國　立　清　華　大　學
工業工程與工程管理學系

智慧化企業整合
Intelligent Integration of Enterprise

應用LSTM預測共享單車需求

110034540 徐阡珊

指導教授：　　邱銘傳　　博士

中　華　民　國　一　一　一　年　一　月

目錄

摘要

共享單車的站點與需求量越來越多，公共自行車的供應商需於各站點間進行調度，才能在現有的單車數量下，盡量滿足使用者需求。本研究使用深度學習中的長短期記憶(Long Short-Term Memory, LSTM)演算法建構共享單車的需求預測模型，在現有時間序列的資料集下，依據歷史數據預測出未來的使用量需求。

壹、簡介

一、背景動機與研究目的

隨著環保意識與健康意識抬頭，騎單車不僅是一項休閒活動，更是通勤上下班的移動運具。現今許多國家皆有設立共享單車系統，個人可以在短期內有償或免費使用共享單車，不僅便利民眾的生活，也是觀光客探索城市的一個好選擇。

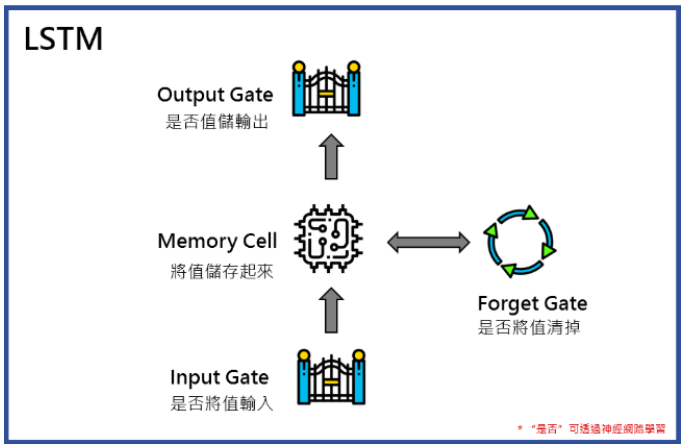
目前大多數的共享單車系統仍採取建置停車樁的方式，確保民眾遵守共享規範，也方便公司進行管理。然而，多數使用者都曾遇過到了站點卻無車可借，或是沒有停車柱可還車的狀況。為了滿足使用者的需求，本研究期望透過共享單車使用數量的歷史資料，預測出未來的共享單車需求數量，讓共享單車的供應商能參考預測的需求量，進行站點單車數量的調度，也提供採購單車數量的依據。

二、5W1H 分析

What	共享單車需求量預測
Why	滿足使用者需求
Where	共享單車的供應商總部
Who	負責數據分析或站點管理的員工
When	每日上班時進行預測，安排當日調度人員的工作
How	運用LSTM進行預測

貳、文獻回顧

RNN是專門設計用於處理時間序列數據的神經網路模型，適用於文本、音節、影片、天氣預報數據與股票交易等其他時間序列數據。而長短期記憶(Long Short-Term Memory, LSTM)演算法是一種特定形式的RNN (Recurrent Neural Network)。LSTM目的在解決RNN在長期序列訓練過程中，梯度消失或梯度爆炸的問題，主要分為三個階段，忘記階段、選擇記憶階段以及輸出階段。忘記階段主要是對上一個節點傳進來的輸入值執行選擇性遺忘，經由計算得到遺忘閥 (Forget Gate)，來控制上一個狀態中哪些需要保留或遺忘。選擇記憶階段將這個階段的輸入值進行選擇性的「記憶」，重要之狀態予以著重記憶，反之不重要者則少記一些。輸出階段則決定哪些因子作為當前狀態之輸出。其簡單的原理如圖一所示。



圖一 LSTM簡單原理圖示

參、研究方法

一、資料前處理

(一) 皮爾森積差相關分析 (Pearson correlation)

皮爾森相關分析用於探討兩變數之間的線性關係，其值介於-1到1之間，當兩變數之間的相關係數絕對值較大，則代表彼此相互共變的程度較大。一般而言，正值越大代表兩變數屬於強正相關，反之負值越大則代表屬於強負相關。其公式如下：

$$r(x,y) = \frac{COV(x,y)}{SxSy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2} \times \sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

(二) 資料特徵縮放(RobustScaler中位數和四分位數標準化)

進行模型訓練前，特徵的前置預處理是必經過程，因為每個特徵有自己的值域與單位，沒有在相同的空間範圍內，會造成擁有較大值域的特徵對模型的影響過大，因此需要將每個特徵縮放至相同的大小。RobustScaler是一種四分位距的計算，以$\frac{(數值-中位數)}{四分位距}$來做資料縮放，其中四分位距為Q3-Q1。

二、超參數優化 (Hyperparameter tuning)

針對 LSTM 模型的超參數優化實驗，本研究採用實驗設計中的田口式直交表實驗法，其構想是以較少的實驗次數來獲得有用的統計資訊。典型的田口式直交表是以 $L_a(b^c)$來命名，代表共有 a 組實驗、最多可以容納 b 個水準的因子 c 個，亦即一個 a 橫列 c 直行的實驗陣列。本研究進行實驗所採用的 $L_9(3^4)$直交表如表一所示：

表一 $L_9(3^4)$直交表

Exp.	1	2	3	4
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1

肆、個案研究

本研究使用資料科學社群平台Kaggle上之公開資料集 “London bike sharing dataset”，該資料來源為倫敦交通局之公開資料。原始資料為CSV檔，共有10個資料欄位，如表二所示，其中cnt(共享單車數量)為本研究之目標預測值。雖然原資料集中之cnt可能為共享單車的總數量，但在此研究中將cnt視為一個站點的共享單車使用量，以便將預測結果用於站點間的調度。

表二 資料集內容

資料欄位	說明
timestamp	時間戳記
cnt	共享單車數量
t1	實際溫度(攝氏)
t2	體感溫度(攝氏)
hum	濕度百分比
wind_speed	風速(公里/小時)
weather_code	天氣類別(1-晴朗；2-分散的雲；3-破碎的雲；4-多雲；7-下雨；10-雷雨；26-降雪；94-冰霧)
is_holiday	是否為假日(0-非假日；1-假日)
is_weekend	是否為周末(0：非周末、1：周末)
season	季節(0-春天；1-夏天；2-秋天；3-冬天)

一、資料前處理

(一) 檢視資料

1. 確認有無缺失值與資料型態

在讀入資料集後，很幸運地資料沒有缺失值，且除時間戳記外，各項特徵值皆為數值型態。

```
df.info()
```

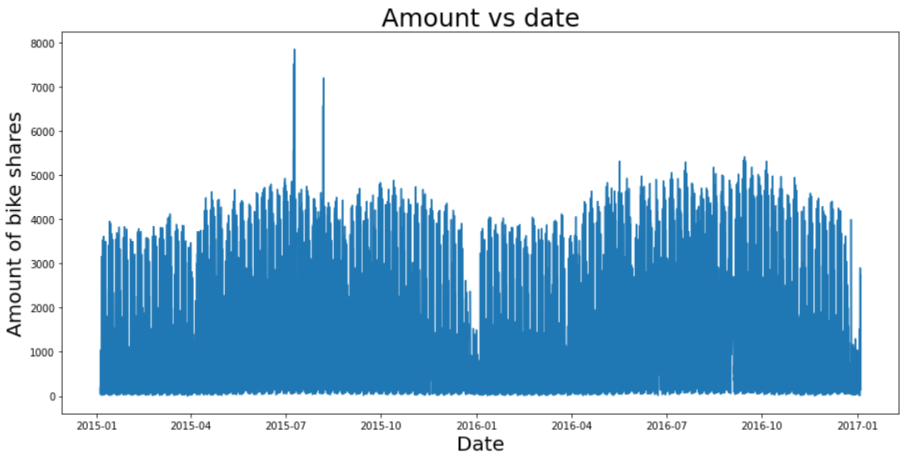
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17414 entries, 0 to 17413
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   timestamp     17414 non-null  object
 1   cnt           17414 non-null  int64
 2   t1            17414 non-null  float64
 3   t2            17414 non-null  float64
 4   hum           17414 non-null  float64
 5   wind_speed    17414 non-null  float64
 6   weather_code  17414 non-null  float64
 7   is_holiday    17414 non-null  float64
 8   is_weekend    17414 non-null  float64
 9   season        17414 non-null  float64
dtypes: float64(8), int64(1), object(1)
memory usage: 1.3+ MB
```

2. 觀察資料趨勢

此筆數據的時間戳記格式為「年-月-日 時-分-秒」，並以小時為單位。為了觀察需求變化，統計在不同時間長度下的共享單車數量，在各筆資料中新增各個時間長度的特徵。

```
df["timestamp"] = pd.to_datetime(df["timestamp"])
df = df.set_index("timestamp")
df["hour"] = df.index.hour
df["day_of_month"] = df.index.day
df["day_of_week"] = df.index.dayofweek
df["month"] = df.index.month
df
```

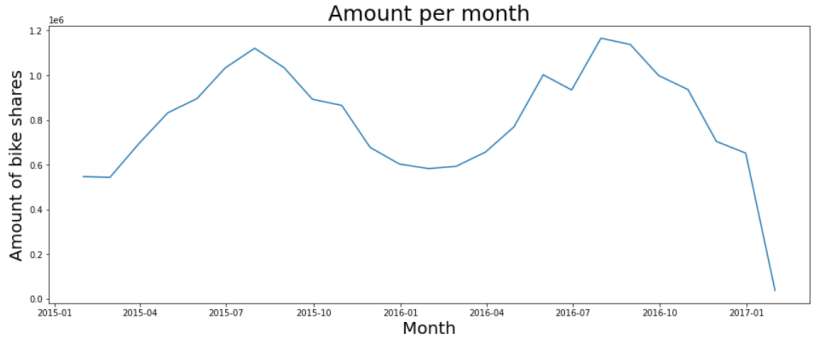
每小時的使用量如圖二所示。



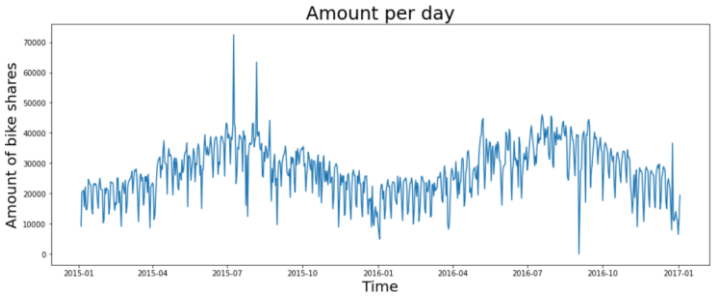
圖二 每小時共享單車使用量

利用 resample()與sum()函數重新建構時間頻率與計算各時間長度的共享單車數量總和，如圖三至圖七所示。

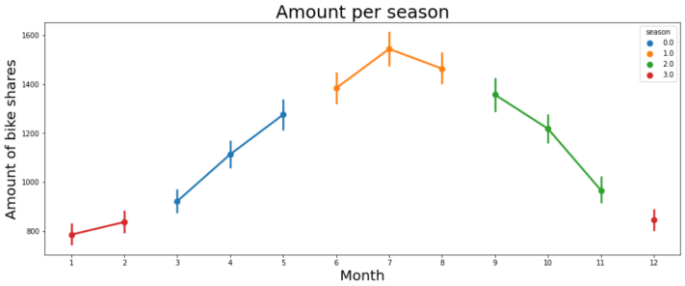
```
df_by_month = df.resample("M").sum()

plt.figure(figsize=(16,6))
ax = sns.lineplot(data=df_by_month, x=df_by_month.index, y=df_by_month.cnt)
ax.set_title("Amount per month", fontsize=25)
ax.set_xlabel("Month", fontsize=20)
ax.set_ylabel('Amount of bike shares', fontsize=20)
plt.show()
```



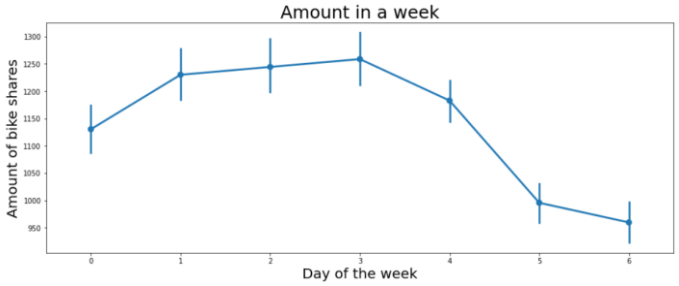
圖三 每月共享單車使用量



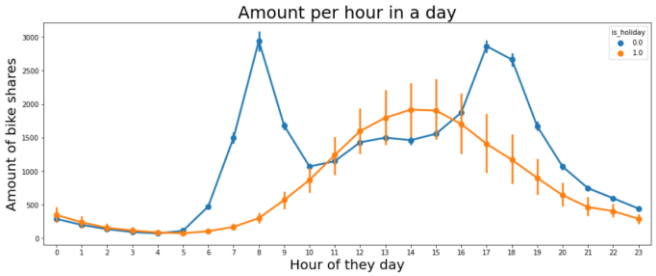
圖四 每日共享單車使用量



圖五 每季共享單車使用量



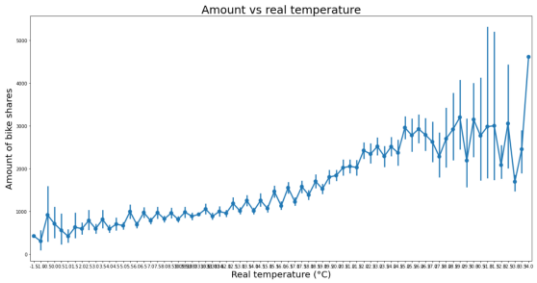
圖六 周間每日共享單車使用量



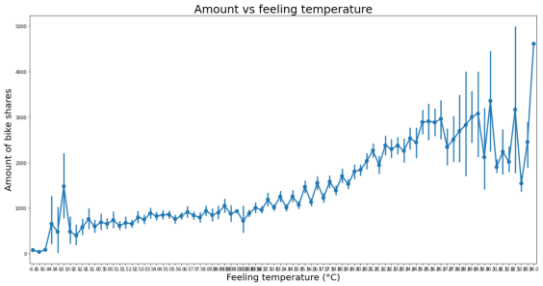
圖七 假日與非假日每小時共享單車使用量

3. 觀察資料相關性

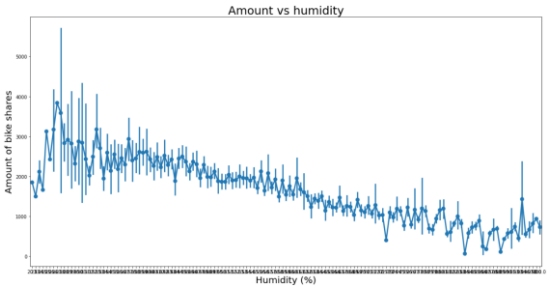
原始數據集的各項屬性與共享單車數量的關係如圖八至圖十二所示，圖十三顯示各屬性的相關係數熱圖。統整各可視化圖表可以發現在溫度較高、中等風速、天氣較佳時會有較多的使用量，但濕度與使用量則呈現負相關(濕度越高，降雨機率越大)。



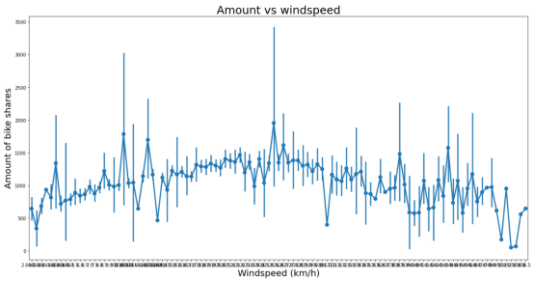
圖八 共享單車使用量與實際溫度



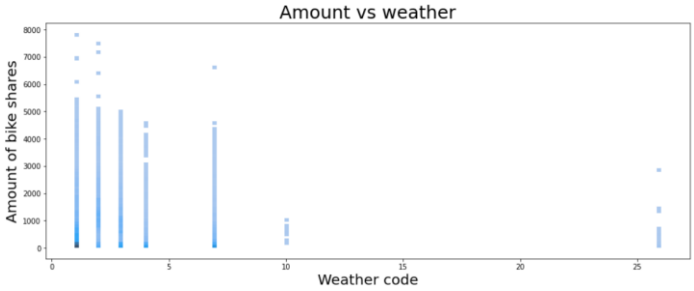
圖九 共享單車使用量與體感溫度



圖十 共享單車使用量與濕度百分比



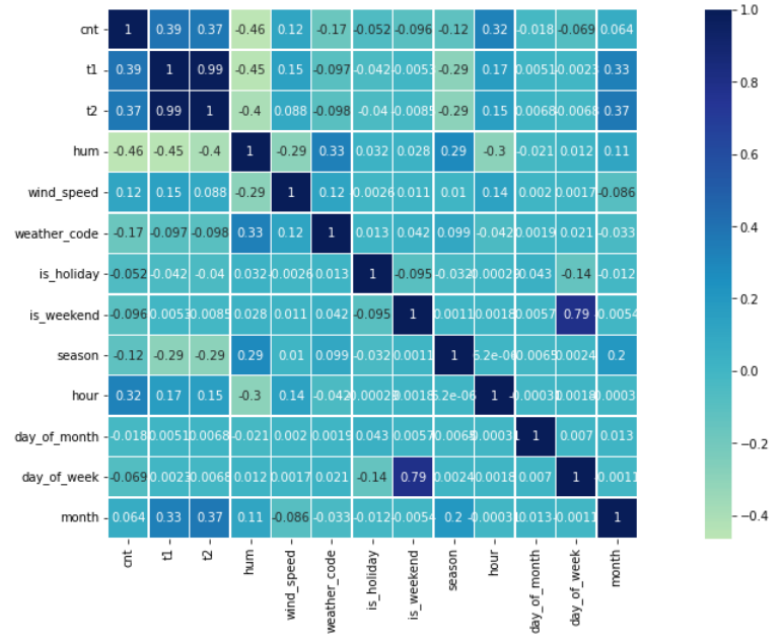
圖十一 共享單車使用量與風速



圖十二 共享單車使用量與天氣類型

```
plt.figure(figsize=(20,8))
sns.heatmap(df.corr(),cmap="YlGnBu",square=True,linewidths=.5,center=0,annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f280cf267d0>
```



圖十三 各屬性間相關係數熱圖

(二) 資料切割與標準化

將資料依時間序列拆分 80%的訓練集及 20%的測試集。

```
import math

training_data_len = math.ceil(len(df) * 0.8)
testing_data_len = len(df) - training_data_len

time_steps = 24
train,test=df.iloc[0:training_data_len],df.iloc[(training_data_len-time_steps):len(df)]
```

利用 sklearn.preprocessing 中的 RobustScaler()函數，可縮放有離群值的數據，分別將訓練集與資料集進行特徵縮放。

```
from sklearn.preprocessing import RobustScaler

train_trans = train[['t1', 't2', 'hum', 'wind_speed','weather_code','is_holiday','is_weekend','season']].to_numpy()
test_trans = test[['t1', 't2', 'hum', 'wind_speed','weather_code','is_holiday','is_weekend','season']].to_numpy()

scaler = RobustScaler()
train.loc[:,['t1','t2','hum','wind_speed','weather_code','is_holiday','is_weekend','season']]=scaler.fit_transform(train_trans)
test.loc[:,['t1','t2','hum','wind_speed','weather_code','is_holiday','is_weekend','season']]=scaler.fit_transform(test_trans)

train['cnt'] = scaler.fit_transform(train[['cnt']])
test['cnt'] = scaler.fit_transform(test[['cnt']])
```

將訓練集與測試集分割出因子 x 與目標值 y，將 x 資料轉為 3 維(樣本數、時間長度和特徵)、型態轉變為 array，供後續放入模型進行訓練與預測。

```
from tqdm import tqdm_notebook as tqdm
x_train = []
y_train = []
for i in tqdm(range(len(train) - time_steps)):
    x_train.append(train.drop(columns='cnt').iloc[i:i+time_steps].to_numpy())
    y_train.append(train.loc[:,'cnt'].iloc[i + time_steps])
x_train = np.array(x_train)
y_train = np.array(y_train)

x_test = []
y_test = df.loc[:,'cnt'].iloc[training_data_len:len(df)]
for i in tqdm(range(len(test) - time_steps)):
    x_test.append(test.drop(columns='cnt').iloc[i:i+time_steps].to_numpy())
x_test = np.array(x_test)
y_test = np.array(y_test)
```

二、深度學習模型建立

首先設定隨機種子以便後續追蹤模型結果。以Sequential()建立序列模型，而後將LSTM層加到模型中，並使用dropout在每一次訓練的迭代(epoch)皆以一定的機率丟棄隱藏層神經元，讓他們在向前傳播時不會傳遞訊息，以防止過度擬合(overfitting)的發生。最後增加Dense全連接層，將輸出的維度降到1維，即預測的結果。

```
tensorflow.random.set_seed(1)
from  keras.models  import  Sequential
from  keras.layers  import  Dense,  Dropout  ,  LSTM
model  =  Sequential()
model.add(LSTM(50,input_shape=(x_train.shape[1],x_train.shape[2])))
model.add(Dropout(0.2))
model.add(Dense(units=1))
model.summary()
```

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 lstm_2 (LSTM)                (None, 50)                12600

 dropout_2 (Dropout)          (None, 50)                0

 dense_2 (Dense)              (None, 1)                 51

=================================================================
Total params: 12,651
Trainable params: 12,651
Non-trainable params: 0
_________________________________________________________________
```

利用compile配置學習過程，進行模型編譯，以mse為loss function衡量模型準確度。之後以fit進行模型訓練，此處固定進行20次迭代，並從訓練集中取出10%作為驗證集。

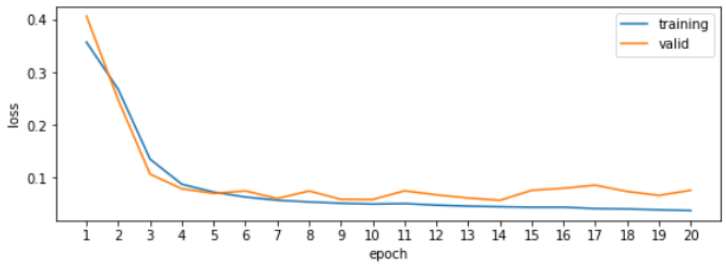
```
model.compile(optimizer='adam',loss="mse")

history  =  model.fit(x_train,  y_train,  epochs=20,
                      batch_size=24,validation_split=0.1,  shuffle=True)
```

```
Epoch 1/20
522/522 [==============================] - 9s 13ms/step - loss: 0.3575 - val_loss: 0.4071
Epoch 2/20
522/522 [==============================] - 7s 13ms/step - loss: 0.2678 - val_loss: 0.2476
Epoch 3/20
522/522 [==============================] - 7s 13ms/step - loss: 0.1351 - val_loss: 0.1067
Epoch 4/20
522/522 [==============================] - 7s 13ms/step - loss: 0.0874 - val_loss: 0.0787
Epoch 5/20
522/522 [==============================] - 7s 13ms/step - loss: 0.0725 - val_loss: 0.0700
Epoch 6/20
522/522 [==============================] - 7s 13ms/step - loss: 0.0630 - val_loss: 0.0747
Epoch 7/20
522/522 [==============================] - 7s 13ms/step - loss: 0.0571 - val_loss: 0.0603
Epoch 8/20
522/522 [==============================] - 7s 13ms/step - loss: 0.0538 - val_loss: 0.0746
Epoch 9/20
522/522 [==============================] - 7s 13ms/step - loss: 0.0512 - val_loss: 0.0587
Epoch 10/20
522/522 [==============================] - 7s 13ms/step - loss: 0.0497 - val_loss: 0.0584
Epoch 11/20
522/522 [==============================] - 7s 13ms/step - loss: 0.0506 - val_loss: 0.0749
Epoch 12/20
522/522 [==============================] - 6s 12ms/step - loss: 0.0477 - val_loss: 0.0673
Epoch 13/20
522/522 [==============================] - 6s 12ms/step - loss: 0.0458 - val_loss: 0.0612
Epoch 14/20
522/522 [==============================] - 6s 12ms/step - loss: 0.0446 - val_loss: 0.0568
Epoch 15/20
522/522 [==============================] - 7s 13ms/step - loss: 0.0437 - val_loss: 0.0759
Epoch 16/20
522/522 [==============================] - 7s 13ms/step - loss: 0.0438 - val_loss: 0.0798
Epoch 17/20
522/522 [==============================] - 7s 13ms/step - loss: 0.0411 - val_loss: 0.0857
Epoch 18/20
522/522 [==============================] - 7s 13ms/step - loss: 0.0404 - val_loss: 0.0736
Epoch 19/20
522/522 [==============================] - 6s 12ms/step - loss: 0.0387 - val_loss: 0.0662
Epoch 20/20
522/522 [==============================] - 7s 13ms/step - loss: 0.0373 - val_loss: 0.0759
```

將loss大小以圖表顯示，可以發現在第6個epoch後就收斂趨於穩定。

```
loss = history.history['loss']
epoch = range(1,21,1)

import matplotlib.pyplot as plt
plt.figure(figsize=(20,3))
plt.subplot(121)
line1,=plt.plot(epoch,loss,label ='training')
line2,=plt.plot(epoch,history.history['val_loss'],label ='valid')
plt.xlabel('epoch')
plt.ylabel('loss')
plt.xticks(np.arange(1,21,1))
plt.legend(handles = [line1, line2], loc='upper right')
```

```
<matplotlib.legend.Legend at 0x7f91b82128d0>
```



以測試集進行模型測試，其預測的誤差(rmse)為0.2624。

```
y_pred = model.predict(x_test)
from sklearn.metrics import mean_squared_error, r2_score
rmse_lstm = np.sqrt(mean_squared_error(y_test, y_pred))
rmse_lstm
```

```
0.26242217143978125
```

三、超參數優化與實驗結果

對於超參數的選擇，本研究以實驗設計的方法，選用四項因子與三個水準，以L9直交表進行實驗，希望可以在有限的實驗次數中找到最好的模型參數，以達到最高的準確率。其各項因子與水準如表三所示，藉由L9直交表的參數組合進行實驗的結果如表四所示。

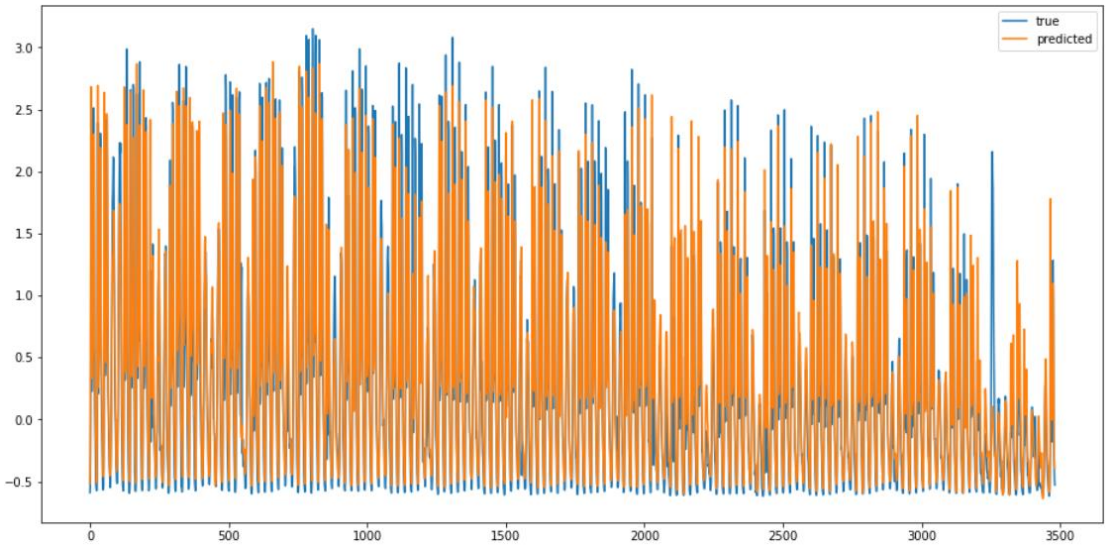
表三 實驗因子與水準

因子	項目	水準1	水準2	水準3
A	Dropout	0.1	0.2	0.3
B	Optimizer	adam	rmsprop	Sgd
C	Activation	relu	sigmoid	Tanh
D	LSTM(Units)	50	60	48

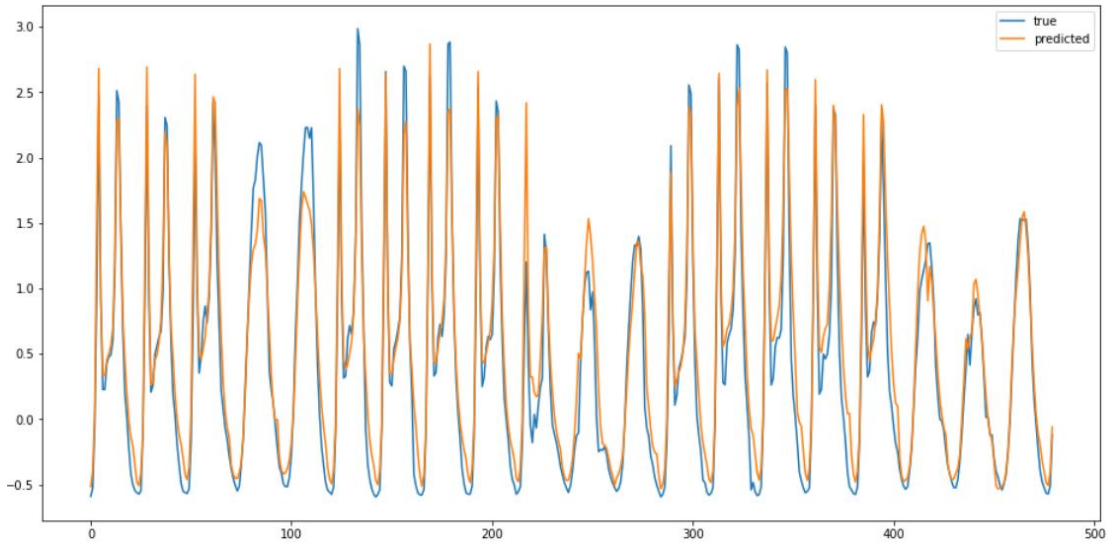
表四 L9直交表超參數組合與實驗結果

實驗	Dropout	Optimizer	Activation	LSTM(units)	誤差(rmse)
1	0.1	Adam	Relu	50	0.25
2	0.1	Rmaprop	Sigmoid	60	0.233
3	0.1	Sgd	Tanh	48	0.4912
4	0.2	Adam	Sigmoid	48	0.2296
5	0.2	Rmsprop	Tanh	50	0.2353
6	0.2	Sgd	Relu	60	1.0435
7	0.3	Adam	Tanh	60	0.2372
8	0.3	Rmsprop	Relu	48	0.2458
9	0.3	sgd	sigmoid	50	0.592

根據L9直交表進行實驗的結果，在不同的超參數組合下，獲得的誤差值有很大的落差。在實驗4時有最小的誤差值0.2296，因此選擇Dropout為0.1、Optimizer為adam、Activation為Sigmoid、LSTM的units為48的作為模型的超參數組合。模型在測試集的預測結果如圖十四所示，近一步觀察其前20天的預測結果，可以發現模型大致預測出資料趨勢，如圖十五所示。



圖十四 測試集預測結果



圖十五 測試集前20天(480小時)預測結果

伍、結論與未來展望

本研究希望能建立一個共享單車的預測模型，提供共享單車供應商在調度站點車輛的參考依據。在個案研究中可以發現共享單車的使用量和天氣狀況(溫度、濕度)以及小時(在一天中的甚麼時後)有較大的關係，符合常理。然而，此資料集來源為倫敦地區，可能會因為文化與生活差異，在各國有不同的使用量週期分布，無法適用於各個地區。再者，近期因為疫情興起，對於共享單車的使用量應有極大的影響，若能取得近期資料，可以測試模型能否預測出準確的使用量。而LSTM雖然能夠預測出共享單車數量的趨勢，但與實際的數量還是有落差，未來可以嘗試使用不同的演算法進行預測，找出最適當的預測模型。

參考資料

Kaggle-London bike sharing dataset
https://www.kaggle.com/hmavrodiev/london-bike-sharing-dataset

【三個共享單車產業發展關鍵】從共享單車發展歷史，看科技如何重新形塑城市文化？
https://buzzorange.com/techorange/2017/08/29/bike-shring-history/

[Day-16] RNN - LSTM 介紹
https://ithelp.ithome.com.tw/articles/10223055

Scikit-Learn sklearn.preprocessing.RobustScaler
https://hackmd.io/@shaoeChen/r1CQ9VY98