Project 3

班機抵達延遲分析

智慧化企業整合

指導教授: 邱銘傳

109034402 | 陳光源

摘要

近年來航班延誤甚至取消等狀況，因高額的賠償金逐漸受到航空運輸業的重視。班機延誤與取消不僅造成乘客不滿，還會釀成機場或飛機鬧事等案件；而各國政府為了保護消費者權益也設立許多賠償條款，致使航空公司不僅償付賠款還會面臨客戶流失的風險。因此本研究使用深度學習方法中的神經網路建立航班抵達延誤預測模型，並利用 kaggle 網站提供之資料進行訓練與驗證。幫助消費者選擇較少班機延誤的航班或時段；也可協助航空公司了解班機延誤的主因並提早進行預防以降抵損失。

關鍵字：航班延誤、神經網路

目錄

一、 研究動機與目的

近年來航班延誤甚至取消等狀況，因高額的賠償金逐漸受到航空運輸業的重視。班機延誤與取消不僅造成乘客不滿，還會釀成機場或飛機鬧事等案件；而各國政府為了保護消費者權益也設立許多賠償條款，致使航空公司不僅償付賠款還會面臨客戶流失的風險。因此本研究使用深度學習方法中的神經網路建立航班抵達延誤預測模型，並利用 kaggle 網站所提供之 2015 年美國國內線各航班資料進行訓練與驗證。一方面幫助消費者選擇較少班機延誤的航班與時段；另一方面也可協助航空公司了解班機延誤的主因並提早進行預防以降抵損失。

Table 1 5W1H 分析表

項目	內容
What	班機抵達延誤預測
When	購買機票時可根據預測結果的協助做出選擇
Who	欲購買機票的個人或旅行社業者
Where	美國運輸部（DOT）交通統計局
Why	2007 年美國 NAS 預估班機延誤造成航空公司損失 330 億美金 (Kim, Young Jin, et al., 2016)
How	透過深度學習分析資料，建立航班抵達延誤預測模型

二、 研究方法

本研究使用深度學習方法中的神經網路回歸器建立模型，並以 kaggle 的公開資料集進行結果與效度驗證。以下段落將分別說明資料來源及其架構分析、資料前處理過程、探索式資料分析以及參數調整與模型訓練成果。

I. 資料來源

本研究資料來源為美國運輸部交通統計局(U.S. Department of Transportation's Bureau of Transportation Statistics)在 kaggle 的公開資料集，內有關於美國航空公司的特徵資料及其起飛與降落有無延誤時間(flights.csv)，共 1048574 筆資料。此外還有航空公司名稱(airlines.csv)與機場及其所在城市檔案(airports.csv)；總共有三個資料檔案。航班檔案(flights.csv)一共有 31 個欄位，如航空公司名稱、起飛與降落時間、有無延誤以及延誤原因等詳細資料。航空公司名稱檔案(airlines.csv)則包含 flights.csv 檔中各航空公司縮寫的詳細名稱。機場名稱(airports.csv)則除了機場詳細名字外還有其對應城市與地理座標。因為地理位置不抱括在這次分析中，所以只取用前兩個資料集。

詳細內容如 Figure 1, Figure 2, Figure 3 所示。

```
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 31 columns):
 #   Column               Non-Null Count    Dtype
---  ------               --------------    -----
 0   YEAR                 1048575 non-null  int64
 1   MONTH                1048575 non-null  int64
 2   DAY                  1048575 non-null  int64
 3   DAY_OF_WEEK          1048575 non-null  int64
 4   AIRLINE              1048575 non-null  object
 5   FLIGHT_NUMBER        1048575 non-null  int64
 6   TAIL_NUMBER          1040825 non-null  object
 7   ORIGIN_AIRPORT       1048575 non-null  object
 8   DESTINATION_AIRPORT  1048575 non-null  object
 9   SCHEDULED_DEPARTURE  1048575 non-null  int64
 10  DEPARTURE_TIME       1009060 non-null  float64
 11  DEPARTURE_DELAY      1009060 non-null  float64
 12  TAXI_OUT             1008346 non-null  float64
 13  WHEELS_OFF           1008346 non-null  float64
 14  SCHEDULED_TIME       1048573 non-null  float64
 15  ELAPSED_TIME         1005504 non-null  float64
 16  AIR_TIME             1005504 non-null  float64
 17  DISTANCE             1048575 non-null  int64
 18  WHEELS_ON            1007279 non-null  float64
 19  TAXI_IN              1007279 non-null  float64
 20  SCHEDULED_ARRIVAL    1048575 non-null  int64
 21  ARRIVAL_TIME         1007279 non-null  float64
 22  ARRIVAL_DELAY        1005504 non-null  float64
 23  DIVERTED             1048575 non-null  int64
 24  CANCELLED            1048575 non-null  int64
 25  CANCELLATION_REASON  40527 non-null    object
 26  AIR_SYSTEM_DELAY     228528 non-null   float64
 27  SECURITY_DELAY       228528 non-null   float64
 28  AIRLINE_DELAY        228528 non-null   float64
 29  LATE_AIRCRAFT_DELAY  228528 non-null   float64
 30  WEATHER_DELAY        228528 non-null   float64
dtypes: float64(16), int64(10), object(5)
```

Figure 1 flights.csv 資料所包含的資料欄位與資料數量

```
In [4]: print(air.info())
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   IATA_CODE  14 non-null     object
 1   AIRLINE    14 non-null     object
dtypes: object(2)
```

Figure 2 arilines.csv 資料所包含的資料欄位與資料數量

```
In [6]: print(airport.info())
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 322 entries, 0 to 321
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   IATA_CODE  322 non-null    object
 1   AIRPORT    322 non-null    object
 2   CITY       322 non-null    object
 3   STATE      322 non-null    object
 4   COUNTRY    322 non-null    object
 5   LATITUDE   319 non-null    float64
 6   LONGITUDE  319 non-null    float64
dtypes: float64(2), object(5)
```

Figure 3 airports.csv 資料所包含的資料欄位與資料數量

II. 資料前處理

i. 時間欄位

將日期時間欄位 YEAR, MONTH, DAY 轉換成 DATE，並將月和星期轉換為各自對應的月份與星期名稱；以便後續探索性資料分析圖形化使用。

檢查各時間相關欄位後發現年(YEAR)只有 2015 年；月(MONTH)僅有 Jan, Feb, Mar 三種類型;星期(DAY_OF_WEEK)則有完整的種類。

```
5    163070
1    162041
4    159800
7    148678
2    144193
3    141753
6    129040
Name: DAY_OF_WEEK, dtype: int64
1    469968
2    429191
3    149416
Name: MONTH, dtype: int64
Friday       163070
Monday       162041
Thursday     159800
Sunday       148678
Tuesday      144193
Wednesday    141753
Saturday     129040
Name: DAY_OF_WEEK, dtype: int64
Jan    469968
Feb    429191
Mar    149416
Name: MONTH, dtype: int64
2015    1048575
Name: YEAR, dtype: int64
```

Figure 4 時間性資料欄位

ii. 已取消航班

因為已取消航班沒有起飛降落，所以刪除已取消航班的對應資料，並再次檢查。

iii. 合併資料集

檢查兩資料集航空公司數量，都有 14 家；並合併資料，使原有資料航空公司名稱完整。

```
Check Total Number of Airlines in flights.csv: 14
Check Total Number of Airlines in airlines.csv: 14
```

Figure 5 航空公司名稱種類

iv. 轉機

經過檢查發現與航班抵達延誤資料欄意義相同，所以刪除轉機資料欄位。同時刪除滑行、滑出與起飛時間等可從航班抵達延誤欄為獲取相同意義的資料欄位。

```
# DIVERTED: 轉機
print('Number of diverted flights(df["DIVERTED"] = 1): ')
print(df['DIVERTED'].value_counts())
df_DIV = df[df['DIVERTED'] == 1]
print(df_DIV['DIVERTED'].nunique())
print(df_DIV['ARRIVAL_DELAY'].nunique())
# 因與班級延遲資料意義一樣 -> 刪除轉機
# 與DEPARTURE_TIME 和 ARRIVAL_TIME 意義一樣 -> 刪除滑行、滑出、起飛
df = df.drop(columns=['DIVERTED', 'TAXI_OUT', 'WHEELS_OFF', 'WHEELS_ON', 'TAXI_IN'])

```

```
Number of diverted flights(df["DIVERTED"] = 1):
0    1005504
1       2544
Name: DIVERTED, dtype: int64
```

Figure 6 轉機資料欄位總數

v. 刪除缺失值

因所有缺失值為飛機飛行時間與航班延誤原因，為無法替補的資料，所以直接刪除有缺失值的資料。經過資料前處裡後的資料集有 228528 筆與 17 欄欄位。

```
# Dealing with missing values
print('Summation of missing values in flights')
print(df.isnull().sum())
df = df.dropna()
df = df.reset_index()
df = df.drop(columns=['index'])
df.info()
print('New shape of df ', df.shape)
```

III. 探索式資料分析

探索式資料分析主要在釐清資料資訊結構與其特性，同時檢視資料有無離群值、分析各變數間關聯性、提取重要的訊息，也能檢查資料有沒有適合後續將進行分析模型前的假設，以便提前做好分析準備。

i. 視覺化數據

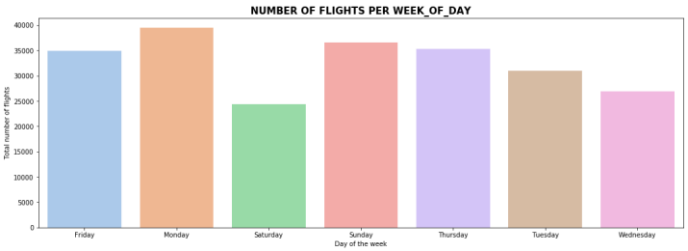


Figure 7 航班數量與每週天數圖

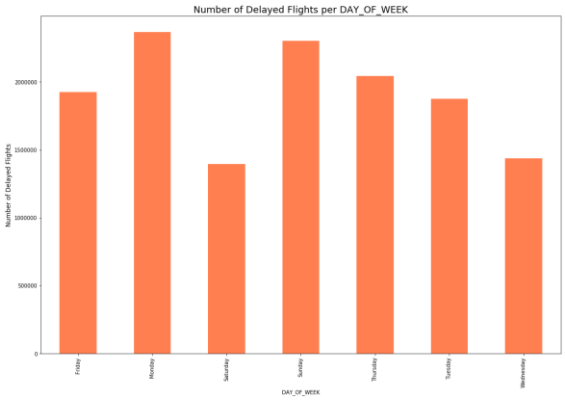


Figure 8 班機抵達延誤與每週天數

由 Figure 7 航班數量與每週天數和 Figure 8 班機抵拔延誤與每週天數兩張圖可得知週一有較多的航班，周日的航班卻是最少。較多航班同樣也有較多的班機延誤數量，與一般認知相符。

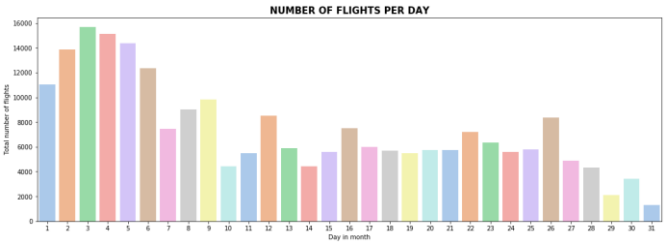


Figure 9 航班數量與每月天數圖

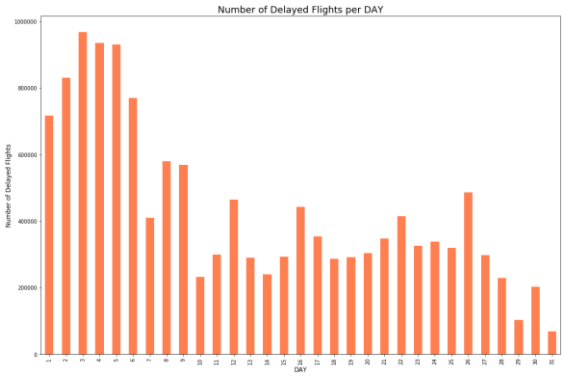
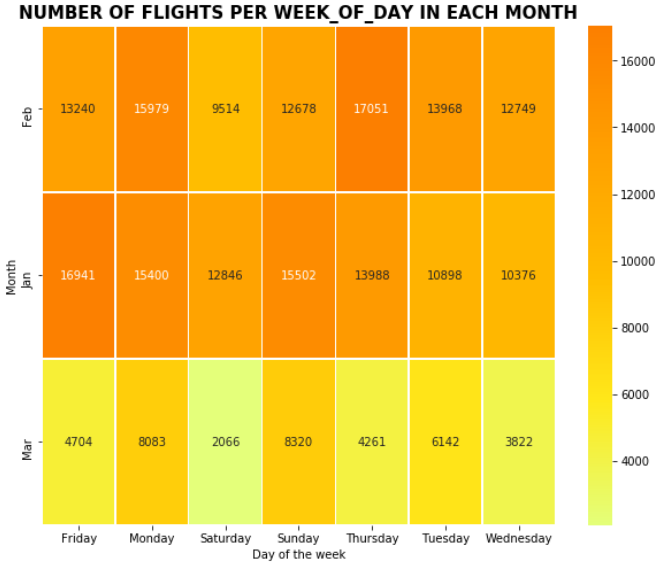


Figure 10 班機抵拔延誤與每月天數

由 Figure 9 航班數量與每月天數和 Figure 10 班機抵拔延誤與每月天數可得知月初與月尾有較多的航班，而月初又比月尾航班數量多出許多，月中航班卻是最少。同樣較多航班的天數有較多的班機延誤數量。



因為整筆資料的月份僅有一到三月，所以無法檢視月份比較航班延遲數量班機數量；但仍可繪制每月的每週天數航班數量熱力關係圖(number of flights per day_of_week in each month)。發現年初有較多的航班數量，且週四、一、五、日有稍多的航班；與航班數量與每週天數圖的結果相符。

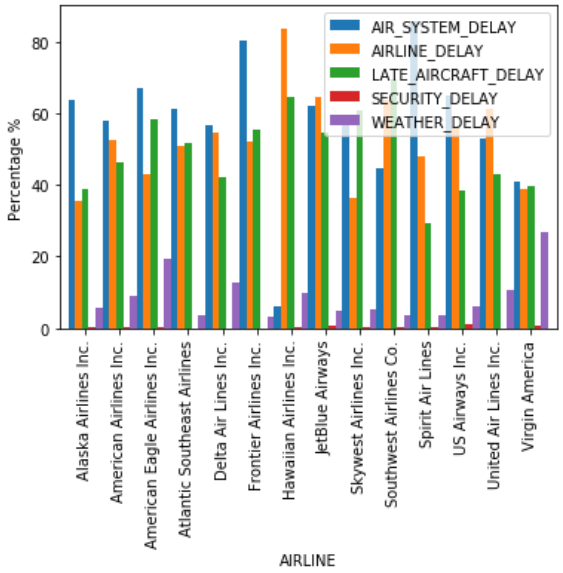


Figure 11 各航空公司班機延誤原因所占百分比

由 Figure 11 可得知不同航空公司班機延誤原因有所不同，大多數原因為航空管理系統延誤(包含機場運作與過高的交通流量)，天氣與安全因素為最低。其中 Hawaiian 航空公司的最多延誤原因卻是航空公司的延誤(機上組員調配與飛機添加燃料等1)。

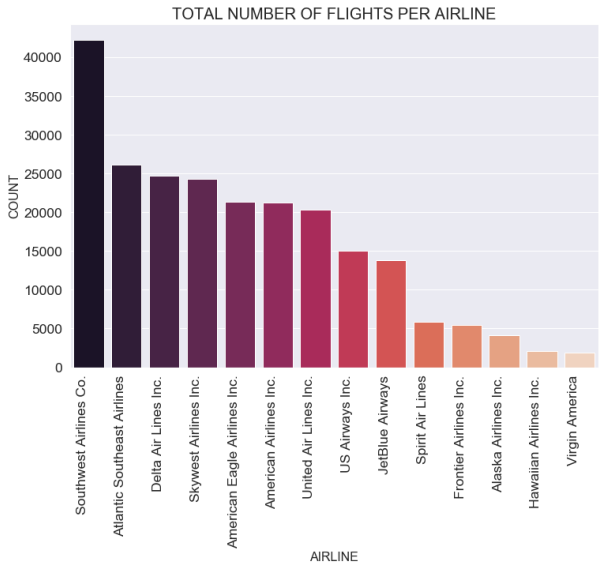


Figure 12 number of destinations by airline

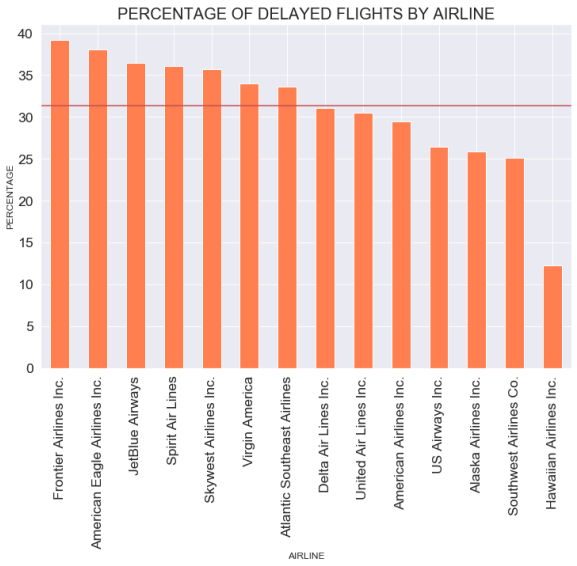


Figure 13 percentage of delayed flights per airline

[1] https://simcept.com/once-upon-a-day-21-flight-delay/

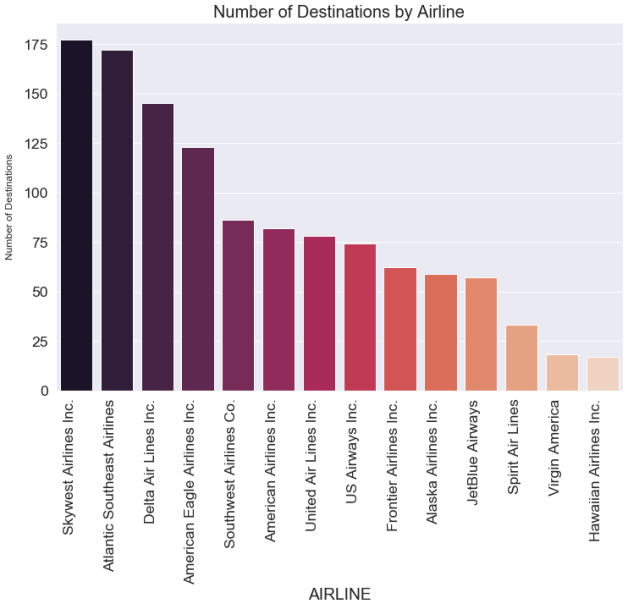


Figure 14 total number of flight per airline

由 Figure 12 與 Figure 13 得知，有 7 家航空公司班機延誤數量高於平均。Southwest 航空公司雖有最多的航班其延誤航班數卻低於平均許多，可說是優質航空公司。相反 Frontier 航空公司的航班是相對較少的，他卻是航班延誤作多的公司之一。值得注意的是與航空公司航點數相比之後，Atlantic 航空公司在航點與航班數量排名一致下，航班延誤僅稍高於平均。同時由 Figure 14 發現航點與航班數量資料所帶來的訊息相同，後續資料分析可以只取一個。

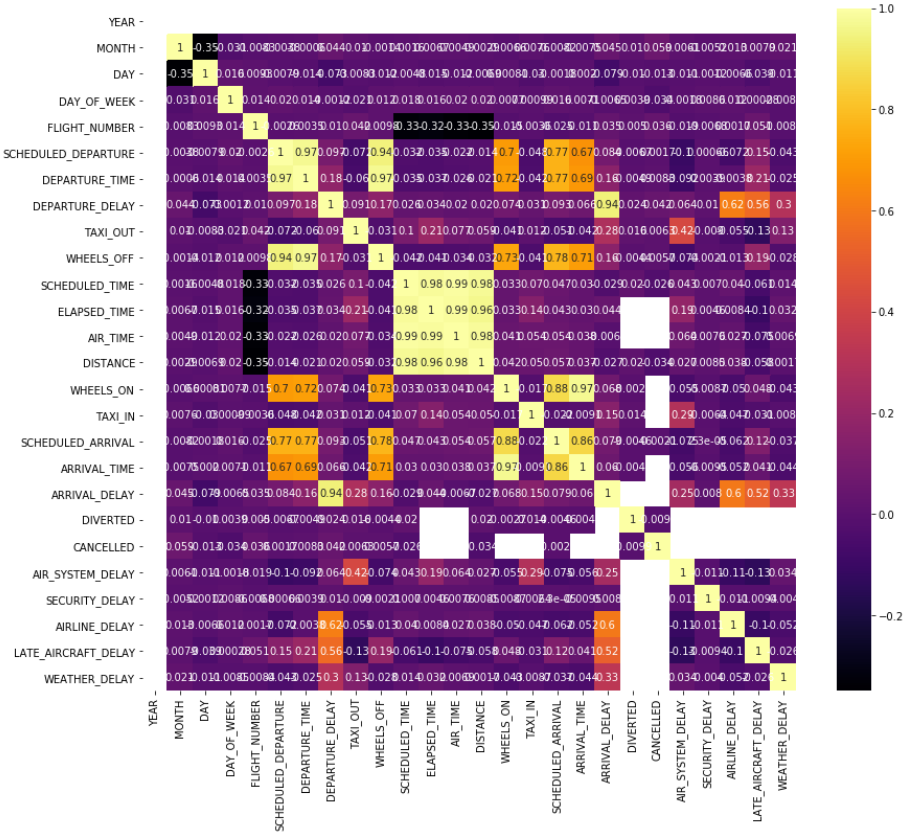


Figure 15 特徵值間的關係圖

最後繪製特徵值間的關係圖，其中航班起飛延誤與抵達延誤有高度正相關；說明大多數的班機因為起飛延誤而導致抵達延誤，也有起飛延誤但因改變航線而提早抵達的航班。因目標特徵值(traget)是由航班抵達延誤(ARRIVAL_DELAY)獲得，且其與航班起飛延誤(DEPARTURE_DELAY)有高度正相關，刪除 ARRIVAL_DELAY 與 DEPARTURE_DELAY。

ii. 資料預處理

- 資料編碼

以 sklearn 資料前處裡提供的 LabelEncoding 針對類別型進行資料編碼，將每一類別對應到某個整數，好處為不增加資料欄位。

```
# Label encoding
# 資料編碼 -> label encoding: 把每個類別 mapping 到某個整數, 不增加新欄位
le = LabelEncoder()
df['AIRLINE']= le.fit_transform(df['AIRLINE'])
df['ORIGIN_AIRPORT'] = le.fit_transform(df['ORIGIN_AIRPORT'])
df['DESTINATION_AIRPORT'] = le.fit_transform(df['DESTINATION_AIRPORT'])
df = df.drop(columns=['DATE'])
#df['DATE'] = le.fit_transform(df['DATE'])
df['MONTH'] = le.fit_transform(df['MONTH'])
df['DAY_OF_WEEK'] = le.fit_transform(df['DAY_OF_WEEK'])
df['SCHEDULED_DEPARTURE'] = le.fit_transform(df['SCHEDULED_DEPARTURE'])
df['DEPARTURE_TIME'] = le.fit_transform(df['DEPARTURE_TIME'])
df['SCHEDULED_ARRIVAL'] = le.fit_transform(df['SCHEDULED_ARRIVAL'])
df['ARRIVAL_TIME'] = le.fit_transform(df['ARRIVAL_TIME'])
df['target'] = le.fit_transform(df['target'])
```

- 資料樣本區分

增加新的一個目標欄位(target)，選取航班抵達延誤(ARRIVAL_DELAY)欄位中大於平均的設為班機延誤(target=1)，其餘航班抵達延誤小於平均的為準時抵達班機(target=0)。以 sklearn 為資料選擇所提供的 train_test_split()將資料區分為 70%的訓練及與 30%的測試集。

```
# Createing target column
df['target'] = df['ARRIVAL_DELAY'] > df['ARRIVAL_DELAY'].mean()
print('Percentage of delayed flights per airline:')
print(df['target'].value_counts(normalize=True))
print('False means arrive ontime. True means delayed.')
```

- 標準化

因目標特徵值欄位來自航班起飛與抵達延誤，需先刪除。在定義 X 與 y 欄位後，以 sklearn 的資料前處理提供的 StandartScaler 進行資料標準化。 為了避免後續模型訓練時特徵值大的資料欄位影響其他特徵值,將所有資料標準化;使其變異數為 0,標準差為 1。

```
# Splitting y and x
# 因為 ARRIVAL_DELAY == target 且 大多DEPARTURE DELAY 就會有 ARRIVAL DELAY
# X 不取 DEPARTURE DELAY 與 ARRIVAL DELAY
X = df.drop(['target', 'ARRIVAL_DELAY', 'DEPARTURE_DELAY'],axis = 1)
y = df['target']

# Splitting into train and test data set
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state = 2)
# 標準化資料
sc1=StandardScaler()
X_train_sc=sc1.fit_transform(X_train)
X_test_sc=sc1.transform(X_test)
num_features = len(X_train.columns)
```

三、 模型訓練成果與參數調整

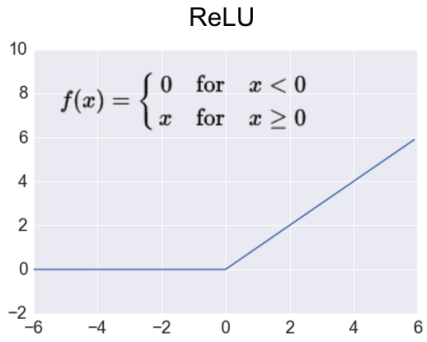
首先利用 Linear Regression 進行分析，得到的效果不佳，因此嘗試利用全連結層神經網絡進行模型訓練。

```
Linear Regression
Mean Absolute Error: 0.26339660172746254
Mean Squared Error: 0.1144467636731898
Root Mean Squared Error: 0.3382998132916863
R2 :  0.46761802741449865
Accuracy:  0.46761802741449865
```

I. 模型所用之激活函數

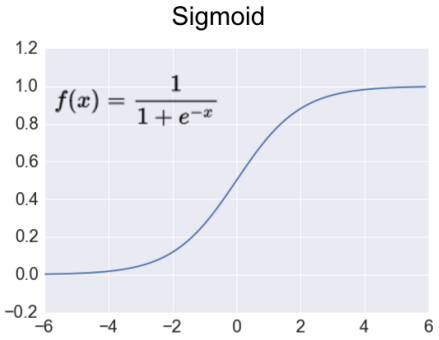
1. relu (Rectified Linear Units)：輸出值介於 [0,∞] 之間。

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



2. sigmoid：輸出值介於 [0,1] 之間，且分布兩極化(1 或 0)，適合二分法。

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



II. 模型所用之初始化方法

初始化方法定義對神經元層設定初始化權重的方法。本實驗使用的 Random Uniform(kernel_initializer='random_uniform') 為採均勻分配(區間內每一點機率都相同)的隨機亂數，並在設定的區間內隨機抽樣。

Figure 17 隨機均勻分配初始化函數

III. 模型所用之正則化參數

正則化函數的作用是對權重(W)矩陣加上懲罰性函數，以防止過度擬合，共有 kernel, bias 與 activity 三種，分別對 weight, bias 與函數輸出值 y 加上懲罰，這裡使用 kernel_regularizer。kernel_regularizer 又有 l1 與 l2 兩種，在梯度下降時 l1 以 sign(w)與權重相減，會讓權重矩陣有過多零值，所以選取以 2w 對權重相減的 l2 做為正則化參數。

L1 regularizer,

$$\mathrm{R(w)} = ||w||_1 = \sum_i |w_i|$$

L2 regulrizer,

$$\mathrm{R(w)} = ||w||_2^2 = \sum |w_i^2|$$

IV. 模型說明

模型 1

共有三層 fully connected layers，第一層有 30 個 units、激活函數為 relu，第二層有 20 個 units、激活函數為 relu，最後一層為輸出層、激活函數為 sigmoid。損失函數為 mean_squared_error，優化函數為 adam，共 10 個 epoch。模型訓練時在第 1 個 epoch 時準確率最高為 68.64%，最小 loss 則在第 3 個 epoch。模型在測試資料時在第 0 個 epoch 時準確率最高為 68.92%，最小 loss 則在第 6 個 epoch。因為在還沒有開始測試時模型就已有最高的準確率，顯然模型並沒有學習成功，進一步檢查混淆矩陣也發現模型無法預測有準時抵達的航班。

模型 2

共有四層 fully connected layers，第一層有 256 個 units、激活函數為 relu，第二層有 128 個 units、激活函數為 relu，第三層有 128 個 units、激活函數為 relu，最後一層為輸出層、激活函數為 sigmoid。損失函數為 binary_crossentropy，優化函數為 adam，共 16 個 epoch。模型訓練時在第 0 個 epoch 時準確率最高為 31.36%，最小 loss 則在第 0 個 epoch。模型在測試資料時在第 0 個 epoch 時準確率最高為 31.08%，最小 loss 則在第 1 個 epoch。因為在還沒有開始測試時模型就已有最高的準確率與最小 loss，顯然模型並沒有學習成功，進一步檢查混淆矩陣也發現模型無法預測沒有準時抵達的航班。

模型 3

共有四層 fully connected layers，第一層有 256 個 units、激活函數為 relu，第二層有 128 個 units、激活函數為 relu，第三層有 128 個 units、激活函數為 relu，最後一層為輸出層、激活函數為 sigmoid。損失函數為 binary_corssentropy，優化函數為 adagrad，共 16 個 epoch。不同於模型 1，2 的是，在這裡加入了 kernel_regularizer=regularizers.l1(0.01)，以減少模型訓練時的過擬合。在第 15 個 epoch 時準確率最高為 96.17%，最小 loss 則在第 15 個 epoch。模型在測試資料時在第 15 個 epoch 時準確率最高為 96.44%，最小 loss 則在第 15 個 epoch。進一步檢查模型 2 在測試集的其他分數均超過 90%。

模型 4

共有四層 fully connected layers，第一層有 256 個 units、激活函數為 relu，第二層有 128 個 units、激活函數為 relu，第三層有 128 個 units、激活函數為 relu，最後一層為輸出層、激活函數為 sigmoid。損失函數為 binary_corssentropy，優化函數為 adagrad，共 16 個 epoch。不同於模型 3 的是，在這裡加入了 kernel_initializer='random_uniform' 與 kernel_regularizer=regularizers.l2(0.01)，以減少模型訓練時的過擬合。在第 15 個 epoch 時準確率最高為 96.05%，最小 loss 則在第 15 個 epoch。模型在測試資料時在第 15 個 epoch 時準確率最高為 96.62%，最小 loss 則在第 15 個 epoch。進一步檢查模型 2 在測試集的其他分數均超過 90%。

模型 5

共有四層 fully connected layers，第一層有 256 個 units、激活函數為 relu，第二層有 128 個 units、激活函數為 relu，第三層有 128 個 units、激活函數為 relu，最後一層為輸出層、激活函數為 sigmoid。kernel_initializer='random_uniform'、kernel_regularizer=regularizers.l2(0.01)，損失函數為 mean_squared_error，優化函數為 adam，共 16 個 epoch。在第 15 個 epoch 時準確率最高為 95.71%，最小 loss 則在第 10 個 epoch。模型在測試資料時在第 15 個 epoch 時準確率最高為 97.97%，最小 loss 則在第 10 個 epoch。進一步檢查模型 2 在測試集的其他分數均超過 90%。

模型 6

共有四層 fully connected layers，第一層有 256 個 units、激活函數為 relu，第二層有 128 個 units、激活函數為 relu，第三層有 128 個 units、激活函數為 relu，最後一層為輸出層、激活函數為 sigmoid。kernel_initializer='random_uniform' 、kernel_regularizer=regularizers.l2(0.01)，損失函數為 binary_crossentropy，優化函數為 adagrad，共 16 個 epoch。在第 15 個 epoch 時準確率最高為 96.66%，最小 loss 則在第 15 個 epoch。模型在測試資料時在第 15 個 epoch 時準確率最高為 97.62%，最小 loss 則在第 15 個 epoch。進一步檢查模型 2 在測試集的其他分數均超過 90%。

總結以上所述，模型最高準確率是在第 5 次實驗的模型，但平均測試集僅有 94.99%；而第 6 次實驗的模型有 97.53%，所以實際運用應該選擇在陌生數據有高準確率的模型 6。

Table 2 將上述 5 種模型訓練與測試結果整理表:

Model #	Performances	Visualization Loss and Accuracy

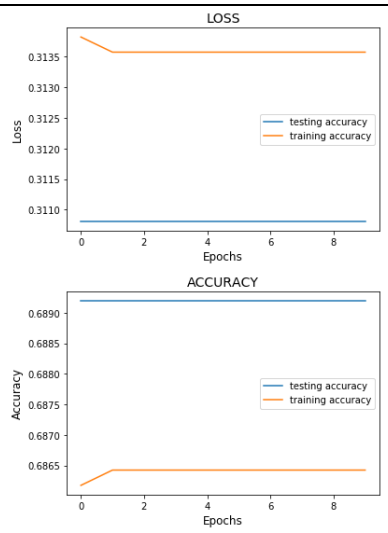
1

```
Maximum accuracy in Training set: 68.64
epoch:  1
Maximum accuracy in Testing set: 68.92
epoch:  0
Minimum loss in Training set: 31.36
epoch:  3
Minimum loss in Testing set: 31.08
epoch:  6
[[47111     0]
 [21448     0]]
              precision    recall  f1-score   support

           0       0.69      1.00      0.81     47111
           1       0.00      0.00      0.00     21448

    accuracy                           0.69     68559
   macro avg       0.34      0.50      0.41     68559
weighted avg       0.47      0.69      0.56     68559

On Testing Set
Accuracy: 68.72 %
Precision score: 0.0 %
Recall score: 0.0 %
F1 score: 0.0 %
None
```



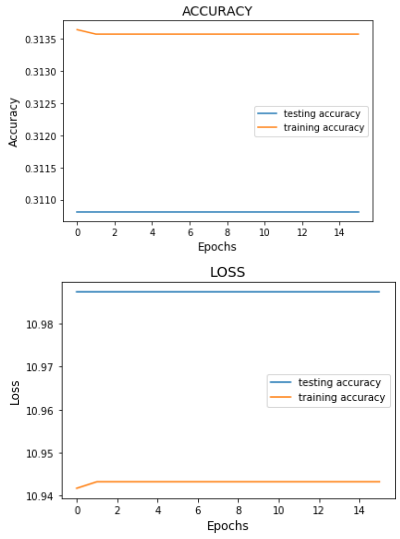
2

```
Maximum accuracy in Training set: 31.36
epoch:  0
Maximum accuracy in Testing set: 31.08
epoch:  0
Minimum loss in Training set: 1.094e+03
epoch:  0
Minimum loss in Testing set: 1.099e+03
epoch:  1
[[    0 47111]
 [    0 21448]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00     47111
           1       0.31      1.00      0.48     21448

    accuracy                           0.31     68559
   macro avg       0.16      0.50      0.24     68559
weighted avg       0.10      0.31      0.15     68559

On Testing Set
Accuracy: 31.28 %
Precision score: 31.28 %
Recall score: 100.0 %
F1 score: 47.66 %
None
```



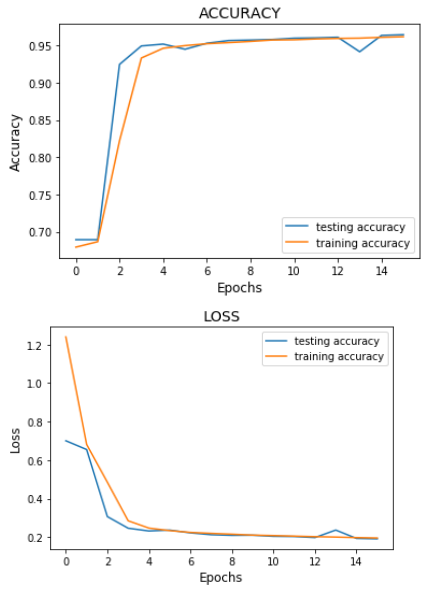
3

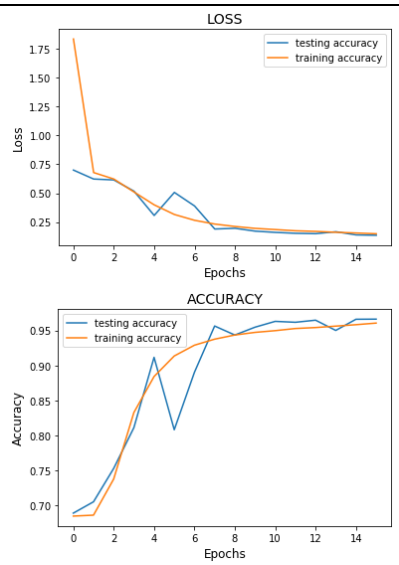
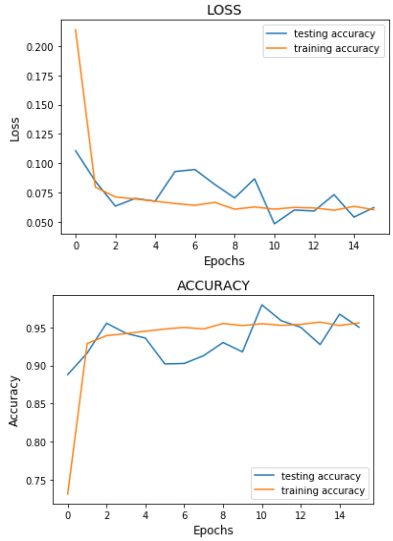
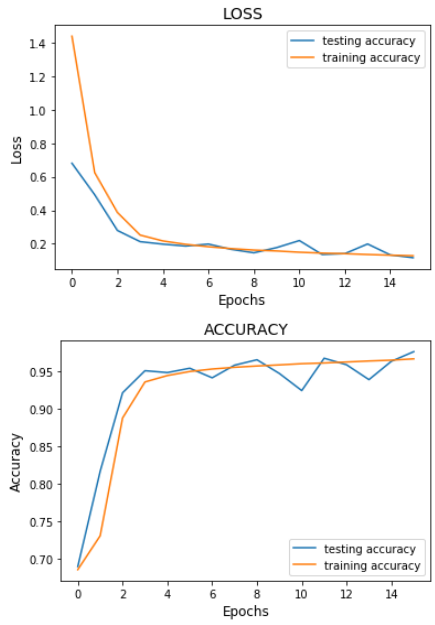
```
Maximum accuracy in Training set: 96.17
epoch:  15
Maximum accuracy in Testing set: 96.44
epoch:  15
Minimum loss in Training set: 19.6
epoch:  15
Minimum loss in Testing set: 19.15
epoch:  15
[[45865  1246]
 [ 1212 20236]]
              precision    recall  f1-score   support

           0       0.97      0.97      0.97     47111
           1       0.94      0.94      0.94     21448

    accuracy                           0.96     68559
   macro avg       0.96      0.96      0.96     68559
weighted avg       0.96      0.96      0.96     68559

On Testing Set
Accuracy: 96.41 %
Precision score: 94.2 %
Recall score: 94.35 %
F1 score: 94.27 %
```



4	Maximum accuracy in Training set: 96.05 epoch: 15 Maximum accuracy in Testing set: 96.62 epoch: 15 Minimum loss in Training set: 14.93 epoch: 15 Minimum loss in Testing set: 13.6 epoch: 15 [[46127 984] [1327 20121]] precision recall f1-score support 0 0.97 0.98 0.98 47111 1 0.95 0.94 0.95 21448 accuracy 0.97 68559 macro avg 0.96 0.96 0.96 68559 weighted avg 0.97 0.97 0.97 68559 On Testing Set Accuracy: 96.63 % Precision score: 95.34 % Recall score: 93.81 % F1 score: 94.57 % None	
5	Maximum accuracy in Training set: 95.71 epoch: 13 Maximum accuracy in Testing set: 97.97 epoch: 10 Minimum loss in Training set: 21.39 epoch: 0 Minimum loss in Testing set: 11.08 epoch: 0 [[43743 3368] [70 21378]] precision recall f1-score support 0 1.00 0.93 0.96 47111 1 0.86 1.00 0.93 21448 accuracy 0.95 68559 macro avg 0.93 0.96 0.94 68559 weighted avg 0.96 0.95 0.95 68559 On Testing Set Accuracy: 94.99 % Precision score: 86.39 % Recall score: 99.67 % F1 score: 92.56 %	
6	Maximum accuracy in Training set: 96.66 epoch: 15 Maximum accuracy in Testing set: 97.62 epoch: 15 Minimum loss in Training set: 12.89 epoch: 15 Minimum loss in Testing set: 11.67 epoch: 15 [[45981 1130] [563 20885]] precision recall f1-score support 0 0.99 0.98 0.98 47111 1 0.95 0.97 0.96 21448 accuracy 0.98 68559 macro avg 0.97 0.97 0.97 68559 weighted avg 0.98 0.98 0.98 68559 On Testing Set Accuracy: 97.53 % Precision score: 94.87 % Recall score: 97.38 % F1 score: 96.1 % None	

四、 結論

綜合以上模型在訓練以及測試資料的結果，可得知模型使用激活函數為 relu 和最後輸出層激活函數為 sigmoid、損失函數為 binary_crossentropy、激活函數為 adagrad 以及加入模型初始化(random uniform)與正則化函數(l2)避免過擬合時，模型表現最佳；訓練集準確率達 96.66 %以及測試集準確率達 97.62 %，平均為 97.53%。

本研究使用深度學習的全連結層神經網路建立航班抵達延誤預測模型，其模型預測結果具良好的效度。未來可以獲取更多月份或時間性資料進行分析，以取得根據連假出遊變動的航班需求資訊，更有效的預測結果；此外在模型建構上可再做更多參數的調適，提升模型效度。

Table 3 模型訓練參數調整整理表

#	Activation function	epoch	Optimizer function	Loss function	Batch size	initializer and regularize	Max. performance
1	Relu and sigmoid	10	adam	mean_squared_error	64	No	68.64 % 68.92 %
2	Relu and sigmoid	16	adam	binary_crossentropy	64	No	31.36 % 31.08 %
3	Relu and sigmoid	16	adagrad	binary_crossentropy	64	No and l1	96.17 % 96.44 %
4	Relu and sigmoid	16	adagrad	binary_crossentropy	64	random_uniform and l2	96.05 % 96.62 %
5	Relu and sigmoid	16	adam	mean_squared_error	64	random_uniform and l2	95.71 % 97.97 %
6	Relu and sigmoid	16	adagrad	binary_crossentropy	64	random_uniform and l2	96.66 % 97.62 %

五、 參考資料

模型訓練參數調整參考

激活函數: https://ithelp.ithome.com.tw/articles/10191725

Kernel initializers: https://keras.io/api/layers/initializers/

https://becominghuman.ai/priming-neural-networks-with-an-appropriate-initializer-7b163990ead

Kernel regularizers: https://keras.io/api/layers/regularizers/

https://stats.stackexchange.com/questions/383310/what-is-the-difference-between-kernel-bias-and-activity-regulizers-and-when-t

https://www.itread01.com/content/1513589905.html

論文參考

Chakrabarty, Navoneel. "A Data Mining Approach to Flight Arrival Delay Prediction for American Airlines." *2019 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference (IEMECON)*. IEEE, 2019.

Kim, Young Jin, et al. "A deep learning approach to flight delay prediction."*2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*. IEEE, 2016.

張惟帆. "航班時間可靠度衡量指標與影響因素分析." *交通大學運輸與物流管理學系學位論文* 2016 年 (2016): 1-56.

何學庸, and 张鈥崴. "航班取消, 延誤及超額訂位致旅客無法登機賠償法规發展之研究-以欧盟, 英國, 美國及台灣航空運輸市場為例." *休閒與遊憩研究* 4.2 (2010): 73-100.

黄明光. "借鉴外国经验 降低民航班机延误及乘客闹事率." *漯河職業技術學院學報* 14.6 (2015): 118-119.