國立雲林科技大學

資訊管理所

機器學習專案報告

學生/學號：

張　靖 M10823001

陳逢麒 M10823028

平靖翔 M10823024

古力維 M10823040

中華民國 109 年 4 月

摘要

本研究針對人口普查資料進行年收入是否大於五萬美金以及每周工作時數之預測，透過演算法訓練資料進行測試資料的驗證準確度與降低誤差。隨著大量資料快速增長與累積以及機器學習與深度學習蓬勃發展、大數據分析已經成為學術界及各行各業中的關注與學習。從海量的資料中篩選出有用的資訊，並進行資料的預測即是機器學習領域方面積極所做的研究。本研究將使用 Python 撰寫類神經網路(Artificial Neural Network)進行程式訓練模型，在演算法的訓練之下在年收入是否大於五萬美元的預測上以準確度作為預測績效達到 0.85 以及損失函數 0.32，在每周工作時數預測以均方根誤差績效評估達到 10.82。

關鍵字：機器學習、類神經網路、正規化

目錄

表目錄

圖目錄

一、 緒論

1.1 動機

根據 US Median Real Income by year(2016)資料指出 1994 年美國平均收入為 51,710 美元。本研究為了瞭解當時美國的就業情況以及薪資結構，採用 UCI-Adult Data Set(1996)所提供 48442 筆關於美國人的薪資狀況(如:性別、教育程度、職位等)。透過此資料集來判別年薪是否大於或小於 50,000 美元，以及預測工作時數。近年來，由於硬體設備的進步，類神經網路模型的運算得以被實現，人工智慧的再度崛起，使得各種行業紛紛投入其中，主要藉此減少人力的浪費以及提升產品生產的效率以及獲取更精確的資訊。為此，本研究採用類神經網路模型來預測美國平均收入以及工作時數。

1.2 目的

本研究主要以機器學習為基礎，透過 Python 撰寫類神經網路之程式，以此進行模型訓練，這項研究針對美國薪資結構以及每周平均上班時數進行預測。並透過類神經網路調整隱藏層層數、節點數、激活函數、優化器、loss 函數以達到此資料最佳的預測績效。

二、研究方法

2.1 程式架構-每周工作時數

【ANN by adult.data 】

```
import numpy as np
import pandas as pd
from sklearn import preprocessing

#讀取資料並做資料前處理將特殊字元處理掉
train_data =pd.read_csv('D:/Python/ML_report1/adult.data',sep=" ", header=None)
train_data = train_data.replace({'\$': '', ',': '','\.':'','<=50K':'1','>50K':'0'}, regex=True) #砍掉換行&逗號
train_data.columns = ['age','workclass','fnlwgt','education','education-num','marital-status','occupation','relationship','race','sex','capital-gain','capital-loss','hours-per-week','native-country','label']

test_data =pd.read_csv('D:/Python/ML_report1/adult.test',sep=" ", header=None)
test_data = test_data.replace({'\$': '', ',': '','<=50K.':'1','>50K.':'0'}, regex=True) #砍掉換行&逗號
test_data.columns = ['age','workclass','fnlwgt','education','education-num','marital-status','occupation','relationship','race','sex','capital-gain','capital-loss','hours-per-week','native-country','label']

#合併 train data 和 test data
data_1 = train_data
data_1=data_1.append(test_data)
#data_1

#將缺值欄位補上平均值
data_1.replace('?', np.nan, inplace=True)
data_1=data_1.fillna(data_1.mean())
data_1 = data_1.apply(lambda x:x.fillna(x.value_counts().index[0]))

#將資料轉成 int
data_1[['age','fnlwgt','education-num','capital-gain','capital-loss','hours-per-week','label'] ]=data_1[['age','fnlwgt',
```

```
'education-num','capital-gain','capital-loss','hours-per-week',
'label']].astype(str).astype(int)
#將文字資料(要做 one hot encoding 和 label 切出)
data_cat = data_1[['workclass','education','marital-
status','occupation','relationship','race','sex','native-country']]
data_hours = data_1[['hours-per-week']]

#one hot encoding
data_cat = pd.get_dummies(data_cat)


#將原始資料合併 one hot encoding 後的資料
newdata1 = data_1.drop(['workclass','education','marital-
status','occupation','relationship','race','sex','native-country'],axis=1)
newdata_merge = pd.concat([newdata1,data_cat],axis=1).reindex(data_1.index)

#newdata_merge

#將資料正規化
min_max_scaler = preprocessing.MinMaxScaler()
np_scaled = min_max_scaler.fit_transform(newdata_merge)
data_norm= pd.DataFrame(np_scaled, columns = newdata_merge.columns)
data_norm.head()

#test_data=newdata_merge.iloc[32561:]
#test_data

#train_data=newdata_merge.iloc[:32561]
#train_data

test_data1=data_norm.iloc[32561:] #分解 train data 和 test data
test_data1

train_data1=data_norm.iloc[:32561] #分解 train data 和 test data
train_data1

train_true=data_hours.iloc[:32561] #擷取 traindata label
#train_true
```

```
test_true=data_hours.iloc[32561:] #擷取 testdata label
#test_true

from keras import layers, optimizers, models
from sklearn.preprocessing import LabelEncoder

X_train1 = train_data1.drop('hours-per-week', axis=1)
y_train1 = train_true

X_test1 = test_data1.drop('hours-per-week', axis=1)
y_test1 = test_true

from sklearn.metrics import mean_squared_error
from keras.models import Sequential
from keras.layers.core import Dense , Dropout
from keras import regularizers
from keras.callbacks import EarlyStopping

#dense 指最普通的全連接層型態
#dropout 減少節點 通常設 0.2~0.5 最高不會超過 0.5
#2 層(最後一層 ouput 不算)



early_stopping = EarlyStopping(monitor='val_loss',
patience=200,restore_best_weights=True)
model = models.Sequential()
model.add(layers.Dense(16, input_shape=(X_train1.shape[1],), activation="relu"))
model.add(layers.Dropout(0.30)) #d 砍上一層的節點
model.add(layers.Dense(8, activation="relu"))
model.add(layers.Dropout(0.30))
model.add(layers.Dense(1))

model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])

history = model.fit(X_train1, y_train1, validation_split=0.2, epochs=1000000,
batch_size=128,callbacks=[early_stopping])
```

```
loss, accuracy = model.evaluate(X_test1, y_test1)
print("Test Acc : " + str(accuracy))
print("Test Loss : " + str(loss))
y_pred1 = model.predict(X_test1)
print('MSE 為：',mean_squared_error(y_test1,y_pred1))
#print('MSE 為(直接计算)：',np.mean((y_test-y_pred)**2))
print('RMSE 為：',np.sqrt(mean_squared_error(y_test1,y_pred1)))
```

2.2 程式架構-年收入預測

【ANN by adult.data 】

```
import numpy as np
import pandas as pd
from sklearn import preprocessing

train_data =pd.read_csv('D:/Python/ML_report1/adult.data',sep=" ", header=None)
train_data = train_data.replace({'\$': '', ',': '','\.':'','<=50K':'1','>50K':'0'}, regex=True) #砍掉換行&逗號
train_data.columns = ['age','workclass','fnlwgt','education','education-num','marital-status','occupation','relationship','race','sex','capital-gain','capital-loss','hours-per-week','native-country','label']

test_data =pd.read_csv('D:/Python/ML_report1/adult.test',sep=" ", header=None)
test_data = test_data.replace({'\$': '', ',': '','<=50K.':'1','>50K.':'0'}, regex=True) #砍掉換行&逗號
test_data.columns = ['age','workclass','fnlwgt','education','education-num','marital-status','occupation','relationship','race','sex','capital-gain','capital-loss','hours-per-week','native-country','label']

data_1 = train_data
data_1=data_1.append(test_data)

data_1.replace('?', np.nan, inplace=True)
data_1=data_1.fillna(data_1.mean())
data_1 = data_1.apply(lambda x:x.fillna(x.value_counts().index[0]))
```

```
data_1[['age','fnlwgt','education-num','capital-gain','capital-loss','hours-per-
week','label'] ]=data_1[['age','fnlwgt','education-num','capital-gain','capital-
loss','hours-per-week','label']].astype(str).astype(int)
data_cat = data_1[['workclass','education','marital-
status','occupation','relationship','race','sex','native-country']]

data_cat = pd.get_dummies(data_cat)

newdata1 = data_1.drop(['workclass','education','marital-
status','occupation','relationship','race','sex','native-country'],axis=1)
newdata_merge = pd.concat([newdata1,data_cat],axis=1).reindex(data_1.index)

newdata_merge

min_max_scaler = preprocessing.MinMaxScaler()
np_scaled = min_max_scaler.fit_transform(newdata_merge[:len(newdata_merge)-1])
data_norm= pd.DataFrame(np_scaled, columns = newdata_merge.columns)
data_norm.head()

test_data=data_norm.iloc[32561:]
test_data

train_data=data_norm.iloc[:32561]
train_data

from keras import layers, optimizers, models

X_train = train_data.drop('label', axis=1)
y_train = train_data['label']

X_test = test_data.drop('label', axis=1)
y_test = test_data['label']


from keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(monitor='val_loss',
patience=200,restore_best_weights=True)
model = models.Sequential()
```

```
model.add(layers.Dense(64, input_shape=(X_train.shape[1],), activation="sigmoid"))
model.add(layers.Dropout(0.30))
model.add(layers.Dense(32, activation="sigmoid"))
model.add(layers.Dropout(0.30))
model.add(layers.Dense(16, activation="sigmoid"))
model.add(layers.Dropout(0.30))
model.add(layers.Dense(8, activation="sigmoid"))
model.add(layers.Dropout(0.30))
model.add(layers.Dense(1, activation="sigmoid"))

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

history = model.fit(X_train, y_train, validation_split=0.2, epochs=1000000,
batch_size=128,callbacks=[early_stopping])

loss, accuracy = model.evaluate(X_test, y_test)
print("Test Acc : " + str(accuracy))
print("Test Loss : " + str(loss))
```

三、實驗

3.1 資料集

表 1

1996 美國人口普查資料集

數據及特徵	多變量	實例數	48842	區域	社會
屬性特徵	分類整數	屬性數量	14	捐贈日期	1996-05-01
相關任務	分類	缺少缺失值	是		

表 2

屬性資料

屬性名稱	屬性類別	類別種類	屬性說明
Age	integer	continuous	年齡
Workclass	categorical	Private Self-emp-not-in Self-emp-inc Federal-gov Local-gov State-gov Without-pay Never-worked	工作類別
fnlwgt	integer	continuous	fnlwgt
education	categorical	Bachelors Some-college 11th, HS-grad Prof-school Assoc-acdm Assoc-voc 9th ,7th-8th ,12th Masters 1st-4th 10th Doctorate 5th-6th Preschool.	教育程度

(續下頁)

屬性名稱	屬性類別	類別種類	屬性說明
Education-non	categorical	continuous	教育人數
Marital-status	categorical	Married-civ-spouse, Divorced, Never-married, Separated,Widowed, Married-spouse-absent, Married-AF-spouse.	婚姻狀況
occupation	categorical	Tech-support,Craft-repair,Other-service, Sales,Exec-managerial, Prof-specialty,Handlers-cleaners,Machine-op-inspct,Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.	職業
relationship	categorical	Wife Own-child Husband Not-in-family Other-relative Unmarried	關係
race	categorical	White, Asian-Pac-Islander Amer-Indian-Eskimo Other Black	種族
sex	categorical	Female Male	性別
Capital-gain	categorical	continuous	資本收益
Capital-loss	categorical	continuous	資本損失
hours-per-week	categorical	continuous	每周時數

(續下頁)

屬性名稱	屬性類別	類別種類	屬性說明
native-country	categorical	United-States, Cambodia, England,Puerto-Rico, Canada,Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica,Vietnam, Mexico,Portugal, Ireland, France,Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary,Guatemala, Nicaragua,Scotland, Thailand, Yugoslavia, El-Salvador,Trinadad&Tobago, Peru,Hong,Holand-Netherlands.	國家

3.2 前置處理

1. 每周工作時數

原始資料分為訓練和測試內容包含文字資料、特殊符號和空值，將兩個資料及合併後，將特殊符號刪除後並將空值全數補平均值，最後將文字資料轉為數值資料再將，前處理完後再將資料切分為訓練集和測試集以及將要預測的 label 提取出來後續做績效評估用。

```
#讀取資料並做資料前處理將特殊字元處理掉
train_data =pd.read_csv('adult.data',sep=" ", header=None)
train_data = train_data.replace({'\$': '', ',': '','\.':'','<=50K':'1','>50K':'0'}, regex=True) #砍掉換行&逗號
train_data.columns = ['age','workclass','fnlwgt','education','education-num','marital-status','occupation','relationship',
                      'race','sex','capital-gain','capital-loss','hours-per-week','native-country','label']

test_data =pd.read_csv('adult.test',sep=" ", header=None)
test_data = test_data.replace({'\$': '', ',': '','<=50K.':'1','>50K.':'0'}, regex=True) #砍掉換行&逗號
test_data.columns = ['age','workclass','fnlwgt','education','education-num','marital-status','occupation','relationship',
                     'race','sex','capital-gain','capital-loss','hours-per-week','native-country','label']
```

```
#合併train data和test data
data_1 = train_data
data_1=data_1.append(test_data)
#data_1
```

```
#將缺值欄位補上平均值
data_1.replace('?', np.nan, inplace=True)
data_1=data_1.fillna(data_1.mean())
data_1 = data_1.apply(lambda x:x.fillna(x.value_counts().index[0]))
#data_1
```

```
#將資料轉成int
data_1[['age','fnlwgt','education-num','capital-gain',
        'capital-loss','hours-per-week','label'] ] = data_1[['age','fnlwgt','education-num',
                                                             'capital-gain','capital-loss','hours-per-week',
                                                             'label']].astype(str).astype(int)
#將文字資料(要做one hot encoding)和label切出
data_cat = data_1[['workclass','education','marital-status','occupation','relationship','race','sex','native-country']]
data_hours = data_1[['hours-per-week']]
```

```
#one hot encoding
data_cat = pd.get_dummies(data_cat)
#data_cat
```

```
#將原始資料合併one hot encoding後的資料
newdata1 = data_1.drop(['workclass','education','marital-status','occupation',
                        'relationship','race','sex','native-country'],axis=1)
newdata_merge = pd.concat([newdata1,data_cat],axis=1).reindex(data_1.index)
```

```
#將資料正規化
min_max_scaler = preprocessing.MinMaxScaler()
np_scaled = min_max_scaler.fit_transform(newdata_merge)
data_norm= pd.DataFrame(np_scaled, columns = newdata_merge.columns)
data_norm.head()
```

```
#分解train data和test data
test_data1=data_norm.iloc[32561:]
test_data1
```

```
#分解train data和test data
train_data1=data_norm.iloc[:32561]
train_data1
```

```
#擷取traindata label
train_true=data_hours.iloc[:32561]
#train_true
```

```
#擷取testdata label
test_true=data_hours.iloc[32561:]
#test_true
```

圖 1 每周工作時數資料前處理

2. 年收入預測

原始資料分為訓練和測試內容包含文字資料、特殊符號和空值，將兩個資料及合併後，將特殊符號刪除後並將空值全數補平均值，最後將文字資料轉為數值資料再將，前處理完後再將資料切分為訓練集和測試集。

```
#讀取資料並做資料前處理將特殊字元處理掉
train_data =pd.read_csv('adult.data',sep=" ", header=None)
train_data = train_data.replace({'\$': '', ',': '','\.':'','<=50K':'1','>50K':'0'}, regex=True) #砍掉換行&逗號
train_data.columns = ['age','workclass','fnlwgt','education','education-num','marital-status','occupation','relationship',
                      'race','sex','capital-gain','capital-loss','hours-per-week','native-country','label']

test_data =pd.read_csv('adult.test',sep=" ", header=None)
test_data = test_data.replace({'\$': '', ',': '','<=50K.':'1','>50K.':'0'}, regex=True) #砍掉換行&逗號
test_data.columns = ['age','workclass','fnlwgt','education','education-num','marital-status','occupation','relationship',
                     'race','sex','capital-gain','capital-loss','hours-per-week','native-country','label']
```

```
#合併train data和test data
data_1 = train_data
data_1=data_1.append(test_data)
#data_1
```

```
#將缺值欄位補上平均值
data_1.replace('?', np.nan, inplace=True)
data_1=data_1.fillna(data_1.mean())
data_1 = data_1.apply(lambda x:x.fillna(x.value_counts().index[0]))
#data_1
```

```
#將資料轉成int
data_1[['age','fnlwgt','education-num','capital-gain',
        'capital-loss','hours-per-week','label'] ] = data_1[['age','fnlwgt','education-num',
                                                             'capital-gain','capital-loss','hours-per-week',
                                                             'label']].astype(str).astype(int)
#將文字資料(要做one hot encoding)和label切出
data_cat = data_1[['workclass','education','marital-status','occupation','relationship','race','sex','native-country']]
data_hours = data_1[['hours-per-week']]
```

```
#one hot encoding
data_cat = pd.get_dummies(data_cat)
#data_cat
```

```
#將原始資料合併one hot encoding後的資料
newdata1 = data_1.drop(['workclass','education','marital-status','occupation',
                        'relationship','race','sex','native-country'],axis=1)
newdata_merge = pd.concat([newdata1,data_cat],axis=1).reindex(data_1.index)
```

```
#將資料正規化
min_max_scaler = preprocessing.MinMaxScaler()
np_scaled = min_max_scaler.fit_transform(newdata_merge)
data_norm= pd.DataFrame(np_scaled, columns = newdata_merge.columns)
data_norm.head()
```

```
#one hot encoding
data_cat = pd.get_dummies(data_cat)
#data_cat
```

```
#將原始資料合併one hot encoding後的資料
newdata1 = data_1.drop(['workclass','education','marital-status','occupation',
                        'relationship','race','sex','native-country'],axis=1)
newdata_merge = pd.concat([newdata1,data_cat],axis=1).reindex(data_1.index)
```

```
#將資料正規化
min_max_scaler = preprocessing.MinMaxScaler()
np_scaled = min_max_scaler.fit_transform(newdata_merge)
data_norm= pd.DataFrame(np_scaled, columns = newdata_merge.columns)
data_norm.head()
```

```
#分解train data和test data
test_data=data_norm.iloc[32561:]
#test_data
```

```
#分解train data和test data
train_data=data_norm.iloc[:32561]
#train_data
```

圖 2 年收入預測資料前處理

3.3 實驗設計

1. 每周工作時數

本研究是利用 Anaconda3 Jupyter Notebook(Python version = 3.7)環境進行開發，使用的套件有 sklearn 的 keras、pandas、numpy。對資料進行前處理後，使用類神經網路進行數值預測，而在數值預測中，神經層總共 2 層(輸出層不算)，節點數分別 16 和 8 最後一層輸出層為 1，使用 Dropout 參數設定 0.3、激活函數使用 relu，優化器使用 adam，loss 使用 mean_squared_error，進而獲得績效以及準確度。

```
#訓練模型

#dense指最普通的全連接層型態
#dropout減少節點 通常設0.2~0.5 最高不會超過0.5
#2層(最後一層ouput不算)

from sklearn.metrics import mean_squared_error
from keras.models import Sequential
from keras.layers.core import Dense , Dropout
from keras import regularizers
from keras.callbacks import EarlyStopping

early_stopping = EarlyStopping(monitor='val_loss', patience=200,restore_best_weights=True)
model = models.Sequential()
model.add(layers.Dense(16, input_shape=(X_train1.shape[1],), activation="relu"))
model.add(layers.Dropout(0.30)) #d砍上一層的節點
model.add(layers.Dense(8, activation="relu"))
model.add(layers.Dropout(0.30))
model.add(layers.Dense(1))

model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])

history = model.fit(X_train1, y_train1, validation_split=0.2, epochs=1000000, batch_size=128,callbacks=[early_stopping])

#RMSE計算
loss, accuracy = model.evaluate(X_test1, y_test1)
y_pred1 = model.predict(X_test1)
#print('MSE為：',mean_squared_error(y_test1,y_pred1))
print('RMSE為：',np.sqrt(mean_squared_error(y_test1,y_pred1)))
```

圖 3 每周工作時數 model 訓練程式碼

2. 年收入預測

我們是利用 Anaconda3 Jupyter Notebook(Python version = 3.7)環境進行開發，使用的套件有 sklearn 的 keras、pandas、numpy。對資料進行前處理後，使用類神經網路進行數值預測，而在數值預測中，神經層總共 4 層(輸出層不算)，節點數分別 64、32、16、8 最後一層輸出層為 1，使用 Dropout 參數設定 0.3 、激活函數使用 sigmoid ，優化器使用 rmsprop ，loss 使用 binary_crossentropy，進而獲得績效以及準確度。

```
#訓練模型
#dense指最普通的全連接層型態
#dropout減少節點 通常設0.2~0.5 最高不會超過0.5
#4層(最後一層ouput不算)
from keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(monitor='val_loss', patience=200,restore_best_weights=True)
model = models.Sequential()
model.add(layers.Dense(64, input_shape=(X_train.shape[1],), activation="sigmoid"))
model.add(layers.Dropout(0.30))
model.add(layers.Dense(32, activation="sigmoid"))
model.add(layers.Dropout(0.30))
model.add(layers.Dense(16, activation="sigmoid"))
model.add(layers.Dropout(0.30))
model.add(layers.Dense(8, activation="sigmoid"))
model.add(layers.Dropout(0.30))
model.add(layers.Dense(1, activation="sigmoid"))

model.compile(loss='binary_crossentropy', optimizer='rmsprop' ,metrics=['accuracy'])
history = model.fit(X_train, y_train, validation_split=0.2, epochs=1000000, batch_size=128,callbacks=[early_stopping])

#計算績效
loss, accuracy = model.evaluate(X_test, y_test)
print("Test Acc : " + str(accuracy))
print("Test Loss : " + str(loss))
```

圖 4 年收入預測 model 訓練程式碼

3.4 實驗結果

本研究小組利用類神經網路進行預測對資料集進行分析的結果如下：

1. 每周工作時數預測績效(數值預測)RMSE 約等於 10.81

```
26048/26048 [==============================] - 1s 25us/step - loss: 116.2745 - acc: 0.0762 - val_loss: 119.6343 - val_acc: 0.
0715
Epoch 383/1000000
26048/26048 [==============================] - 1s 25us/step - loss: 114.9313 - acc: 0.0764 - val_loss: 119.7376 - val_acc: 0.
0754
Epoch 384/1000000
26048/26048 [==============================] - 1s 25us/step - loss: 115.4801 - acc: 0.0754 - val_loss: 119.8731 - val_acc: 0.
0769
Epoch 385/1000000
26048/26048 [==============================] - 1s 25us/step - loss: 114.7145 - acc: 0.0758 - val_loss: 120.0323 - val_acc: 0.
0769
Epoch 386/1000000
26048/26048 [==============================] - 1s 29us/step - loss: 115.0732 - acc: 0.0762 - val_loss: 119.7849 - val_acc: 0.
0792
Epoch 387/1000000
26048/26048 [==============================] - 1s 26us/step - loss: 114.6771 - acc: 0.0749 - val_loss: 119.9477 - val_acc: 0.
0768
16281/16281 [==============================] - 0s 29us/step
RMSE為： 10.811155738008482
```

圖 5 年收入預測績效

2. 年收入預測績效(類別預測) Accuracy 約等於 0.85

```
Epoch 512/1000000
26048/26048 [==============================] - 1s 29us/step - loss: 0.3231 - acc: 0.8555 - val_loss: 0.3172 - val_acc: 0.8552
Epoch 513/1000000
26048/26048 [==============================] - 1s 29us/step - loss: 0.3219 - acc: 0.8582 - val_loss: 0.3175 - val_acc: 0.8561
Epoch 514/1000000
26048/26048 [==============================] - 1s 29us/step - loss: 0.3233 - acc: 0.8562 - val_loss: 0.3180 - val_acc: 0.8551
Epoch 515/1000000
26048/26048 [==============================] - 1s 33us/step - loss: 0.3219 - acc: 0.8575 - val_loss: 0.3176 - val_acc: 0.8578
Epoch 516/1000000
26048/26048 [==============================] - 1s 29us/step - loss: 0.3239 - acc: 0.8585 - val_loss: 0.3178 - val_acc: 0.8552
Epoch 517/1000000
26048/26048 [==============================] - 1s 31us/step - loss: 0.3233 - acc: 0.8562 - val_loss: 0.3175 - val_acc: 0.8563
Epoch 518/1000000
26048/26048 [==============================] - 1s 29us/step - loss: 0.3235 - acc: 0.8567 - val_loss: 0.3172 - val_acc: 0.8572
Epoch 519/1000000
26048/26048 [==============================] - 1s 29us/step - loss: 0.3211 - acc: 0.8578 - val_loss: 0.3176 - val_acc: 0.8571
16280/16280 [==============================] - 1s 31us/step
Test Acc : 0.8531941031648135
Test Loss : 0.320434281918282
```

圖 6 年收入預測績效

四、結論

本研究是以 Adult Dataset 為主的數據資料，並使用類神經網路去進行分類並評估兩者的績效及準確度，最終我們得出的結果個別為每周工作時數預測績效(數值預測)RMSE 約等於 10.81，年收入預測績效(類別預測) Accuracy 約等於 0.85。

五、參考文獻

類神經網路：
https://github.com/erhanbaris/deep_learning_for_uci_adult_dataset/blob/master/train.py

US Median Real Income by Year:
https://www.multpl.com/us-median-real-income/table/by-year

UCI Adult Data Set:
https://archive.ics.uci.edu/ml/datasets/Adult