NCKU Sinopac Workflow

本文件主要說明此計畫程式碼內容與使用流程，將整個專案架構拆分成以下功能說明：

1. 資料前處理
2. 個人化推薦系統 VAECF 模型
3. 冷啟動推薦流程說明
4. 可解釋性處理

資料前處理

進行前處理的檔案一覽

- crm.csv
- fund_info.csv
- fund_info_web.csv
- trans_buy.csv
- crm_new.csv
- fund_crm_feature.csv

執行處理

- 我方實驗所使用的前處理內容以編寫於 `recs/datasets/preprocess.py` 中，分別以不同函式名稱對應 `./data/origin` 資料夾中的不同資料表，處理過後會產生新檔案在 `./data` 資料夾中，以下表所示：

原始資料檔案路徑	處理函式名稱	產出檔案
./data/origin/crm.csv	preprocess_crm()	crm.pkl
./data/origin/fund_info_web.csv	preprocess_fund_web()	fund_info_web.pkl
./data/origin/fund_info.csv	preprocess_fund()	fund_info.pkl
./data/origin/trans_buy.csv	preprocess_trans_buy()	trans_buy.pkl
./data/origin/crm_new.csv ./data/origin/fund_crm_feature.csv	preprocess_crm_cluster()	crm_for_clustering_2020.csv

  請確保執行順序一致：

  - 前處理檔案順序如： crm.csv $\rightarrow$ fund_info_web.csv $\rightarrow$ fund.csv $\rightarrow$ crm_new.csv & fund_crm_feature.csv
  - 函式執行順序為： preprocess_crm() $\rightarrow$ preprocess_fund_web() $\rightarrow$ preprocess_fund() $\rightarrow$ preprocess_trans_buy() $\rightarrow$ preprocess_crm_cluster()

- 或使在專案資料夾底下，開啟terminal直接執行以下指令，可一併處理完成

```
python -m recs.datasets.preprocess
```

處理後產出檔案一覽

- crm.pkl
- fund_info_web.pkl
- fund_info.pkl
- trans_buy.pkl
- crm_for_clustering_2020.pkl

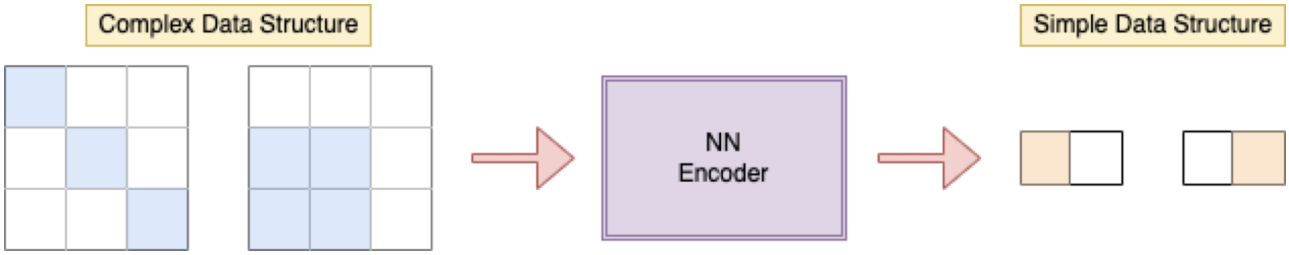
個人化推薦系統 VAECF 模型

程式碼內容可以參閱`./recs/models/vaecf`中的檔案，模型架構可參見以下說明

Autoencoder簡介

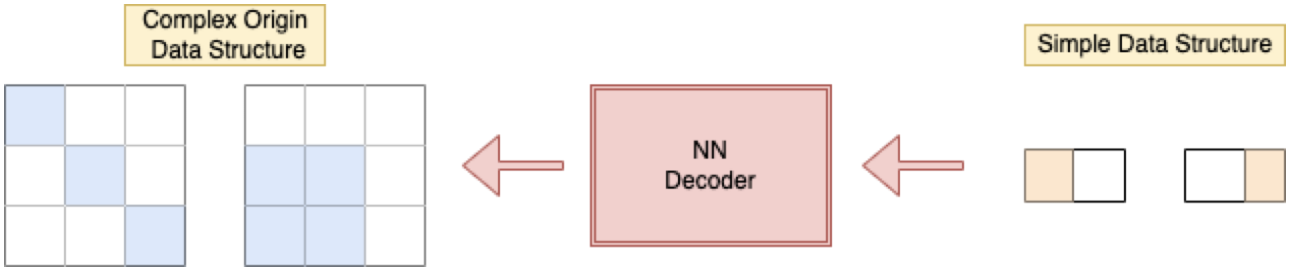
Autoencoder架構會包含Encoder編碼器與Decoder解碼器兩個部分。

- **Encoder編碼器** 負責對Input進行壓縮與降維。



圖一、Encoder 示意圖

- **Decoder解碼器** 負責對encoder壓縮後的資料進行解壓縮與還原成Output。



圖二、Encoder 示意圖

在訓練過程(Training)中，Autoencoder會訓練模型權重使得壓縮過後並解碼的資料(Output)盡可能地還原原始資料(Input)

Variational Autoencoder

VAE的V代表Variational，相較於原本的Autoencoder架構，其應用 Variational Inference 的理論到Autoencoder上。

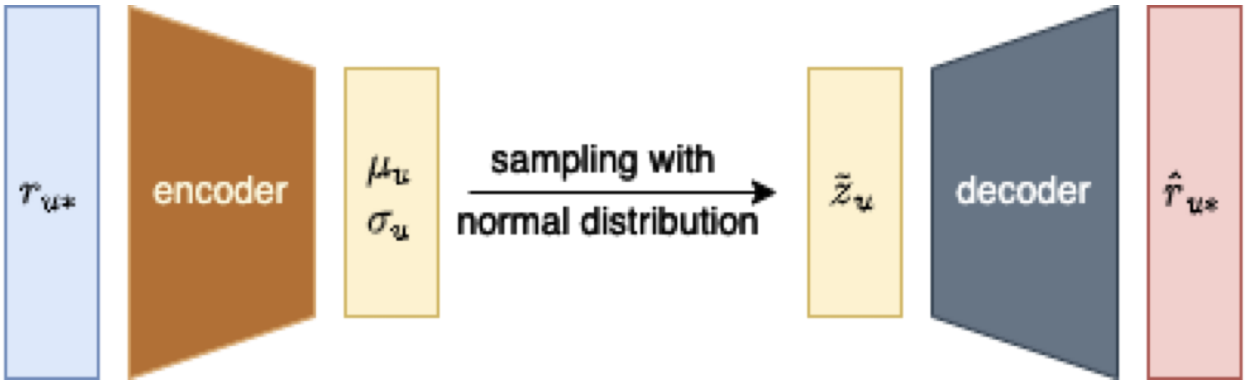
Evidence Lower Bound (ELBO)

為了達成找到$q^*(Z)$的目的，*Variational Inference* 使用 **ELBO**來找出分佈之間的下限，**ELBO** 由*KL-Divergence* 推導而來，公式如下：

$$\mathcal{L}(Q) \equiv \log P(X) - KL\big(q(Z)||P(Z|X)\big)$$

- $KL\big(q(Z)||P(Z|X)\big)$
  - $q(Z)$與$P(Z|X)$間的距離(*KL-Divergence*).
- $\log P(X)$
  - $P(X)$是X的邊際分佈(marginal evidence)，$\log P(X)$ 在此往往是固定值，因此在*ELBO*中將其定義為 *Evidence*.
- $\mathcal{L}(Q)$
  - 即 *ELBO*，根據上述公式，$\mathcal{L}(Q)$ 一定小於等於 $\log P(X)$ (*Evidence*)，因此稱之為 *Evidence* 的下限.
  - Q: 定義 Q 為涵蓋所有 $q(Z)$ 的分佈，為 $q(Z) \in Q$
- 當 **ELBO** 越大，也就代表$q(Z)$ 與 $P(Z|X)$ 間的距離越小，故找到一最大的 **ELBO**，即找到 *Variational Inference* 的 $q^*(Z)$.

VAE 模型學習



圖五、VAE模型架構圖示

定義 Encoder

Encoder 會從 Input Data 中，取出隱變量(latent variable)，並透過兩個Linear層，分別用於取出該變量中位數 μ 與 變異數 σ.

- **Reparameterize**
  透過Encoder 取出的 μ 與 σ 產生一服從常態分佈的新分佈 $\tilde{z}$，作為 *Decoder* 的輸入.
  此產生服從常態分佈的新分佈的過程定義為 *Reparameterize*.

定義 Decoder

Decoder 負責將分佈 $\tilde{z}$ 盡可能地還原為原始的 Input 資料.

定義 Loss function

我們定義可訓練參數 ϕ 與 θ，其意義分別如下：

- ϕ: 定義 q_ϕ 為 *Encoder*，q_ϕ 代表 *Encoder* 中的可訓練參數.
- θ: 定義 p_θ 為 *Decoder*，p_θ 代表 *Decoder* 中的可訓練參數.

透過訓練，找出 ϕ^* 與 θ^*，讓 *Encoder* 壓縮過後的資料透過 *Decoder* 還原之後與原始資料最為相近。 在VAE的訓練中即將 ϕ 與 θ 結合進 ELBO 的公式當作訓練時的 *loss function*，數學式如下：

$$\mathcal{L}(X;\theta,\phi) \equiv \mathbb{E}_{q_\phi(Z|X)}[\log p_\theta(X|Z)] - KL\big(q_\phi(Z|X) \,||\, p(Z)\big)$$

- 在訓練中，會同時訓練與更新 *Encoder* 與 *Decoder* 的參數 ϕ, θ，希望 **ELBO** $\mathcal{L}(X;\theta,\phi)$ 越大越好
- $KL\big(q_\phi(Z|X) \,||\, p(Z)\big)$:
  - 假定先驗分佈 $p(Z)$ 為常態分佈，而經過 Encoder 與 Reparameterize 產生的 $\tilde{z}$ 也是常態分佈，在此狀況下得透過公式計算encoder出來的 *KL-Divergence*.
- $\mathbb{E}_{q_\phi(Z|X)}[\log p_\theta(X|Z)]$:
  - 計算Autoencode 的 Output 與原始資料 Input 間的 **likelihood**.

將VAE應用於推薦系統協同過濾(CF)

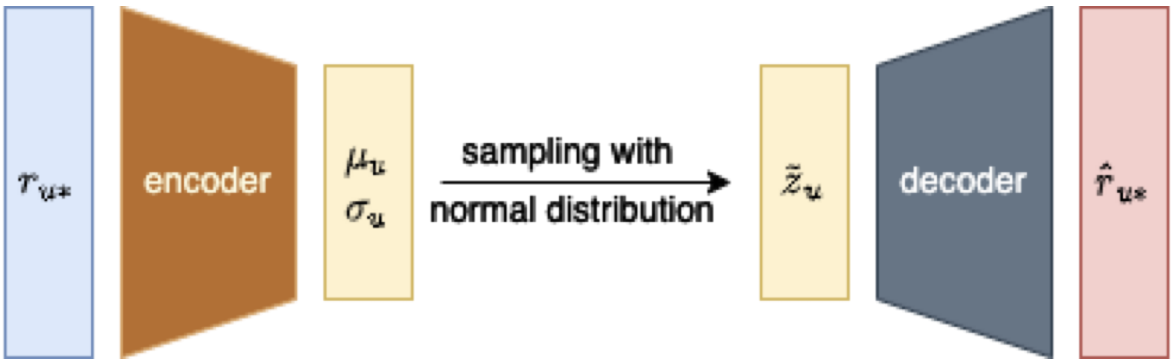
- 將用戶對商品的評分紀錄，轉換成Shape 為 [用戶數量, 商品數量]的 Matrix，如下圖：

itemId	1	10	100	1000	1001	1002	1003	1004	1005	1006	...	990	991
userId													
1	5.0	3.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
10	4.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
100	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	3.0	0.0
101	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
102	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
95	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
96	5.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
97	4.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
98	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
99	4.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0

- 將個別用戶的row取出，作為VAE模型的Input r_{u_*}
  - 用戶

5.0	3.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0

  - shape 為 [1, 商品數量]
  - batch_size 為設定模型每次訓練時的用戶數量，一個epoch要跑(訓練樣本數/batch_size)個 iterations
- 輸入$r_{u_}$ *進模型中，輸出新資料* $\hat{r}_{u}$



  - shape 同樣為 [1, 商品數量]
- **Training Phase:**
  - 將歷史的用戶資料作為Training Data，訓練模型在所有用戶資料下能夠盡可能的還原回原始的資料分佈
- **Inference Phase:**
  - 將 shape 為[1, 商品數量]的用戶評分資料進行 Encoder 與 Decoder 後所得到的新的用戶評分資料即為VAECF預測該用戶對全商品的評分結果

程式碼流程說明

1. 如上述說明，VAECF 主要透過用戶評分資料進行預測，因此我們需要透過 trans_buy 交易資料表產生一評分資料，來供 VAECF 進行預測，評分資料表格式如下：

User	Item	rating
用戶名 (Aiden)	商品名 (apple)	評分 (5.0)

2. 我方使用交易次數來代表評分，每購買一次評分 + 1，轉換程式碼如下，執行後即可產生評分資料格式：

```
trans_buy = pd.read_pickle(DATA_PATH / "trans_buy.pkl") # 讀取資料到
DataFrame
rate_df = trans_buy.groupby(["<USER 欄位名>", "<ITEM 欄位名>"])\
        .size()\
        .reset_index(name="rating")
```

3. 另外為了訓練深度模型，需要將資料輸入是先定義好的 `Dataset` 中，執行以下函式來取得深度學習所需要的 `Dataset` 資料結構：

```
uir_data = rate_df[["<USER 欄位名>", "<ITEM 欄位名>", "rating"]]
train_set = recs.datasets.Dataset.from_uir(uirdata, seed=0)
```

4. 產生模型實例，並使用上述Dataset來訓練模型：

```
vaecf = recs.models.VAECF(
    k=10,
    autoencoder_structure=[20],
    act_fn="tanh",
    likelihood="mult",
    n_epochs=100,
    batch_size=100,
    learning_rate=0.001,
    beta=1.0,
    seed=123,
    use_gpu=True,
    verbose=True,
)

vaecf.fit(train_set)
```

5. 訓練完成後，調用事先定義好`predict_ranking`函式進行預測：

```
df = pd.DataFrame(columns=['userId', 'itemId', 'rating'], data=data)
pred = predict_ranking(vaecf, df, 'userId', 'itemId', 'rating', True)
```

6. 將推薦結果資料儲存到`./data`資料夾，供後續可解釋性使用

```
pred.to_csv(DATA_PATH / "pred.csv", index=False)
```

7. 若要儲存模型，執行pytorch一般的 save 與 load 流程後，即可直接取用訓練好的模型

冷啟動推薦流程說明

使用資料檔案

- crm_new.csv
- fund_xxx.csv
- trans_buy.csv

程式碼流程說明

1. 從前處理完成後的資料中取出已交易用戶的資料，如下圖：

```
u2idx = readjson2dict("crm_idx")
i2idx = readjson2dict("fund_info_idx")
tran_data = pd.read_pickle(DATA_PATH / "trans_buy.pkl")
```

```
# 把 index 化的 id 轉回來
tran_data["id_number"] = tran_data["id_number"].progress_apply(
    lambda x: id2cat(u2idx, x))
tran_data["fund_id"] = tran_data["fund_id"].progress_apply(
    lambda x: id2cat(i2idx, x))

# 讀取分群用的 crm 資料表(透過 recs.datasets.preprocess.crm_new 產生)
crm = pd.read_csv(DATA_PATH / "crm_for_clustering_2020.csv")
crm = crm[crm.yyyymm == 202012]

# 目標用戶
target_users = crm['id_number'].unique().tolist()
```

2. 我們要對未交易用戶 進行推薦，我們透過 `crm_for_clustering_2020.csv` 資料表中的 `has_traded` 欄位取出未交易資料者。並且取出我們需要做分群需要的資料：

```
new_user = crm[crm['has_traded'] == 0]['id_number'].unique().tolist()

# 分群需要的資料
user_data = crm.copy()
u_data = user_data.drop(columns=['id_number', 'CIFAOCODE'])
u_data = u_data.replace(-np.inf, -1)
u_data.fillna(0, inplace=True)
```

3. 透過 umap 對特徵進行降維，使用在`./recs/models/cluster/clustering.py` 中定義好的 `feature_selecter` 函式進行特徵擷取，我方使用經回測得出效果最好的 Hyperparameter 進行降維：

```
# Hyperparameter
embedding_method = "umap"
n_components = 25
x_embedded = feature_selecter(u_data, embedding_method, n_components)
```

4. 以 dbscan 演算法進行分群，使用在`./recs/models/cluster/clustering.py` 中定義好的 `clustering` 函式進行分群，並且在資料上添加`cluster`欄位來表示該 user 分配到哪個群體中

```
c_user_data = clustering(user_data, x_embedded,
                         'dbscan', eps=6.5, min_samples=5)
```

5. 根據分群結果取各群體的Top N 銷售結果，同樣使用在`./recs/models/cluster/clustering.py` 中已定義好的 `recommender_method` 函式，即可得到各個群體所推薦的 Top N 推薦結果，程式碼如下：

```
cluster_top10 = recommender_method(tran_data, c_user_data)
```

6. 最後以此結果推薦給未交易用戶

```
result_by_cluster = {}
for u in tqdm(new_user):
    temp = user_data[user_data['身分證字號'] == u] # 或 id_number
    t_cluster = temp.cluster.tolist()[0]
    result_by_cluster[u] = cluster_top10[t_cluster]
```

可解釋性處理

使用資料檔案

- pred.csv (VAECF產生)
- 前處理後產生資料
  - crm_for_clustering_2020.csv
  - crm_idx.json
  - fund_info_idx.json

程式碼流程說明

1. 讀取所需資料，並且將資料表一同合併

```
u2idx = readjson2dict("crm_idx")
i2idx = readjson2dict("fund_info_idx")

pred = pd.read_csv(DATA_PATH / "pred.csv")
pred['userId'] = pred['userId'].progress_apply(lambda x: id2cat(u2idx,
x))
pred['itemId'] = pred['itemId'].progress_apply(lambda x: id2cat(i2idx,
x))
crm = pd.read_csv(DATA_PATH / "crm_for_clustering_2020.csv")
crm['userId'] = crm['id_number']
crm.drop(columns=['id_number', 'index'], inplace=True)

fund = pd.read_pickle(DATA_PATH / 'fund_info.pkl')
fund['itemId'] = fund['fund_id'].progress_apply(lambda x:
id2cat(i2idx, x))
fund.drop(columns=['fund_id', 'currency_type'], inplace=True)

pred = pred.merge(crm, how='left', on=['userId'])
pred = pred.merge(fund, how='left', on=['itemId'])
```

2. 切分訓練輸入x與預測欄位y

```
keys = pred[['userId', 'itemId']]
y = pred['rating']
x = pred.drop(columns=['userId', 'itemId', 'rating', 'CIFAOCODE'])
```

3. 訓練 LGBMRegressor 模型

```
model = LGBMRegressor(
    boosting_type="gbdt",
    verbose=1,
    random_state=47
)
model.fit(x, y)
```

4. 計算 shap value，來計算參數對模型輸出的影響

```
explainer = shap.Explainer(model)
shap_vals = explainer(x)
feature_names = shap_vals.feature_names
```

5. 將解釋性結果輸出

```
value_df_dict = {}
base_val = shap_vals.base_values[0]

for i in tqdm(shap_vals):
    i_sum = i.values.sum()
    for jidx, j in enumerate(feature_names):
        data_val = i.data[jidx]
        val = round(i.values[jidx] / (i_sum - base_val), 4)
        save_val = (val, data_val)
        if value_df_dict.get(j, 0) == 0:
            value_df_dict[j] = [save_val]
        else:
            value_df_dict[j].append(save_val)

df = pd.DataFrame(value_df_dict)
```

資料夾結構說明

```
.
|-- .gitignore
|-- README.md
|-- data
```

```
|-- result
|-- examples
|   |-- mf_example.py
|   `-- vaecf_example.py
|-- experience
|-- img
`-- recs
    |-- __init__.py
    |-- backtest.py
    |-- datasets
    |   |-- __init__.py
    |   |-- base.py
    |   `-- preprocess.py
    |-- metrics
    |   `-- base.py
    |-- models
    |   |-- __init__.py
    |   |-- base.py
    |   |-- cluster
    |   |   |-- __init__.py
    |   |   `-- clustering.py
    |   |-- explain
    |   |   `-- explain.py
    |   |-- mf
    |   |   |-- __init__.py
    |   |   `-- mf.py
    |   `-- vaecf
    |       |-- __init__.py
    |       |-- recom_vaecf.py
    |       `-- vaecf.py
    `-- utils
        |-- __init__.py
        |-- common.py
        |-- config.py
        `-- path.py
```

- 元資料結構其實已經對實驗性和使用的code進行區分
  - 在 `./experiments` 中的 ipynb 檔案即計畫過程中所進行實驗的程式碼，若無需使用可以直接移除
- 個人化推薦，可執行 `vaecf_example.py` 來得到與實驗相同的推薦結果，會產生檔案放在 `./result` 中
- 冷啟動推薦，可執行以下程式碼得到上月繳交的結果，同樣產生於`./result` 中

```
python -m recs.models.clsuter.clustering
```

- 可解釋性部分，同樣可執行以下程式碼得到上月繳交的結果於`./result` 中

```
python -m recs.models.explain.explain
```