 就是避免模型反應過度！
- Keras 的 Dropout() 參數為隨機關掉的神經元比例



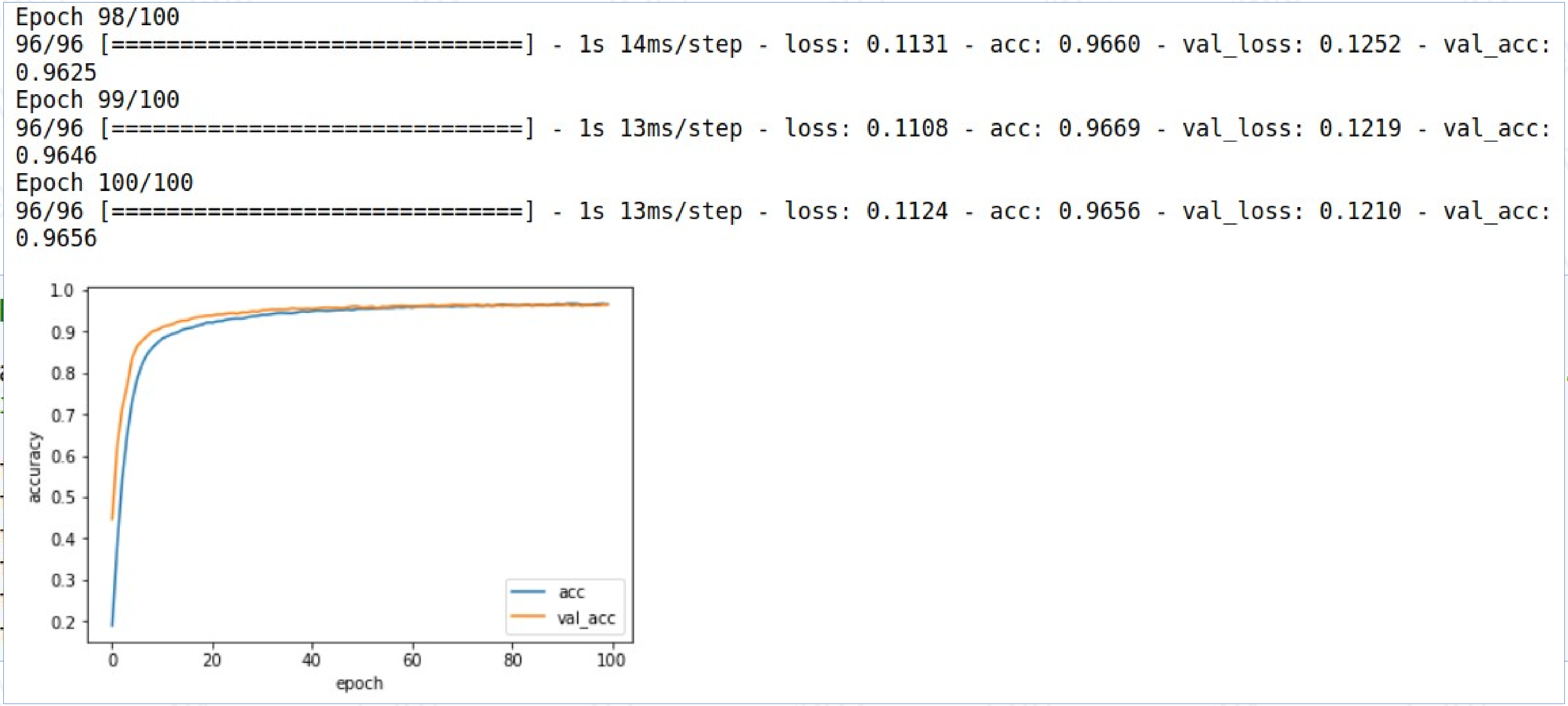
MNIST 使用 MLP 進行學習

- 訓練 train 資料集 (Keras)
  - model.fit() 又出現了!



```python
import matplotlib.pyplot as plt

train_history = model.fit(train_data, train_labels, batch_size=500, epochs=100, validation_split=0.2)
print(train_history.history)

plt.plot(train_history.history['acc'], label='acc')
plt.plot(train_history.history['val_acc'], label='val_acc')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(loc='best')
plt.show()
```

MNIST 使用 MLP 進行學習

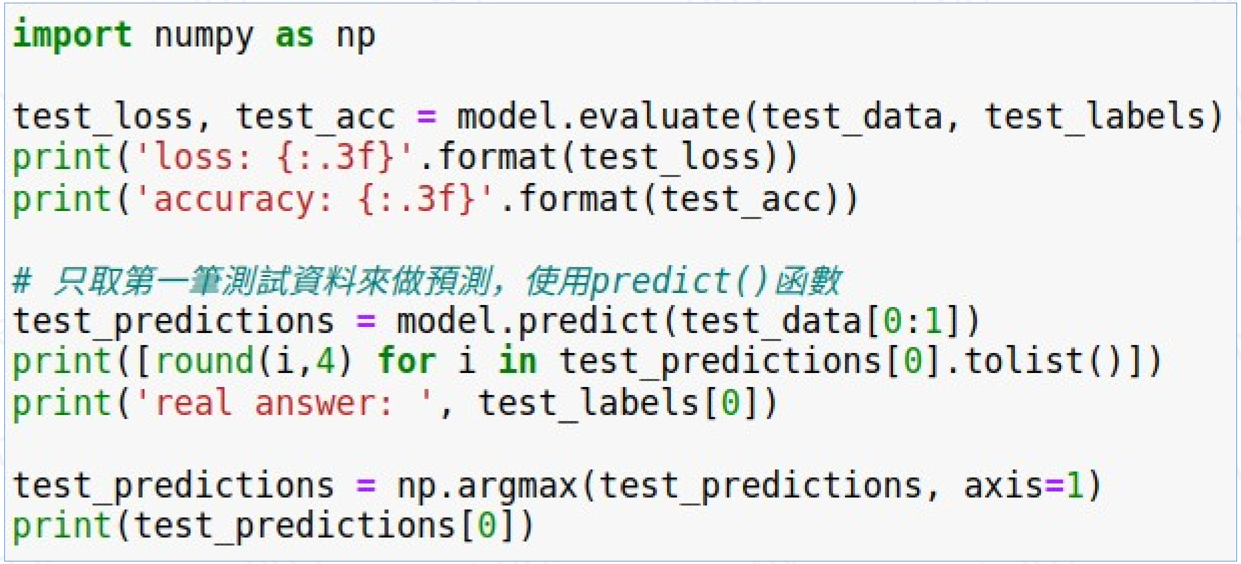


```
Epoch 98/100
96/96 [==============================] - 1s 14ms/step - loss: 0.1131 - acc: 0.9660 - val_loss: 0.1252 - val_acc:
0.9625
Epoch 99/100
96/96 [==============================] - 1s 13ms/step - loss: 0.1108 - acc: 0.9669 - val_loss: 0.1219 - val_acc:
0.9646
Epoch 100/100
96/96 [==============================] - 1s 13ms/step - loss: 0.1124 - acc: 0.9656 - val_loss: 0.1210 - val_acc:
0.9656
```

```
imp

tra                                                                    .2)
pri

pl
pl
pl
pl
pl
pl
```

MNIST 使用 MLP 進行學習

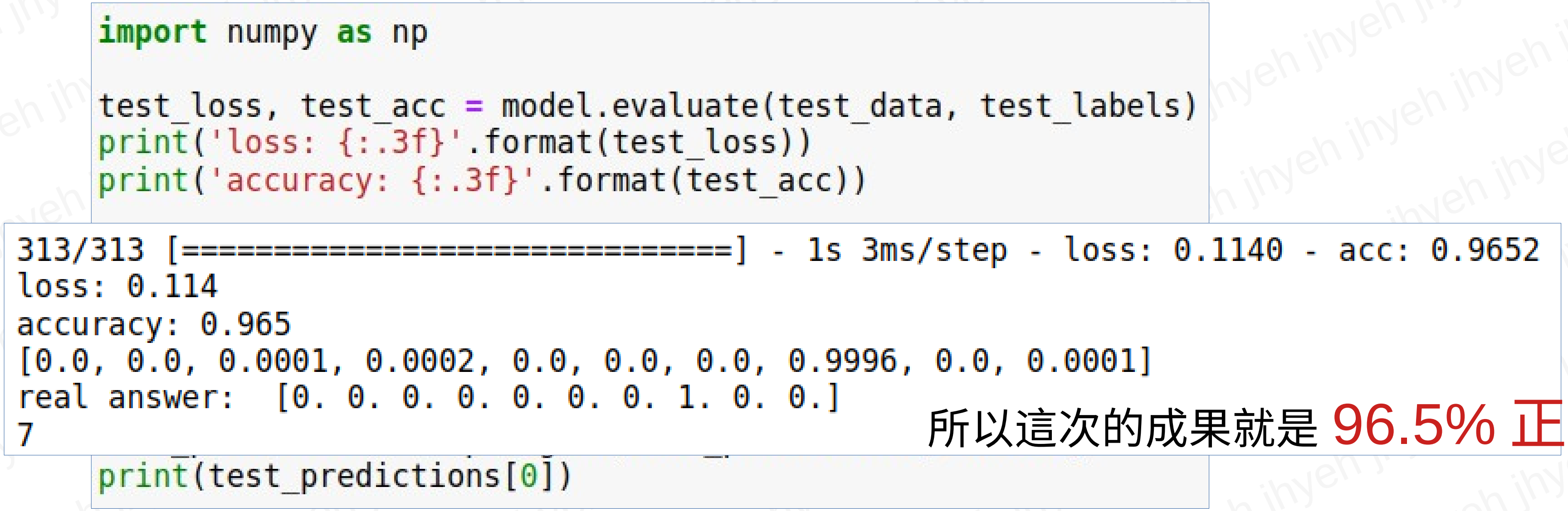
- 用 test 資料集做評估　(Keras)
  - 使用 model.evaluate() 和 model.predict()



```
import numpy as np

test_loss, test_acc = model.evaluate(test_data, test_labels)
print('loss: {:.3f}'.format(test_loss))
print('accuracy: {:.3f}'.format(test_acc))

# 只取第一筆測試資料來做預測，使用predict()函數
test_predictions = model.predict(test_data[0:1])
print([round(i,4) for i in test_predictions[0].tolist()])
print('real answer: ', test_labels[0])

test_predictions = np.argmax(test_predictions, axis=1)
print(test_predictions[0])
```

MNIST 使用 MLP 進行學習

- 用 test 資料集做評估 (Keras)
  - 使用 model.evaluate() 和 model.predict()



```
import numpy as np

test_loss, test_acc = model.evaluate(test_data, test_labels)
print('loss: {:.3f}'.format(test_loss))
print('accuracy: {:.3f}'.format(test_acc))
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.1140 - acc: 0.9652
loss: 0.114
accuracy: 0.965
[0.0, 0.0, 0.0001, 0.0002, 0.0, 0.0, 0.0, 0.9996, 0.0, 0.0001]
real answer:  [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
7
```

```
print(test_predictions[0])
```

所以這次的成果就是 96.5% 正確性!

MNIST 使用 MLP 進行學習

- 回顧檢討一下
  - 這次特徵處理和 Car Evaluation 資料集有什麼不同?
  - 又換回了 SGD，有比較好嗎?
  - model.fit() 的參數 ...

這個講次中，你應該學到了 ...

- Car 的 MLP 處理，注意特徵對應!
- MNIST 的 MLP 處理，二維資料!