# Final Project Report—臺灣海洋廢棄物預測

Team member
109061517 邱俊嘉 / 109061520 陳俊宇 / 109061807 吳亞澤

## -Title:

臺灣海洋廢棄物預測

## -議題簡介:

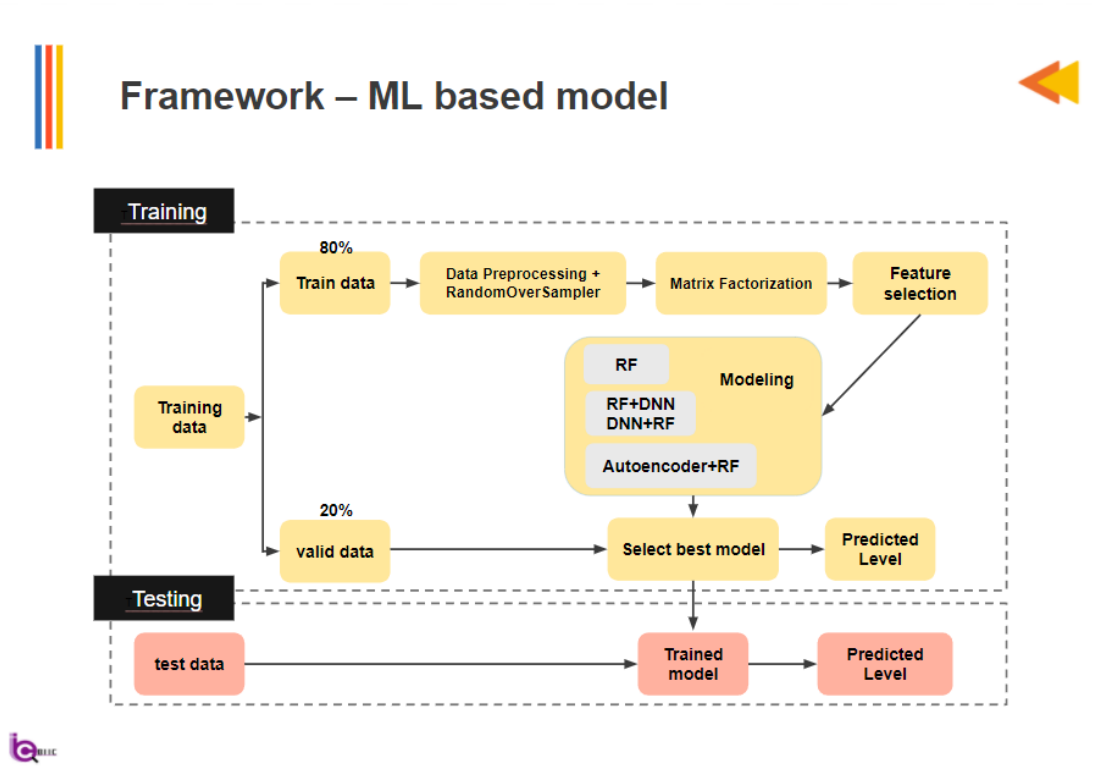
海岸廢棄物快篩調查可在短時間內做大範圍的抽樣調查，並量化廢棄物，可作為測量的方法之一，供淨灘選址參考。快篩的抽樣方式為於海岸線每隔 10 公里取一測站，以臺灣本島 1,210 公里海岸線為母體，即有 121 個測站。本議題希望藉由測站資訊預測相近測站的資訊，以達到減少測站和人力。

## -關鍵字:

機器學習、深度學習、資料分析、神經網路模型、隨機森林模型、支援向量機。

## -Methods:

**總流程**




### 1 資料處理與分析

根據我們過去的經驗用 AI 方法來處理問題時，feature 是非常重要的，同一個 train data 如果處理成不同 feature，最後的 performance 甚至可能會差到十幾%，所以 feature 要如何改良是個關鍵

### 1.1 train dataset

下圖為 AIea 提供的 train dataset 之數量分布

| level | number |
|---|---|
| 1 | 11 |
| 2 | 46 |
| 3 | 29 |
| 4 | 28 |
| 5 | 43 |
| 6 | 48 |
| 7 | 51 |
| 8 | 34 |
| 9 | 20 |
| 10 | 9 |
| total | 319 |

總共有 319 筆 sample，共 10 種 level(海廢等級)，各 level 的數量分布不均勻，因此會做 upsample 讓每個 level 的數量都複製增加到 51 筆，所以最後總 train data 數會是 510 筆，這樣可以避免 data imbalance 的問題，不然 model 會傾向猜數量最多的那類

### 1.2 缺失值處理: 以同一個地點的其他 3 個 season 的眾數來填補缺失值

某些 sample 的部分特徵值可能會有缺失值，如下圖所示，一開始是用-1 來取代缺失值，這個做法可能對於 model 的學習來說不太好，而且其實缺失值還蠻多的，佔了約 25%，應該要更妥善處理，所以我們後來有想到一個的解決方案，那就是以同一個地點的其他 3 個 season 的眾數來填補缺失值，因為經過觀察，發現缺失值都是海岸地形種類，從直觀上來想，海岸地形因為會影響人類活動，理論上來說應該也會和海廢等級有關，因此海岸地形這個 feature 應該要用到，所以缺失值勢必要用某個值去填補，不能直接去掉

此外，我們還發現缺失值會發生在每一個地點的第 4 個 season，而且同一個地點的其他 3 個 season 的海岸地形種類通常都很接近，代表第 4 個 season 很有可能也是同一個海岸地形，這其實蠻合理的，畢竟同一個地點的海岸地形通常不會因為 season 的不同而變化，不過同一個地點在不同 season 時，某些海岸地形會略有不同，所以我們最後決定以同一個地點的其他 3 個 season 的眾數(多數決的方式)來填補缺失值

| Station | Season | County | Location | Lat | Lon | 縣市 | 海岸段 | Region | Seat | Shore shap | Substrate ty | 1暴露岩岸 | 2暴露人造 | 3暴露岩盤 | 4沙灘 | 5砂礫混合 | 6礫石灘 | 7開闊潮間 | 8遮蔽岩岸 | 9遮蔽潮間 | 10遮蔽潟 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E02 | 1 | 宜蘭縣 | 大溪 | 24.92528 | 121.8857 | 16 | 5 | 1 | 4 | 2 | 3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| E02 | 2 | 宜蘭縣 | 大溪 | 24.92528 | 121.8857 | 16 | 5 | 1 | 4 | 2 | 3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| E02 | 3 | 宜蘭縣 | 大溪 | 24.92528 | 121.8857 | 16 | 5 | 1 | 4 | 2 | 3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| E02 | 4 | 宜蘭縣 | 大溪 | 24.92528 | 121.8857 | 16 | 5 | 1 | 4 | 2 | 3 | | | | | | | | | | |
| E03 | 1 | 宜蘭縣 | 頭城 | 24.8573 | 121.8334 | 16 | 5 | 1 | 4 | 1 | 4 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| E03 | 2 | 宜蘭縣 | 頭城 | 24.8573 | 121.8334 | 16 | 5 | 1 | 4 | 1 | 4 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| E03 | 3 | 宜蘭縣 | 頭城 | 24.8573 | 121.8334 | 16 | 5 | 1 | 4 | 1 | 4 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| E03 | 4 | 宜蘭縣 | 頭城 | 24.8573 | 121.8334 | 16 | 5 | 1 | 4 | 1 | 4 | | | | | | | | | | |

### 1.3 特徵值若為中文或英文就以正整數代號取代

station、location 和 country 這三個 feature 的特徵值為中文或英文，這會讓 model 無法學習，所以我們將其改成從 1 開始的正整數代號，如下圖所示

| Station | Season | County | Location |
|---|---|---|---|
| E02 | 1 | 宜蘭縣 | 大溪 |
| E02 | 2 | 宜蘭縣 | 大溪 |
| E02 | 3 | 宜蘭縣 | 大溪 |
| E02 | 4 | 宜蘭縣 | 大溪 |
| E03 | 1 | 宜蘭縣 | 頭城 |
| E03 | 2 | 宜蘭縣 | 頭城 |
| E03 | 3 | 宜蘭縣 | 頭城 |
| E03 | 4 | 宜蘭縣 | 頭城 |
| E05 | 1 | 宜蘭縣 | 清水港尾 |
| E05 | 2 | 宜蘭縣 | 清水港尾 |
| E05 | 3 | 宜蘭縣 | 清水港尾 |
| E05 | 4 | 宜蘭縣 | 清水港尾 |
| E06 | 1 | 宜蘭縣 | 無尾港 |

➔

| Station | Season | County | Location |
|---|---|---|---|
| 0 | 1 | 0 | 0 |
| 0 | 2 | 0 | 0 |
| 0 | 3 | 0 | 0 |
| 0 | 4 | 0 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 2 | 0 | 1 |
| 1 | 3 | 0 | 1 |
| 1 | 4 | 0 | 1 |
| 2 | 1 | 0 | 2 |
| 2 | 2 | 0 | 2 |
| 2 | 3 | 0 | 2 |
| 2 | 4 | 0 | 2 |
| 3 | 1 | 0 | 3 |

## 1.4 將是否為同一個系列測站的資訊納入考慮

如果一個 station 名稱就用一個數字代號的話，那就跟 location feature 一模一樣了，因此我們後來改成英文代號相同的 station 就視為同一個 station 並使用同一個數字代號，數字代號一樣從 1 開始，這樣就可以將是否為同一個系列測站的資訊納入考慮

舉例來說，E01 和 E02 都視為 E 系列的 station，所以他們的代號都是 1

## 1.5 將 train data 和 test data 還原成 raw data

我們發現 train data 和 test data 是由 raw data 拆分而成，所以其實 train data 和 test data 中的 sample 是有順序關係的，可以根據 station 將 test data 插入到 train data 的對應位置，即可還原成 raw data

舉例來說，test data 中的 E04 station 的 4 個 sample 可以插入到 train data 的 E03 和 E05 之間，這樣就可以考慮到鄰近關係，轉換出來的數字才會比較合理，如下圖所示

| | | |
|---|---|---|
| E03 | 1 | 宜蘭縣 |
| E03 | 2 | 宜蘭縣 |
| E03 | 3 | 宜蘭縣 |
| E03 | 4 | 宜蘭縣 |
| E05 | 1 | 宜蘭縣 |
| E05 | 2 | 宜蘭縣 |
| E05 | 3 | 宜蘭縣 |
| E05 | 4 | 宜蘭縣 |

**train data**

+

| | | |
|---|---|---|
| E04 | 1 | 宜蘭縣 |
| E04 | 2 | 宜蘭縣 |
| E04 | 3 | 宜蘭縣 |
| E04 | 4 | 宜蘭縣 |

**test data**

=

| | | |
|---|---|---|
| E03 | 1 | 宜蘭縣 |
| E03 | 2 | 宜蘭縣 |
| E03 | 3 | 宜蘭縣 |
| E03 | 4 | 宜蘭縣 |
| E04 | 1 | 宜蘭縣 |
| E04 | 2 | 宜蘭縣 |
| E04 | 3 | 宜蘭縣 |
| E04 | 4 | 宜蘭縣 |
| E05 | 1 | 宜蘭縣 |
| E05 | 2 | 宜蘭縣 |
| E05 | 3 | 宜蘭縣 |
| E05 | 4 | 宜蘭縣 |

**raw data**

## 1.6 處理同一個地點，但是 County 或 Location 不一樣的情況

還原成 raw data 後會發現 data 中的 County 和 Location 很常出現誤植的情況，像是下圖的例子，明明是同一個地點，但是 County 或 Location 卻不一樣，這會造成後續轉換成代號時，model 會將其視為不同的地點，就有可能會出問題，所以我將其都轉成同一個中文後才轉成數字代號(中文相同數字就相同)

| | | | | | | |
|---|---|---|---|---|---|---|
| 317 | TT03 | 1 | 台東縣長濱鄉 | 白桑安 | 23.24933 | 121.4187 |
| 318 | TT03 | 2 | 台東縣長濱鄉 | 白桑安/長濱觀景平台 | 23.24933 | 121.4187 |
| 319 | TT03 | 3 | 台東縣長濱鄉 | 白桑安/長濱觀景平台 | 23.24933 | 121.4187 |
| 320 | TT03 | 4 | 台東縣長濱鄉 | 寧埔 | 23.24933 | 121.4187 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 30 | HL03 | 1 | 花蓮縣 | 崇德 | 24.16697 | 121.6586 |
| 31 | HL03 | 2 | 花蓮縣 | 崇德 | 24.16697 | 121.6586 |
| 32 | HL03 | 3 | 花蓮縣 | 崇德 | 24.16697 | 121.6586 |
| 33 | HL03 | 4 | 花蓮 | 崇德 | 24.16697 | 121.6586 |

### 1.7 將所有 feature 做個別的 Min-Max Normalization

因為不同 feature 的數值分布範圍差異甚大，某些 feature 只有 0 或 1，某些 feature 從 0 到 87，如果不做 Normalization 的話，某些 feature 值會 dominate 整個 model 的學習，導致 model 學不起來

原本是用 z-score standardization，但是 z-score standardization 會將分布會強行轉成標準高斯分布，原本的分布可能會跑掉，所以當原本的分布很不對稱時會出問題，因此後來改用 Min-Max Normalization，公式如下圖所示，每個 feature 都會按比例轉換成 0~1 之間的數

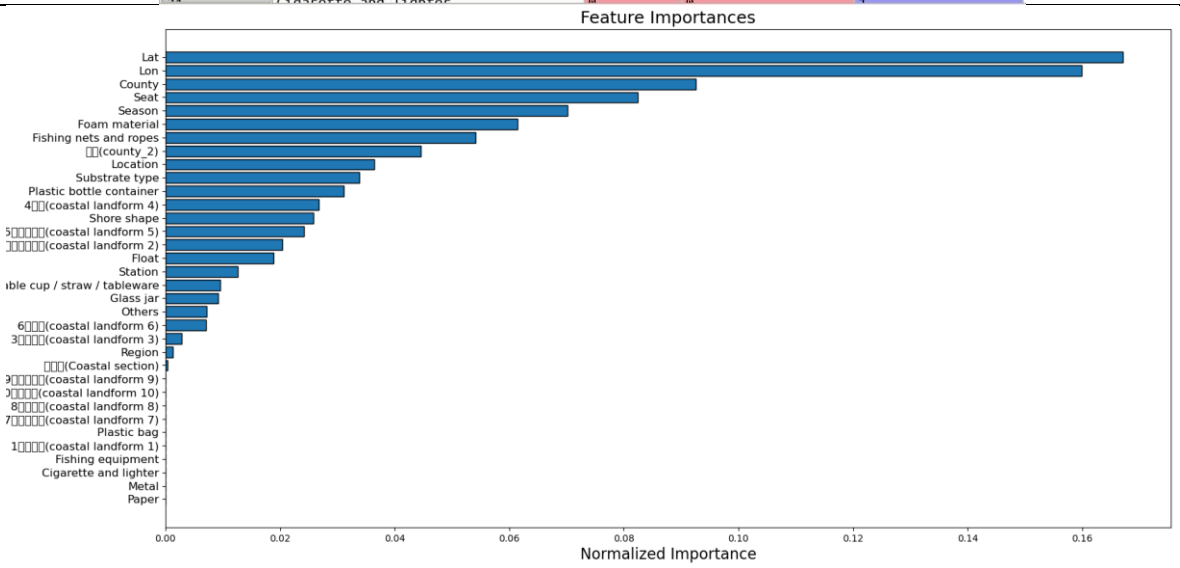
最小值最大值正規化的用意，是將資料等比例縮放到 [0, 1] 區間中，可利用下列公式進行轉換：

$$X_{nom} = \frac{X - X_{min}}{X_{max} - X_{min}} \in [0,1]$$

### 1.8 做 feature selection 找出關鍵 feature

- 海廢等級代表海岸的汙染程度，海廢等級越高就代表海岸越髒，因此海廢等級和垃圾量有關，而垃圾量和人類活動有關，而人類活動可能和地點、季節、地形(ex: 沙灘)、垃圾類型這些 feature 有關，所以我們可以去求這些 feature 和海廢等級的關係，像是 feature importance、correlation 等，藉此找出關鍵 feature
- 我們使用一個名叫 feature-selector 的 package 當作特徵選擇工具，feature-selector 是由 Feature Labs 的一名數據科學家 williamkoehrsen 在 2017 年寫的，在 github 有 1900 多個 star
  - feature-selector 會用 train data 訓練出一個 Gradient Boosting machine(GBM)，然後由 GBM 得到每一個 feature 的重要性分數，再對所有特徵的重要性分數做 normalize
  - 為了使計算得到的 feature 重要性分數具有很小的方差，GBM 會訓練 10 次，取多次訓練的平均值來當作最終的 feature importance
  - 同時為了防止過擬合，會從 train data 中選一部份 data 當作 validation data，在訓練 GBM 的時候，計算 GBM 在 validation data 上的 accuracy，當 accuracy 達一定 epoch 都沒有上升後，就停止訓練 GBM
- 經過計算後，各 feature 的 feature importance 如下圖所示

| Index | feature | importance | normalized_importance | cumulative_importance |
|---|---|---|---|---|
| 0 | Lat | 586 | 0.167118 | 0.167118 |
| 1 | Lon | 560.6 | 0.159875 | 0.326993 |
| 2 | County | 324.5 | 0.0925424 | 0.419535 |
| 3 | Seat | 289.2 | 0.0824754 | 0.502011 |
| 4 | Season | 246 | 0.0701554 | 0.572166 |
| 5 | Foam material | 215.6 | 0.0614858 | 0.633652 |
| 6 | Fishing nets and ropes | 189.8 | 0.054128 | 0.68778 |
| 7 | 縣市(county_2) | 156.5 | 0.0446314 | 0.732411 |
| 8 | Location | 127.8 | 0.0364466 | 0.768858 |
| 9 | Substrate type | 118.5 | 0.0337944 | 0.802652 |
| 10 | Plastic bottle container | 109 | 0.0310851 | 0.833737 |
| 11 | 4沙灘(coastal landform 4) | 94 | 0.0268074 | 0.860545 |
| 12 | Shore shape | 90.5 | 0.0258092 | 0.886354 |
| 13 | 5砂礫混合灘(coastal landform 5) | 84.8 | 0.0241837 | 0.910538 |
| 14 | 2暴露人造結構物(coastal landform 2) | 71.6 | 0.0204192 | 0.930957 |
| 15 | Float | 66 | 0.0188222 | 0.949779 |
| 16 | Station | 44.2 | 0.0126052 | 0.962384 |
| 17 | Disposable cup / straw / tableware | 33.5 | 0.00955369 | 0.971938 |
| 18 | Glass jar | 32.2 | 0.00918295 | 0.981121 |
| 19 | Others | 25.3 | 0.00721517 | 0.988336 |
| 20 | 6礫石灘(coastal landform 6) | 24.7 | 0.00704406 | 0.99538 |
| 21 | 3暴露岩盤(coastal landform 3) | 10.1 | 0.00288037 | 0.99826 |
| 22 | Region | 4.7 | 0.00134037 | 0.999601 |
| 23 | 海岸段(Coastal section) | 1.4 | 0.000399259 | 1 |
| 24 | 9遮蔽潮間帶(coastal landform 9) | 0 | 0 | 1 |
| 25 | 10遮蔽濕地(coastal landform 10) | 0 | 0 | 1 |
| 26 | 8遮蔽岩岸(coastal landform 8) | 0 | 0 | 1 |
| 27 | 7開闊潮間帶(coastal landform 7) | 0 | 0 | 1 |
| 28 | Plastic bag | 0 | 0 | 1 |
| 29 | 1暴露岩岸(coastal landform 1) | 0 | 0 | 1 |
| 30 | Fishing equipment | 0 | 0 | 1 |
| 31 | Cigarette and lighter | 0 | 0 | 1 |

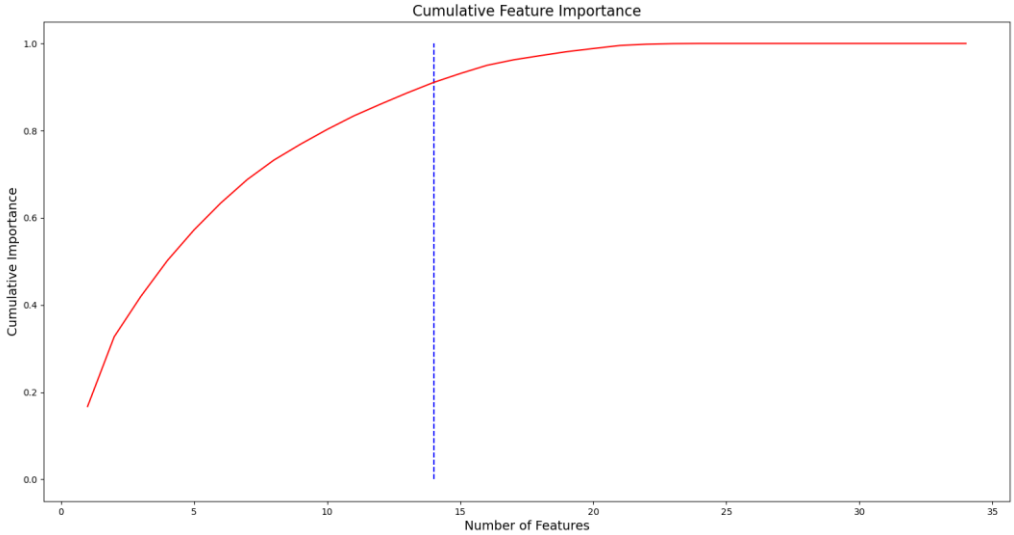



- 因為 train data 只有 319 筆，但是 feature 卻有 34 種，這樣很容易會產生過擬合的情況，必須適度地將一些多餘的 feature 去除，因此我們會根據得到的 feature importance 和 correlation 將以下三種 feature 去掉
  - zero importance feature

- 對模型預測結果毫無貢獻的 feature
- 下圖為 zero importance feature

| nde: | Type | Size | |
|---|---|---|---|
| 0 | str | 1 | 9遮蔽潮間帶(coastal landform 9) |
| 1 | str | 1 | 10遮蔽濕地(coastal landform 10) |
| 2 | str | 1 | 8遮蔽岩岸(coastal landform 8) |
| 3 | str | 1 | 7開闊潮間帶(coastal landform 7) |
| 4 | str | 1 | Plastic bag |
| 5 | str | 1 | 1暴露岩岸(coastal landform 1) |
| 6 | str | 1 | Fishing equipment |
| 7 | str | 1 | Cigarette and lighter |
| 8 | str | 1 | Metal |
| 9 | str | 1 | Paper |

- low importance feature
  - 先將 feature importance 由高排到低，依次將 feature importance 做疊加，當累積和達到 0.9 後剩下的 feature 就視為 low importance feature 去掉，所以 zero importance feature 也必定包含於 low importance feature 中，這有點類似 PCA 中留下主要分量去除不重要分量的概念
  - 下圖中的藍色虛線和紅色實線的交點所對應的 cumulative feature importance 為 0.9，所以只會留下藍色虛線以左的 feature




  - 下圖為 low importance feature

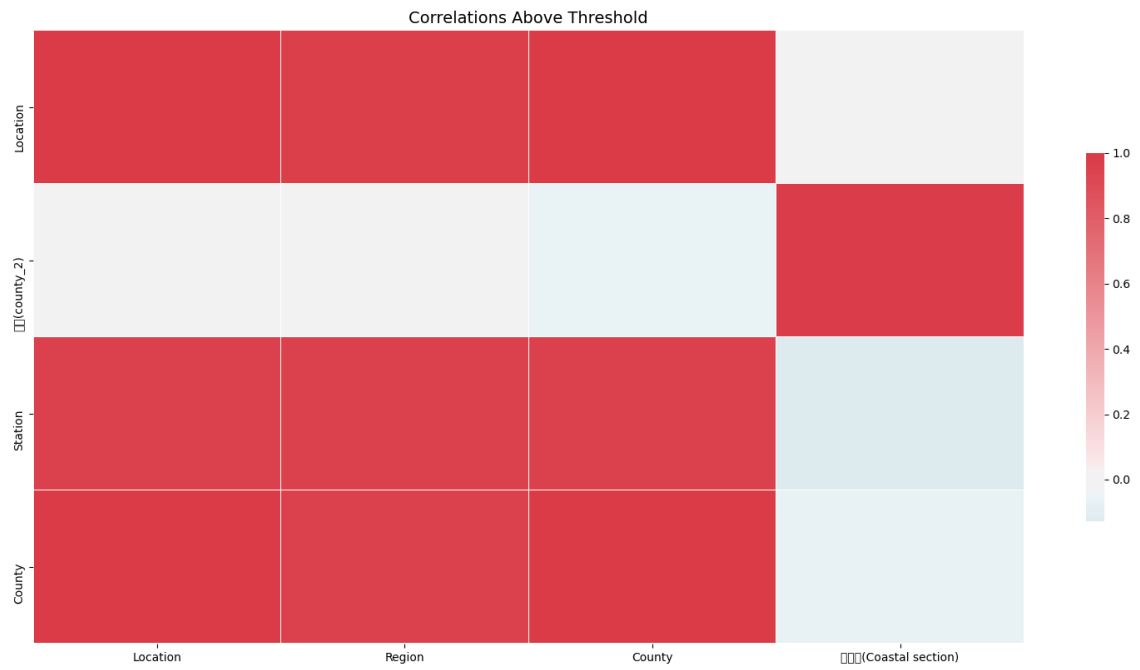
| nde | Type | Size | |
|---|---|---|---|
| 0 | str | 1 | 5砂礫混合灘(coastal landform 5) |
| 1 | str | 1 | 2暴露人造結構物(coastal landform 2) |
| 2 | str | 1 | Float |
| 3 | str | 1 | Station |
| 4 | str | 1 | Disposable cup / straw / tableware |
| 5 | str | 1 | Glass jar |
| 6 | str | 1 | Others |
| 7 | str | 1 | 6礫石灘(coastal landform 6) |
| 8 | str | 1 | 3暴露岩盤(coastal landform 3) |
| 9 | str | 1 | Region |
| 10 | str | 1 | Cigarette and lighter |
| 11 | str | 1 | Fishing equipment |
| 12 | str | 1 | 1暴露岩岸(coastal landform 1) |
| 13 | str | 1 | Plastic bag |
| 14 | str | 1 | 10遮蔽濕地(coastal landform 10) |
| 15 | str | 1 | 8遮蔽岩岸(coastal landform 8) |
| 16 | str | 1 | Metal |
| 17 | str | 1 | 9遮蔽潮間帶(coastal landform 9) |
| 18 | str | 1 | 海岸段(Coastal section) |
| 19 | str | 1 | 7開闊潮間帶(coastal landform 7) |
| 20 | str | 1 | Paper |

- high correlation feature
  - 若兩個 feature 的相關係數超過 0.95，就將 feature importance 比較低的 feature 去掉，因為相關性超過 0.95 就代表這兩個 feature 非常接近，其實只需要一個當代表即可，不然可能會導致過擬合
  - 這邊的相關係數為 Pearson correlation coefficient，公式如下圖所示，其值代表兩個 feature 之間的線性相關(相依)程度，其值介於-1 與 1 之間，越接近 1 或-1 就越相關

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}$$

  - 下圖為 high correlation feature 和其對應的 correlation coefficient

| Index | drop_feature | corr_feature | corr_value |
|---|---|---|---|
| 0 | County | Station | 0.958241 |
| 1 | Location | Station | 0.959144 |
| 2 | Location | County | 0.996055 |
| 3 | 海岸段(Coastal section) | 縣市(county… | 0.976704 |
| 4 | Region | Station | 0.957603 |

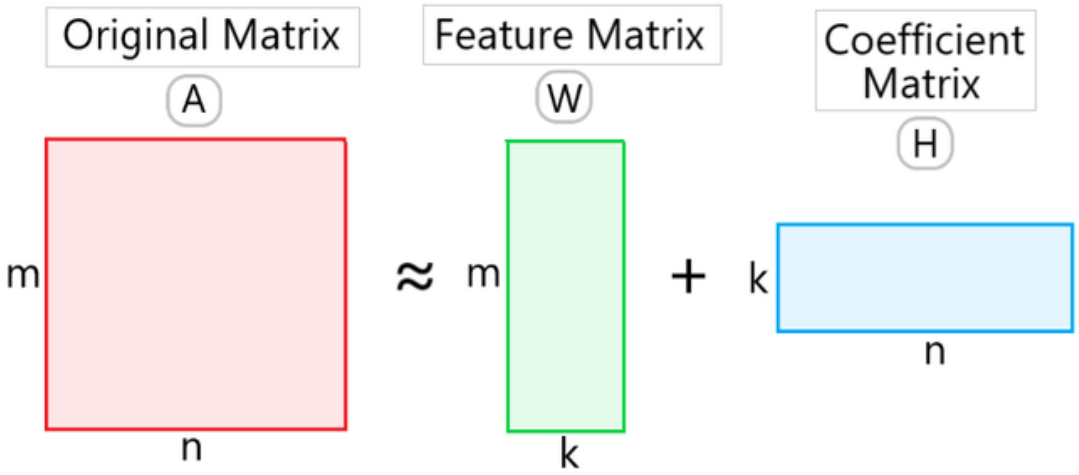



| nde: | Type | Size | |
|---|---|---|---|
| 0 | str | 1 | Station |
| 1 | str | 1 | Location |
| 2 | str | 1 | 海岸段(Coastal section) |
| 3 | str | 1 | Region |

### 1.9 Non-Negative Matrix Factorization

由於此資料庫含有大量 binary 的特徵(0/1)，從"曝露岩岸"到"Others"一共 22 維的特徵皆屬此類。這些特徵形成的矩陣又多為稀疏矩陣，也就是說裡面只有少部份為 1 而大部分為 0，如果拿此特徵進去模型學習會造成模型學習困難，因此我們將採用在機器學習中常拿被量化稀疏矩陣的方法學 Non-Negative Matrix Factorization 對此類特徵進行量化。以下為此演算法的公式與示意圖：

$$
\begin{aligned}
&0.5 * ||X - WH||^2_{loss} \\
&+alpha\_W * l1_{ratio} * n\_features * ||vec(W)||_1 \\
&+alpha\_H * l1_{ratio} * n\_samples * ||vec(H)||_1 \\
&+0.5 * alpha\_W * (1 - l1_{ratio}) * n\_features * ||W||^2_{Fro} \\
&+0.5 * alpha\_H * (1 - l1_{ratio}) * n\_samples * ||H||^2_{Fro}
\end{aligned}
$$




W 為我們最後想要求的矩陣，而我們可以決定此 W 的維度有多少，藉以達到降維和從稀疏矩陣中提取重要特性的目的。在實作上，我們採用了 sklearn 的工具包幫助我們，並依序用 2,4,8 三個不同的 W 維度來進行稀疏矩陣的量化。我們會先把 34 維中後 22 維的矩陣分別變成 2,4,8 維(最終變成 14,16,20 維)，再套用 1.8 的方法來進行重要特徵的篩選，而我們最後給模型的維度=8 維。

## 2 模型架構

我們的 model 主要分成 2 大類，一類是純機器學習方法，像是 Random Forest，另一類是加入 neural network 概念的深度學習方法，像是 encoder + Random Forest、DNN + Random Forest、Random Forest + DNN，下面會做細部說明

### 2.1 隨機森林模型(Random Forest)(RF)

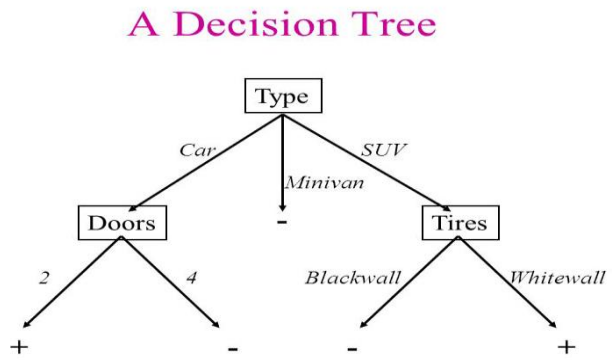
#### 2.1.1 目的

random forest classifier 的時間複雜度較低，為 O(n*log(n)*d*k)，也更適合處理高維度的資料。在分析上能顯示各個特徵的重要性，還能利用 oob error 快速評估模型的表現，非常適合用於本次的實驗

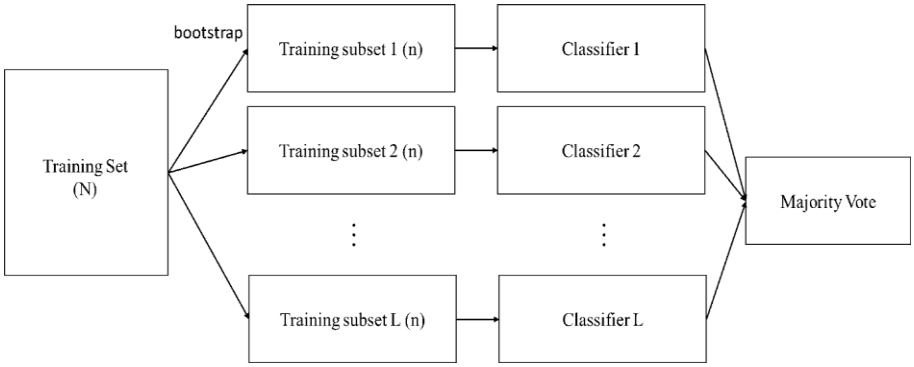
#### 2.1.2 原理

在講 Random forest 之前，必須要先介紹 Decision tree 和 Bagging 這兩個概念

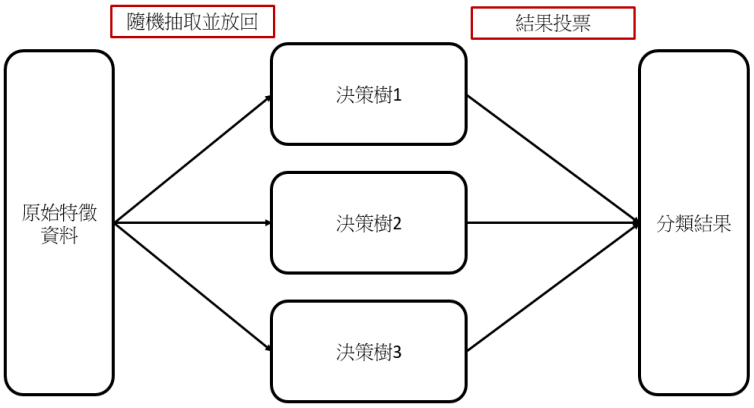
Decision tree：透過一層一層的決策，逐步篩選出符合的結果，如下圖所示



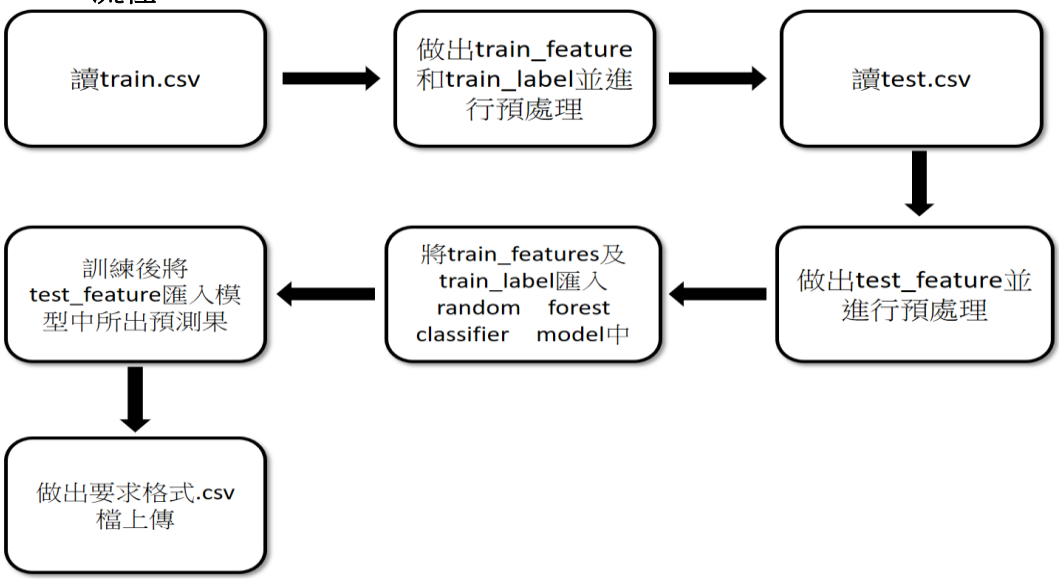
Bagging：從訓練資料中隨機抽取，取出後放回(bootstrap)，利用抽出之樣本訓練多個分類器，每個分類器的權重一致，最後用投票方式(Majority vote)得到最終結果，如下圖所示




Random forest：是一個結合 bagging 及 decision trees 的演算法，如其名字所述，由多個 decision tree 所組成，每顆決策樹獨立運算出結果，並透過投票得到最後的分類結果，如下圖所示




### 2.1.3 流程




### 2.1.4 優缺點

**優點**

1. 時間複雜度低，適合處理大量高維度的資料
2. 附有 feature importance 以及 oob error 等有利分析的功能
3. 訓練速度快
4. 能夠平衡失衡資料集的誤差
5. 對於缺失值以及離群值的敏感度低
6. 能夠避免 overfitting
7. 能解決回歸與分類兩種問題

**缺點**

1. 對於資料數量少，或是低維度的資料，分類結果較差
2. 在某些 noise 較大的分類或迴歸問題上會過擬合
3. 相對於 Decision tree，需要更長的時間以及更多的儲存空間作運算

## 2.2 深度神經網路(Deep Neural Network)(DNN)

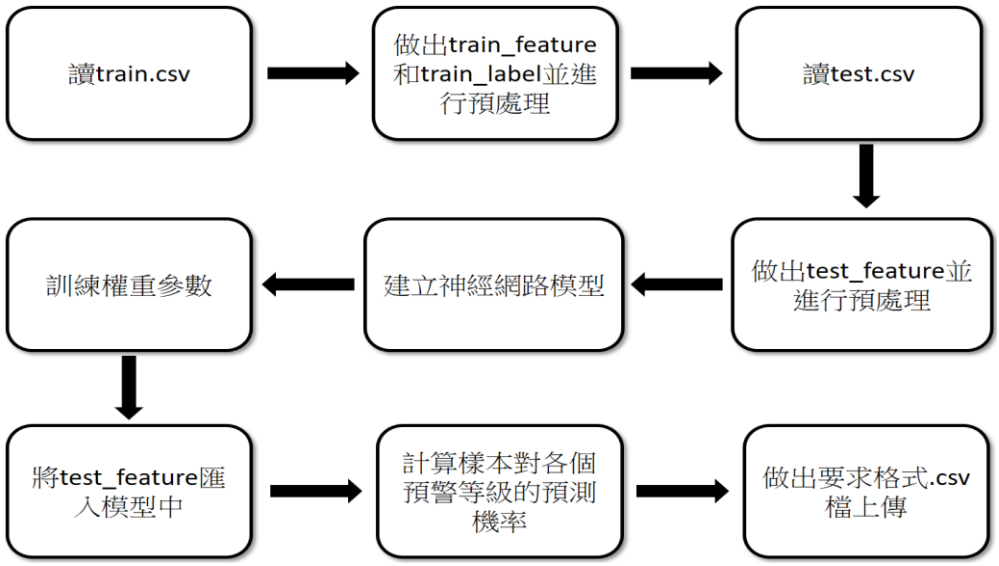
### 2.2.1 目的

在機器學習的世界中，神經網路就像是人類的大腦神經結構，而神經元就像是大腦的神經細胞，是神經網路中最基礎的結構，在它們互相結合下，可以建構龐大的運作網路，能大大的提高預測準確率，也非常適合處理分類問題，故也選用此方法

### 2.2.2 原理

1. 建立輸入層、多層隱藏層、輸出層和每個層的神經元，每個神經元都有一個輸出，其輸出稱為激勵值(介於 0~1)，每一層神經元中激勵值的操作結果會影響下一層的激勵值，一層一層之間激勵值的傳遞最後即可得到輸出判斷結果
2. 前一層的每個神經元的輸出分別乘其對應的「權重值」最後相加 減下一層神經元的「偏置值」帶入激勵函式將其轉為介於 0~1 的「激勵值」
3. 數學形式:
   $0 \leq$ 激勵值 $= f(a1w1+a2w2+\ldots\ldots+anwn - b) \leq 1$

### 2.2.3 流程



### 2.2.4 優缺點

**優點**

1. 可以建構非線性的模型
2. 有良好的推廣性，對於未知的輸入亦可得到正確的輸出
3. 可以接受不同種類的變數作為輸入，適應性強
4. 可應用的領域相當廣泛

**缺點**

1. 以迭代方式更新鍵結值與閥值，計算量大，相當耗費電腦資源

訓練的過程中無法得知需要多少神經元個數，太多或太少的神經元均會影響系統的準確性，因此往往需以試誤的方式得到適當的神經元個數

### 2.3 Encoder + Random Forest

我們訓練了一個 autoencoder 來去對原始的 feature 做 representation。我們用了簡單用了一層 FC 當 encoder，和一層 FC 當 decoder，並採用 gelu 當作 activation function。loss 的部分則是使用 mean square error 當作 Loss function，最佳化的方法則是採用了 adam。實作上，我們使用了三組不同的模型，將維度從原本的 8 維分別升到 64、128、256 維，當作最終分類器的輸入。

### 2.4 DNN + Random Forest

- 接 4 層的 DNN(最後一層是 output layer)來 train  test 在 train data 和 test data 上  取倒數第二層 dense layer 的 feature embedding 當作新的 train data 和 test data  用 random forest 來 train 和 test
- 除 output layer 以外，神經元數都是 600
- 新的 train data 和 test data 的 feature dimension 為 600
- 架構如下圖所示

```
model = Sequential()
model.add(Dense(units = 600, input_dim = 8, kernel_initializer = 'normal', activation='tanh'))
#model.add(Dropout(0.1))
model.add(Dense(units = 600, kernel_initializer = 'normal', activation='tanh'))
#model.add(Dropout(0.1))
model.add(Dense(units = 600, kernel_initializer = 'normal', activation='tanh'))
#model.add(Dropout(0.1))
model.add(Dense(units=10, kernel_initializer = 'normal', activation='softmax'))
```

### 2.5 Random Forest + DNN

- 與 2.4 相反，將 Random Forest 產出的 predicted probability(10 維)當作 DNN 的 features 做訓練，而 DNN 的 output 也是 10 維的結果，此作法用意在於透過 DNN 強化 random forest 的結果，看能否在 testing data 上有更好的表現。
- DNN 架構為 4 層(包含 input layer 與 output layer)，各層神經元個數分別為 (10, 64, 32, 16, 10)，層跟層之間皆有做 batch normalization，activate function 皆為 GELU，Dropout rate 設定為 0.2。

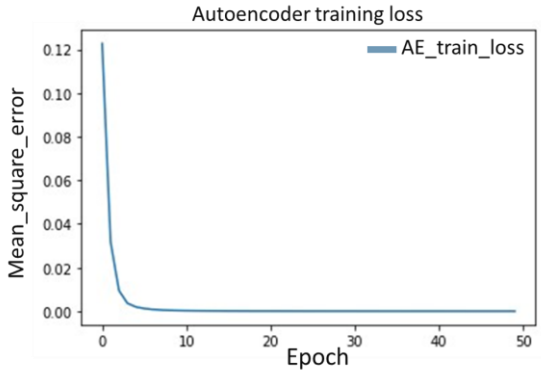
# -結果與分析

- 下面的 test kappa 皆為 AIdea public leaderboard 的評估結果

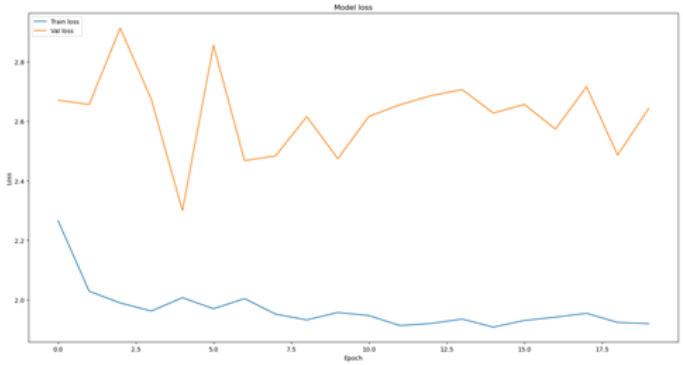
| RandomForest | | | | |
|---|---|---|---|---|
| Feature Name | Train Kappa | Train UAR | Train ACC | Test Kappa |
| no_remove | 0.876 | 91.5 % | 89.03 % | 0.6523 |
| remove_low_importance_and_high_correlation (RLIHC) | 0.8618 | 90.4 % | 87.77 % | 0.6753 |
| remove_low_importance | 0.7235 | 78.94 % | 75.55 % | 0.7059 |
| remove_zero_importance | 0.6122 | 70.55 % | 65.52 % | 0.754 |
| Non-Negative Matrix Factorization_RandomForest | | | | |
| FA2, no_remove | 0.4459 | 57.14 % | 50.47 % | 0.6612 |
| FA2, RLIHC | **0.5823** | **69.22 %** | **62.7 %** | **0.7789*** |
| FA4, RLIHC | 0.6626 | 74.96 % | 69.91 % | 0.7678 |
| FA8, RLIHC | 0.609 | 71.34 % | 65.2 % | 0.6505 |
| RLIHC+upsampling | | | | |
| RandomForest, n_iter=45 | 0.6707 | 75.55 % | 70.85 % | 0.6749 |
| DecisionTreeClassifier, n_iter=40 | 0.9109 | 93.65 % | 92.16 % | 0.5305 |
| Gradient Boosting Classifier, n_iter=100 | 0.603 | 70.38 % | 64.89 % | 0.5817 |
| Non-Negative Matrix FA_RandomForest+upsample | | | | |
| FA2, RLIHC | 0.6673 | 75.59 % | 70.53 % | 0.6878 |
| FA4, RLIHC | 0.7089 | 77.62 % | 74.29 % | 0.6651 |
| DNN(RELU) -> RandomForest | | | | |
| FA2, no_remove | 0.8727 | 91.95% | 88.71% | 0.6864 |
| FA2, RLIHC | 0.8063 | 87.82% | 82.76% | 0.7511 |
| FA4, RLIHC | 0.5923 | 69.79% | 63.64% | 0.6972 |
| RandomForest -> DNN(GELU) | | | | |
| FA2, RLIHC | 0.5726 | 67.48 % | 61.96 % | 0.7643 |
| FA2, RLIHC, weighted loss | 0.5361 | 66.57 % | 58.43 % | 0.6896 |
| Autoencoder (GELU) -> RandomForest | | | | |
| FA2, RLIHC, Hidden dim=64 | 0.6246 | 73.46% | 66.46% | 0.5093 |
| FA2, RLIHC, Hidden dim=128 | 0.4703 | 60.15% | 52.66% | 0.5527 |
| FA2, RLIHC, Hidden dim=256 | 0.5746 | 69.05% | 62.07% | 0.5458 |
| FA2, RLIHC, Hidden dim=512 | 0.5706 | 68.58% | 61.76% | 0.5861 |

| no_remove_train/test_feature+upsampling | | | | |
|---|---|---|---|---|
| | Train Kappa | Train UAR | Train ACC | Test Kappa |
| RandomForest, n_iter=45 | 0.693 | 77.07 % | 72.73 % | 0.6885 |
| DecisionTreeClassifier, n_iter=40 | 0.9216 | 94.24 % | 93.1 % | 0.6241 |
| Gradient Boosting Classifier, n_iter=100 | 0.6744 | 75.87 % | 71.16 % | 0.4532 |

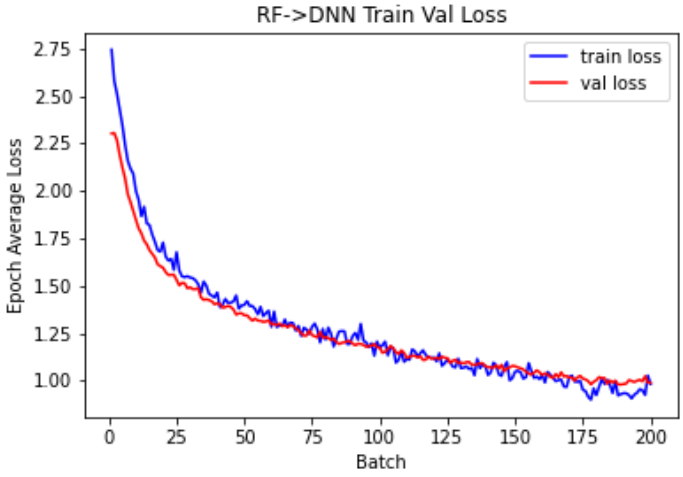
- **non upsampling 和 upsampling 比較**
  觀察 training data 我們發現有嚴重的 data imbalance 問題，我們有嘗試透過 RandomOverSampler 將 training data 的各種類別做隨機上採樣至同樣數量，再當成模型輸入做訓練，而這樣的做法 train kappa 可以逼近 1，但是 test kappa 會很差，表示過擬和於 training data，所以我們後來並沒有採用 upsampling 來開發模型。
- **Feature selection (FS)效益討論：**
  由 Table 可得知，有採用 feature selection 的結果會比較好，其中又以 remove_zero_importance 的表現最為突出，相比完全沒用的 **test kappa 上升 0.1017**。
- **Matrix Factorization (FA)效益討論：**
  Factorization 的結果則是 remove low importance and high correlation (RLIHC) 的平均表現較優秀，其中又以**降到兩維之後再取 FS** 的結果最為突出，比只做 **FS 再上升 0.290**，**test kappa 達 0.7789**。這也是我們最好的結果。
- **傳統機器學習模表現比較：**
  RandomForest (RF)、DecisionTreeClassifier、Gradient Boosting Classifier 三者中，還是以 **RandomForest 的結果最好**，RandomForest 相對 DecisionTreeClassifier 與 Gradient Boosting Classifier 的(Train Kappa-Test Kappa) 較小，表示較沒有 overfitting 的問題，所以我們後來選用 RandomForest 來測試不同 Feature 的表現
- **類神經網路表現討論：**
  - 我們進行了三種 NN 架構對該任務進行不同方式的訓練，並把 training loss 於下方呈現。雖然參數皆有收斂，但表現都沒有優於原始的 FA+FS+RF，由此可以推測此**資料集的數量可能過少**，且**類別過多**，若使用 NN 造成**模型過於複雜**，導致 test kappa 降低。
  - RF + DNN 的 train kappa 值與單純 RF 接近，但是 test kappa 會較差
  - 用 DL 方法後會變得比只有 Random Forest 差，看來 DNN 會把 feature 破壞掉，反而會幫倒忙，可能是因為有效 train data 數只有 319 筆的關係，導致 DL 方法無法發揮
  - Encoder + Random Forest




- DNN + Random Forest



- Random Forest + DNN




# -References

1. PATTERN RECOGNITION AND MACHINE LEARNING
2. PYTHON 機器學習與深度學習特訓班
3. Github
4. [資料分析&機器學習] 第 4.1 講 : Kaggle 競賽-鐵達尼號生存預測-(前 16% 排名)
   https://medium.com/jameslearningnote/%E8%B3%87%E6%96%99%E5%88%86%E6%9E%90-%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92-%E7%AC%AC4-1%E8%AC%9B-kaggle%E7%AB%B6%E8%B3%BD-%E9%90%B5%E9%81%94%E5%B0%BC%E8%99%9F%E7%94%9F%E5

%AD%98%E9%A0%90%E6%B8%AC-%E5%89%8D16-%E6%8E%92%E5%90%8D-a8842fea7077

5. 我如何分析客戶流失預測？Kaggle 比賽思路分享 https://medium.com/finformation%E7%95%B6%E7%A8%8B%E5%BC%8F%E9%81%87%E4%B8%8A%E8%B2%A1%E5%8B%99%E9%87%91%E8%9E%8D/%E6%88%91%E5%A6%82%E4%BD%95%E5%88%86%E6%9E%90%E5%AE%A2%E6%88%B6%E6%B5%81%E5%A4%B1%E9%A0%90%E6%B8%AC-kaggle%E6%AF%94%E8%B3%BD%E6%80%9D%E8%B7%AF%E5%88%86%E4%BA%AB-daecd888a91
6. [資料分析&機器學習] 第 3.4 講：支援向量機(Support Vector Machine)介紹 https://medium.com/jameslearningnote/%E8%B3%87%E6%96%99%E5%88%86%E6%9E%90-%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92-%E7%AC%AC3-4%E8%AC%9B-%E6%94%AF%E6%8F%B4%E5%90%91%E9%87%8F%E6%A9%9F-support-vector-machine-%E4%BB%8B%E7%B4%B9-9c6c6925856b
7. [機器學習 ML NOTES]Kaggle 比賽心得(2%經歷) https://medium.com/@super135799/%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92-ml-notes-kaggle%E6%AF%94%E8%B3%BD%E5%BF%83%E5%BE%97-2-%E7%B6%93%E6%AD%B7-7e8667cf1dc6
8. Feature Engineering 特徵工程中常見的方法 https://vinta.ws/code/feature-engineering.html
9. Basic feature analysis (Date+Categorical+Revenue) https://www.kaggle.com/super13579/basic-feature-analysis-date-categorical-revenue
10. R 筆記–Ensemble Learning(集成學習) https://rpubs.com/skydome20/R-Note16-Ensemble_Learning
11. 機器學習模型的時間複雜度 https://kknews.cc/zh-tw/code/zyv254a.html
12. 機器學習: Ensemble learning 之 Bagging、Boosting 和 AdaBoost https://medium.com/@chih.sheng.huang821/%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92-ensemble-learning%E4%B9%8Bbagging-boosting%E5%92%8Cadaboost-af031229ebc3
13. 隨機森林 RF 算法的原理（一） https://www.twblogs.net/a/5c8a02b2bd9eee35cd6a97fc
14. 隨機森林(RANDOM FOREST)的底層概念、操作細節，與推薦相關資源 http://notebookpage1005.blogspot.com/2018/03/random-forest.html
15. 隨機森林（Random forest,RF）的生成方法以及優缺點 https://www.itread01.com/content/1547100921.html
16. feature selector 官方 code https://github.com/WillKoehrsen/feature-selector