數據科學專題

Predict Future Sales

第三組

李佳揚

國立中正大學經濟學系四年級

406510106

江子瑜

國立中正大學企業管理學系四年級

406520083

吳翔詠

國立中正大學經濟學系三年級

407510062

江彥亨

國立中正大學經濟學系三年級

407510024

關鍵詞:銷售預測、機器學習、XGB、LightGBM、深度學習

1. 前言

此為程式平台 KAGGLE 中的一個競賽「預測未來銷售」，提供每日歷史銷售數據，旨在預測下個月每種產品和商店的總銷售額。

由於商店和產品列表每個月都會略有變化，因此我們的研究目的為透過不同的演算法和資料處理，用現階段我們對數據科學的理解，不斷嘗試各種方法去解讀資料，藉以達成在學習的同時，提升數據科學技能的應用能力。

2. 探索性資料分析

2-1 資料介紹

「預測未來銷售」的資料集為來自俄羅斯最大的軟體公司，1C Company，所抓取的每日歷史銷售數據，其原始提供的欄位如下圖一。

圖一 數據資訊

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2935849 entries, 0 to 2935848
Data columns (total 10 columns):
 #   Column              Dtype
---  ------              -----
 0   date                datetime64[ns]
 1   date_block_num      int64
 2   shop_id             int64
 3   item_id             int64
 4   item_price          float64
 5   item_cnt_day        float64
 6   item_name           object
 7   item_category_id    int64
 8   item_category_name  object
 9   shop_name           object
dtypes: datetime64[ns](1), float64(2), int64(4), object(3)
memory usage: 246.4+ MB
```

ID ：表示測試集中（Shop, Item）的編號

shop_id：商店編號

item_id ：產品編號

item_category_id：產品分類編號

item_cnt_day：日銷售量

item_price ：產品現價

date：日期

date_block_num ：連續的月份編碼，便於分析，如將 2013 的 1 月視為 0…….

則 2015 的 10 月為 33，以此類推。

item_name：產品名稱

shop_name ：商店名稱

item_category_name：產品分類名稱

2-2 資料匯入

首先我們先匯入需要用到和可能會用到的套件，如圖二。

圖二 匯入套件

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from collections import Counter
from scipy import stats
import random

# Standard plotly imports
import chart_studio.plotly as py
import plotly.graph_objects as go
import chart_studio.tools as tls
import plotly.tools as pls
from plotly.offline import iplot, init_notebook_mode
import cufflinks
import cufflinks as cf

#import lightgbm
from functools import partial
from hyperopt import fmin, hp, tpe, Trials, space_eval, STATUS_OK, STATUS_RUNNING

# Using plotly + cufflinks in offline mode
init_notebook_mode(connected=True)
cufflinks.go_offline(connected=True)

import gc
import warnings
from itertools import product
warnings.filterwarnings("ignore")
```

接著我們進行匯入的動作，如圖三。可以注意到，因為是競賽的資料且目的在預測，所以原先就有給定訓練集和測試集。

圖三 資料匯入

```
df_train = pd.read_csv('data\sales_train.csv')

df_categories = pd.read_csv("data\item_categories.csv")
df_items = pd.read_csv("data\items.csv")
df_shops = pd.read_csv("data\shops.csv")

df_test = pd.read_csv("data/test.csv")
```

觀察資料，圖四為訓練集資料的數據框，圖五則為產品的數據框。

圖四 訓練集數據框

df_train

	date	date_block_num	shop_id	item_id	item_price	item_cnt_day
0	02.01.2013	0	59	22154	999.00	1.0
1	03.01.2013	0	25	2552	899.00	1.0
2	05.01.2013	0	25	2552	899.00	-1.0
3	06.01.2013	0	25	2554	1709.05	1.0
4	15.01.2013	0	25	2555	1099.00	1.0
...	...	...	...	...	...	...
2935844	10.10.2015	33	25	7409	299.00	1.0
2935845	09.10.2015	33	25	7460	299.00	1.0
2935846	14.10.2015	33	25	7459	349.00	1.0
2935847	22.10.2015	33	25	7440	299.00	1.0
2935848	03.10.2015	33	25	7460	299.00	1.0

2935849 rows × 6 columns

圖五 產品數據框

df_items

	item_name	item_id	item_category_id
0	! ВО ВЛАСТИ НАВАЖДЕНИЯ (ПЛАСТ.) D	0	40
1	!ABBYY FineReader 12 Professional Edition Full...	1	76
2	***В ЛУЧАХ СЛАВЫ (UNV) D	2	40
3	***ГОЛУБАЯ ВОЛНА (Univ) D	3	40
4	***КОРОБКА (СТЕКЛО) D	4	40
...	...	...	...
22165	Ядерный титбит 2 [PC, Цифровая версия]	22165	31
22166	Язык запросов 1С:Предприятия [Цифровая версия]	22166	54
22167	Язык запросов 1С:Предприятия 8 (+CD). Хрустале...	22167	49
22168	Яйцо для Little Inu	22168	62
22169	Яйцо дракона (Игра престолов)	22169	69

22170 rows × 3 columns

將數據框如圖五的形式，把原本給定的訓練集、產品、類別以及商店合成新的訓練集資料。

圖六 合併數據框

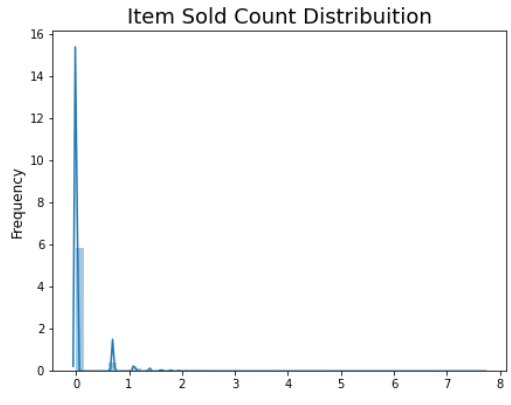
```
df_train = pd.merge(df_train, df_items, on='item_id', how='inner')
df_train = pd.merge(df_train, df_categories, on='item_category_id', how='inner')
df_train = pd.merge(df_train, df_shops, on='shop_id', how='inner')

df_test = pd.merge(df_test, df_items, on='item_id', how='inner')
df_test = pd.merge(df_test, df_categories, on='item_category_id', how='inner')
df_test = pd.merge(df_test, df_shops, on='shop_id', how='inner')

# del df_items, df_categories, df_shops
#清理內存
gc.collect()
```

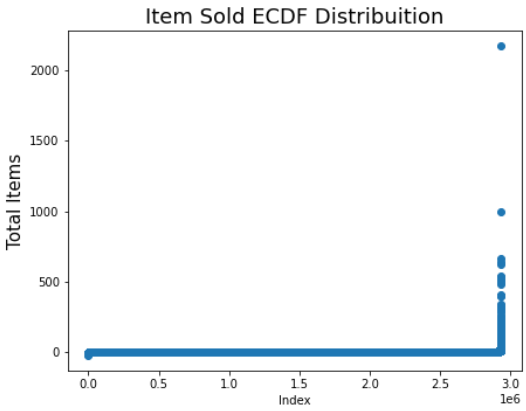
2-3 資料視覺化

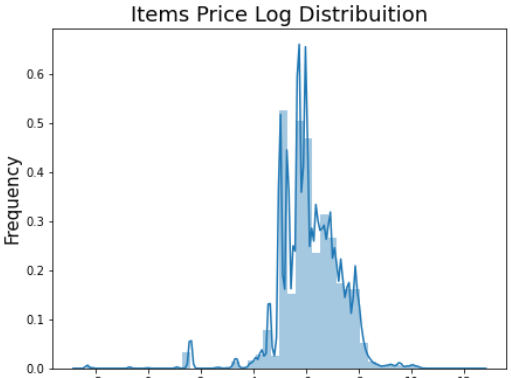
再經過閱讀資料後，我們將資料用視覺化的方式呈現，一來可以用更簡潔有力的方式，達到方便觀察的益處，二來可以藉由控制變數，來達成輔助分析的功用。

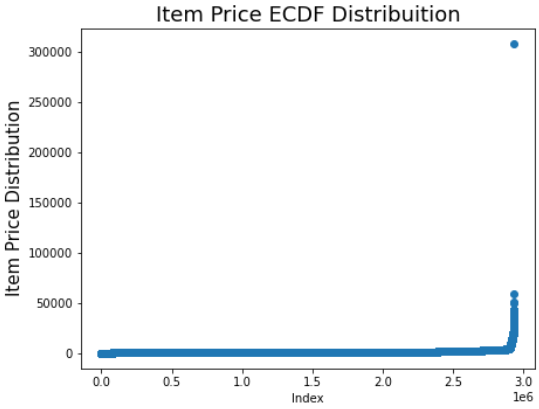
如下圖七，我們開始探索我們的目標變數，每日銷售量，下圖繪製的是已售出的商品和商品價格間的關係，注意右半部份的經驗累積分布函數圖之 x 軸的單位為等比縮小百萬分之一，意即數字 3 表三百萬的銷售額，由此圖我們可以得知，幾乎所有的商品一次的銷售量都為 1，接著我們再針對分位數進行描述。

圖七









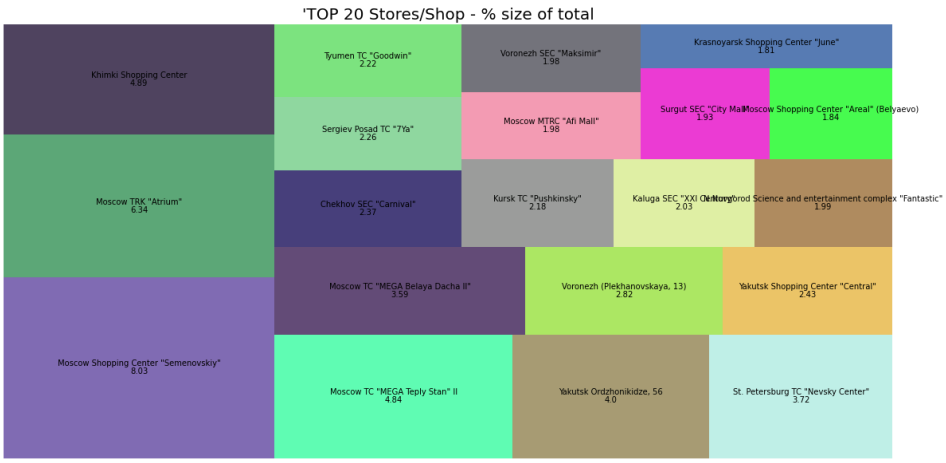
新增每日的銷售額的欄位，公式是數量乘上價格，接著顯示出各個欄位的百分位數，如圖八。

圖八

```
[ ] df_train['total_amount'] = df_train['item_price'] * df_train['item_cnt_day']

#顯示出各種百分位數
quantiles(df_train, ['item_cnt_day','item_price', 'total_amount'])

item_cnt_day quantiles
0.01    1.0
0.25    1.0
0.50    1.0
0.75    1.0
0.99    5.0
Name: item_cnt_day, dtype: float64

item_price quantiles
0.01       5.0
0.25     249.0
0.50     399.0
0.75     999.0
0.99    5999.0
Name: item_price, dtype: float64

total_amount quantiles
0.01       18.0
0.25      249.0
0.50      449.0
0.75     1078.2
0.99    11375.0
Name: total_amount, dtype: float64
```

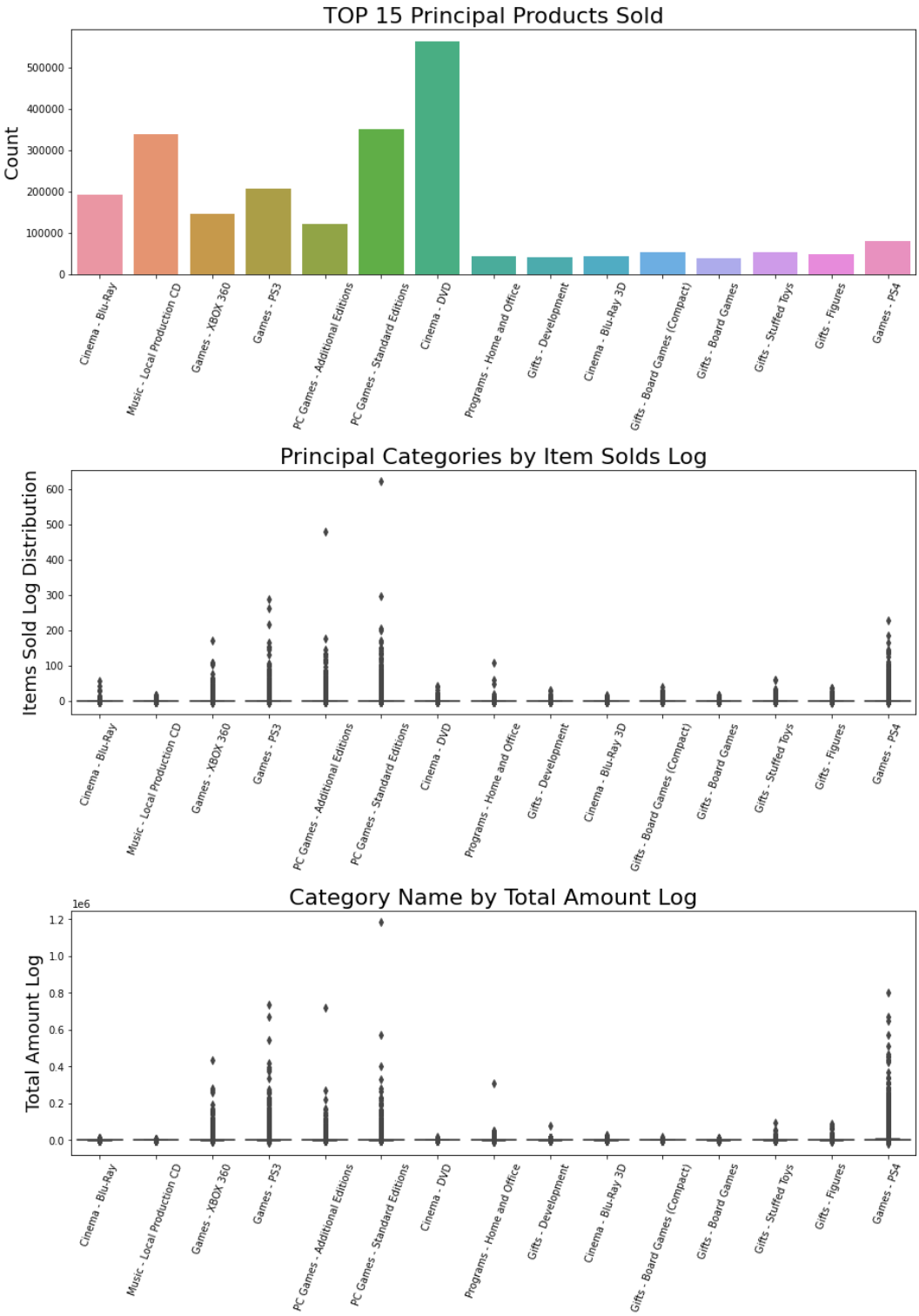
接著我們針對商店這個欄位進行觀察，由圖九可以得知，這是商店的市場規模，我們透過這張圖可以觀察出軟體市場的型態，目測並無特定獨佔的情形。

圖九



分析產品分類，透過圖十去觀察前十五項最熱門的產品，其中最熱門的是電影的 DVD，其次為電腦遊戲和音樂 CD，這相當具有意義，我們得知了銷量最好的產品，隨即我們會再針對這些產品的價格進行延伸的觀察。

圖十


TOP 15 Principal Products Sold
Count
0
100000
200000
300000
400000
500000
Cinema - Blu-Ray
Music - Local Production CD
Games - XBOX 360
Games - PS3
PC Games - Additional Editions
PC Games - Standard Editions
Cinema - DVD
Programs - Home and Office
Gifts - Development
Cinema - Blu-Ray 3D
Gifts - Board Games (Compact)
Gifts - Board Games
Gifts - Stuffed Toys
Gifts - Figures
Games - PS4
Principal Categories by Item Solds Log
Items Sold Log Distribution
0
100
200
300
400
500
600
Cinema - Blu-Ray
Music - Local Production CD
Games - XBOX 360
Games - PS3
PC Games - Additional Editions
PC Games - Standard Editions
Cinema - DVD
Programs - Home and Office
Gifts - Development
Cinema - Blu-Ray 3D
Gifts - Board Games (Compact)
Gifts - Board Games
Gifts - Stuffed Toys
Gifts - Figures
Games - PS4
Category Name by Total Amount Log
1e6
Total Amount Log
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Cinema - Blu-Ray
Music - Local Production CD
Games - XBOX 360
Games - PS3
PC Games - Additional Editions
PC Games - Standard Editions
Cinema - DVD
Programs - Home and Office
Gifts - Development
Cinema - Blu-Ray 3D
Gifts - Board Games (Compact)
Gifts - Board Games
Gifts - Stuffed Toys
Gifts - Figures
Games - PS4

得知銷量分佈後我們想要接著延續觀察，去看看銷售量、價格之間的關係，如下圖十一。我們把銷售量前五千的產品列出來，再利用敘述統計來分析前五千項最貴的商品，如下圖十二。

圖十一

```
sub_categorys_5000 = df_train.sort_values('total_amount',
                                          ascending=False)[['item_category_name', 'item_name',
                                                            'shop_name',
                                                           'item_cnt_day','item_price',
                                                           'total_amount']].head(5000)
sub_categorys_5000.head(10)
```

	item_category_name	item_name	shop_name	item_cnt_day	item_price	total_amount
1068934	Game consoles - PS4	Sony PlayStation 4 (500 Gb) Black (CUH-1008A/1...	St. Petersburg TC "Nevsky Center"	101.0	18118.712871	1.829990e+06
815919	Game consoles - PS4	Sony PlayStation 4 (500 Gb) Black (CUH-1008A/1...	Moscow Shopping Center "Semenovskiy"	90.0	18245.555556	1.642100e+06
378795	Game consoles - PS4	Sony PlayStation 4 (500 Gb) Black (CUH-1008A/1...	Moscow TRK "Atrium"	85.0	18190.000000	1.546150e+06
2695457	Game consoles - PS4	Sony PlayStation 4 (500 Gb) Black (CUH-1008A/1...	Online store of emergencies	84.0	18073.690476	1.518190e+06
2119997	Game consoles - PS4	Sony PlayStation 4 (500 Gb) Black (CUH-1008A/1...	Krasnoyarsk Shopping Center "June"	73.0	18305.068493	1.336270e+06
2695470	Game consoles - PS4	Sony PlayStation 4 (500 Gb) Black (CUH-1008A/1...	Online store of emergencies	52.0	24278.461538	1.262480e+06
465999	Game consoles - PS4	Sony PlayStation 4 (500 Gb) Black (CUH-1008A/1...	Voronezh (Plekhanovskaya, 13)	67.0	18198.955224	1.219330e+06
2682591	PC Games - Collectible Editions	Ведьмак 3: Дикая охота. Коллекционное издание ...	Online store of emergencies	140.0	8499.000000	1.189860e+06
2676429	PC Games - Standard Editions	Grand Theft Auto V [PC, русские субтитры]	Online store of emergencies	624.0	1904.548077	1.188438e+06
2698367	Service Tools - Tickets	Билет "ИгроМир 2015" - 3 октября 2015 (сайт) [...	Online store of emergencies	669.0	1692.526158	1.132300e+06

圖十二

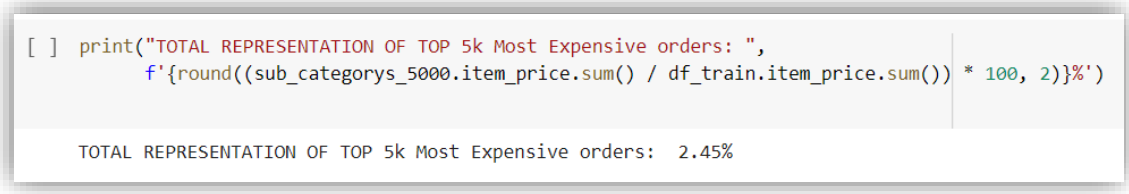
```
sub_categorys_5000.describe(include='all')
```

	item_category_name	item_name	shop_name	item_cnt_day	item_price	total_amount
count	5000	5000	5000	5000.000000	5000.000000	5.000000e+03
unique	35	321	54	NaN	NaN	NaN
top	Game consoles - PS4	Sony PlayStation 4 (500 Gb) Black (CUH-1008A/1...	Moscow TRK "Atrium"	NaN	NaN	NaN
freq	1988	1460	383	NaN	NaN	NaN
mean	NaN	NaN	NaN	17.900400	12833.009561	7.686763e+04
std	NaN	NaN	NaN	36.256439	11246.963467	1.043254e+05
min	NaN	NaN	NaN	1.000000	155.192950	3.020150e+04
25%	NaN	NaN	NaN	2.000000	2753.370629	3.789975e+04
50%	NaN	NaN	NaN	8.000000	10490.000000	4.798400e+04
75%	NaN	NaN	NaN	20.000000	22490.000000	7.296000e+04
max	NaN	NaN	NaN	669.000000	307980.000000	1.829990e+06

最後我們分析這些銷售對總利潤的佔比，如下圖十三，可知這前五千的商品，僅僅只佔了總銷售量的 2.45%，此時我們可以得出一個小結論，意即這個

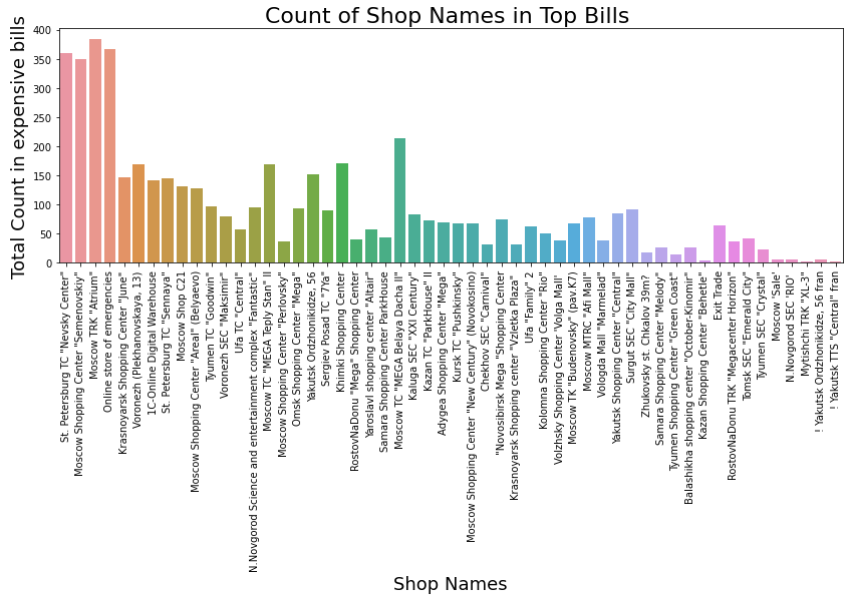
公司算是零售產業，主要的目標客戶跟銷售對象仍是一般的普羅大眾，而非以企業取向，才會有這樣的結果呈現。

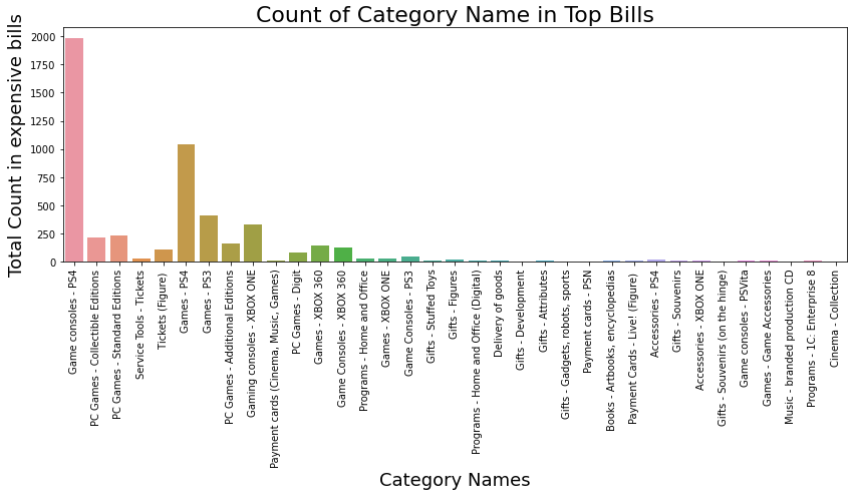
圖十三

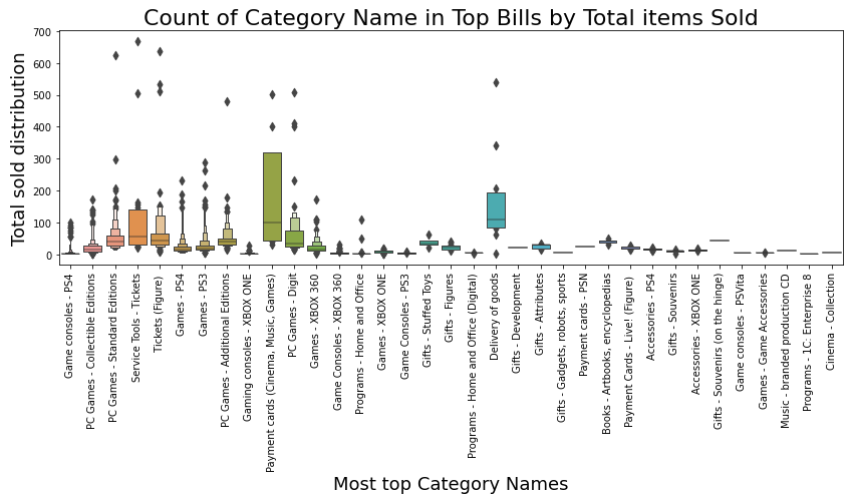


繼續探索資料，如下圖十四，第一張圖為按照商店產值高低所繪製而成的分佈圖：第二章圖為按照產品品項產值高低所繪製的分佈圖，其中以 PS4 的遊戲片為最大宗；第三張圖則是品項對銷售量的盒鬚圖。

圖十四


Count of Shop Names in Top Bills
Total Count in expensive bills
Shop Names


Count of Category Name in Top Bills
Total Count in expensive bills
Category Names


Count of Category Name in Top Bills by Total items Sold
Total sold distribution
Most top Category Names

最後我們將資料的日銷售轉成月銷售，如此一來能更方便我們做後續的迴歸以及其他分析，如圖十五。

圖十五

```
def generate_random_color():
    r = lambda: random.randint(0,255)
    return '#%02X%02X%02X' % (r(),r(),r())

#shared Xaxis parameter can make this graph look even better
fig = pls.make_subplots(rows=2, cols=1)

layout1 = cf.Layout(
    height=500,
    width=200
)
animal_color = generate_random_color()
fig1 = df_train.groupby(['_month'])['item_cnt_day'].count().iplot(kind='bar',barmode='stack',
                                                                  asFigure=True,showlegend=False,
                                                                  title='Total Items Sold By Month',
                                                                  xTitle='Months', yTitle='Total Items Sold',
                                                                  color = 'blue')
fig1['data'][0]['showlegend'] = False
fig.append_trace(fig1['data'][0], 1, 1)


fig2 = df_train.groupby(['_month'])['item_cnt_day'].sum().iplot(kind='bar',barmode='stack',
                                                                title='Total orders by Month',
                                                                xTitle='Months', yTitle='Total Orders',
                                                                asFigure=True, showlegend=False,
                                                                color = 'blue')

#if we do not use the below line there will be two legend
fig2['data'][0]['showlegend'] = False


fig.append_trace(fig2['data'][0], 2, 1)

layout = dict(
    title= "Informations by Date",
    )

fig['layout']['height'] = 800
fig['layout']['width'] = 1000
fig['layout']['title'] = "TOTAL ORDERS AND TOTAL ITEMS BY MONTHS"
fig['layout']['yaxis']['title'] = "Total Items Sold"
fig['layout']['xaxis']['title'] = "Months"
fig['layout']
```

3. 機器學習

3-1 線性迴歸

首先我們利用機器學習基本模型中的線性迴歸進行線性分析，最終得出結果，均方根誤差為 10.455

圖一

```
from sklearn.linear_model import LinearRegression
LN = LinearRegression(n_jobs=-1)
LN.fit(X_train, Y_train)
y_pred_N = LN.predict(X_test)
from sklearn.metrics import mean_squared_error
from numpy import sqrt
rmse_N = sqrt(mean_squared_error(Y_test,y_pred_N))
print('RMSE: %.3f' % rmse_N)
```

```
RMSE: 10.455
```

3-2 決策樹

再來我們利用決策樹方法來分析，看能否得到比前兩者模型更好的結果。我們使用模型中的預設值參數進行套用，並固定隨機性，得出均方根誤差為 6.844，明顯有比前兩者之結果來的更準確

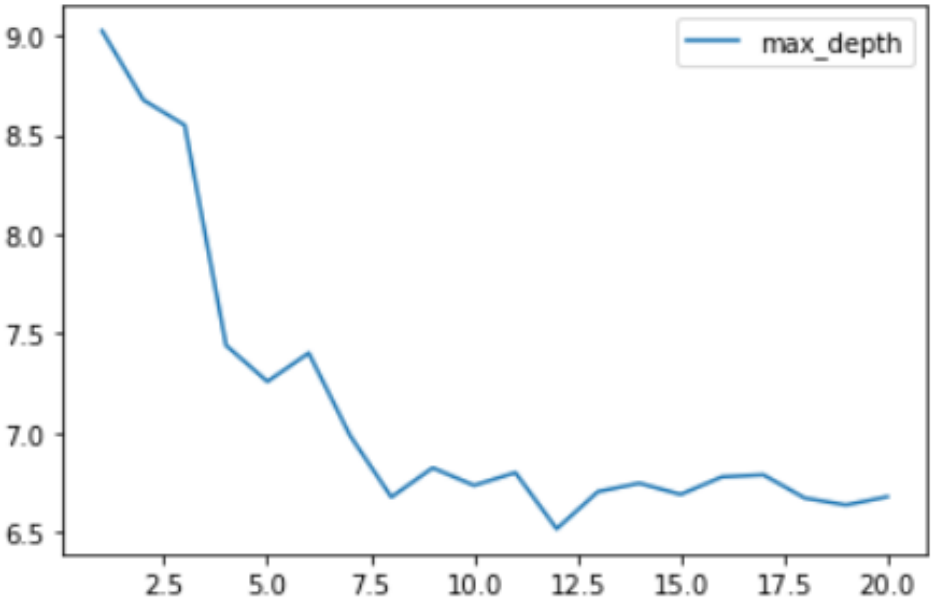
圖三

```
from sklearn.tree import DecisionTreeRegressor
DTR = DecisionTreeRegressor(random_state=1)
DTR.fit(X_train, Y_train)
y_pred_D = DTR.predict(X_test)
rmse_D = sqrt(mean_squared_error(Y_test,y_pred_D))
print('RMSE: %.3f' % rmse_D)
```

```
RMSE: 6.844
```

後續我們嘗試找出能夠最小化誤差之最優剪枝參數，也就是最大樹深，根據下圖顯示，最佳樹深值約莫為 12 層，最小均方根誤差約為 6.5

圖四



3-3 KNN

最後我們使用老師上課所教之 KNN 演算法進行分析，使用預設之參數 K 值套用後，得出均方根誤差為 6.391，相較決策樹又來的更準確

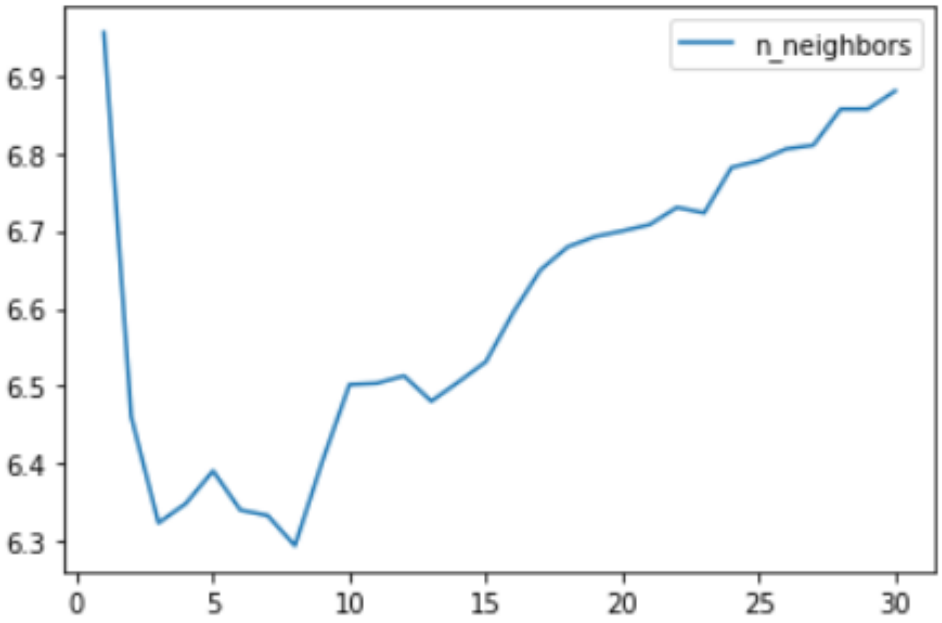
圖五

```
from sklearn.neighbors import KNeighborsRegressor
KNN = KNeighborsRegressor()
KNN.fit(X_train, Y_train)
y_pred_K = KNN.predict(X_test)
rmse_K = sqrt(mean_squared_error(Y_test,y_pred_K))
print('RMSE: %.3f' % rmse_K)
```

```
RMSE: 6.391
```

接著我們一樣嘗試找出最佳參數 K 值，根據下圖顯示，最佳 K 值約莫 7 到 8 之間，最小均方根誤差約為 6.3

圖六



4. 演算法延伸

4-1 時間序列的特徵工程

因為我們的資料具有時序性，而時序性資料一般可分為系統性與非系統性，系統性資料的組成分為趨勢性、周期性、水平(意即:序列的平均值)，而我們的資料為系統性的，因此我們可以採用下列兩種方式，來幫助資料的預測:

- 新增月平均值欄位:系統性時間序列資料的組成其中一個為序列的平均值，因此加入一段時間的平均值可以幫助預估未來趨勢，我們這組採用一個月為區間計算平均值。
- 新增先前時間點的數值欄位:由於時間序列資料的自相關性，先前時間點的數值與當下數值具高相關性，因此我們可以加入先前時間點的值幫助預估未來資料，我們這組這邊透過資料基本探索了解發現，資料週期性為 3 個月一期，因此取 3 個月前的資料做特徵值。

之後，我們分別將未經過處理跟有處理的資料分別丟入 LightGBM 模型可得到結果，如下圖一、圖二，從倆圖可知我們所做的時間序列特徵工程是有用的，他可以有效減少 RMSE 意即為讓我們的資料可以更有效的被模型運算，在之後的模型訓練都會以有經過時間序列特徵工程的資料為主。

圖一

```
[40] other_params_lgb = {'learning_rate': 0.01, 'n_estimators': 500}

     param_grid= {
                                'max_depth': [5, 10, 15, 20, 25],
                                'learning_rate': [0.01, 0.02, 0.05, 0.005, 0.008],
```

```
[38] other_params_lgb = {'learning_rate': 0.01, 'n_estimators': 500}

     param_grid= {
                                'max_depth': [5, 10, 15, 20, 25],
                                'learning_rate': [0.01, 0.02, 0.05, 0.005, 0.008],
                                'n_estimators': [500, 1000, 2000, 3000],
                                'reg_alpha': [0, 0.25, 0.5, 0.75, 1],
                                'reg_lambda': [0.2, 0.4, 0.6, 0.8, 1]
     }
     estimator = LGBMRegressor(**other_params_lgb)
     start = time.time()
     gbm_best, best = get_best_parameter(estimator, param_grid, X_train, y_train)
     end = time.time()
     print("執行時間：%f 分鐘" % ((end - start)/60))

     執行時間：8.080834 分鐘
```

```
preds_g = gbm_best.predict(X_valid)
rmse_g = sqrt(mean_squared_error(y_valid,preds_g))
if best:
        print('RMSE: %.3f for adjusted_light_gbm' % rmse_g)
else:
        print('RMSE: %.3f for light_gbm' % rmse_g)

RMSE: 0.813 for light_gbm
```

圖二

4-2 選取進階的演算法

根據老師建議我們主要考量兩模型，一者為 Light GBM 模型，一者為 XGB 模型，兩者皆有各優缺點，分別跑的結果如下圖三、圖四，雖然與預期不同，LightGBM 運算出的 RMSE 比 XGBoost 還低一點，但我們團隊思考推測後得到的原因是由於分散的層數不夠，在有限的層樹下 LightGBM 可以分割出更有效特徵值，因此計算出較低的 RMSE，也代表 LightGBM 演算法較適合我們的資料，之後模型參數調整與最佳化，我們也會以 LightGBM 演算法建的模型為主。

```
[ ]
preds = xgb_best.predict(X_valid)
rmse = sqrt(mean_squared_error(y_valid,preds))
if best:
        print('RMSE: %.3f for adjusted_xgb' % rmse)
else:
        print('RMSE: %.3f for xgboost' % rmse)
```

```
RMSE: 0.825
```

圖三

圖四

```
[ ] preds_g = gbm_best.predict(X_valid)
rmse_g = sqrt(mean_squared_error(y_valid,preds_g))
if best:
        print('RMSE: %.3f for adjusted_light_gbm' % rmse_g)
else:
        print('RMSE: %.3f for light_gbm' % rmse_g)
```

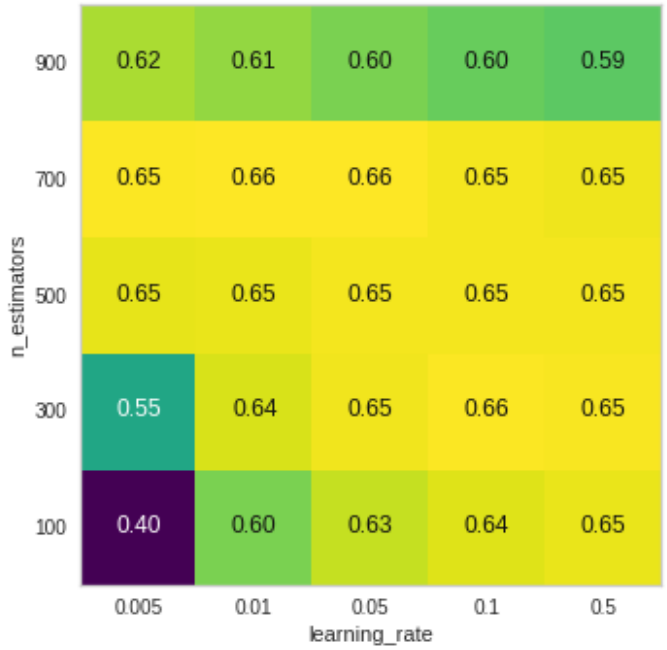
```
RMSE: 0.813 for light_gbm
```

4-3 模型參數調整與最佳化

呈 4-2 所述，我們以 LightGBM 演算法建的模型為主，由於計算效能的緣故，我們組決定選較常見的學習率以及評估器做參數調整，透過均格搜尋，可以得到以下熱區圖與結果，如下圖五與圖六，顯示的最佳的學習率為 0.1，而最佳的評估器參數為 300，比對圖四可以發現模型最佳參數與熱區圖的準確率分佈符合，再得到正確的最佳參數後，我們也得到整個數據科學實驗最後的模型，同時擁有最佳的模型準確率:0.656395561451249。

圖五

```
<matplotlib.collections.PolyCollection at 0x7fd8edaecfd0>
```




```
Fitting 3 folds for each of 25 candidates, totalling 75 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed: 82.1min
[Parallel(n_jobs=-1)]: Done  75 out of  75 | elapsed: 128.6min finished
Best accuracy: 0.6563955614519249
Best parameters: {'learning_rate': 0.1, 'n_estimators': 300}
執行時間: 129.996477 分鐘
```

圖六

5. 深度學習

5-1 模型整理

在做深度學習之前，我們先把資料重新整理成測試的大小，方便在正式模型執行前修正一些可能的錯誤，我們先只取每個商品的月銷售量作為特徵來進行訓練，整理成如圖一所示

圖一

	shop_id	item_id	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	label
0	2	30	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	2	30	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	2	31	0	4	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	2	31	4	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
4	2	32	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	2	2	0	2	0	0	1	0	0	0	0	1	0

5-2 模型建立

我們使用 LSTM 模型來做訓練，其中隱藏層設的節點都不大，因為考慮到後續用的資料量大，怕訓練的時間過長，完整資訊如圖二

圖二

```
lstm_model = Sequential()
lstm_model.add(L.LSTM(10, input_shape=(serie_size, n_features), return_sequences=True))
lstm_model.add(L.LSTM(6, activation='relu', return_sequences=True))
lstm_model.add(L.LSTM(1, activation='relu'))
lstm_model.add(L.Dense(10, kernel_initializer='glorot_normal', activation='relu'))
lstm_model.add(L.Dense(10, kernel_initializer='glorot_normal', activation='relu'))
lstm_model.add(L.Dense(1))
lstm_model.summary()

adam = optimizers.Adam(lr)
lstm_model.compile(loss='mse', optimizer=adam)
```

5-3 模型訓練

訓練的 batch_size 設 256，epochs 設 20，可以看到訓練結果的 RMSE 是比之前機器學習模型來的大的，這並不令人意外，因為這次訓練的資料是簡化過的，後續將會用完整的資料去做訓練，這次訓練的結果如圖三所示

圖三

```
Epoch 10/20
1566/1566 - 58s - loss: 1.3309 - val_loss: 1.2755
Epoch 11/20
1566/1566 - 58s - loss: 1.3232 - val_loss: 1.2692
Epoch 12/20
1566/1566 - 57s - loss: 1.3142 - val_loss: 1.2649
Epoch 13/20
1566/1566 - 58s - loss: 1.3044 - val_loss: 1.2540
Epoch 14/20
1566/1566 - 58s - loss: 1.2919 - val_loss: 1.2472
Epoch 15/20
1566/1566 - 58s - loss: 1.2795 - val_loss: 1.2372
Epoch 16/20
1566/1566 - 58s - loss: 1.2650 - val_loss: 1.2302
Epoch 17/20
1566/1566 - 58s - loss: 1.2535 - val_loss: 1.2347
Epoch 18/20
1566/1566 - 58s - loss: 1.2426 - val_loss: 1.2177
Epoch 19/20
1566/1566 - 57s - loss: 1.2325 - val_loss: 1.2197
Epoch 20/20
1566/1566 - 57s - loss: 1.2237 - val_loss: 1.2052
```

5-3 完整模型訓練

為避免訓練時間過長，再沿用機器學習使用的資料的基礎上，我們使用完整 700 百萬筆的資料量(圖四)，同時我們設定了 keras 內建的兩個函數，幫助我們在長時間的訓練下，避免 colab 突然斷線的狀況

發生，這兩個函數分別是 ModelCheckpoint 和 EarlyStopping，第一個函式幫助我們在每一次 epoch 跑完後儲存其權重，第二個函式則幫我們監測如果 val_loss 的值並未降低則提早結束訓練，幫助我們節省更多時間(圖五)，完整模型的訓練圖大致正常，並沒有過擬合的情況發生(圖六)，而訓練分數來到了 0.7 左右，相比機器學習的 0.8 更為進步(圖七)

圖四

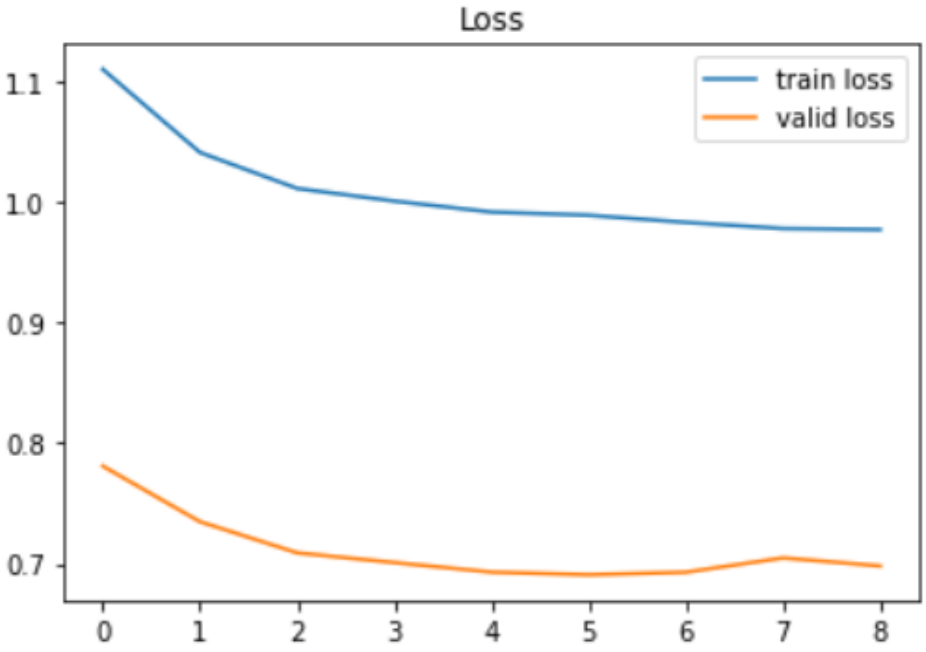
```
X_train_r = X_train.values.reshape((X_train.shape[0], X_train.shape[1], 1))
X_valid_r = X_valid.values.reshape((X_valid.shape[0], X_valid.shape[1], 1))
X_test_r = X_test.values.reshape((X_test.shape[0], X_test.shape[1], 1))
print(X_train_r.shape)
print(X_valid_r.shape)
print(X_test_r.shape)
```

```
(7593847, 30, 1)
(221718, 30, 1)
(214200, 30, 1)
```

圖五

```
model_checkpoint_callback = [tf.keras.callbacks.ModelCheckpoint(
        filepath='model.h5',
        monitor='val_loss',
        save_best_only=True), tf.keras.callbacks.EarlyStopping(patience=3, monitor = 'val_loss')]
```

圖六



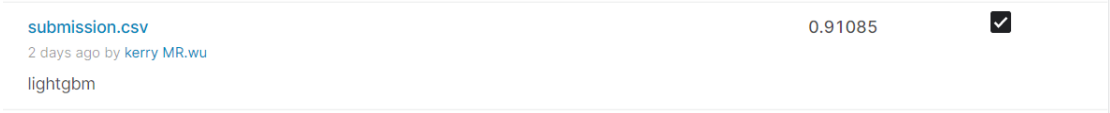
圖七

```
Epoch 1/10
59327/59327 - 425s - loss: 1.1103 - val_loss: 0.7811
Epoch 2/10
59327/59327 - 413s - loss: 1.0413 - val_loss: 0.7349
Epoch 3/10
59327/59327 - 413s - loss: 1.0114 - val_loss: 0.7091
Epoch 4/10
59327/59327 - 413s - loss: 1.0009 - val_loss: 0.7010
Epoch 5/10
59327/59327 - 410s - loss: 0.9918 - val_loss: 0.6931
Epoch 6/10
59327/59327 - 404s - loss: 0.9892 - val_loss: 0.6909
Epoch 7/10
59327/59327 - 404s - loss: 0.9833 - val_loss: 0.6930
Epoch 8/10
59327/59327 - 404s - loss: 0.9782 - val_loss: 0.7049
Epoch 9/10
59327/59327 - 402s - loss: 0.9773 - val_loss: 0.6982
```

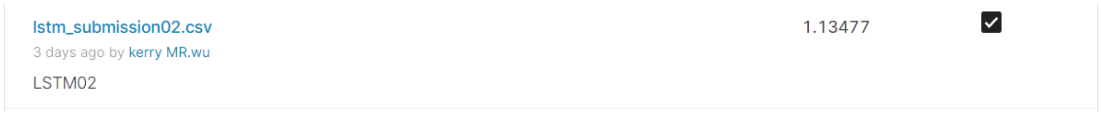
6. 結論

在學習了這麼多模型跟基礎資料處理的方法後，我們將其結果放在 kaggle 上來做最後的評分，分別是 lightgbm(圖八)，LSTM 簡化版(圖九)，LSTM 完整版(圖十)，可以看到 lightgbm 的成績最好，可能的原因有很多，或許是 lstm 初期設定的層數不夠、參數不夠好，還有一個可能的原因是在訓練大量資料時，為了加速訓練時間，在 keras 的官網中，可以使用 cuDNN 加速 LSTM 模型，但模型的啟動函數不行是 relu，只能是 tanh，這也可能影響最終成績。無論如何，差距並不會到非常大，而且這在 kaggle 上是 public 分數而不是 private，最終結果可能還有變動，而排名上的前段分數是在 0.7 左右，我們也還有不小的距離要努力。

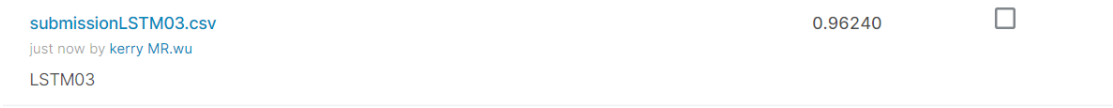
圖八



圖九



圖十



在了解完基礎的原理後，超參數的調整是極為重要的，在電腦硬

體需求上也因此變得更為關鍵，即使在課程的後半段轉為在 colab 上去執行，時間的需求也還是巨大，無論如何，雖然無法在 kaggle 上取得非常好的成績，至少我們已經掌握了機器、深度學習的基礎，在未來需要做相關應用時，我們可以流暢自如地使用它們，為其決策做出良好的價值。