資料科學期末報告
有夠會推薦 -- Santander Product Recommendation

第25組 R08943095 張惟筑 R09921041 顏敬祐 R09942082 楊雅婷

Video link: group_25.mov

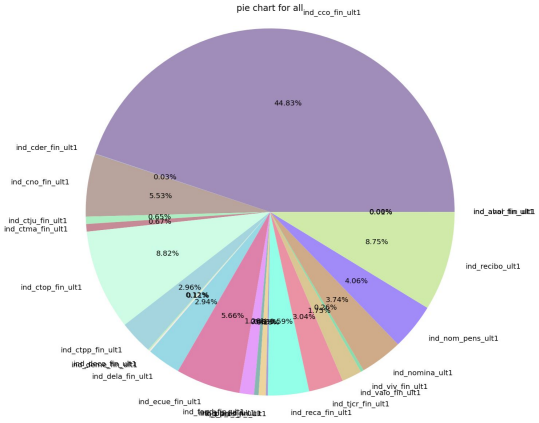
I. Project Description

商品推薦的決策在很多的商業模式中扮演重要的角色。為了透析推薦商品的決策，我們藉由傳統的統計分析、決策樹模型到近代的深度神經網路的技術來分析推薦的決策並且如何提高推薦成功率。我們發現雖然傳統的統計、決策樹分析雖然快速且能被詮釋，但藉由深度學習的技術我們能試圖找到潛在不易解讀的關係或資料型態。我們藉由預訓練的 Auto-Encoder 作為特徵選擇器的狀況下訓練模型，發現在 Recall 上有顯著的進步，也符合推薦的宗旨。

II. 資料分析

1. 對整個資料集的商品進行分析

可以看出商品被購買的頻率非常不一，購買頻率最高的商品佔了將近 45% 的比例。

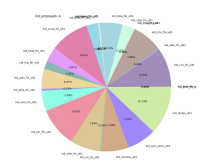
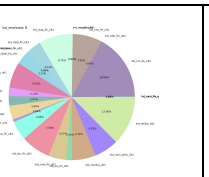
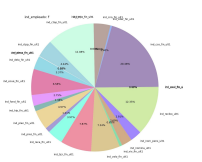
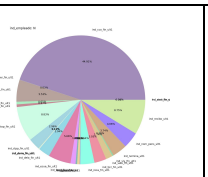
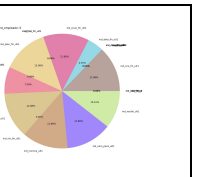


(圖1) 24種商品在整個資料集中所佔的比例

2. 針對不同的 client feature，去觀察商品被購買的比例分佈

可以看出用 empleado 或是 segmento 去區分購買商品的比例會有蠻大的差異；而原本以為可能會有一些季節性的商品，但若是用月份去分析卻在大致上看不出差別。

舉empleado為例：

				
(圖2) empleado A	**(圖3) empleado B**	**(圖4) empleado F**	**(圖5) empleado N**	**(圖6) empleado S**

除此之外還找到一些特例，不過這些情況可能只佔整個資料集的一小部分：

For pais_residencia	
AL, MZ, TW, NI, GT, BA, MD, PK, SL, GN, GW, LV, ML, MK, IS, LY, CF, KZ, CD, BZ	buy 100% ind_cco_fin_ult1
EC	buy 76% ind_cco_fin_ult1 and 24% ind_ctop_fin_ult1
MR, SK	buy 75% ind_cco_fin_ult1 and 25% ind_ctop_fin_ult1
EE, HR, PH, KH	buy 50% ind_cco_fin_ult1 and 50% ind_ctop_fin_ult1
GI	buy 50% ind_cco_fin_ult1 and 50% ind_valo_fin_ult1
RS	buy 50% ind_ctop_fin_ult1 and 50% ind_ctpp_fin_ult1
For tiprel_1mes	
tiprel_1mes == “N”	buy 50% ind_cco_fin_ult1 and 50% ind_ctju_fin_ult1

3. 去找整個資料集商品的 association rules

	antecedents	consequents	confidence
0	ind_nomina_ult1	ind_nom_pens_ult1	1.000000
1	ind_recibo_ult1, ind_nomina_ult1, ind_cno_fin_ult1	ind_nom_pens_ult1	1.000000
2	ind_nomina_ult1, ind_cno_fin_ult1	ind_nom_pens_ult1	1.000000
3	ind_recibo_ult1, ind_nomina_ult1	ind_nom_pens_ult1	1.000000
4	ind_recibo_ult1, ind_nomina_ult1	ind_cno_fin_ult1	0.949016
5	ind_recibo_ult1, ind_nomina_ult1	ind_nom_pens_ult1, ind_cno_fin_ult1	0.949016
6	ind_recibo_ult1, ind_nomina_ult1, ind_nom_pens_ult1	ind_cno_fin_ult1	0.949016
7	ind_recibo_ult1, ind_nom_pens_ult1	ind_cno_fin_ult1	0.947583
8	ind_nomina_ult1	ind_cno_fin_ult1	0.940191
9	ind_nomina_ult1	ind_nom_pens_ult1, ind_cno_fin_ult1	0.940191
10	ind_nomina_ult1, ind_nom_pens_ult1	ind_cno_fin_ult1	0.940191
11	ind_nom_pens_ult1	ind_cno_fin_ult1	0.938339
12	ind_recibo_ult1, ind_nom_pens_ult1, ind_cno_fin_ult1	ind_nomina_ult1	0.924291
13	ind_recibo_ult1, ind_nom_pens_ult1	ind_nomina_ult1	0.922896
14	ind_nom_pens_ult1, ind_cno_fin_ult1	ind_nomina_ult1	0.922661
15	ind_nom_pens_ult1	ind_nomina_ult1	0.920843
16	ind_recibo_ult1, ind_nom_pens_ult1	ind_nomina_ult1, ind_cno_fin_ult1	0.875843
17	ind_nom_pens_ult1	ind_nomina_ult1, ind_cno_fin_ult1	0.865768

(圖7) 找到的 association rules

因為有購買商品的畢竟佔整個資料集的少數，所以這裡列出的是(support >= 0.05 且 confidence >= 0.85的)，不過這樣的作法可能就比較沒辦法找到比較少被購買的商品之間的關聯。除此之外，因為最後想要測試的資料集也是以月份為單位，因此我們也有去針對個別月份，找商品之間的 association rules。不過後來發現，對於經常被購買的商品來說，月份之間 rules 的差異並不大。

4. 透過過去的購買行為作分析

我們想要針對同一個使用者過去曾經購買過的商品做分析，統計所有pattern出現的次數，再根據他後續可能會出現的最可能pattern做預測。以下圖為例，假如他過去買item1再買item2，再買item4，那他再購買其他商品的各個機率是多少。

(圖8) behavior tree 示意圖

我們發現某些 patterns 出現的次數非常多，為了讓結果更容易解讀並且找到有趣的關係，我們傾向尋找短 (length < 5) 的 pattern，採用 FIFO 的方式 prune 掉舊的資料。

```
('t_2',) 466079
('t_2', 't_21') 328880
('t_2', 't_21', 't_22') 328880
('t_2', 't_21', 't_22', 't_2') 303108
('t_2', 't_21', 't_22', 't_2', 't_21') 301473
('t_2', 't_21', 't_22', 't_2', 't_21', 't_22') 301473
('t_2', 't_21', 't_22', 't_2', 't_21', 't_22', 't_2') 294051
('t_2', 't_21', 't_22', 't_2', 't_21', 't_22', 't_2', 't_21') 293439
('t_2', 't_21', 't_22', 't_2', 't_21', 't_22', 't_2', 't_21', 't_22') 293439
('t_2', 't_21', 't_22', 't_2', 't_21', 't_22', 't_2', 't_21', 't_22', 't_2') 288050
('t_2', 't_21', 't_22', 't_2', 't_21', 't_22', 't_2', 't_21', 't_22', 't_2', 't_21') 287632
('t_2', 't_21', 't_22', 't_2', 't_21', 't_22', 't_2', 't_21', 't_22', 't_2', 't_21', 't_22') 287632
('t_2', 't_21', 't_22', 't_2', 't_21', 't_22', 't_2', 't_21', 't_22', 't_2', 't_21', 't_22', 't_2') 283716
('t_2', 't_21', 't_22', 't_2', 't_21', 't_22', 't_2', 't_21', 't_22', 't_2', 't_21', 't_22', 't_2', 't_21') 283408
('t_2', 't_21', 't_22', 't_2', 't_21', 't_22', 't_2', 't_21', 't_22', 't_2', 't_21', 't_22', 't_2', 't_21', 't_22') 283408
('t_2', 't_7') 66034
('t_4',) 50341
('t_2', 't_7', 't_21') 42586
('t_2', 't_7', 't_21', 't_22') 42586
('t_0',) 40892
('t_2', 't_7', 't_21', 't_22', 't_2') 39314
('t_2', 't_7', 't_21', 't_22', 't_2', 't_7') 39237
('t_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21') 39029
('t_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22') 39029
('t_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22', 't_2') 38478
('t_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22', 't_2', 't_7') 38408
('t_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21') 38220
('t_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22') 38220
('t_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22', 't_2') 37746
('t_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22', 't_2', 't_7') 37654
('t_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21') 37494
('t_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22') 37494
('t_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22', 't_2') 37224
('t_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22', 't_2', 't_7') 37174
('t_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21') 37004
('t_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22', 't_2', 't_7', 't_21', 't_22') 37004
```

(圖9) 整理發現某些特定的 pattern 很常出現

```
('t_21', 't_22', 't_2', 't_21', 't_22') 1187813
('t_2', 't_21', 't_22', 't_2', 't_21') 1182532
('t_22', 't_2', 't_21', 't_22', 't_2') 875323
('t_2',) 466079
('t_2', 't_21') 328880
('t_2', 't_21', 't_22') 328880
('t_2', 't_21', 't_22', 't_2') 303108
('t_21', 't_22', 't_2', 't_7', 't_21') 156834
('t_22', 't_2', 't_7', 't_21', 't_22') 156834
('t_2', 't_7', 't_21', 't_22', 't_2') 156694
('t_7', 't_21', 't_22', 't_2', 't_7') 156641
('t_0', 't_3', 't_22', 't_23', 't_0') 111412
('t_22', 't_23', 't_0', 't_3', 't_22') 110839
('t_23', 't_0', 't_3', 't_22', 't_23') 110839
('t_3', 't_22', 't_23', 't_0', 't_3') 110686
('t_21', 't_22', 't_7', 't_21', 't_22') 91699
('t_7', 't_21', 't_22', 't_7', 't_21') 91478
('t_21', 't_22', 't_23', 't_2', 't_21') 78845
('t_22', 't_23', 't_2', 't_21', 't_22') 78845
('t_2', 't_21', 't_22', 't_23', 't_2') 77496
('t_22', 't_7', 't_21', 't_22', 't_7') 68312
('t_23', 't_2', 't_21', 't_22', 't_23') 67845
('t_2', 't_7') 66034
```

(圖10) 採用 FIFO pruning 得到的短 patterns

我們發現雖然第二個商品 (t_2) 是出現最多的，有 466079 筆，可是第二個商品買完再直接買第二個商品 (t_2->t_2) 的樣本數是 0。我們認為這些短 patterns 在某種程度上能作為後續推薦的依據，故在後面做特徵工程的處理上會將這種前後關係的特徵納入考量。accuracy (選機率最大的為下一個的命中機率) 為0.4, top_5 (選機率前五大的為下一個的命中機率) 預測中的機率為0.91。

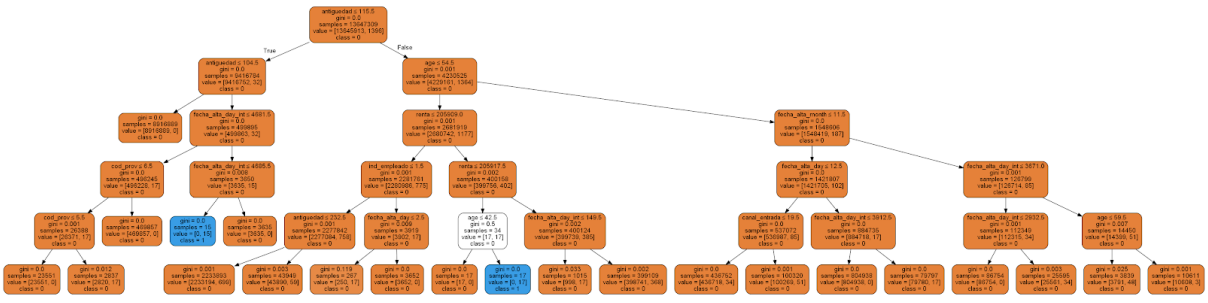
III.模型分析

1. Decision-Tree based

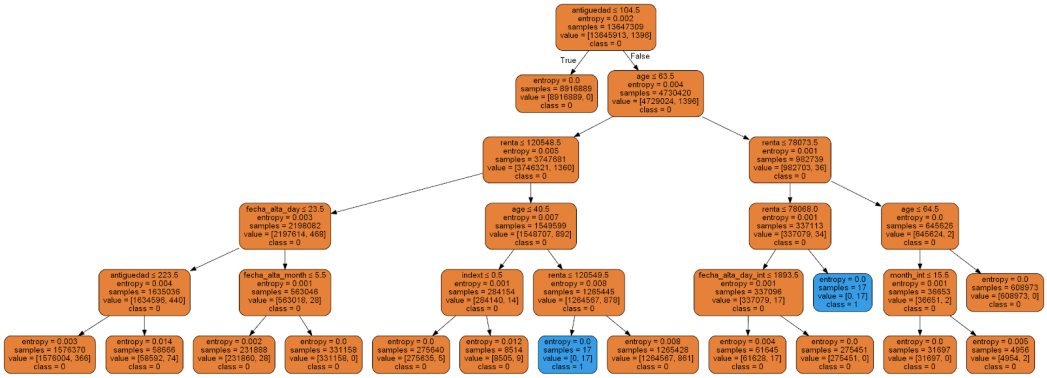
為了讓決策更能夠被人類解讀，我們選用決策樹 (decision tree based) 的方法分析。藉由限定樹的深度 (depth) 來讓結果更加能夠被人類分析。

a. Intuitive CART method

我們選擇 CART 為實作的模型，在樹高並不高並直觀的對 24 種產品映射一棵樹觀察並比較在不同 criterion (gini, entropy)的限制下的結果(圖一)(圖二)。



(圖11) ind_ahor_fin 產品的決策樹 (max_depth=6, criteron=gini)



(圖12) ind_ahor_fin 產品的決策樹 (max_depth=6, criteron=entropy)

交互比對結果後，可以歸納幾個 CART 分析的結論:

⇒ 若數據小且乾淨，CART 能快速且有效的歸納出分類決策
如上述圖例，對某些商品而言，CART 能產生簡易好懂的決策。實際比對資料欄位的敘述，也能與之連結。如對保證金 (Guarantees) 商品而言，我們可以分析出有兩種類型的顧客會買單：(1) 老顧客且顧客年紀大且中低收入 (2) 老顧客且青壯年且中等收入 。假設未來要推薦商品時，便可利用 CART 快速識別淺顯的決策並推薦給目標顧客。

⇒ Gini 和 Entropy 可以找出不同的決策，但分類結果上來說差不多
雖然在不同的 criterion 下可以找到不同的決策，但分出乾淨的資料來說效果有限。若使用情境上較在乎可被解釋的決策，使用不同的 criterion 和參數 (max_depth) 來蒐集決策是個可行的方向。然而若是更在乎那些隱藏在資料中的訊息 (hard to detect information)，捨棄掉可讀性並導入更先進的算法 (boosting) 或模型 (xgboost) 等反而可能是更好的選擇。

b. xgboost method

資料前處理

- 對類別性的 features (如：ind_empleado, tiprel_1mes……)進行編碼，類別性的NaN補0
- 對於連續性 features (如：age, renta, antiguedad……)的NaN補上中位數
- 針對”ncodpers”合併前N個月份的商品購買紀錄

訓練的部份

- 針對每種商品訓練一個分辨「會買」或是「不會買」的分類器
- 因為買/沒買分布不均的緣故，僅用資料集中至少有購買1種商品的row進行訓練
- 用 xgboost

  max_depth=6, learning_rate = 0.05，subsample = 0.9, colsample_bytree = 0.9,

  n_estimators=100, base_score = 該商品在訓練集中出現的頻率, nthread=2

測試以及評估

因為買/沒買的分佈極不平均，若是僅用預測的方式，分類器大都會傾向於分為「不買」的類別，雖然accuracy可能可以很高，但卻不是我們想要的結果，目標反而是找到顧客最有可能買的商品。

所以我們去找了機率前五大的商品，去算他們命中的機率來作為評估

- target: 顧客實際買的商品清單
- emp: 實際上顧客一樣商品都沒買的次數
- pred: 機率前五大的商品清單
- cnt: 命中次數(前五大商品只要有一樣在實際買的商品清單中便算命中)
- 評估標準：cnt / (總共預測筆數 - emp)

若是 antecedents 在推薦清單中，則把 consequents 也加入推薦清單，結果上似乎差不多。

```
測試筆數:          931453
命中:              27530
target empty:   901930
機率:              0.9324933103004437
```

(圖13) 加入前的機率

```
測試筆數:          931453
命中:              27549
target empty:   901930
機率:              0.9331368763337059
```

(圖14) 加入後的機率

2. Neural Network based method

為了挖掘出潛在不易分析的資料，我們嘗試導入神經網路作為模型並分析其結果。我們會先用一般多層 linear layer 的模型作為基礎分析，再嘗試使用DNN技術從訓練資料中萃取出潛在的資訊。

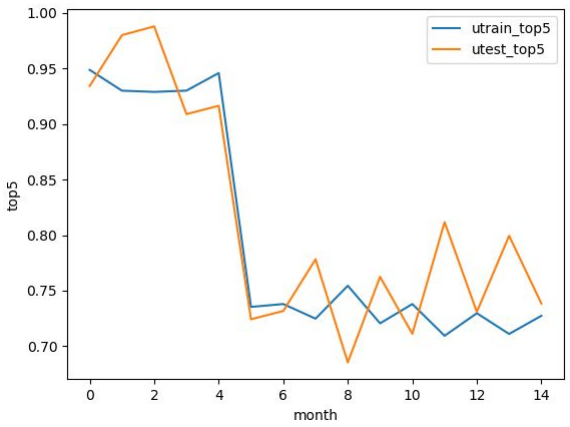
a. Basic Neural Networks Model

資料前處理

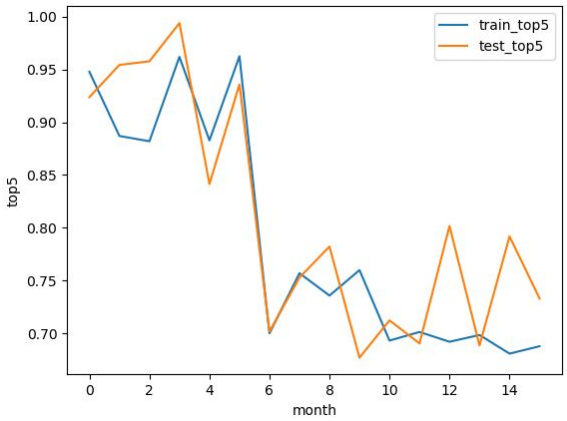
- 我們將部分的cardinal資料做one hot encoding
- 每個顧客有各自未買某個商品的時間，透過每一筆raw data紀錄上次購買這個商品的時間，新增24項feature，記錄他們已經多久沒有買某個商品。

(圖15) 新增feature: 距離上次購買商品的時間

觀察結果：



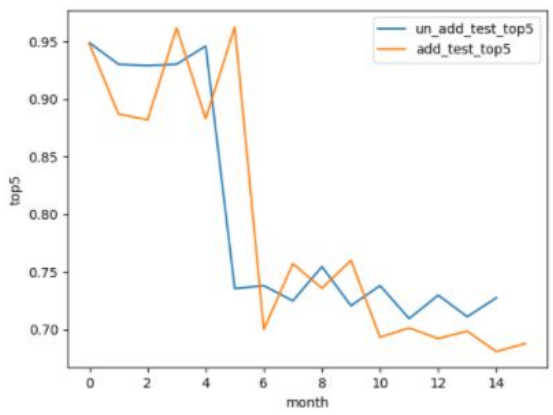
(圖16) 未考慮多久沒買



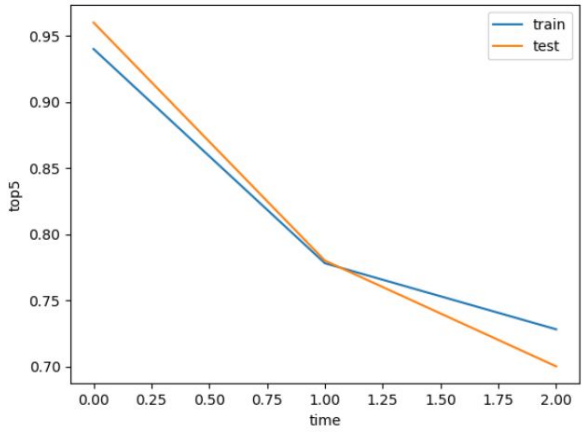
(圖17) 加入考慮多久沒買

我們想觀察不同的時期做會不會有不同的結果

例如短週期的資料自己訓練一個模型以及長週期的資料自己訓練一個模型accuracy會不會有很大的差異。所以我們將資料分成數個月，觀察資料的季節性、流行性。我們發現某些月份非常好預測，有些較困難。觀察結果：



(圖18) 每個月訓練一個model，不同model的top5



(圖19) 依時間長度切三等份，訓練三個model，不同model的top5

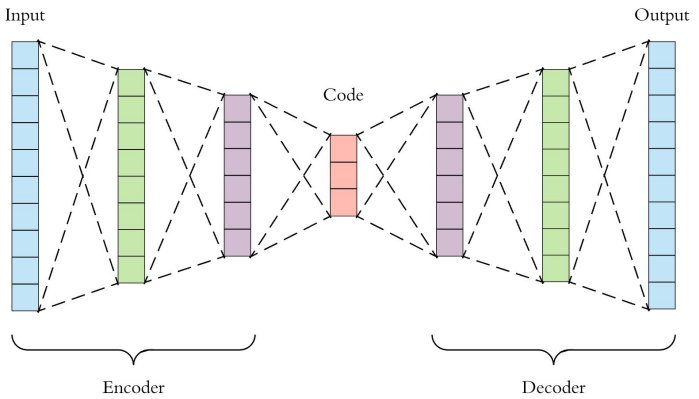
小結論：

我們發現將商品已經多久沒有買當作feature對於NN來說幫助並不大。

b. DNN based analysis

有別於在 kaggle 數據表現良好的 xgboost 模型，我們提出一個以深度學習 (Deep Learning) 為基礎的降維流程，旨在找出具代表性的維度向量 (embeddings) 並驗證其對後續分類/推薦等問題的幫助。

(1) Encode message with Auto-Encoder



(圖20) Auto-Encoder由一個 encoder 和 decoder 組成。藉由訓練還原input的流程，訓練出最具代表的組成 (code, a.k.a embeddings)

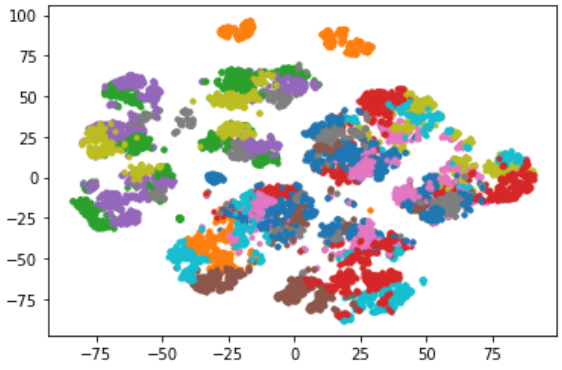
為了提高模型的解讀性，我們嘗試從資料中提取 (extract) 重要的資訊，有別於傳統的主成分分析 (PCA)，我們嘗試導入 DNN based 的 Auto-Encoder (如圖三)。在訓練好 auto encoder 後，只取前段 encoder 作為特徵選擇器。

(2) The robustness of encoder

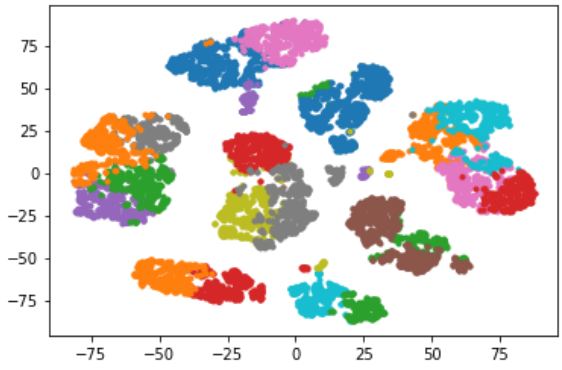
為了驗證前段訓練出的 encoder 可行性，我們用了非監督式分類 (unsupervised clustering) 與監督式分類 (supervised classification) 方法來討論。

- Unsupervised Clustering

我們使用傳統的 kmeans (k = 24) 幫我們解讀 encoder 找出的 embeddings 的品質。藉由tSNE的算法，我們將分類出來的點投射在二維平面上並觀察其分類品質:



(圖21) Clustering row datas



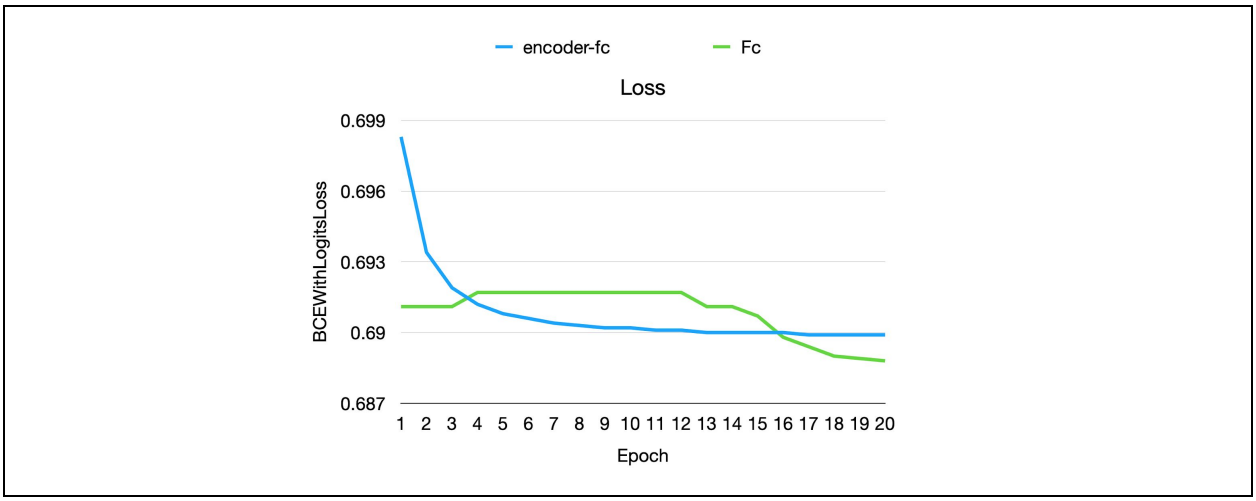
(圖22) Clustering embeddings

可以明顯觀察到 kmeans 在原始資料 (row datas) 的分群效果是有限的。相比之下，經由 encoder 產生的 embeddings 分群的效果十分顯著。

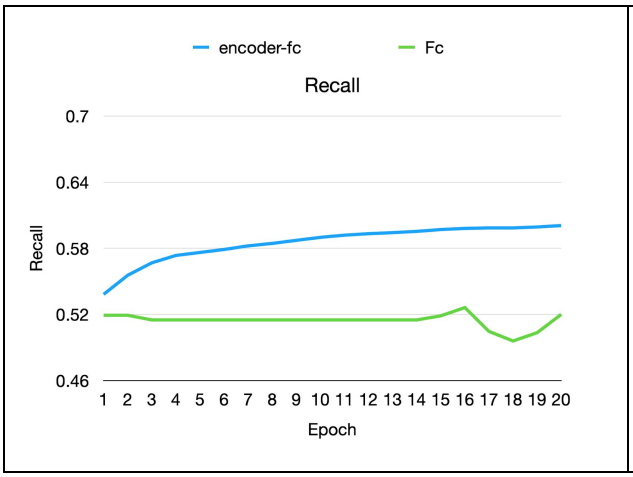
- Supervised Learning

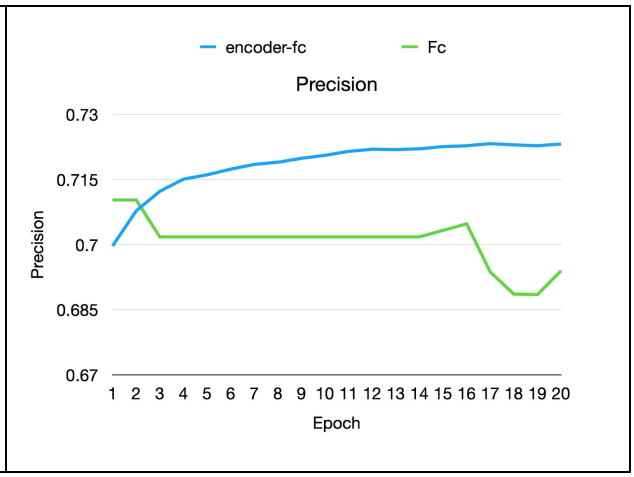
我們將推薦問題簡化成分類問題。對於原題目而言，我們會取輸出最高值的產品為預測結果。然而對推薦問題，我們會取前五高 (top 5) 輸出的產品作為推薦結果。在模型的訓練上，我們使用 模型的。為了驗證 Auto-Encoder 做顏特徵選擇的可行性，將會對Precision, Recall 和 Loss 做分析。

Loss:



Recall & Precision:





藉由 Loss, Recall & Precision的結果我們有幾個結論：

⇒ Encoder作為特徵選擇器可以避免模型過擬合的問題

我們發現雖然在後期，基礎 baseline 的模型雖然 loss 有些許的下降，但在 Precision 和 Recall 的結果上卻是起伏不定，因此我們判斷baseline並沒有學習到有效的知識。相對的在 encoder 的幫助下反而能正確地學到知識，並且成功反映在商品的推薦上。

⇒ Encoder能幫助學習如何推薦產品

回顧 Recall 的結果，和基礎 baseline 相比，我們發現 encoder 的幫助會使得 Recall 有顯著的成長，我們判斷他能有效的學習到推薦的知識。

IV. Conclusion

對於這次的時間序列 (time series) 資料，我們使用了傳統的數據統計 (statics)、決策樹分析 (decision tree)與近代的深度神經網路學習 (Deep Neural Network) 的方式進行分析。在傳統的統計、決策樹分析上，雖然能找到一些規則，但在後續神經網路的學習過程中，我們發現用這些規則所設計做的特徵工程對整體的學習成果並不是那麼有效，因此我們認為導入近代的深度學習作為分析工具是必要的。

導入深度學習的分析後，經由分群 (clustering)，我們發現一個預訓練好的 Auto-Encoder 的 Encoder 在分群上能有好的結果，並且能當其他神經網路的特徵選擇器。實驗結果也發現以 pretrained encoder 作為特徵選擇器的模型有更高的 recall，也符合尋找潛在顧客的目標。

IV. Future works

經過實驗，我們認為深度學習的技術是具有潛力挖掘更難歸納的資料關係。除了調整超參數外，我們預期導入更先進的模型或是所謂的資料蒸餾 (data distillation) 的技術也能有很好的結果。

V. Reference

1. https://www.kaggle.com/c/santander-product-recommendation/overview
2. http://rasbt.github.io/mlxtend/api_subpackages/mlxtend.frequent_patterns/
3. https://xgboost.readthedocs.io/en/latest/
4. https://en.wikipedia.org/wiki/Autoencoder