應用LSTM預測共享單車需求

110034540 徐阡珊

目　　錄

簡　　介

背景動機

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

有樁式共享單車
無車可借，無柱可還

研究目的

透過歷史資料，預測未來共享單車需求數量，
調度站點單車數量，並提供採購單車數量的依據，
以滿足使用者需求

簡　　介

5W1H

What	/	預測共享單車需求量
Why	/	滿足使用者需求
Where	/	共享單車供應商總部
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進行使用量預測

實 驗 流 程

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)

Kaggle
London bike sharing dataset

倫敦交通局(Transport for London, TfL) 公開資料

	timestamp	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weekend	season
0	2015-01-04 00:00:00	182	3.0	2.0	93.0	6.0	3.0	0.0	1.0	3.0
1	2015-01-04 01:00:00	138	3.0	2.5	93.0	5.0	1.0	0.0	1.0	3.0
2	2015-01-04 02:00:00	134	2.5	2.5	96.5	0.0	1.0	0.0	1.0	3.0
3	2015-01-04 03:00:00	72	2.0	2.0	100.0	0.0	1.0	0.0	1.0	3.0
4	2015-01-04 04:00:00	47	2.0	0.0	93.0	6.5	1.0	0.0	1.0	3.0
5	2015-01-04 05:00:00	46	2.0	2.0	93.0	4.0	1.0	0.0	1.0	3.0
6	2015-01-04 06:00:00	51	1.0	-1.0	100.0	7.0	4.0	0.0	1.0	3.0
7	2015-01-04 07:00:00	75	1.0	-1.0	100.0	7.0	4.0	0.0	1.0	3.0
8	2015-01-04 08:00:00	131	1.5	-1.0	96.5	8.0	4.0	0.0	1.0	3.0
9	2015-01-04 09:00:00	301	2.0	-0.5	100.0	9.0	3.0	0.0	1.0	3.0

實驗流程

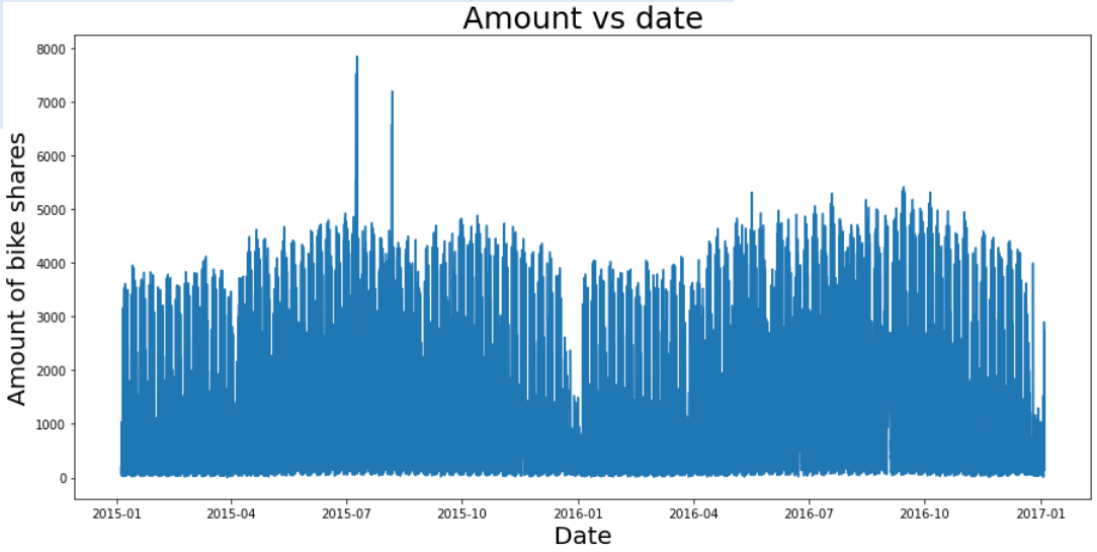
資料缺失值

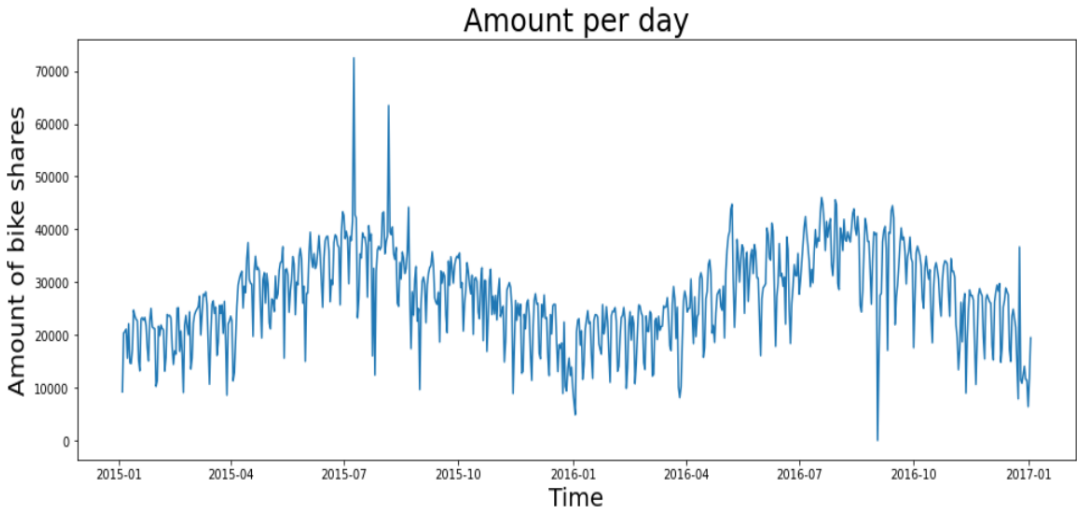
```
df.info()
```

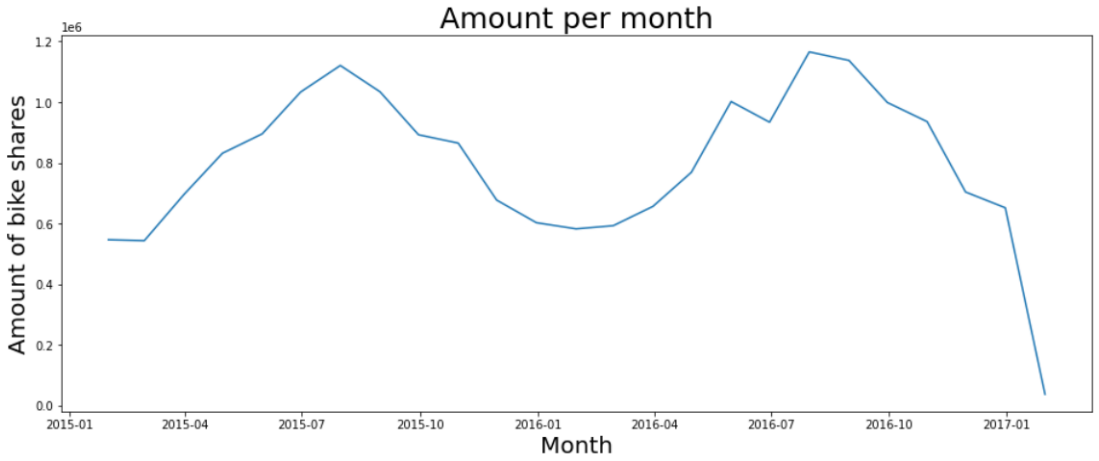
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17414 entries, 0 to 17413
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   timestamp     17414 non-null  object 
 1   cnt           17414 non-null  int64  
 2   t1            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
```

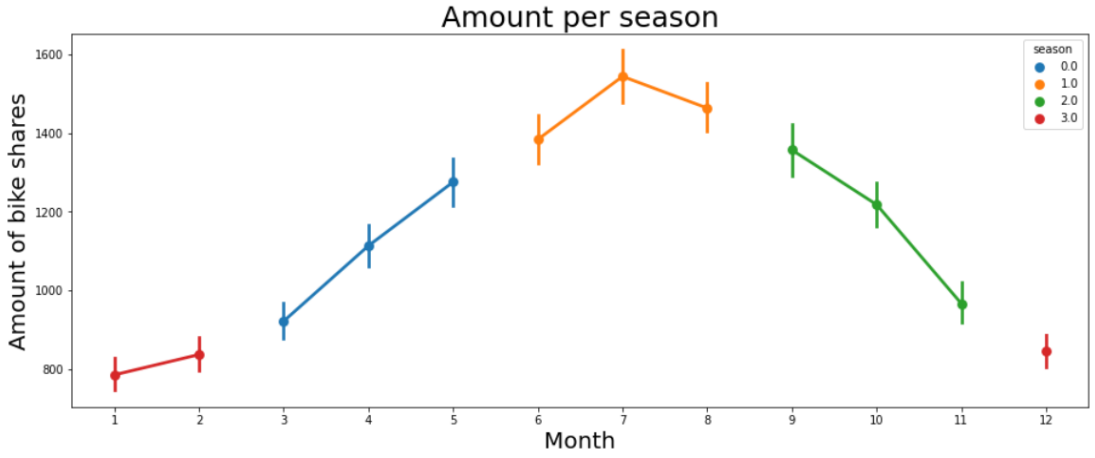
實驗流程

時間趨勢


Amount vs date
Amount of bike shares
Date
0
1000
2000
3000
4000
5000
6000
7000
8000
2015-01
2015-04
2015-07
2015-10
2016-01
2016-04
2016-07
2016-10
2017-01

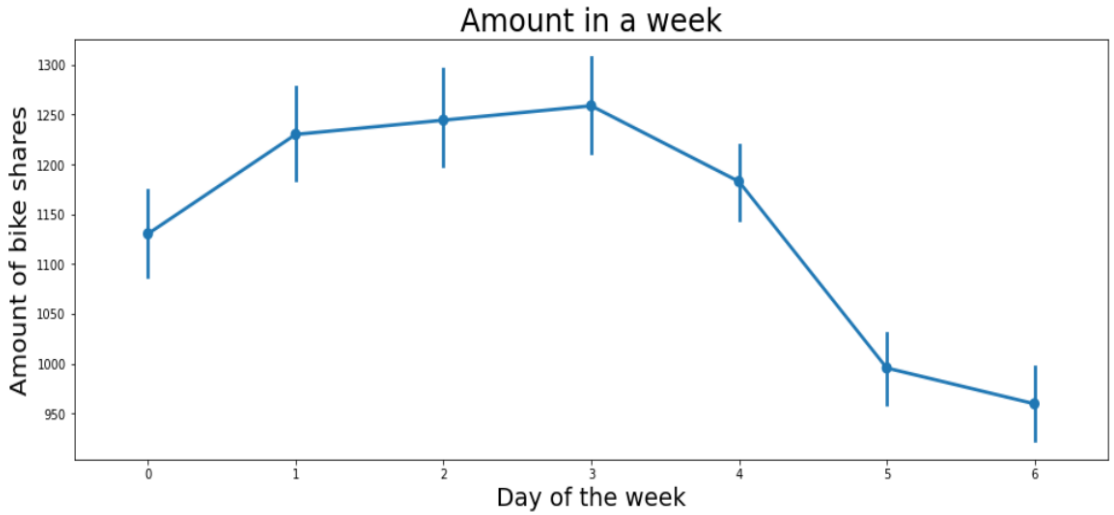

Amount per day
Amount of bike shares
Time
0
10000
20000
30000
40000
50000
60000
70000
2015-01
2015-04
2015-07
2015-10
2016-01
2016-04
2016-07
2016-10
2017-01


Amount per month
1e6
Amount of bike shares
Month
0.0
0.2
0.4
0.6
0.8
1.0
1.2
2015-01
2015-04
2015-07
2015-10
2016-01
2016-04
2016-07
2016-10
2017-01

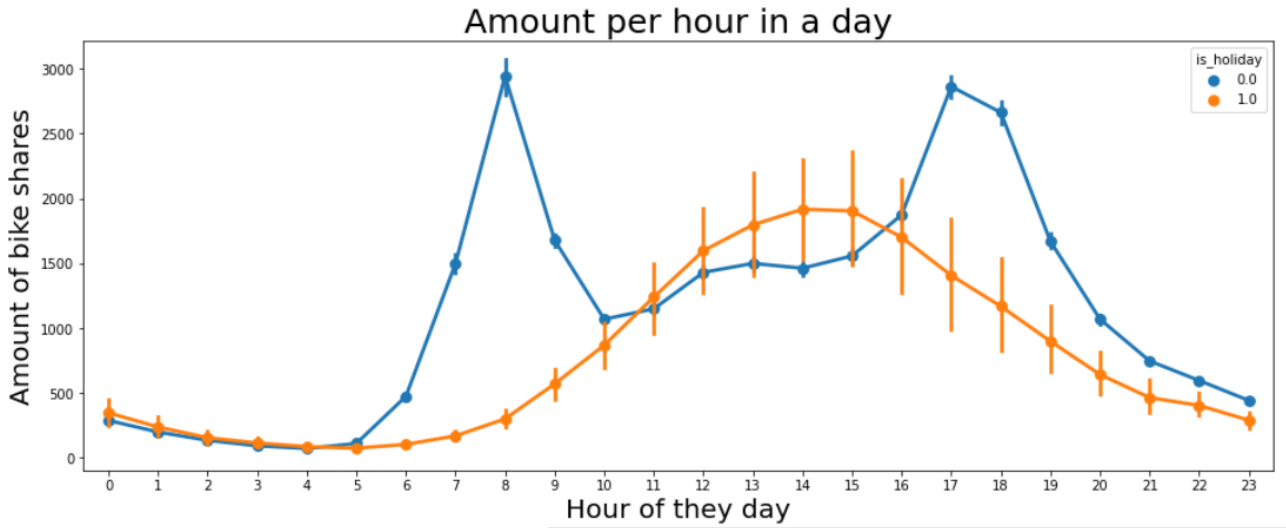

Amount per season
Amount of bike shares
Month
season
0.0
1.0
2.0
3.0
800
1000
1200
1400
1600
1
2
3
4
5
6
7
8
9
10
11
12

實驗流程

時間趨勢

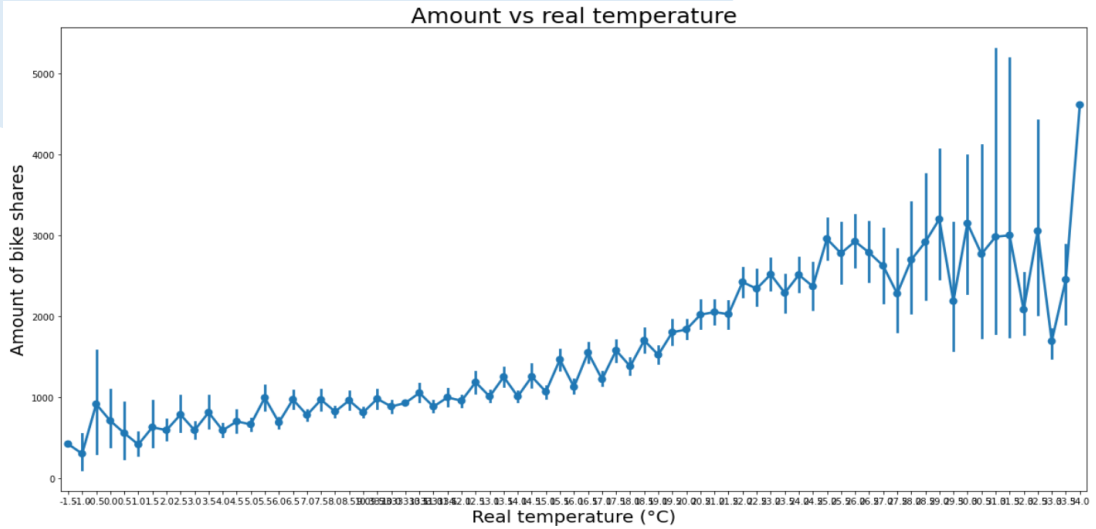


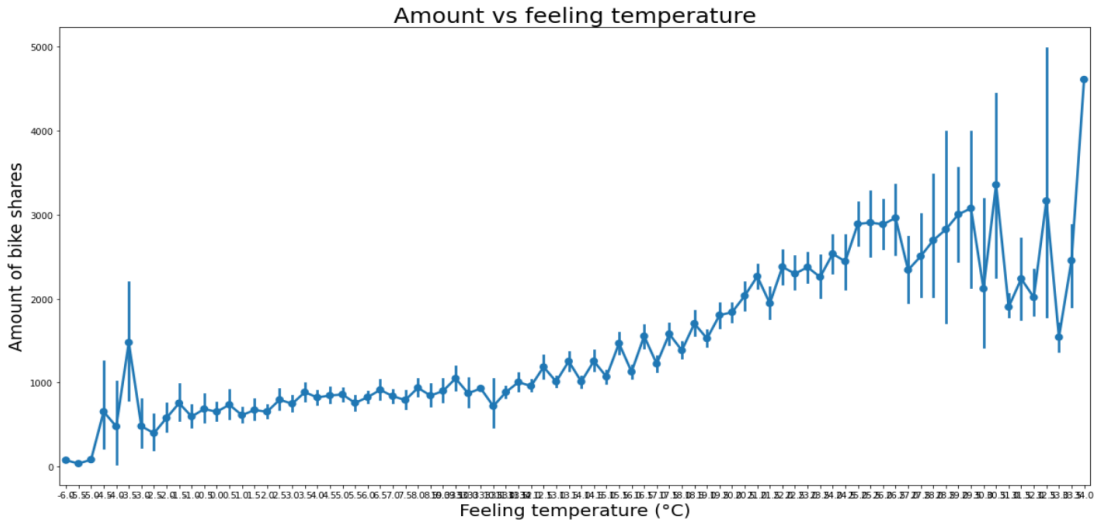
0：星期一　　6：星期日

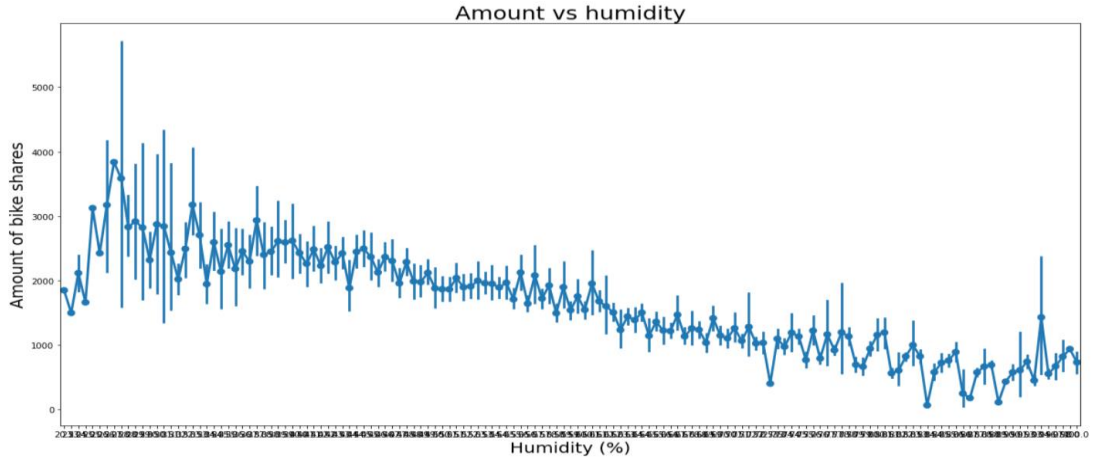


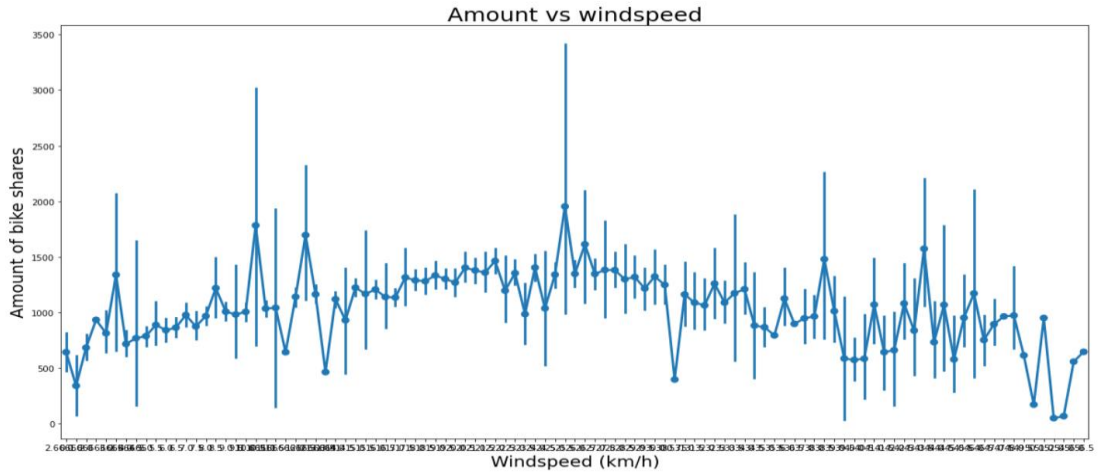
實驗流程

屬性關係


Amount vs real temperature
Amount of bike shares
Real temperature (°C)

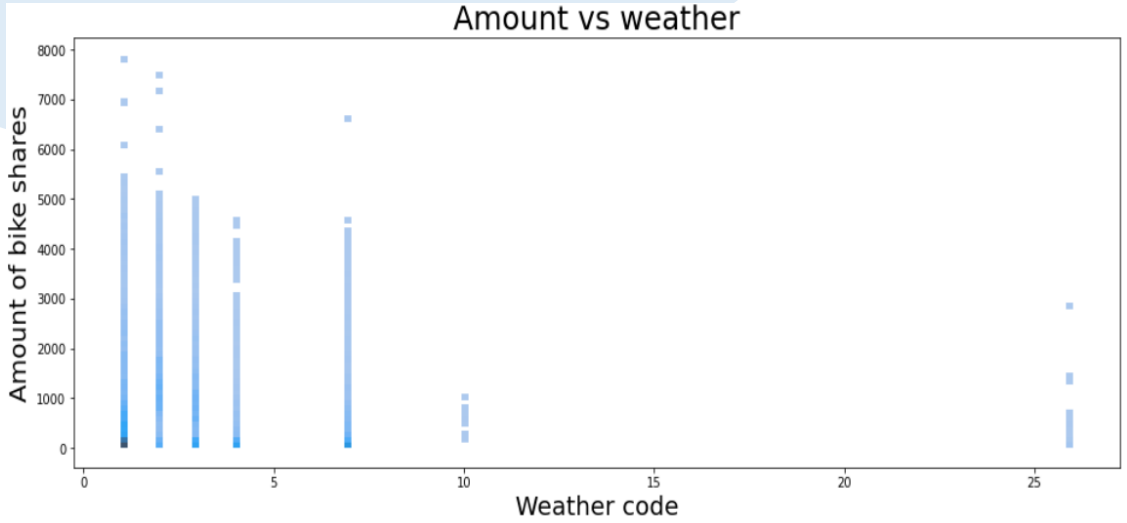

Amount vs feeling temperature
Amount of bike shares
Feeling temperature (°C)


Amount vs humidity
Amount of bike shares
Humidity (%)


Amount vs windspeed
Amount of bike shares
Windspeed (km/h)

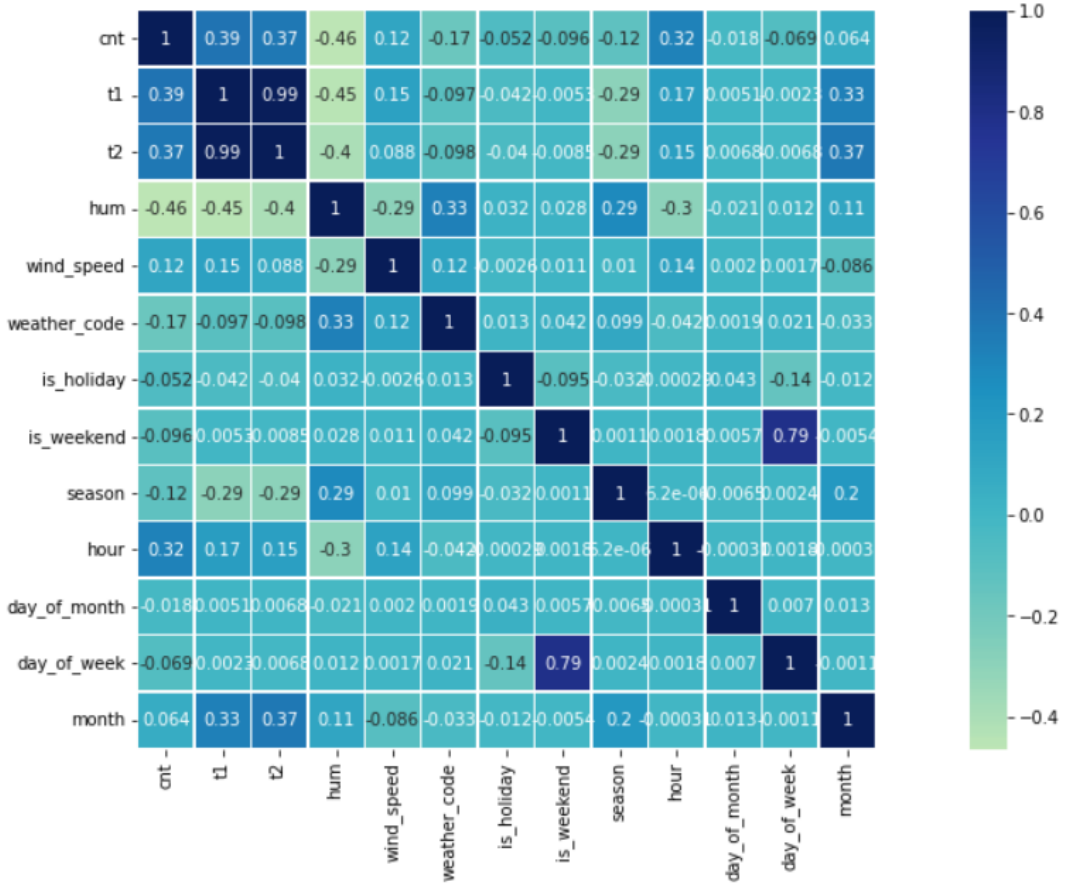
實驗流程

屬性關係



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



實驗流程

資料切割

```
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)]
```

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']])
```

型態轉換

```
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)
```

實驗流程

模型建立

```
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
_________________________________________________________________
```

模型訓練

```
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
```

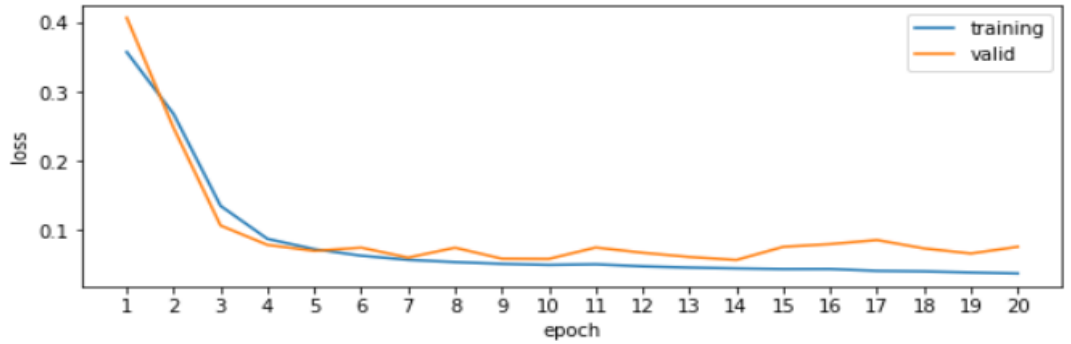
實驗流程

Loss趨勢

```
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>
```



預測誤差

```
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
```

超參數優化

四因子&三水準

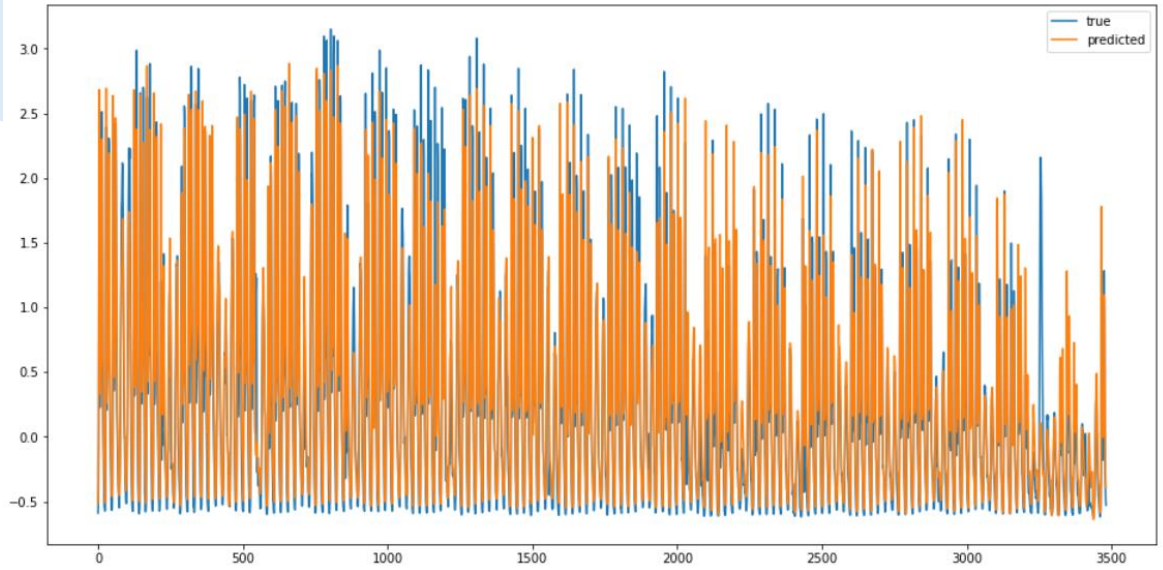
因子	項目	水準1	水準2	水準3
A	Dropout	0.1	0.2	0.3
B	Optimizer	adam	rmsprop	Sgd
C	Activation	relu	sigmoid	Tanh
D	LSTM(Units)	50	60	48

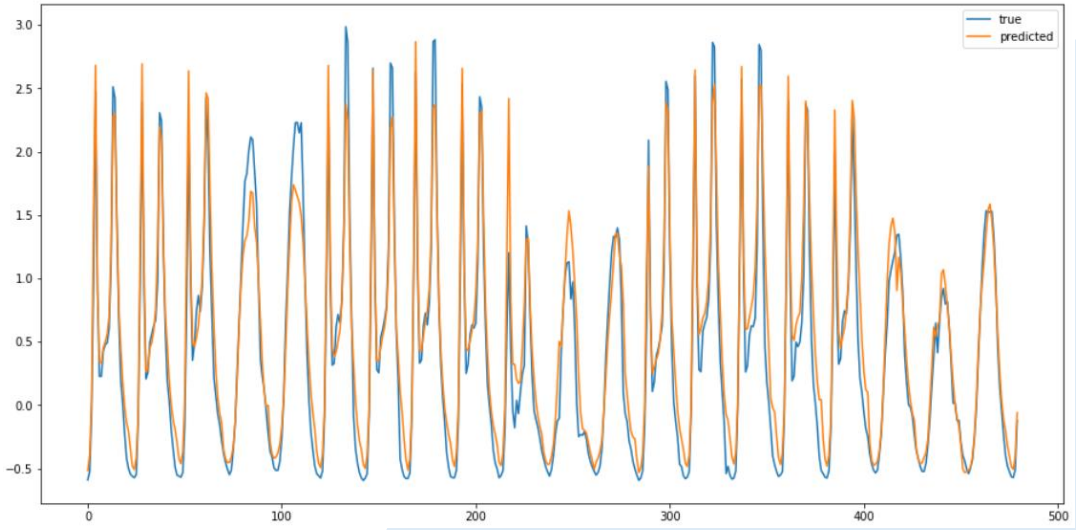
實驗	Dropout	Optimizer	Activation	LSTM(units)	誤差(rmse)
1	0.1	Adam	Relu	50	0.25
2	0.1	Rmaprop	Sigmoid	60	0.233
3	0.1	Sgd	Tanh	48	0.4912
4	0.2	Adam	Sigmoid	48	0.2296
5	0.2	Rmsprop	Tanh	50	0.2353
6	0.2	Sgd	Relu	60	1.0435
7	0.3	Adam	Tanh	60	0.2372
8	0.3	Rmsprop	Relu	48	0.2458
9	0.3	sgd	sigmoid	50	0.592

L_9 直交表

實驗流程

預測結果


true
predicted
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
0
500
1000
1500
2000
2500
3000
3500


true
predicted
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
0
100
200
300
400
500

結論&未來展望

相關指標

天氣狀況

多數人生活作息

演算法適用性

現有數據為2015-2017

能否預測出疫情影響的使用量

地區差異

倫敦資料

生活文化差異

Thanks

Q & A