碩管一甲 M09218001 周彥廷

高等類神經網路 作業 0311 課輔練習 2

作答說明：

請以你準備的資料集進行倒傳遞類神經網路的建模

敘述以下幾點：

1.資料來源、簡介、變數介紹

2.訓練與測試資料筆數

3.你設計的最佳網路架構為何

4.測試資料的混淆矩陣、正確率為多少

1.資料來源、簡介、變數介紹

資料來源 UCI Machine learning

https://archive.ics.uci.edu/ml/datasets/Heart+failure+clinical+records

簡介：

Survival analysis of heart failure patients: A case study

此數據集收集 299 例心臟熱衰竭病患病歷，每個患者具有 13 種臨床特徵。

相關 paper 連結：

Survival analysis of heart failure patients: A case study (plos.org)

https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-020-1023-5

變數介紹：

1. age: age of the patient (years) 病患年齡

2.anaemia: decrease of red blood cells or hemoglobin (boolean) 貧血

3.high blood pressure: if the patient has hypertension (boolean) 高血壓

4.creatinine phosphokinase (CPK): level of the CPK enzyme in the blood (mcg/L) 肌酐磷酸激酶

5.diabetes: if the patient has diabetes (boolean) 糖尿病

6.ejection fraction: percentage of blood leaving the heart at each contraction (percentage) 心臟血液收縮百分比

7.platelets: platelets in the blood (kiloplatelets/mL) 血小板

8.sex: woman or man (binary) 性別(男/女)

9.serum creatinine: level of serum creatinine in the blood (mg/dL) 血清肌酐

10.serum sodium: level of serum sodium in the blood (mEq/L) 血清鈉

11.smoking: if the patient smokes or not (boolean) 是否有抽菸

12.time: follow-up period (days) 時間(天數)

13.death event: if the patient deceased during the follow-up period (boolean) 死亡

2.訓練與測試資料筆數

由於整體資料量不多，訓練/測試比例為 0.1 和 0.2，比數如下：

訓練資料(30/60)

測試資料(269/239)

3.你設計的最佳網路架構為何

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_1018 (Dense)           (None, 50)                600
_________________________________________________________________
leaky_re_lu_581 (LeakyReLU)  (None, 50)                0
_________________________________________________________________
batch_normalization_118 (Bat (None, 50)                200
_________________________________________________________________
dropout_146 (Dropout)        (None, 50)                0
_________________________________________________________________
dense_1019 (Dense)           (None, 50)                2550
_________________________________________________________________
leaky_re_lu_582 (LeakyReLU)  (None, 50)                0
_________________________________________________________________
batch_normalization_119 (Bat (None, 50)                200
_________________________________________________________________
dropout_147 (Dropout)        (None, 50)                0
_________________________________________________________________
dense_1020 (Dense)           (None, 50)                2550
_________________________________________________________________
leaky_re_lu_583 (LeakyReLU)  (None, 50)                0
_________________________________________________________________
dropout_148 (Dropout)        (None, 50)                0
_________________________________________________________________
dense_1021 (Dense)           (None, 50)                2550
_________________________________________________________________
leaky_re_lu_584 (LeakyReLU)  (None, 50)                0
_________________________________________________________________
dropout_149 (Dropout)        (None, 50)                0
_________________________________________________________________
dense_1022 (Dense)           (None, 2)                 102
=================================================================
Total params: 8,752
Trainable params: 8,552
Non-trainable params: 200
```

使用 Leaky Relu 作為隱藏層的活化函數，學習率(lr = 0.001)，批量標準化 batch_normalization 優化初始權重。

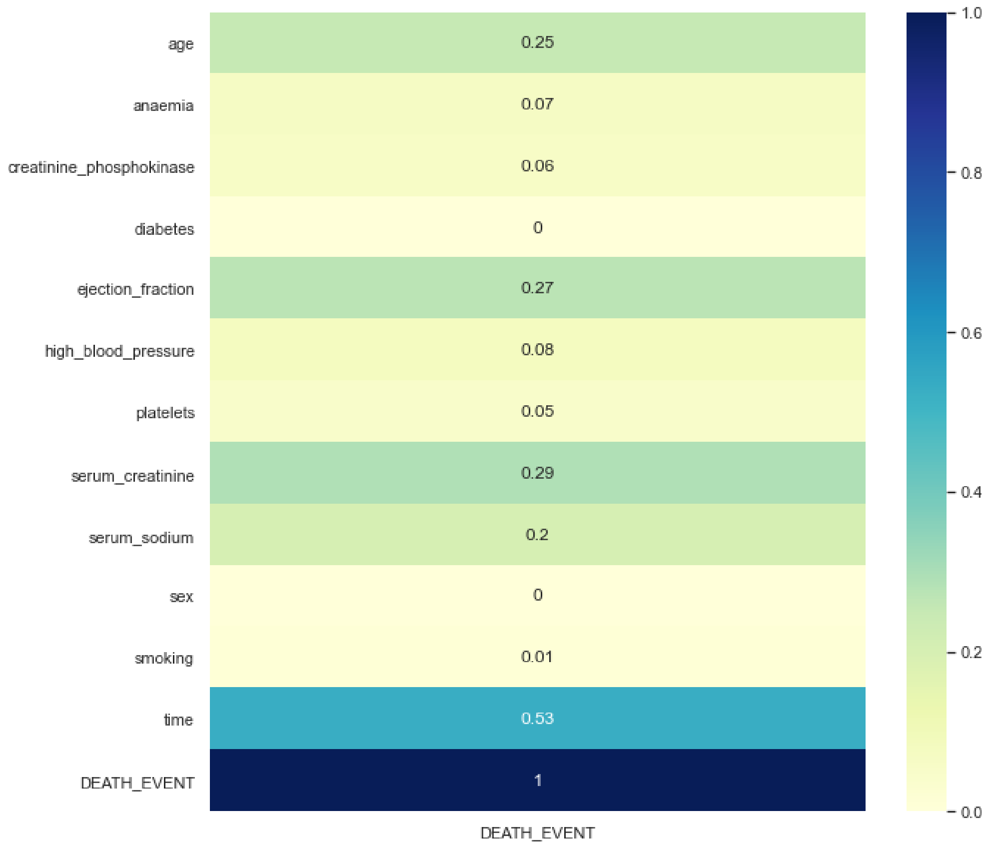
根據 12 種特徵對於是否死亡(0/1)問題為二元分類問題。

執行程式碼如下：

```
#0.載入套件
import numpy as np
import pandas as pd
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.utils import np_utils
from sklearn.preprocessing import StandardScaler,MinMaxScaler
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
from keras import layers
from keras import losses
from keras.layers import LeakyReLU
from keras import regularizers
from sklearn.metrics import classification_report, confusion_matrix
import tensorflow as tf
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score,mean_absolute_error
# 1.讀入訓練資料集
df = pd.read_csv
('C:/Users/MCUT/Desktop/heart_failure_clinical_records_dataset.csv',
                                                            encoding='utf-8')
# 查看資料欄位
df. describe()

# 繪製關聯矩陣
correlation_matrix = df.corr().round(2).loc[:,['DEATH_EVENT']].abs()
sns.heatmap(data=correlation_matrix, annot = True,cmap='YlGnBu')
```

由關係矩陣可以發現到欄位 time(0.53)、serum creatinine (0.29)、ejection fraction (0.27)、age (0.25) 與 serum sodium (0.2) 與 DEATH_EVENT 的關聯性較高。




```
# 取出特徵欄位 X
X = df.iloc[:,1:12]
X.head()
# 死亡事件為 Y
Y = df['DEATH_EVENT']
Y.head()

#切分訓練與測試數據集
X_train, X_test, y_train, y_test = train_test_split(X.values, Y.values,
test_size=0.2,#0.1 random_state=42)

#數據預處理與讀熱編碼(StandardScale and onehot)
print(f'預處理之前 x:{X_train[0]},y:{y_train[0]}')
ss_x = StandardScaler().fit(X_train)
X_train,X_test  = ss_x.transform(X_train),ss_x.transform(X_test)
y_train,y_test =
tf.keras.utils.to_categorical(y_train),tf.keras.utils.to_categorical(y_test)
print(f'預處理之後 x:{X_train[0]},y:{y_train[0]}')
```

```
#訓練模型
model = Sequential()
model.add(Dense(50, input_dim=11, kernel_initializer = 'lecun_normal'))

model.add(LeakyReLU(alpha=0.01))
model.add(layers.BatchNormalization())
model.add(Dropout(0.4))

model.add(Dense(50))
model.add(LeakyReLU(alpha=0.01))
model.add(layers.BatchNormalization())
model.add(Dropout(0.4))

model.add(Dense(50))
model.add(LeakyReLU(alpha=0.01))
model.add(Dropout(0.4))
model.add(Dense(50))
model.add(LeakyReLU(alpha=0.01))
model.add(Dropout(0.4))
model.add(Dense(units=2, activation = 'softmax'))

ADAM    = keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999,
epsilon=None, decay=0.0, amsgrad=False)
#RMSprop = keras.optimizers.RMSprop(lr=0.01, rho=0.9, epsilon=None, decay=0.0)

model.compile(loss='binary_crossentropy', optimizer = ADAM, metrics=['accuracy'])
model.summary()

#  訓練模式
model.fit(X_train, y_train, epochs=30, batch_size=4) #batch_size

#  評估模式
scores = model.evaluate(X_test, y_test)
print("\nAccuracy: %.2f%%" % (scores[1]*100))

# confusion matrix
pred_model = model.predict(X_test)
pred = np.argmax(pred_model, axis=1)
```

pred_1 = np_utils.to_categorical(pred)
print(confusion_matrix(y_test.argmax(axis=1), pred_model.argmax(axis=1)))
print(classification_report(y_test, pred_1))

4.測試資料的混淆矩陣、正確率為多少

```
In [549]: print(confusion_matrix(y_test.argmax(axis=1), pred_model.argmax(axis=1)))
[[32  3]
 [13 12]]

In [550]: print(classification_report(y_test, pred_1))
              precision    recall  f1-score   support

           0       0.71      0.91      0.80        35
           1       0.80      0.48      0.60        25

   micro avg       0.73      0.73      0.73        60
   macro avg       0.76      0.70      0.70        60
weighted avg       0.75      0.73      0.72        60
 samples avg       0.73      0.73      0.73        60
```

結論：對死亡事件(Y)預測正確率經過反覆測試與調整超參數達到 73%，並由混淆矩陣觀察到精度(precision)與召回率(recall)。

PART II. 假設這是一個迴歸問題針對 DEATH EVENT 則使用 R^2 和 *MSE* 作為評估模型效能的指標。

$$R^2 = 1 - \frac{\sum_{1=1}^{N} (y_i - \hat{y}_i)^2}{\sum_{1=1}^{N} (y_i - \bar{y})^2}$$

$$MSE = \frac{1}{n} \Sigma \underbrace{\left(y - \hat{y} \right)^2}_{\text{The square of the difference between actual and predicted}}$$

網路架構與程式碼同上，將 loss 和 評估函數 更改為 'mse'

model.compile(loss='mean_squared_error', optimizer = ADAM, metrics=['mse'])

model.summary()

```
Epoch 21/30
239/239 [==============================] - 1s 5ms/step - loss: 0.1967 - mean_squared_error: 0.1967
Epoch 22/30
239/239 [==============================] - 1s 4ms/step - loss: 0.1810 - mean_squared_error: 0.1810
Epoch 23/30
239/239 [==============================] - 1s 4ms/step - loss: 0.1901 - mean_squared_error: 0.1901
Epoch 24/30
239/239 [==============================] - 1s 4ms/step - loss: 0.1686 - mean_squared_error: 0.1686
Epoch 25/30
239/239 [==============================] - 1s 4ms/step - loss: 0.1865 - mean_squared_error: 0.1865
Epoch 26/30
239/239 [==============================] - 1s 4ms/step - loss: 0.1760 - mean_squared_error: 0.1760
Epoch 27/30
239/239 [==============================] - 1s 4ms/step - loss: 0.1747 - mean_squared_error: 0.1747
Epoch 28/30
239/239 [==============================] - 1s 4ms/step - loss: 0.1685 - mean_squared_error: 0.1685
Epoch 29/30
239/239 [==============================] - 1s 4ms/step - loss: 0.1732 - mean_squared_error: 0.1732
Epoch 30/30
239/239 [==============================] - 1s 4ms/step - loss: 0.1859 - mean_squared_error: 0.1859
Out[562]: <keras.callbacks.History at 0x2145c268d30>
```

結果呈現：

```
In [563]: y_pred = model.predict(X_test)
     ...: y_true = y_test
     ...: print('r2:',r2_score(y_true,y_pred))
     ...: print('mse:',mean_absolute_error(y_true,y_pred))
r2: 0.2040615671345824
mse: 0.38733578
```