README

目錄

一、 實驗環境安裝

1.1 環境檔位置

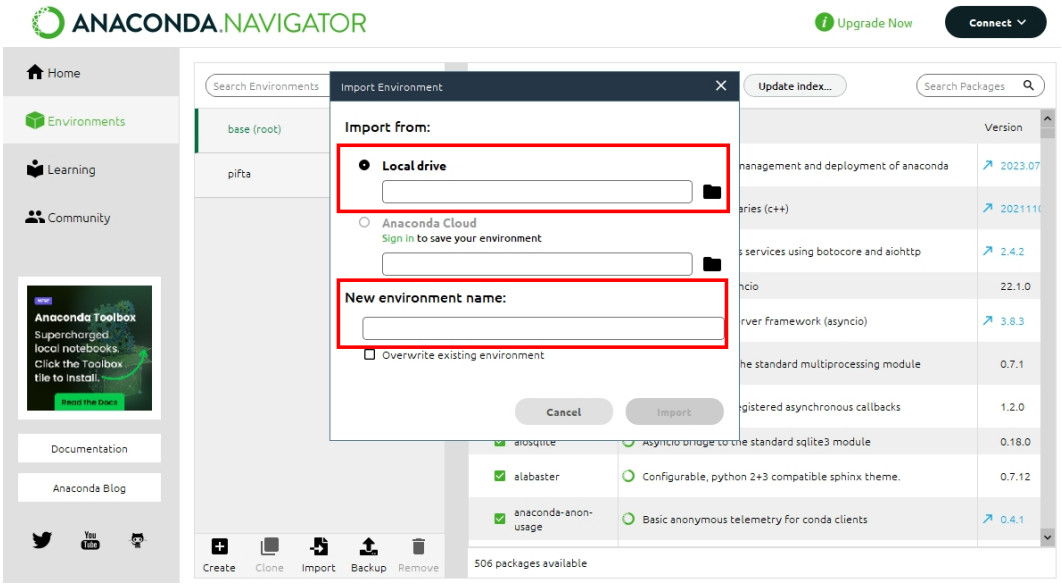
位於「程式碼」資料夾內的「環境」資料夾內，包含以下項目以及說明。

環境檔	使用情境
pifta_env.yml	「PIFTA4Rec _Code」資料夾內的所有檔案。
baselines_env.yml	「Baselines_Code」資料夾內的所有檔案。

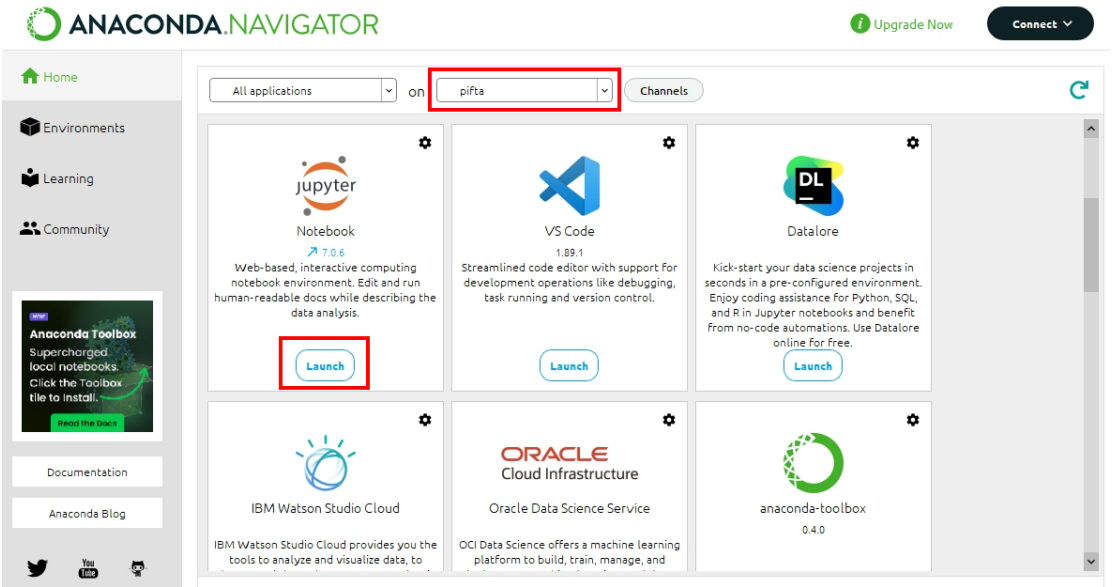
*目前環境檔皆使用 GPU 進行模型訓練和運算，本實驗是在 WIN 10 + RTX 3090 的環境下進行設定，若顯示卡並非 RTX 3090，請依照 GPU 型號下載所需 pytorch 版本及相應程式。

1.2 實驗環境建立說明

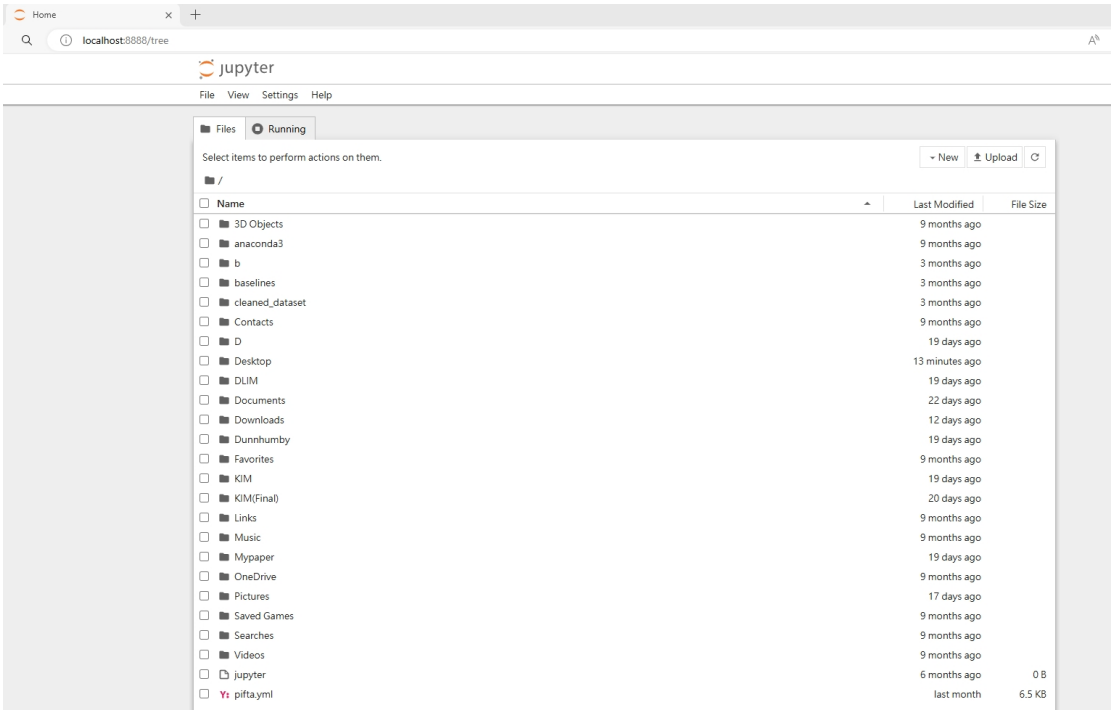
1. 開啟 Anaconda Navigator。
2. 點選左側清單 Environments，點擊下方 Import。從 Local drive「環境」資料夾內選擇環境檔(ex. pifta_env.yml)，在下方欄位可以自行取環境名稱。



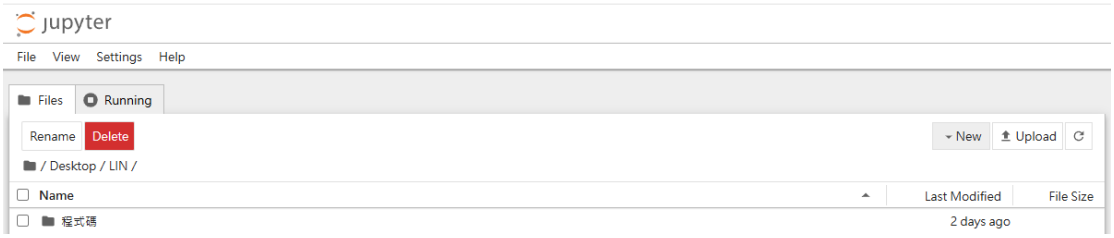
3. 環境建立完成。
4. 點選左側清單 Home，確認上方 Applications on 選擇的是剛剛新增的環境，並點選 Jupyter 下方的 Launch。



5. 開啟以下畫面。



6. 將「程式碼」資料夾複製貼上到自己的電腦中。



7. 直接點選資料夾中的程式檔，即可開始執行程式。

二、 資料夾內容物說明

- 「論文檔案」資料夾
  - 林莉庭_口試簡報.pptx
  - 林莉庭_進度報告.pptx
  - 林莉庭_英文論文.docx
  - 林莉庭_英文論文.pdf
- 「實驗結果」資料夾
  - 林莉庭_實驗結果.xlsx 包含四個分頁： Baselines、Tafeng 敏感度分析、Dunnhumby 敏感度分析、消融實驗。
- 「圖片檔」資料夾
  - 論文使用到的所有照片圖檔
  - 可編輯的圖檔文件「林莉庭_論文圖檔.pptx」
- 「程式碼」資料夾
  - 「PIFTA4Rec_Code」、「Baselines _Code」、「環境」資料夾，會在下一個章節詳細說明。

三、 程式碼說明

3.1 執行環境

「程式碼」資料夾內容架構說明，以下為內部資料夾名稱：

環境檔	使用情境
PIFTA4Rec_Code	包含主模型 PIFTA4Rec 的相關資料集與程式碼。
Baselines _Code	包含所有 Baseline 的程式碼。
環境	包含主模型 PIFTA4Rec 實驗用的環境檔「pifta_env.yml」以及 baseline 使用的環境檔「baselines_env.yml」。

3.2 操作說明

本實驗所使用的資料集為 TaFeng 與 Dunhumby 資料集。程式碼類別分為六類：資料前處理、Item2Vec、KIM、DLIM、PIFTA4Rec 與 Baselines。接著會先針對主程式相關的檔案及資料進行說明，即「資料前處理、Item2Vec、KIM、DLIM、PIFTA4Rec」這五個部分，接著說明 Baselines 相關檔案及資料。

3.2.1 主程式相關檔案

類別	檔名	檔案位置(內)
資料 前處理	dunn_cleaned.ipynb	KIM\KIM 1\Dunnhumby
	tafeng_cleaned.ipynb	KIM\KIM 1\Tafeng
Item2Vec	Item2Vec.ipynb	DLIM
KIM	KIM_Part1.ipynb	KIM\KIM 1\Tafeng
		KIM\KIM 1\ Dunnhumby
	KIM_part2.ipynb	KIM\KIM 2
DLIM	KIM_for_DLIM.ipynb	DLIM
	DLIM 前處理_test.ipynb	DLIM
	DLIM 前處理_training.ipynb	DLIM
	DLIM 前處理_validation.ipynb	DLIM
	Basket_Embedding_Generate.ipynb	DLIM
	DLIM_v1.ipynb	DLIM
PIFTA4Rec	PIFTA4Rec_v1.ipynb	PIFTA4Rec

- **主程式檔案說明**

  - dunn_cleaned.ipynb 與 tafeng_cleaned.ipynb：分別對 Dunnhumby 與 Tafeng 資料集進行資料清理，並進行前處理輸出為 PIFTA4Rec 所需的資料格式。

  - Item2Vec.ipynb：為購物籃中的項目進行編碼，使用 Word2Vec 套件對所有項目進行不同維度的嵌入，並輸出訓練好的模型，其包含項目嵌入資訊。

  - KIM_Part1.ipynb：用於分割訓練集、驗證集與測試集，並輸出各自部分中每位用戶的真實答案（最後一個購物籃）、自己向量與鄰居向量（供 KIM_part2 使用）。**注意：須根據資料集的不同執行不同資料夾中的此檔案。**

  - KIM_part2.ipynb：第一個子模型的主程式，透過一個可學習參數 α 來控制對目標用戶向量與鄰居平均向量之權重，以此產生第一個預測分數。

  - KIM_for_DLIM.ipynb：用於找出訓練集、驗證集與測試集中每位用戶對應的鄰居真實 ID。

  - DLIM 前處理_test.ipynb：根據用戶對應的鄰居真實 ID，取得測試集中每位用戶與其對應的每位鄰居的所有購物籃資訊（購買項目與交易日期）。

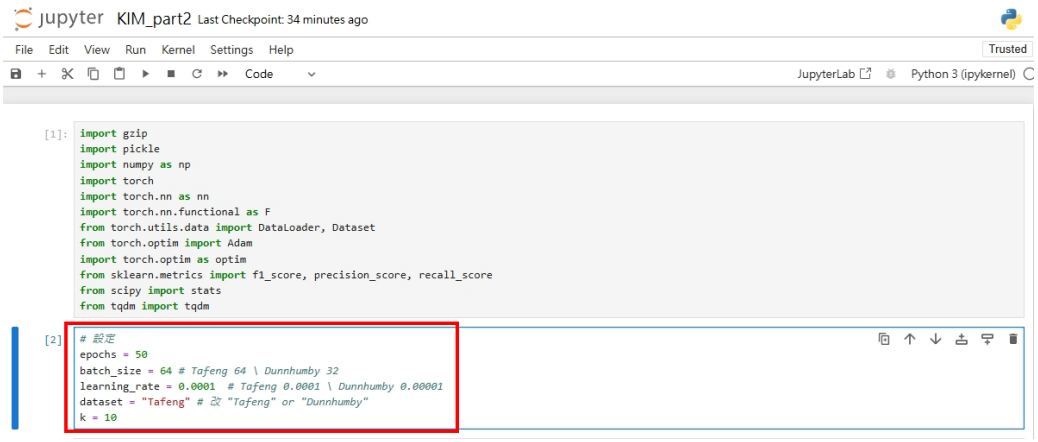
  - DLIM 前處理_training.ipynb：根據用戶對應的鄰居真實 ID，取得訓練集中每位用戶與其對應的每位鄰居的所有購物籃資訊(購買項目與交易日期)。

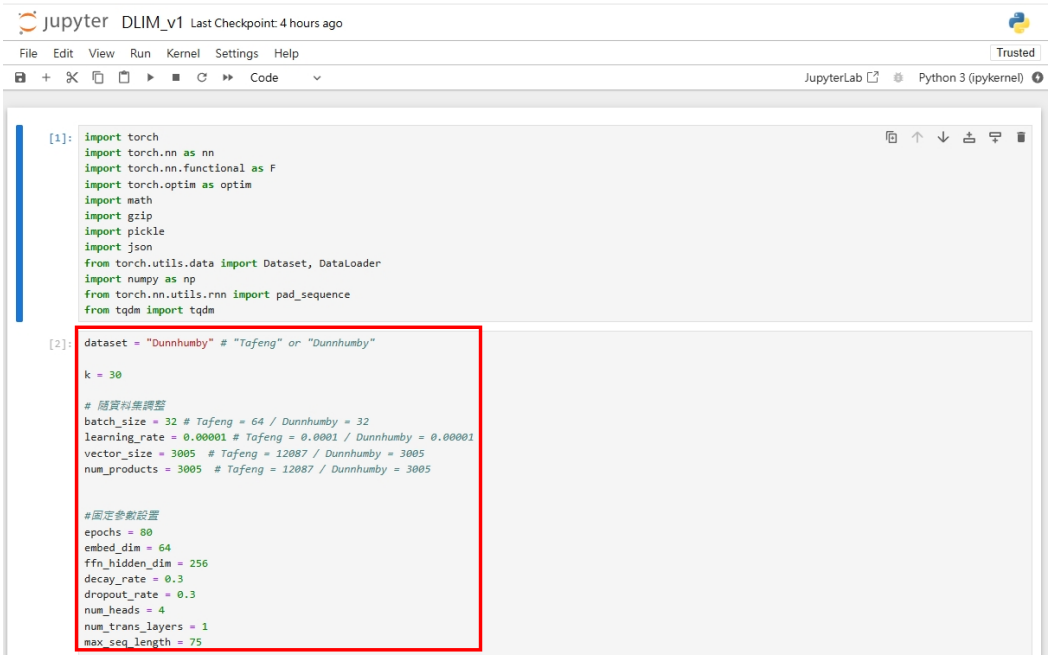
  - DLIM 前處理_validation.ipynb：根據用戶對應的鄰居真實 ID，取得驗證集中每位用戶與其對應的每位鄰居的所有購物籃資訊(購買項目與交

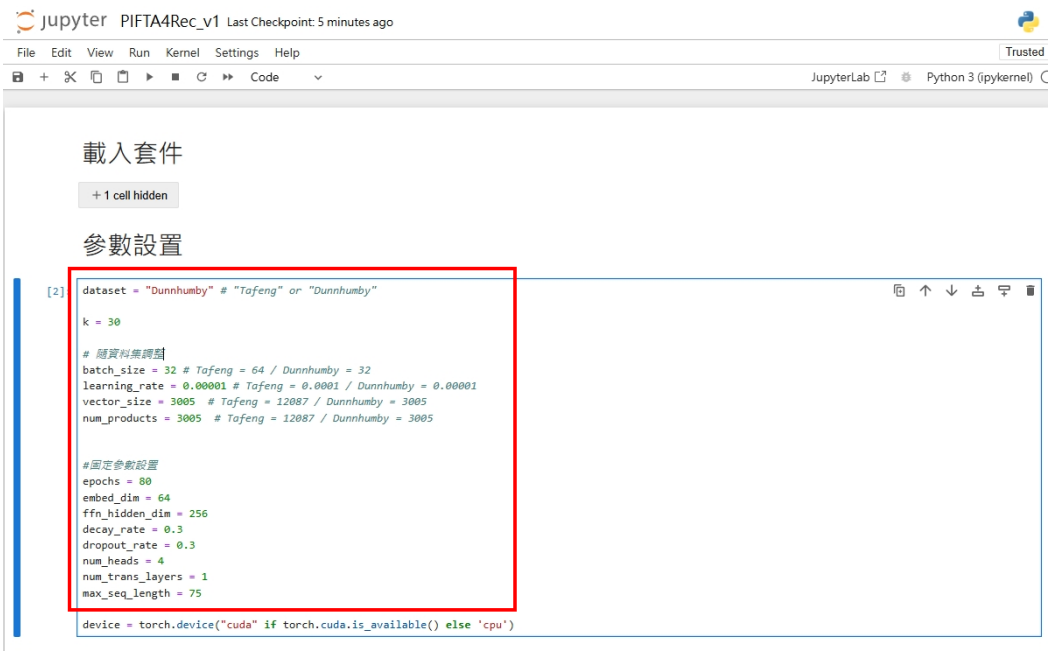
易日期）。

- Basket_Embedding_Generate.ipynb：透過 Item2Vec 的項目嵌入資訊，生成訓練集、驗證集與測試集中每位用戶與其對應的每位鄰居的所有購物籃嵌入向量。
- DLIM_v1.ipynb：第二個子模型的主程式，透過 Temporal Attention 與 Self-Attention 來對用戶的購物籃嵌入向量進行訓練，以此產生第二個預測分數。
- PIFTA4Rec_v1.ipynb：為結合兩個子模型之完整模型 PIFT4Rec 的主程式，產生最終的推薦預測。敏感度分析可透過修改參數使用完整模型進行實驗。

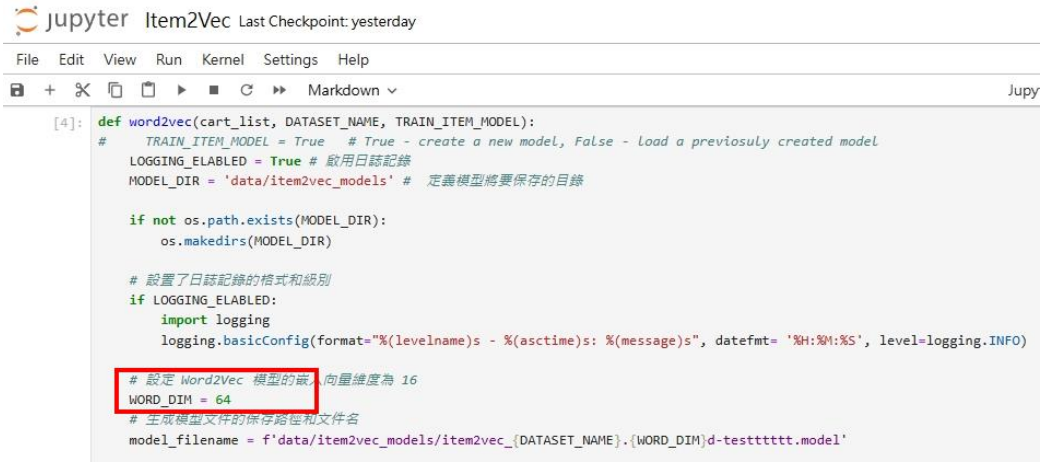
1. 如果需要修改三個模型 KIM & DLIM & PIFTA4Rec 各自的參數，如: Batch Size、Learning Rate、 MLP Hidden Dimension、Decay Rate、Dropout Rate、Multi Head 數量、Transformer Encoder Layer 數量等等，則無需重跑前處理作業，僅需在 KIM_part2.ipynb、DLIM_v1.ipynb 與 PIFTA4Rec_v1.ipynb 內的參數設置中自行設定數值即可。




jupyter DLIM_v1 Last Checkpoint: 4 hours ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
[1]:
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import math
import gzip
import pickle
import json
from torch.utils.data import Dataset, DataLoader
import numpy as np
from torch.nn.utils.rnn import pad_sequence
from tqdm import tqdm
[2]:
dataset = "Dunnhumby" # "Tafeng" or "Dunnhumby"
k = 30
# 隨資料集調整
batch_size = 32 # Tafeng = 64 / Dunnhumby = 32
learning_rate = 0.00001 # Tafeng = 0.0001 / Dunnhumby = 0.00001
vector_size = 3005 # Tafeng = 12087 / Dunnhumby = 3005
num_products = 3005 # Tafeng = 12087 / Dunnhumby = 3005
#固定參數設置
epochs = 80
embed_dim = 64
ffn_hidden_dim = 256
decay_rate = 0.3
dropout_rate = 0.3
num_heads = 4
num_trans_layers = 1
max_seq_length = 75


jupyter PIFTA4Rec_v1 Last Checkpoint: 5 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
載入套件
+ 1 cell hidden
參數設置
[2]
dataset = "Dunnhumby" # "Tafeng" or "Dunnhumby"
k = 30
# 隨資料集調整
batch_size = 32 # Tafeng = 64 / Dunnhumby = 32
learning_rate = 0.00001 # Tafeng = 0.0001 / Dunnhumby = 0.00001
vector_size = 3005 # Tafeng = 12087 / Dunnhumby = 3005
num_products = 3005 # Tafeng = 12087 / Dunnhumby = 3005
#固定參數設置
epochs = 80
embed_dim = 64
ffn_hidden_dim = 256
decay_rate = 0.3
dropout_rate = 0.3
num_heads = 4
num_trans_layers = 1
max_seq_length = 75
device = torch.device("cuda" if torch.cuda.is_available() else 'cpu')

2. 若嵌入維度(Embedding Dimension)需更改為 64 之外的其他數值，除了要更改 DLIM_v1.ipynb 與 PIFTA4Rec_v1.ipynb 中的 embed_dim 外，還須重新跑 Item2Vec.ipynb，並於檔案中自行設定數值。



3.2.2 主程式資料說明

- KIM 子模型

檔案位置	檔名	用途
KIM\KIM 1\Tafeng\raw_data	ta_feng_all_months_merged.csv	從 Kaggle 上下載下來的 TaFeng Dataset 原始資料。
KIM\KIM 1\Dunnhumby\raw_data	transactions_200607.csv transactions_200608.csv transactions_200609.csv transactions_200610.csv transactions_200611.csv transactions_200612.csv transactions_200613.csv transactions_200614.csv transactions_200615.csv transactions_200616.csv transactions_200617.csv transactions_200618.csv	從 Dunnhumby 官方網頁上下載下來的 Dunnhumby Dataset 原始資料。其中每一個檔案皆為一個禮拜的交易資料，共使用三個月的資料(12 個禮拜)。
KIM\KIM 1\{資料集}\cleaned_dataset	{資料集}_clean.csv	分別由「tafeng_cleaned.ipynb」與「dunn_cleaned.ipynb」產生，為清理後並產生所需欄位的資料集。
	{資料集}_history.csv	分別由

		「tafeng_cleaned.ipynb」與「dunn_cleaned.ipynb」產生，為移除掉每位用戶最後一筆交易資料後的剩餘資料，是訓練時使用的歷史資料。
	{資料集}_future.csv	分別由「tafeng_cleaned.ipynb」與「dunn_cleaned.ipynb」產生，為每位用戶最後一筆交易資料，是訓練時使用的真實答案。
KIM\KIM 1\{資料集}\preprocessing-data & KIM\KIM 2\data\preprocessed_data	{資料集}_test_answer.gz {資料集}_training_answer.gz {資料集}_validation_answer.gz	由「KIM_Part1.ipynb」產生，為「KIM_part2.ipynb」中與模型預測結果進行比較的真實購買答案。
	{資料集}_test_user_and_neighbor_set.gz {資料集}_training_user_and_neighbor_set.gz {資料集}_validation_user_and_neighbor_set.gz	由「KIM_Part1.ipynb」產生，為每個用戶向量與對應的鄰居向量，在「KIM_part2.ipynb」使用。
KIM\KIM 2	model_best.pth	由「KIM_part2.ipynb」產生，儲存具有最佳推薦表現的 KIM 模型。

- **DLIM 子模型**

檔案位置	檔名	用途
DLIM\data	TaFeng_clean.csv TaFeng_history.csv TaFeng_future.csv	由 KIM 的「tafeng_cleaned.ipynb」生成，DLIM 延續使用。
	Dunnhumby_clean.csv Dunnhumby_history.csv Dunnhumby_future.csv	由 KIM 的「dunn_cleaned.ipynb」生成，DLIM 延續使用。
DLIM\data\item2vec_models	item2vec_Dunnhumby.64d-testtttttt item2vec_TaFeng.64d-testtttttt	由「Item2Vec.ipynb」產生，為使用 Word2Vec 進行預訓練的模型。
DLIM\data\{資料集}	test_neighbors_for_dlim.json.gz training_neighbors_for_dlim.json.gz validation_neighbors_for_dlim.json.gz	由「KIM_for_DLIM.ipynb」產生，供「DLIM 前處理_test.ipynb」、「DLIM 前處理_training.ipynb」與「DLIM 前處理_validation.ipynb」使用。
	{資料集}_test_users_transactions.json.gz {資料集}_training_users_transactions.json.gz {資料集}_validation_users_transactions.json.gz	由「DLIM 前處理_test.ipynb」、「DLIM 前處理_training.ipynb」與「DLIM 前處理_validation.ipynb」產生，供後續的「Basket_Embedding_Generate.ipynb」使用。

DLIM\data\{資料集}\basketembedding	test_basketembedding_64.pkl.gz training_basketembedding_64.pkl.gz validation_basketembedding_64.pkl.gz	由「Basket_Embedding_Generate.ipynb」產生的購物籃嵌入向量，供後續的「DLIM_v1.ipynb」與「PIFTA4Rec_v1.ipynb」使用。
DLIM\data\preprocessed_data	TaFeng_test_answer.gz TaFeng_training_answer.gz TaFeng_validation_answer.gz	由 TaFeng 資料夾中的「KIM_Part1.ipynb」產生的真實購買答案，直接延續使用進行預測評估。
	Dunnhumby_test_answer.gz Dunnhumby_training_answer.gz Dunnhumby_validation_answer.gz	由 Dunnhumby 資料夾中的「KIM_Part1.ipynb」產生的真實購買答案，直接延續使用進行預測評估。
DLIM	DLIM_Best_model.pth	由「DLIM_v1.ipynb」產生，儲存具有最佳推薦表現的 DLIM 模型。

- **PIFTA4Rec**

檔案位置	檔名	用途
PIFTA4Rec\data\{資料集}	test_neighbors_for_dlim.json.gz training_neighbors_for_dlim.json.gz validation_neighbors_for_dlim.json.gz	由「KIM_for_DLIM.ipynb」產生，直接延續使用。
PIFTA4Rec\data\{資料集}\preprocessed_data	{資料集}_test_answer.gz {資料集}_training_answer.gz {資料集}_validation_answer.gz	由「KIM_Part1.ipynb」產生的真實購買答案，直接延續使用進行預測評估。
	{資料集}_test_user_and_neighbor_set.gz {資料集}_training_user_and_neighbor_set.gz {資料集}_validation_user_and_neighbor_set.gz	由「KIM_Part1.ipynb」產生的每個用戶向量與對應的鄰居向量，直接延續使用。
PIFTA4Rec\data\{資料集}\basketembedding	test_basketembedding_64.pkl.gz training_basketembedding_64.pkl.gz validation_basketembedding_64.pkl.gz	由「Basket_Embedding_Generate.ipynb」產生的購物籃嵌入向量，直接延續使用。
PIFTA4Rec	Best_Model_PIFTA4Rec.pth	由「PIFTA4Rec_v1.ipynb」產生，儲存具有最佳推薦表現的 PIFTA4Rec 模型。

3.2.3 Baseline 相關檔案

類別	檔名	檔案位置(內)
Baseline (Beacon)	Beacon.ipynb Beacon_preprocessing_data.ipynb	baselines\baselines\Beacon
Baseline (CLEA)	config.py CLEA.ipynb CLEA_preprocessing_data.ipynb	baselines\baselines\CLEA
Baseline (DREAM)	DREAM.ipynb	baselines\baselines\DREAM
Baseline (PersonTopFreq)	Baseline_p_top_freq.ipynb dunn_json.ipynb ta_feng_json.ipynb	baselines\baselines\p_top_freq
Baseline (Sets2Sets)	Sets2Sets.ipynb Sets2Sets_preprocessing_data.ipynb	baselines\baselines\Sets2Sets
Baseline (SHAN)	SHAN.ipynb SHAN_preprocessing_data.ipynb	baselines\baselines\SHAN
Baseline (TIFUKNN)	Baseline_TIFUKNN_Final.ipynb	baselines\baselines\TIFUKNN

- **Baselines 檔案說明**
  - Beacon ：

    使用「Beacon_preprocessing_data.ipynb」產生 Beacon 所需格式。

    使用「Beacon.ipynb」進行訓練、驗證和測試模型。

  - CLEA：

    使用「CLEA_preprocessing_data.ipynb」產生 CLEA 所需格式。

    使用「config.py」來調整資料集與相關設定。

    執行「CLEA.ipynb」以訓練、測試模型。

  - DREAM：

    直接執行「DREAM.ipynb」以訓練、測試模型，不需要額外處理資料的輸入格式。

  - PersonTopFreq：

    資料前處理部分直接延續使用由 KIM 中的「dunn_cleaned.ipynb」與「tafeng_cleaned.ipynb」所生成的{資料集}_history.csv 與{資料集}_future.csv。

    再使用「dunn_json.ipynb」與「ta_feng_json.ipynb」分別對兩個資料集產生 PersonTopFreq 所需格式。

    執行「Baseline_p_top_freq.ipynb」以測試模型。

  - Sets2Sets：

    於「Sets2Sets_preprocessing_data.ipynb」產生 Sets2Sets 所需格式。

    執行「Sets2Sets.ipynb」以訓練、測試模型。

- SHAN：

  於「SHAN_preprocessing_data.ipynb」產生 SHAN 所需格式。

  執行「SHAN.ipynb」以訓練、測試模型。

- TIFUKNN：

  資料前處理部分直接延續使用由 KIM 中的「dunn_cleaned.ipynb」與「tafeng_cleaned.ipynb」所生成的{資料集}_history.csv 與{資料集}_future.csv 作為 TIFUKNN 的輸入。

  執行「Baseline_TIFUKNN_Final.ipynb」以測試模型。

3.2.4 Baseline 資料說明

類別	檔案位置	檔名	用途與說明
Dataset	cleaned_dataset	Dunnhumby_clean.csv	分別由「tafeng_cleaned.ipynb」與「dunn_cleaned.ipynb」產生，為清理後並產生所需欄位的資料集，延續使用。
		TaFeng_clean.csv	
Beacon	baselines\Beacon\data_dir\adj_matrix\{資料集}	r_matrix_1w.npz	產生的相關矩陣，訓練時會使用到。
	baselines\Beacon\data_dir	{資料集}_train.txt {資料集}_validate.txt {資料集}_test.txt	於「Beacon_preprocessing_data.ipynb」產生的訓練、驗證與測試集。
	baselines\Beacon\output_dir\{資料集}\topN	output_0.001_1.csv	於「Beacon.ipynb」產生的實驗結果。
		prediction_TopK_1.txt	於「Beacon.ipynb」產生的實驗結果。
CLEA	baselines\CLEA\dataset	{資料集}.txt	於「CLEA_preprocessing_data.ipynb」之後產生的訓練資料。
	baselines\CLEA\result	NB_32.0_{資料集}_test_1.csv	於「CLEA.ipynb」產生的實驗結果。

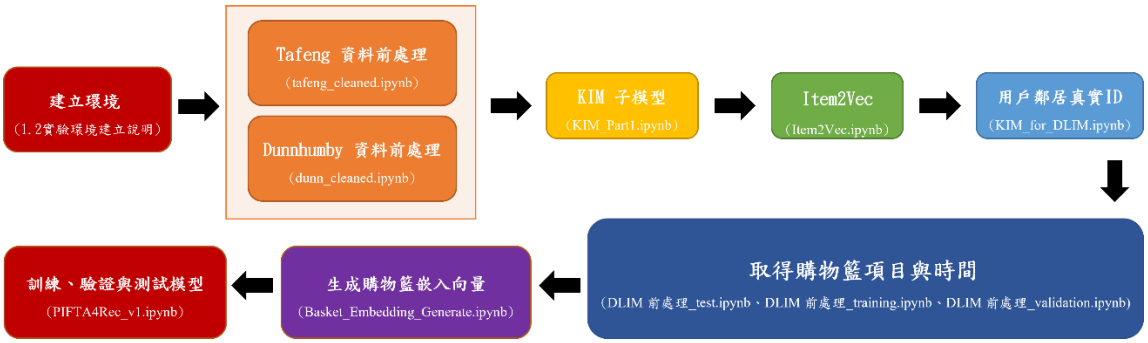
		NB_32.0_{資料集}_test_1.txt	於「CLEA.ipynb」產生的實驗結果。
DREAM	baselines\DREAM\tmp\feat\{資料集}	basket_{資料集}.pkl	執行「DREAM.ipynb」過程產生的購物籃資料。
		test_ub.pkl	執行「DREAM.ipynb」過程產生的測試集資料。
		test_ub_label.pkl	執行「DREAM.ipynb」過程產生的測試集標籤。
		train_ub.pkl	執行「DREAM.ipynb」過程產生的訓練集資料。
	baselines\DREAM\tmp\dream\{資料集}	output_1.csv	於「DREAM.ipynb」產生的實驗結果。
PersonTopFreq	baselines\baselines\p_top_freq\dataset	{資料集}_future.csv {資料集}_history.csv	由 KIM 的「tafeng_cleaned.ipynb」與「dunn_cleaned.ipynb」生成，可直接複製使用。
	baselines\baselines\p_top_freq\jsondata	{資料集}_future.json	分別由「ta_feng_json.ipynb」與「dunn_json.ipynb」所產生，將資料轉為 PersonTopFreq 所需格式。
	baselines\baselines\p_top_freq\keyset	{資料集}_keyset_0.json	為執行「Baseline_p_top_freq.ipynb」過程中所產生的資料。

	baselines\baselines\p_top_freq\p_top_pred	{資料集}_pred0.json	為執行「Baseline_p_top_freq.ipynb」過程中所產生的資料。
Sets2Sets	baselines\Sets2Sets\data\raw_data	{資料集}_future.csv {資料集}_history.csv	由 KIM 的「tafeng_cleaned.ipynb」與「dunn_cleaned.ipynb」生成，可直接複製使用。
	baselines\Sets2Sets\data	{資料集}_future.csv {資料集}_history.csv	為 raw_data 經過「Sets2Sets_preprocessing_data.ipynb」處理後的訓練資料。
	baselines\Sets2Sets\models	decoder{資料集}_model_epoch encoder{資料集}_model_epoch	於「Sets2Set.ipynb」在訓練時，每個 epoch 所儲存的模型。
	baselines\Sets2Sets\result	{資料集}.csv	於「Sets2Sets.ipynb」產生的實驗結果。
SHAN	baselines\SHAN\data	{資料集}.csv	於「SHAN_preprocessing_data.i

			pynb」產生的資料。
	baselines\SHAN\output	1_Shan_{資料集}_{超參數設置}.csv	於「SHAN.ipynb」產生的實驗結果
TIFUKNN	baselines\TIFUKNN\data	{資料集}_future.csv {資料集}_history.csv	由 KIM 的「tafeng_cleaned.ipynb」與「dunn_cleaned.ipynb」生成，可直接複製使用。

四、 操作流程

4.1 主程式操作流程

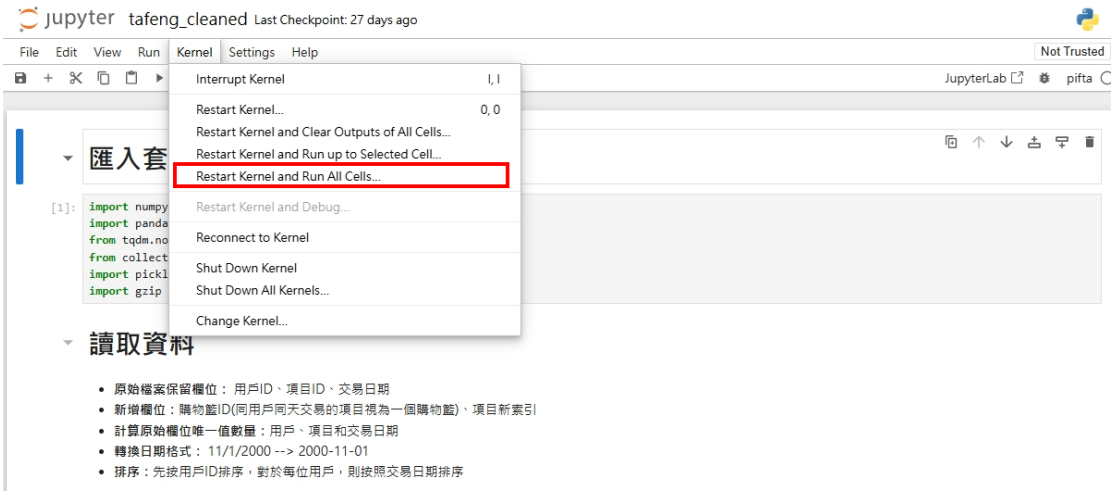


4.1.1 建立環境

請參考本文件 **1.2 實驗環境建立說明**。

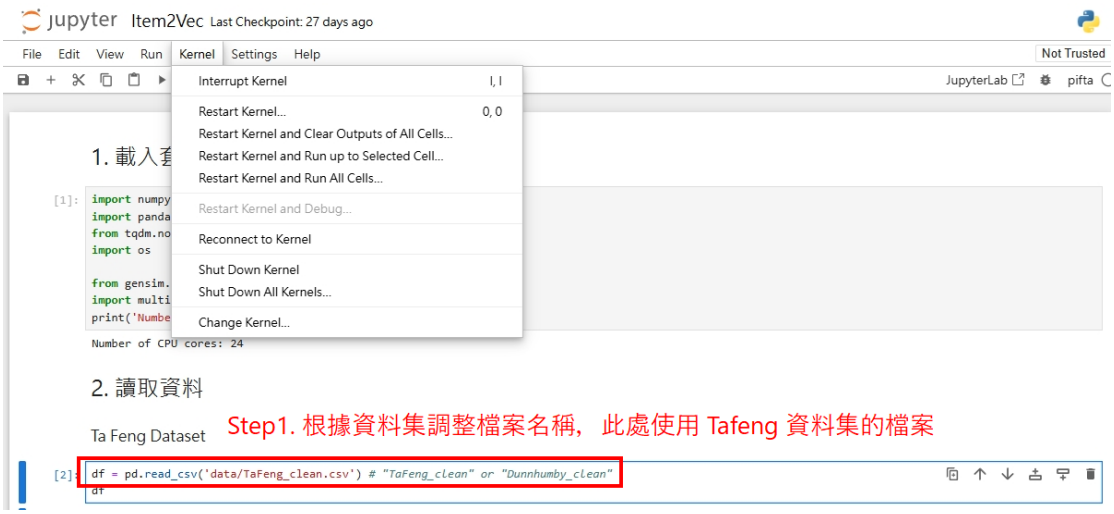
4.1.2 資料前處理

此部分將對資料集進行資料前處理，在過程中我們會檢查空值或重複值與刪除不符合條件的資料(項目/購物籃/用戶)等處理，並且將資料集處理成統一格式。



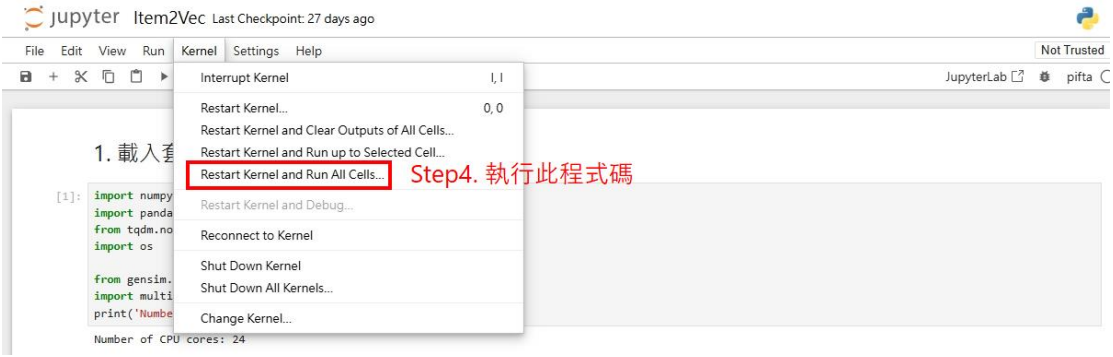
4.1.3 Item2Vec

使用 Item2Vec.ipynb 將資料集中的項目進行嵌入，程式執行結果則會儲存至 DLIM\data\item2vec_models/item2vec_{資料集}.{維度}d-testttttt.model。



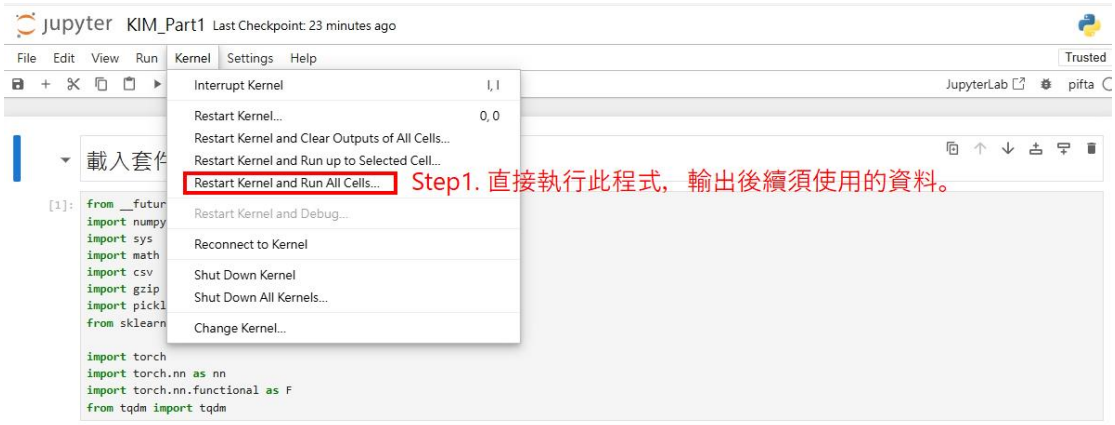

jupyter Item2Vec Last Checkpoint: 27 days ago
File Edit View Run Kernel Settings Help
Not Trusted
Code
JupyterLab pifta
[5]:
# 購物車,新項目id 串列
df_list = df[['CART_ID', 'new_item_id']].values.tolist() # Tafeng: new_item_id / Dunnhumby: NEW_ITEM_ID
df_list[:10]
[5]:
[[0, 0],
[0, 1],
[1, 2],
[1, 3],
[1, 4],
[2, 5],
[2, 1],
[2, 6],
[2, 7],
[2, 8]]
Step2. 根據使用的資料集調整此兩處的內容
[6]:
cart_list = make_item_corpus(df_list)
model = word2vec(cart_list, "TaFeng", True) # "TaFeng" or "Dunnhumby"


jupyter Item2Vec Last Checkpoint: 27 days ago
File Edit View Run Kernel Settings Help
Not Trusted
Code
JupyterLab pifta
[4]:
def word2vec(cart_list, DATASET_NAME, TRAIN_ITEM_MODEL):
# TRAIN_ITEM_MODEL = True # True - create a new model, False - load a previosuly created model
LOGGING_ELABLED = True # 啟用日誌記錄
MODEL_DIR = 'data/item2vec_models' # 定義模型將要保存的目錄
if not os.path.exists(MODEL_DIR):
os.makedirs(MODEL_DIR)
# 設置了日誌記錄的格式和級別
if LOGGING_ELABLED:
import logging
logging.basicConfig(format="%(levelname)s - %(asctime)s: %(message)s", datefmt= '%H:%M:%S', level=logging.INFO)
# 設定 Word2Vec 模型的嵌入向量維度為 16
WORD_DIM = 64
# 生成模型文件的保存路徑和文件名
model_filename = f'data/item2vec_models/item2vec_{DATASET_NAME}.{WORD_DIM}d-testttttt.model'
Step3. 根據想要的嵌入維度大小調整此設定

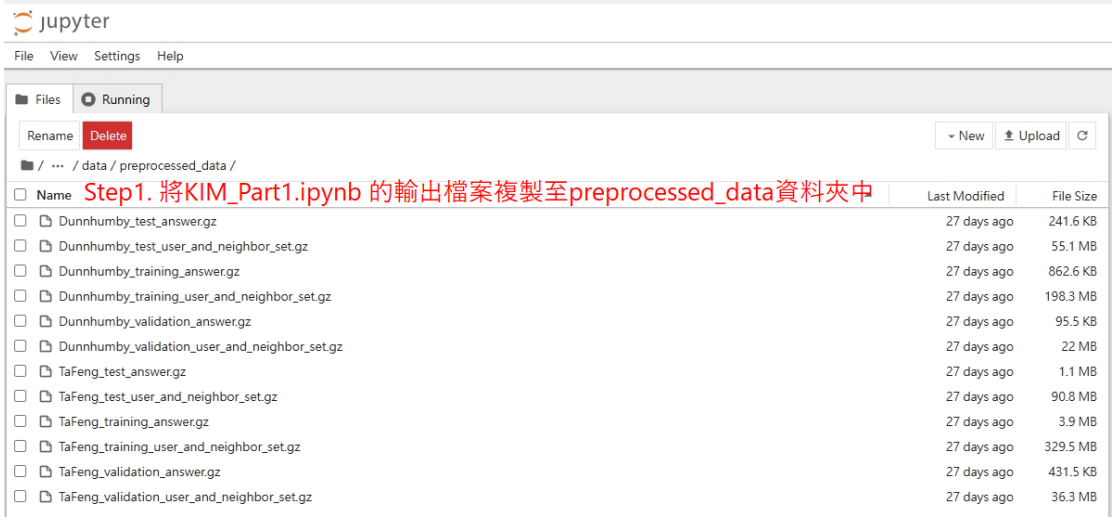

jupyter Item2Vec Last Checkpoint: 27 days ago
File Edit View Run Kernel Settings Help
Not Trusted
JupyterLab pifta
Interrupt Kernel I, I
Restart Kernel... 0, 0
Restart Kernel and Clear Outputs of All Cells...
Restart Kernel and Run up to Selected Cell...
Restart Kernel and Run All Cells...
Step4. 執行此程式碼
Restart Kernel and Debug...
Reconnect to Kernel
Shut Down Kernel
Shut Down All Kernels...
Change Kernel...
1. 載入
[1]:
import numpy
import panda
from tqdm.no
import os
from gensim.
import multi
print('Numbe
Number of CPU cores: 24

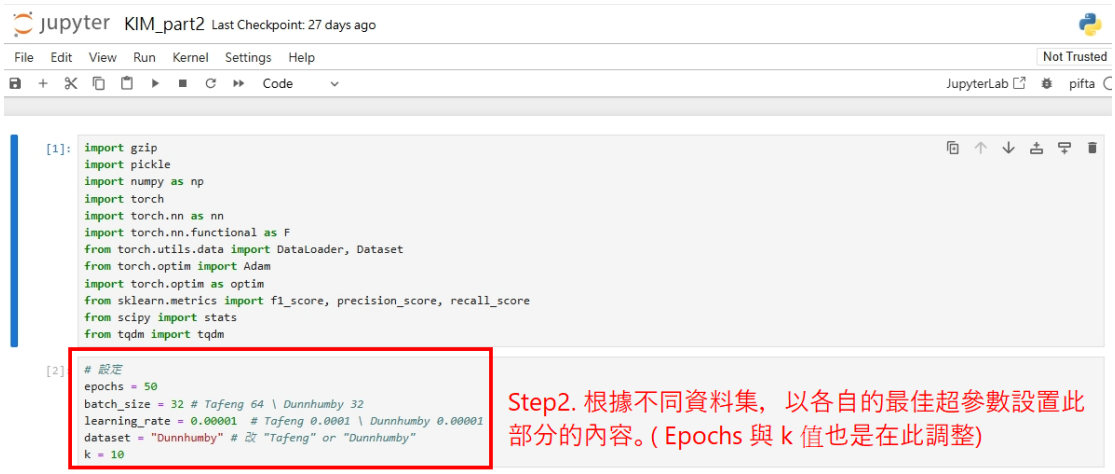
4.1.4 KIM

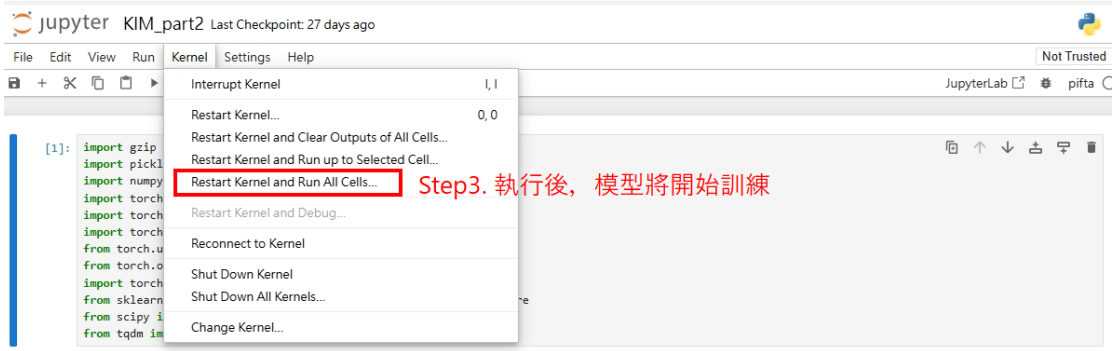
首先，我們將先執行子模型 KIM 的第一支程式碼 KIM_Part1.ipynb。這裡，我們會從 KIM 1 資料夾中，根據不同的資料集挑選特定資料夾中的該支程式碼並直接執行，輸出後續所需使用到的資料。



接著切換至 KIM 2 資料夾，我們需要先將 KIM_Part1.ipynb 輸出的資料放入 data/preprocessed_data 處，再根據不同的資料集調整程式碼中的超參數設置，以及根據評估指標中不同的 k 值來調整 k 值，最後即可執行 KIM_part2.ipynb。



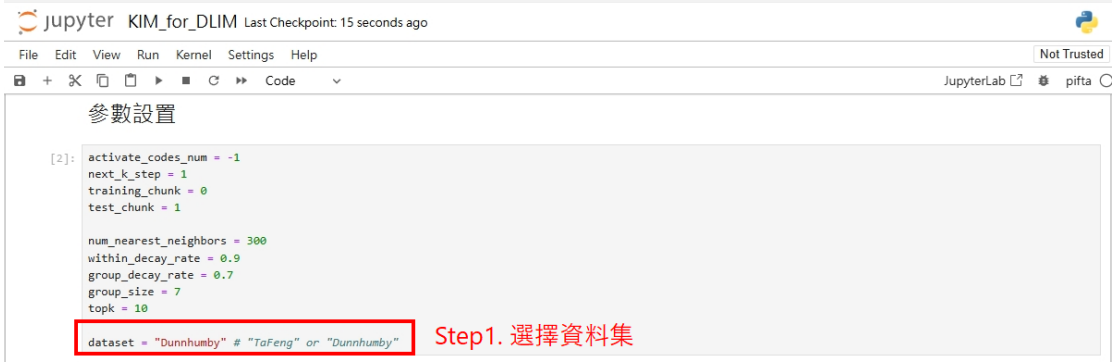


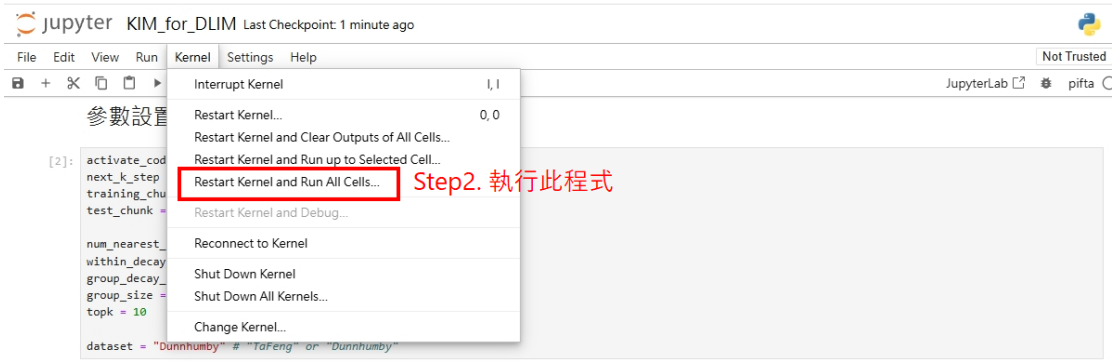


※注意：若想直接執行 PIFTA4Rec 則仍需運行 KIM_Part1.ipynb，而 KIM_part2.ipynb 為單獨評估子模型 KIM 表現之消融實驗，可跳過。

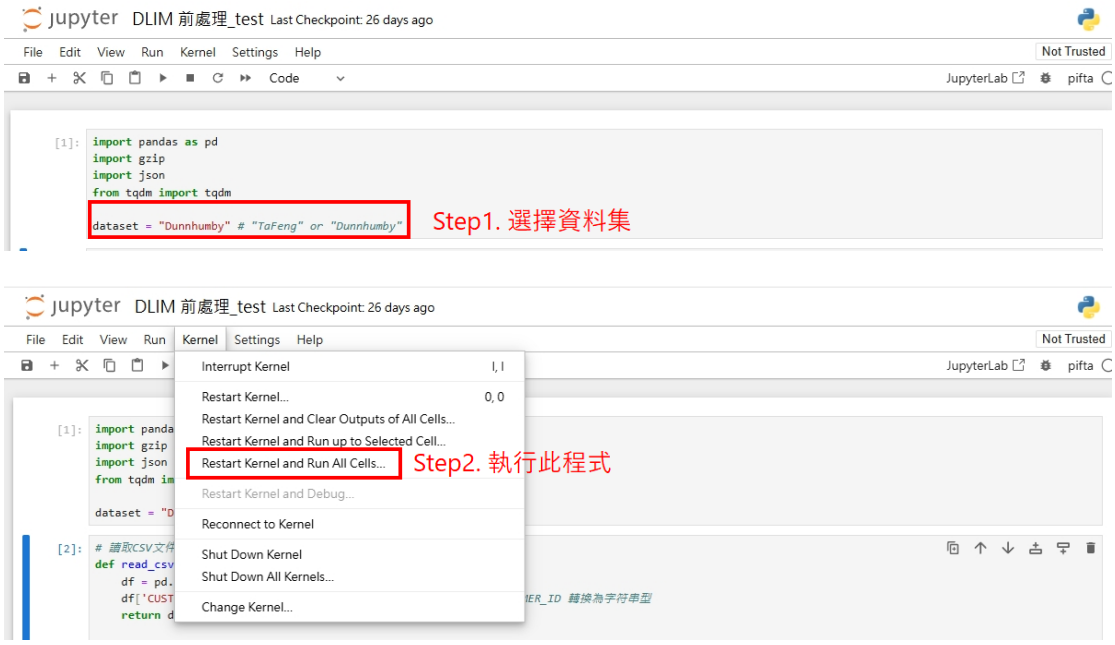
4.1.5 DLIM

此階段共有六個步驟，首先我們會先執行 KIM_for_DLIM.ipynb，透過 KIM 的部分程式碼來為 DLIM 找出每個用戶對應的鄰居真實 ID。

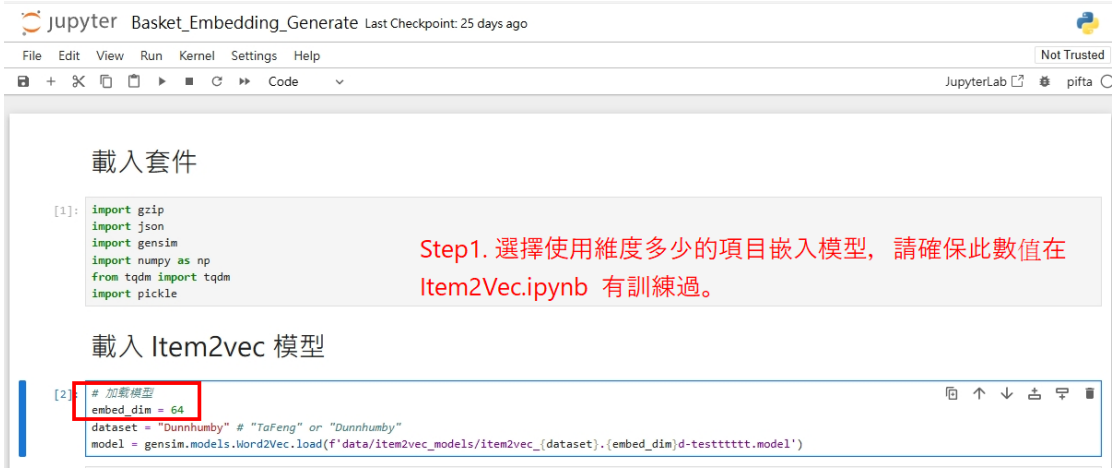


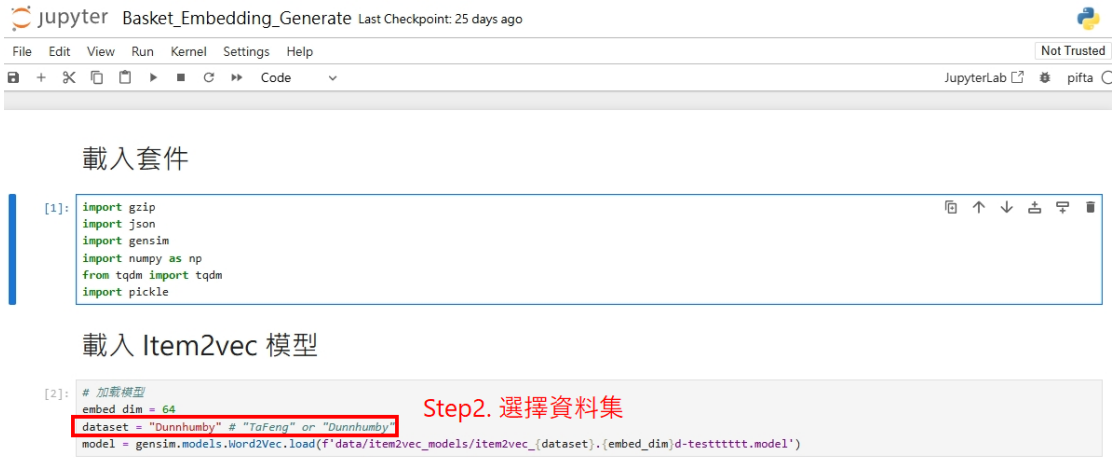


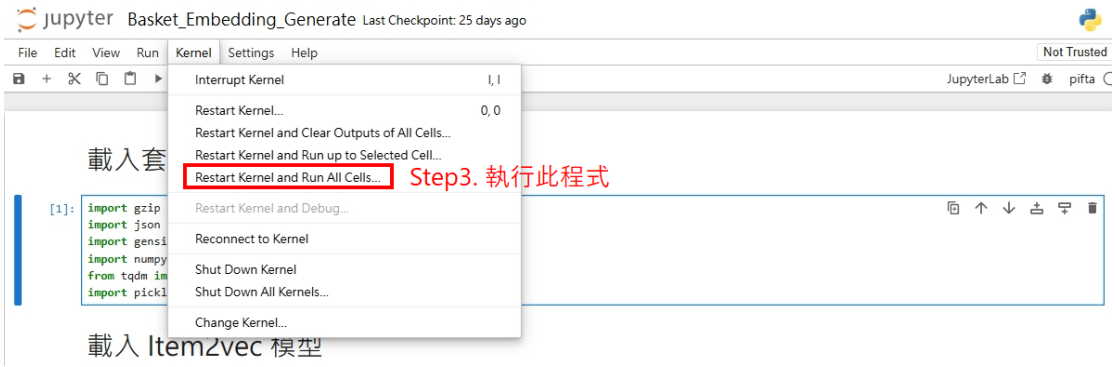
接著步驟二～四是執行 DLIM 前處理_test.ipynb、DLIM 前處理_training.ipynb 與 DLIM 前處理_validation.ipynb 這三支程式碼，目的為獲取特定格式的用戶購物籃交易項目與時間。由於它們內容大致相同且處理方式一致，故下列以運行測試集為例，訓練集與驗證集部分則執行相同步驟。注意：它們都需要較長的執行時間，故可以同時運行以加速處理。



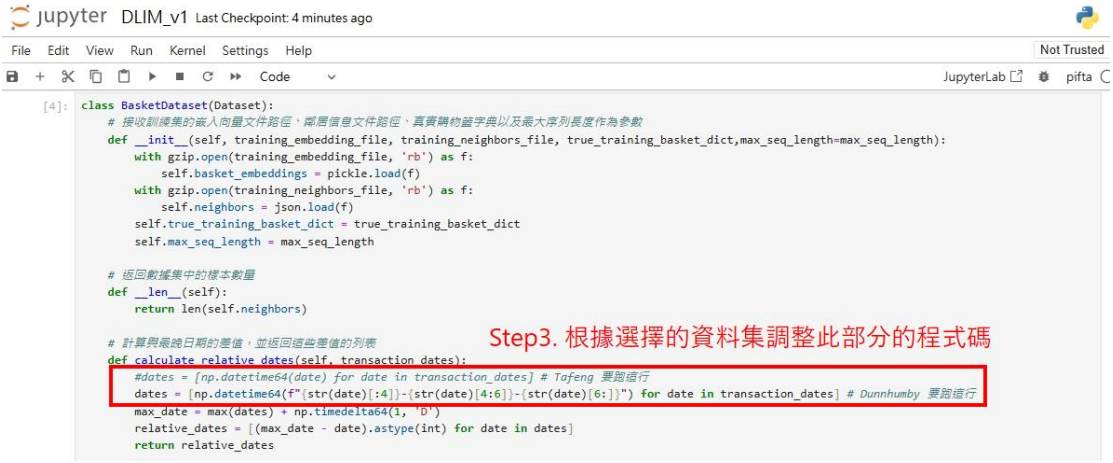
步驟五是執行 Basket_Embedding_Generate.ipynb，其用於為訓練集、測試集與驗證集之用戶生成購物籃嵌入向量。生成的檔案將存放於 DLIM\data\{資料集}\basketembedding 處。

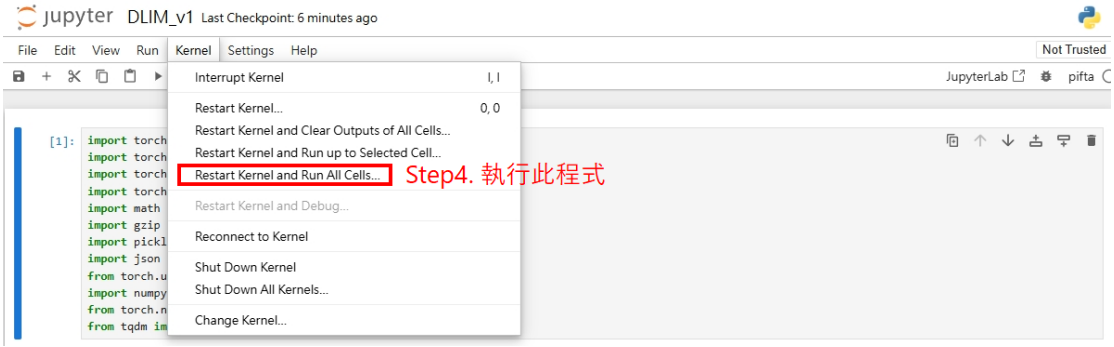






最後，我們需要先將 KIM_Part1.ipynb 輸出的{資料集}_test_answer.gz、{資料集}_training_answer.gz 與{資料集}_validation_answer.gz 檔案放入 DLIM/data/preprocessed_data 處，接著根據不同的資料集調整 DLIM_v1.ipynb 中的超參數設置，以及根據評估指標中不同的 k 值來調整 k 值，調整後即可執行。

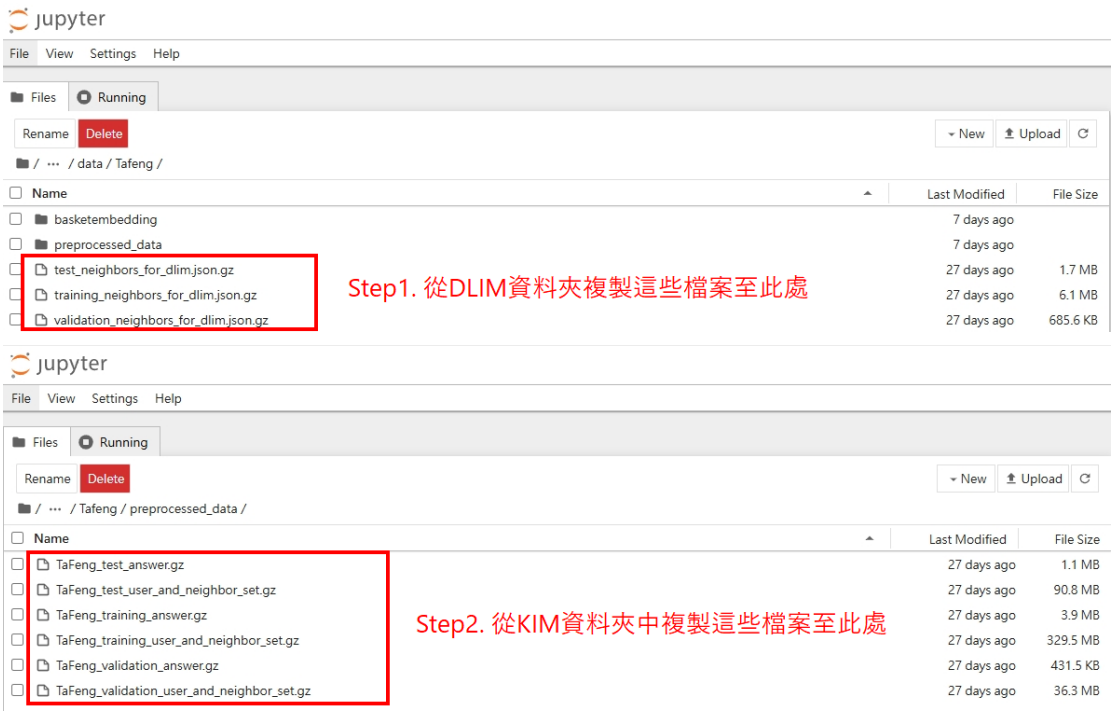




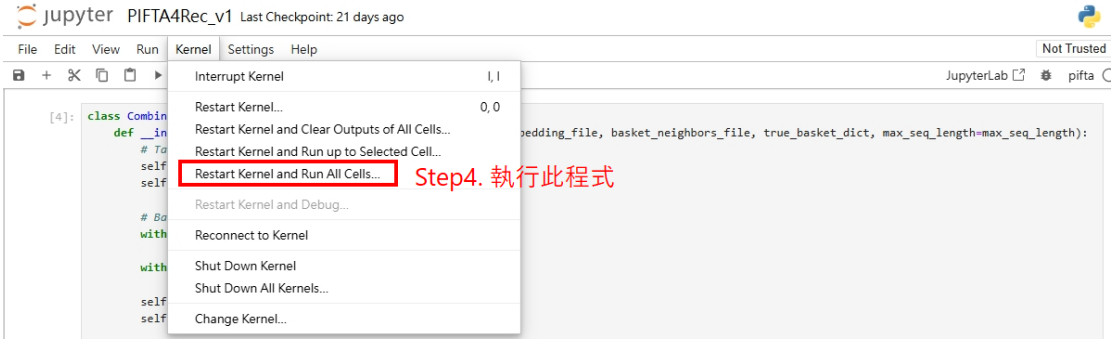
※注意：若想直接執行 PIFTA4Rec 則無須執行 DLIM_v1.ipynb，此程式碼為單獨評估子模型 DLIM 表現之消融實驗，但其他程式碼都需執行。

4.1.6 PIFTA4Rec

首先，我們需要先將在 KIM 與 DLIM 輸出的某些特定檔案複製到 PIFTA4Rec/data 中。以「Tafeng」為例，我們將 test_neighbors_for_dlim.json.gz、training_neighbors_for_dlim.json.gz 與 validation_neighbors_for_dlim.json.gz 複製到 PIFTA4Rec/data/Tafeng 資料夾中。test_basketembedding_64.pkl.gz、training_basketembedding_64.pkl.gz 與 validation_basketembedding_64.pkl.gz 複製到 PIFTA4Rec/data/Tafeng/basketembedding 中。TaFeng_test_answer.gz、TaFeng_training_answer.gz、TaFeng_validation_answer.gz、TaFeng_test_user_and_neighbor_set.gz、TaFeng_training_user_and_neighbor_set.gz 與 TaFeng_validation_user_and_neighbor_set.gz 則複製到 PIFTA4Rec/data/Tafeng/preprocessed_data 中。其他資料集也是相同操作。



接著則是根據不同的資料集調整 PIFTA4Rec_v1.ipynb 中的超參數設置，以及根據評估指標中不同的 k 值來調整 k 值，調整後即可執行。



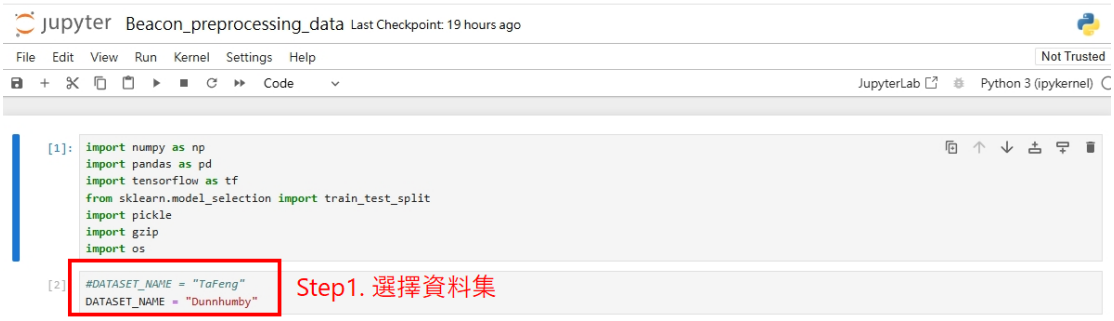
4.1.7 產生結果

不論子模型 KIM、DLIM 或是完整模型 PIFTA4Rec，其訓練結果都會直接呈現在 KIM_part2.ipynb、DLIM_v1.ipynb 與 PIFTA4Rec_v1.ipynb 的輸出中。下圖以 PIFTA4Rec 為例。

4.2 Baselines 操作流程

4.2.1 Beacon

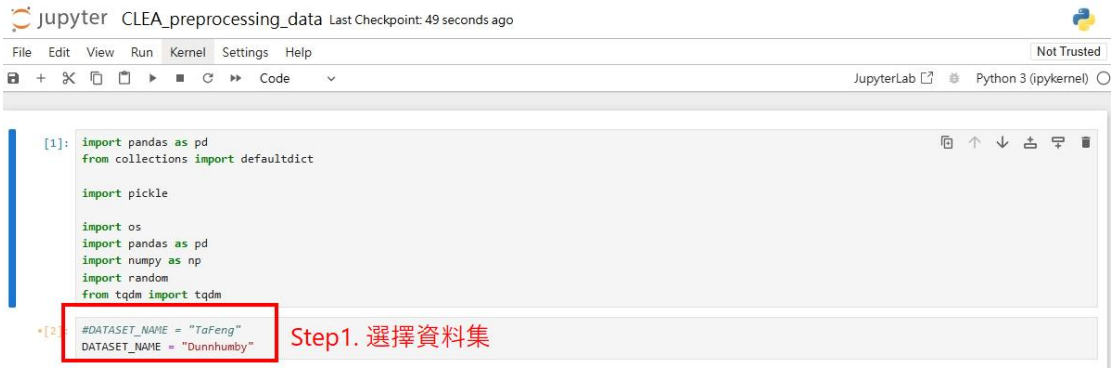
1. 直接執行 Beacon_preprocessing_data.ipynb



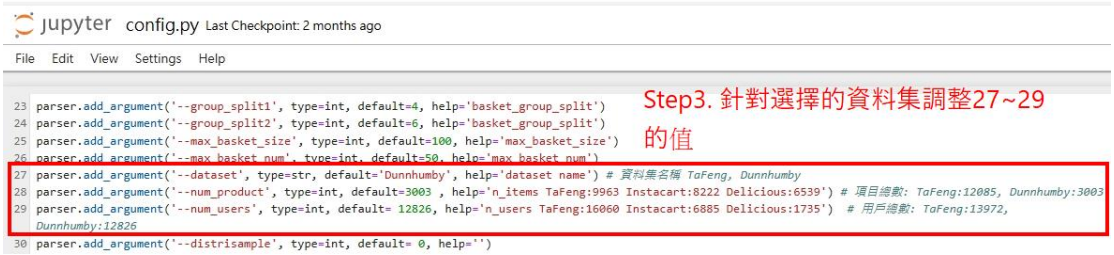
2. 直接執行 Beacon.ipynb

4.2.2 CLEA

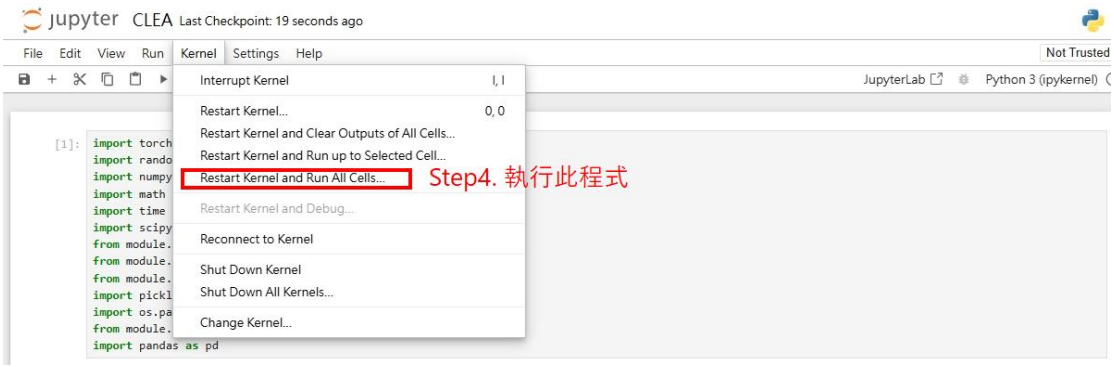
1. 直接執行 CLEA_preprocessing_data.ipynb



2. 調整 baselines\baselines\CLEA\module 的 config.py 檔案設定

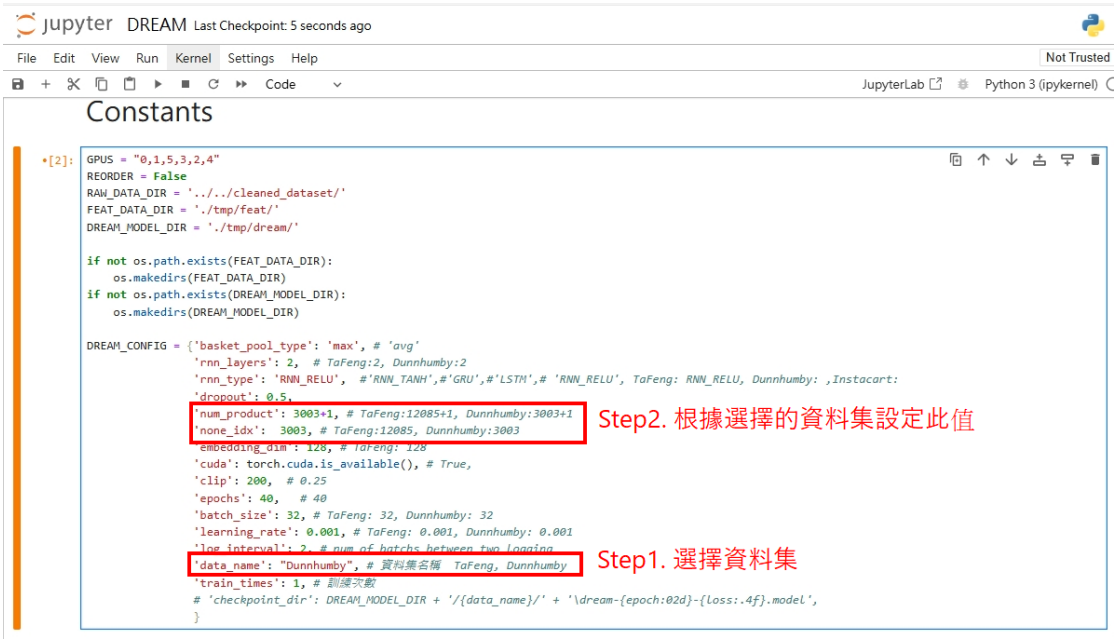


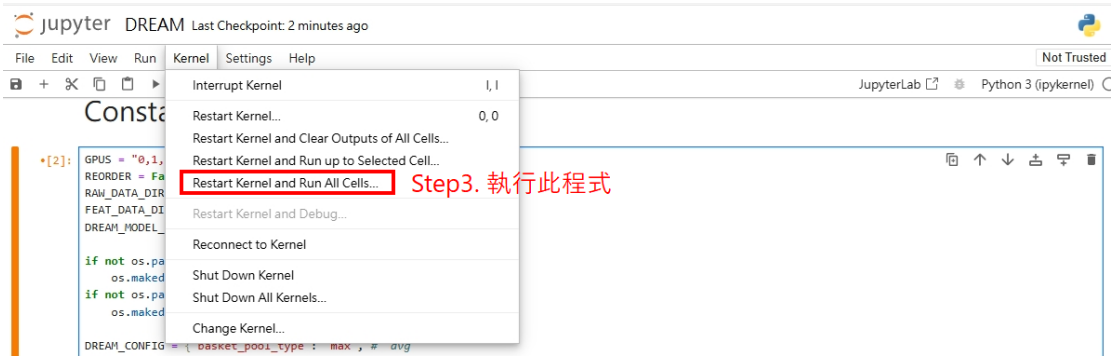
3. 直接執行 CLEA.ipynb



4.2.3 DERAM

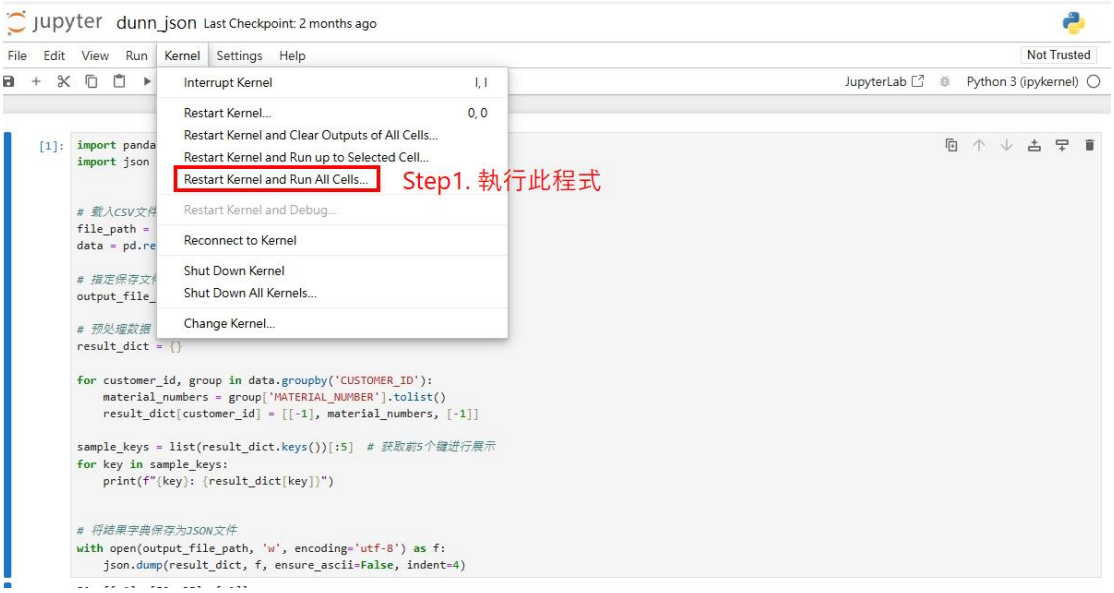
1. 於 DREAM.ipynb 內調整部分設定並直接執行



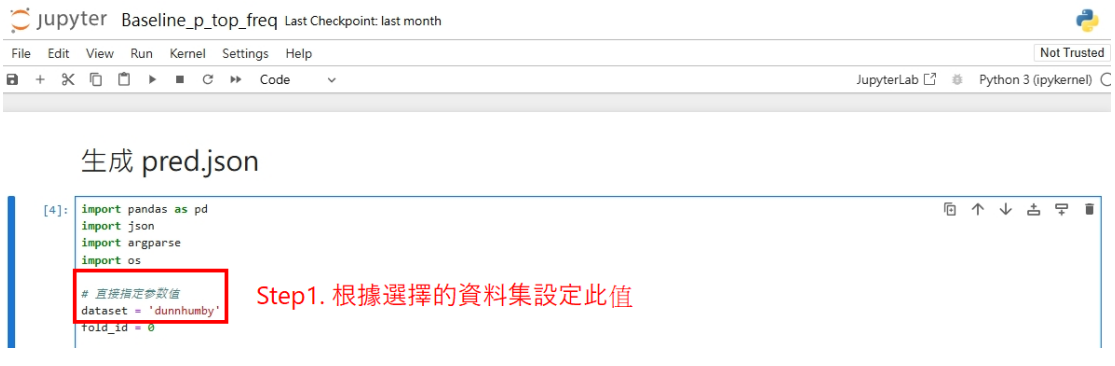


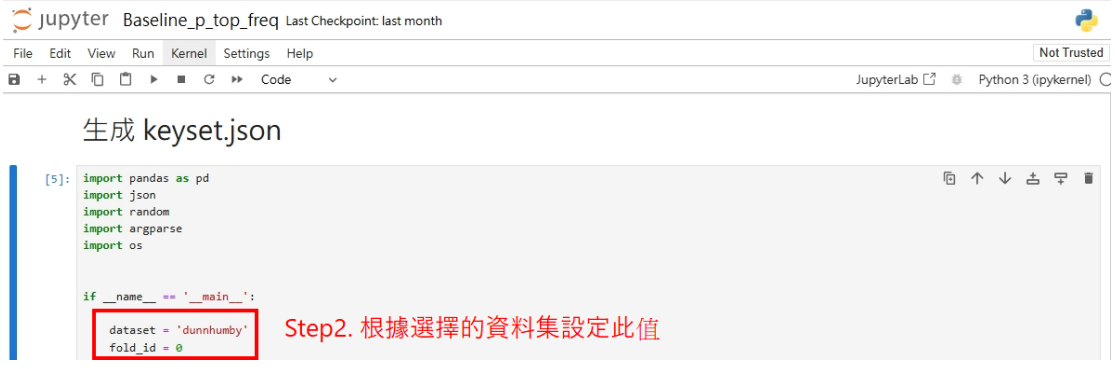
4.2.4 PersonTopFreq

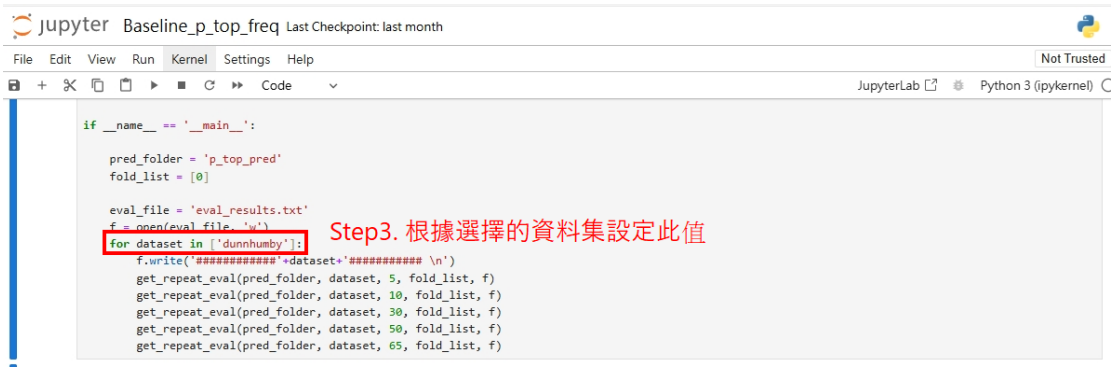
1. 根據資料集直接執行 dunn_json.ipynb 或 ta_feng_json.ipynb (下圖以 Dunnhumby 資料集為例)

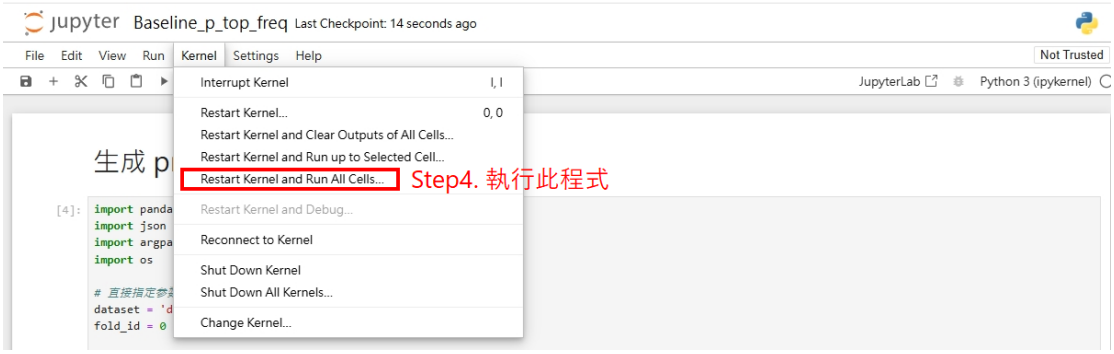


2. 於 Baseline_p_top_freq.ipynb 內調整部分設定並直接執行



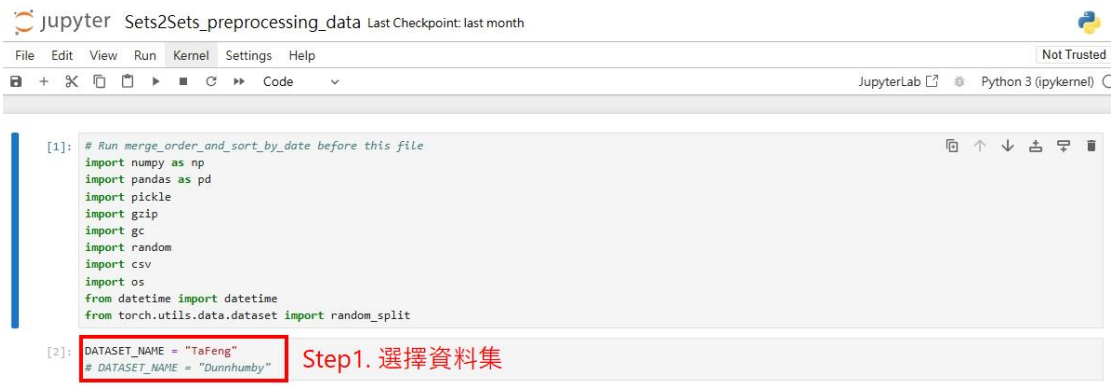



jupyter Baseline_p_top_freq Last Checkpoint: last month
File Edit View Run Kernel Settings Help
Not Trusted
Code
JupyterLab
Python 3 (ipykernel)
if __name__ == '__main__':
pred_folder = 'p_top_pred'
fold_list = [0]
eval_file = 'eval_results.txt'
f = open(eval_file, 'w')
for dataset in ['dunnhumby']:
f.write('############'+dataset+'########## \n')
get_repeat_eval(pred_folder, dataset, 5, fold_list, f)
get_repeat_eval(pred_folder, dataset, 10, fold_list, f)
get_repeat_eval(pred_folder, dataset, 30, fold_list, f)
get_repeat_eval(pred_folder, dataset, 50, fold_list, f)
get_repeat_eval(pred_folder, dataset, 65, fold_list, f)
Step3. 根據選擇的資料集設定此值

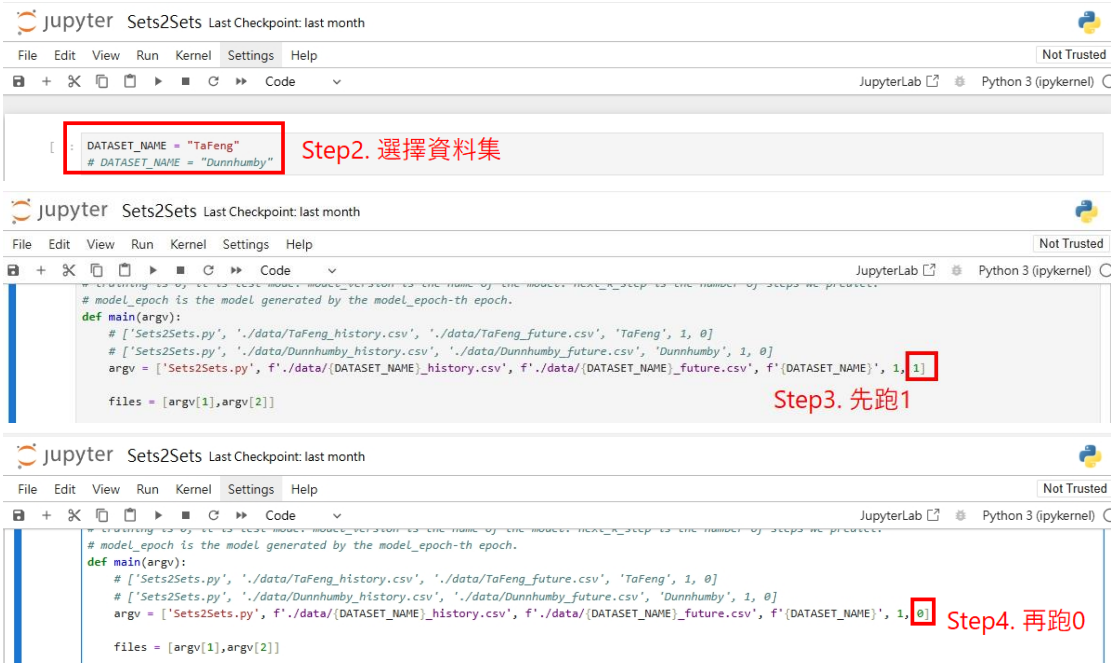

jupyter Baseline_p_top_freq Last Checkpoint: 14 seconds ago
File Edit View Run Kernel Settings Help
Not Trusted
JupyterLab
Python 3 (ipykernel)
Interrupt Kernel I, I
Restart Kernel... 0, 0
Restart Kernel and Clear Outputs of All Cells...
Restart Kernel and Run up to Selected Cell...
Restart Kernel and Run All Cells...
Restart Kernel and Debug...
Reconnect to Kernel
Shut Down Kernel
Shut Down All Kernels...
Change Kernel...
Step4. 執行此程式

4.2.5 Sets2Sets

1. 直接執行 Sets2Sets_preprocessing_data.ipynb

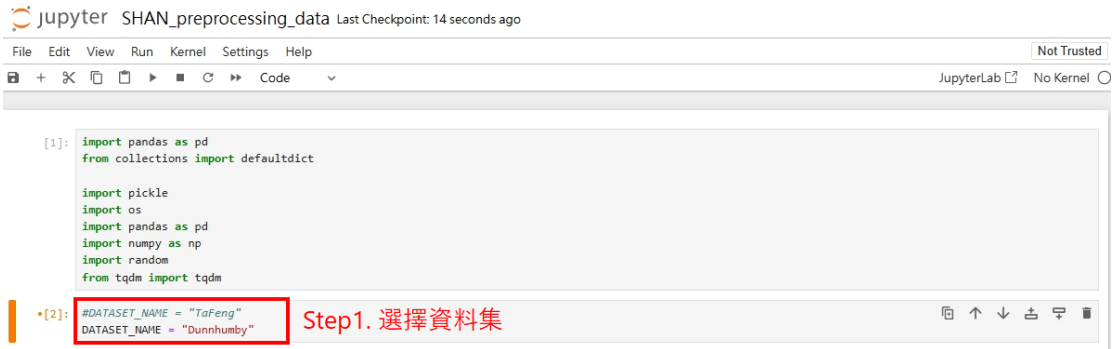


2. 於 Sets2Sets.ipynb 內調整部分設定並直接執行**(注意:此程式碼需執行兩次)**

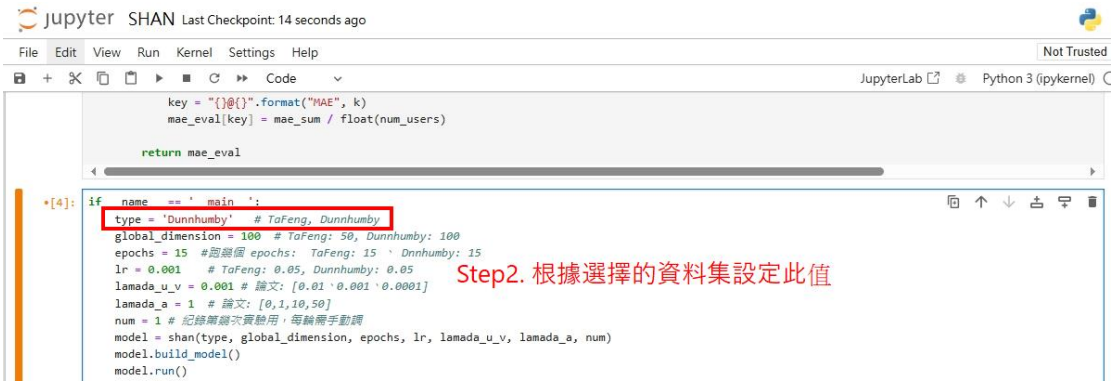


4.2.6 SHAN

1. 直接執行 SHAN_preprocessing_data.ipynb

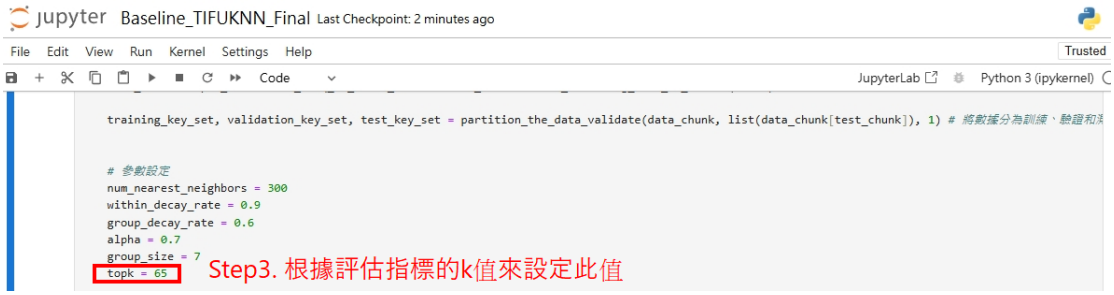


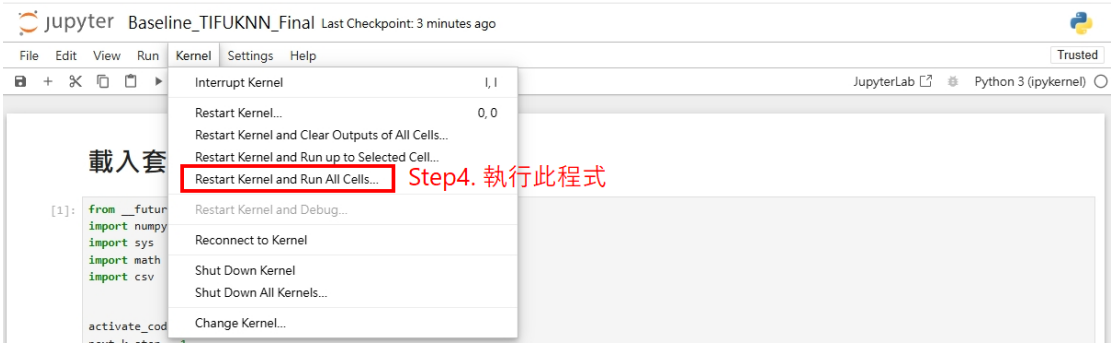
2. 於 SHAN.ipynb 內調整部分設定並直接執行



4.2.7 TIFUKNN

1. 於 Baseline_TIFUKNN_Final.ipynb 內調整部分設定並直接執行





聯絡方式：

學生：林莉庭

手機：0933497107

Email： linnn7322@gmail.com

LIND ID：ting372267