Keras-實作DNN

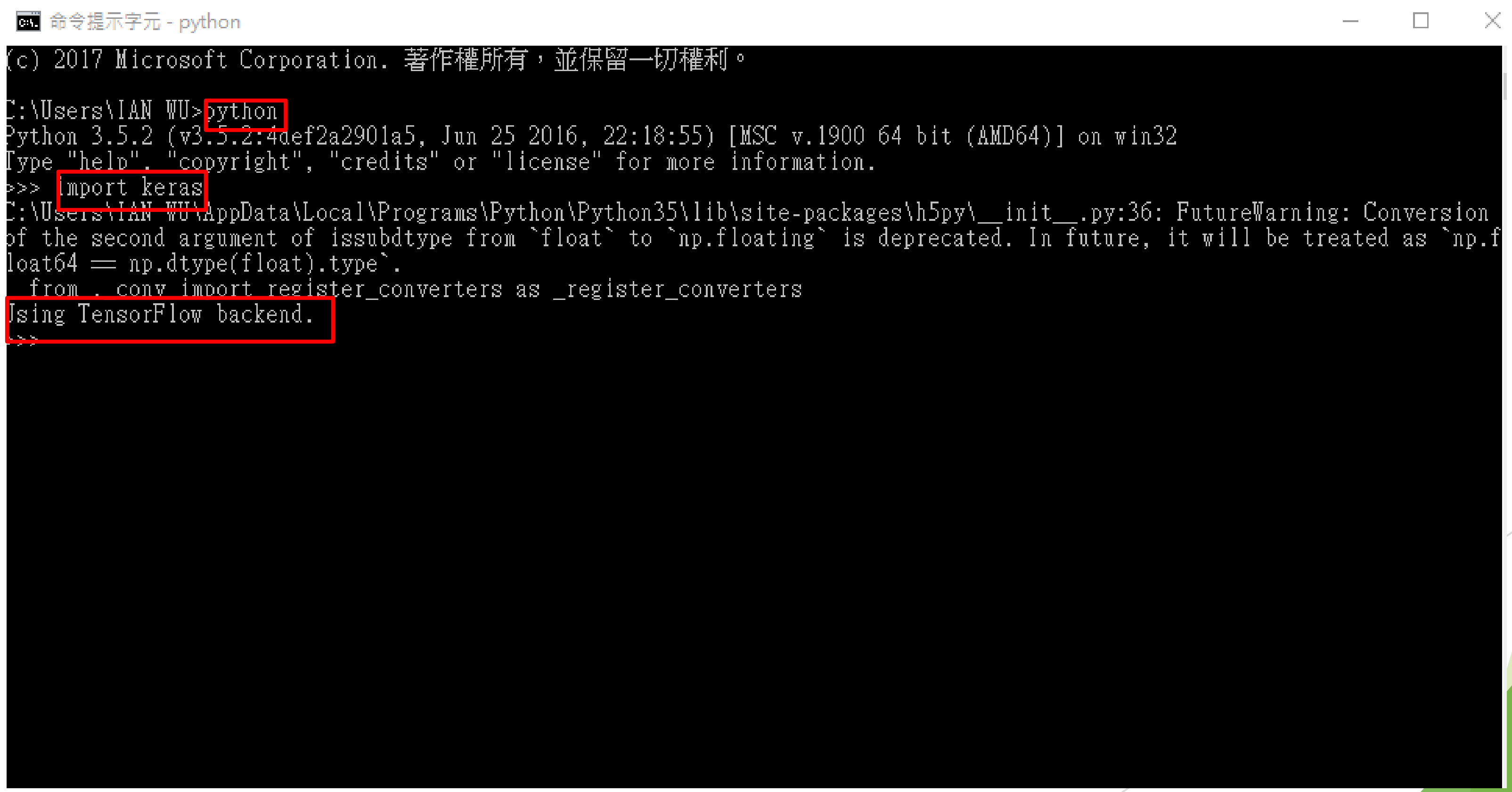
Keras

- Keras是以Tensorflow或Theano為Backend來建立Model的框架。
- Keras 是一個非常高層的庫，Keras 強調極簡主義—你只需幾行代碼就能構建一個神經網絡。
- 本課堂主要以Tensorflow為Backend為主。

	Keras	Tensorflow
使用語言	Python	Python\C++\...等
學習難度	簡單易上手	相對困難
使用彈性	中	高
適合使用者	初學者	進階使用者

Keras-安裝

- 在命令題示字元(cmd)裡輸入: **pip install keras**
- 完成後在cmd的命令行輸入python，接著輸入**import keras**，若顯示**Using Tensorflow backend**，則表示安裝成功。

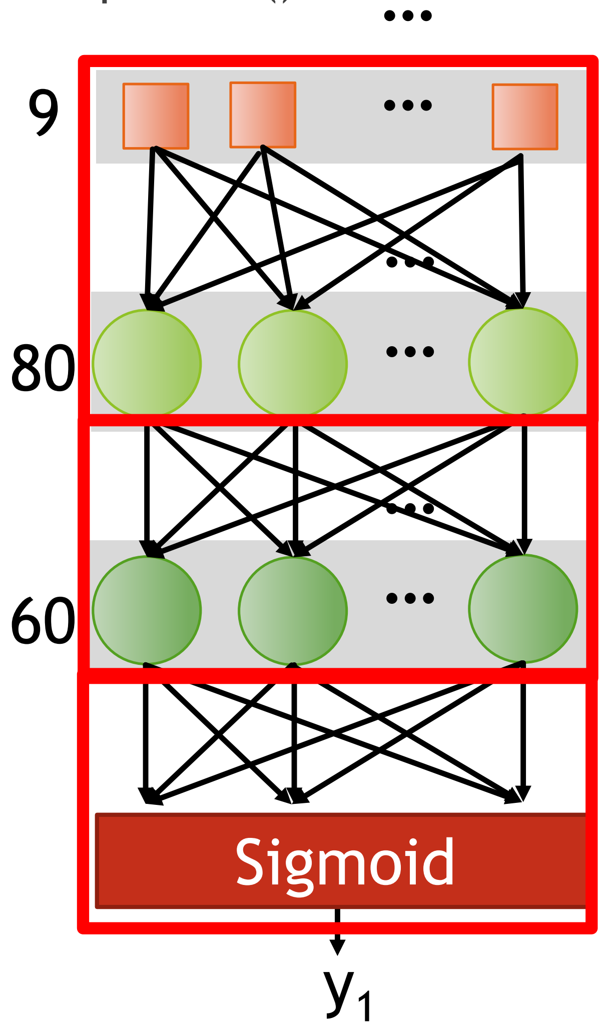

命令提示字元 - python

```
(c) 2017 Microsoft Corporation. 著作權所有，並保留一切權利。

C:\Users\IAN WU>python
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:18:55) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import keras
C:\Users\IAN WU\AppData\Local\Programs\Python\Python35\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion
of the second argument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.f
loat64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
>>>
```

```python
import os
import numpy as np
import pandas as pd
from sklearn import preprocessing
from keras.models import Sequential
from keras.layers import Dense, Dropout
```

Keras-建構DNN Model

- Sequential()模型就像是堆積木一樣，可以疊出我們想要的Model。



```python
model = Sequential()
```

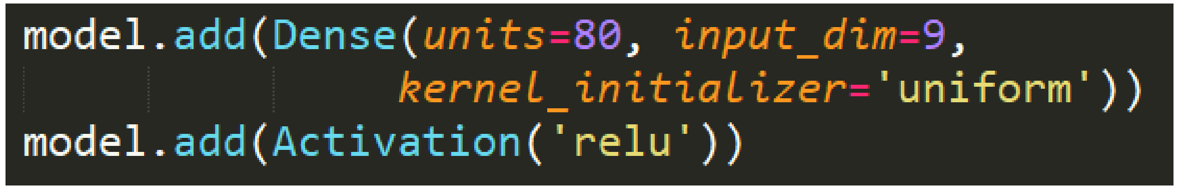
```python
model.add(Dense(units=80, input_dim=9,
                kernel_initializer='uniform'))
model.add(Activation('relu'))
```

```python
model.add(Dense(units=60,
                kernel_initializer='uniform',
model.add(Activation('relu'))
```

```python
model.add(Dense(units=1,
                kernel_initializer='uniform',
model.add(Activation('sigmoid'))
```

Keras-建構DNN Model

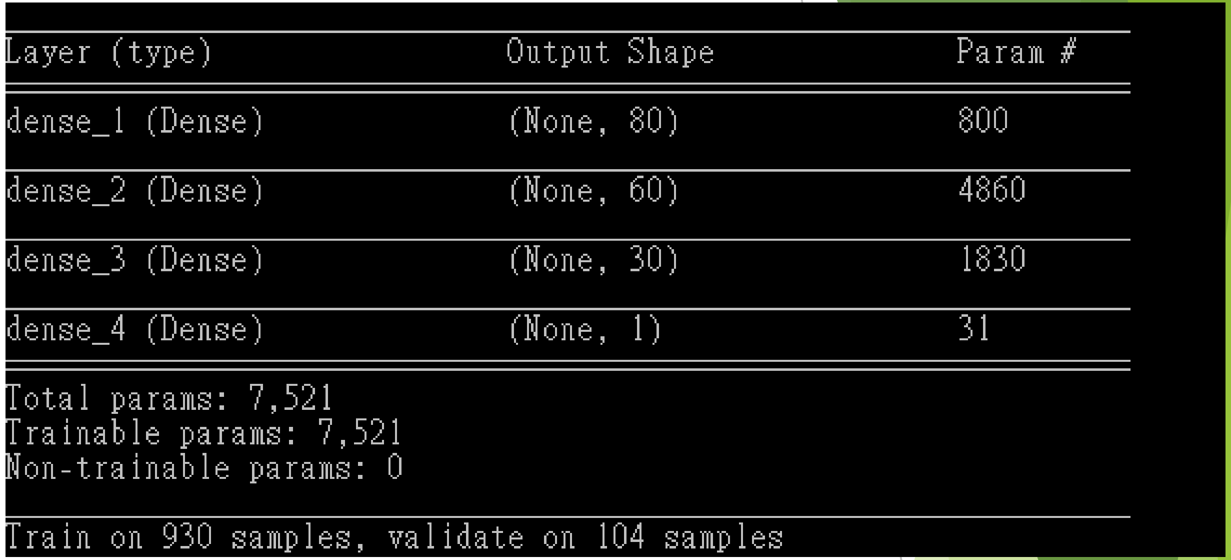
- Dense:Fully Connected Network。
- Units:這一層要設置幾個Neuron 。
- Input_dim:輸入的維度為多少。
- Kernel_initializer:Weight要初始化用什麼方法(Zeros, Ones, Constant, RandomNormal, RandomUniform…等等)
- Activation:要使用什麼激活函數(softmax, softplus, softsign, relu, tanh, hard_sigmoid, linear…等等)



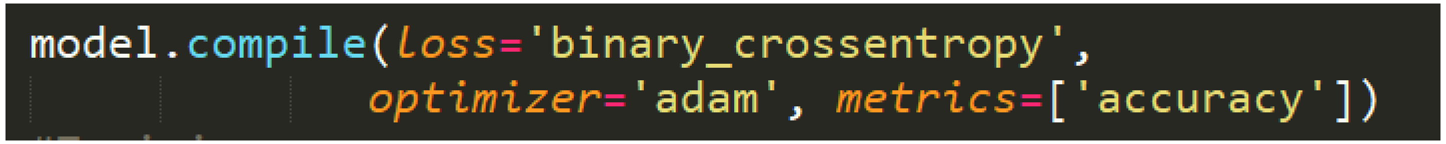
```
model.add(Dense(units=80, input_dim=9,
                kernel_initializer='uniform'))
model.add(Activation('relu'))
```

Keras-建構DNN Model

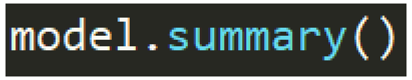
- Loss:要使用哪種Loss Function。
- Optimizer:要使用哪種優化器。
- Metrics=accuracy:評估方式用accuracy。
- Model.summary:可以查看模型摘要。



```
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 80)                800
_________________________________________________________________
dense_2 (Dense)              (None, 60)                4860
_________________________________________________________________
dense_3 (Dense)              (None, 30)                1830
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 31
=================================================================
Total params: 7,521
Trainable params: 7,521
Non-trainable params: 0
_________________________________________________________________
Train on 930 samples, validate on 104 samples
```

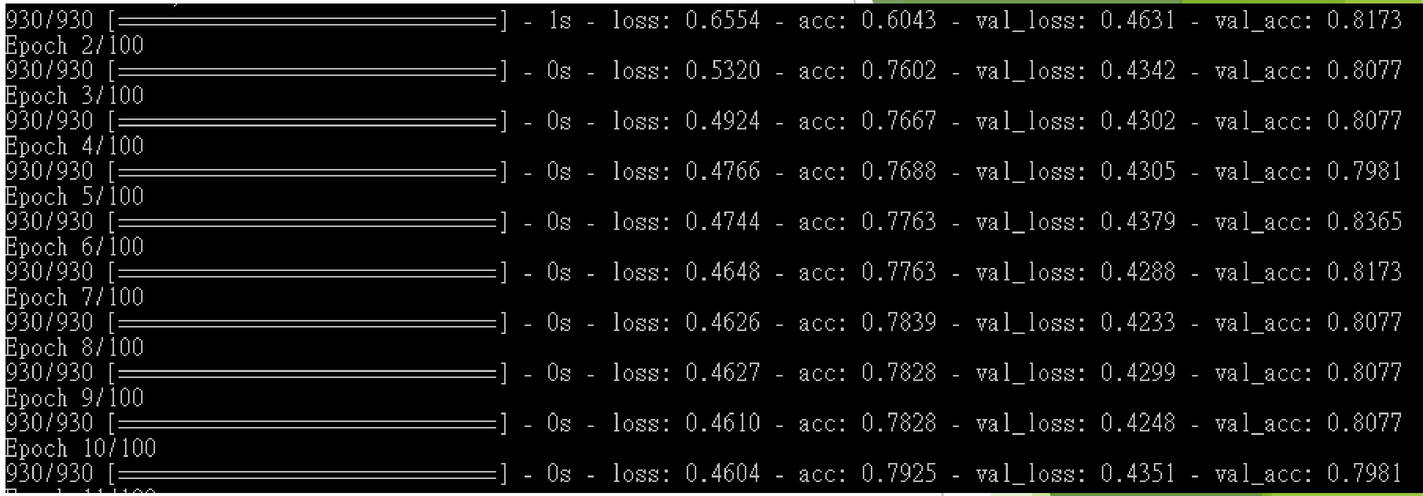


```python
model.compile(loss='binary_crossentropy',
              optimizer='adam', metrics=['accuracy'])
```



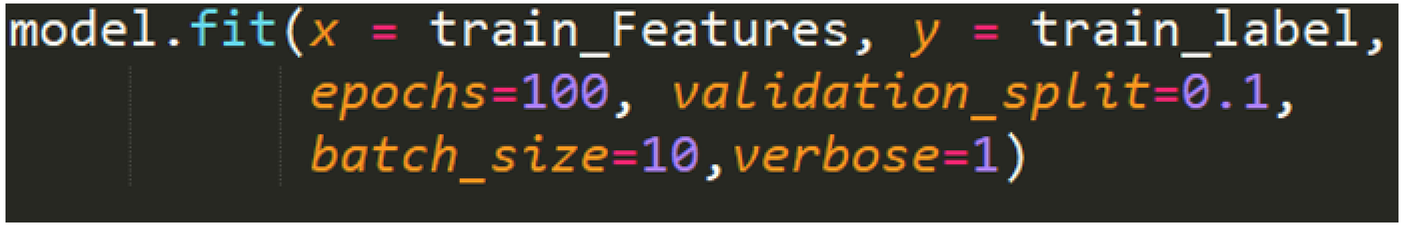
```python
model.summary()
```

Keras-建構DNN Model



```
930/930 [==============================] - 1s - loss: 0.6554 - acc: 0.6043 - val_loss: 0.4631 - val_acc: 0.8173
Epoch 2/100
930/930 [==============================] - 0s - loss: 0.5320 - acc: 0.7602 - val_loss: 0.4342 - val_acc: 0.8077
Epoch 3/100
930/930 [==============================] - 0s - loss: 0.4924 - acc: 0.7667 - val_loss: 0.4302 - val_acc: 0.8077
Epoch 4/100
930/930 [==============================] - 0s - loss: 0.4766 - acc: 0.7688 - val_loss: 0.4305 - val_acc: 0.7981
Epoch 5/100
930/930 [==============================] - 0s - loss: 0.4744 - acc: 0.7763 - val_loss: 0.4379 - val_acc: 0.8365
Epoch 6/100
930/930 [==============================] - 0s - loss: 0.4648 - acc: 0.7763 - val_loss: 0.4288 - val_acc: 0.8173
Epoch 7/100
930/930 [==============================] - 0s - loss: 0.4626 - acc: 0.7839 - val_loss: 0.4233 - val_acc: 0.8077
Epoch 8/100
930/930 [==============================] - 0s - loss: 0.4627 - acc: 0.7828 - val_loss: 0.4299 - val_acc: 0.8077
Epoch 9/100
930/930 [==============================] - 0s - loss: 0.4610 - acc: 0.7828 - val_loss: 0.4248 - val_acc: 0.8077
Epoch 10/100
930/930 [==============================] - 0s - loss: 0.4604 - acc: 0.7925 - val_loss: 0.4351 - val_acc: 0.7981
```

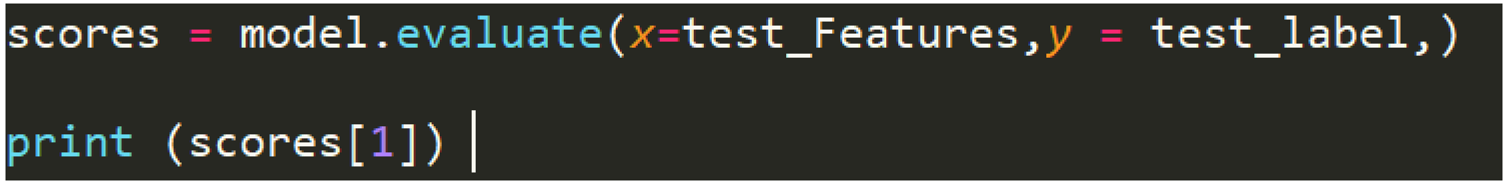
- x為training data 。
- Y為training data對應的label 。
- epochs為訓練週期次數。
- validation_split = 0.1表示從training data中切出10%當作validation set(驗證集)，可以藉此觀察資料是否Overfitting。
- batch_size為每次丟多少筆資料訓練
- Verbose = 1可以顯示Model每次訓練的Loss及準確率。



```
model.fit(x = train_Features, y = train_label,
          epochs=100, validation_split=0.1,
          batch_size=10,verbose=1)
```

準確率

- x為Testing Data 。
- y為Testing Data的Label 。
- scores[0]為最後Loss多少。
- scores[1]為準確率多少。



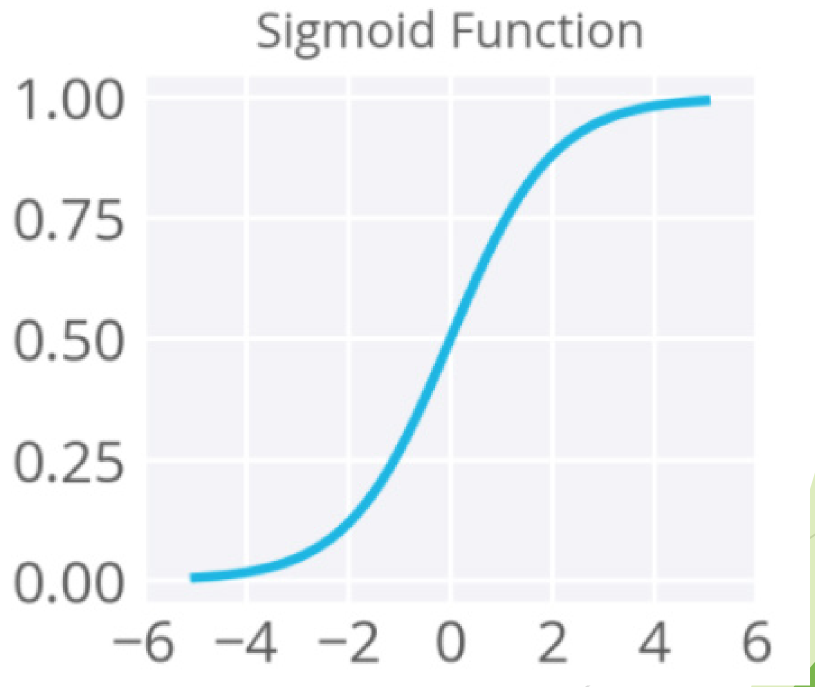
```
scores = model.evaluate(x=test_Features,y = test_label,)

print (scores[1])
```

Activation Function

- Sigmoid:值介於 [0,1] 之間，且分布兩極化，大部分不是 0，就是 1，適合二分法。

公式:

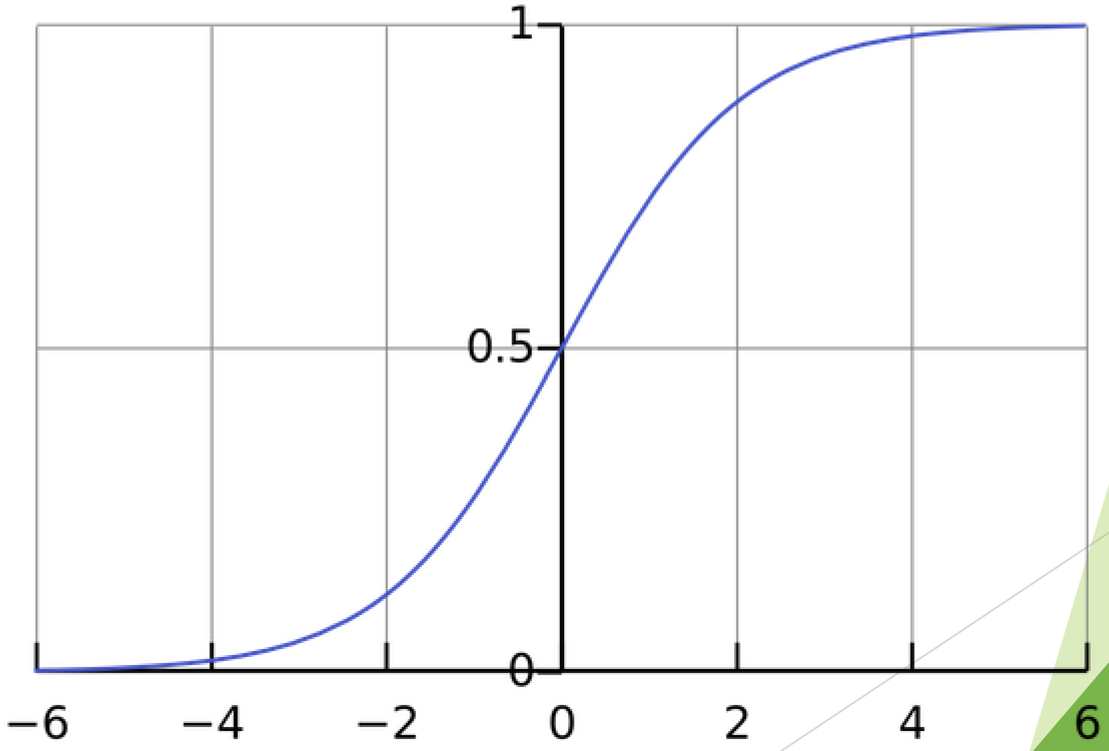
$$f(x) = \frac{1}{1+e^{-x}}$$



Activation Function

- Softmax:值介於 [0,1] 之間，且機率總和等於 1，適合多分類使用。
- 公式:

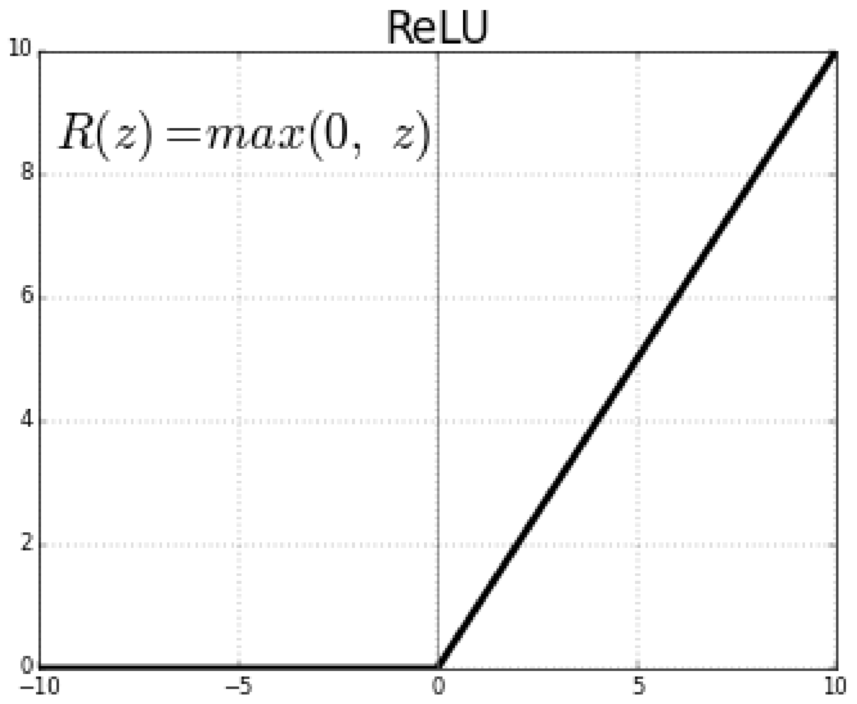
$$f_i(\vec{x}) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}}$$



Activation Function

- Relu (Rectified Linear Units)：忽略負值，介於 [0,∞] 之間。
- 公式:

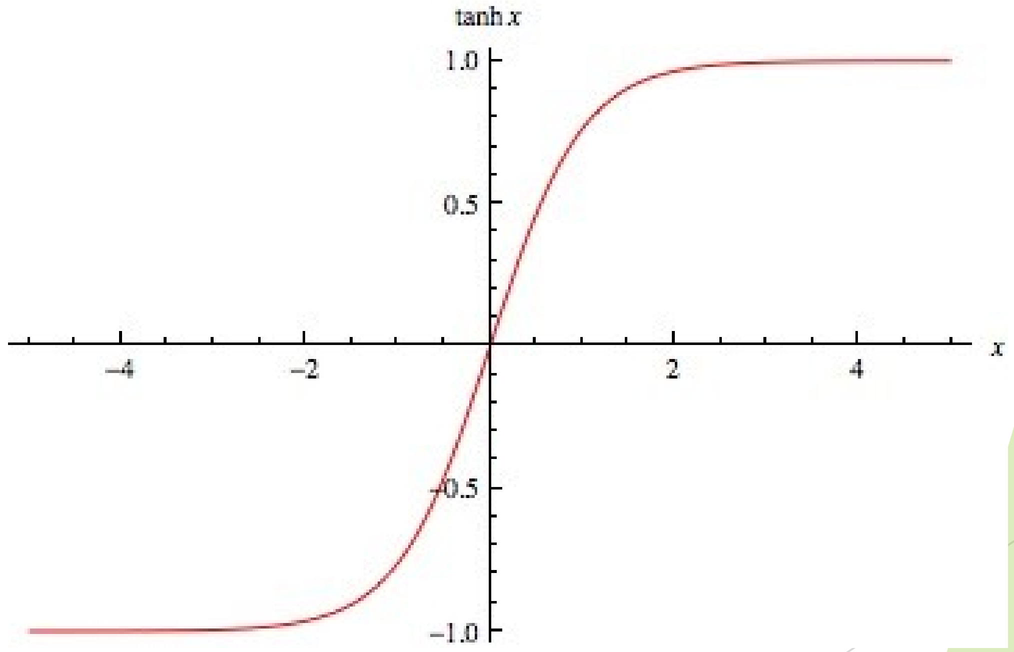
$$f(x) = max(x, 0)$$



Activation Function

- tanh：與sigmoid類似，但值介於[-1,1]之間，即傳導有負值。
- 公式:

$$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$$



Optimizers-BGD,SGD

- Stochastic gradient descent:
  - 梯度更新規則:是指梯度下降演算法在訓練集上，對每一個訓練資料都計算誤差並更新模型。
  - 缺點:
    - 相比其他來說，計算消耗更大，在大資料集上花費的訓練時間更多。
    - 頻繁的更新產生的雜訊可能導致模型參數和模型誤差來回跳動
- Batch gradient descent:
  - 梯度更新規則:批量梯度下降是在每一個訓練epoch之後更新模型
  - 缺點:
    - 更穩定的誤差梯度可能導致模型過早收斂於一個不是最優解的參數集。
    - 批量梯度下降演算法需要把所有的訓練資料都存放在記憶體中,所以在大資料集上，訓練速度會非常慢。

Optimizers-Mini-Batch gradient descent

- Mini-Batch gradient descent:
  - 梯度更新規則:小批量梯度下降把訓練集劃分為很多批，對每一批（batch）計算誤差並更新參數。
  - 比批量梯度下降更快的更新頻率有利於收斂，避免局部最優。
  - 缺點:
    - 小批量梯度下降給演算法增加了一個超參數batch size。
    - 和批量梯度下降一樣，每一個batch上的誤差需要累加。
  - 較小的值讓學習過程收斂更快，但是產生更多雜訊。
  - 較大的值讓學習過程收斂較慢，但是準確的估計誤差梯度。

Loss Function-Mean Squared Error

- 均方誤差(Mean squared error)：就是最小平方法(Least Square) 的目標函數-預測值與實際值的差距之平均值。

- 公式:

$$\sum \left(\hat{y}^2 - y^2\right) / N$$

Loss Function-Cross Entropy

- Cross Entropy (categorical crossentropy)：當預測值與實際值愈相近，損失函數就愈小，反之差距很大，就會更影響損失函數的值，Cross Entropy 取代 MSE，因為，在梯度下時，Cross Entropy 計算速度較快。

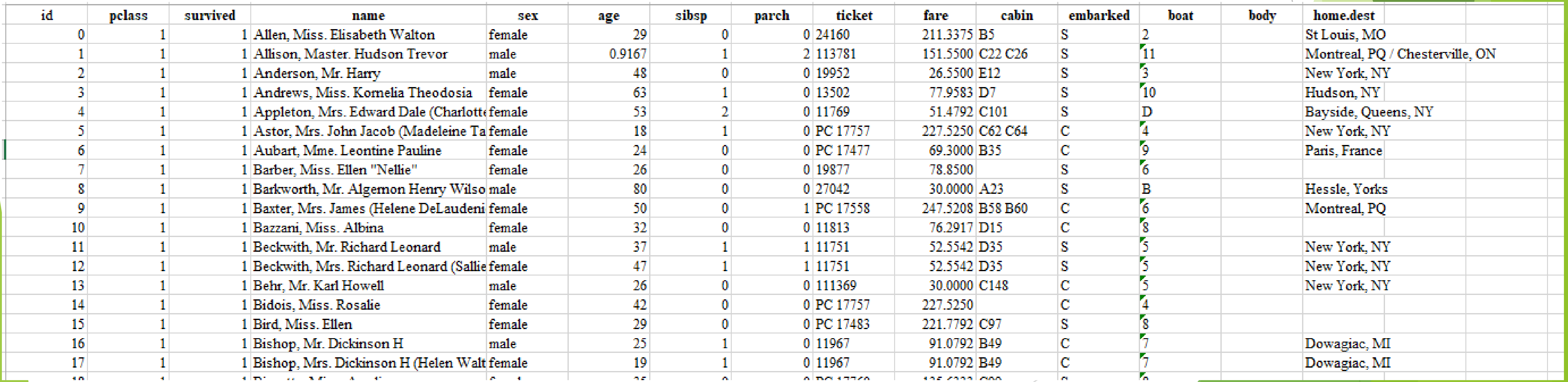
- 公式:

$$L(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} H(p_n, q_n) = -\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log \hat{y}_n + (1 - y_n)\log(1 - \hat{y}_n)\right]$$

Homework

鐵達尼號旅客資料集

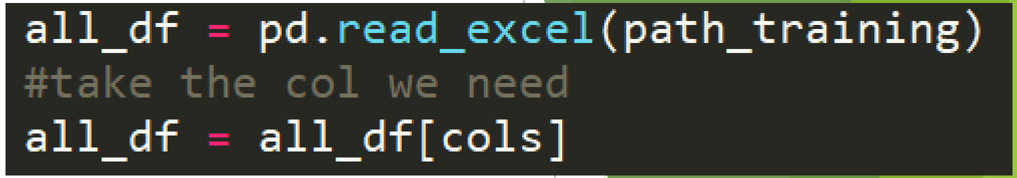
- 1000筆旅客資料當作training data，309筆當作testing data。
- Id:第幾筆資料、Pclass:艙等、Survived:是否生存、Name:姓名、Sex:性別、Age:年齡、Sibsp:手足或配偶在船上的數量、Parch:雙親或子女在船上的數量、Ticket:車票號碼、Fare:旅客費用、Cabin:艙位號碼、Embarked:登船港口、Home.dest:住址



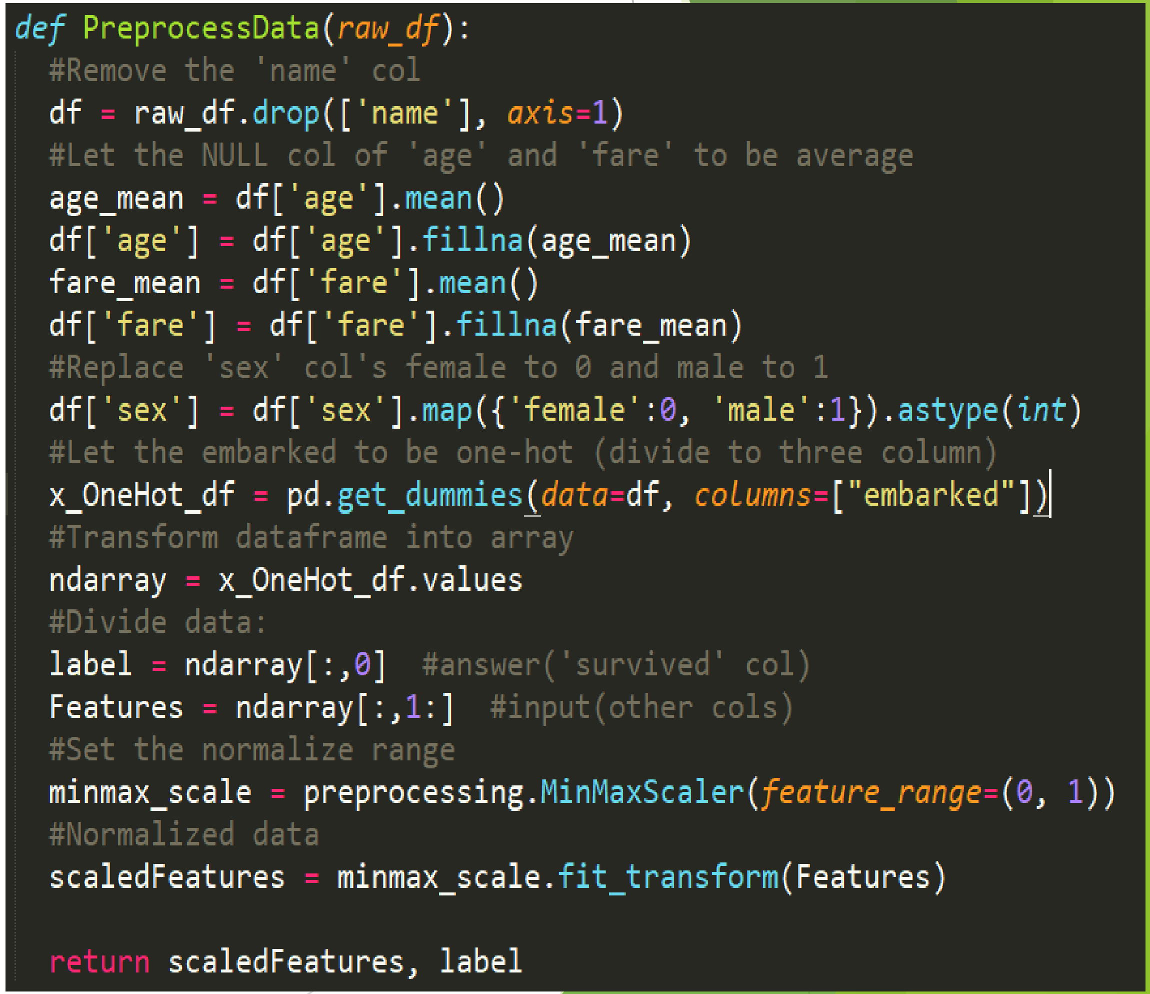
id	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1	1	Allen, Miss. Elisabeth Walton	female	29	0	0	24160	211.3375	B5	S	2		St Louis, MO
1	1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	11		Montreal, PQ / Chesterville, ON
2	1	1	Anderson, Mr. Harry	male	48	0	0	19952	26.5500	E12	S	3		New York, NY
3	1	1	Andrews, Miss. Kornelia Theodosia	female	63	1	0	13502	77.9583	D7	S	10		Hudson, NY
4	1	1	Appleton, Mrs. Edward Dale (Charlotte	female	53	2	0	11769	51.4792	C101	S	D		Bayside, Queens, NY
5	1	1	Astor, Mrs. John Jacob (Madeleine Ta	female	18	1	0	PC 17757	227.5250	C62 C64	C	4		New York, NY
6	1	1	Aubart, Mme. Leontine Pauline	female	24	0	0	PC 17477	69.3000	B35	C	9		Paris, France
7	1	1	Barber, Miss. Ellen "Nellie"	female	26	0	0	19877	78.8500		S	6		
8	1	1	Barkworth, Mr. Algernon Henry Wilso	male	80	0	0	27042	30.0000	A23	S	B		Hessle, Yorks
9	1	1	Baxter, Mrs. James (Helene DeLaudeni	female	50	0	1	PC 17558	247.5208	B58 B60	C	6		Montreal, PQ
10	1	1	Bazzani, Miss. Albina	female	32	0	0	11813	76.2917	D15	C	8		
11	1	1	Beckwith, Mr. Richard Leonard	male	37	1	1	11751	52.5542	D35	S	5		New York, NY
12	1	1	Beckwith, Mrs. Richard Leonard (Sallie	female	47	1	1	11751	52.5542	D35	S	5		New York, NY
13	1	1	Behr, Mr. Karl Howell	male	26	0	0	111369	30.0000	C148	C	5		New York, NY
14	1	1	Bidois, Miss. Rosalie	female	42	0	0	PC 17757	227.5250		C	4		
15	1	1	Bird, Miss. Ellen	female	29	0	0	PC 17483	221.7792	C97	S	8		
16	1	1	Bishop, Mr. Dickinson H	male	25	1	0	11967	91.0792	B49	C	7		Dowagiac, MI
17	1	1	Bishop, Mrs. Dickinson H (Helen Walt	female	19	1	0	11967	91.0792	B49	C	7		Dowagiac, MI

Data Preprocessing

- 首先須import pandas as pd
- Pd.read_excel(路徑)讀取training data
- 選取Feature欄位，這裡我選取了:
  - ['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare', 'embarked']
- 資料前處理:
  - Age和Fare欄位有些為Null值，所以我取平均值填入Null的地方。
  - 將Male及Female轉換成用0以及1來表示。
  - Embarked因為共有三個登船港口，我改用One-Hot Encoding來表示也就是將三個港口用[0,0,1],[0,1,0],[1,0,0]表示。
  - Label就是Survived，剩下的即為自己選取的Features
  - 將Features做Normalized:0-1之間

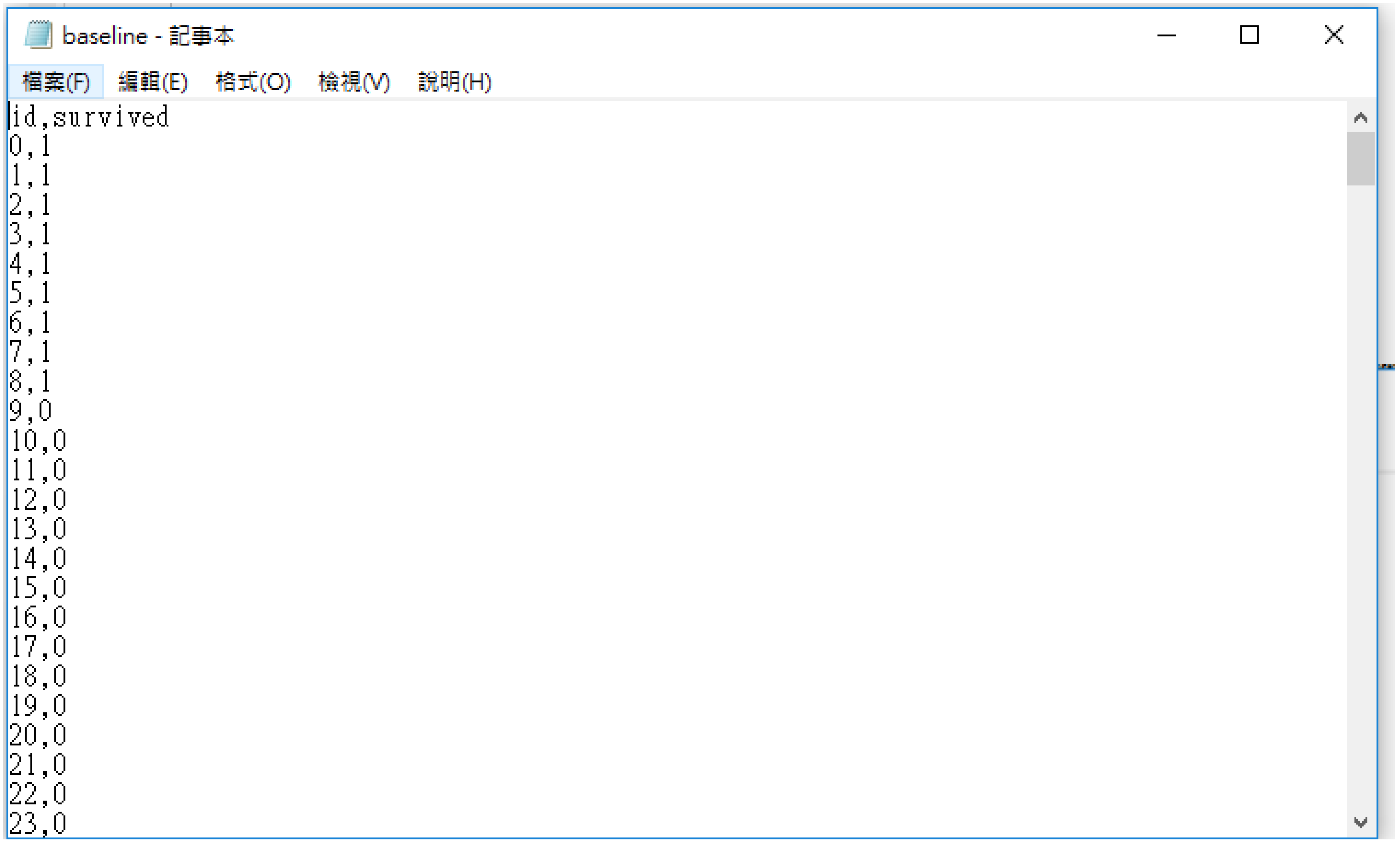


```
all_df = pd.read_excel(path_training)
#take the col we need
all_df = all_df[cols]
```



```
def PreprocessData(raw_df):
    #Remove the 'name' col
    df = raw_df.drop(['name'], axis=1)
    #Let the NULL col of 'age' and 'fare' to be average
    age_mean = df['age'].mean()
    df['age'] = df['age'].fillna(age_mean)
    fare_mean = df['fare'].mean()
    df['fare'] = df['fare'].fillna(fare_mean)
    #Replace 'sex' col's female to 0 and male to 1
    df['sex'] = df['sex'].map({'female':0, 'male':1}).astype(int)
    #Let the embarked to be one-hot (divide to three column)
    x_OneHot_df = pd.get_dummies(data=df, columns=["embarked"])
    #Transform dataframe into array
    ndarray = x_OneHot_df.values
    #Divide data:
    label = ndarray[:,0]  #answer('survived' col)
    Features = ndarray[:,1:]  #input(other cols)
    #Set the normalize range
    minmax_scale = preprocessing.MinMaxScaler(feature_range=(0, 1))
    #Normalized data
    scaledFeatures = minmax_scale.fit_transform(Features)

    return scaledFeatures, label
```

上傳格式



```
id,survived
0,1
1,1
2,1
3,1
4,1
5,1
6,1
7,1
8,1
9,0
10,0
11,0
12,0
13,0
14,0
15,0
16,0
17,0
18,0
19,0
20,0
21,0
22,0
23,0
```

Homework評分方式

- 4/10 24:00前上傳Kaggle，Kaggle名稱請用學號_姓名 (M123456_王大明)
- Moodle需上傳程式碼
- 需上傳心得報告(使用各種層數、Neron數及參數之心得) – 佔5分
- 參加Kaggle競賽並評比分數 – 佔10分
  - 90%以上 - 10分
  - 80%以上 - 9分
  - 70%以上 - 8分
  - Baseline(67.637%)以上 – 7分
- Kaggle: https://www.kaggle.com/t/20b713d4c7c046a2ab36613142decc49

Keras及Pandas官方文檔

- Keras官方文檔:https://keras.io/
- Pandas官方文檔: http://pandas.pydata.org/pandas-docs/stable/
- 如有對用法有疑慮的同學，可以至官方文檔查詢
- 若再無法解決請寄信給助教或至RB-308-3
- 信箱:M10615090@mail.ntust.edu.tw