物聯網應用與資料分析 Assignment5 - Titanic Predict with Keras

姓名：吳嘉偉　學號：5105056013　日期：2018/6/2

1　目標

利用 Kaggle 提供的 Titanic 資料集，透過 Keras 分析乘客是否生還。

2　Prepare Data

2.1　Create Python File

建立一個 preprocess.py 的檔案，把資料集預處理的程式碼邏輯寫在這個地方。

2.2　Get CSV File Data

下載網站上面提供的 train.csv, test.csv。利用 python pandas 套件把這兩個檔案讀出來。

```
import pandas as pd

def getData():
  train = pd.read_csv(’train.csv’)
  test = pd.read_csv(’test.csv’)
  return train, test
```

2.3 Fill Missing Data

因為我們預計要拿 Age, Sex, Pclass 這三個欄位來做預測，發現 Age 是有缺值的。所以利用 Age 這欄位的平均值填在缺值的欄位上

```
def fillMissing(train, test):
  train.Age = train.Age.fillna(train.Age.mean())
  test.Age = test.Age.fillna(test.Age.mean())
  return train, test
```

2.4 One-Hot Encoding

因為 Sex, Pclass 這兩個欄位是離散的資料，為了讓分析更容易，我們使用 One-Hot Encoding 把這兩個欄位作轉換

```
def oneHotEncoding(train, test):
  modify_train = pd.get_dummies(train,
                                columns=['Sex', 'Pclass'])
  modify_test = pd.get_dummies(test,
                               columns=['Sex', 'Pclass'])
  return modify_train, modify_test
```

2.5 Distinction Data

在這邊我們把 Train, Test 分開

```
def prepareXY(train, test):
  x_train = train[['Age', 'Sex_female', 'Sex_male',
                 'Pclass_1', 'Pclass_2', 'Pclass_3']]
  y_train = train['Survived']
  x_test = test[['Age', 'Sex_female', 'Sex_male',
                 'Pclass_1', 'Pclass_2', 'Pclass_3']]
  test['Survived'] = 0
  y_test = test['Survived']
```

```
    return x_train, y_train, x_test, y_test
```

3 Install Environment

我們使用可以使用 PyCharm 軟體安裝 Keras，或利用 terminal 的 pip 安裝詳細步驟請參考Install Python Packages

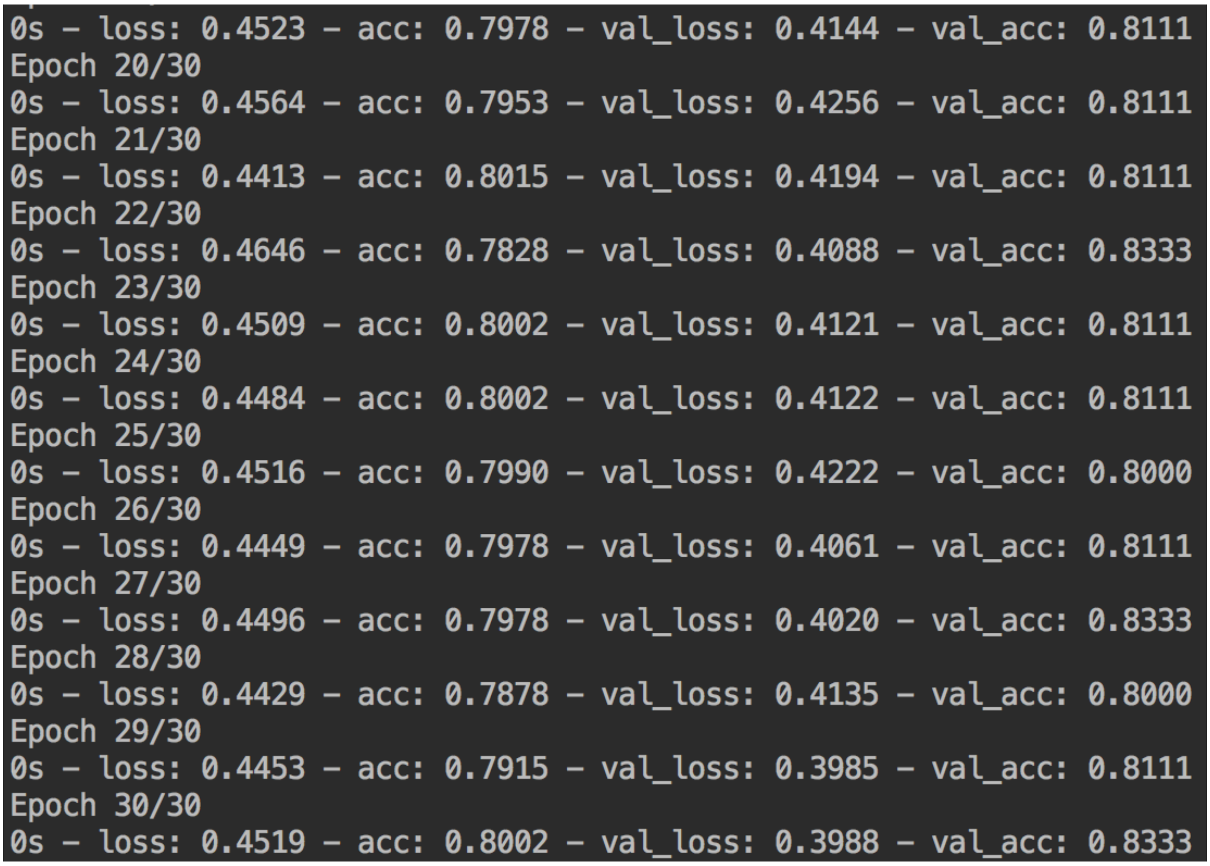
4 Perdict Model

建立一個 deeplearnmodel.py 的檔案，利用 Keras 建立預測模型

```
from keras.models import Sequential
from keras.layers import Dense, Dropout
import numpy as np

def deepNN(x_train, y_train, x_test, y_test):
    model = Sequential()
    model.add(Dense(units=40, input_dim=6,
                    kernel_initializer=’uniform’,
                    activation=’relu’))
    model.add(Dropout(0.2))
    model.add(Dense(units=30,
                    kernel_initializer=’uniform’,
                    activation=’relu’))
    model.add(Dropout(0.3))
    model.add(Dense(units=1,
                    kernel_initializer=’uniform’,
                    activation=’sigmoid’))
    model.compile(loss=’binary_crossentropy’,
                    optimizer=’adam’,
                    metrics=[’accuracy’])
    train_history = model.fit(x=np.array(x_train),
                    y=np.array(y_train),
                    validation_split=0.1,
                    epochs=30,
                    batch_size=30,
                    verbose=2)
```

下圖為執行時的預測準確率 Log



```
0s - loss: 0.4523 - acc: 0.7978 - val_loss: 0.4144 - val_acc: 0.8111
Epoch 20/30
0s - loss: 0.4564 - acc: 0.7953 - val_loss: 0.4256 - val_acc: 0.8111
Epoch 21/30
0s - loss: 0.4413 - acc: 0.8015 - val_loss: 0.4194 - val_acc: 0.8111
Epoch 22/30
0s - loss: 0.4646 - acc: 0.7828 - val_loss: 0.4088 - val_acc: 0.8333
Epoch 23/30
0s - loss: 0.4509 - acc: 0.8002 - val_loss: 0.4121 - val_acc: 0.8111
Epoch 24/30
0s - loss: 0.4484 - acc: 0.8002 - val_loss: 0.4122 - val_acc: 0.8111
Epoch 25/30
0s - loss: 0.4516 - acc: 0.7990 - val_loss: 0.4222 - val_acc: 0.8000
Epoch 26/30
0s - loss: 0.4449 - acc: 0.7978 - val_loss: 0.4061 - val_acc: 0.8111
Epoch 27/30
0s - loss: 0.4496 - acc: 0.7978 - val_loss: 0.4020 - val_acc: 0.8333
Epoch 28/30
0s - loss: 0.4429 - acc: 0.7878 - val_loss: 0.4135 - val_acc: 0.8000
Epoch 29/30
0s - loss: 0.4453 - acc: 0.7915 - val_loss: 0.3985 - val_acc: 0.8111
Epoch 30/30
0s - loss: 0.4519 - acc: 0.8002 - val_loss: 0.3988 - val_acc: 0.8333
```

5 Evaluation

最後，把準確率畫出來，並計算成效

```
import matplotlib.pyplot as plt

def get_evaluate():
  scores = model.evaluate(
              x=np.array(x_test),
              y=np.array(y_test))

def show_train_history(train_history, train, val):
  plt.plot(train_history.history[train])
  plt.plot(train_history.history[validation])
  plt.title('Train␣History')
  plt.ylabel('Train')
  plt.xlabel('Epoch')
  plt.legend(['train', 'validation'],
                    loc ='center␣right')
  plt.show()
```

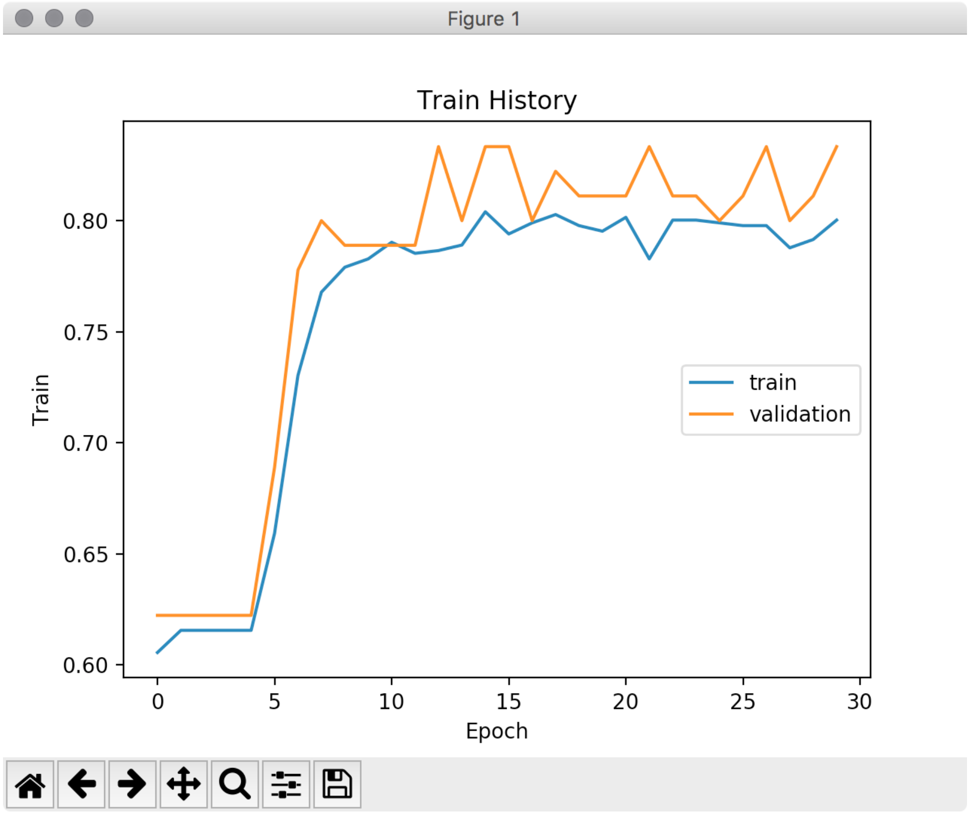


Figure 1
Train History
Train
0.60
0.65
0.70
0.75
0.80
Epoch
0
5
10
15
20
25
30
train
validation